# Inverse problems in networks

Bruno Kauffmann

UPMC PARISUNIVERSITAS

**THÈSE DE DOCTORAT DE**
**L'UNIVERSITÉ PIERRE ET MARIE CURIE**

Spécialité : `Informatique`
(Ecole doctorale : Informatique, télécommunications et électronique (ED 130) )

présentée par
**Bruno KAUFFMANN**

pour obtenir le grade de
**Docteur de l'université Pierre et Marie Curie**

---

# Problèmes inverses dans les réseaux

# Inverse problems in Networks

---

Rapporteurs :
Prof. Michel MANDJES
Prof. Mathew ROUGHAN

soutenue le 24/03/2011, devant le jury composé de :

| | |
|---|---|
| Prof. François BACCELLI | Directeur de thèse |
| Prof. Jean BOLOT | Examinateur |
| Prof. Serge FDIDA | Examinateur |
| Prof. Paulo GON CALVES | Examinateur |
| Prof. Michel MANDJES | Examinateur |
| Prof. Darryl VEITCH | Examinateur |

# Acknowledgement / Remerciements

I would like to thank François Baccelli, my PhD advisor, for his guidance and support for these years. Despite a heavily-loaded agenda, François always took time to discuss difficult points or help me whenever it was needed. The fact that he allies a great kindness to his scientific knowledge and deep creativity made this experience more joyful. His dedication to research, his curiosity and enthusiasm when we were exploring a new idea and close to find a new result is an example of how thrilling research can be, even after years of practice.

It is my good fortune to have Michel Mandjes and Matthew Roughan who have devoted considerable time and effort in reviewing my thesis. Their comments have been valuable to improve the quality of this document. It was an honour and a motivation to write this dissertation for their examination. Thanks also to Serge Fdida, Paulo Gonçalves and Jean Bolot to accept to be part of the committee of my PhD examination.

The journey that lead me to complete this PhD has been enlightened by some great people I met along the way. Darryl Veitch has been for me an unofficial co-advisor, and I thoroughly enjoyed exchanging ideas with him. I learned from him the attention to details and importance of thoroughness. During discussions, he would quickly point the most challenging points, and propose ideas to tackle these difficulties. It is natural that he is also a member of the committee.

I am particularly thankful to Christophe Diot, Augustin Chaintreau, Konstantina Papagiannaki, Vivek Mhatre and all the colleagues from the former Intel Research lab in Cambridge. They are part of my decision to start a PhD, and their guidance made what was my first experience of research a fruitful experience. Thanks also to Steve Uhlig and the members of the Cambridge University Scout and Guide Club, for making this semester also a rich experience on a personal aspect. The years I spent at INRIA and ENS in the TREC team have been rich in meetings with faculty members. Bartek Blaszczyszyn, Pierre Bremaud, Alexandre Proutiere, Thomas Bonald, Ana Bušić and Anne Bouillard all contributed to the richness of the TREC environment. My thanks to them. Many students, including Giovanna, Yogesh, Justin, Frédéric, Furcy, Ahmed, Émilie, Omid, Florence, Mathieu, Van Minh, Calvin, Tien Viet, François-Xavier, Min Ahn, Nadir, Prasanna and Martha also had a key role in this PhD. The joyful discussions we had, both on mathematical and personal subjects, fed me with interests on many topics.

I am thankful to Orange Labs in general, and to the Traffic and Resources Management team in particular, for welcoming me when my dissertation writing was finished. The knowledge that I had a position after my PhD was a comfort that I really appreciated, and the warm welcome I received there has been refreshing after the brunt work and stress of writing this dissertation. On top of that, they made sure that I have the flexibility and time needed to write the final version of this dissertation and prepare serenely my defense.

D'un point de vue personnel, je tiens à remercier les nombreux amis qui m'ont soutenu de façon constante tout au long de mon doctorat, en particulier le groupe des "rôlistes" de l'École Normale Supérieure (et d'ailleurs). Les "soirées bloquées" qui nous réunissent de

façon quasi-hebdomadaire, autour d'un repas et d'un jeu de rôle, d'un jeu de société ou d'un film, furent autant d'occasions de discuter de n'importe quel sujet, y compris de nos bonheurs et malheurs dans la recherche — la majorité faisait, ou avait fini leur doctorat, chacun dans son domaine.

Mes remerciements vont également à Alice, dont le rôle, bien que non scientifique, aura été crucial pour l'aboutissement de ce doctorat. Sa motivation et son enthousiame communicatifs, sa compréhension de mes frustrations et moments difficiles ont été un facteur discret mais décisif.

Enfin, cette thèse n'aurait pas vu le jour sans ma famille. Un grand merci à mes frères et ma sœur (et leurs familles), dont les visites régulières ont été autant de bols d'air. Un merci encore plus grand à mes parents, qui m'ont transmis depuis l'enfance le goût de savoir et de comprendre. Les valeurs et la confiance en soi que je tiens de leur éducation ont été des outils très précieux au cours de ces années, et leur amour parental un socle sur lequel m'appuyer.

# Abstract

The recent impressive growth of Internet in the last two decades lead to an increased need of techniques to measure its structure and its performance. Network measurement methods can broadly be classified into *passive methods* that rely on data collected at routers, and *active methods* based on observations of actively-injected probe packets. Active measurement, which are the motivation of this dissertation, are attractive to end-users who, under the current Internet architecture, cannot access any measurement data collected at routers.

On another side, network theory has been developed for over one century, and many tools are available to predict the performance of a system, depending on a few key parameters. *Queueing theory* emerges as one particularly fruitful network theory both for telephone services and wired packet-switching networks. In the latter case, queuing theory focuses on the packet-level mechanisms and predicts packet-level statistics. At the flow-level viewpoint, the theory of *bandwidth sharing networks* is a powerful abstraction of any bandwidth allocation scheme, including the implicit bandwidth sharing performed by the Transfer Control Protocol. There has been many works showing how the results stemming from these theories can be applied to real networks, in particular to the Internet, and in which aspects real network behaviour differs from the theoretical prediction.

However, there has been up to now very few works linking this theoretical viewpoint of networks and the practical problem of network measurement. In this dissertation, we aim at building a few bridges between the world of network active probing techniques and the world of network theory. We adopt the approach of *inverse problems*. Inverse problems are best seen in opposition to direct problems. A direct problem predicts the evolution of some specified systems, depending on the initial conditions and some known evolution equation. An inverse problem observes part of the trajectory of the system, and aims at estimating the initial condition or parameters that can lead to such an evolution . Active probing technique inputs are the delay and loss time series of the probes, which are precisely a part of the trajectory of the network. Hence, active probing techniques can be seen as inverse problems for some network theory which could predict correctly the evolution of networks.

In this dissertation, we show how active probing techniques are linked to inverse problems in queueing theory. We specify how the active probing constraint can be added to the inverse problems, what are the observables, and detail the different steps for an inverse problem in queueing theory. We classify these problems in three different categories, depending on their output and their generality, and give some simple examples to illustrate their different properties.

We then investigate in detail one specific inverse problem, where the network behaves as

a Kelly network with $K$ servers in tandem. In this specific case, we are able to compute the explicit distribution of the probe end-to-end delays, depending on the residual capacities on each server and the probing intensity. We show that the set of residual capacities can be inferred from the mean end-to-end probe delay for $K$ different probe intensities. We provide an alternative inversion technique, based on the distribution of the probe delays for a single probing intensity. In the case of two servers, we give an explicit characterization of the maximum likelihood estimator of the residual capacities. In the general case, we use the Expectation-Maximization algorithm (E-M). We prove that in the case of two servers, the estimation of E-M converges to a finite limit, which is a solution of the likelihood equation. We provide an explicit formula for the computation of the iteration step when $K = 2$ or $K = 3$, and show that the formula stays tractable for any number of servers. We evaluate these techniques numerically. Based on simulations fed with real network traces, we study independently the impact of the assumptions of a Kelly network on the performance of the estimator, and provide simple correction factors when they are needed.

We also extend the previous example to the case of a tree-shaped network. The probes are multicast, originated from the root and destined to the leaves. They experience an exponentially distributed waiting time at each node. We show how this model is related to the model of a tree-shaped Kelly network with unicast cross-traffic and multicast probes, and provide an explicit formula for the likelihood of the joint delays. We use the E-M algorithm to compute the maximum likelihood estimators of the mean delay in each node, and derive explicit solutions for the combined E and M steps. Numerical simulations illustrate the convergence properties of the estimator. As E-M is slow in this case, we provide a technique for convergence acceleration of the algorithm, allowing much larger trees to be considered as would otherwise be the case. This technique has some novel features and may be of broader interest.

Finally, we explore the case of inverse problems in the theory of bandwidth sharing networks. Using two simple examples of networks, we show how a prober can measure the network by varying the number of probing flows and measure the associated bandwidth allocated to each probing flow. In particular, when the bandwidth allocation maximizes an $\alpha$-fair utility function, the set of server capacities and their associated flow numbers can be uniquely identified in most cases. We provide an explicit algorithm for this inversion, with some cases illustrating the numerical properties of the technique.

**Keywords:** inverse problems — Internet tomography — active probing measurement — statistics — queueing theory —- Expectation-Maximization algorithm

# Résumé

La croissance récente d'Internet lors deux dernières décennies a conduit à un besoin croissant de techniques permettant de mesurer la structure et la performance d'Internet. Les techniques de mesures de réseaux peuvent être classifiées en *méthodes passives* qui utilisent des données collectées au niveau des routeurs, et les *méthodes actives*, reposant sur l'injection active et l'observation de paquets-sondes. Les méthodes actives, qui sont la motivation principale de ce doctorat, sont particulièrement adaptées aux utilisateurs finaux, qui ne peuvent pas accéder aux données mesurées par les routeurs avec l'architecture actuelle d'Internet.

Sur un autre plan, la théorie des réseaux se développe depuis un siècle, et de nombreux outils permettent de prédire la performance d'un système, en fonction de quelques paramètres clés. La *théorie des files d'attentes* émerge comme une solution particulièrement fructueuse, que ce soit pour les réseaux téléphoniques ou pour les réseaux filaires à commutation de paquet. Dans ce dernier cas, elle s'intéresse au mécanisme à l'échelle des paquets, et prédit des statistiques à ce niveau. À l'échelle des flots de paquets, la théorie des *réseaux à partage de bande passante* permet une abstraction de tout schéma d'allocation de bande passante, y compris le partage implicite résultant du protocole TCP. De nombreux travaux ont montré comment les résultats provenant de ces théories peuvent s'appliquer aux réseaux réels, et en particulier à Internet, et dans quels aspects le comportement de réseaux réels diffère des prédictions théoriques.

Cependant, il y a eu peu de travaux établissant des liens entre le point de vue théorique d'un réseau et le problème pratique consistant à le mesurer. Le but de ce manuscrit est de bâtir quelques ponts entre le monde des méthodes de mesure par sondes actives et le monde de la théorie des réseaux. Nous adoptons l'approche des *problèmes inverses*, qui peuvent être vus en opposition aux problèmes directs. Un problème direct prédit l'évolution d'un système défini, en fonction des conditions initiales et d'une équation d'évolution connue. Un problème inverse observe une partie de la trajectoire d'un système défini, et cherche à estimer les conditions initiales ou paramètres pouvant conduire à cette trajectoire. Les données des méthodes de mesure par sondes actives sont les séries temporelles des pertes et délais des sondes, c'est-à-dire précisément une partie de la "trajectoire" d'un réseau. Ainsi, les méthodes de mesures par sondes actives peuvent être considérées comme des problèmes inverses pour une théorie des réseaux qui permettrait une prédiction exacte de l'évolution des réseaux.

Nous montrons dans ce document comment les méthodes de mesures par sondes actives sont reliées aux problèmes inverses dans la théories des files d'attentes. Nous spécifions comment les contraintes de mesures peuvent être incluses dans les problèmes inverses, quels sont les observables, et détaillons les étapes successives pour un problème inverse dans la théorie des files d'attentes. Nous classifions les problèmes en trois catégories différentes, en fonction de la nature de leur résultat et de leur généralité, et donnons des exemples simples pour illustrer leurs différentes propriétés.

Nous étudions en détail un problème inverse spécifique, où le réseau se comporte

comme un réseau dit “de Kelly” avec $K$ serveurs en tandem. Dans ce cas précis, nous calculons explicitement la distribution des délais de bout en bout des sondes, en fonction des capacités résiduelles des serveurs et de l'intensité des sondes. Nous montrons que l'ensemble des capacités résiduelles peut être estimé à partir du délai moyen des sondes pour $K$ intensités de sondes différentes. Nous proposons une méthodes d'inversion alternative, à partir de la distribution des délais des sondes pour une seule intensité de sonde. Dans le cas à deux serveurs, nous donnons une caractérisation directe de l'estimateur du maximum de vraisemblance des capacités résiduelles. Dans le cas général, nous utilisons l'algorithme Espérance-Maximisation (E-M). Nous prouvons que dans le cas à deux serveurs, la suite des estimations de E-M converge vers une limite finie, qui est une solution de l'équation de vraisemblance. Nous proposons une formule explicite pour le calcul de l'itération quand $K = 2$ ou $K = 3$, et prouvons que la formule reste calculable quelque soit le nombre de serveurs. Nous évaluons ces techniques numériquement. À partir de simulations utilisant des traces d'un réseau réel, nous étudions indépendamment l'impact de chacune des hypothèses d'un réseau de Kelly sur les performances de l'estimateur, et proposons des facteurs de correction simples si besoin.

Nous étendons l'exemple précédant au cas des réseaux en forme d'arbre. Les sondes sont multicast, envoyées depuis la racine et à destination des feuilles. À chaque nœud, elles attendent un temps aléatoire distribué de façon exponentielle. Nous montrons que ce modèle est relié au modèle des réseaux de Kelly sur une topologie d'arbre, avec du trafic transverse unicast et des sondes multicast, et calculons une formule explicite pour la vraisemblance des délais joints. Nous utilisons l'algorithme E-M pour calculer l'estimateur de vraisemblance du délai moyen à chaque nœud, et calculons une formule explicite pour la combinaison des étapes E et M. Des simulations numériques illustrent la convergence de l'estimateur et ses propriétés. Face à la complexité de l'algorithme, nous proposons une technique d'accélération de convergence, permettant ainsi de considérer des arbres beaucoup plus grands. Cette technique contient des aspects innovant dont l'intérêt peut dépasser le cadre de ces travaux.

Finalement, nous explorons le cas des problèmes inverses dans la théorie des réseaux à partage de bande passante. À partir de deux exemples simples, nous montrons comment un sondeur peut mesurer le réseau en faisant varier le nombre de flots de sondes, et en mesurant le débit associé aux flots dans chaque cas. En particulier, si l'allocation de bande passante maximise une fonction d'utilité $\alpha$-équitable, l'ensemble des capacités des réseaux et leur nombre de connections associé peut être identifié de manière unique dans la plupart des cas. Nous proposons un algorithme pour effectuer cette inversion, avec des exemples illustrant ses propriétés numériques.

**Mots-clés :** Problèmes inverses — tomographie d'Internet — mesure par sondes actives — statistique — théorie des files d'attentes — algorithme Espérance-Maximisation

# Organization of the dissertation

This dissertation lies at the crossroad of several fields: queueing theory, active network measurement, statistics and the theory of bandwidth sharing networks. The presentation aims at making the manuscript readable by anyone knowledgeable in probability theory. This leads to a large introduction in the first chapter, covering the needed notions of the different fields. The operation of real networks, in particular of Internet, is explored in section 1.1. Section 1.2 covers basic notions of queueing theory. Section 1.3 introduces the theory of bandwidth sharing networks. The relevant definitions and theorems of statistics are presented in section 1.4. Knowledgeable readers can easily skip the corresponding sections, and read directly section 1.5, which is the "classical PhD introduction". For those whose interest lies mostly in the philosophy and aims of this work in particular, and active probing in general, it is possible to have a quick overview by reading sections 1.1.1, 1.1.2, 1.1.3, and then directly section 1.5.

# Contents

# Chapter 1

# Introduction

## 1.1 Networks

This dissertation has a strong focus on measurement based inverse problems in communication networks. Whilst many readers are already familiar with communication networks and their theoretical side, we will introduce very succinctly, in a simple manner, the necessary concepts in this section.

### 1.1.1 What are networks

Throughout this dissertation, a *network* will be considered as a set of *vertices* (or locations), and a set of *edges* (or links) between these vertices. Hence, the network is connecting the different vertices with the edges. Depending on the nature of the network considered, the vertices and edges can have different incarnations.

**Example 1.1.1** (Different examples of networks)**:** The first natural example of network is the Internet. The Internet vertices are the home computers of Internet users, the routers and switches ensuring the connectivity of the network, and the servers that store the data of interest to users. These vertices are not identical, and do not have any symmetrical role for the network; however, each of these classes of nodes has a vital role for the network. In this example, edges are a lot more similar. They consist of any physical connection between any pair of these locations, be it using copper wires, coaxial cables, fiber optics or radio spectrum links. They all carry the data between different locations. The network as a whole is focused on carrying information between data servers and end-users. The Internet is of specific interest because it is one of the dominant network today, which use has become vital for the economy. Additionnally, the Internet is by its very nature dynamical in the middle and long term: new wired links are connected regularly, and some are shut-down, the wireless connections are by their nature not permanent, and most importantly, the content on the Internet and the usage of the network vary significantly along the years. This leads to a strong need of measuring and understanding this network.

A second interesting example is the almost static network of the streets and roads of a

city or country. The edges are easy to identify: they are the roads and streets. The vertices are by definition the extremities of the edges, *i.e.* the crossroads, squares and junctions. Even if they have a natural definition, they do not have any specific role in this network. In fact, this network aims at allowing vehicle movement, and our interest in it lies in the edges, which have a natural definition of weight, corresponding to the capacity (in vehicle per minutes) of the road. Such a graph (or network) is said to be weighted, and a primary metric of interest is the maximum flow from one source to one destination, *i.e.* the maximum number of vehicles per second that the road network can carry between both locations.

The third example of network we will present here is called *social network*: it is a different kind of network, and is a common object of study in social and computer sciences. It arises as a model for the interactions of human beings. Its vertices are the human beings themselves, or the groups of human beings, and edges appear between two vertices when both vertices are in contact. The canonical example for this kind of network is the friendship relation network on the Facebook website: each Facebook profile corresponds (in theory) to one human being, and is a vertex of the social network. Each profile has a list of friend profiles, and to each of these friendships corresponds one edge between both friend profiles in the social graph. Studies of social graphs aim at recovering the different communities from these friendship connections, or at quantifying the diffusion of information due to gossips in a social network. Such networks differ from the two previous example in that there is no physical reality for the edges: the links are not physical connections ensuring that data or vehicles can be transported, but are a formal understanding of relation between people. In fact, in some examples, there is no unique answer whether an edge is present or not: friendship between human beings (except on online social networks as Facebook) is not a binary question, and being someone's friend does not mean you see him regularly, or exchange any piece of information immediately.

Considering that no edge is removed from or added to the network at the time scales we are interested in, we will limit ourselves to *static* networks, which can then be considered as a *graph* $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the (finite) set of vertices and $\mathcal{E}$ the (finite) set of edges. Because of this equivalence, we will use both the terminology of graphs and networks in this thesis.

As shown in Example 1.1.1, networks are of broader interest than only communication networks, and most of the results presented here can be generalized to other kinds of networks: in particular, we will see in section 1.3 that networks can be a good model for and give insight into objects that have no obvious connections with them. However, communication networks will be the connecting thread of this document, and we will now spend some time presenting key concepts concerning them.

The aim of communication networks can be described in a very general way as carrying data between nodes. The specific meaning of "data" and the way to carry them depends on the nature of the network and the technology chosen to operate it. The former can vary from binary packets to real-time streaming, and the latter is for instance chosen among wireless connections, copper links and optical fiber connections[1].

[1] A technology usually includes much more precise information about the encryption and communications

### 1.1.2 Network applications

Before going into a more detailed study of networks, it is useful to start with a few examples of network applications, and how they impact the design of a network. As the Internet is the main focus of this dissertation, the remaining part of this section (up to section 1.2) will be dedicated to introducing the key concepts and mechanisms of the Internet, and in which aspects it is similar to or different from other networks.

In the computer science terminology, a network application is a sequence of "unitary" tasks, which uses the network in order to perform a global task. As such, applications run at the end-point (or edges, or host, or end-system) of the connection, and are by nature distributed over several nodes.

#### Internet applications

We will start with four Internet killer applications, which are responsible for most of today's traffic [BDF+09, MFPA09]:

1. The World Wide Web (WWW);
2. E-mail, including web-accessible e-mail and attachments;
3. Peer-to-peer file sharing, pioneered by Napster, and now dominated by BitTorrent;
4. Live streaming, including radio and television broadcasts on the Internet, but also Skype phone calls or Youtube video watching.

**The World Wide Web** The Web is composed of two main components:

- the *HyperText Markup Language (HTML)*, which is a common language to all web-sites and web browsers, used to specify the content and presentation of any web page;
- the *HyperText Transfer Protocol (HTTP)*, which is a communication protocol, ruling how web-servers and browser exchange their data.

HTML is outside the scope of communication networks. It will be enough to know that it is a structured language, which allows one to write plain text documents and add tags to structure the text, include images, links or other objects. An HTML document can then be interpreted by a web browser (*e.g.* Internet Explorer or Firefox) in order to display its content on a screen. The different tags and the language syntax is specified by the *World Wide Web Consortium (W3C, [W3C])*, such that new browsers and new web sites can be easily designed.

protocol (both GSM and 802.11 networks are wireless communication, but they are for some aspects much more different than 802.11 and ADSL connections) and similar details, but this is outside the scope of this thesis.

HTTP is the network application used to transfer the HTML objects between web servers and browsers. Is is a distributed application, with 2 different pieces of software, called HTTP server (*e.g.* Apache) and HTTP client (*e.g.* web browsers). These software run respectively on the web servers and the client end-hosts (*e.g.* desktops, laptops, smart phones, etc.). HTTP has public specifications, which one can find in [BLFF96, FGM+97, FGM+99].

Now, what happens when one wants to see a webpage? Let us consider for example that Alice wants to learn more about HTTP, and consults the wikipedia page about HTTP[2]. Alice launches her favorite browser Mozilla Firefox, and in the URL bar, types the URL. Her browser, formally a HTTP client with a graphical user interface, tries to open a connection with the node *en.wikipedia.org*. If successful, it then sends a message, which could read[3]:

```
GET /wiki/Http HTTP/1.1
Host: en.wikipedia.org
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; fr;
            rv:1.9.0.1) Gecko/2008070208 Firefox/3.0.1
[blank line]
```

The server then answers a message, which could be:

```
HTTP/1.0 200 OK
Date: Fri, 27 Aug 2010 21:32:09 GMT
Server: Apache
Last-Modified: Fri, 27 Aug 2010 20:06:40 GMT
Content-Length: 114962
Content-Type: text/html; charset=UTF-8
Connection: close
[blank line]
(data...data...data... ...data)
```

In a human fashion, this dialogue would read:

```
Client: - Hello, en.wikipedia.org. Can we talk? (Ask connection)
Server: - I'm listening to you. (Accept connection)
Client: - Could you send me a file (request GET)?  This is the
          file /wiki/Http. I'm using the protocol HTTP/1.1. By
          the way, in case several of you live at the same
          address, I'm speaking to en.wikipedia.org (this is the
          line Host:...). In case you are interested, my web
          browser is currently Firefox, version 3.0.1 for
```

[2] *http://en.wikipedia.org/wiki/Http*

[3] In fact, many options would most likely be included in the message, making it much longer. However, they do not impact directly at the heart of the protocol, and its principles can be explained and understood with a short message.

```
          Windows Vista (line User-Agent:...).
Server: - I prefer protocol HTTP/1.0. I'm sending you the
          required file at the end of this paragraph (code 200
          OK). My current time is 21:32:09 GMT, on Friday
          08/27/2010 (line Date:...). I'm running the Apache
          version of HTTP server (line Server:...). The file I'm
          sending you was last modified on 08/27/2010, at
          20:06:40 GMT. It is 114 962 bytes long (line
          Content-Length...). Is an HTML file, encoding in UTF-8
          characters (line Content-Type:...). And just to let you
          know, I'll break contact as soon as I've sent the file
          (line Connection:close).

          [File here]
```

This dialogue is repeated for every web page that Alice wants to display. Some web pages consist of several objects: the main object is the HTML file, which can include other objects (*e.g.* images), designated by their own URL. When trying to display this page, the browser will realize that other objects are required, and hence will ask the hosting servers to send the corresponding files.

GET is the easiest and most-used request for HTTP clients. On a broad scale, it works via exchanging a few "messages" (either the headers, *i.e.* the lines preceding the HTML object, or the HTML object itself) between the HTTP client and the HTTP server. Other requests are defined, for the cases where one wants to fill an online form, or delete a file on a online server, etc. These requests contain more data and more messages, but the protocol stays based on the exchange of specific messages.

We will finish this brief HTTP introduction with its consequences on network design. Which functionalities are required from the network, in order to let HTTP run on it? The list is short, but meaningful:

1. A naming functionality, which ensures that there is only one node with a given name; the difficulty here is that no server or organization on the Internet has a global view of the set of connected nodes, and the allocation of names hence can't be centralized. Additionnally, somes nodes need to have a fixed static unique name (in particular, web servers, such as *www.google.com*), whereas other nodes don't have this requirement (a end-host PC usually acts only as a client, and can deal with a different name each time it connects to the Internet).

2. A transport functionality, which can deliver messages from one node to any other node specified with its address;

3. A reliability-checking functionality, which ensures that the message was not altered during the transport;

4. A delivery-checking mechanism, which verifies that the message arrived at destination, and raises an alert if needed.

In theory, the fourth requirement could be dealt with in the HTTP application itself. However, this requirement is common, and it is easier to add this functionality to the network than to redesign it for every new application.

**E-mail** The application architecture is more complicated here. We will explain first how web-based mail servers work, and then see how mail user agents (*e.g.* Thunderbird or Outlook) can interact with them.

Consider that Alice (*alice@gmail.com*) wants to send an email to Bob (*bob@yahoo.com*). Alice starts her web browser, loads the gmail.com web page, and logs onto her Gmail account. This is for the moment pure HTTP exchange. She's quickly on the "compose Email" web page, she writes her email and the address of Bob, then pushes the "Send Mail" button. This button triggers a special HTTP request called *POST*, which allows a web client to send data to the web server. Here, the data consists of the email and the order to send it.

Each mail server runs two Simple Mail Transfer Protocol (SMTP) [Kle08, Res08] programs: one SMTP client program, which is responsible for sending mails to other servers, and one SMTP server program, which listens for incoming mails. In our case, the Gmail web server will pass Alice's email to the Gmail SMTP client. Looking at Bob's address, the Gmail SMTP client realizes that Bob's address is not from Gmail, and that he needs to contact the Yahoo SMTP server. The Gmail SMTP client then requests a connection to the Yahoo SMTP server, and starts a connection that looks like (in human presentation):

```
[Gmail requests a connection to Yahoo]
Yahoo: - I'm listening, and my name is smtp.yahoo.com.
Gmail: - Hello! I'm smtp.gmail.com.
Yahoo: - Hello smtp.gmail.com. Glad to hear from you.
Gmail: - I've an email from <alice@gmail.com>.
Yahoo: - OK
Gmail: - It's destinated to <bob@yahoo.com>.
Yahoo: - That's fine: I know Bob.
Gmail: - I'm sending the email.
Yahoo: - I'm ready to copy it. Just signal the end with a "." line.
Gmail: - From: "Alice <alice@gmail.com>"
         To: "Bob <bob@yahoo.com>"
         Date: Mon, 30 Aug 2010, 10:13:42 GMT
         Subject: Weather

         Hello Bob,
         Can you tell me what the weather is at your home? I'm
```

```
          hesitating about taking my umbrella.

          Looking forward to visiting you,
          Alice.
          .
Yahoo: - I've got it, and I'll give it to Bob.
Gmail: - Thanks. Bye!
Yahoo: - Bye!
[Yahoo closes the connection].
```

After reception, the Yahoo SMTP server will copy the email onto its file system, and add it to Bob's inbox. Later that day, Bob eventually connects via the web interface, asks for new mails, and the Yahoo web server will send him an HTML file that lists his mails, including the one from Alice.

We presented the simple case where everything works without any problem. The Simple Mail Transfer Protocol covers many more situations. If, for example, the client SMTP application (the sender) cannot find the server SMTP application (the destination), the client will repeatedly try to send Alice's mail to Bob's mail server, let us say every 30 minutes, until successful. These retries are also specified in the SMTP protocol.

SMTP shares a lot of similarities with HTTP. Both rely on a server-client architecture, and reliable network connections between both hosts. Both use headers as a way to agree on their operations. The main difference is that HTTP is principally a *pull* protocol, where the client requests the data it needs, and the server then sends this data. On the opposite, SMTP is a *push* protocol: the client pushes its data to the server, and the server accepts it.

Our presentation would not be complete if we did not cover the case of non web-based emails. If Alice or Bob do not want to use a web-based email, they can use a mail user agent, such as Outlook or Thunderbird. The advantage of mail user agents is that they work on one's own end-host, and hence are more reactive to user inputs, and allow to work offline. But this adds two new links in the email chain, between Alice's mail user agent and her mail server, and Bob's mail server and his mail user agent. The case of Alice has an easy solution. After all, she must be connected in any case when she wants to send an email: hence, her mail user agent can act as an SMTP client and push her mail directly to her mail server, which will forward it to Bob's mail server[4]. If Bob's mail server is not available, Alice's mail server will retry regularly to send the mail, following the protocol. Hence, Alice can no go offline, and be confident in the fact that her mail will eventually arrive at destination. If Alice's mail server is not available, the case is more complicated. But since she is a client for her mail server, she has a way to complain if her mail server is not available, where as she cannot do anything if Bob's mail server is unreachable.

The last link of the chain, from Bob's mail server to Bob's mail user agent, is more

[4]One might try to remove one step and have Alice's mail user agent push directly her mail on Bob's mail server, but this does not allow to identify Alice, and things become more complex if Bob's server is temporary not available.

complicated. SMTP cannot be used here: remember that SMTP is a push protocol, where the destination server is assumed to be (nearly) always reachable, and Bob might sometimes turn off his computer, have no wifi connection, etc. To solve this issue, new protocols, called Post Office Protocol (POP3) and Internet Mail Access Protocol (IMAP), have been developed. POP3 is the simplest: it is a pull protocol, with a client-server architecture. The mail user agent runs the client side, and asks for the data (*i.e.* the mail list and mail contents) from the POP3 server side, which runs on the mail server. The detailed design of POP3 is fairly similar to HTTP. IMAP is slightly more complex, because it is able to synchronize the mail user agent side and the web server side, and allows mails to be stored in folders on the server side. It acts both as a pull and push protocol in that manner.

To summarize this part, the network requirements to run an email application are again:

1. A naming functionality, enabling one to identify the mail server for any email address;
2. A transport functionality, which allows messages to be sent from one node to another node;
3. A reliability-checking functionality, which ensures that any message sent was not altered during its journey in the network;
4. A delivery-checking mechanism, which notifies whether the message arrived at destination.

To understand in fact the real first requirement, we must signal here that a specific application, called *Domain Name System (DNS)* has been developed and deployed on the whole Internet. The purpose of DNS is to keep and publish lists of human readable host names, such as *en.wikipedia.org*, and the corresponding more binary IP addresses. Hence, when Alice wants to consult a web page on *en.wikipedia.org*, Alice will type the URL in her favorite browser, and the browser will then send a DNS request to know the IP address corresponding to the specified host. After the DNS server answers, the browser can then send its HTTP request to the *en.wikipedia.org* HTTP server, specifying to the network the correct IP address. Similarly, DNS keeps track of mail domains and their associated mail servers. In our previous example, when Alice's Gmail mail client wants to forward Alice's mail to Bob's Yahoo mail server, it will ask a DNS server about which node is responsible for *@yahoo.com* mails, and get the address of Yahoo mail server. Hence, the first requirement now becomes "A naming functionality, which ensures that there is only one node with a given name", similarly to the HTTP requirement.

**Peer-to-peer file sharing** This is maybe the most debated application of the Internet, for legal copyright reasons. However, from a technical measurement point of view, one cannot dismiss peer-to-peer (P2P) file sharing applications, as they are responsible for a significant part of today's traffic [BDF+09, MFPA09].

There is no single protocol or application for P2P file sharing: many different approaches have been used by different programs. However, from Napster to Kazaa to the

currently prevailing BitTorrent, any P2P file sharing system relies on the same few principles. Any P2P application is by definition a distributed application, where the same code runs on most end-hosts. The fact that all hosts belonging to the network run the same code, and hence are "equal" or "peers" is indeed the origin of the name "peer-to-peer".

Assume, for example, that Alice wants to download the song *Yesterday* by *The Beatles*. Alice launches her P2P client, and starts searching for the song. Her client contacts the index of the peer-to-peer system[5], and simultaneously, asks for peers with the *Yesterday* song and publishes on the index the list of files that Alice can share. The index answers with a list of peers that Alice's client can contact, including Bob's peer-to-peer client, and Alice's client can then start to ask Bob's client for the song. This request for the song is then similar to the HTTP case and many other pull protocols: Alice's client wants a specific object, knows where this object is hosted, and contacts that host directly. In fact, many peer-to-peer systems use the HTTP protocol for the download of files. Simultaneously, Carla might be searching for *Hello, Dolly* by Louis Armstrong, and Alice indicated earlier that she made that file available for download. Carla's client will then contact Alice's client, and ask for the file. Alice's client will then simultaneously act as a client downloading from Bob's host and as a server, delivering an object at Carla's client request. When Alice decides to switch her computer off, her P2P client will inform the index, and Carla's client will ask another host with *Hello, Dolly* to send the remaining part of the file.

There are many more details that would need to be specified for the peer-to-peer system to be complete. In particular, the implementation of the index and the request and answer templates must be fully specified. The number of peers one client might contact for the download, as well as the choice of these peers can be the object of optimization. However, any peer-to-peer file sharing system will rely on these three principles:

1. Maintain and use a (centralized or decentralized) index for localizing the content of the system;

2. Peers contact directly other peers in order to download the files they are interested in;

3. Reciprocally, peers answer to other peers' requests for file download, and act hence as servers.

The main advantage of peer-to-peer systems is their scalability. The more peers there are in the system, the more requests are made, but simultaneously, the more hosts can answer these requests. Both numbers grow at the same rate. This means that, outside an eventual centralized index, there is no bottleneck in the system, and the performance should be the same with a thousand members or a million members. At the opposite, the world wide web architecture is not scalable: a single (or a few) hosts have to answer all requests about a web

[5]We are voluntarily not precise here: the index can be either a centralized index on a server, or a fully-distributed index on the peers, which Alice's client can access thanks to a few peers it knows, or some hybrid solution. Different peer-to-peer systems have used different solutions, but these make very little difference to the network requirements.

site. If the web server is not quick enough, requests will have to queue before being served, or can be lost.

The network requirements here are once again identical to the web and email cases: the network needs to be able to name hosts, transport messages between any pair of hosts, and ensure these messages are delivered and not altered.

**Live streaming** It is possible today to listen to radio broadcasts or watch TV on the Internet. Although the current quality is not as high as one can get with classical radio spectrum or dedicated cable transmission, Internet based radio and television do not require a heavy infrastructure to broadcast. Additionally, because radio or television on the Internet are broadcast at the request of users, there is (nearly) no limit to the number of available channels, whereas the radio spectrum is limited, and hence can support a finite number of channels. Finally, as the Internet is deployed all over the world today, it allows anyone to access his favorite broadcast from anywhere.

There are many different protocols for live streaming, suited to different cases. Some are used for radio or television on the Internet, others are more dedicated to phone on Internet systems, such as Skype, and some have been designed for audio or video conferences. We will present here only, and briefly, the Real-time Transfer Protocol (RTP) [HFGC98, SCFJ03], but other protocols are similar. As its name suggests, RTP is a real-time protocol, and this will lead to significant differences for network requirements compared to the three previous cases we studied.

The RTP protocol is designed to work in pair with the Real-time Transfer Control Protocol (RTCP). In short, RTP is responsible for the transfer of the media content, whereas RTCP takes charge of the control of the broadcast, *i.e.* the specification of requests, quality feedback and similar details. The RTCP protocol relies on a reliable exchange of messages, in a close way to previous protocols.

Live streaming starts with a digital copy of the media broadcast. This digital real-time encoding step is not specified by the protocol itself, and any encoding / decoding scheme can be used, as long as the broadcasting source and the final user agree on it. Once a new "chunk" (we will call these chunks frames) of the broadcast is available, the source will add a header that specifies when to play this chunk and append a sequence number, then send it to the source. The sequence number is increased by one for each frame, and will allow the destination to reorder the frames if needed, and detect lost frames. Lost frames will not be sent again, as they would possibly arrive too late, but intelligent software try to minimize as much as possible the impact of such losses. The timestamp in the header allows the destination to replay the frames at the correct speed.

Taking a look at what is required for the network here, one can list:

1. A naming functionality, which allows to designate uniquely hosts;

2. A transport functionality, which carries messages between any pair of hosts;

3. A low loss rate between the source and the destination;

4. Ideally, a bound (or at default, a low variation) on the delay between the hosts;

5. A low delay between hosts.

We can see here the impact of real-time on the design: the fact that one has no time for retransmission makes reliability-checking and delivery-checking functionalities useless. On the other side, one must ensure or hope that enough messages will go through for the broadcast to be correctly decoded, and that the message will arrive in time.

### Postal network

We will present here another network, and three among its potential applications. The postal network is composed of the union of all postal companies or administrations all over the world. The vertices of this network are the letter boxes, the post boxes, as well as any internal center for grouping, sorting, dispatching mails. The edges are the rounds of postmen and the exchanges between different centers.

**Bank statement monthly sending** Banks often propose[6] a free bank statement each month. Internal procedures allow the bank to establish these bank statements. The question then arises about how to make them available to the client. Several solution are of course possible, including Internet access or email for electronic statements, or letting the client fetch the statement from his local bank.

A common if not universal solution is to send these statements directly to the client home using postal mail. Every account holder has some postal address, even if he has no regular Internet access, and many people still prefer to store paper-based archives rather than electronic archives.

What is needed for such an application of postal networks to be possible ? First, the communication must be asynchronous, *i.e.* it must not require the client to be available personally to take delivery of the statement. The sending happens at regular times, but as for most push protocols, these are not controlled by the user, and most people have a regular activity during working hours, such as a job or studies. Second, the bank must be able to send each report privately to the correct client, hence needs to be able to name precisely and uniquely each client. Third, the network delays must be reasonably low compared to the monthly frequency of sending. Finally, the network must be able to transport the message at a sufficiently low cost for this solution to be economically feasible.

We did not put the delivery-control functionality as a requirement here, because the client knows that he is to be sent a bank statement, and can ask for another one if needed.

[6] At least in France, by law, it must be proposed for free to any client.

**Summoning to appear in court** Courts conduct trials, as they should. The norm in democracy is for the defendant to attend the trial, in order to be able to defend himself. If for some reason, he is already in custody, the court can make sure that he will be there. However, in many cases, the defendant is not in custody before the trial, and the court must inform him of the date and place of the trial, and eventually even that he has been brought suit.

This expresses the need for the court to be able to send a summoning message to precise people. Let us list the requirement for a network transporting such messages:

1. It needs to uniquely identify and name any human being in the country;
2. It must keep the message private, to preserve the presumption of innocence;
3. It must be reliable, as one cannot be accused of non-appearance before the court if the message was not received;
4. It must not assume that the addressee is available at the moment of delivery, since he might not be aware that he will be delivered a summoning to appear in court;
5. It must be reasonably fast, so that the court can plan trials with a reasonable delay.

**Urgent parcel periodic delivery** The previous example of court summoning introduced the constraint of reliability for postal networks. This last example will present the need for speed in some cases. Consider the case of a hospital, which periodically requires new drugs. Some of these drugs must be kept cool in a specific state, which necessitates an upper bound on the time it spends in the network. As they are crucially needed for the good running of the hospital, these deliveries cannot be significantly delayed: the hospital needs a minimum amount of drugs per week. Contrary to the previous cases, the hospital is in fact requesting these drugs, and they have significant value for it: it is hence possible for it to make sure that someone is available for reception either permanently or at specific delivery times.

The case of hospital drugs deliveries might seem really restricted. It however appears (slightly modified) in many other fresh-product trading situations. The list of functionalities required from the network is as follows:

1. A naming functionality, allowing to designate uniquely the addressee of the message;
2. A transport functionality, which carries messages between any pair of hosts;
3. Ideally a bounded low delay, between the source and the destination;
4. A guaranteed transport capacity, which ensures that enough deliveries can be made per unit of time to meet the required (daily or weekly) demand on drugs. One do not only need to deliver drugs, but also to deliver *reliably enough* drugs for the good running of the hospital.

**Social networks**

We take here a last example of network and applications that can be run on it. Social networks have already been presented in example 1.1.1, and we will consider them here with the specific incarnation of Facebook in mind. Recall here that we consider only the social network of Facebook profiles, and not the specific Facebook Internet application, based on HTTP exchange between profile owner and the Facebook web server.

Social networks in general allow two main applications: private message exchange, and public opinion or news release. Private message can of course be destined to a direct friend. However, the power of some social networks is that one can ask friends to forward a message (for example, a request for favor), or to present the sender to the next friend in the path. Since these forwarded messages or presentations come from personal acquaintances, they are often better received than when they come from an unknown person. Private messages can of course include direct requests for favor, but also opinions and requests for advice about some project in a professional network, reform proposition, personal application or government composition in political networks. A public opinion or a news release corresponds to cases where one wants everybody, in the long term, to know that one has moved to another city, had a child, or is looking for a job opportunity, or expresses a global opinion on a subject that affects many people. Whilst this is mostly not urgent, any acquaintance can access the information, and is free to forward it.

In the Facebook website, private messages are called, well, private messages, and are basically similar to web-based emails. Public opinion or news releases are a lot more developed. All the following actions fall into that category:

- personal status update: Facebook personal status usually indicates the current mood of the profile owner;
- personal information update, such as email, city, hobbies or job;
- public comment, on one's profile *wall*: a profile *wall* is a place where every friend of this profile can write public comments, which can then be consulted by any other friend of this profile. It is often used for some casual group discussion, or comments about the personal status updates.

We have here two applications. One is point-to-point, *i.e.* from a single source to a single destination, and requires a non-obvious task of addressing and routing to reach the intended destination: one needs to be aware of who knows who. It also requires a reasonable reliability in the message forwards, or a delivery-checking mechanism (which can be just a simple acknowledgment). The other one is point-to-multipoint, *i.e.* has multiple (here anyone who is interested in the message) destination. It does not require fast or reliable transmission, and low-cost opportunistic transmission seems ideal for it.

**Elastic and streaming applications**

Most of the applications we have presented have no delay or bandwidth constraints. They can work with and adapt to low bandwidth. More is always better, but emails can still be used if they need one hour to be sent, and in principle (outside the human user comfort), nothing prevents the world wide web usage if one hour is needed to display a web page. We will call such applications *elastic*, because their bandwidth allocation can grow or shrink. Elastic applications have this interesting particularity that they usually do not use the network for a specific time, but for a specific number of messages. The time of connection then depends on the bandwidth used by the application, and the higher the bandwidth is, the shorter the time of connection is, and the faster the web page is displayed on the screen. Most of the elastic applications use or prefer a reliable transfer protocol. These applications are based on the exchange of many messages: what could be considered as only one network operation by the user (for example, the download of one single web page, or the sending of one email to one addressee) often requires the exchange of several messages between end-hosts. These messages are hence generated in a bursty non-fluid manner.

Some applications require a minimum bandwidth or a maximum latency to function: this is the case of live streaming and urgent parcel delivery in the example we presented. These constraints are usually originated by the real-time nature of the system: one cannot delay the sending of a new message in case of insufficient bandwidth, and this messages must arrive in time. A side-effect of these time and bandwidth constraints is that these applications usually neither require nor use reliable transfer protocols, because they cannot afford the time to resend lost messages. They rely on error-correcting codes to be able to recover from the losses. Such applications will be called *inelastic*, *bandwidth-sensitive* or *streaming* applications. Compared to elastic applications, streaming application are less bursty: they also rely on the exchange of several messages, but these are generated on a regular, almost periodic basis.

### 1.1.3 Functionalities of networks

We have seen in the previous section that several functionalities are required from the network, in order to be able to run applications. We will here list the main functionalities. In two examples of networks, we will provide additional insight on how these functionalities are provided, which will be useful in section 1.1.4, where we will present mathematical models of networks based (in part) on these functionalities.

There are five broad requirements stemming from applications for any communication network:

1. An addressing functionality, which allows one to designate precisely the nodes of the network;

2. A localizing and routing functionality, which determines a route from any node to any address;

3. A transport functionality, which can carry messages over a network;

4. A reliability-checking functionality, which ensures that the message transmitted over the network have not been altered;

5. A (non-mandatory) delivery-checking functionality, which ensures that the message has been received, and raises an alert if needed.

One must add a sixth requirement, in order to keep a network operational:

6. A congestion-control functionality, which ensures that the network load does not increase too much, and prevents network collapse.

Functionalities 3 and 4 are highly dependent on the specific network, or even on the technology used for a specific link. Transportation over a link is not the same between a post box and the local center and between the Paris city postal center and New-York city postal center. The reliability-checking functionality usually involves either error-checking codes (*e.g.* parity check bits), or a more materiel process such as a closed envelope. As it is highly dependent on specific implementations, we will not cover details, and assume that the network provides it.

In the specific case of packet-switching networks, including the Internet, the transport functionality divides any message in little chunks, called packets. Small messages can be a single packet, but large messages will be composed of many packets, possibly millions of packets for large files of size in the order of giga-bytes. This allow to interleave packets belonging different messages, and hence serve simultaneously several messages (with a lower rate).

The congestion control functionality is critical to keep networks efficient. It's aim is to keep the loss rates low. As losses in communication networks are mostly due to temporary buffer overflows[7], a side effect of congestion control is to also decrease the network delays. To understand the importance of this functionality, consider a heavily loaded network where most routes have several links. If at each link, each packet has the same probability to be dropped, this means that some packets will be lost close to their destination. If they already went through a bottleneck, part of this precious limited capacity of the network will have been lost to transport messages that do not reach their destination. The throughput of the network can drop to low levels, because of these losses.

Congestion control is often implemented in two different ways. For elastic traffic, the most common solution is rate control, *i.e.* every application is accepted and can send traffic, but the rate at which it can send traffic is limited (either directly, or in an indirect way). This works well with elastic traffic, which has by definition no bandwidth constraint. On the other hand, streaming traffic is bandwidth sensitive, and it makes no sense to require a bandwidth sensitive application to send messages at a lower rate than their minimal rate.

[7]This means that locally on a server, packets arrive suddenly and the number of packets to be stored before being sent further on their route exceeds the storage capacity (called buffer): the exceeding packets are then dropped.

Hence, the usual solution is admission control, *i.e.* new applications are refused access to the network when they would induce overload. In phone networks, this corresponds to the message "All circuits are currently busy. Please try again in a few moments."

The other functionalities are covered in the following examples:

**Example 1.1.2** (Postal network)**:** The addressing functionality of this network is easy to identify. It's the postal addresses, which identifies a unique box.

The localization and routing is also quite natural. Postal addresses are (partially) recursive: they include the country, which is unique, and cover a small and (usually) connected part of the world. The state (or French *departement*, etc.) is then the next division, again in small and (usually) connected parts. Cities often come next, once again division is in to smaller and connected sets. Most systems then use different post codes for smaller areas, and finally the street, the number and the name for the precise letter box. This hierarchical addressing leads to a natural easy solution for routing. To make it more concrete, consider a letter posted from somewhere in Los Angeles, California, USA, to somewhere in Marseille, in France. Starting from the first post box, the next node is automatically the local grouping and sorting center. This center will (usually, detailed organization differs between companies) determine whether the letter is to be sent within the same city or not. If this is the case, the letter is then forwarded to the right part of the city. Here, this is not the case, hence it will be forwarded to the state postal center (which might in fact be the same). In the state center, the letter will then be sent to some "foreign outgoing center", for letters to be sent abroad. The next node is then some French incoming center, which receives letters from abroad (likely in Paris). The Parisian center will then realize that the destination is in France, and forward the letter to the department *Bouches-du-Rhône* center. This center will then use the detailed post code and forward the letter to the post office closest to the destination, where it will be sorted and handed to the postman who will put it in the correct letter box. To generalize this example, one can divide the postal network into layers: post boxes and letter boxes belongs to the layer 0. The local post offices, grouping letters from or to a small area of the city are the layer 1, and cities centers, which centralize everything leaving or entering a city are layer 2, and so forth. Edges between nodes occur only between two nodes of neighboring layers, or in the same layer[8]. Usually, one node would have only one (or a very few) links with nodes of higher layers (this is the hierarchical part), a few links with nodes of the same layer, and lots of links with nodes of lower layer. Each node then forwards a letter to:

1. one of its children nodes (a node from a lower layer), if the destination belongs to the area covered by that node;

2. a peer node (a node from the same layer), if the destination belongs to the area covered by that node;

[8]In fact, if two companies have a different number of layers, this would be true only if we were numbering from the highest layer. But we can assume without any loss of generality that all companies have the same number of layers.

3. its parent node (or one of its parent nodes, depending on the country or state of the destination), if the destination does not belong to its area, or to the area of any peering node.

The hierarchical structure makes it easy to determine the cases 1 and 2, since one only needs to read a specific part of the address.

Most letters have no explicit delivery-checking functionality. Whilst almost all letters arrive correctly at destination, there is no guarantee that a typical letter is not lost, and no "specific" warning if it is (but one may realize it because of the absence of an answer, or during a discussion with the receiver). For important mails, there is a reliability-checking option, which lets you know when the receiver actually received your letter: these are registered letters with recorded delivery.

Finally, there is no specific congestion-control mechanism in postal networks. However, they can have only a finite number of letters entering the network each day: the post boxes have a finite size, and they are emptied only a few times a day. Once a post box is full, clients cannot post their letters, and hence the total load is bounded. To ensure that the networks do not collapse, it is then enough to increase the capacity of the postal network until it is higher than this upper bound[9].

**Example 1.1.3** (the Internet)**:** The second example we take is our main focus: the Internet. The addressing functionality of the Internet are the well-known Internet Protocol (IP) addresses. Each IP address is a succession of four eight-bits numbers[10] separated by periods, often written in decimal manner (hence the addresses like 125.84.3.247, where each number is lower than 256). IP addresses are allocated by the Internet Assigned Numbers Authority (IANA), usually in "blocks" to the Regional Internet Registries (RIR), which further allocate lower blocks to Internet Service Providers (ISP) and other entities. Hence, for example, all IP addresses whose first group of 8 bits is 41, 154, 196 or 197 are allocated to the African Regional Internet Registry (the African Network Information Center, or AfriNIC), which will further allocate it in smaller blocks (usually blocks of 1024 IP addresses, which will share the same prefix of 22 bits) to African ISPs. Whilst this is not an absolute requirement, ISPs tend to also allocate their IP addresses to their end-users following a hierarchical pattern.

This hierarchical addressing leads to the same natural localizing and routing techniques, which are also part of the Internet Protocol, as in postal networks. Looking at the first 8 bits of the IP address is enough to localize the RIR managing this address. Looking at the next 14 bits (for the AfriNIC case, the number might change for other RIRs) allows one to determine the ISPs providing this IP address. Similarly, these prefixes allow ISPs to forward

[9] In practice, it is likely that post companies only set their capacity in order to match the empirical load observed on busy days. GSM networks, which are in a similar situation (antennas can serve a finite number of calls simultaneously), serve most calls during the year. However, they are known to collapse at midnight for New Year, because the load is exceptionally high at this time.

[10] For the sake of simplicity, we consider only IPv4 addresses. The details of IP protocol are far beyond the scope of this thesis.

an IP packet to the correct node, depending on whether the prefix indicates they belong in the area they cover or not. A property of this routing scheme is that the sender is not aware of the route that its packets will follow. At each hop, the router locally decide what the next hop of the packet will be, based on the single destination address (which is stored in the packet header) and its routing table (which is stored on the router, and is independent of the incoming packets). This allow to limit strongly the need of information exchanges (a local end-user computer needs very little routing information, for example, since most packets will go to the contracting ISP). A second consequence is that the servers do not need to know to which message a packet belongs, they do not need to maintain "a state" of the connections, but only a routing table.

The Internet has two main "transport" protocols, whose main difference is their reliability. Transmission Control Protocol (TCP) is a relatively slow protocol, where the destination acknowledges each received packet to the source. In case of absence of acknowledgement, the source can resend the packet to the destination, hence ensuring a reliable transmission. It is used mostly for elastic traffic, where there is no prerequisite rate. User Data Protocol (UDP), on the other side, sends packets, and does not care (or verify) whether the destination received them. Its use consists mainly in real-time streaming traffic.

### 1.1.4 Abstraction of networks

From managing networks to provisioning new links to designing new protocols to troubleshooting networks, it is useful to have key insights of what happens in a network. It is possible to (re)play step by step the sequence of events in a lab experience with a real network. However, this solution is not really practical, and any result will be strongly specific to this particular network and traffic. There has hence been a strong effort to model both networks and their traffic, in the hope of having reliable but general solutions or guidelines for networks. This effort can be divided in two parts:

1. An effort to model networks, and in particular, routers behaviour. Most of the "interesting" events in wired networks appear at the routers and switches[11], and links usually lead to constant delays and no loss. With a good network model, it is then possible to simulate a network. The simulation of "big" networks requires plenty of computation power and time, but this is easier to perform than real measurements;

2. An effort to have a probabilistic description of traffic, depending on a few parameters and which has the same properties as real traffic. Full traces of traffic (*i.e.* the sequence of messages, specified by their arrival time, source, destination, and size) are difficult to measure with precision, and require lots of space to store, and are not easy to manipulate. A quick number might give a rough idea here: Cisco's CRS-1, one of today's Internet "best" routers, can scale up to 92Tb/s, or 11.5 terabytes per

[11] The distinction between routers and switches is beyond the scope of this dissertation. It is enough to know that both can forward messages to their next node in the path, looking at their address in the message headers. They just look at different kinds of address.

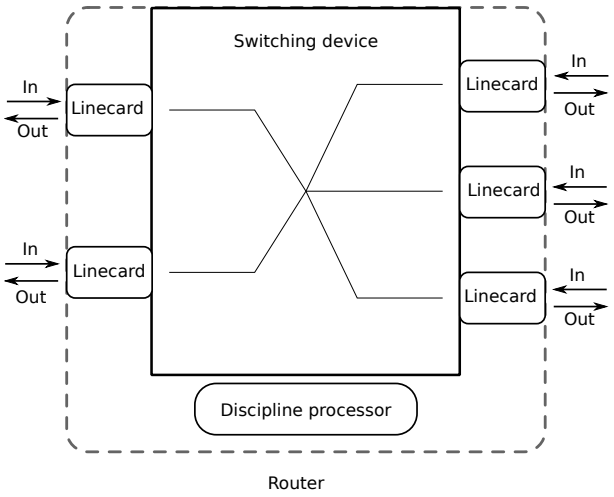



Figure 1.1: A diagram of a router.

second (a byte is 8 bits); the maximal size of an Ethernet packet being 1500 bytes, this means that this router can serve up to 7.6 billions of packet per second. It is hence useful to be able to characterize traffic with only a few parameters. Simulation can then be handled by sampling random traffic, and this allows also to tune the load finely to see the impact of different parameters, instead of being limited to the (few) cases of actual traffic measurement.

### Router model

We will study here routers of the *store & forward* type, which are the majority of today's routers in the Internet. Other networks have of course different routers, and this model might not be valid outside the Internet[12]. But the Internet is our main focus. As depicted in figure 1.1, a router of the *store & forward* type is composed of mainly three different elements: a *switching fabric*, controlled by a *centralized scheduler*, and *linecards* (also called *interfaces*). Each linecard controls two links: one input and one output.

A typical packet will cross the router as follows. When it arrives at the input link of a linecard, it is stored (hence the *store* part of store & forward) in the linecard's memory (called buffer). After the packet has fully left the input link and is stored in the buffer, its destination address is looked up in the forwarding table, in order to determine to which output link it must be forwarded. The packet is then stored in the First In First Out (FIFO) queue corresponding to that output interface. When it reaches the head of this queue, it is transmitted to the output linecard, possibly in different separated chunks (cells), where it is reassembled and handed to the output link scheduler (that's the *forward* part). It might then experience some queueing time, and eventually is serialised without interruption onto the output link. In the 'queueing' terminology, the packet is said to be served at a rate equal to the bandwidth of the output link, and the output process is said to be of fluid type, because

[12] In fact, it **will** not be valid for many different networks, such as the postal network.

packets flow out gradually, instead of leaving instantaneously.

The delay experienced by such a packet, defined as the difference between the time when the last bit of the packet left the router and the time when the last bit of the packet arrived in the router, can be decomposed in 6 parts:

1. the time needed to cross the input linecard;
2. the queueing delay in the input linecard queue, before being transmitted by the switching fabric;
3. the time needed to cross the switching fabric;
4. the time needed to cross the output linecard;
5. the queueing delay at the output link;
6. the service time to be serialized at the output link.

Losses appear when a queue is full, and a new packet has to be stored in it: it is then dropped by the server.

In practice, the delay structure can be simplified. First, the switching fabric is usually (at least for core network routers) overprovisioned, meaning that there is no queueing time (nor loss) at the input linecard queue. Second, the first, third and fourth components of the total delay will depend only on the specific implementation of the switching fabric and linecards, and on the size of the packet. They can hence be merged in a single router-crossing time, which is a function of the router, input and output linecards, and packet size.

In [HVPD04], Hohn *et al.* monitored all the traffic going through a core network router for 13 hours, and got detailed delay statistics. They proposed a further simplified model, which was shown in their case to match the actual delay for packets. To the best of our knowledge, very few detailed studies have been conducted that validate or refute this model. In [CFS08], Chertov *et al.* proposed a more complete model for routers. However, they stated that their model performs better only for routers that are (nearly) overloaded, at the edge on the Internet. We will keep the simplified model of Hohn and his coauthors, which is as follows:

1. Fully arrived packets at the input linecard are instantaneously transported to the output linecard; hence, in particular, packets stemming from different links but going to the same output link are multiplexed according to their arrival time;
2. Packets at the linecard first experience a minimum crossing delay $\Delta_j(S)$, depending on their size $S$ and the output linecard $j$. This delay represents the time needed for a packet to cross the input line card, the switching fabric and output line card, before finally reaching the output queue. Packets cannot overtake each other at this stage, hence they might be blocked behind another packet;

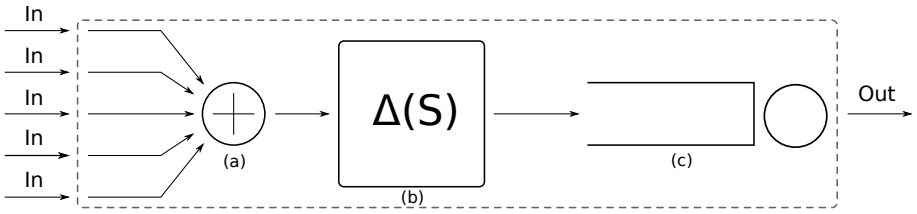



Figure 1.2: A diagram of a path in a simplified router: a multiplexer (a), an internal minimum delay (b) and an output queue (c).

3. Packets then join a fluid queue (see section 1.2 for a complete definition of queues), where they wait before being serialized. The rate of service is $\mu_j$ and the service time is proportional to the size of the packet, hence a packet of size $S$ will require $\frac{S}{\mu_j}$ time to be serialized. Any order of service (called discipline) can be considered here, but we will mostly stick to the First In First Out (FIFO) case, *i.e.* packets are served in their order of arrival.

Hence, from the point of view of a single output link (*i.e.* ignoring all traffic that does not exit the router through this link), the router can be modeled as in figure 1.2. In [HVPD04], the authors plot $\Delta(S)$ as a function of the size $S$, for the specific case of the router they monitored. The shape is roughly affine:$\Delta(S) = a + bS$, with $a = 18.8\mu$s and $b = 1.8125$ [ns/bit]. This internal delay is hence (at least in that case) of the order of dozens of microseconds: if the queuing delay or service time is larger than this order, it makes sense to ignore this internal delay as a first approximation[13].

### Traffic model

It is difficult to exhibit a "good" traffic model: traffic varies highly between different points in the networks (for example, there are many differences between the traffic on an ADSL line linking the client ADSL box to the DSLAM[14] and the highly aggregated traffic on high-speed links in the core Internet network). Moreover, traffic varies highly in time [BDF+09, HcJS03]: bandwidth increases and new applications can have significant impact. Hence, any traffic model will be valid only for a specific time span and at a specific set of network locations. However, due to the crucial need of both understanding traffic characteristics for better network management and of allowing easier simulation, a considerable effort has been devoted to this issue in the last two decades. Most of this work models traffic arrivals as a one-dimensional marked point process. This choice is natural: the traffic can be characterized by the sequence of message arrival times, with a label being added to each specifying the nature of this message (its source, destination, nature, size, etc.). This is precisely what the marked point processes model is.

[13]For comparison, on a 1 Gbps link, the service time of a 1500 bytes packet is 12 $\mu$s. The links used on the server in that experiment were had respective bandwidth of 150 Mbps, 600 Mbps and 2.4 Gbps.

[14]A DSLAM is the first router in ADSL access networks, making the connection between the ADSL line and the "classical Internet IP" network.

**Plain old telephone heritage: the Poisson assumption** Teletraffic engineering and theory started with the design and development of a circuit-based telephone service in the early $20^{\text{th}}$ century. A communication required then a dedicated circuit, and blocked call events occurred when it was not possible to book a circuit between the source and the destination of the call.

In order to better provision and manage these networks, Agner K. Erlang studied empirical traffic traces. Erlang found that call arrivals were correctly modeled by a non-homogeneous Poisson process. Retrospectively, this could be expected: new calls are generated by a large set of users, in a cross-user independent manner. If the set of users is large enough for self-dependence to be negligible, there is nearly no auto-correlation in the arrival process at most time scales, and this is a characteristic feature of Poisson point processes. Additionnally, Poisson processes, due to the independence of points or so-called memoryless property, are particularly tractable for a mathematical analysis. Hence, Poissonian models for call arrivals are still considered as valid today and used in telephonic networks models, and lead to accurate results [WP98].

**Self-similarity and long range dependence: the fall of Poisson** Due to this early work and success, Poisson processes have been an obvious choice for traffic model when the Internet and other packet-switching networks have been designed (see [Kle75]). However, contrary to telephonic networks, the actual performance of these new networks has been repeatedly below the model expectation [ENW96]. In [LTWW94] Leland *et al.* showed that Ethernet[15] traffic and packet arrivals exhibited self-similar property[16] and in [PF95], Paxson and his coauthors found a long range dependence[17] in traffic. These properties are incompatible with simple Poisson point process. This was considered as "The failure of Poisson modeling", and was the starting point of intensive research, both to understand why Internet traffic had these properties, and to find better traffic models.

**Flows, heavy tails, mice and elephants** A brief return to the physical meaning of traffic is now useful. How do the Internet and other packet-switching networks work ? Old telephonic networks were circuit based: a physical circuit (with relays) is booked between both end-hosts for each call, and physical coding-decoding happens at end-hosts. As we have seen in section 1.1.3, packet-switching networks have a different approach and divide messages into (smaller) packets. There are hence two levels of viewpoints for the network

[15] Without going into details, Ethernet is one of the main technologies for the transport of messages and the share of the media on the physical layer. This self-similar property is not particular to Ethernet, and was found on many other traces in the following years.

[16] Without going into details which are of little relevance for this dissertation, a random process is said self-similar if it is similar to a part of itself (in a fractal-like way). In mathematical notations, a process $X_n$ is self-similar if for all $m$ and $k$, $\frac{1}{m^H}(X_{km+1}+\ldots+X_{km+m})$ has the same distribution as $X$ for some parameter $H$. In contrast to self-similarity, Poisson processes are known to "smooth" when one zooms out.

[17] Unformally, a process $X(t)$ is said to have a long-range dependence if the value of $X(t)$ has a "significant impact" in the future for a long time. Formally, long-range dependence processes are characterized with an autocorrelation function $\mathbb{E}[X(t)X(t+\tau)]-\mathbb{E}[X(t)]\,\mathbb{E}[X(t+\tau)]$ whose decrease is slower than exponential in $\tau$.

traffic: the packet-level makes sense for the routers and links, which are aware only of packets, and ignore to which "message" they belong. But from the application (or end-host) point of view, packets do not really matter, and the whole message must be considered. For better understanding, it is hence useful to try to classify together packets which belong to the same message, although the network is unaware of it. That is what the notion of *flow* does. There is no single definition of flows, but we will use here a common definition in the literature: a flow is the set of packets which share common source address and port, common destination address and port, and common transport protocol (TCP or UDP), with no interarrival time greater than a threshold (usually, of the order of a dozen seconds).[18] Our definition is network-centric (it uses information that is available to any network router), but it is useful to imagine flows as the set of many different packets forming the same application message, although this is not strictly equivalent.

A crucial discovery was that Internet flows have a heavy-tailed[19] size ([Pax94] for Telnet, NNTP, SMTP and FTP flows, [CB97] for HTTP flows). In [CB97], Crovella and Bestravos found that Internet files size distribution is also heavy-tailed, which explains why flow sizes have a heavy-tailed distribution. This is often referred to as the *elephant and mice phenomenon*, meaning that most flows are really short (mice), but most of traffic comes from very few large flows (elephants).

In fact, phone calls most likely also have no exponentially distributed length. However, it has been shown ([Tak62]) that the actual performence (*e.g.* the blocking probability) of plain old telephone service (or circuit-based networks) is insensitive to the precise distribution of call durations. Only their mean matters. The same results does not hold for packet-switching networks (*e.g.* Internet), where these few huge flows will have a long range impact on the system. In opposition to the previously assumed exponential distribution with a nice memoryless property, this was empirically explaining why Internet traffic was not Poisson. Indeed, it can be shown that the superposition of many independent ON-OFF sources[20] with a heavy-tailed ON distribution leads to self-similar long range dependent traffic [WTSW97, CB97, BMSV00]. The ON-OFF source model refers here to the fact that an end-host sends (or receives) a (set of) message, then is idle for some time, until the next connection is opened. ON periods corresponds to the transmission periods, with a message fragmented in several packets.

[18] RFC 2722 defines traffic flow as "an artificial logical equivalent to a call or connection." RFC 3697 defines traffic flow as "a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label as a flow. A flow could consist of all packets in a specific transport connection or a media stream." Note that our definition is in fact similar but not equivalent to the previous definitions, nor to the definition of flows as the set of packets belonging to the same message: one client could download several web pages or files simultaneously from the same server.

[19] A random variable $X$ is said heavy-tail if the probability of having arbitrarily large values decreases slowly. Formally, $X$ is heavy-tail if $\mathbb{P}(X > \tau)$ is a slower than exponential decreasing function for large $\tau$.

[20] An ON-OFF source is a source that alternates between constant rate emission (ON periods) and silence (OFF periods). The simplest model assumes that the ON periods are independent and identically distributed (i.i.d.), that OFF periods are also i.i.d. (with a potentially different distribution), and that ON and OFF periods are independent.

**Looking for Poissonianity: human user and sessions** Even if the packet arrivals are not distributed according to a Poisson process, there are still two reasons to look for some Poissonianity in the arrival process. First, Poisson processes and their variants are remarkably amenable from a mathematical point of view, and hence provide of a natural first choice for a model. Second, and more importantly, Internet traffic is *in fine* generated by human impulses, and the human impulses are often by nature independent [PSHC+06]. Hence, the question is more to find the right scale for Poissonian impulses to appear.

Poisson clusters processes (also called Bartlett-Lewis processes) are a natural choice when looking for Poissonianity in a point process which is not Poisson. They consist of seeds, distributed according to a Poisson process. Each seed then expands into a cluster point process, which can have any independent and identically distributed (i.i.d.) distribution: a classic choice for these in-cluster processes is to take a renewal process with an independent random number of points. In [HVA03], Hohn *et al.* showed that interaction between flows is not significant. They hence proposed a model with a Poisson flow arrival process, a Gamma renewal in-flow packet process and a heavy-tailed per flow number of packet distribution. In particular, they exhibited a bi-scaling phenomenon. At small (sub-second) time scale, the packet arrival process is nearly Poissonian (see [KMFB04]), has little correlation (which is due to in-flow structure caused by feedback of TCP (see [JD05])), and is mostly characterized by the number of competing flows and their rates. At large time scale, a long range dependence effect is prevailing, originated by the heavy-tail nature of the per flow packet number distribution. This model was validated with real data in [HVY05], and its properties have been explored in [FGAMS06].

This model, whilst natural and fitting the data, failed to explain why flow arrivals also exhibit a long range dependence [RCD+09, PSHC+06]. A first physical reason can be proposed to explain why flows are not the right time scale for Poissonianity: first, files are often transferred in batches[21], leading to several simultaneous or close flows. In fact, flows are not generated only by human impulses, and Poisson processes fail to reproduce this fact. However, in [PSHC+06], it was shown that even web document downloads (*i.e.* all the flow exchanges needed for a single webpage), which are *in fine* generated by human impulses, do not follow a Poisson process. There is a human correlation between web page requests: web surfers often request several web pages (or files) simultaneously, and then are idle (from the network point of view) during the time needed to read (or analyze) these documents. The correct time scale to look for Poissonianity is in fact the user session. Similar to how flows group packets, a session is the set of flows initiated by the same user with inter-arrival time lower than a threshold. Park *et al.* propose in [PSHC+06] a threshold between 12s and 30s.

**Three different time scales, and a basic approximation** From these works, we must distinguish three different time scales. At a large time scale, a long range effect is prevailing, paired with the self-similarity property. The real traffic does not smooth out as fast as

[21] To name a few reasons: protocols often rely on exchange of several messages, and webpages typically consist of more than one object.

Poisson traffic when the time scale is increased. At the opposite, on a very small time scale, the traffic is also not Poisson. Real traffic is highly bursty at a very small time scale, and Poisson models are not adapted[22]. In an intermediate range, from dozens of milliseconds to a few minutes, it seems that the traffic is somehow more similar to Poisson. This timescale is large enough for the precise synchronisation of packet arrivals and their corresponding size to be of little effect. On the other side, most of the traffic at this time scale is characterized by the number of competiting flows (or messages) being exchanged at that time, and the size of these flows has little impact at this time scale. In other words, the total number of mice and elephants is more important than the size (*i.e.* mice or elephant specie) of each animal.

Similarly, the addressing, localizing and routing functionalities of the network need in practice the exchange of messages between different network elements, which will interfer with data packets. But on this intermediate time scale, their effect can nearly be ignored, and we will assume that they are provided in a transparent manner.

Finally, although we are aware of no recent work on this subject, the growth of streaming applications can be imagined to increase the Poisson nature of the traffic. Whilst the heavy-tail size distribution is valid for both type of traffic, streaming applications tend to send their message in a more fluid regular way. Elastic applications mostly use TCP, which is in some aspect "aggressive", increasing repetitively its sending rate until it experiences losses, and creating hence more variation in the traffic rate.

## 1.2 Queueing theory: a microscopic model for networks

Queueing theory is the mathematical study of waiting lines, or *queues*, which have to be understood as classic "real-life" queues that one experiences everyday, when waiting at the medical office, in any shop or at a taxi station (at least in certain countries). It aims at deriving and computing performance metrics of queueing systems, such as the average waiting time in the queue, the expected number of customers when a new customer arrives, the stability of the system or the distribution of the total time spent in the system. Typical incarnations of queueing systems include obviously plain old telephone service and packet-switching networks (such as the Internet), but also healthcare emergency rooms, factories or call-centers. Since the pioneering work of Erlang in 1909 [Erl09], it has been an active research field, with numerous results and publications. We do not aim here at covering it exhaustively: this would require a whole book, or even more. However, we present here

[22]There is no consensus for this effect, but several origins have been proposed. First, at a packet level, the packet arrival times are obviously not independent of their size: for example, two packets can not be closer than the service time of the first packet. Another proposition is the rate control mechanism of TCP, which induces a self-clocking mechanism. TCP flows sharing the same bottleneck tend to share the loss periods, and hence decrease their rate in a correlated manner and then slowly increase them. Finally, a third candidate is the merging at the router of different link traffic flows, and the discrete nature of packet sizes. Link traffic is composed of back-to-back packets period, followed by an idle period. The precise effect of merging several such flows at a router is not well understood, but it is obvious that because packets size distribution is not exponential but trimodal (with many packets at the maximum size, many at the minimum size, and some in a intermediate range around 600 bytes), these back-to-back and idle period are far from Poisson.

briefly the key results that are needed in this dissertation. Queueing theory is of particular interest for us because it models the behaviour of the system at the scale of customers (we will also use interchangeably the terms of tasks, jobs or packets). This scale is natural when dealing with packet-switching networks such as the Internet. We refer to [Kle75, BB03, Kel79] for a more complete survey.

### 1.2.1 A single queue

The simplest queueing system is a single queue.

**Definition 1.2.1** (Queue)**.** A queue is defined by the following elements:

- A packet arrival process, *i.e.* an increasing sequence $(t_n)_{n\in\mathbb{Z}}$ where $t_n$ denotes the arrival time of the n$^{\text{th}}$ customer;
- A service requirement distribution, *i.e.* a sequence of non-negative real values $(\sigma_n)_{n\in\mathbb{Z}}$, where $\sigma_n$ is the service time required by the n$^{\text{th}}$ customer before leaving the queue. Alternatively, the service is specified in size (and not time), and the capacity or speed of the server is added. The service time of a packet is then its size divided by the speed of the server;
- A service discipline, determining which packet(s) of the queue is (are) served when there are several packets in the queue;
- A buffer size, indicated the maximum number of packets (in number, or sometimes in total size) that can be stored in the queue, before additional packets are lost. When this is not specified, it is assumed to be infinite, *i.e.* no packet is lost by the queue;
- A number of servers, indicating how many packets can be served simultaneously and independently; this is usually assumed to be 1, unless otherwise specified.

Many disciplines can be imagined, and the proper definition of a discipline is somewhat technical. However, many usual disciplines can be easily understood.

**Example 1.2.1** (Common disciplines)**:** We will list here the most common disciplines:

1. First In First Out (FIFO) discipline: the server serves its customers in their order of arrival. This is the most common discipline in human waiting queues (at shops, taxi stations, etc.). This is also the most common discipline in Internet routers, and is assumed by default unless otherwise specified;
2. Last In First Out (LIFO) discipline: when a packet leaves, the last packet to have arrived is served. This happens in the case of "stacks" (*e.g.* for washing-up or for bills): the last element is on the top of the stack, and is the first one the be dealt with;
3. Processor Sharing (PS) discipline: the server capacity is evenly split among all the customers, *i.e.* for all the customers remaining, remaining services decrease at rate $\frac{1}{n}$ when there are $n$ customers in the queue;

4. Uniform Random discipline: the next customer is chosen uniformly at random among the waiting customers;

5. Priority discipline: the customers are divided in two (or more) different classes. When a new customer is to be served, it is chosen first in the highest priority class. If no packet belongs to the high priority class, then a customer of a lower priority class can be served. This can be combined with any in-class priority discipline. Emergency medical services apply a similar discipline, with classes depending on the degree of emergency of each patient;

6. Preemptive Priority discipline: this is the same as above, but a higher priority customer can interrupt the service of a lower priority customer[23]. Hence, low priority customers can be served only when there is no high priority customer. In case of interruption, it must be specified whether the interrupted service is resumed later (*i.e.* no service is lost), or started afresh (*i.e.* the interrupted service is lost);

7. Shortest Remaining Processing Time (SRPT) discipline: this is a preemptive discipline which serves the packet with the shortest remaining service time. When a new packet arrives, it is served immediately if the server is idle or if the current packet has a longer remaining service time. In the later case, the previously served packet is put back at the head of the queue (its remaining service time is obviously shorter than any other packet of the queue) , and its service will be resumed later. If the arriving packet has a longer service time than what remains for the packet being currently served, the new packet is put at the right place in the queue. This discipline is known to minimize the mean waiting time per packet and the mean number of packets in the queue, but is highly unfair for large packets.

We will call *conservative* any discipline which does not "lose" any service. This includes all non-preemptive disciplines, but also all preemptive disciplines which later resume the interrupted service at the time of interruption (in contrast to fresh restart).

The Kendall notation, proposed by D. G. Kendall, allows for a compact description of most queues. It reads as $A/S/N_s/B/D$, where $A$ denotes the arrival process, $S$ the service requirements, $N_s$ the number of servers, $B$ the capacity of the system, or the maximum number of customers allowed in the system including those in service, and $D$ the discipline. $B$ and $D$ are often not specified, and have then default value of $\infty$ and FIFO. The arrival process is specified with the distribution of inter-arrival times. A few letters cover most classical distribution:

- M denotes an exponential distribution;

- D denotes a deterministic distribution, *i.e.* a fixed value;

[23] Other preemptive disciplines obviously exists, such as preemptive LIFO discipline. We do not present them here for simplicity.

- GI means any i.i.d. distribution;
- G covers any other case.

Hence, an M/M/1 queue denotes a queue with Poisson arrival process of intensity $\lambda$ (the interarrivals of a point process are exponentially i.i.d. if and only if the process is a Poisson stream), exponentially distributed service times (of mean $\frac{1}{\mu}$), a single server and an infinite buffer. The GI/G/K/K+2/PS queue is a queue with a renewal arrival process, any service distribution, K servers, a buffer limited to only 2 packets and a processor-sharing discipline.

The next example presents a typical result of queueing theory. It was shown in 1917 by Agner K. Erlang [Erl17], and is furthermore considered to be one of the very first queueing theory results.

**Example 1.2.2** (Erlang blocking formula)**:** Consider an M/M/K/K queue, *i.e.* a queue where customers arrive according to a Poisson point process of intensity $\lambda$ and have i.i.d. exponentially distributed service times of mean $\mu^{-1}$. Up to $K$ customers can be served simultaneously, but no customer can be queued. New customers arriving when all the $K$ servers are busy are rejected and lost forever. What is the proportion of rejected customers, or equivalently, what is the probability for a new customer to be rejected?

This probability is called the blocking probability, and has interesting application in industrial systems. Before presenting its proof, let us spend a few lines to explain why this is an meaningful quantity to estimate. Erlang was an engineer and mathematician at the Copenhagen Telephone Exchange. He had already shown that calls were initiated according to a Poisson process, and that calls duration could be approximated with an exponential random variable [Erl09]. In Plain Old Telephone Service, a call required booking a circuit between both end points. If the service exchange has a capacity of $K$ circuits, the blocking probability is exactly the probability for a new call to be rejected because all circuits are busy. Hence, this quantity is one measure of the performance of the telephone network, and can be used to estimate the eventual gain in performance of increasing the service exchange capacity.

Erlang established that the blocking probability $P_B$ is

$$P_B = \frac{\frac{\rho^K}{K!}}{\sum_{i=0}^{K} \frac{\rho^i}{i!}} \quad .$$

To prove this result, let $N(t)$ denote the number of calls being served at time $t$. $N(t)$ is a birth-and-death process. The birth rate $\lambda_i$ when the call population is $i$ is:

$$\lambda_i = \begin{cases} \lambda & \text{if } 0 \leq i < K \\ 0 & \text{if } i = K \end{cases} \quad .$$

The death rate $\mu_i$ is $\mu_i = i \times \mu$.

Indeed, just after a new arrival, the inter-arrival is distributed with an exponential distribution of parameter $\lambda$. But due to the memoryless property of exponential distribution, this

is also the case when a call leaves the system. If the current population is $K$, this new call will be rejected, and the birth rate is hence 0.

Similarly, at any time, the residual service time of any call is an exponential random variable of parameter $\mu$, thanks to the same memoryless property of exponential random variables. The next departure happens at the minimum of the residual service times. As the minimum of $i$ i.i.d. exponential random variables of same parameter $\mu$ is an exponential random variable of parameter $i \times \mu$, we get the death rates.

Denoting by $\rho = \frac{\lambda}{\mu}$ the load of the system and by $\boldsymbol{\pi} = (\pi_0, \pi_1, \ldots, \pi_K)$ the stationary distribution of the population $N(t)$, we have from the detailed balance equations that $\lambda\pi_{i-1} = i\mu\pi_i$, for $1 \leq i \leq K$, and hence:

$$\forall 0 \leq i \leq K, \qquad \pi_i = \frac{\rho^i}{i!}\pi_0 \quad .$$

By normalisation, we get that

$$\pi_i = \frac{\frac{\rho^i}{i!}}{\sum_{j=0}^{K} \frac{\rho^j}{j!}} \quad .$$

Calls arrives according to a Poisson process, and see the system at the equilibrium state. They are blocked if the population is already $K$. Hence, the blocking probability is $\pi_K$.

Erlang's formula is a typical (and historical) example of the many fruits of queueing theory. It is today still widely used, for example in the case of inventory stocks and lost sales.

### 1.2.2 The M/M/1 queue

In this section, we will present succinctly a few results about the M/M/1 queue. Recall that an M/M/1 queue is the queue with Poisson arrivals, i.i.d. exponential service times, 1 server, an infinite buffer and a FIFO discipline. Throughout this section, $\lambda$ will be the intensity of the arrival process and $\mu$ the inverse of the mean service time.

**Stability and steady-state population distribution** The first result we establish is whether the population of the queue diverges or converges to a steady-state distribution.

**Proposition 1.2.2** (Stability and population distribution)**.** *Let $N(t)$ denote the total population of the queue at time t, and $\rho = \frac{\lambda}{\mu}$ its load. Then:*

1. *If $\rho < 1$, then $N(t)$ admits a single state steady distribution;*
2. *If $\rho \geq 1$, $N(t)$ admits no steady state distribution.*

*Additionally, in the first case, the steady state distribution $\boldsymbol{\pi}$ will be:*

$$\pi_k = \mathbb{P}\left(N(t) = k\right) = (1-\rho)\rho^k \quad . \tag{1.1}$$

*Proof.* Let $Q$ denote any steady-state measure of $N(t)$, and consider $N(t)$ at the arrival times. The inter-arrival time is an exponential random variable of parameter $\lambda$. Due to the memoryless property of exponential distribution, the residual service time is also an exponential random variable, of parameter $\mu$. Similarly, at departure time, the next packet service time is an exponential r.v. of parameter $\mu$, and the time to next arrival is still exponentially distributed with parameter $\lambda$ (thanks to the same memoryless property).

Hence, $N(t)$ is a birth and death process with birth (resp. death) rate $\lambda$ (resp. $\mu$), and for all non-negative $k$, we have $\lambda Q(k) = \mu Q(k+1)$. The following then holds:

$$\forall k \geq 0, \quad Q(k) = \rho^k Q(0) \quad . \tag{1.2}$$

If there exists any steady-state distribution $\boldsymbol{\pi}$, its mass is 1, and from (1.2), it satisfies

$$\pi_0 \times \sum_{k \geq 0} \rho^k = 1 \quad .$$

When $\rho \geq 1$, as the series $\sum_{k\geq 0} \rho^k$ diverges, there is no steady-state distribution. When $\rho < 1$, we have $\sum_{k\geq 0} \rho^k = \frac{1}{1-\rho}$, and hence (1.1). □

From (1.1), we can find that the mean number of customers in the queue is

$$\overline{N} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda} \quad . \tag{1.3}$$

For birth-and-death processes, the global balance equations are equivalent to the detailed balance equation, and hence, any stationary birth-and-death process, including the M/M/1 queue when $\rho < 1$, is reversible. This leads to the following theorem:

**Theorem 1.2.3.** *The departure of an M/M/1 queue form a Poisson point process of intensity $\lambda$. Moreover, the state of the queue $N(t)$ at time t is independent of the departure process prior to time $t$.*

*Proof.* Both parts are direct application of the reversibility of the M/M/1 queue. Indeed, the departures in the forward process correspond to the arrivals in the reversed process. They hence have the same distribution, which is by assumption a Poisson point process of intensity $\lambda$. Similarly, the state at time $t$ has the same distribution as the state of the reversed process at time $-t$, and the departures prior to time $t$ in the direct process correspond to arrivals in the reversed process past time $-t$. It is enough to conclude by realizing that in the reversed process, the state at time $-t$ is obviously independent of arrivals past time $-t$. □

The departure process is the mixture of a Poisson process of intensity $\mu$ when the queue is busy, and null when the queue is empty. Theorem 1.2.3 shows that despite this "bi-scaling" nature, the output of the queue has the same shape as the input, when the latter is Poisson. This will allows us to easily build networks of queues in section 1.2.3. The second

part might seem of little interest for the moment. It will however be crucial in section 1.2.3 to compute the steady-state distribution of such networks.

**Continuous-time process and discrete-time chain** Up to now, we have considered the continuous-time Markov process $N(t)$. It expresses which state the Markov process is in at time $t$, and its steady state distribution expresses the fraction of time that the process spends in any state. However, we might have a special interest in the queue at times when a packet arrives, since these are the times which will determine what packets experiences. Is the distribution identical ? Let$(N_n)_{n\in\mathbb{N}}$ be the discrete time Markov chain where $N_n$ is the value of $N(t)$ just before the arrival time of the $n^{\text{th}}$ packet. Because the arrival process is Poisson, it verifies the *Poisson Arrivals See Time Average* (PASTA) rule, and the distribution of the queue size $N_n$ just before the $n^{\text{th}}$ packet arrives is also:

$$\mathbb{P}(N_n = k) = (1-\rho)\rho^k \quad .$$

Due to reversibility, the same holds for the Markov chain at times just after the departure of a packet.

Finally, we must quickly mention that the *embedded Markov chain* of the process $N(t)$ does not have the same distribution. The embedded Markov chain of a continuous-time Markov process is the Markov chain obtained by observing the process just after any jump. Compared to the continuous-time process, it focuses only on the sequence of states, and does not include any information about how much time the process spends in each state. Its steady state distribution expresses fraction of the *jumps* which go into a specific state. Because the *holding time* in a given state can depend on the state, this is not equivalent to the continuous-time process. Here, when the jump is a departure, we have seen that stationary distribution after the jump is the same as the stationary continuous-time distribution. However, when the jump is an arrival, we know that the distribution just *before* the arrival is also identical to the continuous-time distribution. Hence, after the arrival, we do not have the same distribution (*e.g.* there is no chance of having no customer in the system, since one just arrived), and the embedded Markov chain does not have the same distribution.

Note that up to now, the results we have presented did not assume that the discipline was FIFO. In fact, they hold for any conservative discipline independent of the service times. The discipline will be important when we study the delay of individual packets.

**Delay and waiting-time** How long does a packet need to wait before being served? How much time elapsed between its arrival and its departure from the queue? These questions are natural when trying to predict the performance of a queue.

In this dissertation, we will call the waiting time the time spent by a packet in the queue before its service start. The delay[24] will be the total time spent in the queue, including the

[24]Other terms are also used in the literature. In particular, the delay is sometimes called system time or sojourn time.

service time. Hence, the delay is equal to the waiting time plus the service time.

Using the previous result on the mean number of packets in the system, it is easy to get the mean waiting time and delay of an M/M/1 queue. Indeed, the service times of packets waiting to be served are i.i.d., with mean $\overline{\sigma} = \frac{1}{\mu}$. Additionally, at any customer arrival (or at any time), the remaining service time of the currently served packet has the same distribution, thanks to the memoryless property of exponential distribution. Hence, assuming from this point that the discipline is FIFO[25], the mean waiting time $\overline{W}$ is:

$$\overline{W} = \sum_{k=0}^{\infty} P(k) \times k \times \overline{\sigma} = \overline{N} \times \overline{\sigma} = \frac{\rho}{\mu - \lambda} \quad . \tag{1.4}$$

To compute the mean delay $\overline{D}$, we just need to add the service time of the packet, which is independent of the number of packets in the queue at arrival time and the service times of these packets. Hence, we also have that

$$\overline{D} = \overline{W} + \overline{\sigma} = \frac{1}{(\mu - \lambda)} \quad . \tag{1.5}$$

Using the distribution of the queue size at packet arrivals, it is also possible to compute the whole queuing time and delay distributions.

The probability density function (p.d.f.) $f_W(t)$ of the waiting time can be computed by conditioning on the number of packets $N(0^-)$ in the queue before arrival. The waiting time is either 0 if $N(0^-) = 0$, or the sum of $k$ i.i.d. exponential of parameter $\mu$, *i.e.* a gamma distribution random variable of parameters $(k, \mu)$ if $N(0^-) = k$. Hence, we have:

$$\begin{aligned}
f_W(t) &= \sum_{k=0}^{\infty} f_W\left(t | N(0^-) = k\right) \mathbb{P}\left(N(0^-) = k\right) \\
&= (1-\rho)\delta_0(t) + \sum_{k=1}^{\infty} \frac{\mu^k t^{k-1} e^{-\mu t}}{\Gamma(k)} (1-\rho)\rho^k \\
&= (1-\rho)\delta_0(t) + (1-\rho)e^{-\mu t} \sum_{k=1}^{\infty} \frac{\lambda^k t^{k-1}}{(k-1)!} \\
&= (1-\rho)\delta_0(t) + \rho(\mu-\lambda)e^{-\mu t} \sum_{k=0}^{\infty} \frac{(\lambda t)^k}{k!} \\
f_W(t) &= (1-\rho)\delta_0(t) + \rho(\mu-\lambda)e^{-(\mu-\lambda)t} \quad ,
\end{aligned} \tag{1.6}$$

where $\delta_0(t)$ denotes the Dirac function centered at $t = 0$.

This can be summarized as follows: the waiting time distribution for a typical packet in an M/M/1 queue is the mixture of an atom at 0 with probability $1 - \rho$, and an exponentially distributed random variable of parameter $\mu - \lambda$ with probability $\rho$.

[25] In fact, the mean delay and waiting time formula hold for any conservative discipline, but they are more complicated to establish when the discipline is not FIFO. The delay and waiting time *distributions* given here are valid only for FIFO discipline.

Similarly, the p.d.f. $f_D(t)$ of the delay of a new packet can be computed as follows:

$$\begin{aligned} f_D(t) &= \sum_{k=0}^{\infty} f_D\left(t|N(0^-)=k\right)\mathbb{P}\left(N(0^-)=k\right) \\ &= \sum_{k=0}^{\infty} \frac{\mu^{k+1}t^k e^{-\mu t}}{\Gamma(k+1)}(1-\rho)\rho^k \\ &= (1-\rho)\mu e^{-\mu t}\sum_{k=0}^{\infty}\frac{\lambda^k t^k}{k!} \\ f_D(t) &= (\mu-\lambda)e^{-(\mu-\lambda)t}. \end{aligned} \tag{1.7}$$

Hence, the delay is an exponential random variable of parameter equal to the residual bandwidth $\mu - \lambda$.

### 1.2.3 Network of queues

We have currently presented only models of a single queue. How can one combine queues in order to build a network? We will present here 3 models for network, each model generalizing the previous one.

#### Queues in series

We will first present the notion of queues in tandem, that is on a line, where each queue departure process is the arrival process of the next queue. Hence, packets arrive in the system only at the first queue, and leave the system only after the last queue. An example of such a network is given in Figure 1.3.

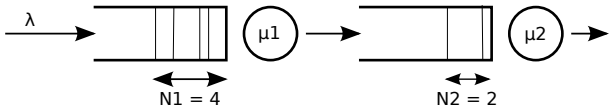



Figure 1.3: Two queues in tandem.

More formally, consider a series of $K$ single-server queues. Assume that all packets at queue $i$ require an i.i.d. exponential service time of parameter $\mu_i$ (hence, the average capacity of queue $i$ is $\mu_i$ packets per second), and that these service times are independent for different queues. Assume that the first queue see an "external" arrival process intensity $\lambda$. Assume finally that when a packet leaves queue $i$, it immediately enters the next queue $i+1$, unless $i = K$, in which case it leaves the system. Let $N_i(t)$ denote the number of packets in the queue $i$ at time $t$, and $\boldsymbol{N}(t) = (N_1(t), \ldots, N_K(t))$ be the state of the whole system. What is the distribution of $\boldsymbol{N}(t_0)$ at a given time $t_0$? A network of M/M/1 queues in tandem is a *product form* network, because its steady-state distribution is equal to the product of the steady-states distribution of each queue, as it is shown by the following theorem:

**Theorem 1.2.4.** *Let $\boldsymbol{N}(t)$ denote the state at time $t$ of a system of a series of $K$ single-server queues, with exponentially distributed independent service times with rates $(\mu_1, \ldots, \mu_K)$, and Poisson arrivals of rate $\lambda$ at the first queue. Let $\rho_i = \frac{\lambda}{\mu_i}$ denote the load of the queue $i$. If $\exists i, \rho_i \geq 1$, then $N(t)$ admits no steady-state distribution. Otherwise, $N(t)$ admits a steady state distribution $\pi$, with*

$$\pi(n_1, \ldots, n_K) = \prod_{i=1}^{K} (1 - \rho_i)\rho_i^{n_i} \quad .$$

*Proof.* Consider the first queue only. This is an M/M/1 queue with arrival rate $\lambda$ and service rate $\mu_1$. Hence, if $\lambda \geq \mu$, this queue, and hence the whole system, do not admit any steady state distribution. If $\lambda < \mu$, the previous results apply here, and we have that $N_1(t_0)$ admits a geometric steady-state distribution of parameter $\rho_1 = \frac{\lambda}{\mu_1}$.

Using the first part of theorem 1.2.3, we know that the output of queue 1, and hence the input of queue 2, is a Poisson process of intensity $\lambda$. Hence, queue 2 is also an M/M/1 queue, and $N_2(t_0)$ admits a geometric steady-state distribution of parameter $\rho_2 = \frac{\lambda}{\mu_2}$, provided that $\lambda < \mu_2$. If $\rho_2 \geq 1$, the second queue, and hence the whole system admits no steady-state solution. The similar reasoning extends recursively for all queues.

However, before concluding for the whole distribution, we must prove that $(N_1(t_0), \ldots, N_K(t_0))$ are independent random variables, or study their correlation. Using the second part of theorem 1.2.3, we know that $N_1(t_0)$ is independent of the departure of queue 1 prior to $t_0$. But $(N_2(t_0), \ldots, N_K(t_0))$ depends only on the arrivals in queue 2 (or departures from queue 1) prior to $t_0$ (and on their service requirements). Hence, $N_1(t_0)$ is independent of $(N_2(t_0), \ldots, N_K(t_0))$. Using the same recursion, we can show that $N_2(t_0)$ is independent of $(N_3(t_0), \ldots, N_K(t_0))$, and hence, the steady-state distribution of $\boldsymbol{N}(t)$ is the product of the steady-states distributions of $N_1(t), \ldots, N_K(t)$. □

These line networks can easily be generalized to any acyclic topologies where every queue has a single output, including possible external arrivals at different queues. This leads to tree topologies, where the packets flow from the leaves down to the root. It relies on the fact that the superposition of Poisson point processes is still a Poisson point process. The acyclic assumption implies that Theorem 1.2.3 can be applied recursively to queues where all arrivals are Poisson point processes, starting from queues with external arrivals.

Theorem 1.2.4 does not assume that the queue disciplines are FIFO. It holds for any discipline which leads to the single queue steady-state distribution and theorem 1.2.3, that is any conservative independent of the service times discipline. When the disciplines are such that no packet can overtake other packets, it is possible to compute the delay distribution of packets. This is done for the specific cases of FIFO discipline in chapter 3 for line-shaped networks and chapter 4 for tree-shaped networks.

### Jackson network

Networks of queues in tandem can not deal with any topology including a loop, or with different destinations for packets in the same queue. We present here a more general class of networks, called Jackson networks or open migration networks, which covers all possible topologies.

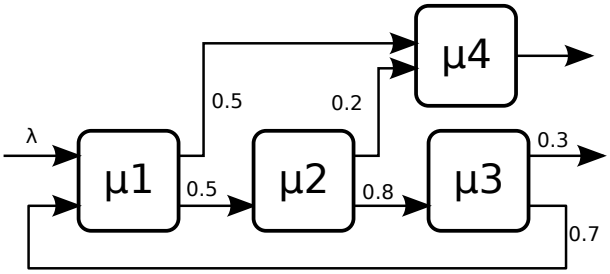



Figure 1.4: An example of Jackson network.

Consider a set of $K$ queues. Packets in queue $i$ have i.i.d. exponential size of mean 1, and are served at a global rate of $\mu_i$[26]. When leaving queue $i$, each packet has a probability $p_{i,j}$ to join queue $j$, and probability $p_{i,0} = 1 - \sum_j p_{i,j}$ to exit the system. We can assume without loss of generality that $p_{i,i} = 0$. Packets arrive from outside to queue $i$ according to a Poisson point process of rate $\lambda_i$. We shall require additionally that there be a path of positive rates from any queue to an exit, either directly or indirectly through other queues. We shall assume also, without any loss of generality, that there is a path to any queue from an external arrival (otherwise this queue will receive no new customer, and after some time, be endlessly empty). Queues in tandem are a Jackson network, with $p_{i,j} = 1$ if and only if $j = i + 1$ or $i = K$ and $j = 0$, and $p_{i,j} = 0$ otherwise.

**Proposition 1.2.5** (Total arrival rates in Jackson networks)**.** *Given a Jackson network of $K$ queues, with services rates $\mu_i$, external arrival rates $\lambda_i$, and transition probabilities $p_{ij}$ from queue $i$ to queue $j$, there exists a unique vector $\tilde{\boldsymbol{\lambda}} = \left(\tilde{\lambda}_1, \ldots, \tilde{\lambda_K}\right)$ of positive weights, such that*

$$\forall 1 \leq i \leq K, \quad \tilde{\lambda}_i = \lambda_i + \sum_{j=1}^{K} \tilde{\lambda}_j p_{j,i} \quad . \tag{1.8}$$

*This vector represents the total arrival rate in the queues.*

*Proof.* Consider a continuous-time Markov process of $K + 1$ states $\{0, 1, \ldots, K\}$, with transition rates $Q = (q_{i,j})_{0 \leq (i,j) \leq K}$, where $q_{0,i} = \lambda_i$ and $Q_{i,j} = p_{i,j}$. This process is irreducible (there is a path from any state to 0 and from 0 to any state) and time-homogeneous, with a finite state space. It hence admits an unique steady-state distribution $\pi$, which verifies

[26] We do not specify here how this global rate is shared among the packets, or, more generally, the discipline. It does not matter, as long as the discipline is conservative. It is also possible to let the global rate vary with the queue size, but for the sake of simplicity, we consider here constant rate: the generalization is straightforward.

the equilibrium equations:

$$\forall 0 \leq i \leq K, \quad \pi(i) \sum_{j=0}^{K} q_{i,j} = \sum_{j=0}^{k} \pi(j) q_{j,i} \quad .$$

Dividing each equation by $\pi(0)$ and replacing the $q_{i,j}$ by their value, we get the equations (1.8), and hence the existence and uniqueness of the solution $\tilde{\lambda_i} = \frac{\pi(i)}{\pi(0)}$ in the proposition.

By definition, the arrival rate in queue $i$ is the sum of the external arrival rate $\lambda_i$ and the arrival rate from other queues $j$. The total arrival rate of queue $j$ is $\lambda_j$, and a proportion $p_{j,i}$ of it go to queue $i$ just after leaving queue $j$. Hence, $\tilde{\lambda}_i = \lambda_i + \sum_{j=1}^{K} \tilde{\lambda}_j p_{j,i}$ is the total arrival rate at queue $i$. □

Let $N_i(t)$ denote the state of queue $i$, *i.e.* the number of customers in queue $i$, and $\boldsymbol{N}(t) = (N_1(t), \ldots, N_K(t))$ the state of the whole networks. Knowing the total arrival rates of the queues in the network, it is easy to compute the steady-state $\pi$ distribution of the network:

**Theorem 1.2.6.** *Consider a Jackson network of $K$ queues, with services rates $\mu_i$, external arrival rates $\lambda_i$, and transition probabilities $p_{ij}$ from queue $i$ to queue $j$. Let $\tilde{\boldsymbol{\lambda}} = \left(\tilde{\lambda}_1, \ldots, \tilde{\lambda_K}\right)$ be the total arrival rate vector, as defined in proposition 1.2.5. Let $\tilde{\rho}_i = \frac{\tilde{\lambda}_i}{\mu_i}$ be the total load of queue $i$, and assume that $\tilde{\rho}_i < 1$ for all $1 \leq i \leq K$. Let $\boldsymbol{N}(t) = (N_1(t), \ldots, N_K(t))$ denote the state of the network.*

*Then the Jackson network admits an equilibrium distribution. In equilibrium, $N_1, N_2, \ldots, N_K$ are independent and*

$$\forall 1 \leq i \leq K, \qquad \pi_i(n_i) = (1 - \tilde{\rho}_i)\tilde{\rho}_i^{n_i} \quad .$$

*Proof.* It is sufficient (and straightforward) to verify that $\pi(\boldsymbol{N}) = \prod_{i=1}^{K} \pi_i(n_i)$ satisfies the partial balance equations. The independence of $N_1, N_2, \ldots, N_K$ follows from the fact that both $\pi(\boldsymbol{N})$ and the state space have a product form. □

The independence established in theorem 1.2.6 is the independence of the random variables $N_1, \ldots, N_K$, observed at a fixed point $t_0$ in time. The stochastic processes $(N_1(t), \ldots, N_K(t))$ are clearly not independent. Interestingly, the equilibrium distribution for queue $i$ in isolation is just what it would be if it were the only colony in the system, with customers arriving in a Poisson stream of rate $\tilde{\lambda}_i$ and leaving at rate $\mu_i$. This is even more interesting when one realize that, because of the eventual loops in the network, the combined arrivals at queue $i$, from outside and other queues, is in general not a Poisson process.

The process is in general not reversible. In fact, the process is reversible iff $\tilde{\boldsymbol{\lambda}}$ satisfies

$$\begin{aligned} \forall 1 \leq (i,k) \leq K, \qquad & \tilde{\lambda}_i p_{i,k} = \tilde{\lambda}_k p_{k,i} \\ \forall 1 \leq i \leq K, \qquad & \lambda_i = \tilde{\lambda}_i p_{i,0} \quad . \end{aligned} \tag{1.9}$$

However, even when (1.9) does not hold, the reversed process is of a similar form:

**Theorem 1.2.7.** *Let $\boldsymbol{N}(t)$ be a process corresponding to a Jackson network. Then the reversed process $\boldsymbol{N}(-t)$ corresponds also to a Jackson network on the same queues with the same topology. In particular, the reversed process has transition probabilities $p'_{i,k}$ and external arrival rates $\lambda'_i$ such that:*

$$\begin{aligned} p'_{i,0} &= \frac{\lambda_i}{\tilde{\lambda}_i} \\ p'_{i,k} &= \frac{\tilde{\lambda}_k p_{k,i}}{\tilde{\lambda}_i} \\ \lambda'_i &= p_{i,0}\tilde{\lambda}_i \quad . \end{aligned}$$

*Proof.* It is easy to verify that such a Jackson network process is indeed the reversed process of $\boldsymbol{N}(t)$. □

We will call the *exit* process the points in time at which customers leaves the whole network. The arrival processes in the reversed process are the exit processes in the original process. Hence, we have the following corollary, similar to theorem 1.2.3:

**Corollary 1.2.8.** *If $\boldsymbol{N}(t)$ is a Jackson network process, then the exit process from queue $i$ is a Poisson process of rate $\tilde{\lambda}_i p_{i,0}$. Additionally, the exit processes from queues $1, \ldots, K$ are independent and $N(t_0)$ is independent of the exit processes prior to time $t_0$.*

In general, the delay in Jackson networks is difficult to compute. A specific case however occurs when the topology is loopless and queues discipline are such that no packet may overtake another one. Since the thinning of a Poisson point process and the superposition of Poisson point processes are Poisson point processes, the repetitive use of theorem 1.2.3 allows us to consider each queue as an M/M/1 queue independent of the state of prior queues, and hence compute the delay in each queue.

### Kelly network

Jackson networks do not allow us to specify specific routes. When leaving a queue, each packet can go to any following queue of the network, with a probability which depends only on the topology and not on the packet. Therefore, some situations cannot be described with Jackson networks. For example, consider the example depicted in Figure 1.5. This is a networks with fives nodes. Two streams of packet arrive respectively at node 1 (with intensity $\lambda_1$) and node 2 (with intensity $\lambda_2$). Both streams then cross server 3. However, their exit point of the network is different. Packets stemming from the stream of queue 1 (resp. queue 2) go to queue 4 (resp. queue 5), and then exit the network. Whilst on average, a proportion $\frac{\lambda_1}{\lambda_1+\lambda_2}$ of the packets that leave queue 3 go to queue 4, this choice is not independent of their past. We will present here a more general model of network, introduced by F. Kelly in [Kel79], that allows such networks.

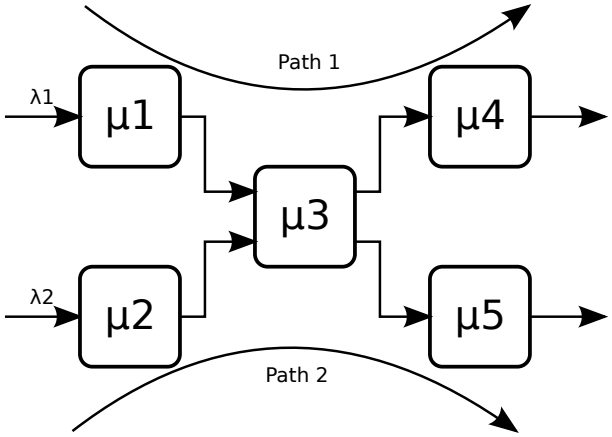



Figure 1.5: An example of Kelly network.

Assume that there are $|I|$ different type of packets, and that the packets of class $i \in I$ arrive according to a Poisson point process of rate $\lambda_i$. $|I|$ might be infinite, but the total arrival rate $\sum_{i \in I} \lambda_i$ must be finite. Packets from the same class $i$ follow a predefined finite path. Let $S_i$ denote the length of the path for class $i$, and $r(i, s)$ with $s \leq S_i$ denote the $s^{\text{th}}$ queue visited by packets of class $i$. Packets may visit the same queue several times. We will denote $r_i = (r(i, 1), \ldots, r(i, S_i))$ the path of class $i$.

This can obviously deal with any queues in series. It is also simple to see how to represent a loopless Jackson networks as a Kelly network: it is sufficient to create one class for each possible path in the Jackson network, and adapt the arrival rate of the class such that the rate of the path correspond to the arrival rate at the first queue, multiplied by all transition probabilities. If there are possible loops in the Jackson network, it is a little bit more tricky to represent it as a Kelly network: the number of visits to the same queue by a packet is not bounded, and packets in Kelly networks have a predetermined path. It is hence necessary to use an infinite number of classes, so as to represent any possible path.

**Example 1.2.3:** Consider for example the simple network depicted in figure 1.6. It's a two-server network. Packets arrive in the network at queue 1, according to a Poisson process of intensity $\lambda$. They then go to queue 2. After completion of their service in queue 2, they leave the network with probability 0.9, and go back to queue 1 with probability 0.1. How to represent it as a Kelly network? Let the class 1 have the path $r_1 = (1, 2)$, and arrival rate $\lambda_1 = 0.9\lambda$. In a more general manner, for any positive integer $i$, let the class $i$ have arrival rate $\lambda_i = 0.1^{i-1} \times 0.9\lambda$, and have a path $r_i$ of length $S_i = 2i$ where $r(i, s) = s \mod 2$. In other words, the class $i$ corresponds to packets which go exactly $i$ times through queue 1.

By construction, the sum of the arrival rates is finite, and is exactly $\lambda$. Moreover, in Jackson networks, the *external* arrival process of the packets which will visit queue 1 exactly once (*i.e.* will leave the network as soon as possible) is a thinning with probability $0.9$ of the total external arrival process. As the thinning with probability $p$ of a Poisson point process of intensity $\lambda$ is a Poisson point process of intensity $p\lambda$, traffic from class 1 in the Kelly

network corresponds exactly to these packets in the Jackson network. The same extends to packets which visit queue 1 exactly $i$ times.

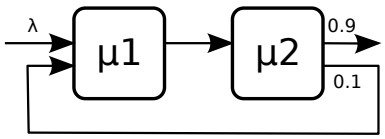



Figure 1.6: A toy example of Jackson network.

Describing the state of the network in a Kelly network is more complicated than the state of a Jackson network. In Jackson networks, all packets behave similarly, and hence, we can just count the number of packets $N_k(t)$ in each queue. Here, we have in the state to include the class of each packet. In order to simplify the notations, we will restrict ourselves here to networks of single server FIFO queues. This can be generalized to many more disciplines (even with variable queue service rates which depend on the number of packets in the queue), but we will not need it in this dissertation.

Let $t_k(l)$ (resp. $s_k(l)$) denote the class (resp. the stage) of the packet in queue $k$ at position $l$. Hence, we have in particular $r(t_k(l), s_k(l)) = k$. Let $c_k(l) = (t_k(l), s_k(l))$ denote the state of the packet at position $l$ in queue $k$. If the packet visits queue $k$ more than once, its state contains more information than its class. The vector

$$\boldsymbol{c_k} = (c_k(1), c_k(2), \ldots, c_k(n_k))$$

is the state of queue $k$, and

$$\boldsymbol{C} = (\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_K})$$

is a Markov process which describes the state of the network, where $K$ is the number of queues.

To describe the transition rates of this process, let $T_k(\boldsymbol{C})$ for $1 \leq k \leq K$ denote the new state obtained from state $\boldsymbol{C}$ with $N_k \geq 1$ when the packet currently served at queue $k$ leaves this queue, and either exits the network or joins the next queue of its path. Similarly, let $T^i(\boldsymbol{C})$ for $i \in I$ denote the new state obtained when a new packet of class $i$ enters a network of state $\boldsymbol{C}$ and joins the queue $r(i, 1)$. The transition rates $q(.,.)$ are then:

$$\begin{aligned} q\left(\boldsymbol{C}, T^i(\boldsymbol{C})\right) &= \lambda_i \\ q\left(\boldsymbol{C}, T_k(\boldsymbol{C})\right) &= \mu_k \\ q\left(\boldsymbol{C}, \boldsymbol{C'}\right) &= 0 \qquad \text{otherwise.} \end{aligned} \tag{1.10}$$

Compared to the case of Jackson networks, the total arrival rate and the load of each queue are easier to compute: the routes are deterministic, and everything depends only on the different classes. In fact, the total arrival rate at a queue $k$ is the sum of class arrival rates, multiplied by the number of times the class path goes through the queue. More formally,

one can define the total arrival rates $\tilde{\lambda}_k$ and load $\tilde{\rho}_k$ as follows:

$$\begin{aligned}\tilde{\lambda}_k &= \sum_{i\in I}\left(\sum_{s=1}^{S_i}\mathbb{1}_{r(i,s)=k}\right)\lambda_i \\ \tilde{\rho}_k &= \frac{\tilde{\lambda}_k}{\mu_k} \quad .\end{aligned} \tag{1.11}$$

The Markov process is unstable if at least one of the queues is overloaded, which corresponds to $\tilde{\rho}_k \geq 1$ for some $k$. Assuming that this is not the case, it is stable and admits a stationary distribution:

**Theorem 1.2.9.** *The equilibrium distribution $\pi$ for a Kelly networks with $K$ queues of capacities $(\mu_k)_{1\leq k\leq K}$ and packet classes of arrival rates $(\lambda_i)_{i\in I}$ and path $(r_i)_{i\in I}$ is:*

$$\pi(\boldsymbol{C}) = \prod_{k=1}^{K}(1-\tilde{\rho}_k)\frac{\prod_{l=1}^{N_k}\lambda_{t_k(l)}}{\mu_k^{N_k}} \quad .$$

*Proof.* We first prove that $\pi$ sums to unity. By definition of $\tilde{\lambda}_k$, we have that $\sum \lambda_{t_k(l)} = \tilde{\lambda_k}$ when we sum over all possible packet *states*. Hence, when considering only the queue sizes and forgetting packet states, we have that $\pi(N_1, N_2, \ldots, N_K) = \prod_{i=1}^{K}(1-\tilde{\rho}_k)\left(\frac{\tilde{\lambda_k}}{\mu_k}\right)^{N_k}$. This sums to unity because by definition $\tilde{\rho}_k = \frac{\tilde{\lambda}_k}{\mu_k}$.

The rest of the proof uses a reversed process. Consider that packets of class $i$ still enter the system as a Poisson stream of rate $\lambda_i$, but now follow a path

$$r_i' = \big(r(i,S_i), r(i,S_i-1), \ldots, r_(i,1)\big)$$

before leaving the network. This is still a Kelly network. For simplicity of the notation, we will inverse the direction of the queue: packets in the reversed process join the head of the queue, but the served packet is at the back of the queue. This is still a FIFO queue, but will remove us some burden of notation. The transition rates $q'$ of the reversed process are as follows:

$$\begin{aligned} q'(T_k(\boldsymbol{C}),\boldsymbol{C}) &= \lambda_i \quad \text{if } c_k(1) = (i,S_i) \\ q'(T_k(\boldsymbol{C}),\boldsymbol{C}) &= \mu_j \quad \text{if } r\left(t_k(1), s_k(1)+1\right) = j \\ q'(T^i(\boldsymbol{C}),\boldsymbol{C}) &= \mu_k \quad \text{if } r(i,1) = k \\ q'(\boldsymbol{C},\boldsymbol{C'}) &= 0 \quad \text{otherwise .} \end{aligned} \tag{1.12}$$

It is now straightforward to verify that for any states $\boldsymbol{C}$ and $\boldsymbol{C'}$, we have $\pi(\boldsymbol{C})q(\boldsymbol{C},\boldsymbol{C'}) = \pi(\boldsymbol{C'})q'(\boldsymbol{C'},\boldsymbol{C})$, which is enough to deduce that $\pi$ is the equilibrium distribution. □

As for Jackson networks, we have several corollaries, similar to theorems 1.2.7 and 1.2.8.

**Corollary 1.2.10.** *If $C(t)$ is a Kelly networks as described in this section, then its reversed process $C(-t)$ is also a Kelly network.*

**Corollary 1.2.11.** *In equilibrium, packets of class $i$ exit the network in a Poisson stream at rate $\lambda_i$. These Poisson point processes are independent, and $C(t_0)$ is independent of departures from the network prior to time $t_0$.*

Finally, define $\pi_k(\boldsymbol{c_k}) = (1-\tilde{\rho}_k)\frac{\prod_{l=1}^{N_k}\lambda_{t_k(l)}}{\mu_k^{N_k}}$ as the steady-state measure of the queue $k$. The network is then said to be of product form, and at a given time the steady-state distribution is as if each queue behaved independently as an M/M/1 queue. Note that this independence is valid only at a given time, and not for the continuous time process. The following corollaries express this product form.

**Corollary 1.2.12.** *In equilibrium, the state of queue $k$ is independent of the rest of the network and is $\boldsymbol{c}_k$ with probability $\pi_k(\boldsymbol{c_k})$. When forgetting the customer classes, the probability that queue $k$ contains $n$ packets is*

$$\mathbb{P}\left(N_k = n\right) = (1-\tilde{\rho}_k)\tilde{\rho}_k^n \quad .$$

*Additionally, the probability that the customer at position $l$ in queue $k$ is of class $i$ and at stage $s$ of its route is $\frac{\lambda_i}{\tilde{\lambda}_k}\mathbb{1}_{r(i,s)=k}$.*

**Corollary 1.2.13.** *A customer of class $i$ reaching the queue $k$ at stage $s$ of his path sees the queue $k$ in its equilibrium state distribution,* i.e. *the probability that the queue $k$ is in state $\boldsymbol{c_k}$ just before its arrival is $\pi_k(\boldsymbol{c_k})$.*

This is trivial for networks with loopless topologies, as all packets arrival processes are Poisson streams. In general, arrivals in a queue of a Kelly network are not a Poisson point process. However, the probability rate that a packet of class $i$ at stage $s$ of his path leaves the queue $k$ in state $\boldsymbol{c_k}$ after his transition can be easily expressed. Let $\boldsymbol{c'_k} = ((i,s), c_k(i), c_k(2), \ldots, c_k(N_k))$ be the state of queue $k$ before the transition. The probability flux is then

$$\pi(\boldsymbol{c'_k})q(\boldsymbol{c'_k}, \boldsymbol{c_k}) = \pi_k(\boldsymbol{c_k})\frac{\lambda_i}{\mu_k} \times \mu_k \quad .$$

Hence, if a packet of class $i$ at stage $s$ just left the queue $k$, the probability that queue $k$ is now in state $\boldsymbol{c_k}$ is $\pi_k(\boldsymbol{c_k})$. But the departures of queue $k$ in the direct process are the arrivals in queue $k$ in the reversed process, which is enough to conclude the corollary.

### 1.2.4 The M/GI/1 queue

We have presented up to now only queueing systems which are remarkably Markovian. There are two reasons for this: first, Markovian systems are easier to analyze, and hence more results are known about them. Second, these models exhibit memoryless properties

and low interaction between customers, which can be expected when dealing with a large system with a large customer population.

However, queueing theory is not limited to Markovian systems, and many results are known about other (albeit less complicated) systems. We shall present here a few results about the M/GI/1 queue as an example, but since these systems are not central to this dissertation and rather more difficult to study, we will limit ourself to the single M/GI/1 case. Although we will not use it here, it should be noted that the Palm calculus approach, such as presented in [BB03] is extremely powerful for non-Markovian systems.

For an M/M/1 queue, the queue size $N(t)$ is a Markov process. However, this relies on the fact that service times are exponential and i.i.d., and hence have a memoryless property. Here, this memoryless property of services is no longer valid, and $N(t)$ is no more a Markov process: the probability of the next jump can depend on the past jumps[27] through the remaining service time.

It is possible to construct a Markov process for an M/GI/1 queue by adding to the state the remaining service time. But the Markov chain is then a continuous state process, and this leads to slight technical difficulties. A more elegant approach is to consider the specific embedded chain of $N(t)$ just after packet departures, which will be a Markov chain.

Let $t_n$ (resp. $\tau_n$) denote the arrival (resp. departure) time of packet $n$, and $\sigma_n$ its service requirement. Let $L_n$ denote the queue size just after the $n^{\text{th}}$ departure, *i.e.* $L_n = N(\tau_n^+)$. Let $A$ be the arrival point process. $L_n$ then verifies the following relation:

$$L_{n+1} = \begin{cases} L_n - 1 + A]\tau_n, \tau_{n+1}] & \text{if } L_n \geq 1 \\ A]t_{n+1}, \tau_{n+1}] & \text{if } L_n = 0 \end{cases} \quad . \tag{1.13}$$

Define $A_n = A]\tau_{n+1} - \sigma_{n+1}, \tau_{n+1}]$ as the number of arrivals during the service of packet $n$, and define $x^+ = \max(x, 0)$. The previous relation be rewritten as follows:

$$L_{n+1} = (L_n - 1)^+ + A_{n+1} \quad . \tag{1.14}$$

Because the arrival process is Poisson, the arrivals prior to $\tau_{n+1} - \sigma_{n+1}$ and the arrivals after $\tau_{n+1} - \sigma_{n+1}$ are independent. Moreover, $L_n$ depends only on the arrivals and service times prior to $\tau_n \leq \tau_{n+1} - \sigma_{n+1}$. Hence, $A_{n+1}$ is independent of $(L_k)_{k<n}$, and $(L_n)_{n\in\mathbb{Z}}$ is a Markov chain. Contrary to the previous cases we have presented, $(L_n)$ is not a birth and death process. Because we "stop" at each departure, downward jumps are limited to one,

[27] To give a rough intuition about this dependence on the past, consider the case where service times have a bimodal distribution on $(\sigma_1, \sigma_2)$ with equal probability for each case. Assume that $\sigma_1$ (resp. $\sigma_2$) is very low (resp. large), and let $\lambda$ denote the arrival intensity. Consider the jump probabilities $\mathbb{P}(N_{n+1} = k|N_n = i)$ of the embedded chain $N_n$, when the queue is not empty (the embedded chain will be a Markov chain if the process is Markovian). If the last event was a departure of a packet, then the remaining service time is either $\sigma_1$ or $\sigma_2$, and we have $\mathbb{P}(N_{n+1} = i - 1|N_n = i, N_{n-1} = i + 1) = 0.5 * (e^{-\lambda\sigma_1} + e^{-\lambda\sigma_2}) \to 0.5$ and $\mathbb{P}(N_{n+1} = i + 1|N_n = i, N_{n-1} = i + 1) = 1 - 0.5 * (e^{-\lambda\sigma_1} + e^{-\lambda\sigma_2}) \to 0.5$, where the limits corresponds to $\sigma_1 \to 0$ and $\sigma_2 \to \infty$. However, if the last event was an arrival and the penultimate event was a departure, it means that there was at least one arrival during the current service time, and the next event will be a departure only if there is a single arrival in the current service time. Hence, $\mathbb{P}(N_{n+1} = i - 1|N_n = i, N_{n-1} = i + 1) = 0.5\lambda * (\sigma_1 e^{-\lambda\sigma_1} + \sigma_2 e^{-\lambda\sigma_2}) \to 0$. This is enough to conclude that $(N_n)$ does not have the Markov property.

but upward jumps can be arbitrarily large.

From (1.14), the transition probabilities $p(n,i)$ of $L_n$ are:

$$p(n,n+k) = \begin{cases} \mathbb{P}\left(A(]0,\sigma]) = k+1\right) = \mathbb{E}\left[\frac{(\lambda\sigma)^{k+1}}{(k+1)!}e^{-\lambda\sigma}\right] & \text{if } k \geq -1 \text{ and } n \geq 1 \\ \mathbb{P}\left(A(]0,\sigma]) = k\right) = \mathbb{E}\left[\frac{(\lambda\sigma)^{k}}{(k)!}e^{-\lambda\sigma}\right] & \text{if } k \geq 0 \text{ and } n = 0 \\ 0 & \text{if } k < -1 \end{cases} . \tag{1.15}$$

We can now determine the stability region of the queue:

**Theorem 1.2.14.** *An M/GI/1 queue is stable if its load $\rho = \lambda\mathbb{E}[\sigma] < 1$ and the second moment $\mathbb{E}[\sigma^2]$ of the service requirement is finite, and unstable otherwise. Additionally, if the queue is stable, it's size distribution $\boldsymbol{\pi} = (\pi_0, \pi_1, \ldots, \pi_k, \ldots)$ has as characteristic function*

$$\psi_N(z) = \mathbb{E}[z^N] = \frac{(1-\rho)(1-z)\mathbb{E}\left[e^{-\lambda\sigma(1-z)}\right]}{\mathbb{E}\left[e^{-\lambda\sigma(1-z)}\right] - z} \quad .$$

*Proof.* From (1.14), $L_n$ is a reflected random walk on $\mathbb{Z}^+$, with increments $A_n - 1$. Hence, $L_n$ is recurrent iff $\mathbb{E}\left[A_n - 1\right] < 0$ and the increment second moment is finite.

We have seen that $\mathbb{P}\left(A_n = k\right) = \mathbb{E}\left[\frac{(\lambda\sigma_n)^k}{k!}e^{-\lambda\sigma_n}\right]$. Hence the following:

$$\begin{aligned} \mathbb{E}\left[A_n - 1\right] &= \sum_{k=0}^{\infty}(k-1)\mathbb{P}\left(A_n = k\right) \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty} k\frac{(\lambda\sigma_n)^k}{k!}e^{-\lambda\sigma_n}\right] - \mathbb{E}\left[\sum_{k=0}^{\infty}\frac{(\lambda\sigma_n)^k}{k!}e^{-\lambda\sigma_n}\right] \\ &= \mathbb{E}\left[\lambda\sigma_n e^{\lambda\sigma_n}e^{-\lambda\sigma_n}\right] - \mathbb{E}\left[e^{\lambda\sigma_n}e^{-\lambda\sigma_n}\right] \\ \mathbb{E}\left[A_n - 1\right] &= \rho - 1 \quad . \end{aligned}$$

Similarly, $\mathbb{E}\left[(A_n-1)^2\right] = \lambda^2\mathbb{E}[\sigma^2] + 1 - \rho$ is also finite, and $(L_n)$ admits a stable distribution.

It remains to show that $(L_n)$ and $N(t)$ have the same distribution. Since $L_n$ converges to its equilibrium distribution, $\mathbb{P}\left(L = k\right) = \lim_{A\to\infty}\sum_{n=0}^{A}\frac{1}{A}\sum_{i=n}^{A}\mathbb{1}_{L_n=k}$. But since $N(t)$ has only increments of size 1, each upward jump has a corresponding downward jump, and we have that $|\sum_{n=0}^{A}\mathbb{1}_{N(t_n)=k} - \mathbb{1}_{N(\tau_n)=k}| \leq 1$. Taking the limit of this last inequality leads to $\lim_{A\to\infty}\sum_{n=0}^{A}\frac{1}{A}\sum_{i=n}^{A}\mathbb{1}_{N(t_n)=k} = \mathbb{P}\left(L = k\right)$, which prove that the queue size just before packet arrivals admits the same stationary distribution as $(L_n)$. As the arrivals are a Poisson stream, we can conclude using the PASTA property that $(L_n)$ and $N(t)$ have the same equilibrium distribution.

This distribution $\boldsymbol{\pi}$ satisfies the balance equations, and the following holds:

$$\forall k \geq 0, \pi_k = \sum_{n=0}^{\infty} \pi_n p(n,k)$$
$$\pi_k = \sum_{n=1}^{k+1} \pi_n \mathbb{E}\left[\frac{(\lambda\sigma)^{k-n+1}}{(k-n+1)!} e^{-\lambda\sigma}\right] + \pi_0 \mathbb{E}\left[\frac{(\lambda\sigma)^k}{k!} e^{-\lambda\sigma}\right] \quad .$$

By summation, we have

$$\begin{aligned}
\psi_N(z) &= \sum_{k=0}^{\infty} \pi_k z^k \\
&= \sum_{k=0}^{\infty} \left( \sum_{n=1}^{k+1} \pi_n \mathbb{E}\left[\frac{(\lambda\sigma)^{k-n+1}}{(k-n+1)!} e^{-\lambda\sigma}\right] + \pi_0 \mathbb{E}\left[\frac{(\lambda\sigma)^k}{k!} e^{-\lambda\sigma}\right] \right) z^k \\
&= \sum_{k=0}^{\infty} \pi_0 \mathbb{E}\left[\frac{(\lambda\sigma z)^k}{k!} e^{-\lambda\sigma}\right] + \sum_{n=0}^{\infty} \sum_{k=n}^{\infty} \pi_{n+1} \mathbb{E}\left[\frac{(\lambda\sigma)^{k-n} z^k}{(k-n)!} e^{-\lambda\sigma}\right] \\
&= \pi_0 \mathbb{E}\left[e^{-\lambda\sigma(1-z)}\right] + \sum_{n=0}^{\infty} \pi_{n+1} z^n \mathbb{E}\left[e^{-\lambda\sigma(1-z)}\right] \\
\psi_N(z) &= \frac{\psi_N(z) - \psi_N(0)}{z} \mathbb{E}\left[e^{-\lambda\sigma(1-z)}\right] + \pi_0 \mathbb{E}\left[e^{-\lambda\sigma(1-z)}\right] \quad .
\end{aligned}$$

Since $\psi_N(0) = \pi_0$, we get that

$$\psi_N(z) = \psi_N(0) \frac{(1-z)\mathbb{E}\left[e^{-\lambda\sigma(1-z)}\right]}{\mathbb{E}\left[e^{-\lambda\sigma(1-z)}\right] - z} \quad .$$

Using the normalization constraint $\psi_N(1) = 1$, we get that $\psi_N(0) = \pi_0 = 1 - \rho$, and the last part of the theorem is proven. □

This theorem characterizes the stability and size distribution of the M/GI/1 queue. In particular, the probability that the queue is empty is $1 - \rho$, which is a valid result for general independent arrivals. Similarly, the GI/GI/1 queue is stable under the same conditions as the M/GI/1 queue. The characteristic function of the queue size is however specific to the M/GI/1 case.

Using this result, it is easy to deduce the Pollazcek-Khintchine formula:

**Theorem 1.2.15** (Pollazcek-Khintchine formula)**.** *For an M/GI/1 queue with arrival intensity $\lambda$ and service requirement distribution $f_\sigma$, the Laplace transform $\mathcal{L}_W(s)$ of the waiting-time distribution $f_W(t)$ is:*

$$\mathcal{L}_W(s) = \mathbb{E}\left[e^{-sW}\right] = \frac{(1-\rho)s}{s - \lambda(1 - \mathcal{L}_\sigma(s))} \quad .$$

*Proof.* We will prove it[28] by expressing the characteristic function of the queue size as a

[28]This result can be also proven directly, from the expression $e^{-sW(t)} = e^{-sW(0)} + \int_{x=0}^{t} \frac{\partial e^{-sW(z)}}{\partial z}\Big|_{z=x} dx +$

function of the Laplace transform of the delay. Since the discipline is FIFO, the number of packets in the queue just after a departure is equal to the number of packets during the delay of this packet (*i.e.* between its arrival and its departure). Hence, we have:

$$\begin{aligned}
\psi_N(z) &= \sum_{k=0}^{\infty} \pi_k z^k \\
&= \sum_{k=0}^{\infty} \int_{t=0}^{\infty} \mathbb{P}\left(L_n = k | D_n = t\right) f_D(t) dt z^k \\
&= \sum_{k=0}^{\infty} \int_{t=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^k}{k!} f_D(t) z^k dt \\
&= \int_{t=0}^{\infty} e^{-\lambda(1-z)t} f_D(t) dt \\
\psi_N(z) &= \mathcal{L}_D(\lambda(1-z)) \quad .
\end{aligned}$$

Using theorem 1.2.14, we get that

$$\mathcal{L}_D(s) = \psi_N(1 - \frac{s}{\lambda}) = \frac{(1-\rho)s\mathcal{L}_\sigma(s)}{s - \lambda(1 - \mathcal{L}_\sigma(s))} \quad .$$

Finally, we conclude the proof by saying that for all packets, $D_n = W_n + \sigma_n$, and hence $\mathcal{L}_D(s) = \mathcal{L}_W(s) \times \mathcal{L}_\sigma(s)$. □

Since $\mathbb{E}[W] = -\frac{\partial \mathcal{L}_W(s)}{\partial s}\Big|_{s=0}$, we can deduce the following Pollaczek-Khintchine mean-value formulas:

$$\mathbb{E}[W] = \frac{\lambda \mathbb{E}\left[\sigma^2\right]}{2(1-\rho)} \tag{1.16}$$

and

$$\mathbb{E}[D] = \frac{\lambda \mathbb{E}\left[\sigma^2\right]}{2(1-\rho)} + \mathbb{E}[\sigma] \quad . \tag{1.17}$$

We conclude this section by Little's law. Little's law is not restricted to M/GI/1 queue, and is valid for any stable queueing system with non-preemptive discipline:

$$\mathbb{E}[N] = \lambda \mathbb{E}[D] \quad . \tag{1.18}$$

As written here, Little's law is easily deduced for M/GI/1 queue from the relation

$\sum_{0<T_n\leq t} \left(e^{-sW(T_n^+)} - e^{-sW(T_n^-)}\right)$, where $(T_n)$ are the discontinuity points of $W(t)$, *i.e.* the packets arrival times. The result follows by taking the expectation and the limit $t \to \infty$.

$\psi_N(z) = \mathcal{L}_D(\lambda(1-z))$ that we saw in the proof of theorem 1.2.15. Indeed, we have

$$\begin{aligned}\mathbb{E}[N] &= \left.\frac{\partial \psi_N(z)}{\partial z}\right|_{z=1} \\ &= \mathbb{E}\left[\lambda D e^{-\lambda(1-z)D}\right]_{z=1} \\ \mathbb{E}[N] &= \lambda \mathbb{E}[D] \quad .\end{aligned}$$

Little's law is valid for any system or even sub-system, as long as $N$ is the law of the number of customers in the (sub)system, and $D$ the law of the time spent by customers in the (sub)system. In particular, if $\tilde{N}$ is the number of customers in the buffer of a queue, we have

$$\mathbb{E}\left[\tilde{N}\right] = \lambda \mathbb{E}[W] \quad .$$

As $N = \tilde{N} = 0$ when the queue is empty, and $N = \tilde{N} + 1$ otherwise, we have that $\mathbb{E}[N] - \mathbb{E}\left[\tilde{N}\right] = \mathbb{P}(N \geq 1)$, and using Little's law for both the queue and the buffer (excluding the server), we get once again the probability that the queue is busy (or empty):

$$\mathbb{P}(N \geq 1) = \lambda(\mathbb{E}[D] - \mathbb{E}[W]) = \lambda \mathbb{E}[\sigma] = \rho \quad .$$

## 1.3 Bandwidth sharing networks: a macroscopic model

### 1.3.1 Bandwidth sharing networks

Queueing theory focuses on the microscopic scale of networks. Mechanisms are solved and explained at the packet level. However, as application messages correspond to many packets, it makes sense to try to study the network performance as perceived by the user, *i.e.* the performance of all packets stemming from the same user. In other words, we are here more interested in the connection or flow point of view of the network than in its packet-level analysis.

A natural question that arises in this context is for example to determine which share of the network capacity should (or will) get a specific application or a specific user. In the network community, this is usually called bandwidth sharing, and has been studied for a long time. The objective of bandwidth sharing is usually to use all the available bandwidth, whilst keeping the system stable and maintaining a kind of "fairness" in the allocation to different users. Many algorithms, including most notably TCP, are meant to allocate bandwidth to flows in a stable and fair way.

#### Fairness and utility

Stability is a well-defined notion, both from the mathematical and the engineering points of view. Fairness is less natural. The first natural notion of fairness is the max-min fair allocation, where bandwidth (or resources in general) is shared in the most equal possible way, meaning that any individual bandwidth increase within the region of feasible allocations

must be at the cost of a decrease of some already smaller bandwidth. This natural interpretation has been the definition of fairness for a long time. In [Kel97], Kelly questioned the optimality of max-min allocation, and introduced the notion of proportional fairness. In [MW00], Mo and Walrand generalized this allocation with the family of weighted $\alpha$-fair allocations, which is defined as follows:

**Definition 1.3.1** ($\alpha$-fairness)**.** Let $\mathcal{S}$ be the set of users of a network, and let $\Gamma \subset (\mathbb{R}^+)^{|\mathcal{S}|}$ be a set of feasible allocations. Let $\boldsymbol{w} = (w_s)_{s\in\mathcal{S}} \in (\mathbb{R}^+)^{|\mathcal{S}|}$ be a sequence on non-negative weights. Let $\alpha \geq 0$ be a (possibly infinite) value.

An allocation $\boldsymbol{\gamma} = (\gamma_s)_{s\in\mathcal{S}}$ is said to be $(\boldsymbol{w}, \alpha)$-fair if it maximizes among all feasible allocations the following $(\boldsymbol{w}, \alpha)$-utility $U_\alpha(\boldsymbol{\gamma})$:

$$U_\alpha(\boldsymbol{\gamma}) = \begin{cases} \sum_{s\in\mathcal{S}} w_s \log(\gamma_s) & \text{if } \alpha = 1 \\ \sum_{s\in\mathcal{S}} w_s \frac{\gamma_s^{1-\alpha}}{1-\alpha} & \text{otherwise} \end{cases}. \tag{1.19}$$

The weights are often omitted, and assumed to be all equal to 1 in this case.

*Remark.* The max-min allocation is the limits of $(w, \alpha)$-fair allocation when $\alpha$ goes to infinity. Symmetrically, maximum-throughput allocation (which maximizes the sum of the rates) is the limit of $(w, \alpha)$-fair allocation when $\alpha$ goes to zero.

This notion of bandwidth sharing extends naturally to servers and networks. Consider for example a single server with a limited capacity $C$, with clients of different classes. How shall one share this capacity among the different clients? A natural answer is to do it in a way maximizing a well-chosen utility function, whilst keeping the total allocated bandwidth lower than $C$. We will not consider here the question of how to do such a sharing. Many have proposed (distributed or centralized) algorithms that achieve some desired bandwidth sharing. Such a goal is indeed important, but we will consider here as granted a way to do an optimal bandwidth allocation that maximizes any utility function.

### Static networks

Let us now generalize this approach to networks. Consider for example a network as depicted in figure 1.7 with three different servers $S_1$, $S_2$ and $S_3$ of respective capacities $C_1$, $C_2$ and $C_3$, and for different class of users. Each class of users uses a fixed route, consisting of a list of 1 or more servers, possibly with repetitions. For example, consider that route 1 of clients of class 1 consists of the path $(S_1, S_2, S_3)$ (meaning that users of class 1 first cross $S_1$, than $S_2$ and finally $S_3$ before exiting the network), route 2 is $(S_1, S_3)$, route 3 crosses only the server $S_2$, and route 4 consists of $(S_1, S_3, S_2, S_1)$. Let $n_i$ denote the number of users of class $i$, and $\gamma_s$ be the bandwidth allocated to user $s$.

Allocating a bandwidth $\gamma_s$ to a client $s$ of class $i$ consumes $\gamma_s$ resources on all servers belong to the route $i$, multiplied by the multiplicity of the server in the route (in the example above, a client $s$ of class 4 would consume $2\gamma_s$ on $S_1$). A bandwidth allocation is then said feasible if on any server, the bandwidth consumption is less than the capacity of the server.

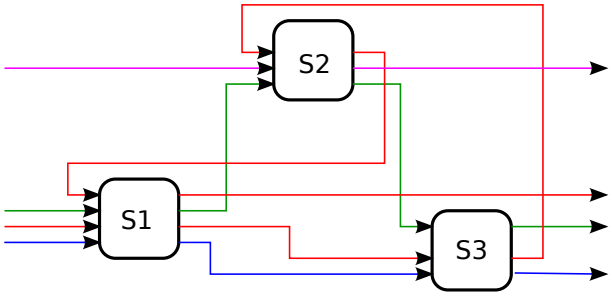



Figure 1.7: An example of bandwidth sharing network.

**Lemma 1.3.2.** *Assume that the utility function $U$ is strictly concave. Then any bandwidth allocation maximizing this utility among a compact set of feasible allocations allocates the same bandwidth $\gamma_i$ to all users of the class $i$.*

*Proof.* Let $\boldsymbol{\gamma} = (\gamma_s)_{s\in\mathcal{S}}$ be a feasible utility maximizing $U$. Let $i$ be a class, and $n_i$ the number of clients of class $i$.

Let $\boldsymbol{\gamma'} = (\gamma'_s)s \in \mathcal{S}$ be another allocation, defined by:

$$\gamma'_s = \begin{cases} \frac{\sum_{u\in\text{class}(i)}\gamma_u}{n_i} & \text{if } s \in \text{class}(i) \\ \gamma_s & \text{otherwise} \end{cases} .$$

Then $\boldsymbol{\gamma'}$ is a feasible allocation, since for any class, the total bandwidth allocated to the class in $\boldsymbol{\gamma'}$ is identical to the total bandwidth allocated to the same class in $\boldsymbol{\gamma}$. Additionally, by convexity of $U$, we have that

$$\sum_{s\in\text{class }(i)} U(\gamma'_s) \geq \sum_{s\in\text{class }(i)} U(\gamma_s)$$

with a strict inequality if there is at least one user $s$ where $\gamma_s \neq \gamma'_s$. Hence, since $\boldsymbol{\gamma}$ maximizes $U$, we have that $\boldsymbol{\gamma} = \boldsymbol{\gamma'}$, and the result follows. □

*Remark.* The $(w, \alpha)$-fair utilities are concave utilities.

**Example 1.3.1:** With these notations, an allocation will be feasible in the above example if it respects the following inequalities, called the capacity constraints:

$$\begin{aligned} n_1\gamma_1 + n_2\gamma_2 + 2n_4\gamma_4 &\leq C_1 \\ n_1\gamma_1 + n_3\gamma_3 + n_4\gamma_4 &\leq C_2 \\ n_1\gamma_1 + n_2\gamma_2 + n_4\gamma_4 &\leq C_3 \end{aligned} .$$

### Dynamical networks

This allocation is defined for any set of users. When this set of clients and their routes, the set of servers and their capacities does not evolve in time, the bandwidth sharing network

is said to be static: users cannot join or leave the network. However, today's networks are often dynamic: a small fraction of the potential users is actually using the network. Users sometimes join the system (*i.e.* start using their web browser or initiate a file download for example). As for queuing theory, in the simplest model, users of class $i$ arrive at a fixed rate $\lambda_i$, require a random amount of service and leave when their service is finished.

The instantaneous bandwidth allocation at any time is performed as if the network was static with its current set of clients. This instantaneous bandwidth allocation hence determines the time needed for a client to have its required service finished. We do not give here details about the detailed clients arrival process: it can be any point process with intensity $\lambda_i$, meaning that in average, $\lambda_i$ clients of class $i$ join the system per unit of time. This arrival process can be steady or bursty, the most canonical example being the Poisson process. Similarly, one can imagine many service requirement distribution, including the canonical exponential distribution. The only assumption is that the mean service requirement $\sigma_i$ of users of class $i$ is finite.

How do such fair allocations work? For any strictly concave utility functions, including the $\alpha$-fairness family, it attributes a lower bandwidth to classes with long route, since they use more resource than shorter routes for the same bandwidth. In this aspect, they differ from the max-min allocation, which can dramatically reduce the bandwidth allocated to some users in order to gain a negligible increase in the allocation to a user who has a lower bandwidth, and they have some global efficiency criteria. However, contrary to the maximum-throughput, they do not allow starvation easily (in fact, $\alpha$-fairnesses do not allow any starvation), meaning that users will (nearly) always have a positive bandwidth allocation. This comes from the fact that these utility functions have some fairness criteria, which value more bandwidth increases on lower-bandwidth users than on higher-bandwidth users. For $\alpha$-fairnesses, this trade-off between efficiency and fairness (or social care) is controlled by the value of $\alpha$: the lower $\alpha$ is, the more efficient the allocation is, and the higher $\alpha$ is, the fairer the allocation is.

Bandwidth sharing networks were introduced by Kelly in [KMT98], in order to discuss the optimality of max-min bandwidth sharing. Their main advantage is that, ignoring the microscopic interaction of packets, they capture well the macroscopic behaviour of networks. Hence, they provide a useful tool to compare the global performance (such as the mean delay for service completion) of different bandwidth allocations and utilities. They can be used also to compare different routing schemes, including multipath routing. Bandwidth sharing networks are a good approximation of the steady state distribution of queuing networks: This is best illustrated by the following result, proved by Kelly *et al.* in [KMT98]: in a stable regime of a static network, (a simplified version of) TCP-algorithm leads to a bandwidth allocation which is a random process whose stable point is a weighted proportionally-fair allocation. This result was generalized in the following years by many people, in order to include some additional properties of TCP. The generalization to dynamical networks was mostly introduced by Bonald, Massoulié, Proutière and Roberts, and lead to important results. In [BM00], Bonald and Massoulié established the crucial result

that $(w, \alpha)$-fair allocations achieve stability under the necessary condition that no individual link is overloaded, and $\alpha \in (0, \infty)$. In [BJP04], Bonald *et al.* stated that bandwidth allocation should be insensitive, meaning that the stable state of the system should depend only on the networks topology and capacities, and on the traffic intensities. The precise arrival processes and service distributions should not matter. These insensitive (or balanced) bandwidth allocations have a very specific stationary flow distribution, which leaves little room to inverse problems.

### 1.3.2 Bandwidth sharing networks are useful outside communication networks

We have introduced bandwidth sharing networks as a model for communication networks. However, their applications are much broader than communication networks. Bandwidth sharing networks can model any system with known utility and (linearly) limited resources, as shown in the two following examples.

**Example 1.3.2** (Factory production)**:** Consider the case of a factory, with production constraints. The servers are then limited resources, such as manpower, periodic supplies or equipment. Classes are then product that need to be produced, and require part of the resources. The "route" of a product determines which and how much of the resources are needed. A resource can be heavily needed for a product, and hence appear several times in the route. The bandwidth allocation corresponds to the total production per unit of time. The utility function expresses the global gain of the production. Two specific utility functions must be mentioned here: first, the weighted maximum throughput allocation, which corresponds to the global gain if all products can be sold independently at fixed prices equal to their weights ; second, the max-min allocation, which express the global gain if one need to assemble together one of each product to product the final good. $(w, \alpha)$-fairness is an intermediate step, where the price of goods decreases when the number of available goods increases. The bigger $\alpha$ is, the sharper the decrease is. The limiting case is $\alpha = \infty$, as described above, where additional goods have no value if you cannot assemble them. This example of bandwidth sharing networks as production constraints will be later referred as the production example, or production context.

**Example 1.3.3** (Budget allocation)**:** Another example of application of bandwidth sharing networks outside communication networks is social care and governmental (or association) budgets, which we will refer to as the budget allocation example or context. Assume that one can divide the population in different groups with distinct characteristics, which one could call classes. Consider that the government (or association) has different levers at its disposal, all of which needs some budget, and have inhomogeneous efficiency in increasing the "condition" of different classes. This levers could be for example public education, public universities, public transport, family welfare, unemployment security, economic development aid, taxes reduction, etc. More precisely, assume that in order to increase the

wealth of one person of one class $i$ of $\gamma_i$, the government needs to spend at least $\gamma_i \times a_{i,k}$ of the lever $k$, where those values $a_{i,k}$ are known integer values.[29] What are the constraints of feasible budgets in this case? Denote respectively by $C_k$, $N$ and $n_i$ the budget allocated to lever $k$, the total number of classes, and the number of persons in class $i$. Then for any lever $k$, the expenses must be lower than the budget, which in mathematical form, reads as: $\forall k, \sum_{i=1}^{N} n_i a_{i,k} \gamma_i \leq C_k$. This is equivalent to the capacities constraints for bandwidth sharing networks, where users of class $i$ would cross the server $k$ $a_{i,k}$ times. Now, one needs to evaluate the benefits of budget decisions. Let us assume that there is a known function $f_i$, that indicates the global benefit $f_i(\gamma_i)$ of ensuring a wealth increase $\gamma_i$ to class $i$. The global aim of budget decision is then to find the class wealth increases that maximize the global benefit, *i.e.* find $\mathrm{argmax}_{\boldsymbol{\gamma}} \sum_{i=1}^{N} n_i f_i(\gamma_i)$. This is equivalent to the global utility that is maximized by the bandwidth allocation in bandwidth sharing networks. The same utility function could of course be used here: for example, maximum throughput utility would mean here that one tries to maximize, as in the capitalism theory, the global wealth of the country. Max-min allocation is a lot more social, up to the point of being communist: it aims at giving the best situation to the poorest, in spite of having to dramatically cut the wealth of richer people. $\alpha$-fairnesses are somewhere between maximum throughput and max-min, and are strictly concave (meaning that one more dollar is more valuated for poor people than for rich people, but has a positive value for all people). In contrast to communication networks, we have here allowed different utility functions for different classes, since their "starting situation" might be different. What would be the aim of modelling welfare as such a bandwidth sharing network? It provides an easy way to compare the global gain of different budget allocations, as can be shown in the following example: assume that the total budget for welfare is known, and that one has to split it among the different levers. Then, if the utility functions and class populations are known, it is easy to estimate the global effect of splitting the total budget $C$ in $(C_1, C_2, \ldots, C_K)$ for the different levers, and one can try to find the best splitting.

### 1.3.3 One single path

This section provides a first example of bandwidth sharing network and detailed computation for its bandwidth allocation in the case of $\alpha$-fair allocation. The network is the most simple example we can imagine, that is a single path from a source to a destination. We consider only the static network, where the number of users in each class is fixed. Users may not enter or leave the system. This section aims both at giving a first concrete example of bandwidth sharing network and their bandwidth allocations, and providing preliminary results we will use in chapter 5 of this dissertation.

Consider a single static path, as depicted in Fig. 1.8. The path consists of $K$ servers $(S_1, \ldots, S_K)$ in series, where the server $S_j$ has capacity $C_j$. There are $K+1$ class of users:

[29] This assumption is not restricting: rational case is easily extended by multiplication of costs and budgets by a common constant, and real case as limit of the rational case.

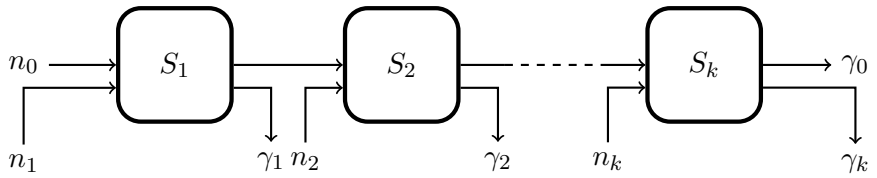



Figure 1.8: An example of path.

users of class 0 use the whole path, and users of class $i$ ($1 \leq i \leq K$) enter the path just before server $S_i$, and exit the path after server $S_i$. Each class $i$ has $n_i$ users, and each of these users receives a bandwidth equal to $\gamma_i$.

According to the settings, the capacity constraints are:

$$\forall 1 \leq i \leq K, \qquad n_0\gamma_0 + n_i\gamma_i \leq C_i \quad .$$

Because users of class $i$ (assuming their existence) have no other limits on their bandwidth than these capacity constraints, the constraints must be tight, and we get the following:

$$\forall 1 \leq i \leq K, \qquad n_0\gamma_0 + n_i\gamma_i = C_i \quad . \tag{1.20}$$

Considering the $\alpha$-fairness[30] defined in (1.19), and using (1.20), we get the following utility:

$$\begin{aligned} U_\alpha(\boldsymbol{\gamma}) &= \sum_{i=0}^{K} n_i w_i \frac{\gamma_i^{1-\alpha}}{1-\alpha} \\ &= \frac{1}{1-\alpha}\left( n_0 w_0 \gamma_0^{1-\alpha} + \sum_{i=1}^{K} n_i w_i \left( \frac{C_i - n_0\gamma_0}{n_i} \right)^{1-\alpha} \right) \quad , \end{aligned} \tag{1.21}$$

where for simplicity of notation, when $n_i = 0$ we abusively define

$$n_i \left( \frac{C_i - n_0\gamma_0}{n_i} \right)^{1-\alpha} = \begin{cases} 0 & \text{if } C_i - n_0\gamma_0 \geq 0 \\ -\infty & \text{otherwise} \end{cases} \quad .$$

**Maximum throughput**

The bandwidth allocation chosen here is one of those that maximize the (weighted) global throughput, *i.e.* the (weighted) sum of all individual bandwidth[31]. It corresponds to the case $\alpha = 0$. The utility is then

$$U_0(\boldsymbol{\gamma}) = \sum_{i=1}^{K} n_i w_i \frac{C_i}{n_i} + n_0\gamma_0 \left( w_0 - \sum_{i=1}^{K} w_i \mathbb{1}_{n_i>0} \right) \quad .$$

[30] We will abusively write $U_1(\boldsymbol{\gamma}) = \sum_{s\in\mathcal{S}} \frac{\gamma_s^{1-\alpha}}{1-\alpha}$ with $\alpha = 1$, instead of $U_1(\boldsymbol{\gamma}) = \sum_{s\in\mathcal{S}} \log(\gamma_s)$. This is not correct formally, but computations (and in particular differentiation) remains valid, as well as the final results.

[31] In the case of maximum throughput, it is often implicitely assumed that all weights are equal.

Its derivative with regards to $\gamma_0$ depends on the sign of the expression $\left(w_0 - \sum_{i=1}^{K} w_i \mathbb{1}_{n_i>0}\right)$.

Hence, if $w_0 < \sum_{i=1}^{K} w_i \mathbb{1}_{n_i>0}$, the maximum utility allocation in such a case is

$$\begin{cases} \gamma_0 = 0 \\ \gamma_i = \mathbb{1}_{n_i>0} \frac{C_i}{n_i} \end{cases} . \tag{1.22}$$

Otherwise, the maximum utility is

$$\begin{cases} \gamma_0 = \min_{1\leq j\leq K} \frac{C_j}{n_0+x} \\ \gamma_i = \frac{C_i - \min_{1\leq j\leq K} C_j}{n_i} \end{cases} . \tag{1.23}$$

**Max-min allocation**

A bandwidth allocation is said to be max-min fair if and only if any individual bandwidth increase within the region of feasible allocations must be at the cost of a decrease of some already smaller bandwidth. It corresponds to the case $\alpha \to \infty$. The only max-min allocation in our case is

$$\begin{cases} \gamma_0 = \min_{1\leq i\leq K} f_i(C_1, n_0, n_i) \\ \gamma_i = \frac{C_i - n_0\gamma_0}{n_i} \end{cases} , \tag{1.24}$$

where we define for all $i$, $f_i(C_i, n_0, n_i) = \frac{C_i}{n_0+n_i}$.

**Other $\alpha$-fair allocations**

The parameter $\alpha$ is now strictly positive and finite. Taking the derivative of (1.21) leads to the following:

$$\frac{\partial U_\alpha(\boldsymbol{\gamma})}{\partial \gamma_0} = n_0 \left( w_0 \gamma_0^{-\alpha} - \sum_{i=1}^{K} w_i \mathbb{1}_{n_i>0} \left( \frac{C_i - n_0\gamma_0}{n_i} \right)^{-\alpha} \right) . \tag{1.25}$$

Since $U_\alpha$ is continuously differentiable on the region of feasible allocations, and as the limit of the derivative when $\gamma_0$ tends to 0 by positive values is $+\infty$ and the limit when $\gamma_0$ tends to $\min_{1\leq i\leq K} \frac{C_i}{n_0}$ is $-\infty$ (assuming that the corresponding $n_i$ is not null), we know that the maximum of the function on the feasible region is in the interior of the region, and is a stationary point of $U_\alpha$. The $\alpha$-fair allocation hence verifies the following stationary equation:

$$\frac{w_0}{\gamma_0^\alpha} = \sum_{i=1}^{K} w_i \mathbb{1}_{n_i>0} \left( \frac{n_i}{C_i - n_0\gamma_0} \right)^{\alpha} . \tag{1.26}$$

We are not able to solve the stationary equation in a general case.

**Identical capacity along the path**

However, a solution can be found in the particular case when all servers have the same capacity $C$. The equation then reads

$$(C - n_0\gamma_0)^\alpha w_0 = \gamma_0^\alpha \sum_{i=1}^{K} w_i n_i^\alpha \qquad \Leftrightarrow \qquad C - n_0\gamma_0 = \gamma_0 \times \tilde{n}$$

$$\Leftrightarrow \qquad \gamma_0 = \frac{C}{n_0 + \tilde{n}} \quad , \qquad (1.27)$$

where we define the "weighted $\alpha$ sum" as $\tilde{n} = \left( \frac{\sum_{i=1}^{K} w_i n_i^\alpha}{w_0} \right)^{\frac{1}{\alpha}}$. Using the capacity constraints (1.20), we get that $\gamma_i = \frac{\tilde{n}}{\tilde{n}+n_0} \frac{C}{n_i}$.

### 1.3.4 The triangle network

The previous example was the single source-destination path, which can hardly be called a network. We extend here the results to a non-trivial (but small) network topology: a "triangle network", as depicted in Fig. 1.9. Similar to the single path network, we will use this network in chapter 5. However, in contrast to the previous case, we will see here that there is no closed-form formula for the bandwidth allocation. This situation is in fact the most common for bandwidth sharing network: bandwidth allocations are implicitly defined. They can be approximated numerically for a given set of parameters, but no closed form formula exists for most networks.

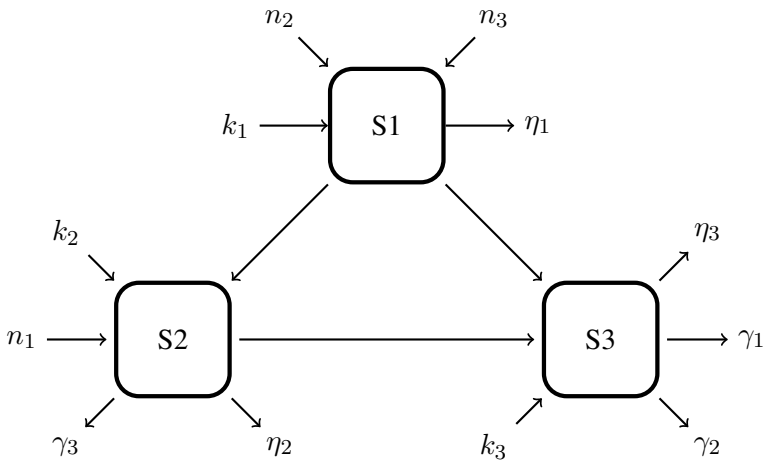



Figure 1.9: The triangle network.

The triangle network consists of 3 servers, each server being connected to both other servers. Server $S_i$ has a capacity $C_i$. $k_i$ flows cross the server $S_i$, and each of them gets an allocated bandwidth $\eta_i$. There are also flows using two servers: $n_3$ (resp. $n_2$ and $n_1$)

flows use the route $(S_1, S_2)$ (resp. $(S_1, S_3)$ and $(S_2, S_3)$) and each of them gets an allocated bandwidth $\gamma_3$ (resp. $\gamma_2$ and $\gamma_1$). For simplicity, we will call flows that use the route $(S_2, S_3)$ (resp. $(S_1, S_3)$ and $(S_1, S_2)$) of class 1 (resp. 2 and 3), and the flows that have route $(S_i)$ of class $i'$.

The capacity constraints read as follows:

$$\begin{aligned} k_1\eta_1 + n_2\gamma_2 + n_3\gamma_3 &\leq C_1 \\ k_2\eta_2 + n_2\gamma_1 + n_3\gamma_3 &\leq C_2 \\ k_3\eta_3 + n_1\gamma_1 + n_2\gamma_2 &\leq C_3 \quad . \end{aligned} \tag{1.28}$$

Assuming that $k_1$, $k_2$ and $k_3$ are positive, these constraints are tight, and we get:

$$\begin{aligned} \eta_1 &= \frac{C_1 - n_2\gamma_2 - n_3\gamma_3}{k_1} \\ \eta_2 &= \frac{C_2 - n_1\gamma_1 - n_3\gamma_3}{k_2} \\ \eta_3 &= \frac{C_3 - n_1\gamma_1 - n_2\gamma_2}{k_3} \quad . \end{aligned} \tag{1.29}$$

Assume also that the bandwidth allocation maximizes the $\alpha$-fairness utility in the capacity region, with weights $w_i$ (resp. $v_i$) for flows of class $i$ (resp. $i'$):

$$\begin{aligned} U_\alpha(\boldsymbol{\gamma}, \boldsymbol{\eta}) &= \sum_{i=1}^{3} k_i v_i \frac{\eta_i^{1-\alpha}}{1-\alpha} + (x_i + n_i) w_i \frac{\gamma_i^{1-\alpha}}{1-\alpha} \\ &= (x_1 + n_1) w_1 \frac{\gamma_1^{1-\alpha}}{1-\alpha} + (x_2 + n_2) w_2 \frac{\gamma_2^{1-\alpha}}{1-\alpha} + (x_3 + n_3) w_3 \frac{\gamma_3^{1-\alpha}}{1-\alpha} \\ &\quad + k_1 v_1 \frac{(C_1 - (x_2 + n_2)\gamma_2 - (x_3 + n_3)\gamma_3)^{1-\alpha}}{(1-\alpha) k_1^{1-\alpha}} \\ &\quad + k_2 v_2 \frac{(C_2 - (x_1 + n_1)\gamma_1 - (x_3 + n_3)\gamma_3)^{1-\alpha}}{(1-\alpha) k_2^{1-\alpha}} \\ &\quad + k_3 v_3 \frac{(C_3 - (x_1 + n_1)\gamma_1 - (x_2 + n_2)\gamma_2)^{1-\alpha}}{(1-\alpha) k_3^{1-\alpha}} \quad . \end{aligned} \tag{1.30}$$

Except in very specific cases (max-min and maximum throughput fairnesses), there is no explicit solution $(\boldsymbol{\gamma}, \boldsymbol{\eta})$ maximizing the utility, even in the simple case where $C_1 = C_2 = C_3$. This situation is representative of the general case for bandwidth allocation in bandwidth sharing networks. The bandwidth allocation is defined implicitly, and numerical approximation, *e.g.* through convex optimization, is the most common way to compute them.

## 1.4 Statistics

The previous sections were devoted to explain the behaviour of networks, and introduce the theoretical tools used to model them. This section aims at presenting basic notions of measurement and statistics, which are the other basis of this dissertation. Statistics is the branch of mathematics concerned with collecting and interpreting data. The theory is rich, and our presentation is far from complete. We refer to [She95, Bor98] for additional results.

### 1.4.1 Parametric estimation and estimators

Assume that you have access to empirical measurement $(X_1, \ldots, X_N)$ of some quantity which has a physical meaning in your system. For example, the samples could be delays of individuals packets, or loss events of the system. Assume also that you can guess, *e.g.* through theoretical modeling, the shape of the distribution of these samples, depending on a few parameters. Taking the example of delays and guessing that the system behaves as an M/M/1 queue, the distribution of the sample should follow an exponential random variable of parameter $\mu - \lambda$. Is it possible then, from this empirical sample and the guessed shape, to guess the value of the unknown parameter? This is exactly what parametric estimators aim at doing.

More formally, consider a family $\mathcal{D}_\theta$ of probability distributions $f_\theta(.)$, where $\theta$ is a parameter, possibly a vector. $f_\theta(.)$ can denote here a probability mass function in the discrete case, or a probability density function in the continuous case (the latter will be used in this presentation).

If $\boldsymbol{X} = (X_1, \ldots, X_N)$ is distributed according $f_{\theta_0}(.)$ for some $\theta_0$, an estimator $\widehat{\theta}$ is a function of the observation data $\boldsymbol{X} = (X_1, \ldots, X_N)$ to the parameter space that aims at recovering the value of some function $g(\theta_0)$ of the real parameter $\theta_0$. In most cases, $g(.)$ is the identity function or some "simple" function . As $\boldsymbol{X}$ is a random vector, $\widehat{\theta}(\boldsymbol{X})$ is also a random variable or vector.

*Remark.* A given estimator should formally always use samples of the same size. However, in practice, we will often consider a class of estimators, one for each sample size, and denote them as a single estimator.

**Example 1.4.1:** Assume for example that the family of considered parametric distributions $\mathcal{D}_\theta = \{f_\theta\} = \left\{U[\theta, \theta + 1[\,\middle|\, \theta \in \mathbb{R}\right\}$ is the family of uniform distributions on an interval of length 1, and the $N$ samples $\boldsymbol{X} = (X_1, \ldots, X_N)$ are uniformly distributed and i.i.d. with distribution $f_{\theta_0}$ with $\theta_0 \in \mathbb{R}$.

An estimator of $\theta_0$ is for example

$$\widehat{\theta}_1(\boldsymbol{X}) = \min_{1 \leq i \leq N} X_i \quad .$$

This estimator returns the maximum possible value for $\theta_0$. Symmetrically,

$$\widehat{\theta}_2(\boldsymbol{X}) = \max_{1 \leq i \leq N} X_i - 1$$

is also an estimator, but $\theta_2$ minimizes the estimation. A median estimator could for example be

$$\widehat{\theta}_3(\boldsymbol{X}) = \frac{\min_{1 \leq i \leq N} X_i + \max_{1 \leq i \leq N} X_i - 1}{2} \quad .$$

$\widehat{\theta}_3$ is the estimator which puts equal distance between $\widehat{\theta}_3$ and the lowest sample and between the highest sample and $\widehat{\theta}_3 + 1$.

The previous example showed that one can design many different estimators for the same problem. This raises a natural question: which estimator should one choose ? And how to measure the quality of an estimator? If building estimators is the primary objective of parametric inference (the branch of statistics which studies the estimators based on parametric distribution families), the second task is to quantify their performance. We will present a few classical definitions and results about this in the following section.

### 1.4.2 A few classical results

**Definition 1.4.1.** The bias $B\left(\widehat{\theta}\right)$ of an estimator $\widehat{\theta}$ is defined as

$$B\left(\widehat{\theta}\right) = \mathbf{E}_{f_{\theta_0}}\left[\widehat{\theta}\left(X\right)\right] - g\left(\theta_0\right) \quad ,$$

where $\mathbf{E}_f\left[h(Y)\right]$ denotes the expectation of $h(Y)$ when $Y$ is sampled with distribution $f$.

*Remark.* The bias depends only on the estimator, and not on the observed data (in contrast to the error for a given sample $X$, which would be $\widehat{\theta}\left(X\right) - g\left(\theta_0\right)$ and depend on the estimator and the sample).

**Definition 1.4.2.** The variance of an estimator is defined as

$$\mathrm{Var}\left(\widehat{\theta}\right) = \mathbf{E}_{f_{\theta_0}}\left[\left(\widehat{\theta}\left(X\right) - \mathbf{E}_{f_{\theta_0}}\left[\widehat{\theta}\left(X\right)\right]\right)^2\right] \quad .$$

As $\widehat{\theta}\left(X\right)$ is in itself a random variable, this definition is precisely the definition of variance for any random variable.

**Definition 1.4.3.** The mean squared error of an estimator is defined as

$$\mathrm{MSE}\left(\widehat{\theta}\right) = \mathbf{E}_{\mathrm{f}_{\theta_0}}\left[\left(\widehat{\theta}\left(\mathrm{X}\right) - \mathrm{g}\left(\theta_0\right)\right)^2\right] \stackrel{\mathrm{def}}{=} \mathrm{Var}\left(\widehat{\theta}\right) + \mathrm{B}\left(\widehat{\theta}\right)^2 \quad .$$

**Definition 1.4.4.** An estimator $\widehat{\theta}$ is said to be unbiased if its bias is 0.

*Remark.* For finite size samples, as shown in the example 1.4.2, unbiased estimators perform often poorly. The problem is that bias speaks only about the mean of the error, and not the mean squared error or the mean absolute error.

**Example 1.4.2:** Assume that X is distributed according to a Poisson law of mean $\theta_0$. We wish to estimate $g(\theta_0) = \mathbb{P}(X = 0)^2 = e^{-2\theta_0}$ with only one observation on $X$. In particular, this is the probability to have no arrival during two units of time for any network with Poisson arrivals of intensity $\theta_0$.

If $\widehat{\theta}(X)$ is an unbiased estimator, then we have

$$\begin{aligned} g(\theta_0) &= \sum_{k=0}^{\infty} \widehat{\theta}(k)\, \mathbb{P}(X = k) \\ e^{-2\theta_0} &= e^{-\theta_0} \sum_{k=0}^{\infty} \widehat{\theta}(k)\, \frac{\theta_0^k}{k!} \\ e^{-\theta_0} &= \sum_{k=0}^{\infty} \widehat{\theta}(k)\, \frac{\theta_0^k}{k!} \quad . \end{aligned}$$

By the uniqueness of Taylor series, we know that

$$\widehat{\theta}(k) = (-1)^k \quad .$$

The estimator is unbiased, and indeed, if $X_1, \ldots, X_N$ are i.i.d. with a Poisson distribution, we have

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \widehat{\theta}(X_i) = g(\theta_0) = e^{-2\theta_0} \quad .$$

However, if one wants to use this estimator on a single sample $X$, it gives no valuable information about $g(\theta_0)$. For even $k$, the unbiased estimator states that it is almost sure that no arrivals will occur during any time interval of length 2. Clearly, that probability is strongly over-estimated. But for odd $k$, the result is a non-sense, since a probability cannot be negative!

Finally, the mean squared error of the unbiased estimator is

$$\begin{aligned} \mathrm{MSE}\left(\widehat{\theta}\right) &= \sum_{k=0}^{\infty} \left(e^{-2\theta_0} - (-1)^k\right)^2 \frac{\theta_0^k}{k!} e^{-\theta_0} \\ &= 1 - e^{-4\theta_0} \quad . \end{aligned}$$

For $\theta_0 \geq \frac{\ln 2}{4}$, the mean squared error of the unbiased estimator is larger than the mean squared error of the very crude estimator which always returns $\frac{1}{2}$ (its mean squared error is obviously $\left|\frac{1}{2} - e^{-2\theta_0}\right|$, hence less than $\frac{1}{2}$).

We are therefore sometimes interested in estimators[32] that are biased for any finite size samples, but whose bias tends to 0 as the sample size tends grows. Such an estimator will be called consistent.

**Definition 1.4.5.** An estimator sequence $\left(\widehat{\theta}_n\right)_{n\in\mathbb{N}}$ is called weakly (resp. strongly) consistent if $\lim_{n\to\infty}\widehat{\theta}_n(X) = g(\theta_0)$ in probability (resp. almost surely).

This interest for finitely-biased consistent estimator is increased by the Cramer-Rao lower bound on variance of (unbiased) estimators. This means than if we can find a biased estimator that has a mean squared error equal to the Cramer-Rao lower bound, we know that it is as efficient as any unbiased estimator can be.

**Definition 1.4.6.** Let us denote $\theta = (\theta_1, \ldots, \theta_N)$ as the parameter vector. The Fisher information matrix $\mathcal{I}_f(\theta)$ for the density $f_\theta$ is the $N \times N$ matrix defined by $(\mathcal{I}_f(\theta))_{(i,j)} = \mathbf{E}_{f_\theta}\left[\frac{\partial}{\partial\theta_i}\ln f_\theta(X)\frac{\partial}{\partial\theta_j}\ln f_\theta(X)\right]$.

*Remark.* Under suitable regularity conditions, the Fisher information matrix is also the covariance matrix of the vector $(\frac{\partial}{\partial\theta_i}\ln f_\theta)_{1\le i\le N}$. This holds because when one can exchange the differentiation and integral signs, we have

$$\begin{aligned}\mathbf{E}_{f_\theta}\left[\frac{\partial}{\partial\theta_i}\ln f_\theta(X)\right] &\overset{\text{def}}{=} \int\left(\frac{\partial}{\partial\theta_i}\ln f_\theta(X)\right) f_\theta(X)dX \\ &= \int\frac{\partial}{\partial\theta_i} f_\theta(X)dX \\ &= \frac{\partial}{\partial\theta_i}\int f_\theta(X)dX \\ \mathbf{E}_{f_\theta}\left[\frac{\partial}{\partial\theta_i}\ln f_\theta(X)\right] &= 0 \quad . \end{aligned} \tag{1.31}$$

This last relation will be also useful later.

**Theorem 1.4.7** (Cramer-Rao lower bound)**.** *Let $X$ be a random variable according to some density $f_{\theta_0}$, where $\theta_0$ is a $(1\times N)$ vector. Let $\widehat{\theta}(X) \in \mathbb{R}^K$ be an estimator of $g(\theta_0)$. Let $h(\theta_0) \in \mathbb{R}^K$ be the mean of $\widehat{\theta}(X)$. Let $H'(\theta)$ denote the derivative $(K\times N)$ matrix of $h(\theta)$, that is $(H'(\theta))_{(i,j)} = \frac{\partial h_i(\theta)}{\partial\theta_j}$. Assume that:*

1. *The Fisher information matrix is always defined, or equivalently, for all $X$ such that $f_{\theta_0}(X) > 0$ and all $i < N$, $\frac{\partial}{\partial\theta_i}\ln f_{\theta_0}(X)$ exists and is finite.*
2. *One can differentiate under the integral sign,* i.e. *for any $i$ and $j$:*
   $\frac{\partial}{\partial\theta_j}h_i(\theta_0) \overset{\text{def}}{=} \frac{\partial}{\partial\theta_j}\left(\int\widehat{\theta}_i(X) f_{\theta_0}(X)\,dX\right) =^{33} \int\widehat{\theta}_i(X)\left(\frac{\partial}{\partial\theta_j}f_{\theta_0}(X)\right)dX$

   *Then*

$$Cov_{f_{\theta_0}}\left[\widehat{\theta}(X)\right] \ge H'(\theta_0)\left(\mathcal{I}_f(\theta_0)\right)^{-1\,t}\left(H'(\theta_0)\right)$$

*where $\ge$ is defined by positive-definite matrix ordering.*

[32] Formally, these are families of different estimators, one for each sample size.

[33] Recall that $\widehat{\theta}(X)$ is a function which depends only on $X$, and not on $\theta$.

*Proof.* From assumption 2, we have

$$\begin{aligned}\frac{\partial h_i(\theta_0)}{\partial \theta_j} &= \int \widehat{\theta}_i(X)\left(\frac{\partial f_{\theta_0}(X)}{\partial \theta_j}\right) dX \\ &= \int \widehat{\theta}_i(X)\left(\frac{\partial \ln f_{\theta_0}(X)}{\partial \theta_j} f_{\theta_0}(X)\right) dX \\ \frac{\partial h_i(\theta_0)}{\partial \theta_j} &= \mathbb{E}_{f_{\theta_0}}\left[\widehat{\theta}_i(X)\frac{\partial \ln f_{\theta_0}(X)}{\partial \theta_j}\right] \quad .\end{aligned}$$

Considering the vector $K = \left(\widehat{\theta}(X), \frac{\partial \ln f_{\theta_0}(X)}{\theta_1}, \ldots, \frac{\partial \ln f_{\theta_0}(X)}{\theta_N}\right)$, we have thanks to (1.31) that

$$\text{Cov}_{f_{\theta_0}(X)}(K) = \begin{pmatrix} \text{Cov}\left(\widehat{\theta}(X)\right) & H'(\theta_0) \\ {}^tH'(\theta_0) & \mathcal{I}_f(\theta_0) \end{pmatrix} \quad .$$

Using $z = \left(t, -tH'(\theta_0)\left(\mathcal{I}_f(\theta_0)\right)^{-1}\right)$ and the fact that covariance matrix is positive semi-definite, we have that for any vector $t$ in $\mathbb{R}^K$:

$$\begin{aligned} 0 \leq & z\,\text{Cov}(K)\,{}^tz \\ \leq & \left(t, \quad -tH'(\theta_0)\left(\mathcal{I}_f(\theta_0)\right)^{-1}\right)\begin{pmatrix} \text{Cov}\left(\widehat{\theta}(X)\right) & H'(\theta_0) \\ {}^tH'(\theta_0) & \mathcal{I}_f(\theta_0) \end{pmatrix}\begin{pmatrix} {}^tt \\ -\left(\mathcal{I}_f(\theta_0)\right)^{-1}{}^tH'(\theta_0){}^tt \end{pmatrix} \\ 0 \leq & t\,\text{Cov}\left(\widehat{\theta}(X)\right){}^tt - tH'(\theta_0)\left(\mathcal{I}_f(\theta_0)\right)^{-1}{}^tH'(\theta_0){}^tt \\ & - tH'(\theta_0)\left(\mathcal{I}_f(\theta_0)\right)^{-1}{}^tH'(\theta_0){}^tt + tH'(\theta_0)\left(\mathcal{I}_f(\theta_0)\right)^{-1}{}^tH'(\theta_0){}^tt \\ 0 \leq & t\,\text{Cov}\left(\widehat{\theta}(X)\right){}^tt - tH'(\theta_0)\left(\mathcal{I}_f(\theta_0)\right)^{-1}{}^tH'(\theta_0){}^tt \quad . \end{aligned}$$

□

**Example 1.4.3:** This formula may seem complicated, but in most cases, $H'$ will be quite simple. If $g$ is the identity function and $\widehat{\theta}$ is unbiased, then $h(\theta) = \theta$, and the lower bound reads $\text{Cov}\left(\widehat{\theta}(X)\right) \geq \left(\mathcal{I}_f(\theta_0)\right)^{-1}$.

**Example 1.4.4:** Assume now $g$ is the identity, $\theta$ is a scalar and $\widehat{\theta}$ is biased. We have $h(\theta) = \theta + B\left(\widehat{\theta}\right)$. Then $\text{Var}\left(\widehat{\theta}\right) \geq \frac{\left(1+\frac{\partial B(\widehat{\theta})}{\partial \theta_0}\right)^2}{\mathcal{I}_f(\theta_0)}$.

**Definition 1.4.8.** An estimator is said to be efficient if it is unbiased and if it achieves the Cramer-Rao lower bound.

*Remark.* Note that we require an efficient estimator to be unbiased. Therefore, the mean squared error of any efficient estimator is $\text{Tr}\left(\mathcal{I}_f(\theta_0)\right)^{-1}$ when $g$ is the identity. There can be some biased estimators that have a lower mean squared error.

**Definition 1.4.9.** An estimator sequence $\left(\widehat{\theta_n}\right)_{n\in\mathcal{N}}$ is called asymptotically normal if $\sqrt{n}\left(\widehat{\theta_n} - g(\theta_0)\right)$ converges in distribution to $\mathcal{N}(0, V)$ for some $V > 0$.

### 1.4.3 Maximum likelihood estimator

In this section, we will introduce a class of estimators called Maximum likelihood estimators (MLE). A MLE infers parameters by finding the parameter values that maximize the likelihood of the observation data. For the remaining part of this dissertation, all estimators will be MLEs, unless otherwise specified.

**Definition 1.4.10.** Given observations $\mathbf{X} = (X_1, \ldots, X_N)$ distributed according to a density $f_{\theta_0}$, the likelihood function $\mathcal{L}_{\mathbf{X}}(\alpha)$ of $\alpha \in \mathcal{I}m(g)$ is defined by $\mathcal{L}_{\mathbf{X}}(\alpha) = f_{g(\theta)=\alpha}(\mathbf{X}) \stackrel{\text{def}}{=} \max_{\theta \text{ s.t. } g(\theta)=\alpha} f_\theta(\mathbf{X})$.

*Remark.* We will often use the log-likelihood function $\mathcal{L}^*(.) = \log(\mathcal{L}(.))$ when the logarithm simplifies computations.

*Remark.* If $(X_1, \ldots, X_N)$ are i.i.d. distributed, then $\mathcal{L}_{\mathbf{X}}(\alpha) = \prod_{i=1}^N f_{g(\theta)=\alpha}(X_i)$ and $\mathcal{L}^*_{\mathbf{X}}(\alpha) = \sum_{i=1}^N \log f_{g(\theta)=\alpha}(X_i)$.

**Definition 1.4.11.** Let $\mathcal{D}_\theta$ a family of probability distribution $f_\theta(.)$. Given observations $\mathbf{X} = (X_1, \ldots, X_N)$, the maximum likelihood estimator is defined by $\widehat{\theta}(\mathbf{X}) = \operatorname{argmax}_\theta \mathcal{L}_{\mathbf{X}}(\theta) = \operatorname{argmax}_\theta \mathcal{L}^*_{\mathbf{X}}(\theta)$.

*Remark.* We will often denote by $\widehat{\theta_k}$ the maximum likelihood estimator given $k$ i.i.d. observations. This defines a sequence of estimators as the sample size grows.

**Lemma 1.4.12.** *The maximum likelihood estimator is function invariant,* i.e. $\widehat{g(\theta)} = g(\widehat{\theta})$.

**Example 1.4.5:** We consider again the case of example 1.4.2. We want in a first stage to estimate $\theta_0$ (*i.e.* $g$ is the identity). For one observation $X$, the MLE estimator $\widehat{\theta}(X)$ verifies:

$$\begin{aligned} 0 =& \frac{\partial \mathbb{P}_{\widehat{\theta}(X)}(X)}{\partial \widehat{\theta}} \\ 0 =& \frac{X e^{-\widehat{\theta}(X)} \left(\widehat{\theta}(X)\right)^{X-1}}{X!} - \frac{e^{-\widehat{\theta}(X)} \left(\widehat{\theta}(X)\right)^{X}}{X!} \\ \widehat{\theta}(X) =& X \quad . \end{aligned}$$

In a second stage, we estimate $e^{-2\theta_0}$, and hence consider $g(\theta) = e^{-2\theta}$. Thanks to the function invariance of MLE, $\widehat{g(\theta)} = g(\widehat{\theta}) = e^{-2X}$.

Note that the MLE is biased in this case. We have

$$\mathbf{E}_{f_{\theta_0}}\left[\widehat{\theta}(X)\right] = \sum_{X=0}^{\infty} e^{-2X} e^{-\theta_0} \frac{\theta_0^X}{X!} = e^{-\theta_0(1-\frac{1}{e^2})} \neq e^{-2\theta_0} \quad .$$

The mean squared error is

$$\text{MSE}\left(\widehat{\theta}\right) = \mathrm{e}^{-4\theta_0} - 2\mathrm{e}^{-\theta_0\left(3-\mathrm{e}^{-2}\right)} + \mathrm{e}^{-\theta_0\left(1-\mathrm{e}^{-4}\right)} \quad ,$$

which has to be compared to the mean squared error $(1 - e^{-4\theta_0})$ of the unbiased estimator.

In fact, this situation is representative of the general case: the MLE is often biased for finite sample sizes, but has low mean squared error.

*Remark.* MLE need not exist, or can exist and not be unique.

**Example 1.4.6:** Consider random variables sampled uniformly in $[\theta; \theta + 1]$. Any value between $\max X_i - 1$ and $\min X_i$ is a maximum likelihood estimator. The MLE is not unique in such a case.

Similarly, consider random variables sampled uniformly in $[0, \theta[$. A maximum likelihood estimator would be the lowest value strictly greater than $\max X_i$, which do not exist for any finite sample. MLE is not defined in such a case.

We now give two fundamental theorems. The first one states that under mild conditions, maximum likelihood estimators are consistent when the sample size grows. In practice, this convergence is quite fast, meaning that the bias for finite samples is not a real problem. We do not have however any results to "guarantee" that the convergence is indeed fast.

The second result is that maximum likelihood estimators are asymptotically normal and asymptotically efficient. This means that we know the variance of maximum likelihood estimators, and we can control easily the deviation between one estimation and the true parameter. Once again, mild conditions will be necessary.

**Theorem 1.4.13.** *Let $\mathbf{X} = (X_1, \ldots, X_N)$ are i.i.d. observations with density $f_{\theta_0}$, where $\theta_0$ belongs to the parameter space $\Omega$. Define for every $M \subseteq \Omega$ and every observation $X \in \mathcal{X}$*

$$Z(M, X) \stackrel{\text{def}}{=} \inf_{\theta \in M} \log \frac{f_{\theta_0}(X)}{f_\theta(X)} \quad .$$

*Assume that for each $\theta \neq \theta_0 \in \Omega$, there is an open set $N_\theta$ including $\theta$ such that $\mathbb{E}_{f_{\theta_0}(X)}[Z(N_\theta, X)] > 0$. Assume further that there is a compact set $C$ such that $\mathbb{E}_{f_{\theta_0}}[Z(\Omega \backslash C, X)] > 0$. Then*

$$\lim \widehat{\theta}_n = \theta_0 \ \textit{almost surely (a.s.).}$$

*Proof.* We want to prove that

$$\forall \epsilon > 0, \mathbb{P}_{\theta_0}\left(\limsup_{n \to \infty} \|\widehat{\theta}_n - \theta_0\| \geq \epsilon\right) = 0 \quad .$$

Let $\epsilon > 0$ and $N_0$ be the open ball of radius $\epsilon$ around $\theta_0$. Since the open sets $\{N_\theta : \theta \in C \backslash N_0)\}$ cover the compact set $C \backslash N_0$, we can extract a finite subcover $(N_1, \ldots, N_{K-1})$. Rename $\Omega \backslash C$ as $N_K$. Then $\Omega = N_0 \cup \left(\bigcup_{i=1}^K N_i\right)$, and $\forall i \geq 1, \mathbb{E}_{f_{\theta_0}(X)}[Z(N_i, X)] > 0$.

Let us denote by $X_\infty$ an infinite sequence of observations $(X_1, X_2, \ldots)$. Then

$$\begin{aligned}
&\left\{ X_\infty \middle| \limsup_{n\to\infty} \|\widehat{\theta}_n(X_1,\ldots,X_n) - \theta_0\| \geq \epsilon \right\} \\
&\subseteq \bigcup_{i=1}^{K} \left\{ X_\infty \middle| \widehat{\theta}_n(X_1,\ldots,X_n) \in N_i \text{ infinitely often} \right\} \\
&\subseteq \bigcup_{i=1}^{K} \left\{ X_\infty \middle| \inf_{\theta\in N_i} \frac{1}{n}\sum_{j=1}^{n} \log\frac{f_{\theta_0}(X_j)}{f_\theta(X_j)} \leq 0 \text{ infinitely often} \right\} \\
&\subseteq \bigcup_{i=1}^{K} \left\{ X_\infty \middle| \frac{1}{n}\sum_{j=1}^{n} Z(N_i, X_j) \leq 0 \text{ infinitely often} \right\} \quad .
\end{aligned}$$

But $\forall i \geq 1, \mathbb{E}_{f_{\theta_0}(X)}[Z(N_i, X)] > 0$. Therefore, by the strong law of large number,

$$\forall i, \mathbb{P}\left( \left\{ X_\infty \middle| \frac{1}{n}\sum_{j=1}^{n} Z(N_i, X_j) \leq 0 \text{ infinitely often} \right\} \right) = 0\,, \text{ and}$$

$$\mathbb{P}\left( \bigcup_{i=1}^{K} \left\{ X_\infty \middle| \frac{1}{n}\sum_{j=1}^{n} Z(N_i, X_j) \leq 0 \text{ infinitely often} \right\} \right) = 0 \quad . \qquad \square$$

**Lemma 1.4.14.** *Assume that $f_\theta(X)$ is continuous at $\theta$ for every $\theta$, almost surely for every $X$ according to $f_{\theta_0}(X)$. Then the condition $\mathbb{E}_{f_{\theta_0}}[Z(N_\theta, X)] > 0$ in Theorem 1.4.13 can be changed to $\mathbb{E}_{f_{\theta_0}}[Z(N_\theta, X)] > -\infty$.*

*Proof.* Let $N_\theta^{(k)}$ be a closed ball with radius at most $\frac{1}{k}$ included in $N_\theta$.We therefore have $N_\theta^{(k+1)} \subseteq N_\theta^{(k)} \subseteq N_\theta$, $\bigcap_{i=1}^{\infty} N_\theta^{(k)} = \{\theta\}$ and $Z\left(N_\theta^{(k)}, X\right)$ is increasing with $k$ for every $X$. Since $N_\theta^{(k)}$ is a compact set, and $f_\theta(X)$ is continuous at $\theta$ for every $\theta$, there is a $\theta_k \in N_\theta^{(k)}$ such that $Z\left(N_\theta^{(k)}, X\right) = \frac{f_{\theta_0}(X)}{f_{\theta_k}(X)}$. As $\theta_k \to \theta$,

$$\lim_{k\to\infty} Z(N_\theta^{(k)}, X) = \log\frac{f_{\theta_0}(X)}{f_\theta(X)} \quad .$$

If $\mathbb{E}_{f_{\theta_0}}[Z(N_\theta, X)] < 0$, we use Fatou's lemma for every $\theta \neq \theta_0$ to get

$$\begin{aligned}
\liminf_{k\to\infty} \mathbb{E}_{f_{\theta_0}}\left[Z(N_\theta^{(k)}, X)\right] \geq & \mathbb{E}_{f_{\theta_0}}\left[\lim_{k\to\infty} Z(N_\theta^{(k)}, X)\right] \\
\geq & \mathbb{E}_{f_{\theta_0}}\left[\log \frac{f_{\theta_0}(X)}{f_\theta(X)}\right] \\
\geq & -\int \log\left(\frac{f_\theta(X)}{f_{\theta_0}(X)}\right) f_{\theta_0}(X)dX \\
> & -\log\left(\int \frac{f_\theta(X)}{f_{\theta_0}(X)} f_{\theta_0}(X)dX\right) \\
> & \log(1) \\
\liminf_{k\to\infty} \mathbb{E}_{f_{\theta_0}(X)}\left[Z(N_\theta^{(k)}, X)\right] > & 0
\end{aligned}$$

We can now choose $k^*(\theta)$ and take the open ball of center $\theta$ and radius at most $\frac{1}{k^*(\theta)}$ such that $\mathbb{E}_{f_{\theta_0}(X)}\left[Z(N_\theta^{(k^*(\theta))}, X)\right] > 0$. □

**Example 1.4.7:** Let $X_j$ be a random variable with an exponential law of parameter $\theta_0$. We want to show that the MLE is consistent in such a case. The hard work to use Theorem 1.4.13 is to verify the assumptions.

$f_\theta(X) = \theta e^{-X\theta}$ is continuous for every $\theta$ and $X$.

Let $N \in \mathbb{N}$ and $\theta \neq \theta_0$. We set $N_\theta = \left[\theta - \frac{1}{N}; \theta + \frac{1}{N}\right]$. Then

$$Z(N_\theta, X) = \begin{cases}
\log(\theta_0) - X\theta_0 - \log\left(\theta + \frac{1}{N}\right) + X\left(\theta + \frac{1}{N}\right) & \text{if} \quad x \leq \frac{1}{\theta + \frac{1}{N}} \\
\log(\theta_0) - X\theta_0 - \log\left(\frac{1}{X}\right) + X\frac{1}{X} & \text{if} \quad \frac{1}{\theta + \frac{1}{N}} \leq x \leq \frac{1}{\theta - \frac{1}{N}} \\
\log(\theta_0) - X\theta_0 - \log\left(\theta - \frac{1}{N}\right) + X\left(\theta - \frac{1}{N}\right) & \text{if} \quad \frac{1}{\theta - \frac{1}{N}} \leq x
\end{cases}$$

and

$$\begin{aligned}
\mathbb{E}_{f_{\theta_0}}[Z(N_\theta, X)] \geq & \log(\theta_0) - \theta_0 \mathbb{E}_{f_{\theta_0}}[X] - \log\left(\theta + \frac{1}{N}\right) + \left(\theta - \frac{1}{N}\right)\mathbb{E}_{f_{\theta_0}}[X] \\
\geq & \log(\theta_0) - 1 - \log\left(\theta + \frac{1}{N}\right) + \frac{\theta - \frac{1}{N}}{\theta_0}
\end{aligned}$$

$\mathbb{E}_{f_{\theta_0}}[Z(N_\theta, X)] \geq -\infty \qquad$ for $N$ big enough.

Finally, let $C = \left[\frac{1}{N}; N\right]$. Then:

$$Z(\Omega \backslash C, X) = \begin{cases}
\log(\theta_0) - X\theta_0 - \log\frac{1}{X} + X\frac{1}{X} & \text{if} \quad x \leq \frac{1}{N} \\
\log(\theta_0) - X\theta_0 - \log N + XN & \text{if} \quad \frac{1}{N} \leq x \leq \frac{2N\log N}{N^2 - 1} \\
\log(\theta_0) - X\theta_0 - \log\frac{1}{N} + X\frac{1}{N} & \text{if} \quad \frac{2N\log N}{N^2-1} \leq x \leq N \\
\log(\theta_0) - X\theta_0 - \log\frac{1}{X} + X\frac{1}{X} & \text{if} \quad N \leq x
\end{cases}$$

and

$$\begin{aligned}
\mathbf{E}_{f_{\theta_0}}\left[Z(\Omega\backslash C, X)\right] \geq & \log(\theta_0) - \theta_0 \mathbf{E}_{f_{\theta_0}}\left[X\right] \\
& + \int_0^{\frac{1}{N}} \log(X)\theta_0 e^{-\theta_0 X} dX - \log N \int_{\frac{1}{N}}^{\frac{2N\log N}{N^2-1}} \theta_0 e^{-\theta_0 X} dX \\
& + \log N \int_{\frac{2N\log N}{N^2-1}}^{N} \theta_0 e^{-\theta_0 X} dX + \log N \int_N^{\infty} \theta_0 e^{-\theta_0 X} dX \\
\geq & \log(\theta_0) - 1 + \theta_0 e^{-\frac{\theta_0}{N}} \int_0^{\frac{1}{N}} \log X dX - \theta_0 \log N \int_{\frac{1}{N}}^{\frac{2N\log N}{N^2-1}} dX \\
& + \log N \mathbb{P}_{\theta_0}\left(X \geq \frac{2N\log N}{N^2-1}\right) \\
\mathbf{E}_{f_{\theta_0}}\left[Z(\Omega\backslash C, X)\right] \geq & \log(\theta_0) - 1 + \theta_0 \left(\frac{1}{N}\log(\frac{1}{N}) - \frac{1}{N}\right) \\
& - \theta_0 \log(N) \left(\frac{2N\log N}{N^2-1} - \frac{1}{N}\right) \\
& + \log N \mathbb{P}_{\theta_0}\left(X \geq \frac{2N\log N}{N^2-1}\right)
\end{aligned}$$

$$\begin{aligned}
\mathbf{E}_{f_{\theta_0}}\left[Z(N_\theta, X)\right] \geq & \log(\theta_0) - 1 - \theta_0 \left(\frac{1}{N} + \frac{2N\log N}{N^2-1}\right) \\
& + \log N \mathbb{P}_{\theta_0}\left(X \geq \frac{2N\log N}{N^2-1}\right) \\
\mathbf{E}_{f_{\theta_0}}\left[Z(N_\theta, X)\right] \geq & 0 \qquad \text{for } N \text{ big enough.}
\end{aligned}$$

We have verified the conditions of Theorem 1.4.13.

We know state the following theorem, showing that under some mild regularity conditions, maximum likelihood estimators are asymptotically normal, with a covariance matrix equal to the inverse of the Fisher information matrix. We will give no rigorous proof of the theorem, as it is quite long and complicated and can be found in most textbooks. But we will give the flow of ideas.

**Theorem 1.4.15.** *Let $(X_1, \ldots, X_N)$ be i.i.d. random variables, each with density $f_{\theta_0}$, and let $\widehat{\theta}_N$ be an MLE. Assume that $\widehat{\theta}_n$ is consistent, and that the density $f_\theta(X)$ has continuous second partial derivatives with respect to $\theta$, and that differentiation can be passed under the integral sign. Assume that the Fisher information matrix $\mathcal{I}_f(\theta)$ is finite and non-singular. Assume that there exists $K_r(X, \theta)$ such that*

1. $\forall \phi, k, j \sup_{\|\theta-\phi\|\leq r} \left| \frac{\partial^2}{\partial\phi_j \partial\phi_k} \log\left(f_\phi(X)\right) - \frac{\partial^2}{\partial\theta_j \partial\theta_k} \log\left(f_\theta(X)\right)\right) | \leq K_r(X, \phi)$
2. $\forall \phi \lim_{r\to 0} \mathbb{E}_{f_\phi(X)}\left[K_r(X, \phi)\right] = 0$

*Then under $\mathbb{P}_{\theta_0}$, $\sqrt{n}\left(\widehat{\theta}_n - \theta_0\right)$ converges in distribution to $\mathcal{N}\left(0, \mathcal{I}_f^{-1}(\theta_0)\right)$.*

**Sketch of the proof** Let $\mathbf{X} = (X_1, \ldots, X_N)$ and $l'_\theta(\mathbf{X})$ denote the gradient of $\frac{\mathcal{L}^*(\mathbf{X})}{N}$. Then one can show that $l'_{\widehat{\theta}_N}(\mathbf{X}) = o_P\left(\frac{1}{\sqrt{N}}\right)$.

Using a first-order Taylor expansion, we can get that

$$l'_{\theta_0}(\mathbf{X}) + B_N\left(\widehat{\theta}_N - \theta_0\right) = o_P\left(\frac{1}{\sqrt{N}}\right) \quad ,$$

where the column j of $B_N$ is $\left(\frac{\partial l'_\theta(\mathbf{X})}{\partial \theta_j}\bigg|_{\theta=\theta^*_{N,j}}\right)$ for some $\theta^*_{N,j}$ between $(\theta_0)_j$ and $(\widehat{\theta}_N)_j$.

Next step is to show that $\mathbf{E}_{f_{\theta_0}}\left[l'_{\theta_0}(\mathbf{X})\right] = 0$, $\sqrt{N}l'_{\theta_0}(\mathbf{X}) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \mathcal{I}_f(\theta_0))$ and $\sqrt{N}l'_{\theta_0}(\mathbf{X}) = O_P(1)$. Similarly, $B_N = -\mathcal{I}_f(\theta_0) + C_N$, with $C_N = o_p(1)$.

Then $C_N(\widehat{\theta}_N - \theta_0) = o_p(\frac{1}{\sqrt{N}})$. It is enough to see that

$$\sqrt{N}l'_{\theta_0}(\mathbf{X}) - \mathcal{I}_f(\theta_0)\sqrt{N}(\widehat{\theta}_N - \theta_0) = o_p(1) \quad ,$$

and we get that

$$-I_f(\theta_0)\sqrt{N}(\widehat{\theta}_N - \theta_0) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \mathcal{I}_f(\theta_0)) \quad ,$$

which is enough to conclude since multiplication by a matrix is a continuous function. □

### 1.4.4 Expectation-Maximization (E-M) algorithm

#### Heuristic idea and toy example

The Expectation-Maximisation algorithm is one of the algorithms that allows one to numerically compute the maximum likelihood estimator when a straightforward maximization is difficult. The E-M algorithm is especially powerful when some data is missing or unobserved and the likelihood of the whole data, including the missing part, is easy to maximise. The heuristic idea is then to estimate (estimation step) the missing data based on current parameter estimation and the observed data, and to update the parameters (maximisation step) according to the observed data and the estimated missing data. In most cases, iterating these two steps can be proved to converge to some stationary point of the likelihood. A more complete presentation can been found in [MK08].

**Example 1.4.8:** Let us take a simple model taken from genetic studies, where four outcomes (phenotypes) are possible for each observation (organism), with respective i.i.d. probabilities of $\frac{\theta}{4}$, $\frac{1-\theta}{4}$, $\frac{1-\theta}{4}$ and $\frac{1}{2} + \frac{\theta}{4}$, for some parameter $\theta$ (representing the probability of one gene to be expressed). Let $x_i$ denote the number of observations of type $i$, and $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$.

Then the likelihood of observation $\boldsymbol{x}$ is

$$\mathcal{L}_\theta(\boldsymbol{x}) = \frac{(x_1 + x_2 + x_3 + x_4)!}{x_1!x_2!x_3!x_4!}\left(\frac{\theta}{4}\right)^{x_1}\left(\frac{1-\theta}{4}\right)^{x_2+x_3}\left(\frac{1}{2} + \frac{\theta}{4}\right)^{x_4} \quad .$$

This expression is simple enough and could be directly maximised over $\theta$. But we will

apply however the E-M algorithm, for a better understanding.

Let us divide the last case in two different cases, with probability $\frac{\theta}{4}$ and $\frac{1}{2}$. If $y_i$ denotes the number of observations of type $i$ after this division, then $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ is a reduced data of $\boldsymbol{y} = (y_1, y_2, y_3, y_4, y_5)$, where $x_4 = y_4 + y_5$ and $x_i = y_i$ otherwise.

The log-likelihood of the total data $\boldsymbol{y}$ is

$$\begin{aligned}\mathcal{L}^*_\theta(\boldsymbol{y}) = &\log\left((y_1+y_2+y_3+y_4+y_5)!\right) - \sum_{i=1}^{5}\log\left(y_i!\right)\\ &+ (y_1+y_4)\log\frac{\theta}{4} + (y_2+y_3)\log\frac{1-\theta}{4} - y_5\log 2 \quad .\end{aligned}$$

If one knows the full data $\mathbf{y}$, it is easy to maximise the likelihood:

$$\begin{aligned} & & \frac{\partial \mathcal{L}^*_\theta(\boldsymbol{y})}{\partial\theta} &= 0\\ \Rightarrow & & \frac{y_1+y_4}{\theta} - \frac{y_2+y_3}{1-\theta} &= 0\\ \Rightarrow & & \theta &= \frac{y_1+y_4}{y_1+y_2+y_3+y_4} \quad .\end{aligned}$$

All what is left to do is now to estimate $y_4$ and $y_5$. If we know $x_4 = y_4 + y_5$, and we currently estimate the parameter to be $\theta^{(p)}$, then the natural (and maximum likelihood) estimate for $y_4^{(p)}$ is $y_4^{(p)} = x_4 \frac{\frac{\theta^{(p)}}{4}}{\frac{\theta^{(p)}}{4}+\frac{1}{2}}$.

In this example, the E-M algorithm is the following:

1. Choose one random value $\theta^{(0)}$ ;
2. Compute $y_4^{(p)} = x_4 \frac{\frac{\theta^{(p)}}{4}}{\frac{\theta^{(p)}}{4}+\frac{1}{2}}$ ;
3. Compute $\theta^{(p+1)} = \frac{y_1+y_4^{(p)}}{y_1+y_2+y_3+y_4^{(p)}}$;
4. Loop to step 2 until convergence.

For the numerical example, assume we have 197 observations, and the following data $\mathbf{x} = (34, 18, 20, 125)$. The maximum likelihood estimator is $\widehat{\theta} = 0.6268214980$.

Starting from $\theta^{(0)} = 0.5$, and using E-M algorithm, we get:

| p | $\theta^{(p)}$ | $\theta^{(p)} - \widehat{\theta}$ | $\frac{\theta^{(p+1)}-\widehat{\theta}}{\theta^{(p)}-\widehat{\theta}}$ |
|---|---|---|---|
| 0 | 0.5 | 0.126821498 | 0.1465 |
| 1 | 0.608247423 | 0.018574075 | 0.1346 |
| 2 | 0.624321051 | 0.002500447 | 0.1330 |
| 3 | 0.626488879 | 0.000332619 | 0.1328 |
| 4 | 0.626777323 | 0.000044176 | 0.1328 |
| 5 | 0.626815632 | 0.000005866 | 0.1328 |
| 6 | 0.626820719 | 0.000000779 | 0.1335 |
| 7 | 0.626821395 | 0.000000104 | 0.1346 |
| 8 | 0.626821484 | 0.000000014 | |

**Algorithm**

Let $\boldsymbol{X} = (X_1, \ldots, X_N)$ be i.i.d. observations according to a density $f_{\theta_0} \in \mathcal{D}_\theta$. $\boldsymbol{X}$ will be called the incomplete data. Let $\boldsymbol{Y} = (y_1, \ldots, y_K)$ denote additional data. Assume that for each $\theta$, there is a density $f_\theta(\boldsymbol{X}, \boldsymbol{Y})$ such that $f_\theta(\boldsymbol{X}) = \int f_\theta(\boldsymbol{X}, \boldsymbol{Y})\, d\boldsymbol{Y}$, and let of density $f_\theta(\mathbf{Y}|\mathbf{X}) = \frac{f_\theta(\mathbf{X},\mathbf{Y})}{f_\theta(\mathbf{X})}$ denote the conditional density of the missing data given the incomplete data, for any parameter $\theta$.

Then one can approximate the log-likelihood $\mathcal{L}^*_\phi(\boldsymbol{X})$ with the expected complete log-likelihood $Q_{\boldsymbol{X}}(\phi|\theta) = \mathbb{E}_{f_\theta(\boldsymbol{Y}|\boldsymbol{X})}[\log f_\phi(\boldsymbol{X}, \boldsymbol{Y})]$.

The E-M algorithm is the following:

**E-M Algorithm:** *Take any random value for $\theta^{(0)}$ and iterate the following for each step $k$:*

- *Expectation Step: Compute $Q_{\boldsymbol{X}}\left(\theta|\theta^{(k)}\right)$*
- *Maximisation Step: Compute $\theta^{(k+1)} = \operatorname{argmax}_\theta Q_{\boldsymbol{X}}\left(\theta|\theta^{(k)}\right)$*

**Properties**

The E-M algorithm has several useful properties, ensuring that the likelihood can only increase at each step, that any fixed point of the algorithm is a stationary point of the likelihood, and that under some mild conditions, the algorithm will converge.

**Theorem 1.4.16.** *Let $\boldsymbol{X} = (X_1, \ldots, X_N)$ be i.i.d. observations according to a density $f_{\theta_0} \in \mathcal{D}_\theta$. Let $\boldsymbol{Y} = (y_1, \ldots, y_K)$ denote the additional data. Let $\left(\theta^{(k)}\right)_{k\in\mathbb{N}}$ denote the sequence of parameter estimations from the E-M algorithm.*

*Then $\mathcal{L}_{\theta^{(k+1)}}(\boldsymbol{X}) \geq \mathcal{L}_{\theta^{(k)}}(\boldsymbol{X})$, with equality iff $Q_{\boldsymbol{X}}\left(\theta^{(k+1)}|\theta^{(k)}\right) = Q_{\boldsymbol{X}}\left(\theta^{(k)}|\theta^{(k)}\right)$ and $f_{\theta^{(k+1)}}(\boldsymbol{Y}|\boldsymbol{X}) = f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})$ almost everywhere.*

*Proof.* For any parameter value $\phi$ and data $\boldsymbol{X}$ and $\boldsymbol{Y}$, we have

$$f_\phi(\boldsymbol{X}, \boldsymbol{Y}) = f_\phi(\boldsymbol{Y}|\boldsymbol{X}) \times f_\phi(\boldsymbol{X}) \quad . \tag{1.32}$$

Taking the log, then the expectation with respect to the density $f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})$, we get that:

$$\mathcal{L}^*_\phi(\boldsymbol{X}) = Q_{\boldsymbol{X}}\left(\phi|\theta^{(k)}\right) - \mathbf{E}_{f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})}\left[\log f_\phi(\boldsymbol{Y}|\boldsymbol{X})\right] \tag{1.33}$$

and

$$\begin{aligned}\mathcal{L}^*_{\theta^{(k+1)}}(\boldsymbol{X}) - \mathcal{L}^*_{\theta^{(k)}}(\boldsymbol{X}) = Q_{\boldsymbol{X}}\left(\theta^{(k+1)}|\theta^{(k)}\right) - Q_{\boldsymbol{X}}\left(\theta^{(k)}|\theta^{(k)}\right) \\ + \mathbf{E}_{f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})}\left[\log f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})\right] - \mathbf{E}_{f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})}\left[\log f_{\theta^{(k+1)}}(\boldsymbol{Y}|\boldsymbol{X})\right] \quad .\end{aligned}$$

By definition of $\theta^{(k+1)}$, $Q_{\boldsymbol{X}}\left(\theta^{(k+1)}|\theta^{(k)}\right) \geq Q_{\boldsymbol{X}}\left(\theta^{(k)}|\theta^{(k)}\right)$. And Jensen's inequality is sufficient to prove that $\mathbf{E}_{f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})}\left[\log f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})\right] \geq \mathbf{E}_{f_{\theta^{(k)}}(\boldsymbol{Y}|\boldsymbol{X})}\left[\log f_\phi(\boldsymbol{Y}|\boldsymbol{X})\right]$ for any $\phi$, with equality only if both densities are equal almost everywhere. □

**Corollary 1.4.17.** *If the likelihood of the data $\boldsymbol{X}$ can be bounded above, then the sequence $\left(\mathcal{L}_{\boldsymbol{X}}\left(\theta^{(k+1)}\right)\right)_{k\in\mathbb{N}}$ converges.*

**Corollary 1.4.18.** *Assume that $\boldsymbol{X}$ admits a unique maximum likelihood $\widehat{\theta}$. Then $\widehat{\theta}$ is a fixed point for E-M algorithm.*

**Theorem 1.4.19.** *We use the same notations and assumptions as in theorem 1.4.16. Let $\theta^*$ denote a fixed point of E-M algorithm. Assume that the functions $\phi \to Q_{\boldsymbol{X}}(\phi|\theta^*)$ and $\phi \to \mathbf{E}_{f_{\theta^*}(\boldsymbol{Y}|\boldsymbol{X})}\left[\log f_\phi(\boldsymbol{Y}|\boldsymbol{X})\right]$ are differentiable. Then $\phi \to \mathcal{L}^*_\phi(\boldsymbol{X})$ is differentiable at $\theta^*$ and every partial derivative is zero at this point.*

*Proof.* Equation (1.33) proves that the log-likelihood is differentiable at $\theta^*$. As $\theta^*$ is a maximum for both $\phi \to Q_{\boldsymbol{X}}(\phi|\theta^*)$ and $\phi \to \mathbf{E}_{f_{\theta^*}(\boldsymbol{Y}|\boldsymbol{X})}\left[\log f_\phi(\boldsymbol{Y}|\boldsymbol{X})\right]$, both functions have partial derivatives equal to zero at that point, and it is the same for the log-likelihood. □

**Theorem 1.4.20.** *We use the same notations and assumptions as in theorem 1.4.16. Assume further that $\mathcal{L}_{\boldsymbol{X}}(\theta^{(k)})$ is bounded or converges, and that there is $\alpha > 0$ such that for any $k$, $Q_{\boldsymbol{X}}\left(\theta^{(k+1)}|\theta^{(k)}\right) - Q_{\boldsymbol{X}}\left(\theta^{(k)}|\theta^{(k)}\right) \geq \alpha\|\theta^{(k+1)} - \theta^{(k)}\|$. Then the sequence $\theta^{(k)}$ converges.*

*Proof.* $\mathcal{L}_{\boldsymbol{X}}(\theta^{(k)})$ is an increasing bounded sequence, therefore converging.

Let $\epsilon > 0$. $\exists N$, such that $\forall n \geq N$, $\forall p \geq 0$, $\mathcal{L}_{\boldsymbol{X}}(\theta^{(n+p)}) - \mathcal{L}_{\boldsymbol{X}}(\theta^{(n)}) = \sum_{i=1}^{p} \mathcal{L}_{\boldsymbol{X}}(\theta^{(n+i)}) - \mathcal{L}_{\boldsymbol{X}}(\theta^{(n+i-1)}) < \epsilon$.

Theorem 1.4.16 shows that for any $k$, $\mathcal{L}_{\boldsymbol{X}}(\theta^{(k+1)}) - \mathcal{L}_{\boldsymbol{X}}(\theta^{(k)}) \geq Q_{\boldsymbol{X}}\left(\theta^{(k+1)}|\theta^{(k)}\right) - Q_{\boldsymbol{X}}\left(\theta^{(k)}|\theta^{(k)}\right)$. We get then that $\sum_{i=1}^{p} Q_{\boldsymbol{X}}\left(\theta^{(k+1)}|\theta^{(n+i)}\right) - Q_{\boldsymbol{X}}\left(\theta^{(k)}|\theta^{(n+i-1)}\right) < \epsilon$, which leads to $\alpha\sum_{i=1}^{p}\|\theta^{(n+i)} - \theta^{(n+i-1)}\| < \epsilon$ and $\|\theta^{(n+p)} - \theta^{(n)}\| < \frac{\epsilon}{\alpha}$.

The sequence $\theta^{(k)}$ is Cauchy, therefore converging. □

The last theorem we present here specifies at which speed the sequence of estimated parameters converges to the final value. More specifically, it shows that the difference between the current estimation and the limit is multiplied by a constant at each step: convergence is exponentially fast. We refer to [MK08] for the proof.

**Theorem 1.4.21** (Speed of convergence)**.** *We keep the same notations. Assume further that the sequence $\theta^{(k)}$ converges toward some value $\theta^*$. Assume also that each matrix $\frac{\partial^2 Q_{\boldsymbol{X}}(\phi|\psi)}{\partial \phi^2}\Big|_{(\theta^{(k+1)},\theta^{(k)})}$ is definite negative, bounded away from zero uniformly in $k$.*

*Then we have that $\frac{\partial^2 Q_{\boldsymbol{X}}(\phi|\psi)}{\partial \phi^2}\Big|_{(\theta^*,\theta^*)}$ is definite negative, and that*

$$\theta^{(k+1)} - \theta^* = \frac{\partial^2 \boldsymbol{E}_{f_\phi(\boldsymbol{Y}|\boldsymbol{X})}\left[\log f_\psi\left(\boldsymbol{Y}|\boldsymbol{X}\right)\right]}{\partial \psi^2}\bigg|_{(\theta^*,\theta^*)} \left[\frac{\partial^2 Q_{\boldsymbol{X}}\left(\phi|\psi\right)}{\partial \phi^2}\bigg|_{(\theta^*,\theta^*)}\right]^{-1} \left(\theta^{(k)} - \theta^*\right) + o\left(\theta^{(k)} - \theta^*\right) + o\left(\theta^{(k+1)} - \theta^*\right) \quad .$$

### 1.4.5 Design of Experiment

Given a parametric distribution family and a sample, parametric inference focuses on, determining the "true" parameters that match the sample. Design of experiment (or, depending on the context, survey of sampling) is a complementary approach. It aims at characterizing experiments which, within some constraints, will lead to the best samples and parametric distribution families for inference. Here, the quality is measured in terms of bias and variance of the final estimator. In other words, design of experiments studies how one can shift a little bit an experiment in order to get more exploitable results. There is little theory of design of experiment, outside the theory of inference. This is more a case by case practice, where each case depends on the particular constraints and objectives of the experiment. However, the following elements are often useful:

- *repetition*: repeating independently the experiences allows one to reduce the bias of the measurements;
- *blocking*: this corresponds to grouping some elements, in order to remove unneeded random effects;
- *factorization*: exploring the effect of different factors can usually lead to better results than methods that explore one factor at the time. This effect is particularly clear in the example that follows.

The subject of design of experiment for communication networks has been partially explored in [Par09].

Design of experiment can be illustrated by the following example:

**Example 1.4.9:** Consider eight objects of weights $(w_1, w_2, \ldots, w_8)$, that have to be estimated using a pan balance. One might use the balance only eight times, and each weighing has a normal random error, of null mean and variance $\sigma^2$. It is possible to put any combination of objects on each pan, provided that no object is present simultaneously on the two pans. A natural way to proceed would be to use the weighting $i$ to measure $w_i$, leading to an estimation with independent white Gaussian error of variance $\sigma^2$ on each weight. Another possibility is to use combination of objects, as follows:

| Experiment | Left pan | Right pan |
|---|---|---|
| 1 | (1 2 3 4 5 6 7 8) | Empty |
| 2 | (1 3 4 8) | (2 5 6 7) |
| 3 | (1 2 5 8) | (3 4 6 7) |
| 4 | (1 6 7 8) | (2 3 4 5) |
| 5 | (2 3 7 8) | (1 4 5 6) |
| 6 | (3 5 6 8) | (1 2 4 7) |
| 7 | (2 4 6 8) | (1 3 5 7) |
| 8 | (4 5 7 8) | (1 2 3 6) |

Let $X_i$ denote the (possibly negative) result of experiment $i$, *i.e.* the weight that must be added to the right pan to obtain balance. We then have the following estimators:

$$\widehat{w}_1 = \frac{X_1 + X_2 + X_3 + X_4 - X_5 - X_6 - X_7 - X_8}{8}$$
$$\widehat{w}_2 = \frac{X_1 - X_2 + X_3 - X_4 + X_5 - X_6 + X_7 - X_8}{8}$$
$$\dots$$
$$\widehat{w}_8 = \frac{X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + X_7 + X_8}{8} \quad .$$

They are unbiased estimator, and their variance is $\frac{\sigma^2}{8}$. This means that using a new combination of objects, we have been able to reduce the estimation error. However, it must be said that the error for different objects is no more independent.

## 1.5 Network measurements

### 1.5.1 Communication networks measurement

The ability to measure any computer system is vital for a variety of functions including troubleshooting, managing, optimizing and forecasting, and communication networks are no exception for it. The impressive growth of communication networks in general and of the Internet in particular in the last two decades, paired with the globalization of economy and cultural exchange, has lead to an increasing need of networks measurement. This need is shared by most actors of today's network:

- network operators: these are the companies which run the networks. They need to manage their network, and detect and localize possible faulty links to repair them (ideally before they impact client service). They also need to evaluate the load on their network, in order to provision it and eventually upgrade links. Finally, knowing the main features of the traffic on their network allows them to optimize the network for these features.

- online service providers: we mean here the many organizations that either propose services that make sense only on a network (*e.g.* Google), or heavily rely on a net-

work to propose a service that is not intrinsically needing a network (*e.g.* Amazon). Note that this may include vital service for a country, such as the bank system or public administration. These organizations need to be able to measure both qualitatively and quantitatively their access to the network, in order to make sure that they are available on their target network or can use the network on which they rely on. The massive cyberattack in 2007 on Estonia [34] is a clear example of the impact that a network collapse can have on a broad scale, and why organizations or countries reliant on a network need to measure it. The rise of business over the Internet in the recent years and the deployment of cloud computing and distributed network services[35] used by big companies increase the need to watch and measure the Internet, and in a more general manner, the development of the current society of information and communication extends this need to many other networks. Communication networks in general, and the Internet in particular, now play a vital role in western countries economy and administration, and the recent history suggests that the importance of this role will still increase in the coming years.

- individual end clients: these are the users of the Internet in a non professional way, including all the families that pay for ADSL (or similar) lines. Whilst they usually do not absolutely need the network, they pay for it, and are often in a situation where it is difficult for them to evaluate what they get in exchange of their fee, or even to determine whether their problems of connection are related to their Internet Service Provider (ISP) or not.

The distributed nature of the Internet makes it even more difficult to measure it: the Internet is the interconnection of many independent networks, and most paths involve several different sub-networks. In addition, the operators managing these different networks are often competing firms, and hence do not always cooperate above the required minimum. Finally, from the fundamental design of the Internet, Internet routers are not aware about which data they carry, and the endhosts ignore which routers are responsible of their messages, or even where or precisely when losses or queueing delays occur. All these factors contribute to the fundamental difficulty of Internet measurements. In [LC06], Laskowsky and Chuang claim that the distributed nature of the Internet also increases the need of measurement: the fact that network operators are collectively and inseparably responsible of bad performance incites them to free-ride on the others' investment. This lack of incentive to invest (or ensure high performance paths) can be countered only with strong measurement techniques, which are able to precisely localize the faulty links and network operators.

[34] Estonia had adopted a paperless operation system based on the Internet for many of its administration, and hence was highly vulnerable to such an attack.

[35] Many companies now do not anymore store their data on local servers, but use servers from server farms that are located kilometers away from their offices. Applications may even be run on these far-away servers, and the local computer are used just as distant input and output terminals. Google services, such as Gmail, Google docs and Picasa are a similar concept, applied to individuals.

The question of network (and Internet) measurement is not a new one. Some communities have arisen to discuss these questions, including the ACM Internet Measurement Conference (IMC) and the IP Performance Metrics (ippm) working group of the Internet Engineering Task Force (IETF, [iet]). These communities are not an exclusivity of scientists or network operators. There are also user associations or communities, such as *Grenouille* [gre] in France which has about one hundred of thousands of members, which rely on end-to-end measurement to determine the service levels and fairness of access actually provided by ISPs. *Grenouille* is particularly relevant for us because it's a case where people have actually felt the need of measurement and organized themselves to provide it in a primitive way. They did not subscribe to a service: they created their own (free) service because nobody was offering it.

The need for accurate and efficient network measurements is increased by the fact that measurement techniques are already deployed today on operational systems. The network operator Sprint (and many other operators) use passive measurements to monitor their network [FDL+01], they can be used in overlays [FM05] or sensor networks [MKLP05, HL04] or to predict the expected performance of connections [MIP+06].

Internet measurement techniques can be broadly classified in two different approaches: *passive measurement techniques* and *active probing tools*. Passive measurement techniques rely on link-level statistics, either measured by derivation directly on the link (*e.g.* using a DAG card [dag]) or exported by the router or network card[36]. Due to the high number of packets per second crossing any link or server today, packet sampling is often necessary (*i.e.* only a fixed proportion of packets are counted). The main difficulties of this approach are **(1)** the inversion of the sampling scheme, such as to recover the whole statistic of the data; **(2)** the reconstruction of the whole network performance (or network anomalies), based on link-level measurements; **(3)** the estimation from the network performance (or link-level statistics) of user-perceived performance.

On the other hand, active probing techniques send test packets, called probes, across the network between a set of sources and a set of receivers. The network is unaware of the probe nature of these packets, and transports them just like any other packet. The analysis of the probe data, mainly the loss and delay time series, aims to estimate network characteristics such as link capacities and server loads, to provide path bottleneck localization and characterization, and to remotely measure traffic characteristics. The interaction between the probes, the network and the rest of the traffic is crucial here.

In any measurement schemes, it is useful to distinguish the *primary* metrics and the *secondary* metrics. A primary metric is a quantity that can be directly measured. A secondary metric is a quantity that can be deduced from primary metrics measurement. Consider for example a cook who wants to determine whether a chocolate cake is baked or not. A classical method is to stab a knife into the cake, and observe whether (or how much, or how it looks) the cake dough stays on the knife. The primary metric is the presence or absence

[36]Most of today's Internet router support this function.

of dough on the knife. This is what one can observe directly. The secondary metric is to know whether the cake is baked or not. This is deduced from the primary metrics, because the cook knows that for this kind of cakes, both are equivalent. He can even maybe deduce the remaining baking time from the aspect of the dough if there is any. That is again a secondary metric. Note that this equivalence, or the inversion from a primary metric to a secondary metric, is valid only for specific cakes. It does not work for apple pies or boiled potatoes. In the case of active Internet probing, the primary metrics are easy to identify: these are the loss and delay time series[37]. They are the only quantities can directly measure, and from these, we must deduce any other quantity (*e.g.* capacity, available bandwidth or path lengths).

### 1.5.2 Internet Tomography

This end-to-end probing approach is particularly adapted to measurements from individual clients or online service providers, which do not have access to the internal networks statistics and hence can perform only very limited passive measurements. In particular, this allows us to evaluate the performance of whole paths which cross several independent networks, as this is the case in the Internet. Even if there was a contract between the Internet Service Provider and its client which would allow the client to access passive measurements, these would be limited on the contracting ISP, and would not include the whole path.

A particularly interesting paradigm is the use of the path diversity in the network. When the set of measurement points is more than a single source–destination pair, it is possible to conduct measurements on different paths and use the joint measurements to leverage quantities on the sub-paths, or even at the link level if enough path measurements are available. We will call *network tomography* any technique which uses path diversity and exclusively end-to-end measurements to leverage per-link characteristics of the network: in particular, we exclude here techniques that require that the internal routers send an ICMP echo message[38].

Except for the direct estimation of path loss rates and delays, Internet tomography uses in most cases a parametric inference approach, as seen in section 1.4.1. A model for the delay or loss series is postulated, with a few parameters allowing a "fine" tuning of the shape. This model is often postulated "a priori", based on the analysis of a few actual measurements.

**Example 1.5.1** (Medical Imaging)**:** The tomography approach can be compared to medical imaging[39]. For a long time, the only way to have a precise image of what is happening

[37] Other metrics, *e.g.* connectivity, are sometimes considered as primary metric. Whislt the distinction is not vital, we argue here that they are *in fine* in fact deduced from loss and delay time series. Two nodes are for example said to be not connected if all probe messages between them are lost, and to be connected if at least one probe message is not lost.

[38] This last requirement is sometimes omitted in the definition of tomography. Tomography is also sometimes used to denote methods that estimate *per-link* characteristics, as opposed to *per-path* characteristics.

[39] In fact, Internet tomography was named as such precisely because of this similarity with medical tomography.

inside a human body was to open it and observe directly. This is similar to the passive measurement case for networks: direct access to the location of the anomaly is needed[40].

Radiography, ultrasonography and Magnetic Resonance Imaging (MRI) use a different approach. Without going into technical details, all run on the same principles: some signal (X-rays or ultrasound waves) is sent, and interact with the human body. The modified signal is then measured at the output. Using known models for the human body[41] and the interaction it should have with the signal, it is possible to deduce whether there is something anormal or everything is sane, and the precise location of an eventual anomaly. Note that in particular, these techniques, except the basic radiography approach, use the spatial diversity to leverage more signal and get more precise information. There are more than one source and one destination, and more than one signal is sent.

Active probing techniques often suffer from two difficulties. First, except for delay or loss rate estimation, there is an *inversion* step, that is an estimator of the (aimed) secondary metric, from the input of the measured primary metric. When the measure of the primary metric is deterministic, inversion is sometimes tricky due to the numerical instability of the estimator. When the primary measurements are random variables, such as the delays of individual probe packets in a queueing network, as seen in section 1.2, the difficulty can then be increased by the randomness of the input. It then becomes impossible to find the groundtruth in each case, as we have seen in theorem 1.4.7 of section 1.4. There is an intrinsic imprecision for any estimator, due to the nature of the random system. In [Rou05], Roughan shows for example that even with perfect sampling with an infinite number of stealth non-intrusive probes of a M/M/1 queue during a time interval of size $T$, the estimation of (for example) the delay has a minimal variance. This is due to the fact that the system has a strong correlation in time, and hence, one might be "stuck" into a unlikely excursion which is far from the steady state.

The second difficulty is what we call in this dissertation *restitution*. Active probing is often described as injecting test packets into the network. These packets are *intrusive*: they will be forwarded by the routers, and as any packet, require time to be served. They hence perturb the observed system, where one is interested in the state of the *unperturbed* system. There is a need to remove the impact of the probes on the system. This step can be removed if the impact of the probes on the system is minimal[42]. This is particularly the case when the probing rate is rare enough compared to the system load, but this puts hence a strong limit on the number of probes one can use, and on their spacing. The impact is even null when, instead of adding specific probes packets, one tags some already present packets as probes,

[40] We do not mean here to say that passive measurement tools are obsolete, or anything similar to that. They are useful in some cases. But they do require a direct access to the links or servers, which is (sometimes) possible in the networking context. For medical imaging, this direct access is sometimes impossible, in most cases very intrusive, hence the method has been (nearly) abandoned in this context. We just strengthen here the fact that in both cases, the direct access to the anomaly is needed.

[41] This includes the fact that doctors empirically know the signal (or images) of sane bodies, and the images of most classical anomalies, and are capable also, based on this knowledge, to interpret many unknown images.

[42] In this case, the probing scheme will often be said *uninstrusive*, even if this is not formally the case.

and measures their delay and loss time series. As these packets would have been anyway sent at these precise times, the system is unperturbed. However, the prober is strongly restricted in the choice of the probe patterns, or in their precise timing.

### 1.5.3 Inverse problems

The notion of inverse problem stems from physics. Consider a dynamical system governed by some known evolution equation. In a direct problem, the parameters of this dynamical system are known and the goal is to calculate the associated 'trajectory'. This is the most classical approach in physics and many other sciences. In an inverse problem, one or more trajectories are observed and, using the evolution equations of the system, one tries to deduce (some of) the parameters which gave rise to those trajectories.

A typical example of an inverse problem is in acoustics. In the direct problem, the parameters could be the location and shape of some obstacle as well as some input signals with a given spatial and temporal structure. These parameters, when used together with the theory of wave propagation and scattering, allow one to determine the acoustic signal at any location and time. A classical inverse problem consists in selecting appropriate input signals, measuring the resulting acoustic signal at certain locations where such measurements are possible, and then leveraging the shape of the solution of the direct problem to determine the unknown location and shape of the obstacle.

Inverse problems are in fact ubiquitous in physics, and have well established incarnations in many other fields such as fluid dynamics and electromagnetism. They have major applications in seismology, oil detection, geophysics, medical imaging, and industrial process monitoring to quote just a few.

**Example 1.5.2** (gravity)**:** The following toy example exhibits many of the key features of inverse problems which we consider in this paper, and allows us to introduce some terminology.

A mass initially at height $y_0$ and with vertical speed $v_0$ has a trajectory given by the direct equation $y(t) = y_0 + v_0 t - gt^2$ . Assume that the initial conditions $y_0$ and $v_0$ are hidden to some observer, who can only glimpse the trajectory at n different epochs, and assume each glimpse allows the observer to make an accurate measurement, an observation, of the mass's location. Our inverse problem consists in determining the unknown parameters $y_0$ and $v_0$ from the observations. It is easy to see that if $n \geq 2$ and if the observer knows this direct evolution equation, then the observations are enough for him to determine the unknown parameters unambiguously. If $n = 1$ the observer can only infer a linear relationship between $y_0$ and $v_0$, and the inverse problem is ill posed or ambiguous, lacking a unique solution. Furthermore, if $g$ is unknown then the triple $(y_0, v_0, g)$ can also be determined from such observations, and this is in fact one of the ways for estimating local values of $g$.

Our toy example is deterministic. One obtains stochastic scenarios if random measurement errors are considered, or more fundamentally, when replacing the direct equation by

a stochastic evolution equation. The inversion problem now becomes one of statistical estimation of the unknown parameters from the observable time series.

Note that the direct equation of our toy system lives in continuous time (and space). A natural inverse problem is to determine the parameters given observations over continuous time. Instead, we consider a more difficult problem which consists in inverting for the parameters based only on a finite number of observations. Part of the great richness of inverse problems in general is that the nature of the observations may be constrained in many different ways, often corresponding to practical limitations from applications, each case demanding different solution methods. Here we will focus on discrete observations, and we distinguish two subcases:

- passive observations where the glimpse times are not controlled by the observer;
- active observations where the observer can choose when the process is glimpsed.

In the latter setting certain constraints still apply, for example often there is a fixed budget $n$ of available glimpses, or $n$ may be infinite but a fixed average observation rate is imposed. A natural question is then that of an optimal spacing of the glimpse times, for example in the sense of minimal estimation bias and variance.

Our toy example and its stochastic versions are non-intrusive in that the act of making observations did not perturb the system. A natural extension is to examine the associated intrusive or perturbation problem, for example each glimpse could add a random impulse to the motion. Would it still be possible to measure the parameters even in this case? What would the optimal trade-off between the accuracy of the estimation and the disturbance of the system? In the particular case of network active probing, probes will add load to the system, and hence increase loss rates and delays, or lower the quality of service. Whilst disturbing slightly the system is feasible, it is desirable to limit the perturbation, and even better to have an almost non-intrusive method.

Finally, the richness of inverse problems also lies in part in the fact that one might have only a partial knowledge of the direct problem. Newtonian physics is a well-known theory, which is nearly fully-solved[43]. In many cases, when systems are non-linear with a high degree, only qualitative answers can be given, or quantitative answers for only a part of the solution. It may be the case also that even if the explicit solution is not known, one can say that it satisfies a specific relation. When the problem is stochastic, full distributions or time series may not be computed in the direct problem, but specific transition of the system may however specified. In some cases, this partial knowledge will be enough for an inverse problem.

Inverse problem theory can be applied to network measurement. The direct problem is to predict the evolution of the network. In this dissertation, we will use two network theory: the queueing theory, as presented in section 1.2 and the theory of bandwidth sharing

[43] Some problems, such as the n-body problem for n greater than 3, are not solved (or even unsolvable). But much is known is Newtonian physics.

networks, which we presented in section 1.3. However, any other theory predicting the evolution of networks can be used.

The observables, *i.e.* the trajectories that one can measure, are the results of the direct problem. In the case of bandwidth sharing networks, these are the (eventually dynamical) bandwidth allocations, or any statistic based on them. In the case of queueing theory, the observables are the delay and loss series (or statistics of these, such as their moments, distribution, etc.) and buffer occupancy statistics. In this dissertation, we will focus on the Internet active probing paradigm, and hence restrict ourselves to observables that can be measured from an end-to-end point of view. However, within the IP network framework, there are many meaningful ISP-centric inverse problems, which can use internal network observables.

It is important to realize that the application of inverse problems to network measurement is not a direct measurement of the quantities we aim to estimate. Inverse problems theory estimates the parameters of the model, and not the parameters of the actual system. Furthermore, the trajectories one will observe are not trajectories of the theoretical model, but trajectories of the real network. Since neither queueing theory nor the theory of bandwidth sharing networks are perfect model for actual networks, there will be a (small) difference between the modelled and the observed trajectories. However, it the modeled trajectories used for inverse problems are "close to" the real system behaviour, the imprecision of inverse problem theory might also be small.

Inverse problems could be seen as yet another name for parametric inference of network characteristics. There are two main difference between "basic" parametric inference and network inverse problems. First, inverse problem theory is based on classical theoretical model results. The parametric distribution families on which the inference is based on corresponds to the prediction of theoretical models. This is clearly possible to state this a priori for classical inference, but many of the current tomography results are not based on theoretical model prediction. The inverse problem approach is based on this proximity between inference and direct theory, and using this name strengthens this fact. As such, it is a bridge between classical network theory and practical network measurement. Whilst the (slight) difference of behaviour between actual networks and theoretical models might lead to the fact that some of the results are not directly applicable to real networks, classical theoretical models share many key properties with real networks, and hence, the inverse problems approach can be fruitful and give interesting insight about network measurement.

Second, inverse problem theory is larger than parametric inference and includes the design of (probing) experiment. A natural question is how to get the "best" observations within some constraints, be it through a careful design of the probing signal (for example in oil detection or in medical imaging, where one can control the signal that will be sent and will interact with the probed object), or through the control of the sampling times (in the case of the gravity example, or for passive probing of a network). In the case of interest for us, both aspect will be used. Trains of probe packets can be sent to measure the network,

and their design can improve the quality of the measurement[44]. On another side, when the number of observations (or probes) is limited, the timing of probe packets or probe trains is crucial to the quality of the observation. Finally, a prober needs to take into account the effect of the probe packets onto the network, in order to both not be make it collapse and to take this perturbation into account, as the quantities that one aims to estimate are the network characteristics in absence of probes. They are tradeoff here between the increased accuracy of the estimation that additional probes will most likely allow and the increased perturbation of the system they will be responsible for. In all these aspects, inverse problem theory also aims at providing not only estimators of network characteristics, but also guidelines about how to efficiently probe a network, and which characteristics will be accessible or not through active network probing.

### 1.5.4 Bibliography

We end this introduction with a brief bibliography of known results about active network measurements. The first part quickly goes through the main steps of the active probing history, and presents existing tools for active probing. The second part focuses on the literature of active network tomography and the use of path diversity to infer network characteristics. Finally, the third section presents results that we classify as inverse problems, in that that they don't aim at providing estimators of network characteristics, but rather guidelines about what can be inferred or how one can optimally try to infer these quantities.

**A brief history of active network measurements** The very first beginning of active measurement tools is the *ping* program [Muu83], which uses ICMP Echo request[45] to measure whether an host is alive or not, and the round-trip time to this host. Van Jacobson's *traceroute* [Jac87] is another widely-used tool that sends a stream of packets with increasing Time-To-Live value to a destination host. The resulting ICMP echos allow to determine the route from the source to the destination, coupled with rough estimates of the round-trip times to each of the intermediate hops. Both of these tools are used in a daily manner by network administrators.

In [Bol93], Bolot conducted in 1993 one of the first systematic study of packet delays and losses on a single path. He used periodic UDP probes sent with the tool *NetDyn* [SB93], to characterize the behaviour of the Internet. Paxson also performed a large-scale study of routing and packet dynamics on the Internet, deploying measurement tools on many host over the Internet. In [Pax97], he used repetitive Traceroute measurement between 37 Internet sites to analyze the routing behaviour for pathological conditions, routing stability,

[44] Common examples are the use of a large train to ensure (or at least increase the likelihood) that the buffers are non-empty, or the use of back-to-back packets of different sizes, in order to explore the effect of the packet size on the delay.

[45] The Internet Control Message Protocol (ICMP) is a transport protocol, similar to TCP and UDP. It is used to carry control messages, such as error reports (in particular when a packet has exceeded its maximum number of hops, called *Time-To-Live*) or (in the case of ping) a request to generate a "host-alive" answer. Contrary to UDP and TCP, ICMP is not used to carry data, but to exchange information about the state of the network.

and routing symmetry. In [Pax99], he studied the packet dynamics of TCP transfers between 35 Internet sites.

Infrastructures have been developed in order to allow the deployment of measurement tools on many different sites. *Traceroute.org* [tra] maintains a list of servers which accept to conduct traceroute or ping measurement from their server to any destination address. The Internet End-to-End Performance Monitoring (IEPM) group, at Stanford, is monitoring connectivity and end-to-end performance for sites and universities involved in High Energy physics. These sites also allow to conduct network experiments between their sites. PlanetLab [Pla] is an overlay of Internet hosts, where institution members can develop and run their own distributed software. It has become popular to conduct Internet measurement experiences.

Many tools have been developed in the recent years with the use of active probes to measure the loss rates of links. Loss rates are difficult to measure, because losses are caused in batch by short and infrequent buffer overflows. *Sting* [Sav99] allows to measure the loss rates on a TCP path between a node and a web servers, and to distinguish between losses on the forward path and the return path. *Zing* [MPAM98] evaluates the loss rate on a single one-way path, between any pair of nodes, with user-specified probe sizes and probes rate. *Badabing* [SBDR05] and its slightly modified version *Slam* [SBDR07] use trains of probe packets to estimate the loss rates and congestion episodes. The precision of these tools can be increased with the use of longer (and more) trains, at the cost of a greater intrusiveness. *Tulip* [MSWA03] allows in addition to localize the lossy links on a one-way path, but requires 10 to 30 minutes to run, which is longer than most other tools. It does so by estimating the loss rates between the source and any node in the path.

Bandwidth also received considerable interest. The initial approach [Kes95] was to send two back-to-back probe packets, and measure their dispersion, *i.e.* the difference in their arrival time, to deduce the minimum capacity, called *bottleneck* along the path. *bprobe* [CC96], *Nettimer* [LB01] and *pipechar* [JYCA01] use such an approach. *Pathchar* [Jac97] and its variant *Bing* [Bey95] and *Pchar* [Mah99] analyze packets' Round Trip Time linearity with respect to their size, and deduce the capacity of the hop. Repetitive use of this approach allows to deduce the capacity of all links along a path. In [PV02a], Pásztor and Veitch use a similar idea on carefully constructed sequence of packets, which allows to reduce the probing overhead.

Packet pair method was also extended to trains of packets[46], in order to estimate the available bandwidth (that is the space capacity on the link) along a path. Many tools use this approach, with careful constructed trains and a slightly different analysis of the delay series. We must mention here *Cprobe* [CC96], *Pathrate* [DRM01, DRM04], *Pathload* [JD02], *PTR* [HS03], *Pathchirp* [RRB+03], *TOPP* [MBG00], *Spruce* [SKK03], *Delphi* [RCR+00] and *IGI* [HS02]. Depending on the train construction and the delay analysis, they allow to estimate either the cross-traffic intensity, the available (or sometimes achievable) bandwidth

[46]By a packet train, we mean a set of more than 2 closely spaced probe packets.

or the load (defined as the ratio between the cross-traffic intensity and the capacity) of the path.

**Internet tomography literature** The tomography literature can be broadly divided into three parts: literature speaking about loss inference, literature whose subject is the delay inference, and finally literature which focus on topology inference.

The first tomography paper [CDHT99] used multicast probes to infer the internal loss characteristics in a tree. The correlation of losses for different receivers allowed to distinguish the contribution of each link to the end-to-end loss rates. The method was evaluated on a large scale in [ABC$^+$00], and generalized to the case of partial information (where some data points are unknown) by the use of the E-M algorithm in [DHT$^+$02]. In [BDLPT02], it was generalized to no tree-shaped topologies, and sufficient and necessary conditions for identifiability of individual link loss rates are given. Losses are in practice not independent, and the Bernoulli model is an approximation. This weakness is studied in [ADV07], where the temporal loss characteristics are infered from multicast probes.

As multicast is not always feasible, unicast alternatives have been considered. Coates and Nowak *et al.* in [CN00] use pairs of closely spaced unicast packets, exploiting the strong correlation of the losses of packets in the same pair. Duffield *et al.* use a same approach in [DPPT01], with *stripes* of packets that can be larger than pairs. In [ZCB09], the authors identify the loss rates of *minimal identifiable link sequences* in a general topology, using as little assumption as possible.

Finally, loss rates have a particular pattern on the Internet: most links have a (near) zero loss rate, and losses are concentrated on a few links only. It hence makes sense to try to infer which links have a positive (or a null) loss rate, without focusing on the value of the loss rates. This is known as the *binary* (loss) tomography, and has been proposed in [Duf06]. Using the assumption that faulty links are few, they propose a *smallest consistent failure set* algorithm to identify these faulty links. [PQW02] propose 3 different algorithms for the same problem based on web-server measurement: random sampling and linear optimization algorithm also operates on the assumption that the failure probability are identical and that lossy links are uncommon, but the Bayesian inference they propose can work with any prior distribution of loss rates. Nguyen and Thiran have relaxed these assumptions: in [NT07b], they estimates prior distribution for failure rates using special properties of the boolean algebra, and localize congested links based on this prior. In [NT07a], they first estimates the variance of the link loss rates, and using the fact that in practice on the Internet, the variance increases with the loss rates and that many links have a low loss rates, they remove these un-congested links from the system until they have a full rank linear system that they can solve.

Delay tomography followed a similar path: the first delay tomography paper [DP00] exploited probe multicasting. It used non-parametric estimators, but only recovered the delay variance at each node rather than the full delay distribution. The related work [PDHT02] extended the approach to the entire distribution by discretizing delay, effectively introduc-

ing a multinomial model for each node delay, and therefore a large number of parameters. The non-parametric estimators described were based on recursive conditional independence of child subtrees and deconvolution, and have no direct link to the MLE. In [LY03] a similar multinomial delay model was taken, but a pseudo MLE approach was employed . A full MLE was avoided in order to reduce complexity.

Since multicasting is not always practically feasible, a number of works, including [CN01, TCN03, SH03] examined unicast alternatives based on a packet-pair scheme where probes are sent in as closely spaced pairs so that they will experience similar delays until a branch point is reached, after which they follow different paths. Here the likelihood is simpler as probes approximately 'multicast' over two paths only, but the packet-pair assumption introduces additional noise and a much higher probing overhead. In [LMN06], hybrid 'flexicast' combinations of unicast and multicast probing are explored in order to tradeoff estimation accuracy against computational and probing costs.

The use of discretized node delay models make tradeoffs between computational cost and accuracy difficult. A small number of papers address this, as we do, by using parametric approaches involving continuous distributions. Using unicast probing, [SH03] proposes a mixture model for node delays including Gaussian densities and an atom representing the minimum propagation delay. A penalized likelihood was adopted to control the number of Gaussians in the mixture which is maximized using an associated E-M algorithm. More recently, using multicast probing [CCB07] also employs a mixture model including a single atom, this time combined with multiple uniform and exponential densities. The analysis is performed in the transform domain with sampled characteristic functions and performs an $L_2$ based optimization using quadratic programming, which scales better than E-M to large trees. In [LMN07] a mixture model, consisting of an atom combined with a continuous density satisfying certain conditions, is considered for flexicast probing. Based on examples on simple trees, MLE based approaches are considered but then discarded as intractable in favour of moment based methods using least squares. The study is preliminary but the observations on identifiability are important.

Finally, the actual (multicast or unicast) topology is not always known or accessible. This has been considered in different papers: in [RM99], Ratsanamy *et al.* infer the multicast topology from the correlation of losses experienced by multicast probes. The bottleneck of the path is also identified from the delays. The inference is also made from delays of multicast probes in [DHPT00], and generalized to any *mark* (including significant delay, loss, or any flag) that can be added to the probes by the server they cross on their path. The covariance of multicast of probes is used in [DP04] to estimate the variance of link delays and the multicast topology. Thess different approaches are merged in a single efficient method in [DHLP01]. An penalized MLE approach is proposed in [DHPT02] based on the delay measurement of careful construct sandwich unicast probes, and a Markov Chain Monte Carlo algorithm is used to perform the maximization. Rabbat *et al.* in [RNC04] use probes sent from a pair of sources to a pair of destinations and their order of arrival to estimate (part of) the topology of a general network. An overview can be found in [CCL$^+$04], where the

authors presents an interesting focus on the scalability of the algorithms and cover mostly pseudo-likelihood approaches.

**Inverse problems in networks related work** A main question about active (and passive) probing is how one should sample a network. The first active measurements used periodic probing. The ease of sending periodic streams made it a natural choice. Paxson in [Pax97, Pax99] addressed explicitly the issue of when probes packets should be sent. He applied the "Poisson Arrivals See Time Averages (PASTA)" principle of the classical queueing theory, and advocated to use exponentially distributed interprobing time such as to have a sampling at Poisson epochs. Since then, Poisson probing has become part of the conventional wisdom of active network measurements. [MACM05] showed that in their particular experiments, Poisson probing and periodic sampling are both yielding estimations with no significant difference. The authors also note that the Poisson structure of the probe stream may not be preserved through the network. [BMVB06] proved that Poisson probing is adapted mostly when the probes perturb significantly the network. In the case of rare (or stealth) probes, they propose the "Non Intrusive Mixing Arrivals See Time Averages (NIMASTA)" rule, and advocate for a probes with interarrival uniformly distributed on a small interval around the mean, in order to avoid eventual phase locks but stay "close to" periodic sampling. In [BMVB07], the authors show that an optimal probing strategy (in terms of mean-squarred error) when the probes do not perturb the network is to use the family of Gamma renewal probing processes. In [Rou06], Roughan compares periodic and Poisson probing and shows that (near) periodic sampling usually is slightly more accurate for estimating first order statistics , but that Poisson (or irregular) sampling can be (at least in some cases) efficiently used to estimate time series properties, such as auto-correlation functions or periodicities. [HST07] use a simple two states Markov chain to show that the number of probes needed to reach a given accuracy threshold in the loss rate estimation is much higher than what a naive approach would suggest. Parker *et al.* in [PGS09] study the impact of the probing rate, using the same model.

The link with (linear) inverse problems is well explained in [CHNY02]. The complexity of the estimation algorithm is considered in [LMN06, XMN06, LMN07]. The main suggestion is to use "flexicast" experiments, that is probes sent with multicast protocol to a small set of receivers (instead of all receivers). When using enough (and well designed) experiments, this flexicast approach allows to keep the needed correlation in measurements for the same total number of probes, but allows quicker computation of estimators. The Maximum Likelihood estimator is computed, and an alternative less accurate but quicker to compute moment-based estimator is proposed. [DLM$^+$07] follows the same approach, and considers also optimal design of experiment. In particular, the question of probe allocation (with a total budget) is raised (but not solved). The authors also suggests an interesting technique to merge "expensive" but accurate measurements (probes) and cheap but error-prone measurements (traceroute) in a single estimator.

The identifiability of particular metrics is a classical question in inverse problems. Many

articles provide results in a particular case. To the best of our knowledge, the most general identifiability results can be found in [LMN06] for discrete delay distributions and [CCB07] for general delay distributions. In the particular case of binary tomography, a related question is to determine the minimal number of paths to monitor and the associated optimal placement of measurement points called *beacons* such as to ensure identifiability. [HA03] for example shows that the problem is usually NP-hard, and propose an efficient heuristic for its particular model. [NT04] consider a similar problem with a slightly different model: the placement of beacons remains a NP-hard problem.

Finally, there has been recently some work that aims at doing a link between classical queuing theory models and network measurements. [Rou05] uses a M/M/1 model to assess fundamental bounds on the accuracy of network performance measurement. When measurements are limited in a finite time frame, they can at best sample the exact trajectory of the network during that time frame, but there is always a chance that the network is not in that time frame in its stationary distribution. Liu *et al.* [LRLL04, LRL05] evaluates with a queueing-theoretic rigourous approach the dispersion of probe trains caused by the cross-traffic, and show the simplified fluid approach, on which most tools evaluating the cross-traffic intensity are based, is biaised in many cases. Machiraju *et al.* in [MVBB07] study rigorously the case of a single queue with a probabilistic queuing-theoretic treatment, and provide identifiability results for the cross-traffic distribution and proven estimators when this is possible. In [NTV06], the authors computes the distribution of the number of arrival in a M/D/1 queue between two consecutive probe packets, conditionned on the event that both belong to the same busy period, and use this distribution to evaluate the cross-traffic intensity on the queue. In a different more ISP-centric framework, [MvdM09] provides carefully justified guidelines for link dimensioning, based on measurement of the buffer occupancy. [MZ09] proposes statistical tests based on a M/G/$\infty$ model to detect changepoints in the load of a voice call system.

## 1.6 Contribution of this dissertation

**Chapter 2** We formulate extensively the end-to-end active probing techniques as inverse problems in queueing theory. Whilst this connection has been stated for a long time, we are the first as far as we know to explore it thouroughly. This formulation connects the active measurement field and queueing theory field, and is aimed mostly at the part of the community of queueing theory which is not aware of the end-to-end probing paradigm. The constraints of active probing are formulated in a queueing theoretical manner.

The different steps of active problems in queueing theory are enumerated, and we identify the main potential difficulties. We classify these inverse problems into three different categories:

1. *Analytical inverse problems*, where we assume that the traffic arrivals and the network behave as a given queueing model, use queueing theory to predict some statistics

(*e.g.* moments, distribution or series of loss events and individual delays) about the observables, and propose an analytical formula or algorithm providing the desired parameters from the observation. The particularity here is that we assume a perfect noiseless statistic of the primary metric, which could be obtained for example with infinite time series.

2. *Statistical inverse problems*, which are similar, but take into account the intrisic randomness of the queueing system; the output in this case is a statistical estimator, such as presented in section 1.4.

3. *Optimal probing strategies*, where the aim is to find inversion techniques that work or are optimal for a large class of queueing systems; the output are general guidelines, bounds, feasibility or impossibility results about inverse problems in queueing theory.

We give examples for the first and third cases, based on simple classical queueing models, and illustrate through them some properties of these inverse problems.

This chapter is based on results which have been published in [BKV09].

**Chapter 3** In this chapter, we consider in details the specific queueing model of a single path in a Kelly-type network. We first compute the mean and the distribution of probe end-to-end delays in such a network. We then show that the set of residual bandwidths (*i.e.* the difference between the capacity of a link and the sum of the cross-traffic intensities on this link) is identifiable from the mean end-to-end delays for $K$ different probing intensities, where $K$ is the length of the path. We propose an algorithm to compute this set of residual bandwidths, and give numerical applications that show that the algorithm is exact for perfect end-to-end mean delays, but is intrisically unstable when the empirical mean delays do not match perfectly the theoretical mean delays.

We then study the maximum likelihood estimator of the set of residual bandwidths, based on the family of theoretical delay distribution for each set of residual bandwidths. For $K = 2$, we present a method to compute it based on a fixed-point equation, and show that the maximum likelihood estimator is asymptotically consistent in this case. For larger paths, we compute explicitely the E-M algorithm, and show that it will converge for $K = 2$[47]. Numerical examples for different path lengths are given, on which the estimator seems to perform well and converge in a small confidence set for reasonnable number of probes.

Finally, we present preliminary simulations that empirically study the difference between this model and real networks, for networks of length $K = 1$ and $K = 2$. The simulation is based on real traffic traces from the core network of a Tier-1 ISP. We study independently the impact of each assumption of Kelly networks, compared to real networks case, and propose simple correction factor or techniques when this is needed.

[47] The classical result is the convergence of the likelihood of E-M estimates. The convergence of the estimates of the E-M algorithm usually happens in practice, but is difficult to prove theoretically. The general assumptions are restrictive, and can not be applied in our case. The proof we propose in this chapter is specific to the studied case.

The key result in this chapter is the identification of the whole set of available bandwidth based on pure end-to-end measurements. As far as we are aware of, previous techniques for the estimation of available bandwidth were:

- either relying on the cooperation of the internal servers of the networks, *e.g.* using ICMP echo replies or ICMP Time Exceeded error messages;
- either are based on dispersion techniques (or packet pairs/trains techniques[48]), which can accurately estimate the bottleneck link or the tight link of the path, but are (mostly) silent about other nodes.

The results of this chapter have been published in [BKV09] and [KBV09].

**Chapter 4** This chapter extends the results of chapter 3 to the case of a Kelly tree, with unicast cross-traffic and multicast probes from the root to the leaves of the tree. Whilst multicast is not fully deployed, the recent development of television and radio on Internet increases the realism of this model.

Using combinatorial arguments, we compute the distribution of the joint delays at all leaves for a general tree. We then propose explicit formulas for E-M algorithm, and give numerical applications on a few different trees. As one could expect, the estimation is less precise for nodes that have relative higher available bandwidth. It is also clear (at least on these examples) that the case of a single path (or more generally, of a few successice nodes with a single child) leads to less accurate estimation: the reason is that because each branching point replicates the multicast probes, it becomes "easier" to distinguish the contribution of each node to the end-to-end delay for (part of ) the trees with many branching points.

The E-M algorithm is known to suffer from its low speed. This weakness is not critical for the case of a single path, but it strongly limits the size of the trees where one can realistically computes the maximum likelihood estimator. We propose hence three acceleration techniques, which decrease the computation time by a large factor (up to $10^3$). First, for computability reasons, the E-M algorithm does not maximize at each step the (complete) likelihood, but the difference between the likelihood and a Kullback-Leibler distance. For this reason, E-M steps are smaller that what would be "optimal", and we increase their size in order to decrease the number of steps. More precisely, we double the step size as long as this operation increases the likelihood of the estimate. We hence keep the convergence properties of E-M, and find quickly an right-order estimate of the optimal step size. Second, on a few examples, it appears that E-M trajectories are roughly piecewise linear. Hence, at each step, we compute, in addition of the classical step, two steps with a slightly modified direction in order to increase this linearity with previous steps, and execute only the step which performs the best in terms of likelihood. Paired with the previous size increase

[48] Packet trains technique send carefully constructed trains of packet with precise departure time, and measure the arrival times of the packets at destination. By analizing the difference in their arrival times with respect to their size, it is possible to deduce either the service time on the tight link or the available bandwidth of the bottleneck link.

heuristic, this direction correction allows to have larger steps, and the computation time is greatly reduced. Finally, the E-M algorithm can start from any (random) point. From this start, we first find a rough estimate of the MLE by performing a few quick steps with only a (random) part of the data, before running the full precision algorithm. This allows to quickly reach the neighbourhood of the maximum likelihood estimate, and reduces the computation time.

We believe that the acceleration techniques presented in this chapter are useful outside this precise case. They can be adapted to any iterative algorithm whose trajectory is approximately piecewise linear. In the case we studied here, the key points are the agressive algorithm used to find quickly a correct order for the ideal step size, and the fact that the objective function is much quicker to compute than one algorithm step, hence allowing to check the objective value for different step sizes.

This chapter is based on the results published in [PVK10].

**Chapter 5** In this chapter, we consider inverse problems in bandwidth sharing theory, such as presented in section 1.3. Bandwidth sharing theory is less developped than queueing theory, and fewer results can be exploited. In particular, there is no explicit bandwidth allocation formula in most (including some simple) cases[49]. We show that however, given an $\alpha$-fair utility function, it is possible for two simple but generic networks (a single path with arbitrary number of hops, and a "triangle" network with 3 servers) to infer the server capacities and number of competiting flows from the allocation with different probe flows number. We conjecture that this method can be applied to any given topology. As it has been shown that for specific parameters, bandwidth sharing networks allocation corresponds to the mean allocation performed by TCP on the equivalent network, it means that in theory, it should be possible to infer the capacity of the different links and the number of competiting flows on a network, when one measure the bandwidth allocated by TCP to the (TCP) probing flows for different flow numbers. Numerical applications show that this inversion is numerically highly unstable, even with very little error in the measured bandwidth allocation.

Bandwidth sharing networks can be used to represent other objects that communication networks. The formulation is general, and can be adapted to many cases with linear constraints and general utility maximization. In a few different examples, we also examine what an inverse problem would mean in this context, and how our proposed technique could be applied.

These results have not been published yet outside this dissertation.

[49] We mean here that the bandwidth is implicitely and uniquely defined as maximizing some function, but that there is no explicit closed-form formula which corresponds to the bandwidth allocation that performs this maximization. For any concave utility function, the maximization is then a convex optimization problem with linear constraints, and many techniques exist to approximately compute an optimal solution.

# Chapter 2

# Inverse Problems in Queueing Networks

## 2.1 Introduction

The general aim of this chapter is to discuss a class of inverse problems of queueing theory which find their origin in Internet probing. It is our belief that much of what is attempted in the Internet active probing area can be cast into the framework of inverse problems in queueing theory, or more generally, of inverse problems in discrete event dynamical systems theory. The present chapter contains new and recent results in this connection and proposes a classification of questions and problems within this setting. The choice of queueing theory as direct theory for inverse problems is a natural choice, due to its historical interaction with the design of telephone networks and its fundamental role in the design of Internet, to its proximity with the actual behaviour of communication networks, and to the richness of its many results. A small number of recent works provide rigorous results of this type. The great majority of the literature however is focussed on heuristic inversion methods.

This theoretical approach for a practical problem is motivated by the following:

1. the (hidden) assumptions about the behaviour of the network are systematically stated here, due to the specification of the considered direct problem;

2. the connections between network theory and network measurement is explored in a structured way; in particular, inverse problem terminology allows one to take into account the natural constraints of network active probing; in this aspect, inverse problems theory is a fundamental theoretical approach to the practical problem of measuring networks;

3. it is suited to provide general recommendations about network measurement, such as optimal probe sequences; the design of experiment is a useful resource for this aim.

Additionally, we will see in chapter 3 that although queueing theory does not model perfectly the Internet, it is possible to adapt the inverse problem technique to real networks.

In this dissertation, we focused on the settings of end-to-end active probing, as a particular case of great interest for Internet measurement. In particular, the primary metrics will be restricted to quantities that can be measured in an end-to-end setting, and no cooperation from the network will be assumed. This excludes techniques that rely on the clever use of ICMP messages for example. Some of the constraints we put here can be removed in a second step, and it is possible to consider inverse problems related to ISP-centric (or server-centric) measurement, or that include limited cooperation from the network. We however think that it is useful to start with the most restrictive (and hence general) case first.

This chapter is organized as follows: section 2.2 describes the main concepts of inverse problems in queueing theory and gives a first classification of these problems. The chapter is then structured into sections with increasing levels of realism. Section 2.3 focuses on the case where the observations provide noiseless estimates of certain stationary distributions or moments. This leads to a class of *analytical inverse problems*, where the main output of the method is a closed form formula or a terminating algorithm providing the exact value of the unknown parameters from the observations.

Although it does not belong to this chapter, we mention here that the case of *statistical inverse problems*, where observations are *finite time series* and where the need is therefore for robust inversion methods taking the noise into account, deserves great attention, and is the subject of chapters 3 and 4. The main outputs of the method are 1) a set of estimators that are shown to be asymptotically consistent and 2) recursive algorithms allowing one to implement the estimation of the unknown parameters from the time series.

Both analytical and statistical inverse problems are based on rather specific parametric models which may not be realistic for representing IP networks. The drawback of such parametric methods is that they have to be checked on testbeds and adapted using heuristic modifications in order to cope with real IP networks and traffic (as amply exemplified in *e.g.* the papers published in the proceedings of the IMC conference). We will not pursue this line of thought here. We will rather investigate methods which do not suffer from this weakness. This is the object of section 2.4 which is centered on inversion techniques that work for general classes of models. For these more general systems, we will limit ourselves to the non intrusive case, as defined in section 1.5.2. In this case, we show that there exist probing strategies leading to asymptotically consistent and minimal variance estimators of the unknown parameters, and this regardless of the specific instance of model taken from this class. These examples are taken from the literature, and have been published in [BMVB06, BMVB07]. The conclusions of section 2.4 are guidelines and recommendations on how to 'optimally' act in this more general setting. This is linked to the general framework of the *design of experiments* in statistics (see the thesis of B. Parker for the application of this methodology to packet networks). Section 2.5 concludes this chapter.

## 2.2 Inverse problems in queueing theory

Our discussion of inverse problems in queueing theory will be from the viewpoint of an Internet prober. That is, an entity whose network observations are derived from probes which are inserted into the network, where the latter is modelled as a queueing system. The default assumption is that only end-to-end measurements on probes are available, that is that the network does not cooperate in any way and so must be treated as a 'black box'. The reason for this are that Internet service providers are generally either unable, or unwilling, to provide information on their network or the traffic flowing on it. In addition, a route may traverse several Autonomous Systems (administrative domains), implying the need for cooperation across multiple, and competing, providers. Probing is one of the main ways in which knowledge of the growth and performance of the Intenet, for example its interconnection graph or topology, is known today. Indeed, service providers themselves use probing, despite the fact that they have the option of making measurements directly on their switching infrastructure. The flexible nature of probing, and its direct access to end-to-end metrics important for network applications, makes it an important tool for providers to learn about their own networks. For the end user, it is perhaps their only option. Due to its practical importance, and a considerable and growing literature, we focus on this end-to-end probing viewpoint, although of course there exist many other types of inverse problems pertaining to queueing theory. Within the IP network framework, there are for instance many interesting *ISP-centric* inverse problems too, which will be briefly discussed in section 2.2.10. There are also interesting problems in connection with other domains of applications of queueing theory. Let us quote for instance the queue inference engine of R. Larson [Lar90]. This inference engine was designed for ATM machines where the operator of the (cash) machine wants to evaluate the distribution of the customer queue size. The observables are here the epochs of the beginning and the end of all transactions (as recorded by the machine). The busy periods of the single server queue representing the ATM machine can hence be reconstructed from these observations; from this, the law of the queue size can then be evaluated. As we see, the nature of the problem is quite different from what was described above because the observables are quite different (the beginning and end of each service time in the latter case, the arrival times to and the departure times from the queue in the Internet probing case, assuming that one represents the IP path as a single server queue).

### 2.2.1 Direct equations of queueing theory

Queueing theory studies the dynamics of stochastic processes in a network of queueing stations, such as queue sizes, losses and delays, as a function of certain parameters. These parameters can be related to the structure of the stations (the number of servers, buffer sizes, service disciplines) or can be the distribution of the stochastic processes driving the queueing network (*e.g.* the rate of some exogenous Poisson arrival point process, or the law of the service times in a given station). The associated direct equations may bear either on the joint law of these stochastic processes (*e.g.* the queue sizes form a Markov chain in a

Jackson network), or on the recursions satisfied by the random variables themselves (*e.g.* Lindley's equation for the end-to-end delays for $./GI/1$ FIFO queues in series).

The solution of the direct equation bears on the law of these stochastic processes and might be the steady state or the transient *distribution*. The solution in the recursion viewpoint might be the steady state or the transient random state *random variable*.

In the network probing setting, there are two types of customers in the network: the customers (or packets) sent by regular users, often referred to as *cross–traffic*, and the customers (or probes) sent by the prober performing the measurement experiment. The former are typically fixed, namely the prober has no way to act on the cross–traffic offered to the network, whereas the latter can be sent at will, at least in the case of active probes.

Note that probes are themselves packets. In the active measurement case, their sizes may be chosen at will within a range of values. In the case of the Internet, all IP packets contain a header carrying essential information such as the IP address of the destination, so that 0 size probes are not possible. The maximal size of an IP packet is also fixed, which translates to an upper bound on probe size. In the passive measurement case, probes are just normal packets sent as part of a given application, for instance the packets of a Transport Control Protocol (TCP) flow in charge of a file transfer. The probe sizes are then determined by the selected application and associated network protocol.

A key question within this setting is whether the chosen parametric *queueing model* is an acceptable approximation of the concrete communication network with its cross–traffic and its probes. One most often needs a solution for the direct equation in order to solve the inverse problem. There is hence a crucial tradeoff between the realism of the queueing model and the mathematical tractability of its direct equation.

### 2.2.2 Noise

Deviations from ideal assumptions, which we denote generically by 'noise', are present at several levels within this setting:

- Most queueing problems are random by nature: for instance cross–traffic is best represented as a random process. A key question here is whether the underlying random processes are stationary or not. Since stationarity is most often desirable for tractability, this will lead to upper-bounds on the probing period which should not exceed the time scale at which macroscopic, for example diurnal, changes occur.

- There may also be actual measurement noise in the data. In the probing framework, most raw measurements consist of probe departure and arrival timestamps. Neither timestamping, nor the clocks that underlie them, are perfect, and high precision is important in order to resolve small differences in latencies (system times) arising from high capacity links (high service rates). The probability law of the measurement errors can however be well approximated in many cases.

- Finally, there may be noise stemming from the nature of the data itself: all practical

time series obtained from measurement experiments are finite, and so the resulting estimators for parameters are non-degenerate random variables. In other words, there are statistical errors in the parameter estimates.

In spite of all these random phenomena, it may still make sense to consider deterministic direct equations. For instance, the law of a stationary and ergodic stochastic process is a deterministic object, and the pointwise ergodic theorem shows that when the observables contain an infinite time series of samples of such a process, these allow one to reconstruct the stationary law in question with arbitrary precision. In what follows, we will distinguish between noiseless inverse problems, which correspond to a kind of mathematical idealization of reality (*e.g.* obtained with infinite stationary and ergodic time series, which allow one to determine the exact value of all mean quantities), and noise-aware or robust inverse problems where the intrinsic randomness of the problem is faced.

### 2.2.3 Probing actions

The observables are generated through certain actions of the network prober. We below describe what actions are allowed.

**Choice of topology** Whenever probes traverse more than a single station, the *route* they follow must be specified. We have seen in section1.1.3), and that the routes on the Internet are determined by the network functionalities, based on the addresses of the origin and destination, and that the sender has no control on (or even knowledge about) that route. Hence, a route is here an input-output/origin-destination pair. Within the IP network setting these end points correspond to interfaces in IP routers. In queueing theory, a natural incarnation is that of a route in the sense of Kelly-type networks as presented in section 1.2.3. The chief scenarios are as follows. The network probing is:

- *point-to-point* when probes are sent from a single source to a single destination;
- *point-to-multipoint* when probes are sent from a single source to multiple destinations (the network of queues traversed then has a tree-like topology);
- *multipoint-to-point* in the case of multiple sources to a single destination;
- *multipoint-to-multipoint* in the case of multiple sources to multiple destinations.

In the point-to-multipoint case the actual IP network experiment may differ depending on whether the network has native IP *multicast* available or not. In the former case, probes fork-out at each node of the network with a degree larger than 1, and this is well represented by what happens in a Fork-Join queueing network [BMT89]; in the latter case, the experiment will in fact consist of a collection of coordinated point-to-point schemes. In the other cases, the only possibility directly supported by the current Internet is that of a collection of point-to-point schemes. Note that it is generally assumed that all probes traveling from a given

source to a given destination pass by the same sequence of internal stations (routers), though this can be generalized.

**Passive probing actions** Purely passive probing is in fact monitoring, and the only freedom the experimenter enjoys is an ability to filter packets according to various criteria, for example to only take note of TCP packets, and to decide which of these to 'baptise' or *tag* as probes. A less restrictive case is when the prober can in addition control certain overall parameters of probing traffic. In the IP setting, he could for instance select an HTTP application which would initiate several TCP connections whose packets would act as probes, or alternatively a UDP based application like Voice over IP (VoIP) could be used to generate a probe stream. Here the prober can nsure that probes of the desired transport and application type are present, and also decide on when to start and end the flow(s), but there is still no control at the level of individual packet timing. This is for example the approach taken in Grenouille [gre], where FTP downloads and uploads are initiated, and their bandwidth measured.

**Active probing actions** Active probing consists in sending a set of probes at carefully selected epochs and with carefully chosen sizes. Complete control is possible subject only to constraints on probe size and/or rate as noted above. We include in this category the important case where the probe sizes and their emission times are defined through stochastic processes with fully controlled parameters. Because the network functionalities of section 1.1.3 require some exchange of data and identification of packets, the size of a packet has a lower bound. For example, a TCP/IP packet is at least 40 bytes long: the Internet Protocol (IP), the routing protocol over Internet, adds a (minimum) 20 bytes header to any packet, and TCP, the transfert protocol which provides reliable transfert and congestion control via acknowledgements, also adds its own header of at least 20 bytes. For this reason, it is impossible to send perfect 'stealth' probes. Active probing techniques impact the network, and this (possibly negligible) impact will be considered in section 2.2.6.

### 2.2.4 Observables

Observables are the raw data quantities available to the prober through conducting a probing experiment, and derive from the probing actions just described. In the end-to-end viewpoint, for each route, this data consists of probe packet sizes and departure timestamps at the origin, and loss indication and arriving timestamps (if applicable) at the destination. Effectively therefore, the information is of two types for each route: a loss indication for each probe marking whether it arrived at the destination or not, and if applicable, the probe latency or *delay* in traversing the route.

In the case of active observations, the packet sizes and departure times are in fact controlled by the prober and therefore already known. For simplicity we nonetheless refer to these as observables.

### 2.2.5 Unknown parameters and performance metrics

In the context of communication network probing, typical parameters to be identified would be:

- *structure parameters* of the nodes/queueing stations traversed by the probes such as the speed of the link/server, the buffer size, the service discipline used (*e.g.* to check neutrality, an important requirement of the IETF that packets should not be discriminated against on the basis of the application they stem from);
- *cross–traffic parameters* at a given node if the law of the cross–traffic is in a known parametric class, or otherwise its full distribution.

It is often desirable to estimate certain *performance metrics* such as the packet loss probability, or the distribution of packet latency, along a route or at a given node, in the context of incomplete knowledge of the system parameters.

### 2.2.6 Intrusiveness, bias and restitution

Since probes are processed as customers by the queueing system, and moreover have a minimum size which is positive, they interact with cross–traffic and so are inherently intrusive. At first glance, this seems to make the inverse problems more difficult. In fact, as we shall see, intrusiveness may be useful and can be leveraged in many cases (for example see the poly-phase methods introduced below).

As a result of intrusiveness, in general, the performance metrics of the system with cross–traffic and probes differ from those of the system with cross–traffic only. The performance metrics (or the parameters) of the system "without the probes" are often referred to as the *ground truth* in the network probing literature. For instance the probability that a typical packet of the cross–traffic on a given route will be lost if there were no probes, or the mean cross–traffic load at the $k$-th router on a route, belong to the ground truth. More generally, the parameters listed above (structural or pertaining to cross–traffic) are by definition part of the ground truth.

An important question is the reconstruction of some ground truth metric from the observation or the estimation of the metric for the perturbed system. This will be referred to as *restitution* below.

Restitution may even be needed even in the non-intrusive case (for example when probes have zero size and system time is the metric of interest) because of the *sampling bias* problem: a typical example is when the ground truth can be evaluated from certain time-averages and where probe-averages do not coincide with time-averages.

### 2.2.7 Identifiability, ambiguity

The observables, either implicitly or explicitly, carry information regarding a spatio-temporal slice of the network experienced by the probes. This information is clearly partial,

which gives rise to a set of system identifiability questions. For example, in the context of intrusive probing, it is not clear whether the restitution of many ground truth metrics is possible even in principle.

We shall see below that some parameters or performance metrics of a queueing system are not always identifiable from the observables. In some cases, different parameters can lead to the same observations.

### 2.2.8 Estimation problems

As mentioned above in 2.2.2, in practice the duration of a probing experiment compatible with stationarity is finite, and the number of probes that can be sent during a finite time interval is likewise finite. As a result, in practice the observables consist of time series of finite length, and inversion for the unknown parameters based on them is no longer a deterministic problem, but one of statistical estimation. This leads to a new class of problems in the design of such estimators, and the establishment of their properties, in particular the classical ones of bias, variance, asymptotic consistency and asymptotic normality.

In the case of active probing, the degrees of freedom in how probes are sent allows for another level of problems built on optimizing the statistical properties above. For example a natural question is to ask how probes should be spaced so as to minimize estimation variance.

### 2.2.9 The prober's path(s) to Ground Truth

Let us summarize by stressing that all paths to a given ground truth or performance metric require the following series of steps:

1. a tractable and yet realistic direct equation for the dynamics of the observables;
2. a proof of the identifiability of the perturbed metric from the observables;
3. the definition (and possibly the optimization) of estimators for these metrics;
4. the design of a restitution mechanism allowing one to reconstruct the ground truth from the perturbed or biased metrics.

The aim of the following sections is to illustrate the above in a few fundamental scenarios. Fortunately enough, some of the requirements may be relaxed in some cases, one may for instance

- idealize step 3, by assuming an infinite time series and therefore, for example, a full knowledge of the stationary distribution of some observable; this leads to deterministic problems that will be illustrated in section 2.3.
- avoid step 4, by selecting an active probing strategy involving probes rare and small enough to have almost no impact, which justifies a claim that the perturbed and unperturbed systems are the same in practice.

Of course, the validity of such simplifications will have to be discussed in detail.

### 2.2.10 ISP-centric inverse queueing problems

The scenarios considered in the remainder of the chapter focus on point-to-point inverse problems (which are often more challenging than their multipoint counterparts) arising in active Internet probing with end-to-end observables. For the sake of completeness, we now add a few words on other practical incarnations of inverse problems in queueing theory stemming from the ISP viewpoint.

The simplest observables for an ISP are time series of individual queue sizes and traffic (service times and packet sizes and arrival times) at the input or output ports of its own routers. The ISP has the privileged option of directly and non-intrusively monitoring these. Its actions then primarily consist in choosing when and what queues or traffic processes to monitor. The parameters and metrics of interest are quite different from, in some sense inverse to, those alluded to above. An elegant example is that of the reconstruction of end-to-end metrics, such as the packet loss point process or the fluctuations of end-to-end delays (jitter) experienced by a typical user whose packets pass by the monitored router, given the node based observables.

Other aspects of the problem, such as the direct equations to be used, their random nature, the resulting need for estimators of the metrics of interest, are all quite similar to what was described above in the Internet prober case.

## 2.3 Noiseless Inverse Queueing Problems

As mentioned above, in this section we assume that the availability of an infinite time series has provided perfect knowledge of the distribution function of the end-to-end stationary observables, so that step 3 from section 2.2.9 may be skipped. This is an idealization of the noise-aware case, which we study in chapters 3 and 4.

Within this context, we discuss three types of classical models of queueing theory on which Internet probing type inversion is possible: M/G/1, M/M/1 and M/M/1/B. The methods described in this section all leverage the fact that probes are intrusive. They consist in varying the probing rate and in observing how the system reacts to this variation. There are again various levels of realism: one can either assume, as in section 2.3.1, that the *mapping* that describes the variation of the observation as a function of the probing rate can be deduced from the observations, or pursue a more realistic scenario (considered in the other subsections) where one knows the value of this variation at some finite number of points (probing rates), as in the 'finite number of glimpses' scenario of the introduction.

There is a small literature on this analytic approach, scattered in the communication network literature, particularly the proceedings of venues with a strong Internet focus. Among these the first seems to be [SM98]. Another early paper advocating an analytical inversion for the estimation of loss processes in networks is [ANT01]. The approach in the latter is

moment based (see below).

### 2.3.1 The M/G/1 Queue

Before probes are injected, the system consists of a FIFO M/G/1 queue with a single server with speed 1. The service distribution $G$ is the unknown parameter of cross–traffic, but the input rate $\lambda$ is known. The sizes of probes obey a law $K$ (this is the service time for probes) and arrive according to a Poisson point process with rate $x$. The active prober only has access to the distribution of end-to-end delays of probes. Can he reconstruct the unknown parameter $G$?

The direct equation is the Pollaczek-Khinchin (PK) formula of theorem 1.2.15 which stipulates that the stationary waiting times of probes have for Laplace Transform (LT)

$$\mathcal{L}_W(s) = \frac{(1 - x\overline{K} - \lambda\overline{G}))s}{s - x(1 - \mathcal{L}_K(s)) - \lambda(1 - \mathcal{L}_G(s))} \quad ,$$

where $\mathcal{L}_K(.)$ and $\mathcal{L}_G(.)$ denote the LT of $K$ and $G$ respectively, and $\overline{K}$ and $\overline{G}$ their means. We assume that

$$x\overline{K} + \lambda\overline{G} < 1$$

which is necessary and sufficient for the existence of a stationary regime. Since $\overline{G}$ is unknown, it is impossible to check this condition without prior knowledge. Most Internet resources have a moderate utilization factor (*i.e.* $\lambda\overline{G}$ rarely exceeds 3/4 or even 1/2) and if $x\overline{K} \ll 1$, then the last condition is quite likely to hold. Note that as a general principle probing overhead should be kept small, in order to avoid consuming network bandwidth, to reduce intrusiveness, and to prevent probes being confused with network attacks, so assuming $x\overline{K} \ll 1$ is quite reasonable.

From our infinite time series assumption, we have access to any function of the stationary end-to-end delay process of probes. In particular, the function $\mathcal{L}_W(s)$ is *indirectly observable* (*i.e.* can be obtained from the direct delay observable) for all values of $x$ and $K$ since the waiting time of a probe is obtained by subtracting its service time – which is known to the prober – from its end-to-end delay.

We now proceed to *invert* the direct equation. By letting $s$ go to infinity, we have

$$\mathcal{L}_W(\infty) = P_x(W = 0) = 1 - x\overline{K} - \lambda\overline{G} = \kappa(x)$$

which is also indirectly observable. Hence for all $x$, $1 - \kappa(x) = x\overline{K} + \lambda\overline{G}$, and one can determine $\overline{G}$, which, substituting into the PK transform,

$$\mathcal{L}_G(s) = \frac{(1 - x\overline{K} - \lambda\overline{G})s}{\mathcal{L}_W(s)\lambda} - \frac{s - x - \lambda + x\mathcal{L}_K(s))}{\lambda} \tag{2.1}$$

determines the transform of the entire law $G$. Therefore, our unknown parameter can be unambiguously estimated from such observables.

This approach also allows us to estimate the ground truth stationary end-to-end delay distribution. The restitution formula consists in again applying the PK formula for waiting time, but this time without the probe traffic, which is possible since $\lambda$ and $G$ are now known.

The main weakness of the present approach should be clear: it requires the estimation of *distribution functions* (here LTs) rather than *moments*; it may be desirable to have *moment-based* methods (see section 2.3.2 and 2.3.3 below);

### 2.3.2 The M/M/1 Queue

The setting of this section is slightly different from that of the last section. The system is a M/M/1 FIFO queue with a server of unknown speed $\mu$. Cross–traffic is Poisson with unknown intensity $\lambda$ and exponential packets with mean 1. The active prober sends Poisson probes with rate $x$ to the system. All probes have exponential size of mean 1. Can one reconstruct $\lambda$ and $\mu$ when observing only the mean stationary end-to-end delays experienced by the probes?

The stationary mean number of packets and probes in the station is

$$\overline{N}(x) = \frac{\lambda + x}{\mu - \lambda - x},$$

under the condition $\lambda + x < \mu$. From Little's formula the mean end-to-end delay $D$ of probes (or packets) is

$$\overline{D}(x) = \frac{\overline{N}}{\lambda + x} = \frac{1}{\mu - \lambda - x} \quad . \tag{2.2}$$

This formula, which is our direct equation, shows that the constant $\mu - \lambda$, which carries the interpretation of *residual bandwidth*, can be reconstructed from the observation of $\overline{D}$ associated with the value of $x$. However, the individual constants $\mu$ and $\lambda$ *cannot* be reconstructed individually from this alone. Fortunately enough, this mean residual bandwidth is sufficient for the restitution of the ground truth cross–traffic delay $\overline{D}(0) = \frac{1}{\mu - \lambda}$.

Let us summarize our conclusions on this case: we have here a first-moment based probing strategy allowing one to determine unambiguously the mean residual bandwidth of an M/M/1 queue solely from the measurement of the empirical mean end-to-end delays experienced by probes. Within this context, the problem of identifying the intensity of cross–traffic or the speed of the server is however ill-posed.

When adding second order estimates, one obtain the additional information needed to resolve the two parameters. For instance, when sending packet pairs with size $y$ at the same time, one gets that their system times, $D$ and $D'$ are such that $D' - D = y/\mu$ so that $\mu$ can be determined (this packet pair method actually holds for all G/G/1 FIFO queues). In reality, two packets cannot arrive exactly at the same time. It is shown in Appendix 2.6.1 that in the M/M/1 queue, two packets with size $y$ sent $t$ seconds apart have system times $D$

and $D'$ which are such that, as $t$ goes to 0,

$$\mathbf{E}(DD') = K(y) - t\left((1-\rho)y + \frac{\rho}{\mu}\right) + o(t)$$

where $K(y)$ is some constant. The slope w.r.t. $t$ of the function $t \to \mathbf{E}(DD')$ is $(1-\rho)y + \frac{\rho}{\mu}$ and it can be estimated, so that $1-\rho$ and $\frac{\rho}{\mu}$ can also be estimated to arbitrary precision, using different values of $y$. This determines both $\lambda$ and $\mu$ unambiguously.

There are other practical methods to evaluate $\mu$ not based on moments. The simplest one consists in sending probes with constant size $y$ and in looking for the probes with minimal delay. This minimal delay of course allows one to determine $\mu$ unambiguously.

### 2.3.3 The M/M/1/B Queue

The setting is the following: the prober sends Poisson probes with rate $x$ into a system which, without the probes, would be an M/M/1/B queue with Poisson (cross–traffic) input point process of unknown intensity $\lambda$. Cross–traffic packets are assumed to have exponential sizes of parameter 1, and the prober emulates this by choosing to send probes with the same size distribution.

Under natural independence assumptions, the full system (with cross–traffic and probes) is an M/M/1/B queue with arrival rate $\lambda + x$ and service rate $\mu$. The direct equation is the following classical expression for the stationary loss probability $p(x)$ (see for example [Tak62]):

$$p(x) = \frac{\left(\frac{\lambda+x}{\mu}\right)^B - \left(\frac{\lambda+x}{\mu}\right)^{B+1}}{1 - \left(\frac{\lambda+x}{\mu}\right)^{B+1}} \quad . \tag{2.3}$$

Similarly, the probability $q(x)$ that the queue is empty is

$$q(x) = \frac{1 - \frac{\lambda+x}{\mu}}{1 - \left(\frac{\lambda+x}{\mu}\right)^{B+1}} \quad . \tag{2.4}$$

Can one determine $\lambda$, $\mu$ and $B$, assuming that these parameters (or some of them) are unknown?

From our infinite time series assumption, we have access to the loss rate $p(x)$ as well as to the sequence of end-to-end delays for each probe. Using packet pair techniques [PV02b], or alternatively by observing delay minima when probes are chosen of constant size, it is possible to extract the server speed $\mu$. We therefore assume that $\mu$ is known. One key consequence of knowing $\mu$ is that the prober then knows the service time of each probe, and he can therefore measure the empirical probability $q(x)$ that the queue is empty, since for probes which encounter an empty queue the observed end-to-end delay is equal to the service time.

Assume a poly-phase probing scheme with $N$ different probe intensities $x_i$, $i =$

$1, \ldots, N$. Within our noiseless setting, the prober's measurements allow him to determine the associated loss rate $p_i$ and empty queue probability $q_i$, and hence to compute the ratio $r_i = \frac{p_i}{q_i}$. From (2.3) and (2.4), the following should hold for all measured ratios:

$$\forall 1 \leq i \leq N, \qquad r_i = r(x_i) = \frac{p(x_i)}{q(x_i)} = \left(\frac{\lambda + x_i}{\mu}\right)^B,$$

where $r(x)$ is the polynomial $(\lambda + x)^B/\mu^B$. For all $N \geq 1$ let $L_N(x)$ denote the Lagrange polynomial interpolating the points $(x_i, r_i)$, $i = 1, \ldots, N$, namely the polynomial in $x$ of degree at most $N - 1$ defined by the formula

$$L_N(x) = \sum_{i=1}^{N} r_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \quad . \tag{2.5}$$

For $N \geq B + 1$, we have $L_N(x) = r(x)$ for all $x$. Hence $\lambda$ and $B$ can be determined as follows:

- $B$ is the degree of $L_N(x)$;
- $(\frac{\lambda}{\mu})^B$ is the constant term of $L_N(x)$ (or $-\lambda$ is the unique real root of $L_N(x)$).

The main limitation of this characterization is that we don't know when $N \geq B + 1$, *i.e.* how many phases are needed. In other words, we have an algorithm which converges to the correct values when letting $N$ go to infinity, but we have no termination criterion for this algorithm. The following lemma and theorem provide such a termination criterion.

**Lemma 2.3.1.** *Consider the set of polynomials $r_{\lambda,\mu,B}(x)$ with $B$ ranging over the positive integers and $\lambda$ and $\mu$ over the positive real line. Two different polynomials of this family intersect in at most 2 points of the positive real line.*

*Proof.* Consider the polynomials $P_1(x) = r_{\lambda_1,\mu_1,B_1}$ and $P_2(x) = r_{\lambda_2,\mu_2,B_2}$. One can assume without loss of generality that $\lambda_1 > \lambda_2$. Let $\delta = \lambda_1 - \lambda_2$. Setting $y = x + \lambda_2$, the equality $P_1(x) = P_2(x)$ now reads

$$(y + \delta)^{B_1} = \frac{\mu_1^{B_1}}{\mu_2^{B_2}} y^{B_2}.$$

If $B_1 \leq B_2$, let $k = B_2 - B_1$. The equality is equivalent to

$$\left(1 + \frac{\delta}{y}\right)^{B_1} y^{-k} = \frac{\mu_1^{B_1}}{\mu_2^{B_2}}.$$

The left hand term is a decreasing function of $y$ for positive $y$, and the right hand term is constant. There is therefore at most 1 solution for positive $y$ and hence for positive $x$.

If $B_1 > B_2$, let $k = B_1 - B_2$. The equality is equivalent to

$$\left(1 + \frac{\delta}{y}\right)^{B_1} y^k = \frac{\mu_1^{B_1}}{\mu_2^{B_2}}.$$

Assume there exists at least 3 positive solutions $0 < y_1 < y_2 < y_3$. Then applying Rolle's theorem to the function $f(y) = \left(1 + \frac{\delta}{y}\right)^{B_1} y^k$, we get that there are two points $y_4 \in ]y_1; y_2[$ and $y_5 \in ]y_2; y_3[$ such that $\frac{\partial f(y_4)}{\partial y} = 0$ and $\frac{\partial f(y_5)}{\partial y} = 0$. Now, note that the derivative

$$\frac{\partial f}{\partial y} = y^{k-1}(1 + \frac{\delta}{y})^{B_1 - 1}\left(k(1 + \frac{\delta}{y}) - \frac{B_1\delta}{y}\right)$$

admits only one zero $y = \frac{\delta B_2}{k}$, which contradicts the existence of 3 solutions $y_1 < y_2 < y_3$. □

**Theorem 2.3.2.** *Assume we have a set of observation points $(x_i, r_i)$, $i = 1, \ldots, N$, stemming from an M/M/1/B queue with parameters $\lambda$ and $\mu$. If $N > 2$ and if the Lagrange polynomial $L_N(x)$ interpolating the points $(x_i, r_i)$, $i = 1, \ldots, N$, can be written as $\left(\frac{\widehat{\lambda}+x}{\widehat{\mu}}\right)^{\widehat{B}}$ for some positive integer $\widehat{B}$ and some positive numbers $\widehat{\lambda}$ and $\widehat{\mu}$, then $B = \widehat{B}$ and $\widehat{\lambda} = \lambda$.*

*Proof.* This is a consequence of Lemma 2.3.1. The polynomials $L_N(x)$ and $r_{\lambda,\mu,B}(x)$ intersect in $N > 2$ points, and therefore are equal. □

We have hence a termination rule: increase the cardinal $N$ of the set of points $(x_i, r_i)$, $i = 1, \ldots, N$ until the Lagrange polynomial $L_N(x)$ interpolating these points is of the form $\left(\frac{\widehat{\lambda}+x}{\widehat{\mu}}\right)^{\widehat{B}}$.

We can hence reconstruct the ground truth (on the intensity of cross–traffic and on the loss probability for cross–traffic packets in the absence of probes) by using the formulas for the M/M/1/B queue again, since all the missing parameters are now determined.

### 2.3.4 The Erlang loss system

The same method can be easily applied to an Erlang loss system, *i.e.* an M/G/B/B queue, where the service time distribution $G$ and the arrival intensity $\lambda$ are unknown. The probe packets arrive according to a Poisson point process with rate $x$, and their sizes obey a law $K$.

From example 1.2.2, we know that when the service time distribution is exponential, the steady-state blocking probability is

$$P_B(x) = \frac{\frac{\rho(x)^B}{B!}}{\sum_{i=0}^{B} \frac{\rho(x)^i}{i!}} \quad , \tag{2.6}$$

where $\rho(x) = \lambda\overline{G} + x\lambda\overline{K}$ is the load of the system. Similarly, the empty queue probability

is

$$P_0(x) = \frac{1}{\sum_{i=0}^{B} \frac{\rho(x)^i}{i!}} \quad . \tag{2.7}$$

A remarkable property of Erlang loss systems (see [Tak62] for example) is that they are insensitive to the precise distribution of service times: the empty queue and blocking probabilities depend only on the total load $\rho(x)$ and the number of servers $B$.

For any probing intensity $x_i$, the prober can easily measure the loss rate $P_B(x_i)$. Similarly, using fixed size packets, the prober can measure the empty queue probability $P_0(x_i)$ (this corresponds to the proportion of packets which experiences a minimal delay). Hence, the ratio $r_i = r(x_i) = \frac{P_B(x_i)}{P_0(x_i)} = \frac{(\lambda\overline{G}+x_i\overline{K})^B}{B!}$ is indirectly measurable, and the method developped in section 2.3.3 can be used to estimate the capacity $B$ and load $\lambda\overline{G}$ of the system in absence of probes. Due to the insensivity of Erlang loss system to the service time distribution, it is clear that these are the only parameters that one can be inferred.

A main weakness of this inversion scheme is that it in fact requires several infinite time series, one per value of $x$; for instance, the successive interpolations of (2.5) would in practice require $N$ successive *phases*: for each $1 \leq i \leq N$, a phase where the prober sends probes at rate $x_i$ and collects enough samples to have a precise enough estimate of the stationary probability ratio $r(x_i) = \frac{p(x_i)}{q(x_i)}$, which is a new system requiring a new time series for each $i$. It would be desirable to have *mono-phase* inversion techniques.

More elaborate questions can be addressed along similar lines, for example concerning the determination of the parameters when $\mu$ is unknown, but we will not pursue this line of thought here as our aim is more to illustrate of the set of problems and solution methods than to provide an exhaustive set of solutions.

## 2.4 Optimal Probing Strategies

We have already pointed out that in the error-prone case, once statistical estimators of parameters have been derived based on a given probing stream, one could consider going further by asking how their performance can be optimized by taking advantage of the free parameters of active probing. The difficulty here is that exploring richer probing streams, for example moving away from Poisson probing, implies dealing with more complex direct equations.

In this section we show how taking a more general point of view can lead to insight into the nature of probing streams which are likely to lead to good properties for the associated estimators, such as low estimation variance. To simplify the problem, we focus on the case of non-intrusive probes which have no impact on the system, namely the network and its cross–traffic.

Section 2.4.1, which builds upon ideas discussed in [BMVB06], bears on a question which is often referred to as the *sampling bias* problem and which in fact addresses the issue of the asymptotic consistency of empirical mean estimators.

Section 2.4.2 bears on the minimization of variance within this context. The main ideas

stem from [BMVB07].

Section 2.4.3 discusses a few open problems in the case of maximum likelihood estimators.

### 2.4.1 Sampling bias

Consider the following non-intrusive variant of the problem considered in section 2.3.2. The network consists of a single station with cross–traffic consisting in a Poisson point process (with intensity $\lambda$) of exponentially sized packets (with mean service time $\mu$). One wants to estimate the residual bandwidth $\mu - \lambda$.

For this, one sends probes of zero size to this system according to some stationary point process which is not necessary Poisson. Let $N = \{T_n\}_{n\in\mathbb{N}}$ denote the points of this point process and let $\{W(t)\}_{t\in\mathbb{R}}$ denote the stationary workload process in the station (since probes have 0 size, this workload is also the ground truth workload). We will assume this stochastic process to be right continuous. For all $n$, let $D_n = W(T_n)$. Since the system is FIFO and all probes have 0 size, $D_n$ is the end-to-end delay measured from probe $n$. If $N$ and $\{W(t)\}$ are jointly stationary, then the sequence $\{D_n\}$ is stationary too. If in addition $N$ and $\{W(t)\}$ are jointly ergodic, then the pointwise ergodic theorem implies that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} D_n = \mathbb{E}_N^0[W(0)]\, a.s. \quad . \tag{2.8}$$

In the last equation, $E_N^0$ denotes expectation w.r.t. the Palm probability $P_N^0$ of the point process $N$ (see [BB03]). But if $N$ and $\{W(t)\}_t$ are independent, then $E_N^0[W(0)] = E[W(0)]$, namely probe averages see time averages. Hence, under our assumptions,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} D_n = \frac{1}{\mu - \lambda}\, a.s. \quad , \tag{2.9}$$

so that we then always have an asymptotically consistent estimator for the residual bandwidth.

Assume now that the network and its cross–traffic form a G/G/1 queue with a server with speed 1 and packets with size distributed according to some probability law $F$ on the positive real line. Let $\{W(t)\}$ denote the workload process in this queue. Assume one sends non intrusive probes according to the point process $N$. If we have joint stationarity and ergodicity of the two last processes, then

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} 1(D_n = 0) = P_N^0[W(0) = 0]\, a.s. \quad . \tag{2.10}$$

If $N$ and $\{W(t)\}$ are independent, then $P_N^0[W(0) = 0] = P[W(0) = 0]$. But for all G/G/1 queues, $P[W(0) = 0] = 1 - \rho$, where $\rho$ is the load factor of the queue. Hence, under the foregoing assumptions, we have an asymptotically consistent estimator for the load factor,

which holds for all G/G/1 systems.

Until relatively recently, whenever the ground truth was some time average (or some function of a time average as above where the available bandwidth is the inverse of the mean stationary workload), it was recommended to use Poisson probes, namely probes sent at the epochs of a Poisson point process[50]. The rationale for that was that since *Poisson Arrivals See Time Averages* [BB03], the samples of the metrics estimated by Poisson probes allow one to estimate this ground truth.

The arguments used above show that that there is in fact no fundamental reason for using Poisson probes in the non intrusive case and that a wide variety of other probing strategies share the same 'lack of sampling bias' or more precisely asymptotic consistency property.

Let us list and discuss the key assumptions of the last derivation so as to reach a general statement. We consider some system with a continuous time state $\{W(t)\}_{t\in\mathbb{R}}$ assumed to be stationary and ergodic and where the unknown parameters can be determined from the knowledge of $E[W(0)]$. If the prober chooses some probing point process $N = \{T_n\}_{n\in\mathbb{N}}$ which is

1. non intrusive;
2. stationary;
3. independent of $\{W(t)\}$;
4. jointly ergodic with $\{W(t)\}$,

and if he can observe the quantities $D_n = W(T_n)$, then the empirical mean of the observations is an asymptotically consistent estimator of $E[W(0)]$ and hence of the unknown parameters.

All the above assumptions are necessary. For instance, in the G/G/1 queue example, 3 does not hold when $N$ is the point process of all or some selected arrivals of the cross–traffic. In this case (which could be seen as an incarnation of passive measurement), the empirical mean converges but to $E^0_N[W(0)]$ which is then different from $E[W(0)]$ in general. As for 4, if for instance $N$ and $\{W(t)\}$ are both periodic, then there is no joint ergodicity (we have a phase lock) and empirical averages converge to a random variables that depends on some random phase. In none of theses cases do we have an asymptotically consistent estimator of $E[W(0)]$.

It is easy for the prober to build a stationary point process independent of $\{W(t)\}$, for instance by making use of a stationary renewal process. A simple way to guarantee 4 is to require that this point process be mixing. Indeed, the product of a mixing and an ergodic shift is ergodic [Pet83].

Hence the general *NIMASTA recommendation*: Non Intrusive and Mixing probing Arrivals See Time Averages. Poisson processes are mixing and there is no harm using such

[50] See the paragraph "Inverse problems in network related work" of section 1.5.4 for a quick literature review.

processes within this setting. But the class of 'good' probing point processes is much larger as we see.

The property that the sampling of an ergodic stochastic process at the epochs of a mixing and independent point process leads to no sampling bias was first proved in [GS98].

We conclude this section with a few observations:

- Consider the above framework. If $\{W(t)\}$ is known to be mixing, then all stationary ergodic point processes which are independent of $\{W(t)\}$ lead to an empirical mean estimator of the mean value $E[W(0)]$ which is asymptotic consistent.

- In the intrusive case and when the inversion method is based on the empirical mean estimator of the mean value $E[D_x(0)]$ of some characteristic of the system with its cross–traffic and its probes, Poisson probing is a natural choice as it guarantees asymptotic consistency, as a consequence of the PASTA property.

- NIMASTA is valid only when the system state $\{W(t)\}$ is independent of the probes. If the network experiences self-synchronization with the probes, or if a network operator unethically perturbs the system just before probe arrivals (*e.g.* to increase the apparente performence), the indepence holds no more, and PASTA is a sure fallback. This raises interesting questions, with another framework where one needs to include either the self-synchronization effect of the probes or the malicious behaviour of the network operator. We also argue here that if the probes are rare and small enough, the self-synchronization effects will most likely be negligible, and that network operators are more likely to change the apparente performance through different routing policies or priority queues (which will be valid whatever the timing of the probes) than throught attempting to empty the buffers just before a probe arrives (which is a technically difficult operation).

### 2.4.2 Variance

The setting is the same as that of the last subsection, with $N$ a stationary point process with intensity $\mu$. We denote the mean value to be estimated by $p = E[W(0)]$ and we denote the auto-covariance function of $\{W(t)\}_{t\in\mathbb{R}}$ by

$$R(\tau) = \mathbb{E}[W(t)W(t+\tau)] - p^2.$$

We assume that the function $\tau \to R(\tau)$ exists and is *convex* for $\tau \geq 0$.

The sample mean estimator of $p$ using $K$ samples is

$$\hat{p}_1 = \frac{1}{K}\sum_i W(T_i) \quad . \tag{2.11}$$

The underlying probability is the Palm probability of $N$. So $T_0 = 0$ by convention and $T_i$ is the sum of $i$ inter-sample times, which due to stationarity, each have law $F$ with mean $\mu^{-1}$.

Hence $T_i$ has mean $i\mu^{-1}$, and we denote its law by $f_i$.

Using the independence assumptions, we get that the variance of $\hat{p}_1$ (which coincides with its mean square error as the estimator is unbiased) is given by

$$\begin{aligned}\mathrm{Var}[\hat{p}_1] &= \frac{1}{K^2}\left(K\mathbb{E}[W(0)^2] + 2\sum_{i=1}^{K}\sum_{j=i+1}^{K}\mathbb{E}[W(T_i)W(T_j)]\right) - p^2 \\ &= \frac{1}{K^2}\left(K\mathbb{E}[W(0)^2] + 2\sum_{i=1}^{K}\sum_{j=i+1}^{K}\int R(\tau)f_{|i-j|}(d\tau)\right) - \frac{p^2}{K} \quad .\end{aligned} \tag{2.12}$$

As a special case of Equation (2.11), we pick out the estimator based on periodic samples of period $\mu^{-1}$, namely

$$\hat{p}_2 = \frac{1}{K}\sum_i W(i\mu^{-1}) \quad , \tag{2.13}$$

for which the integral $\int R(\tau)f_{|i-j|}(d\tau)$ in Equation (2.12) degenerates to $R(|i-j|\mu^{-1})$.

**Theorem 2.4.1.** *Under the above convexity assumption,* $\mathrm{Var}[\hat{p}_1] \geq \mathrm{Var}[\hat{p}_2]$.

*Proof.* Equation (2.12) holds for all processes. So, to compare the variances it is enough to compare, for all $i \neq j$, the cross terms, namely $\int R(\tau)f_{|i-j|}(d\tau)$ and $R(|i-j|\mu^{-1})$. But, if $R(\tau)$ is convex, Jensen's inequality says that

$$\int R(\tau)f_k(d\tau) \geq R\left(\int \tau f_k(d\tau)\right) = R(k\mu^{-1}) \quad , \tag{2.14}$$

for all $k$. □

We see that under the foregoing assumptions, *no* other sampling process has a variance which is lower than that of periodic sampling. As just one example, by taking $F$ to be exponential in $\hat{p}_1$ and inter-sample times to be independent, we learn that Poisson sampling yields a higher variance than periodic. However, the result is much more powerful than this. It shows that, if $R(\tau)$ is convex, no kind of train or other structure, no matter how sophisticated, can do better than periodic.

Unfortunately periodic sampling has a disadvantage already discussed: it is not mixing, which makes it vulnerable to phase locking effects. Assuming that $R(\tau)$ is convex, we now determine sampling schemes that offer the best of both worlds: mixing to guarantee asymptotic consistency, but with variance close to that offered by periodic sampling.

For this, we will consider sampling using renewal processes with inter-probe times that are Gamma distributed, namely with density

$$\Gamma_{\alpha,\lambda}(x) = \frac{\lambda}{\Gamma(\alpha)}(\lambda x)^{\alpha-1}e^{-\lambda x} \quad , \tag{2.15}$$

on $x > 0$, where $\Gamma(\cdot)$ is the familiar Gamma function. Its mean is $\mu^{-1} = \alpha/\lambda$ and its

variance $\sigma^2 = \alpha/\lambda^2$. Gamma laws are well known to be stable with respect to the shape parameter $\alpha$, that is, if $\{T_i \sim \Gamma_{\alpha_i,\lambda}\}$ are independent, then $\sum_i T_i \sim \Gamma_{\sum_i \alpha_i,\lambda}$. The exponential law corresponds to the 1-parameter sub-family $\Gamma_{1,\lambda}$. Another special sub-family are distributions with the Erlang law. These have only integral shape values.

We will need one more technical result regarding Gamma laws, the proof of which we leave to the appendix in section 2.6.

**Lemma 2.4.2.** *Let $T \sim \Gamma_{\alpha,\lambda}$, $Z \sim \Gamma_{\beta,\lambda}$ be independent, and set $Y = T + Z$. Then $C = \mathbb{E}[T|Y] = \alpha Y/(\alpha+\beta)$ has density $\Gamma_{\alpha+\beta,(\alpha+\beta)\lambda/\alpha}$, with mean $\mathbb{E}[C] = a/\lambda = \mathbb{E}[T]$.*

We can now prove

**Theorem 2.4.3.** *The family of renewal sampling processes $G(\beta)$, parameterized by $\beta > 0$, with inter-sample time density $\Gamma_{\beta,\beta\lambda}(x)$, provides, at constant mean sampling rate $\lambda$, sampling variance for $\hat{p}_1$ that monotonically decreases with $\beta$. The variance is larger (equal or smaller) than Poisson sampling as $\beta$ is smaller (equal or larger respectively) than 1, and tends to that of periodic sampling in the limit $\beta \to \infty$.*

*Proof.* We assume an underlying probability space on which the family of inter-sample variables are defined for each $\beta > 0$. Equation (2.12) holds for each inter-sample law $G(\beta)$. As the means for each are equal to $\mu = \beta/(\beta\lambda) = 1/\lambda$, proving the variance result reduces to showing that, for each $k > 0$, $\int R(\tau) f_{k,1}(d\tau) \geq \int R(\tau) f_{k,2}(d\tau)$ for any $\beta$ values $\beta_1$, $\beta_2$ satisfying $\beta_2 > \beta_1$, where $f_{k,i}$ is the density of the sum $T_{k,i}$ of $k$ inter-sample times, each with law $G(\beta_i)$. We can apply Jensen's inequality to show that

$$\mathbb{E}[\mathbb{E}[R(T_{k,1})|Y_{k,1}]] \geq \mathbb{E}[R(\mathbb{E}[T_{k,1}|Y_{k,1})] = \mathbb{E}[R(T_{k,2})] = \int R(\tau) f_{k,2}(d\tau)$$

where to show $\mathbb{E}[T_{k,1}|Y_{k,1}] = T_{k,2}$, we identified $(T, Y, \alpha, \beta, \lambda)$ with

$$(T_{k,1}, Y_{k,1}, k\beta_1, k(\beta_2 - \beta_1), \beta_1\lambda)$$

and used Lemma 2.4.2. Since this holds for any $\beta_1$, $\beta_2$ with $\beta_2 > \beta_1$, we have monotonicity of the variance in $\beta$. As $\beta$ tends to infinity, there is weak convergence of $\Gamma_{\beta,\beta\lambda}(x)(dx)$ to a Dirac measure at $1/\lambda$, as is easily seen using Laplace transforms. Since the function $R$ is convex, it is continuous, and as it is also bounded (as a second order process), the property

$$\lim_{\beta\to\infty} \int R(x)\Gamma_{\beta,\beta\lambda}(x)(dx) = \int R(x)\delta_{1/\lambda}(dx)$$

follows from the very definition of weak convergence. This shows that the limit of the variances of the Gamma renewal estimators is that of the deterministic probe case, namely the optimal variance. □

This result provides a family of sampling processes with the desired properties. By selecting $\beta > 1$, we can ensure lower (more precisely, no higher) variance than Poisson

sampling. By selecting $\beta$ large, we obtain sampling variance close to the lowest possible, whilst still using a mixing process. The important point is that the parameter $\beta$ can be used to continuously tune for any desired trade-off, and to set the sampling variance arbitrarily close to the optimal case. Note that this optimality is only valid for the expectation of a primary (directly observable) metric $\mathbb{E}[W(0)]$: if the metric of interest is deduced from $\mathbb{E}[W(0)]$, the inversion step can increase the variance of the secondary metric. The amplitude of this increase depends on the inversion step and on the distribution of the error.

There is therefore a need to better understand what classes of queueing systems/networks lead to second order state processes enjoying the above convexity property beyond the few classes quoted below.

**Known convex examples**

A natural question is, how likely is it that networks of interest satisfy the convexity property for delay and/or loss? There are simple systems for which exact results are known. For example, Ott [Ott77] showed that convexity holds for the virtual work process (equal to the delay of probes with $x = 0$) of the M/G/1 queue. Mandjes and Es-Saghouani [ESM09] extended this result to the case of queues fed by a spectrally positive Lévy process, and this was extended to the case of spectrally negative Lévy processes in [GM09].

We now show that the loss process $I(t)$ of the $M/M/1/B$ queue, (namely the indicator function that the number of customers is $B$, *i.e.* the set of periods where arriving packets are lost) has a convex auto-covariance function. Denote by $\lambda$ and $\mu$ the arrival and the service rates and by $\rho = \lambda/\mu$ the load factor. From [Tak62] (p.13, Theorem 1), the probability that the number of customers in the queue is $B$ at time $t$, given that it is $B$ at time 0, is

$$P_{B,B}(t) = \frac{1-\rho}{1-\rho^{B+1}}\rho^B + \frac{2}{B+1}\sum_{j=1}^{B} \frac{e^{-(\lambda+\mu)t+2t\sqrt{\lambda\mu}\cos\left(\frac{\pi j}{B+1}\right)}}{1-2\sqrt{\rho}\cos\left(\frac{\pi j}{B+1}\right)+\rho} \cdot \left(\sin\left(\frac{Bj\pi}{B+1}\right) - \sqrt{\rho}\sin(j\pi)\right)^2 \quad (2.16)$$

in the case when $\rho \neq 1$ and

$$P_{B,B}(t) = \frac{1}{1+B} + \frac{1}{B+1}\sum_{j=1}^{B} \frac{e^{-2\lambda t+2\lambda t\cos\left(\frac{\pi j}{B+1}\right)}}{1-\cos\left(\frac{\pi j}{B+1}\right)} \cdot \left(\sin\left(\frac{Bj\pi}{B+1}\right) - \sin(j\pi)\right)^2 \quad (2.17)$$

in the case $\rho = 1$. In both cases, the auto-covariance function of $I_x(t)$, which is equal to $\pi(B)P_{B,B}(t)$ (with $\pi(B)$ the stationary probability that the queue has $B$ customers) is a convex combination of convex decreasing functions of $t$ and is hence itself convex and decreasing in $t$.

### 2.4.3 Maximum Likelihood

Consider some network with a non-intrusive probing process $N$ where the unknown parameters are obtained by some maximal likelihood method. An example of such a system would be that of sections 2.4.1 and 2.4.2 when the output is an estimator $\widehat{\gamma}\,(W_1, \ldots, W_m)$ depending on the sequence $\{W_n\} = \{W(T_n)\}$ , where $W(t)$ is the virtual end-to-end delay in the network at time $t$. Hence $W(T_n)$ is the end-to-end delay seen by the $n$-th (stealthly) probe. Here, $\{W(t)\}$ is a continuous time Markov chain and if $N$ is an independent renewal process, then the sequence $\{W_n\}$ is Markov. If one knows the transition kernel $P_t$ of the continuous time Markov chain $\{W(t)\}$, then one can compute the likelihood function associated with the samples $W_n$, $1 \leq n \leq m$ through a formula that involves $P_t$ and the stationary law of $\{W(t)\}$. Here are a few open problems within this setting:

- What renewal point processes are asymptotically efficient within this setting? We conjecture that if $\{W(t)\}$ is mixing, then all renewal point processes are asymptotically efficient.

- For $m$ fixed, what renewal point process gives the MLE with the smallest variance among the set of all renewal point processes with intensity $\mu$? Is the deterministic point process again optimal in terms of variance? These questions are deeply correlated with the interaction between the sampling of the primary metric and the inversion from the primary metric to the secondary metric.

## 2.5 Summary

In this chapter, we have seen how Internet end-to-end probing techniques can be seen as inverse problems on queueing theory. The specific constraints of active probing can be easily integrated in the inverse problem framework. The main steps of such problems have been identified, and difficulties that can arise and some of their potential workarounds are examined. We have classified the inverse problems in different classes, depending on their properties. Simple examples have been used to illustrate these different properties in the case of analytical inverse problems. Two known results are cited as examples of optimal probing strategies, and a few open questions on this topic are finally formulated in a general setting.

The (important) question to determine which system is being measured is not tackled here. The answer could come from an *a priory* knowledge of the nature of the network. Alternatively, it may be possible to try a few measurement techniques and determine which technique leads to meaningful results, hence identifying the corresponding system. These are no general solutions, and the development of specific techniques to determine to which class a network belongs, or of description which can accurately describe most of the networks (*e.g.* the Gaussian assumption for aggregated traffic bit rate in Internet links) is left as an open question.

## 2.6 Appendix

### 2.6.1 Packet pairs in the M/M/1 queue

Consider an M/M/1 queue in steady state with the usual notation. Assume one sends to this system two additional customers at time 0 and $t > 0$ respectively, both with size $x$. Below we assume that $t$ is small and that $x > t$. Let us denote by $V_0$ the system time of the first customer and by $W_t$ that of the second. We are interested in the quantity $\mathbb{E}(V_0 W_t)$. Let $S$ be an exponential random variable with parameter $\mu$. Conditioned on the fact that the first customer finds an empty system, the latter is

$$(1-\lambda t)(x(2x-t)) + \lambda t \mathbb{E}(x(2x+S-t)) + o(t) = x(2x-t) + tx\rho + o(t).$$

Let $A(n)$ be the sum of $n$ independent random variables, all exponential with parameter $\mu$. Conditioned on the fact that the first customer finds $n$ customers in the system, the quantity of interest is

$$\begin{aligned}
&(1-\lambda t)\mathbb{E}[(A(n)+x)(A(n)+2x-t)] + \lambda t \mathbb{E}[(A(n)+x)(A(n)+2x+S-t)] + o(t) \\
&= x(2x-t) + \mathbb{E}[A(n)^2] + (3x-t)\frac{n}{\mu} + \lambda t \left(\frac{n}{\mu} + x\right)\frac{1}{\mu} + o(t) \\
&= x(2x-t) + n\frac{2}{\mu^2} + n(n-1)\frac{1}{\mu^2} + (3x-t)\frac{n}{\mu} + \lambda t \left(\frac{n}{\mu} + x\right)\frac{1}{\mu} + o(t) \quad .
\end{aligned}$$

Hence

$$\begin{aligned}
\mathbb{E}(V_0 W_t) &= x(2x-t) + tx\rho + \frac{1}{\mu^2}\left(\frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{1-\rho}\right) + \\
&\quad + \frac{\rho}{1-\rho}\left(\frac{3x-t}{\mu} + \frac{1}{\mu^2} + \rho t \frac{1}{\mu}\right) + o(t) \\
&= \mathbb{E}(V_0 W_0) - t\left((1-\rho)x + \rho\frac{1}{\mu}\right) \quad ,
\end{aligned}$$

with

$$\mathbb{E}(V_0 W_0) = 2x^2 + \frac{1}{\mu^2}\left(\frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{1-\rho}\right) + \frac{\rho}{1-\rho}\left(\frac{3x}{\mu} + \frac{1}{\mu^2}\right).$$

### 2.6.2 Proof of Lemma 2.4.2

*Let $T \sim \Gamma_{\alpha,\lambda}$, $Z \sim \Gamma_{\beta,\lambda}$ be independent, and set $Y = T + Z$. Then $C = \mathbb{E}[T|Y] = \alpha Y/(\alpha+\beta)$ has density $\Gamma_{\alpha+\beta,(\alpha+\beta)\lambda/\alpha}$, with mean $\mathbb{E}[C] = a/\lambda = \mathbb{E}[T]$.*

*Proof.* From the scaling property of Gamma, $Y \sim \Gamma_{\alpha+\beta,\lambda}$. Since $T$ and $Z$ are independent,

the density of $(T|Y=y)$ is

$$\begin{aligned} P(T=x|Y=y) &= \frac{P(T=x,Y=y)}{P(Y=y)} = \frac{P(T=x,Z=y-x)}{P(Y=y)} \\ &= \frac{\Gamma_{\alpha,\lambda}(x)\Gamma_{\beta,\lambda}(y-x)}{\Gamma_{\alpha+\beta,\lambda}(y)} \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(y-x)^{\beta-1}y^{1-(\alpha+\beta)} \quad . \end{aligned}$$

Recall the *Beta function* $B(x,y) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$. The required conditional expectation is given by

$$\begin{aligned} \mathbf{E}[T|Y=y] =& \frac{y^{1-(\alpha+\beta)}}{B(\alpha,\beta)} \int_0^y x^{\alpha}(y-x)^{\beta-1}\,dx \\ =& \frac{y^{1-(\alpha+\beta)}}{B(\alpha,\beta)} y^{\alpha+\beta} B(\alpha+1,\beta) \\ =& \frac{\alpha\, y}{\alpha+\beta} \end{aligned} \tag{2.18}$$

using the integral identity 3.191(1) from [GR00]. Now viewing $y$ as a sample of $Y$, we have $C = \mathbf{E}[T|Y] = \alpha Y/(\alpha+\beta)$, which is Gamma as stated by the scaling property. □

# Chapter 3

# The Single-path Kelly Network

## 3.1 Introduction

The examples treated in chapter 2 were hardly networks. In the present chapter, we focus on delay based available bandwidth estimation in a point-to-point tomography context. In other words, our problem is to determine, based on the end-to-end delays experienced by probes along a single path, the residual capacity at each node (router) along it. This problem is of particular interest in practice, because it is the simplest measurement scheme one can imagine. It requires neither cooperation from the network nor large scale deployment on a large set of users and coordination between these many end-hosts. In fact, a single pair of end-hosts can measure the path connecting them, provided they have a reliable way to measure the (one-way)[51] probe delays on this path.

There are very few published works in this area. Recently Liu *et al.* [LRLL04, LRL05] provided a rigorous result for available bandwidth of the path, however they focused on the convergence of average available bandwidth estimates to certain simplified fluid model limits, which is of limited practical use, and do not attempt to recover bandwidths for *each* node. As for parametric approaches, we are aware only of [NTV06, SM98, ANT01] which treat only a single node, and do not attempt any validation on network data. As far as we aware of, there is currently no work that aims at estimating the residual bandwidth on *each* node of a path, using pure point-to-point end-to-end delays, without any cooperation from the network nor any path diversity.

This last point is possible because of the (rather strong) assumption that we know the (exponential) parametric family for the probe delay distribution, and hence, our work will be valid only when this assumption (nearly) holds. On a theoretical side, the model we use in this chapter is one of the canonical models in queueing theory. Whilst it is known not to be perfect, it is surprising given the accepted queueing origin of network delays that such a choice has escaped attention until now. One can also note that this approach can most likely

[51] One-way delays are difficult to measure, because it requires a precise clock synchronisation between end-hosts. But the techniques we propose in this chapter obviously work for measuring the round-trip path, using round-trip time of probe packets.

be adapted (with some likely computation difficulties) to most other parametric families for the delay distribution, if one is able to exhibit a "good" parametric model for the delays in a network. For both reasons, one of the main insights in this chapter that we expect to stay valid in practice is the statement that probe delay distribution is sufficient to determine the set of available bandwidth along a path. The exact procedure to estimate it might differ from what is presented here, but this work suggest that the distribution of end-to-end delays contains enough "information" to estimate the available bandwidth on each node.

In additional to our theoretical contributions based on an idealised network model leading to a parametric estimation problem, we investigate both theoretically and using real data the errors induced by deviations from that model. This validation is based on simulations of network using traces from the core network of a tier 1 ISP. Whilst it is clearly simplistic in some aspects and can't be considered as a thoroughly validation, these preliminary results suggest that the proposed technique could be adapted in practice with little modification, at least in the core networks.

The chapter is organized as follows. Section 3.2 is primarily probabilistic. It describes the parametric model, discusses its limitations, and gives its stationary solution leveraging classical results. A first analytical mean-based poly-phase inversion technique, based on is presented in section 3.3. It relies on the interpolation of the expected mean delay, for different probing intensities. Whilst having "good" theoretical results, this method is unstable in presence of error-prone measurement. In section 3.4, we propose an adaptation of the previous technique that takes into account the randomness of the system. This technique is still moment-based and poly-phase. Section 3.5 is primarily statistical, and use an alternative distribution-based mono-phase path for the inversion. Multihop inverse problems are posed in relation to the parametric model, and rigorously solved using maximum likelihood estimators and Expectation-Maximization algorithm. We prove that the associated estimators are asymptotically efficient, and illustrate this using discrete event simulation. Section 3.6 is experimental and queueing theoretic. It uses traces from a core network router to drive experiments exploring estimator accuracy, to determine the influence of the different modelling assumptions, and to test corresponding correction factors which we derive. Section 3.7 concludes this chapter.

## 3.2 The parametric model

### 3.2.1 The system

We first describe the system without its probes. It consists of a Kelly network with $K$ stations $S = s_1, \ldots, s_K = D$ and $K + 1$ routes. Route 0 has an exogenous arrival point process which is Poisson of intensity $\lambda_0$ and follows the path $s_1, \ldots, s_K$. Route $i$, for $i \neq 0$ has an exogenous Poisson arrival process of intensity $\lambda_i$ and its path is the singleton $s_i$. All packets have exponential size with mean 1. The service rate (or the speed) of $s_i$ is $\mu_i$. An instance of the basic setting with a two router path is depicted on Figure 3.1.

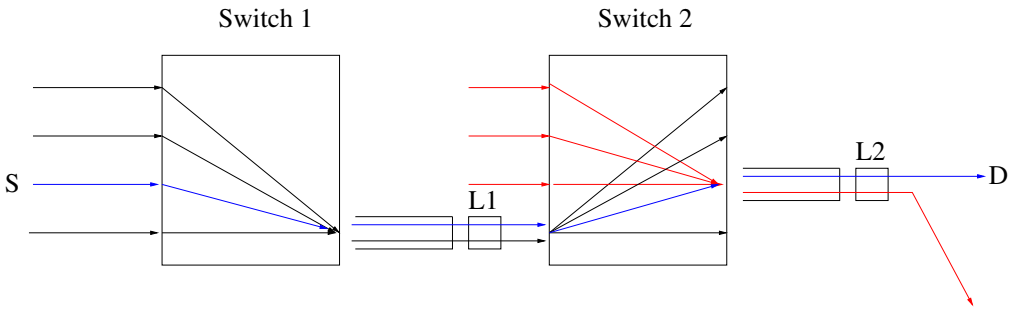



Figure 3.1: Example of path with two routers with non-persistent cross traffic streams, whereas the probes (blue) pass end-to-end.

The prober sends probes according to a Poisson point process with rate $x$ and with exponential sizes with mean 1. Probes follow the same path as flow 0 (namely from S to D). We are hence within the context of point-to-point probing.

The unknown parameters are $\lambda_0, \lambda_1, \ldots, \lambda_K, \mu_1, \ldots, \mu_K$. The observables are the stationary end-to-end delays experienced by the probes.

### 3.2.2 Model Limitations

The adoption of a Kelly network model gives us parametric access to each hop of the path, however it comes at the price of a number of strong assumptions on traffic structure. Some of the most important of these, each of which has the potential to make a large impact on packet delays, are:

- **Routers as FIFO queues** Actual routers may follow complex scheduling disciplines, and real packets experience delays on the incoming side, and contention across the backplane, in addition to the output buffer queueing that the commonly used FIFO model nominally represents.

- **Poisson cross traffic** It is of course well known that Internet traffic is not Poisson (see section 1.1.4), for example both the packet and TCP flow arrival processes exhibit long-range dependence. Although Poisson may nonetheless be a good assumption below some timescale (say 1 second) due to the 'noising' effect of multiplexing tens of thousands of largely independent flows, practical probing schemes will in many cases exceed this timescale due to the need to control the impact on the network, and to collect sufficient samples for reasonable estimation variance.

- **Exponential packet size** It is well known (e.g. [CMT98]) that this distribution is strongly discrete, and can even be modelled as trimodal. For example $S \in \{40, 576, 1500\}$ bytes, with probabilities $(0.5, 0.1, 0.4)$, captured its rough shape well in many cases. This is very far from exponential, however its coefficient to mean ratio $\mathrm{Cov}\,[S] = \sqrt{\mathrm{Var}[S]}/\mathbb{E}[S] \approx 1.05$, which is very close to the 1 of the exponential case.

- **Independence of service times** In real networks packets have a size which, in terms of bytes (ignoring effects like changes in encapsulation), does not change as it traverses the network. In Kelly networks, packet sizes are modelled by service times which are chosen independently at each station.

The errors induced by ignoring the above effects will be explored, challenged, and corrected one by one in section 3.6.

### 3.2.3 The direct equation

Let us first give the stationary distribution of the end-to-end delays of probes, our direct equation within this setting.

Let us denote by $N_i^j$ and $X^j$ the number of packets of class $i$, and the number of probes respectively, in station $j$ in steady state. From the product form of Kelly networks (theorem 1.2.9), we know that if $x + \lambda_0 + \lambda_j < \mu_j$ for all $j$, then

$$\mathbb{P}(X^j = k^j, N_0^j = n_0^j, N_j^j = n_j^j, j = 1, \dots, K) = \prod_{j=1}^{K} \frac{(n_0^j + n_j^j + k^j)!}{n_0^j! n_j^j! k^j!} \frac{\lambda_0^{n_0^j} \lambda_j^{n_j^j} x^{k^j}}{\mu_j^{n_0^j + n_j^j + k^j}} \frac{\mu_j - \lambda_0 - \lambda_j - x}{\mu_j}. \quad (3.1)$$

Let $\gamma_j = \mu_j - \lambda_0 - \lambda_j$ denote the residual bandwidth on station $j$. Direct calculations show that the marginal distribution of the number of probes is

$$\begin{aligned}
&\mathbb{P}(X^j = k^j, j = 1, \dots, K) \\
&\quad = \sum_{n_1^1 \geq 0} \cdots \sum_{n_K^K \geq 0} \sum_{n_0^1 \geq 0} \cdots \sum_{n_0^K \geq 0} \mathbb{P}(X_j = k^j, N_0^j = n_0^j, N_j^j = n_j^j, j = 1, \dots, K) \\
&\quad = \sum_{n_1^1 \geq 0} \cdots \sum_{n_K^K \geq 0} \sum_{n_0^1 \geq 0} \cdots \sum_{n_0^K \geq 0} \prod_{j=1}^{K} \frac{(n_0^j + n_j^j + k^j)!}{n_0^j! n_j^j! k^j!} \frac{\lambda_0^{n_0^j} \lambda_j^{n_j^j} x^{k^j}}{\mu_j^{n_0^j + n_j^j + k^j}} \frac{\gamma_j - x}{\mu_j} \\
&\quad = \prod_{j=1}^{K} \sum_{n_j^j \geq 0} \sum_{n_0^j \geq 0} \frac{(n_0^j + n_j^j + k^j)!}{n_0^j! n_j^j! k^j} \frac{\lambda_0^{n_0^j} \lambda_j^{n_j^j} x^{k^j}}{\mu_j^{n_0^j + n_j^j + k^j}} \frac{\gamma_j - x}{\mu_j} \\
&\quad = \prod_{j=1}^{K} \left(\frac{x}{\mu_j}\right)^{k^j} \frac{\gamma_j - x}{\mu_j} \times \sum_{n_0^j \geq 0} \left(\frac{\lambda_0}{\mu_j}\right)^{n_0^j} \binom{n_0^j + k^j}{n_0^j} \times \\
&\qquad\qquad \sum_{n_j^j \geq 0} \binom{n_0^j + n_j^j + k^j}{n_j^j} \left(\frac{\lambda_j}{\mu^j}\right)^{n_j^j} \quad . \quad (3.2)
\end{aligned}$$

As there is exactly $\binom{n+k}{n}$ $k+1$-tuple of non-negative integers which sum exactly to $n$[52],

[52]To show this, realize that such a $k+1$-tuple can be coded as a sequence of $n+k$ binary symbols with $n$ 0s and $k$ 1s. The first integer is the number of 0s before the first 1, and recursively, the $i^{th}$ integer is the number of 0s between the $i-1^{th}$ and the $i^{th}$ 1.

one can realize that

$$\sum_{n\geq 0}\binom{n+k}{n}x^n = \left(\sum_{n\geq 0}x^n\right)^{k+1} \quad .$$

Hence, (3.2) reads:

$$\begin{aligned}
&\mathbb{P}(X^j = k^j, j = 1,\ldots,K)\\
&\quad = \prod_{j=1}^{K}\left(\frac{x}{\mu_j}\right)^{k^j}\frac{\gamma_j - x}{\mu_j} \times \sum_{n_0^j\geq 0}\left(\frac{\lambda_0}{\mu_j}\right)^{n_0^j}\binom{n_0^j + k^j}{n_0^j}\left(\sum_{k\geq 0}\left(\frac{\lambda_j}{\mu_j}\right)^k\right)^{n_0^j+k^j+1}\\
&\quad = \prod_{j=1}^{K}\left(\frac{x}{\mu_j}\right)^{k^j}\frac{\gamma_j - x}{\mu_j} \times \sum_{n_0^j\geq 0}\left(\frac{\lambda_0}{\mu_j}\right)^{n_0^j}\binom{n_0^j + k^j}{n_0^j}\left(\frac{1}{1-\frac{\lambda_j}{\mu_j}}\right)^{n_0^j+k^j+1}\\
&\quad = \prod_{j=1}^{K}\left(\frac{x}{\mu_j - \lambda_j}\right)^{k^j}\frac{\gamma_j - x}{\mu_j - \lambda_j} \times \sum_{n_0^j\geq 0}\left(\frac{\lambda_0}{\mu_j - \lambda_j}\right)^{n_0^j}\binom{n_0^j + k^j}{n_0^j}\\
&\quad = \prod_{j=1}^{K}\left(\frac{x}{\mu_j - \lambda_j}\right)^{k^j}\frac{\gamma_j - x}{\mu_j - \lambda_j}\left(\frac{1}{1-\frac{\lambda_0}{\mu_j-\lambda_j}}\right)^{k^j+1}\\
&\quad = \prod_{j=1}^{K}\left(\frac{x}{\mu_j - \lambda_j - \lambda_0}\right)^{k^j}\frac{\gamma_j - x}{\mu_j - \lambda_j - \lambda_0}\\
&\mathbb{P}(X^j = k^j, j\in[0;K]) = \prod_{j=1}^{K}\left(\frac{x}{\gamma_j}\right)^{k^j}\frac{\gamma_j - x}{\gamma_j} \quad . \qquad (3.3)
\end{aligned}$$

These equations tell us that our system is equivalent, from the point of view of the probes, to a new system with $K$ M/M/1 stations in series, without any cross–traffic, and where the server of station $j$ has speed $\gamma_j = \mu_j - \lambda_j - \lambda_0$, namely the residual bandwidth on station $j$ in the initial system. From this point on, we will therefore consider such a network. The fact that residual bandwidths are sufficient to characterize (as well as the best one can hope to determine from) stationary end-to-end delays is in line with what was already observed in the 1 station case considered in section 2.3.2.

The generating function of the total number of probes in the (reduced) system in equilibrium is:

$$\psi_N(z) = \prod_{j=1}^{K}\frac{\gamma_j - x}{\gamma_j - xz} \quad . \qquad (3.4)$$

Since probe arrivals are Poisson, PASTA tells us that the distribution of the total number of probes in the system in steady state as given by (3.3) is the same as that just before a probe arrives. The latter also coincides with the probability distribution of the number of probes in the system just after a probe leaves it.

This allows us to state the following lemma:

**Lemma 3.2.1.** *Let $\phi(t)$ denote the density at $t \geq 0$ of the stationary delay $D$ of a probe in*

*the system. Then*

$$\phi(t) = \left(\prod_{i=1}^{K} \gamma_i'\right) \sum_{i=1}^{K} \frac{e^{-\gamma_i' t}}{\prod_{j \neq i} \gamma_j - \gamma_i} \quad , \tag{3.5}$$

*with* $\gamma_i' = \gamma_i - x$.

*In addition, the mean value of* $\overline{D}(x)$ *of the stationary end-to-end delay of a probe in the network is*

$$\overline{D}(x) = \sum_{i=1}^{K} \frac{1}{\gamma_i - x} \quad . \tag{3.6}$$

*Proof.* Let us now consider the system when a tagged probe leaves the system. Since the queueing discipline is FIFO, the number of probes $N$ in the system at that time is equal to the number of probes arrived during the time $D$ the probe spent in the system. So denoting by $\phi(t)$ the density of $D$ at $t \geq 0$, we get:

$$\mathbb{P}(N = k) = \int_0^\infty \phi(t) \mathbb{P}(N = k | D = t)\, dt = \int_0^\infty \phi(t) e^{-xt} \frac{(xt)^k}{k!}\, dt \quad .$$

So the generating function $\psi_N(z)$ of the number of probes in the system at a probe departure epoch verifies:

$$\begin{aligned} \psi_N(z) &= \sum_{k \geq 0} z^k \mathbb{P}(N = k) = \sum_{k \geq 0} \int_0^\infty \phi(t) e^{-xt} \frac{(xtz)^k}{k!}\, dt \\ &= \int_0^\infty \phi(t) e^{-x(1-z)t}\, dt = \mathcal{L}_D\left(x(1-z)\right) \quad , \end{aligned}$$

where $\mathcal{L}_D(z)$ is the Laplace transform of $D$. Hence, setting $s = x(1-z)$ the Laplace transform of the end-to-end delay $D$ is:

$$\mathcal{L}_D(s) = \psi_N\left(1 - \frac{s}{x}\right) = \prod_{j=1}^{K} \frac{\gamma_j - x}{\gamma_j - x + s}\,, \tag{3.7}$$

where we used the fact that $\psi_N$ coincides with the steady state distribution of the number of probes in the system (3.3), so that $\psi_N(z)$ is given by (3.4).

Note that (3.7) is the product of the Laplace transform of exponential variables of parameters $\gamma_j - x$. By injectivity of the Laplace transform of random variables admitting a density, this proves that the end-to-end delay of probes is the sum of independent exponential random variables of parameters $\gamma_j - x$. The mean value is hence $\overline{D} = \sum_j \frac{1}{\gamma_j - x}$. Using the Laplace inversion formula and the residue theorem, and setting $\gamma_i' = \gamma_i - x$,

$$\phi(t) = \frac{1}{2\pi i} \int_{\alpha - i\cdot\infty}^{\alpha + i\cdot\infty} e^{st} \mathcal{L}_D(s)\, ds = \sum \mathrm{Res}\left(e^{st} \prod_{i=1}^{K} \frac{\gamma_i'}{\gamma_i' + s}\right) \quad ,$$

so that using $\alpha = 0$ and then the curve going from $-i\infty$ to $i\infty$ and back on a half circle of

infinite radius in the left half-plane, we get (3.5). □

*Remark.* Note that both the mean delay and the delay distribution of the probes are a function of residual capacities, but not the service or arrival rates, showing that only the $\gamma_j$ are accessible by this technique from the stationary delay distribution.

*Remark.* More general classes of cross–traffic paths can also be considered within this framework. In such an extension, there are as many traffic paths as there are pairs of integers $(i, j)$ with $1 \leq i \leq j \leq K$. A path of type $(i, j)$ brings cross–traffic which is Poisson and enters the network on station $i$ and leaves it from station $K$. The methodology described above works in this more general setting. It is easy to show that the final result is exactly the same as above, namely (3.4) and (3.6) still hold with $\gamma_i$ now equal to $\mu_i - \xi_i$ where $\xi_i$ denotes the sum of the intensities on all paths traversing node $i$.

## 3.3 An analytical solution

### Linear system inversion

In this case, we use a first-moment poly-phase inversion technique, under the following assumption: the prober can measure the mean end-to-end delay of probes for each phase, and the number of stations is known (in real IP networks the latter can be measured by tools such as *traceroute*). We will explain how the prober can compute the coefficients of the polynomial whose roots are the residual bandwidths of each station on the path.

From (3.6) the mean end-to-end delay can be expressed as follows:

$$\overline{D}(x) = \sum_{i=1}^{K} \frac{1}{\gamma_i - x} = \frac{\sum_{k=0}^{K-1} a_k x^k}{\sum_{k=0}^{K} b_k x^k} \quad , \tag{3.8}$$

where $a_k, b_k$ are real numbers defined by

$$\sum_{k=0}^{K} b_k x^k = \prod_{i=1}^{K} (\gamma_i - x)\,, \qquad \sum_{k=0}^{K-1} a_k x^k = \sum_{i=1}^{K} \prod_{j \neq i} (\gamma_j - x).$$

So

$$\begin{aligned} b_k &= (-1)^k \sum_{(i_1,\ldots,i_{K-k}), i_j \neq i_l} \gamma_{i_1} \cdots \gamma_{i_{K-k}}, \\ a_k &= (-1)^k (k+1) \sum_{(i_1,\ldots,i_{K-1-k}), i_j \neq i_l} \gamma_{i_1} \cdots \gamma_{i_{K-1-k}} = (-1)(k+1) b_{k+1} \quad . \end{aligned}$$

The $\gamma_i$s are the roots of the denominator polynomial $\sum_{i=0}^{K} b_k x^k$. Therefore, if we identify the $b_k$ variables, we have solved the inverse problem that consists in determining all residual bandwidths from the observations.

We now show how to find the coefficients of the polynomial. Assume we have $K$ perfect measurements $d_j = \overline{D}(x_j)$ of the mean delays for $K$ different values $x_1, \ldots, x_K$ of the probe rate (we will consider the situation with a number of phases larger than $K$ in section 3.4.0.0). The method is hence moment-based and poly-phase. We want to find $(b_k)_{k=0,\cdots,K}$ such that:

$$\forall j = 1, \ldots, K, \quad d_j = \frac{\sum_{k=0}^{K-1} a_k x_j^k}{\sum_{k=0}^{K} b_k x_j^k} = \frac{\sum_{k=0}^{K-1} -(k+1) b_{k+1} x_j^k}{\sum_{k=0}^{K} b_k x_j^k} \quad . \tag{3.9}$$

Rational fractions are defined up to a multiplicative factor: we can hence always assume that $b_K = 1$. The system is now equivalent to:

$$\forall j = 1, \ldots, K, \quad \sum_{k=0}^{K-1} d_j x_j^k b_k + \sum_{k=1}^{K-1} k x_j^{k-1} b_k = -d_j x_j^K - K x_j^{K-1} \quad , \tag{3.10}$$

which can be written as the matrix equation $Y = XB$, where $X$ is the $K \times K$ square matrix

$$X_{j,k} = ((k-1) x_j^{k-2} + d_j x_j^{k-1}), \quad j, k = 1, \ldots, K$$

and $Y$ (resp. $B$) the column vector $Y_j = -K x_j^{K-1} - d_j x_j^K$ (resp. $B_j = b_{j-1}$). When $X$ is invertible, there is only one solution $B = X^{-1} Y$.

We lack sufficient conditions for $X$ to be invertible. The prober will therefore have to continue adding phases until $X$ becomes invertible.

**Numerical illustration** Table 3.1 gives some numerical results for this method. The first column indicates the ground truth, i.e. the real values of $(\gamma_1, \ldots, \gamma_K)$. The second column specifies the probing intensities that were used, that is the vector $(x_1, \ldots, x_K)$. The third column consists of the coefficients of the polynomial $\sum_{i=0}^{K} b_i x^i$, which we write as the vector $B^t = (b_0, \ldots, b_{K-1})$. Finally, the last column gives the estimation of our method, i.e. the values of $(\widehat{\gamma}_1, \ldots, \widehat{\gamma}_K)$. The technique was implemented using Maple, and provides accurate results in all the cases we tried. However, with 7 (or even 5) stations, one can already notice some rounding errors in the calculations. These errors, which stem both from the inversion of the matrix $X$ and the determination of the roots of the polynomial $\sum_{k=0}^{K} b_k x^k$, grow as the number of stations increases.

## 3.4 Noise Aware moment-based solution

#### Minimizing quadratic-like error in Kelly networks

The setting is that of section 3.3, but we now take into account the fact that the variable $d_j$ in (3.9) is some error-prone measurement of the stationary mean delays of the probes of phase $j$. Assuming that the linear system is of full rank, (3.10) has still one unique solution.

| Ground truth | Intensities | vector B | Estimation |
|---|---|---|---|
| (10, 30, 70) | (1, 2, 7) | $(-21000,\ 3100,\ -110)$ | (10, 30, 70) |
| (10, 25, 30, 60, 70) | (0.3, 1, 2, 4, 7) | $(-3.15 \times 10^7,\ 6.43 \times 10^6,$ $-4.49 \times 10^5,\ 1390,\ -195)$ | (10, 25, 29.99, 60.08, 69.92) |
| (10, 12, 25, 30, 60, 85, 130) | (0.001, 0.3, 1, 2, 4, 7, 9.7) | $(-6 \times 10^{10},\ 1.76 \times 10^{10},$ $-1.97 \times 10^9,$ $1.08 \times 10^8,\ -3.12 \times 10^6,$ $4.74 \times 10^4,\ -354)$ | (10, 12, 25.05, 29.84, 62.72, 78.78, 135.3) |

Table 3.1: Linear inversion in Kelly networks: numerical results

However, as shown in Table 3.2, the method is extremely sensitive to the presence of noise, and solutions are meaningless with as little as 1% error in the measurements.

This sensitivity to noise is due to several reasons: first, the algorithm finds one exact rational fraction, but this fraction interpolates the noised measurements (this is the overfitting phenomenon). Second, the imprecision is multiplied when taking the inverse of $X$ and then when finding the roots of the polynomial. The concatenation of these operations is quite unstable.

In order to prevent the overfitting phenomenon, we explored the classical solution consisting in increasing the number of measurements. Let us assume we have $N > K$ error-prone measures $d_j = \overline{D}(x_j)$ of the mean delays for $N$ different values $x_1, \ldots, x_N$ of the probing rate.

Following the same lines as in the previous section, we arrive at the matrix equation $\tilde{X}B = \tilde{Y}$, where $\tilde{X}$ is the $N \times K$ matrix with $(i, k)$ entry equal to $(k-1)x_i^{k-2} + d_i x_i^{k-1}$, and where $\tilde{Y}$ is the $N \times 1$ vector with $i$ entry $-Kx_i^{K-1} - d_i x_i^K$.

This corresponds to a multiple linear regression, with more measurements than parameters. There is often no unique solution to such a system. A common way to circumvent this difficulty is to select the value $\widehat{B}$ that minimizes the sum of the square errors in each

| Ground truth | Vector B | Estimation |
|---|---|---|
| (10, 30, 70) | $(6564, -938, 19.9)$ | (-44.4, 10.9, 13.6) |
| (10, 25, 30, 60, 70) | $(-14405, 3039,$ $-358, 86, -15.82)$ | $(-2.46 - 5.22i, -2.46 + 5.22i,$ $6.191 - 3.66i, 6.191 + 3.66i, 8.35)$ |
| (10, 12, 25, 30, 60, 85, 130) | $(1.55 \times 10^6, -3.82 \times 10^5,$ $3100, 1186,$ $-891, 232, -25.5)$ | $(-3.42, 0.1 - 4i, 0.1 + 4i,$ $5.31 - 2.62i, 5.31 + 2.62i,$ $8.21, 9.91)$ |

Table 3.2: Numerical results for linear interpolation. Delays are measured with 1% error (half with 1% more, half with 1% less). Intensities are similar to the ones used in Table 3.1.

equation:

$$\widehat{B} = \min_{B}(\tilde{Y} - \tilde{X}B)^t(\tilde{Y} - \tilde{X}B) = \min_{(b_0,\ldots,b_{K-1},1)} \sum_{j=1}^{N} \left[ \sum_{k=0}^{K} (kx_j^{k-1} + d_j x_j^k) b_k \right]^2 \quad . \quad (3.11)$$

The least squares error solution to (3.11) is $\widehat{B} = (\tilde{X}^t\tilde{X})^{-1}\tilde{X}^t\tilde{Y}$.

Notice that finding the coefficients $b_k$ which minimize the sum in (3.11) is not equivalent to minimizing the square of the differences between the left hand side and the right hand side of (3.9). We have in fact multiplied the $j$-th difference by $\sum_{k=0}^{K} b_k x_j^k = \prod_{i=1}^{K} (\gamma_i - x_j)$ before looking for the minimum. The last product is positive and decreasing as $x_j$ increases, so that we put more weight on less intrusive measures. There are several other ways of estimating $B$ (e.g. through total least square methods [GL80]) and it would be interesting to compare them. We will not pursue this line of thought as the last step of the inversion method (that consisting in determining the zeros of a polynomial from its coefficients) is in any case likely to be unstable, as illustrated by the following numerical example.

**Numerical illustration** A Maple implementation indicates that the overfitting correction is not sufficient. We still get complex roots to the polynomial. We conjecture that this is due to the instability when inverting the matrix $\tilde{X}^t\tilde{X}$ and when finding the roots of the polynomial. A small error in the measured delay is amplified by the matrix inversion, and it is well-known that a small difference in the coefficients of a polynomial can have a huge impact on its roots. Table 3.3 provides a few numerical results for the 3 station case. This instability motivates the maximum likelihood methods studied in the next section.

| $N$ | **Intensities** | **Vector B** | **Estimation** |
|---|---|---|---|
| 3 | (1, 2, 7) | $(6563, -938, 19.9)$ | (-44.4, 10.9, 13.6) |
| 5 | (0.3, 1, 2, 4, 7) | $(-6075, 914, -39.8)$ | $(9.8, 14.99 \pm 19.9i)$ |
| 7 | (0.001, 0.3, 1, 2, 4, 7, 9.7) | $(-9583, 1417, -55.14)$ | $(9.88, 22.6 \pm 21.4i)$ |
| 10 | (0.001, 0.3, 0.5, 1, 2, 4, 4.3, 7, 8.7, 9.7) | $(-10766, 1610, -62.7)$ | $(9.9, 26.4 \pm 19.8i)$ |

Table 3.3: Least squares linear regression in the 3 servers case. The ground truth is (10, 30, 70). Error in mean delay is 1%.

## 3.5 Maximum likelihood estimators

The network and its probes are as in sections 3.3 and 3.4. The observables are now a finite time series of probe end-to-end delays and not an exact moment or distribution as in that section. In this section, we will assume that all samples are identically distributed (i.e. we assume stationarity) and *independent*. The latter assumption is of course not true in general as samples collected at two epochs with a finite time difference are in fact (Markov)

correlated. However, if inter-probe times are chosen larger in mean than the mixing time of the system, then it is justified to assume independence.

*Remark.* It is a well-known that the mixing times of heavily loaded queues can be potentially very large. We refer to the classical litterature (*e.g.* [AW87, AW94, Mor55, AW88]) for more details.

Lemma 3.2.1 showed that the probability density function $\phi(d)$ at $d \geq 0$ of the stationary delay $D$ of a probe in the system is,

$$\phi_{\gamma_1,\ldots,\gamma_K}(d) = \left(\prod_{i=1}^{K} \gamma_i'\right) \sum_{i=1}^{K} \frac{e^{-\gamma_i' d}}{\prod_{j\neq i}(\gamma_j' - \gamma_i')} \quad ,$$

with $\gamma_i' = \gamma_i - x$.

The problem can hence be viewed as a classical statistical problem, that of fitting distributions of this class.

### 3.5.1 The one station case

For $K = 1$, one can somewhat simplify the notation: the speed of the link is $\mu$; the cross–traffic intensity is $\lambda$ and the probe intensity is $x$. The system is a FIFO M/M/1 queue. The distribution of the delay $D$ of probes is exponential of parameter $\gamma' = \mu - \lambda - x$, namely it admits the density $\phi_\gamma(d) = \gamma' e^{-\gamma' d}$, for all $d \geq 0$. Assume we have several independent delay samples $(d_1, \ldots, d_n)$. Let $\mathbf{d} = (d_1, \ldots, d_n)$. For independent probe delays, the likelihood of the parameter $\gamma$ is defined as:

$$f_{\mathbf{d}}(\gamma) = \prod_{i=1}^{n} \phi_\gamma(d_i) = {\gamma'}^n e^{-\gamma' \sum_{i=1}^{n} d_i} \quad .$$

The maximum likelihood estimator of the parameter $\widehat{\gamma}$ is the maximum of the likelihood function. This function is positive, and has 0 as a limit when $\gamma'$ tends to 0 or to $\infty$. At any local maximum, and therefore at $\widehat{\gamma}$, we have $\frac{df_{\mathbf{d}}(\gamma)}{d\gamma} = 0$, which is equivalent to:

$$n\widehat{\gamma'}^{n-1} e^{-\widehat{\gamma}' \sum_{i=1}^{n} d_i} - \widehat{\gamma'}^n \sum_{i=1}^{n} d_i e^{-\widehat{\gamma}' \sum_{i=1}^{n} d_i} = 0 \quad .$$

Hence

$$\widehat{\gamma}' = \frac{n}{\sum_{i=1}^{n} d_i} = \frac{1}{\overline{d}} \quad . \tag{3.12}$$

The maximum of likelihood for the available bandwidth is hence: $\widehat{\mu} - \widehat{\lambda} = \widehat{\gamma} = \frac{1}{\overline{d}} + x$. This together with the strong law of large numbers show asymptotic consistency: *i.e.* the estimator converges to the ground truth when the number of probes tends to infinity.

### 3.5.2 The two stations case

In what follows, we will use the notation $\gamma_i$ to mean $\gamma_i'$ for the sake of notational simplification.

We first evaluate the log likelihood function and then pose the likelihood equations (3.15). The key results are (i) the fact that (3.15) allow one to determine the MLE estimator and (ii) that the latter is asymptotically efficient (Theorem 3.5.1). This convergence is illustrated by simulation results.

The end-to-end delay of a probe is the sum of two independent exponential random variables of parameter $\gamma_1$ and $\gamma_2$ (see Eq. (3.5)). Its density at $d > 0$ is hence

$$\phi_{\gamma_1,\gamma_2}(d) = \gamma_1\gamma_2\frac{e^{-\gamma_1 d} - e^{-\gamma_2 d}}{\gamma_2 - \gamma_1} \quad . \tag{3.13}$$

If $\gamma_2 = \gamma_1 = \gamma$ (which has essentially no chance of occurring in practice) the density becomes $\gamma^2 d e^{-\gamma d}$, which coincides with the limit $\gamma_2 \to \gamma_1$ of (3.13).

The likelihood function when we have $n$ independent probe delays $(d_1, \ldots, d_n) = \mathbf{d}$ is

$$f_{\mathbf{d}}(\gamma_1, \gamma_2) = \prod_{i=1}^{n} \phi_{\gamma_1,\gamma_2}(d_i) \quad . \tag{3.14}$$

We proceed as above by determining the values of the residual capacities that maximize the log-likelihood function $\log f$:

$$\log f_{\mathbf{d}}(\gamma_1, \gamma_2) = n\left(\log(\gamma_1) + \log(\gamma_2) - \log(\gamma_2 - \gamma_1)\right) + \sum_{i=1}^{n} \log\left(e^{-\gamma_1 d_i} - e^{-\gamma_2 d_i}\right) \quad .$$

At any local extremum, therefore at $(\widehat{\gamma}_1, \widehat{\gamma}_2)$, we have:

$$\begin{aligned}
\left.\frac{\partial \log f_{\mathbf{d}}(\gamma_1, \gamma_2)}{\partial \gamma_1}\right|_{\widehat{\gamma}_1,\widehat{\gamma}_2} &= 0 = \frac{n\widehat{\gamma}_2}{\widehat{\gamma}_1(\widehat{\gamma}_2 - \widehat{\gamma}_1)} - \sum_{i=1}^{n} \frac{d_i}{1 - e^{-(\widehat{\gamma}_2 - \widehat{\gamma}_1)d_i}} \\
\left.\frac{\partial \log f_{\mathbf{d}}(\gamma_1, \gamma_2)}{\partial \gamma_2}\right|_{\widehat{\gamma}_1,\widehat{\gamma}_2} &= 0 = \frac{-n\widehat{\gamma}_1}{\widehat{\gamma}_2(\widehat{\gamma}_2 - \widehat{\gamma}_1)} + \sum_{i=1}^{n} \frac{d_i}{e^{(\widehat{\gamma}_2 - \widehat{\gamma}_1)d_i} - 1} \quad .
\end{aligned} \tag{3.15}$$

These equations, which are instrumental in determining the MLE numerically, will be referred to as the *likelihood equations* in what follows. Here are important observations: under the natural non degeneracy assumption satisfied here, the value of $\widehat{\gamma}_1, \widehat{\gamma}_2$ which maximizes the likelihood is a stationary point, namely a solution of the likelihood equation. However, even in this simple two station case, there may be spurious solutions to this equation, like e.g. local maxima or minima or saddle points. So for locating the global maximum (i.e. the ML estimator) one should first determine all the solutions of the likelihood equation and then determine the solution with maximal likelihood. More can be said on the matter when

the number of samples is large. Setting $X = \frac{\widehat{\gamma}_1}{\widehat{\gamma}_2}$ and $Y = (\widehat{\gamma}_2 - \widehat{\gamma}_1)$, (3.15) now read:

$$\frac{1}{X} = \frac{Y}{n}\sum_{i=1}^{n} \frac{d_i}{1 - e^{-d_i Y}}, \quad X = \frac{Y}{n}\sum_{i=1}^{n} \frac{d_i}{e^{d_i Y} - 1} \quad . \tag{3.16}$$

Note that this transformation is reversible, and we have

$$\begin{aligned} \widehat{\gamma}_2 =& \frac{Y}{1 - X} \quad \text{and} \\ \widehat{\gamma}_1 =& \frac{XY}{1 - X} \quad . \end{aligned} \tag{3.17}$$

Multiplying both equations, we get that $Y$ is a solution of the fixed point equation

$$Y = g(Y) = \frac{1}{\sqrt{\left(\frac{1}{n}\sum_{i=1}^{n} \frac{d_i}{1 - e^{-d_i Y}}\right)\left(\frac{1}{n}\sum_{i=1}^{n} \frac{d_i}{e^{d_i Y} - 1}\right)}} \quad . \tag{3.18}$$

Notice that 0 is always a solution of (3.18), when extending the right hand side by continuity. Once a non-zero solution $Y$ of (3.18) is obtained, $X$ is derived from (3.16) and this gives a non degenerate solution to (3.15). In general (3.18) can have either no other solution (than 0), or several other solutions, depending on $n$ and on the sequence of random samples which are chosen. However, the situation simplifies significantly when $n$ is large. Assume that $\gamma_2 > \gamma_1$. Then, by the strong law of large numbers, for all $Y > 0$,

$$\begin{aligned} \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \frac{d_i}{1 - e^{-d_i Y}} =& \mathbb{E}\left(\frac{D}{1 - e^{-DY}}\right) \\ =& \frac{\gamma_1\gamma_2}{\gamma_2 - \gamma_1}\int_0^{\infty} \frac{t}{1 - e^{-Yt}}(e^{-\gamma_1 t} - e^{-\gamma_2 t})dt \\ =& \frac{\gamma_1\gamma_2}{\gamma_2 - \gamma_1}\sum_{k\geq 0}\left(\frac{1}{(\gamma_1 + kY)^2} - \frac{1}{(\gamma_2 + kY)^2}\right) \quad . \end{aligned}$$

Similarly

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \frac{e^{-d_i Y} d_i}{1 - e^{-d_i Y}} = \frac{\gamma_1\gamma_2}{\gamma_2 - \gamma_1}\sum_{k\geq 1}\left(\frac{1}{(\gamma_1 + kY)^2} - \frac{1}{(\gamma_2 + kY)^2}\right) \quad .$$

Hence, for $n$ large, (3.18) is approximately equivalent to

$$\frac{1}{\frac{\gamma_1\gamma_2}{\gamma_2 - \gamma_1}\sqrt{\xi(0)\xi(1)}} - Y = 0 \tag{3.19}$$

with $\xi(i) = \sum_{k\geq i}\left(\frac{1}{(\gamma_1 + kY)^2} - \frac{1}{(\gamma_2 + kY)^2}\right)$. It is easy to show that (3.19) always admits 0 and $\gamma_2 - \gamma_1$ as solutions. The function on the L.H.S of (3.19) is depicted in Figure 3.2 where one sees that 0 and $\gamma_2 - \gamma_1$ are the only solutions. Hence, we argue that for $n$ large enough, spurious solutions will concentrate around 0 so $\gamma_2 - \gamma_1$ will be the only other solution.

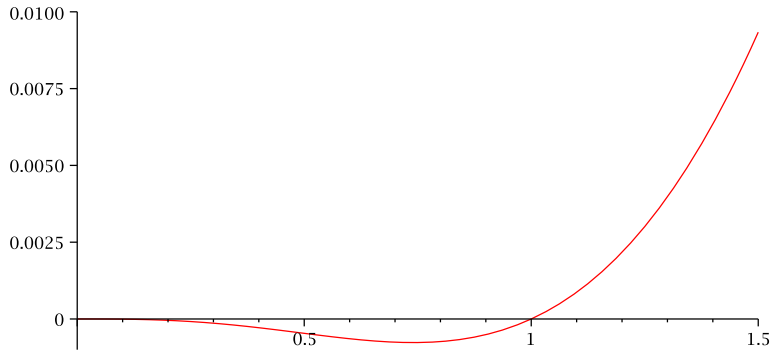



Figure 3.2: Shape of the fixed point equation: LHS of (3.19).

*Remark.* This techniques does not hold when $\gamma_1 = \gamma_2$. In particular, from equations (3.17), we can see that the estimation in this case would be $\widehat{\gamma_1} = \widehat{\gamma_2} = 0$. However, this problem can happen only when the vector of end-to-end delays is sampled exactly according to a theoretical distribution with $\gamma_1 = \gamma_2$. This equality is unlikely to happen (as it substracts traffic intensities, which is unlikely to be equal on different links), and the probability of having a negligible noise in the measured data is low. Additionnally, this question is solved with the technique proposed in section 3.5.3.

The main result on this MLE approach is:

**Theorem 3.5.1.** *The MLE $(\widehat{\gamma}_1, \widehat{\gamma}_2)$ is asymptotically consistent. That is, $(\widehat{\gamma}_1, \widehat{\gamma}_2)$ almost surely converges to the true parameters $(\gamma_1, \gamma_2)$ when the number of samples $n$ tends to infinity.*

*Proof.* The proof relies on Theorem 1.4.13 and Lemma 1.4.14 which state that if

1. $\phi_{\psi_1,\psi_2}(d)$ is continuous in $(\psi_1, \psi_2)$ for every $d$;
2. $\forall \theta \neq (\gamma_1, \gamma_2), \exists N_\theta$ open set s.t. $\theta \in N_\theta$ and
$$\mathbf{E}_{\gamma_1,\gamma_2}\left[\inf_{\psi \in N_\theta} \log\left(\frac{\phi_{\gamma_1,\gamma_2}(d)}{\phi_{\psi_1,\psi_2}(d)}\right)\right] > -\infty;$$
3. The parameter space $\Omega$ is a compact set,

then the MLE estimator $(\widehat{\gamma}_1, \widehat{\gamma}_2)$ converges almost surely to the true parameters $(\gamma_1, \gamma_2)$. In the last expression and below, $\mathbf{E}_{\gamma_1,\gamma_2}[g(d)]$ means integration of the function $g(d)$ w.r.t. the density $\phi_{\gamma_1,\gamma_2}(.)$.

Let us show that our problem verifies the conditions of the theorem. The function: $\phi_{\gamma_1,\gamma_2}(d)$ is continuous in $(\gamma_1, \gamma_2)$, so that Property 1 is verified. By convexity of the exponential function, for all $a < b$ real, $(b-a)xe^{-bx} \leq e^{-ax} - e^{-bx} \leq (b-a)xe^{-ax}$. Therefore,

$$\gamma_1\gamma_2 d e^{-\gamma_2 d} \leq \phi_{\gamma_1,\gamma_2}(d) \leq \gamma_1\gamma_2 d e^{-\gamma_1 d} \quad , \tag{3.20}$$

up to a re-ordering of $\gamma_1$ and $\gamma_2$. Therefore, we have:

$$\frac{\gamma_1\gamma_2}{\psi_1\psi_2}e^{(\psi_1-\gamma_2)d} \leq \frac{\phi_{\gamma_1,\gamma_2}(d)}{\phi_{\psi_1,\psi_2}(d)} \quad .$$

This implies

$$\inf_{\psi\in N_\theta}\left(\log\left(\frac{\gamma_1\gamma_2}{\psi_1\psi_2}\right)+(\psi_1-\gamma_2)d\right) \leq \inf_{\psi\in N_\theta}\log\frac{\phi_{\gamma_1,\gamma_2}(d)}{\phi_{\psi_1,\psi_2}(d)} \quad .$$

Since $\mathbf{E}_{\gamma_1,\gamma_2}[d] = \left(\frac{1}{\gamma_1}+\frac{1}{\gamma_2}\right)$, we have

$$\log\left(\frac{\gamma_1\gamma_2}{\sup_{\psi\in N_\theta}\psi_1 \sup_{\psi\in N_\theta}\psi_2}\right) - (\gamma_2 - \inf_{\psi\in N_\theta}\psi_1)(\gamma_1^{-1}+\gamma_2^{-1}) \leq \mathbf{E}_{\gamma_1,\gamma_2}\left[\inf_{\psi\in N_\theta}\log\left(\frac{\phi_{\gamma_1,\gamma_2}(d)}{\phi_{\psi_1,\psi_2}(d)}\right)\right] \quad .$$

Hence for all bounded open sets $N_\theta$,

$$\mathbf{E}_{\gamma_1,\gamma_2}\left[\inf_{\psi\in N_\theta}\log\left(\frac{\phi_{\gamma_1,\gamma_2}(d)}{\phi_{\psi_1,\psi_2}(d)}\right)\right] > -\infty \quad ,$$

so that Property 2 is verified. Finally, remember that the parameters are residual bandwidth. Therefore, without losing any meaningful solution, we can restrict the natural parameter space $]0;\infty[^2$ to a space $[\epsilon, A]^2$, where $\epsilon$ is a very small capacity (for example, 1 packet per year) and $A$ is the highest capacity of existing routers. □

Theorem 1.4.13 is in fact more general, and that Property 3 can be replaced by the following: $\exists C \subseteq \Omega$ compact set s.t. $(\gamma_1, \gamma_2) \in C$ and

$$\mathbf{E}_{\gamma_1,\gamma_2}\left[\inf_{\psi\in\Omega\setminus C}\log\left(\frac{\phi_{\gamma_1,\gamma_2}(d)}{\phi_{\psi_1,\psi_2}(d)}\right)\right] > 0 \quad ,$$

which would allow us to consider any positive value as an acceptable parameter. We are confident that the general form of the theorem holds, and simulations were consistent with this. We choose to use the restricted parameter space because when $\epsilon$ and $A$ are well chosen, the restricted parameter space includes all meaningful parameters for the system we consider in practice. Therefore, restricting the parameter space is equivalent to rejecting solutions that we know to be impossible. The question whether the result still holds when taking $\Omega = ]0;\infty[^2$ is still open.

**Numerical illustration**

We now evaluate the MLE by simulation where delays are generated according to the theoretical law. Residual capacity estimates are obtained using the following technique inspired by the above: we numerically locate the first zero of (3.18) which is not in the neighborhood

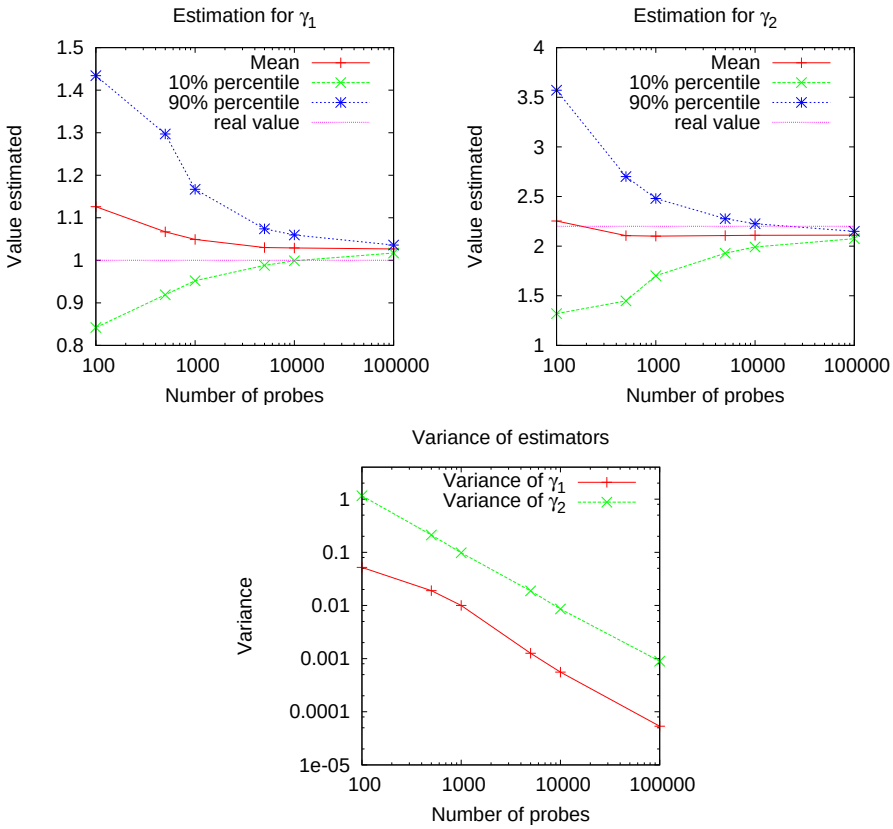



Figure 3.3: Precision of the estimated $\gamma_1 = 1$ (left) and $\gamma_2 = 2.2$ (middle), and variances (right, note log scale) as a function of $n$

of the origin. We use a stopping precision of $10^{-4}$ in the procedure for finding this zero (a value of $10^{-8}$ produced the same estimator). In each case results are averaged over 1000 independent experiments.

Figure 3.3 plots $\widehat{\gamma}_1$ and $\widehat{\gamma}_2$ when $(\gamma_1, \gamma_2) = (1, 2.2)$ as a function of the number of probes $n$. The results are quite satisfying: for 1000 samples 80% of estimates have error below 10%, and this drops to 4% for 100000 probes. It is clear that the estimation variance drops, and the right hand plot shows that it does so as $O(1/n)$ as expected. Notice that $\gamma_2 - \gamma_1$ is underestimated. The bias decreases with $n$ also, though this is less obvious in the plots since the decay is much slower than the decay of variance. In other words, the MSE is dominated by the bias for large $n$. If instead we use $(\gamma_1, \gamma_2) = (1, 7.4)$ we approximately obtain the same precision for $\widehat{\gamma}_2$ and improved precision for $\widehat{\gamma}_1$.

It is well known, and to be expected, that the maximum likelihood estimator can be biased (although the consistency property implies that asymptotically it is not). For example in the case of a single server of residual capacity $\gamma$ and a single probe, the estimator $\widehat{\gamma}$ is simply the inverse of the probe delay $D$. By convexity of the function $f(x) = \frac{1}{x}$, we get:

$$\mathbf{E}\left[\widehat{\gamma}\right] = \mathbf{E}\left[\frac{1}{D}\right] > \frac{1}{\mathbf{E}[D]} = \gamma \quad .$$

*Remark.* The bias depends obviously of the metric one wants to estimate. Say, for example, that the prober is interested in the mean time $\alpha_j$ spend in each station $j$. As each server behaves as an M/M/1 queue, and from the function invariance of the maximum likelihood estimator (see Lemma 1.4.12), we have that

$$\alpha_j = \frac{1}{\gamma_j} \quad \text{and} \quad \widehat{\alpha_j} = \frac{1}{\widehat{\gamma_j}} \quad .$$

It follows obviously in the same single server case that

$$\mathbf{E}\left[\widehat{\alpha}\right] = \mathbf{E}\left[D\right] = \alpha \quad ,$$

which shows that, contrary to the residual bandwidth estimator, the MLE of the mean-time spent in the server is unbiased.

### More than two stations

This section is focused on the generalization to a path with $K$ routers. We follow the same approach as for the two station case. We still use $\gamma_i$ in place of $\gamma_i'$.

According to (3.5), the likelihood function for $n$ independent end-to-end probe delays $d_1, \ldots, d_n$ is

$$\begin{aligned} f_{\mathbf{d}}(\gamma_1, \ldots, \gamma_K) =& \prod_{i=1}^{n} \sum_{j=1}^{K} \left( \prod_{k \neq j} \frac{\gamma_k}{\gamma_k - \gamma_j} \right) \gamma_j e^{-\gamma_j d_i} \\ f_{\mathbf{d}}(\gamma_1, \ldots, \gamma_K) =& \left( \prod_{p=1}^{K} \gamma_p^n \right) \prod_{i=1}^{n} \sum_{j=1}^{K} \left( \prod_{k \neq j} \frac{1}{\gamma_k - \gamma_j} \right) e^{-\gamma_j d_i} \quad . \end{aligned}$$

Therefore, we get the following expression for the log likelihood:

$$\log\left(f_{\mathbf{d}}(\gamma_1, \ldots, \gamma_K)\right) = n \sum_{p=1}^{K} \ln(\gamma_p) + \sum_{i=1}^{n} \log \left( \sum_{j=1}^{K} \left( \prod_{k \neq j} \frac{1}{\gamma_k - \gamma_j} \right) e^{-\gamma_j d_i} \right) \quad , \quad (3.21)$$

so that the likelihood equation reads:

$$\begin{aligned} \frac{\partial \log\left(f_{\mathbf{d}}\gamma_1, \ldots, \gamma_K\right))}{\partial \gamma_l} =& \frac{n}{\gamma_l} + \sum_{i=1}^{n} \frac{1}{\sum_{j=1}^{K} \left( \prod_{k \neq j} \frac{1}{\gamma_k - \gamma_j} \right) e^{-\gamma_j d_i}} \times \\ & \left[ \left( e^{-\gamma_l d_i} \prod_{k \neq l} \frac{1}{(\gamma_k - \gamma_l)} \sum_{k \neq l} \frac{1}{(\gamma_k - \gamma_l)} \right) - \right. \\ & \left. \left( d_i e^{-\gamma_l d_i} \prod_{k \neq l} \frac{1}{(\gamma_k - \gamma_l)} \right) - \left( \sum_{j \neq l} \frac{e^{-\gamma_j d_i}}{\gamma_l - \gamma_j} \prod_{k \neq j} \frac{1}{\gamma_k - \gamma_j} \right) \right] \quad . \end{aligned}$$

We found no closed form solution to this system of equation, and instead turn to the

Expectation-Maximization algorithm considered below.

### 3.5.3 Expectation-Maximization Algorithm

The Maximum Likelihood estimator is very often analytically difficult or even impossible to derive. One way to overcome this difficulty is to use Expectation-Maximization (E-M), which we presented in section 1.4.4. The use of E-M algorithm for fitting general phase-type distributions was first described by Asmussen *et al.* in [ANO96]. The setting considered in the present paper, namely the fitting of sums of independent exponential random variables, is much more specific and this allows us to give explicit iteration formulas and also to prove the convergence of the algorithm, which has not been done for general phase-type distributions to the best of our knowledge.

#### The two station case

In the two link case, the *incomplete data* are the end-to-end delays $d_i$ of probes, $i = 1, \ldots, n$. We *complete* them by the delay on the first link $l_i$ for all probes, $i = 1, \ldots, n$, and $\mathbf{l} = (l_1, \ldots, l_n)$ denotes their vector. The section starts with the definition of the E-M algorithm in this setting, and then shows that it converges to a solution of the likelihood equation. This proof, which is one of the main mathematical results of this chapter, is structured in two lemmas 3.5.2 and 3.5.3.

Let

$$Q_{\mathbf{d}}(\theta_1, \theta_2 | \gamma_1, \gamma_2) = \mathbb{E}_{\phi_{\gamma_1,\gamma_2}(\mathbf{l}|\mathbf{d})} \log \left( \widetilde{f}_{\mathbf{l},\mathbf{d}}(\theta_1, \theta_2) \right) \quad , \tag{3.22}$$

where $\widetilde{f}_{\mathbf{l},\mathbf{d}}(\theta_1, \theta_2) = \phi_{\theta_1,\theta_2}(d_1, l_1, \ldots, d_n, l_n)$ is the complete likelihood of the complete data and $\widetilde{l}_i = d_i - l_i$ is the delay on the second link,

The E-M-algorithm can be defined as follows:

**E-M Algorithm:** *Take any random* $(\gamma_1^{(0)}, \gamma_2^{(0)})$ *and for each positive integer* $k$*, do the following:*

- *Expectation step: compute* $Q_{\mathbf{d}}(\theta_1, \theta_2 | \gamma_1^{(k)}, \gamma_2^{(k)})$ .
- *Maximization step: compute*

$$(\gamma_1^{(k+1)}, \gamma_2^{(k+1)}) = \underset{(\theta_1,\theta_2)}{\operatorname{argmax}} Q_{\mathbf{d}}(\theta_1, \theta_2 | \gamma_1^{(k)}, \gamma_2^{(k)}) \quad . \tag{3.23}$$

The following lemma illustrates the tractability of this approach:

**Lemma 3.5.2.** *In the two router case, for all* $k \geq 0$*, (3.23) is equivalent to*

$$\frac{1}{\gamma_1^{(k+1)}} = \frac{1}{n} \sum_{i=1}^{n} \frac{d_i e^{(\gamma_2^{(k)} - \gamma_1^{(k)}) d_i}}{e^{(\gamma_2^{(k)} - \gamma_1^{(k)}) d_i} - 1} - \frac{1}{\gamma_2^{(k)} - \gamma_1^{(k)}} \quad , \tag{3.24}$$

*and*

$$\frac{1}{\gamma_2^{(k+1)}} = \frac{1}{\gamma_2^{(k)} - \gamma_1^{(k)}} - \frac{1}{n}\sum_{i=1}^{n} \frac{d_i}{e^{(\gamma_2^{(k)} - \gamma_1^{(k)})d_i} - 1} \quad . \tag{3.25}$$

*Proof.* We have

$$\begin{aligned}
\phi_{\gamma_1,\gamma_2}(l|d) =& \frac{\phi_{\gamma_1,\gamma_2}(l,d)}{\phi_{\gamma_1,\gamma_2}(d)} \\
=& \frac{\gamma_1\gamma_2 e^{-\gamma_1 l} e^{-\gamma_2(d-l)}}{\gamma_1\gamma_2 \frac{e^{-\gamma_1 d} - e^{-\gamma_2 d}}{\gamma_2 - \gamma_1}} \\
\phi_{\gamma_1,\gamma_2}(l|d) =& \frac{(\gamma_2 - \gamma_1)e^{(\gamma_2-\gamma_1)l}}{e^{(\gamma_2-\gamma_1)d} - 1} \quad ,
\end{aligned} \tag{3.26}$$

so that

$$\phi_{\gamma_1,\gamma_2}(\mathbf{l}|\mathbf{d}) = \frac{(\gamma_2 - \gamma_1)^n e^{(\gamma_2-\gamma_1)\sum_{i=1}^n l_i}}{\prod_{i=1}^n (e^{(\gamma_2-\gamma_1)d_i} - 1)} \quad . \tag{3.27}$$

The expectation step gives:

$$\begin{aligned}
Q_{\mathbf{d}}(\theta_1,\theta_2|\gamma_1,\gamma_2) =& \sum_{i=1}^{n} \int_0^{d_i} \log\left(\theta_1\theta_2 e^{-\theta_2 d_i} e^{(\theta_2-\theta_1)l_i}\right) \frac{(\gamma_2-\gamma_1)e^{(\gamma_2-\gamma_1)l_i}}{e^{(\gamma_2-\gamma_1)d_i} - 1} dl_i \\
=& \sum_{i=1}^{n} \log(\theta_1) + \log(\theta_2) - \theta_2 d_i - \frac{\theta_2 - \theta_1}{\gamma_2 - \gamma_1} + \frac{(\theta_2-\theta_1)d_i e^{(\gamma_2-\gamma_1)d_i}}{e^{(\gamma_2-\gamma_1)d_i} - 1} \quad ,
\end{aligned} \tag{3.28}$$

so that

$$\frac{\partial Q_{\mathbf{d}}(\theta_1,\theta_2|\gamma_1,\gamma_2)}{\partial\theta_1} = \frac{n}{\theta_1} + \frac{n}{\gamma_2 - \gamma_1} - \sum_{i=1}^{n} \frac{d_i e^{(\gamma_2-\gamma_1)d_i}}{e^{(\gamma_2-\gamma_1)d_i} - 1} \quad ,$$

and

$$\frac{\partial Q_{\mathbf{d}}(\theta_1,\theta_2|\gamma_1,\gamma_2)}{\partial\theta_2} = \frac{n}{\theta_2} - \frac{n}{\gamma_2 - \gamma_1} + \sum_{i=1}^{n} \frac{d_i}{e^{(\gamma_2-\gamma_1)d_i} - 1} \quad .$$

The announced result then follows from the maximization step. □

Three important remarks are in order:

- For all $k$, $\frac{1}{\gamma_1^{(k+1)}} > 0$ and $\frac{1}{\gamma_2^{(k+1)}} > 0$. This follows from the fact that

$$e^{(\gamma_2^{(k)} - \gamma_1^{(k)})d_i} - 1 < (\gamma_2^{(k)} - \gamma_1^{(k)}) d_i e^{(\gamma_2^{(k)} - \gamma_1^{(k)})d_i} \quad .$$

Therefore

$$\frac{d_i e^{(\gamma_2^{(k)} - \gamma_1^{(k)})d_i}}{e^{(\gamma_2^{(k)} - \gamma_1^{(k)})d_i} - 1} > \frac{1}{\gamma_2^{(k)} - \gamma_1^{(k)}}$$

and (3.24) shows that $\frac{1}{\gamma_1^{(k+1)}} > 0$. Similarly,

$$e^{(\gamma_2^{(k)}-\gamma_1^{(k)})d_i} - 1 > (\gamma_2^{(k)} - \gamma_1^{(k)})d_i \quad .$$

Therefore

$$\frac{d_i}{e^{(\gamma_2^{(k)}-\gamma_1^{(k)})d_i} - 1} < \frac{1}{\gamma_2^{(k)} - \gamma_1^{(k)}}$$

and (3.25) implies $\frac{1}{\gamma_2^{(k+1)}} > 0$.

- For all $k \geq 0$,

$$\frac{1}{\gamma_1^{(k+1)}} + \frac{1}{\gamma_2^{(k+1)}} = \frac{1}{n}\sum_{i=1}^{n} d_i \quad . \tag{3.29}$$

  This is immediate when adding up (3.24) and (3.25).

- It can be shown that the limit of (3.24) and (3.25) when $\gamma_2^{(k)} - \gamma_1^{(k)} \to 0$ is $\frac{\sum_{i=1}^n d_i}{2n}$ for both equations. Hence, the case $\gamma_1 = \gamma_2$ is not a problem with the E-M algorithm.

Here is now the main result on the E-M algorithm in this case. From Theorem 1.4.16 and Corollary 1.4.17, we know that the sequence $\log f_{\mathbf{d}}(\gamma_1^{(k)}, \gamma_2^{(k)})$ (and hence also the sequence $f_{\mathbf{d}}(\gamma_1^{(k)}, \gamma_2^{(k)})$) is increasing and converges to a finite limit.

The fact that the sequence $f_{\mathbf{d}}(\gamma_1^{(k)}, \gamma_2^{(k)})$ converges does not prove yet that $(\gamma_1^{(k)}, \gamma_2^{(k)})$ converges, and even if it does so, it could converge to some value which is not a solution of the likelihood equation.

However, for this particular case:

**Lemma 3.5.3.** *The sequence $(\gamma_1^{(k)}, \gamma_2^{(k)})$ converges to a finite limit $(\gamma_1^*, \gamma_2^*)$ which is a solution of the likelihood equation.*

*Proof.* From theorem 1.4.19, the second part is obvious once the convergence has been shown. The proof of the convergence appears in Appendix 3.8.1. Note that we provide an original proof based on a continuity argument, because the natural sufficient conditions from 1.4.20 for the convergence of $(\gamma_1^{(k)}, \gamma_2^{(k)})$ do not hold here. □

As a direct corollary of these lemmas, if the likelihood equation has a unique solution which is a maximum, then this is the maximal likelihood estimator and the E-M algorithm converges to it, which itself converges to the ground truth as $n$ increases (Theorem 3.5.1).

**More than two stations**

Denote by $l_{j,i}$ the time spent by probe $i$ on link $j$. If there is only one probe, we just write $l_j$ for the time it spends on link $j$. Hence

$$\begin{aligned}
\phi_{\gamma_1,\ldots,\gamma_K}(l_1,\ldots,l_{K-1}|d) =& \frac{\phi_{\gamma_1,\ldots,\gamma_K}(l_1,\ldots,l_{K-1},d)}{\phi_{\gamma_1,\ldots,\gamma_K}(d)} \\
=& \frac{\gamma_1\cdots\gamma_K e^{-\gamma_1 l_1}\cdots e^{-\gamma_{K-1}l_{K-1}}e^{-\gamma_K(d-l_1-\cdots-l_{K-1})}}{\gamma_1\cdots\gamma_K\sum_{j=1}^K\left(\prod_{k\neq j}\frac{1}{\gamma_k-\gamma_j}\right)e^{-\gamma_j d}} \\
=& \frac{e^{-\gamma_K d}\prod_{j=1}^{K-1}e^{(\gamma_K-\gamma_j)l_j}}{\sum_{j=1}^K\left(\prod_{k\neq j}\frac{1}{\gamma_k-\gamma_j}\right)e^{-\gamma_j d}} \\
\phi_{\gamma_1,\ldots,\gamma_K}(l_1,\ldots,l_{K-1}|d) =& \frac{\prod_{j=1}^{K-1}e^{(\gamma_K-\gamma_j)l_j}}{\sum_{j=1}^K\left(\prod_{k\neq j}\frac{1}{\gamma_k-\gamma_j}\right)e^{(\gamma_K-\gamma_j)d}} \quad . \qquad (3.30)
\end{aligned}$$

Then, for a sample of $n$ independent probe delays, we have (with the same notation as above):

$$Q_{\mathbf{d}}(\theta_1,\ldots,\theta_K|\gamma_1,\ldots,\gamma_K) = \sum_{i=1}^n \mathbb{E}\left[\log\left(\widetilde{f}_{(l_{(1,i)},\ldots,l_{(K-1,i)},d_i)}(\theta_1,\ldots,\theta_K)\right)\right] \quad ,$$

where the expectation bears on the variables $l_{(1,i)},\ldots,l_{(K-1,i)}$ and is with respect to the conditional density

$$\phi_{\gamma_1,\ldots,\gamma_K}(l_{(1,i)},\ldots,l_{(K-1,i)}|d_i) \quad .$$

This leads to the following integral expression

$$\begin{aligned}
&Q_{\mathbf{d}}(\theta_1,\ldots,\theta_K|\gamma_1,\ldots,\gamma_K) \\
&\quad= \sum_{i=1}^n \int_{l_{(1,i)}=0}^{d_i}\cdots\int_{l_{(K-1,i)}=0}^{d_i-\sum_{j=1}^{K-2}l_{(j,i)}} \frac{\prod_{j=1}^{K-1}e^{(\gamma_K-\gamma_j)l_{(j,i)}}}{\sum_{j=1}^K\left(\prod_{k\neq j}\frac{1}{\gamma_k-\gamma_j}\right)e^{(\gamma_K-\gamma_j)d_i}} \\
&\qquad \log\left(\theta_1\cdots\theta_K e^{-\theta_K d_i}\prod_{j=1}^{K-1}e^{(\theta_K-\theta_j)l_{(j,i)}}\right)dl_{(1,i)}\cdots dl_{(K-1,i)} \\
&\quad= \sum_{i=1}^n \beta_i(d_i)\int_{l_{(1,i)}=0}^{d_i}\cdots\int_{l_{(K-1,i)}=0}^{d_i-\sum_{j=1}^{K-2}l_{(j,i)}}\prod_{j=1}^{K-1}e^{(\gamma_K-\gamma_j)l_{(j,i)}} \\
&\qquad \left(\log(\theta_K)-\theta_K d_i+\sum_{j=1}^{K-1}\left(\log(\theta_j)+(\theta_K-\theta_j)l_{(j,i)}\right)\right)dl_{(1,i)}\cdots dl_{(K-1,i)} \quad ,
\end{aligned}$$

with

$$\beta_i(d_i) = \frac{1}{\sum_{j=1}^K\left(\prod_{k\neq j}\frac{1}{\gamma_k-\gamma_j}\right)e^{(\gamma_K-\gamma_j)d_i}} \quad . \qquad (3.31)$$

These integrals show that $Q_{\mathbf{d}}(\theta_1,\ldots,\theta_K|\gamma_1,\ldots,\gamma_K)$ is an affine function of the variables $\theta_j$ and $\log\theta_j\,\forall 1\le j\le K$. This means that taking its partial derivative with regards of any $\theta_j$ and setting it to zero will give a simple equation of the form $\frac{a}{\theta_j}+b=0$ to solve, which will provide the solution of the maximization step in closed form. Let us illustrate this by:

**Lemma 3.5.4.** *For the three router case, for all $k\ge 0$, (3.23) is equivalent to*

$$\frac{1}{\gamma_1^{(k+1)}} = -\frac{1}{\gamma_2^{(k)}-\gamma_1^{(k)}} + \left(\frac{\gamma_3^{(k)}-\gamma_2^{(k)}}{\gamma_3^{(k)}-\gamma_1^{(k)}}\times \frac{1}{n}\sum_{i=1}^{n}\frac{(\gamma_3^{(k)}-\gamma_1^{(k)})d_i e^{-\gamma_1^{(k)}d_i}+e^{-\gamma_3^{(k)}d_i}-e^{-\gamma_1^{(k)}d_i}}{(\gamma_3^{(k)}-\gamma_2^{(k)})e^{-\gamma_1^{(k)}d_i}-(\gamma_3^{(k)}-\gamma_1^{(k)})e^{-\gamma_2^{(k)}d_i}+(\gamma_2^{(k)}-\gamma_1^{(k)})e^{-\gamma_3^{(k)}d_i}}\right) \tag{3.32}$$

$$\frac{1}{\gamma_2^{(k+1)}} = \frac{1}{\gamma_2^{(k)}-\gamma_1^{(k)}} - \left(\frac{\gamma_3^{(k)}-\gamma_1^{(k)}}{\gamma_3^{(k)}-\gamma_2^{(k)}}\times \frac{1}{n}\sum_{i=1}^{n}\frac{(\gamma_3^{(k)}-\gamma_2^{(k)})d_i e^{-\gamma_2^{(k)}d_i}+e^{-\gamma_3^{(k)}d_i}-e^{-\gamma_2^{(k)}d_i}}{(\gamma_3^{(k)}-\gamma_2^{(k)})e^{-\gamma_1^{(k)}d_i}-(\gamma_3^{(k)}-\gamma_1^{(k)})e^{-\gamma_2^{(k)}d_i}+(\gamma_2^{(k)}-\gamma_1^{(k)})e^{-\gamma_3^{(k)}d_i}}\right) \tag{3.33}$$

*and*

$$\frac{1}{\gamma_3^{(k+1)}} = \frac{1}{n}\sum_{i=1}^{n}d_i - \frac{1}{\gamma_2^{(k+1)}} - \frac{1}{\gamma_1^{(k+1)}} \quad . \tag{3.34}$$

*Proof.* The proof (composed mostly of direct computation) can be found in Appendix 3.8.2. . □

*Remark.* In chapter 4, we will prove an explicit formula for general tree topologies. Applied to the particular case of $K$ servers in a line, we get that the E-M recursion is

$$\forall k\ge 0,\forall 1\le j\le K,\quad \frac{1}{\gamma_j^{(k+1)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\xi_{\boldsymbol{\gamma}^{(k)}}(l_j|d_i)}{\phi_{\boldsymbol{\gamma}^{(k)}}(d_i)} \quad ,$$

where

$$\phi_{\boldsymbol{\gamma}}(d) = \Gamma\times\sum_{j=1}^{K}\frac{e^{-\gamma_j d}}{\prod_{k\ne j}(\gamma_k-\gamma_j)}$$

is the density of the delay distribution at $d\ge 0$ given parameters $\boldsymbol{\gamma}=(\gamma_1,\ldots,\gamma_K)$, and

$$\xi_{\boldsymbol{\gamma}}(l_j|d) = \Gamma\times\left(\frac{e^{-\gamma_j d}}{\prod_{k\ne j}(\gamma_k-\gamma_j)}\left(d-\sum_{k\ne j}\frac{1}{\gamma_k-\gamma_j}\right)+\sum_{i\ne j}\frac{e^{-\gamma_i d}}{(\gamma_j-\gamma_i)^2\prod_{\substack{k\ne j\\ k\ne i}}(\gamma_k-\gamma_i)}\right)$$

is such that the fraction represents the conditional expectation of the sojourn time in server $j$, given that the total end-to-end delay $d>0$ and the parameters $\boldsymbol{\gamma}$, with $\Gamma=\prod_{i=1}^{K}\gamma_i$ being a multiplicative constant for both sides of the fraction.

Table 3.4 provides simulation results for the 3 station case.

| Ground truth | Mean | 10% percentile | 90% percentile | Variance |
|---|---|---|---|---|
| (1, 10, 100) | (1, 9.99, 101) | (0.98, 9.11, 78.9 ) | (1.02, 10.9, 129) | (1.5 $10^{-4}$, 0.43, 320) |
| (1, 10, 20) | (1, 9.83, 22.2) | (0.99 , 7.93, 15.7) | (1.02, 11.9, 30.6) | (1.5 $10^{-4}$, 2.3, 35) |
| (1, 10, 11) | (1, 8.35, 14.4) | (0.99, 6.83, 11.02) | (1.02, 9.87, 18.5) | (1.5 $10^{-4}$, 1.35, 9.43) |
| (1, 100, 110) | (1, 68.7, 188) | (0.99, 59.4, 165) | (1.01, 77.7, 213) | (1.1 $10^{-4}$, 52.5, 418) |
| (1, 2, 100) | (1, 2.01, 91.4) | (0.97, 1.88, 72.1) | (1.04, 2.15, 111) | (7 $10^{-4}$, 0.01, 223) |
| (1, 1.2, 100) | (1, 1.2, 89,7) | (0.93, 1.1, 72.2) | (1.08, 1.32, 107) | (3.3 $10^{-3}$, 6.8 $10^{-3}$, 212) |
| (1, 1.2, 10) | (1.07, 1.09, 13) | (1.05, 1.07, 9.85) | (1.08, 1.1, 17.1) | (1.4 $10^{-4}$, 1.7 $10^{-4}$, 8.19) |
| (1, 1.2, 1.4) | (1.04, 1.105, 1.48) | (1 , 1.04, 1.36) | (1.1, 1.2, 1.67) | (1.2 $10^{-3}$, 3 $10^{-3}$, 0.015) |

Table 3.4: Precision of estimator $(\widehat{\gamma}_1, \widehat{\gamma}_2, \widehat{\gamma}_3)$ for various ground truths. Experiments have $10^4$ probes and are repeated 200 times.

### 3.5.4 Additive measurement noise

We consider now the case with additive noise in measurements. We come back to the single station case but we now assume that all delays have some measurement noise which consists in adding an independent random variable which is uniform in $[-b, a]$. The density of the noised delay $D$ is then

$$\phi_\gamma(d) = \begin{cases} \int_{-b}^{d} \frac{1}{a+b}\gamma' e^{-\gamma'(d-x)}dx = \frac{1-e^{-\gamma'(d+b)}}{a+b} & \text{if } -b \leq d < a \\ \int_{-b}^{a} \frac{1}{a+b}\gamma' e^{-\gamma'(d-x)}dx = \frac{e^{-\gamma' d}(e^{\gamma' a}-e^{-\gamma' b})}{a+b} & \text{if } d \geq a \end{cases} .$$

The likelihood to measure $n$ delays $d_1 \leq d_2 \leq \ldots \leq d_{k-1} < a \leq d_k \leq \ldots \leq d_n$ is:

$$f_{\mathbf{d}}(\gamma) = \frac{1}{(a+b)^n} \prod_{i=1}^{k-1} (1 - e^{-\gamma'(d_i+b)}) e^{-\gamma' \sum_{i=k}^{n} d_i} (e^{\gamma' a} - e^{-\gamma' b})^{n-k+1} .$$

Direct calculations give that

$$\frac{\partial \log f_{\mathbf{d}}(\gamma)}{\partial \gamma} = -\sum_{i=1}^{n} d_i - nb + \sum_{i=1}^{k-1} \frac{d_i + b}{1 - e^{-\gamma'(d_i+b)}} + (n-k+1)\frac{a+b}{1-e^{-\gamma'(a+b)}} .$$

Hence, the maximum likelihood estimator $\widehat{\gamma}$, which verifies the relation

$$\frac{\partial \log f_{\mathbf{d}}(\gamma)}{\partial \gamma}(\widehat{\gamma}) = 0 ,$$

is such that

$$\sum_{i=1}^{k-1} \frac{d_i + b}{1 - e^{-\widehat{\gamma}'(d_i+b)}} + (n-k+1)\frac{a+b}{1-e^{-\widehat{\gamma}'(a+b)}} - nb = \sum_{i=1}^{n} d_i . \quad (3.35)$$

The function

$$\widehat{\gamma}' \to \sum_{i=1}^{k-1} \frac{d_i + b}{1 - e^{-\widehat{\gamma}'(d_i+b)}} + (n-k+1)\frac{a+b}{1-e^{-\widehat{\gamma}'(a+b)}} - nb$$

is decreasing. There is therefore only one solution to (3.35), which can easily be found using numerical techniques. It is easy to check that

$$\begin{aligned}
&\mathbf{E}_{\gamma'}\left[\frac{D+b}{1-e^{-\gamma' D+b}}\mathbb{1}_{D<a}\right]+\frac{a+b}{1-e^{-\gamma'(a+b)}}\mathbb{P}_{\gamma'}(D\geq a)-b \\
&=\int_{-b}^{a}\frac{t+b}{1-e^{-\gamma'(t+b)}}\frac{1-e^{-\gamma'(t+b)}}{a+b}dt+\frac{a+b}{1-e^{-\gamma'(a+b)}}\int_{a}^{\infty}\frac{e^{-\gamma' t}(e^{\gamma' a}-e^{-\gamma' b})}{a+b}dt-b \\
&=\int_{-b}^{a}\frac{t+b}{a+b}dt+e^{-\gamma' a}\int_{a}^{\infty}e^{-\gamma' t}dt-b \\
&=\frac{a^2-b^2}{2(a+b)}+\frac{1}{\gamma'} \\
&=\frac{a-b}{2}+\frac{1}{\gamma'} \\
&=\mathbf{E}_{\gamma'}[D] \quad . \qquad (3.36)
\end{aligned}$$

Hence (3.36) is equivalent to (3.35) when the number of probes $n$ tends to infinity. This shows the asymptotic consistency of MLE estimator for one station and uniform noise.

In practice, timestamps are measured at the departure and the arrival of packets. Assuming that timestamps suffer from a uniformly distributed noise, the measured delay is the real delay plus two independent uniform noise variables. The design of maximum likelihood techniques for such noise structures and working for several station in series is an interesting open question.

## 3.6 Experimental Validation

We test our tomography method using an experimental methodology involving simulations driven by traces collected on a core Internet router. Although such an approach has limitations, it enables an examination of performance in a context where important real world issues can be observed, evaluated and understood.

### 3.6.1 Data Sets and Traces

The traces we use were collected at a gateway router of the Sprint IP backbone network. The router had 4 linecards supporting 6 active interfaces: **1**: OC-48 (BB1); **2**: OC-48 (BB2); **3**: OC-12 (C4); and **4**: OC-3 (C1, C2, C3) as shown in Figure 3.4. The interfaces BB1 and BB2 connect to backbone routers and carry the bulk of the traffic, while the others connect customer links. Traffic on 11 links over the 6 interfaces was monitored, accounting for more than $99.95\%$ of all through-router traffic. DAG cards [dag], synchronized to a common GPS signal, were used to capture a fixed length $64$ byte record for each packet, and record a timestamp accurate to $2.2\mu$s on OC-3 links and below $1\mu$s on others.

We use two 'full-router' datasets, Exp1 and Exp2, each collected with the experiment over 24 hours, some 4 terabytes of data each. The first was taken in August 2003 and has

| Trace | Exp#: Input – Output | # Packets | Rate Mbps | $\mathbf{E}[S]$ bytes | Cov $[S]$ | Cov $[\tau]$ |
|---|---|---|---|---|---|---|
| P1-BB1 | Exp1: BB1-in – C2-out | 2647128 | 47.1 | 658.0 | 0.90 | 1.16 |
| P1-BB2 | Exp1: BB2-in – C2-out | 3221776 | 60.1 | 689.5 | 0.87 | 1.16 |
| P2-BB2 | Exp1: BB2-in – C2-out | 2899816 | 53.8 | 686.7 | 0.89 | 1.15 |
| P3-BB1 | Exp1: BB1-in – C2-out | 2130118 | 35.2 | 608.5 | 0.94 | 1.16 |
| Q1-BB1 | Exp2: BB1-in – C2-out | 1557533 | 24.1 | 571.1 | 1.11 | 1.14 |
| Q1-BB2 | Exp2: BB2-in – C2-out | 1957099 | 36.0 | 680.6 | 0.96 | 1.15 |
| Q4-BB1 | Exp2: BB1-in – C2-out | 1583546 | 23.6 | 549.9 | 1.15 | 1.14 |
| Q4-BB2 | Exp2: BB2-in – C2-out | 1864193 | 31.2 | 617.6 | 1.04 | 1.14 |

Table 3.5: Traces used to feed simulation, each 300 seconds long. The coefficients of variation of packet size $\text{Cov}[S] = \sqrt{\text{Var}[S]}/\mathbf{E}[S]$ and inter-arrival time $\tau$ should be compared with 1 (exponential case).

been used in several works including [KSC+02, HVPD04, MVBB07, BMVB07], and the second, from January 2004, was used in [BMVB07].

A thorough description of the experimental setup including the issues involved in the processing of the raw DAG timestamps into valid through-router delays, and the careful management of header overhead effects, which we follow here, can be found in [HVPD04] (in particular Section 3). Two points are relevant here concerning serialization times, which equate to service times and therefore waiting times, of packets at the input and output of the router. First, with SONET headers removed (the linecards use Packet over SONET (PoS)), the raw (OC-3, OC-12, OC-48) bandwidths are effectively reduced from (155, 620, 2,480) Mbps to (149.76, 599.04, 2,396.16) Mbps. Second, the DAG records IP packet sizes but they are transmitted with a 9 byte HDLC transport layer encapsulation (5 leading, 4 trailing). These modified capacities and packet sizes are used below.

The traces exhibit marked diurnal variation whereas we require stationary conditions. We follow [BMVB07] in selecting from both experiments a number of time windows which

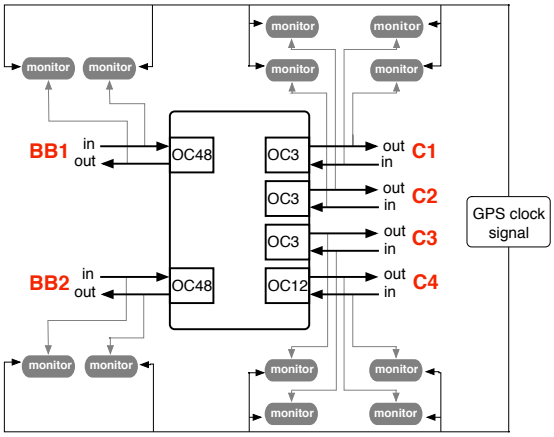



Figure 3.4: Full-router experimental setup.

give variety in average link load, but within which stationarity approximately holds. Table 3.5 describes the incoming subtraces. The traces used are from different links and/or well separated in time and so are close to independent.

### 3.6.2 Semi-Experimental Methodology

Traces from the full-router experiments are static, a fixed web of packet delays spanning all input and output interfaces. The only way to vary parameters in such a context is to search through the trace hoping for interesting variations, which is not very flexible, and even more seriously, probes cannot be added. One could try to 'baptize' selected packets as probes, however such probes would observe Palm (self-conditional) probabilities rather than the desired equilibrium probabilities, which introduces an inherent (strong) bias. Finally, with the traces alone we would be restricted to single hop routes.

To gain the flexibility we need, we use trace driven simulation whereby selected traces from incoming links are fed to a queueing system representing the router, to which we can also add probes, and direct the output to subsequent hops fed by additional traces. It is well known that a drawback of this approach is that in practice feedback mechanisms (in particular TCP) would alter the traffic flows as a function of the experimental parameters. Nonetheless, it enables us to study the effect of breaking the technical assumptions of Section 3.5. A live experiment involving passive capture in the Internet core combined with simultaneous active probing has been attempted before [MVBB07], but is very challenging to put in place and could not be performed here.

A simulator depends crucially on the choice of system model. As presented in section 1.1.4, we follow [HVPD04] which investigated this issue in detail for Exp1 and is therefore relevant for this router. Two models were described, the first of which was shown to predict through-router delays very well, and the second extremely well (to within a few $\mu$s for almost all packets):

**One Stage Model:** A FIFO queue with service time given by $S/C$, where $S$ is the packet size in bits (including the 9 HDLC bytes) and $C$ the capacity (overhead-corrected as above) in bps.

**Two Stage Model:** Packets must remain in a 'front end' FIFO system for at least a time $\Delta(S) = a + bS$ prior to entering the output FIFO queue (a more precise description is given below). This models the time for a packet to cross the switch fabric and enter the output buffer. The values of $a$, $b$, depend on the output interface type and capacity. For OC-3 output here for example, we use $a = 18.8\mu$s and $b = 1.8125$ [ns/bit] from [HVPD04].

As noted in Section 3.2.2, there are several strong technical assumptions underlying our MLE based inversion for available bandwidths along a path. The flexibility of simulation can also be employed to explore the impact of these separately. We use the methodology known as the 'semi-experimental method', from [HVA03], which was used to investigate the underlying causes of key statistical traffic features, and thereby to select meaningful traffic models. Here we use it to determine which of the technical assumptions is most cru-

cial/least valid, and to test corrections which we derive below. Specifically, in the following sections we systematically explore the impact of the errors in assuming the

(i) One stage router model;

(ii) Exponential nature of packet size;

(iii) Equality of probe and cross traffic distributions;

(iv) Poisson nature of packet arrivals;

(v) Independence of packet size over multiple stations.

Typically results will be given using 1200 probes for 1 station, and 120000 for 2. In Section 3.6.7 the case of two node will be treated.

### 3.6.3 Challenge: Router Model

A naive model, almost universally employed in the active probing literature, assumes that the delays experienced by packet streams destined for a particular output link obeys that of a single server FIFO queue. The single stage model above from [HVPD04] justified this using Exp1 data. Here (for the first time) we use Exp2 to test the same models. Using the Q1 time period we study all input streams (in fact only Q1-BB1 and Q1-BB2) headed for C2-out. Figure 3.5 shows the true versus surrogate delays of BB1-in packets using both models (histograms for BB2-in packets are similar). The one stage model histogram has a very similar shape to the true one, but is visibly offset from it by 50$\mu$s or so. This error could play havoc with inversion methods under light loads.

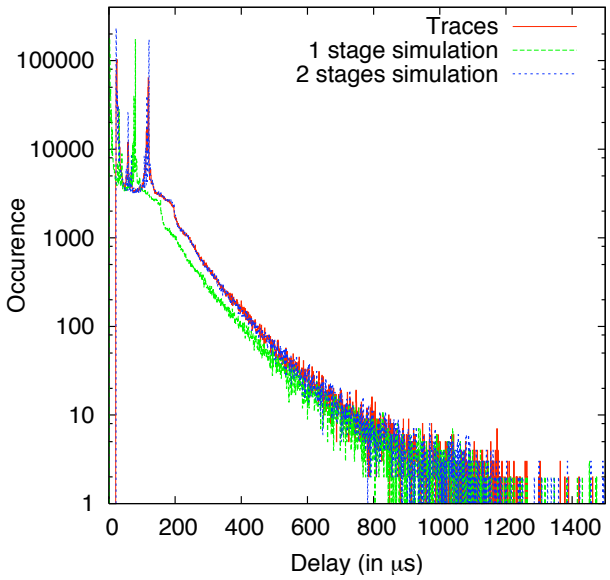



Figure 3.5: Comparison of true and model delays: histograms for BB1-in packets for a single station fed with Q1-BB1 and Q1-BB2 inputs (other inputs negligible), against true delays. The two stage and true distributions overlap almost perfectly.

**Response part 1: Two stage model correction**

The correction in this case is given by the time spent in the first stage of the two stage model

given above. For the $n$-th packet of size $S_n$, this is given by $\xi_n = \max(\xi_{n-1} - \tau_n, \Delta_n)$ where $\tau_n$ is the inter-arrival time (in other words, when sentenced to $\Delta_n$ seconds of hard waiting, time in court is counted).

The impact of the correction is seen in Figure 3.5 where the agreement with the true delay is extremely close.

In practice, the relevant router delay functions $\Delta(S)$ must be identified and known for each hop, however it is feasible for these to be tabulated and made available, at least for network operators.

**Response part 2: Estimation correction for two stage model**

With an appropriate model chosen, the next task is to account for the impact on the estimation algorithm itself. To test this, we follow the semi-experimental methodology and replace almost all aspects of the real experiment by surrogates. Thus in this case, arrivals are made to be Poisson, and service times exponential (with the same mean) for both cross traffic and probes. The router is replaced by a two stage model which represents a simplified but accurate ground truth. As seen in Table 3.6, the errors in

| Input Traces | Output | $\gamma$ | $\widehat{\gamma}$ | $\widehat{\gamma}$ corrected |
|---|---|---|---|---|
| P1-BB1 + P1-BB2 | OC-3 | 42.5 | 34.6 | 42.0 |
| Q1-BB1 + Q1-BB2 | OC-3 | 89.5 | 55.0 | 77.7 |

Table 3.6: Impact of approximate two-stage correction.

a Kelly based E-M estimate are large, however a simple correction based on replacing $\widehat{\gamma} = \mathbb{E}[S]/\mathbb{E}[D]$ by $\widehat{\gamma} = \mathbb{E}[S]/(\mathbb{E}[D] - \mathbb{E}[\Delta(S)])$, that is multiplying by a correction factor $F_M = \mathbb{E}[D]/(\mathbb{E}[D] - \mathbb{E}[\Delta(S)])$, largely succeeds in correcting it.

### 3.6.4 Challenge: Exponential Sizes

Following the semi-experimental methodology, we investigate the impact of service times (packet sizes) which are not exponentially distributed by nulling all other effects: we use the one stage model, Poisson arrivals, and as before we inject a Poisson probe stream, also with exponential packet sizes with a matched mean. The only non-ideal components left are the non-exponential service times of the cross traffic packet streams. Two examples are given in Table 3.7, each of which exhibit small errors. Although this is good news

| Input Traces | Output | $\gamma$ | $\widehat{\gamma}$ | $\widehat{\gamma}$ corrected |
|---|---|---|---|---|
| P1-BB1, P1-BB2 | OC-3 | 42.5 | 42.1 | 38.7 |
| Q1-BB1, Q1-BB2 | OC-3 | 89.5 | 85.7 | 86.5 |

Table 3.7: Impact of Exponential Size assumption, (1200 probes).

for our E-M estimator, it is not the strongest test since the packet sizes, although highly non-exponential, have a coefficient of variation close to 1 (recall Table 3.5). We according significantly modify the true packet size distribution of the P1 trace in two ways (keeping the

mean constant at 675 bytes): (i) constant packet sizes (top line in Table 3.8), (ii) bimodal distribution on $(40, 3000)$ bytes with probability $(0.785, 0.215)$ (bottom line). The E-M estimates are now significantly different.

| Input Traces | Output | $\kappa$ | $\gamma$ | $\widehat{\gamma}$ | $\widehat{\gamma}$ corrected |
|---|---|---|---|---|---|
| P1-BB1, P1-BB2 | OC-3 | $-0.5$ | 42.5 | 70.3 | 44.9 |
| P1-BB1, P1-BB2 | OC-3 | 0.215 | 42.5 | 25.5 | 45.9 |

Table 3.8: Experiment with modification packet sizes.

**Response: Variance correction factor**

We can relax the exponential packet size assumption in our M/M/1 model by considering the M/G/1 queue, where the service times are i.i.d. with general distribution. Let the arrival and service rates be given by $\lambda$ and $\mu$ packets per second respectively. The load factor is $\rho = \lambda/\mu$. If $S$ denotes a random packet size and $C$ the server capacity (in bits per second), then $C = \mu\mathbb{E}[S]$, the service time for any packet is $\sigma = S/C$, and the average service time is $\mathbb{E}[\sigma] = \mathbb{E}[S]/C = 1/\mu$.

According to the Pollaczek–Khinchin formula, the expected value of the system time $D$ (just the end-to-end delay) for M/G/1 is:

$$\mathbb{E}[D] = \frac{\lambda\mathbb{E}\left[\sigma^2\right]}{2(1-\rho)} + \mathbb{E}[\sigma] = \frac{1+\kappa\rho}{\mu-\lambda} \quad ,$$

where we have introduced the constant $\kappa$, defined by $\mathbb{E}\left[S^2\right] = 2(1+\kappa)\mathbb{E}[S]^2$ or equivalently $\mathbb{E}\left[\sigma^2\right] = 2(1+\kappa)\mathbb{E}[\sigma]^2$, to help compare the general M/G/1 against M/M/1. If $S$ is exponentially distributed then $\kappa = 0$, and $\kappa > 0$ corresponds to greater variability than exponential, increasing delay (note that $\kappa$ is just a rewriting of the coefficient of variation of $S$: $\kappa = (\mathrm{Cov}[S]^2 - 1)/2$ ).

By definition, the true available bandwidth is $C - \mathbb{E}[S]\lambda = (\mu - \lambda)\mathbb{E}[S]$, and in practice we will estimate it using the maximum-likelihood estimator

$$\widehat{\gamma} = \frac{\mathbb{E}[S]}{\mathbb{E}[D]} = \frac{1}{1+\kappa\rho}(\mu-\lambda)\mathbb{E}[S] \quad ,$$

which differs from the desired value by a factor $F_S = 1 + \kappa\rho$. In the M/M/1 case $\kappa = 0$, so $F_S = 1$ and we recover the correct value. Otherwise, if $F_S > 1$ we obtain a value that is too small because the larger delays fool the estimator into thinking there is more offered load. To approximately compensate, we propose modifying the estimator by multiplying it by the corrective factor $F_S$.

The impact of the correction is not large in the examples of Table 3.7, as expected, since the errors were already small. On the other hand, the large errors of Table 3.8 have been successfully corrected.

In practice, $\kappa$ can be estimated by using representative values from packets collected at the receiver as it is not expected to vary much in the Internet core (see Table 3.5). To

measure $\rho$, probes of fixed size can be sent, and a measurement made of the proportion which experience the minimum delay. This is just the probability of experiencing zero waiting time, which is equal to $1 - \rho$ for a broad class of queueing systems including M/G/1.

### 3.6.5 Challenge: Equality of Distribution

In this case we again use the one stage model, make all arrivals Poisson, and all packet sizes exponential, however the mean size of probes and cross traffic packets differ. Table 3.7 shows that this one factor is enough to induce a systematic error of around 50% in an example where probes are 200 bytes compared to 675 bytes for cross traffic.

| Input Traces | Output | $\gamma$ | $\widehat{\gamma}$ | $\widehat{\gamma}$ corrected |
|---|---|---|---|---|
| P1-BB1, P1-BB2 | OC-3 | 42.5 | 17.4 | 45.8 |
| Q1-BB1, Q1-BB2 | OC-3 | 89.5 | 47.8 | 86.0 |

Table 3.9: Impact of Equality of Distribution assumption.

**Response: Mean correction factor**

Under the assumptions given above, the system is a M/G/1 queue where the service time is not distributed as an exponential but as an exponential mixture. We now derive a correction which will cover the more general case where the probes may not even be exponential, provided they are 'rare' compared to cross traffic.

We define 2 different classes of packets, 1: cross traffic, 2; probes. Each arrive according a Poisson process of intensity $\lambda_i$, $i = 1, 2$. Packets of class $i$ have a size $S_i$ with mean $A_i$, have a service time $\sigma_i = S_i/C$ with expected value $\mathbb{E}[\sigma_i] = 1/\mu_i = A_i/C$, experience a waiting time $W_i$ and a system time $D_i = W_i + \sigma_i$, and collectively contribute a load $\rho_i = \lambda_i/\mu_i$.

We assume that $S_1$ is exponentially distributed whereas $S_2$ is general. On the other hand, we assume that probes are rare, so that the total arrival rate can be expressed as $\lambda = \lambda_1 + \lambda_2 \approx \lambda_1$, from which it follows that the mean service time obeys $\mathbb{E}[\sigma] = \frac{\lambda_1}{\lambda_1+\lambda_2}\mathbb{E}[\sigma_1] + \frac{\lambda_2}{\lambda_1+\lambda_2}\mathbb{E}[\sigma_2] \approx \mathbb{E}[\sigma_1]$, and similarly $\mathbb{E}[\sigma^2] \approx \mathbb{E}[\sigma_1^2]$. Since $S_1$ is exponential, this implies that $\mathbb{E}[\sigma^2] \approx 2\mathbb{E}[\sigma]^2$. The Pollaczek–Khinchin formula for waiting time then reads

$$\mathbb{E}[W] = \frac{\lambda\mathbb{E}[\sigma^2]}{2(1-\rho)} = \frac{\rho\mathbb{E}[\sigma]}{1-\rho} \quad .$$

We introduce the constant $\kappa$ to express the difference in mean packet sizes through $\mathbb{E}[S_2] = (1+\kappa)\mathbb{E}[S]$. Since arrivals are Poisson, queueing delays are independent of packet class. In particular, $\mathbb{E}[D_2] = \mathbb{E}[W] + \mathbb{E}[\sigma_2]$, and using the rare probing approximations

$$\mathbb{E}[D_2] = \frac{\rho\mathbb{E}[\sigma]}{1-\rho} + (1+\kappa)\mathbb{E}[\sigma] = \left(\frac{1}{1-\rho} + \kappa\right)\frac{\mathbb{E}[S]}{C} \quad .$$

By definition, the true available bandwidth is $(1-\rho)C$, whereas the likelihood estimator yields

$$\widehat{\gamma} = \frac{\mathbf{E}[S_2]}{\mathbf{E}[D_2]} = \frac{1+\kappa}{1+(1-\rho)\kappa}(1-\rho)C$$

which differs by a factor $F_E = (1+(1-\rho)\kappa)/(1+\kappa)$ from the desired value. In the M/M/1 case $\kappa = 0$, so $F_E = 1$ and we recover the correct value. If probes are smaller then $F_E > 1$ and we obtain a value that is too small because the delays are larger than they would be under M/M/1 since cross traffic packets are actually larger. To approximately compensate, we propose modifying the estimator by multiplying it by the corrective factor $F_E$.

As seen in Table 3.9 using small probes, the corrected value is very close to the actual available bandwidths, to within what we would expect from statistical variability given 1200 probes.

In practice, $\kappa$ can be estimated by using representative packet size distributions, which should be very stable in the Internet core. We can measure $\rho$ as described in section 3.6.4.

### 3.6.6 Challenge: Poisson Arrivals

In this section we again use the one stage model, we replace true packet sizes with i.i.d. exponentials (with mean $\bar{S}$ matched to the average over all inputs), and inject a Poisson probe stream, also with exponential sizes of the same mean. Thus, in this semi-experiment the only non-ideal components left are the original non-Poisson arrival processes of the cross traffic packet streams. Three example impacts are given in Table 3.10, one quite large (-34%) and the others relatively small (<-10%).

| Input Traces | Output | $\gamma$ | $\widehat{\gamma}$ | $\widehat{\gamma}$ corrected |
|---|---|---|---|---|
| P1-BB1, P1-BB2 | OC-3 | 42.5 | 33.8 | 44.6 |
| Q1-BB1, Q1-BB2 | OC-3 | 89.5 | 87.8 | 100.1 |
| Q4-BB1 + Q4-BB2 | OC-3 | 95.0 | 92.3 | 98.4 |

Table 3.10: Impact of Poisson assumption, (1200 probes).

**Response: Poisson batch correction**

The correction in this case is based on the idea that packets arrive at the input in batches of back to back packets, due to the queueing at the upstream hop.

Assume that rather than a Poisson point process of packet arrivals, we have batch arrivals with the following structure: batches arrive according to a Poisson point process with intensity $\beta$ and batch contains a random number $N$ of packets, which have the same i.i.d. sizes $\sigma_i$. Assume batch sizes are independent of everything else. The workload in a single server queue fed with such a process is the same as that in a M/G/1 queue with arrival rate $\beta$ and service times $S = \sum_{i=1}^{N} \sigma_i$.

We now show that one can identify the second moment of $S$ by measuring the first and the second moments of $S_t$, the workload brought by such a point process in an interval of

length $t$. It is not difficult to see that $C_t^{(1)} = \mathbb{E}(S_t) = \beta\mathbb{E}(S)t$, whereas

$$C_t^{(2)} = \mathbb{E}(S_t^2) = \beta t\mathbb{E}(S^2) + (\beta t\mathbb{E}(S))^2 \quad .$$

Hence

$$\mathbb{E}(S^2) = \frac{C_t^{(2)}}{\beta t} - \beta t(\mathbb{E}(S))^2 = \frac{C_t^{(2)} - (C_t^{(1)})^2}{\beta t} \quad .$$

We now show that we can identify $\beta$ by measuring the first and the second moment of the packet inter-arrival time $\tau$. It is not difficult to see that $\mathbb{E}(\tau) = \frac{1}{\beta\mathbb{E}(N)}$ and $\mathbb{E}(\tau^2) = \frac{2}{\beta^2\mathbb{E}(N)}$. Hence $\beta = 2\frac{\mathbb{E}(\tau)}{\mathbb{E}(\tau^2)}$, so that

$$\mathbb{E}(S^2) = \frac{C_t^{(2)} - (C_t^{(1)})^2}{2t}\frac{\mathbb{E}(\tau^2)}{\mathbb{E}(\tau)} \quad .$$

It is also straightforward to note that $\mathbb{E}(S) = \frac{C_t^{(1)}}{\beta t}$.

Using then the same method as for the Challenge entitled 'Exponential Sizes', we get that the correction coefficient is

$$\begin{aligned} 1 + \kappa\rho =& \frac{\mathbb{E}(S^2)}{2\mathbb{E}(S)^2}\rho + 1 - \rho \\ =& \frac{t\mathbb{E}(\tau)}{\mathbb{E}(\tau^2)}\frac{C_t^{(2)} - (C_t^{(1)})^2}{(C_t^{(1)})^2}\rho + 1 - \rho \quad . \end{aligned}$$

Note that the mean batch size $\mathbb{E}(N)$ can be estimated with the formula $\mathbb{E}(N) = \frac{\mathbb{E}(\tau^2)}{2\mathbb{E}(\tau)^2}$.

As seen in Table 3.10, the correction succeeds in reducing the 10% error, but increased others. As in the previous section here the traffic is close to Poisson so the correction factors are small.

**Response: Extension to more routers**

The correction of interest here consists in transforming the delay samples in some M/G/1 queue to those that would have been experienced in an M/M/1 queue with the same arrival rate $\lambda$ and the same mean service time $\mu$. We use the ladder epoch representation for the waiting time distribution in a M/G/1 queue (see [Kle75])

$$f_W(dx) = (1-\rho)\left(\delta_0(dx) + \sum_{k\geq 1}\rho^k f_R^{(k)}(x)dx\right) \quad ,$$

with $f_R^{(k)}$ the $k$-fold convolution of the residual service time density

$$f_R(x) = \frac{1 - F_\sigma(x)}{\mathbb{E}(\sigma)} \quad ,$$

where $F_\sigma(x)$ is the CDF of the service times. The mean of the latter density is $M_R =$

$\mathbb{E}(\sigma^2)/(2\mathbb{E}(\sigma))$. Two observations are in order (i) the density is exponential of parameter $\mu$ in the case when $\sigma$ is exponential of parameter $\mu$); (ii) we find back that the mean waiting time is

$$\mathbb{E}(W) = (1-\rho)\sum_{k\geq 1}\rho^k k\frac{\mathbb{E}(\sigma^2)}{2\mathbb{E}(\sigma)} = \frac{\lambda\mathbb{E}(\sigma^2)}{2(1-\rho)}$$

which is the mean value of the Pollaczek–Khinchin formula. In order to proceed with the announced correction, we argue that if we have a sample waiting time $w$, then it is likely that the number of ladder epochs was $k(w) = \frac{w}{M_R}$ and that a natural sample for the M/M/1 queue waiting time is then

$$\widetilde{w} = w - k(w)(M_R - \mathbb{E}(\sigma)) = w - w\left(1 - \frac{\mathbb{E}(\sigma)}{M_R}\right) = w\frac{2\mathbb{E}(\sigma)^2}{\mathbb{E}(\sigma^2)} = \frac{w}{1+\kappa} \quad ,$$

where $\kappa$ is defined in section 3.6.6. If the sample one has access to is the delay $d$ of a probe of service time $\sigma$ rather than its waiting time, then the correction formula is

$$\widetilde{d} = \frac{d-\sigma}{1+\kappa} + \sigma = \frac{d+\kappa\sigma}{1+\kappa} \quad .$$

It is easy to check that this correction of the sample leads to a correction of the mean which is precisely that proposed from the Pollaczek–Khinchin mean value formula above.

For the case with several stations, when assuming that $\kappa$ is the same in all stations, we propose the correction formula

$$\widetilde{d} = \frac{d+\kappa\sum_i \sigma_i}{1+\kappa} \quad , \tag{3.37}$$

where $\sigma_i$ now denotes the service time of the probe over the $i$-th router of the path. Various heuristic extensions can be contemplated to handle the case with different $\kappa$ parameters (mean, weighted mean, etc.).

### 3.6.7 The Two Station Case

In the previous section we showed how to correct, approximately but with considerable effectiveness, deviations from each of the core assumptions in the case of a single station. We now briefly consider two aspects of the two station correction problem. In each cross traffic will be non-persistent, with exponentially distributed packets, Poisson probe streams, and OC-3 output capacities for each hop.

#### Challenge: Size Independence

In a Kelly network, service times at different stations are independent. This is not the case in real networks for any packet which traverses more than one path, as its packet size is constant. As ground truth we set probe sizes to be independent at each hop, and compare against the practical case when they are not. Three scenarios are considered: 1: input traces

(Q1-BB1, Q1-BB2) and (P1-BB1, P1-BB2) on stations 1 and 2 respectively, 2: input traces (Q1-BB1, Q1-BB2, P2-BB2) and (P1-BB1, P1-BB2, P3-BB1), 3: input traces (Q4-BB1, Q4-BB2) and (P4-BB1, P4-BB2). Table 3.11 shows that the impact can be quite large.

| | load | $(\gamma_1, \gamma_2)$ | $(\widehat{\gamma}_1, \widehat{\gamma}_2)$ | $(\widehat{\gamma}_1, \widehat{\gamma}_2)$ corr |
|---|---|---|---|---|
| 1 | (0.42, 0.70) | (45.5 , 87.5) | (33.9 , 166,4) | (42.7 , 88.2) |
| 2 | (0.77 , 0.96) | (5.4 , 33.7) | (5.5 , 34.7) | (6.4 , 35.8) |
| 3 | (0.36 , 0.38) | (92.7 , 96) | (50.6 , 748.0) | (94.4 , 94.4) |

Table 3.11: Impact of Service Time assumption.

**Response: Random Probe Split**

We emulate a probe which is of exponential size and different at each hop by sending a back-to-back probe pair, the first of which will drop out after hop 1. The key observation is that if $A$, $B$ are independent exponential random variables with parameter $\alpha$ and $\beta$ respectively, then $C = \min(A, B)$ and $D = \max(A, B) - \min(A, B)$ are also independent and exponential random variables with parameters $\alpha + \beta$ and $2\alpha\beta/(\alpha + \beta)$ respectively. If the two probes sent back to back meet no cross traffic, the aggregate probe has a service time of $\max(A, B)$ on router 1, and the surviving probe $C = \min(A, B)$ on router 2, and so the end-to-end delay of the surviving probe is $C + D = A + B$, the sum of two independent exponential random variables with parameters $\alpha$ and $\beta$ as expected! Of course, this is only a heuristic: for instance the load brought by the probe is always larger on the first station, and furthermore in practice cross traffic packets can split the pair. We argue that if probes are rare enough to not perturb the system, then this does not matter too much and that this trick allows one to emulate the appropriate behavior when it is most important, namely for probes meeting no cross traffic.

As seen in Table 3.11, the dual probe technique corrects most of the large error due to the probe size dependence. The improvement is largest at higher load (scenario 2), since there the remaining dependence in probe service times will be small compared to their waiting times.

**Challenge: Poisson Arrivals**

We test the extension of the Poisson batch correction (3.37) using true traffic arrivals and probes with sizes chosen independently at each hop. We examine: 1: input traces (Q1-BB1,Q1-BB2) and (P1-BB1,P1-BB2) on stations 1 and 2 respectively, 2: inputs (Q4-BB1,Q4-BB2) then (P4-BB1,P4-BB2), after removing the largest 1%of delays, as these outliers disrupt the batch fitting process, which is based primarily on means. Table 3.12 shows that the impact is large in the first case, and that the correction manages to correct most, but not all of it (delay histograms showed good agreement after correction) In case 2 the error, and correction, are small.

| Scenario | $(\gamma_1, \gamma_2)$ | $(\widehat{\gamma}_1, \widehat{\gamma}_2)$ | $(\widehat{\gamma}_1, \widehat{\gamma}_2)$ corrected |
|---|---|---|---|
| Q1–P1 | $(45.5, 87.5)$ | $(27.8, 147.9)$ | $(43.6, 97.3)$ |
| Q4–P4 | $(92.7, 96)$ | $(95.0, 95.0)$ | $(101.5, 101.5)$ |

Table 3.12: Impact of Service Time assumption.

## 3.7 Summary

The chapter opens two main new lines of thought. First, it showed that it is possible to pose an effective Internet tomography problem as an inverse problem in queueing theory that uses active probes as the external observation vehicle for the inversion method. A poly-phase technique based on the interpolation of the expected mean delays has been investigated, but it has been found to be numerically unstable, in addition to the practical constraint of running several phases of measurement. The second proposed inversion methodology leverages the stochastic nature of the system to be analyzed and is based on a rigorous maximization of likelihood which we showed to be tractable in high dimension thanks to the E-M algorithm. As standard approaches did not apply, our contribution includes original proofs of the asymptotic efficiency of the estimators and convergence of the E-M algorithm. This methodology could in principle be extended to other network models (e.g. Whittle and Max-Plus networks), and to other network metrics (e.g. loss rates, scheduling disciplines). Finally, one could also explore the flexibility offered by slowly varying probe intensity in order to explore the set of stationary distributions over a wider range of intensities, which could render the inversion methodology more robust. All this illustrates the approach of inverse problems as the foundation of a comprehensive network tomography methodology.

Second, the chapter investigated the effectiveness of this tomographic method on what is a difficult problem, the estimation of the residual bandwidth on all links on a path, not only the path bottleneck. Queuing theory together with traces from a core Internet router were used in order to correct the errors associated with the use of the tractable parametric model which is needed for the inversion step. It was shown that combined with the knowledge of a few basic statistical properties of Internet traffic, the dominant corrections can in principle be identified and processed so as to lead to an effective estimation scheme, which works better in case of higher load. The design of a systematic construction allowing one to build estimators combining all individual corrections would would work ‘out of the box’ under all load conditions is a significant challenge for which we have laid several promising first steps. On this more practical side, it would also be interesting to take the timestamping ‘noise’ into account in the estimation methodology. This is particularly important for high speed links where such errors play a dominant role.

## 3.8 Appendix

### 3.8.1 Proof of Lemma 3.5.3

The proof relies on the following lemmas, which will be proved at the end of this section. Lemma 3.8.1 is technical and correspond to a classical property of limit points of a sequence. It will be used only to prove Lemma 3.8.2, which will be the basic block of the result.

**Lemma 3.8.1.** *Let $(x_n)_{n\in\mathbb{N}}$ be a sequence with values in $\mathbb{R}$, s.t. $(x_{n+1}-x_n)$ converges to zero. Assume that $a$ and $b$ are both limit points of $(x_n)$. Then every point $c$ in $[a,b]$ is also a limit point of $(x_n)$.*

**Lemma 3.8.2.** *Let $(x_n)$ be a sequence with values in $\mathbb{R}$ and $f$ a continuous function from $\mathbb{R}$ to $\mathbb{R}$. Assume that the sequence $(f(x_n))_{n\in\mathbb{N}}$ is convergent, and that $(x_n)$ is bounded. Assume further that the following relation holds:*

$$f(x_{k+1}) - f(x_k) \geq g(x_{k+1} - x_k) \quad , \tag{3.38}$$

*where $g(.)$ is a positive continuous function, null at and only at zero. Then the sequence $(x_n)$ is also convergent.*

First, using (3.29), let express $\gamma_2^{(k)}$ as a function of $h(.)$ of $\gamma_1^{(k)}$), where

$$h(x) = \left( \frac{\sum_{i=1}^n d_i}{n} - \frac{1}{x} \right)^{-1} .$$

Let us evaluate

$$\Delta_k = Q_{\mathbf{d}}(\gamma_1^{(k+1)}, \gamma_2^{(k+1)} | \gamma_1^{(k)}, \gamma_2^{(k)}) - Q_{\mathbf{d}}(\gamma_1^{(k)}, \gamma_2^{(k)} | \gamma_1^{(k)}, \gamma_2^{(k)}).$$

Using (3.28) we get that

$$\begin{aligned}
\Delta_k \quad = \quad & n \log \gamma_1^{(k+1)} + n \log \gamma_2^{(k+1)} - n \log \gamma_1^{(k)} - n \log \gamma_2^{(k)} \\
& -\gamma_1^{(k+1)} \sum d_i + \gamma_1^{(k)} \sum d_i \\
& -n \frac{\gamma_2^{(k+1)} - \gamma_1^{(k+1)}}{\gamma_2^{(k)} - \gamma_1^{(k)}} + n \frac{\gamma_2^{(k)} - \gamma_1^{(k)}}{\gamma_2^{(k)} - \gamma_1^{(k)}} \\
& +((\gamma_2^{(k+1)} - \gamma_1^{(k+1)}) - (\gamma_2^{(k)} - \gamma_1^{(k)})) \sum_i \frac{d_i}{e^{(\gamma_2^{(k)} - \gamma_1^{(k)}) d_i} - 1} \quad .
\end{aligned}$$

Using optimality (Eq. (3.25)) to reexpress the sum $\sum_i \frac{d_i}{e^{(\gamma_2^{(k)} - \gamma_1^{(k)}) d_i} - 1}$ in terms of $\gamma_1^{(k)}, \gamma_2^{(k)}$ and $\gamma_2^{(k+1)}$ and using (3.29) to reexpress the sum $\sum_i d_i$ in terms of $\gamma_1^{(k+1)}$ and $\gamma_2^{(k+1)}$, direct

calculations reduce this to

$$\Delta_k \quad = \quad ng\left(\frac{\gamma_1^{(k)}}{\gamma_1^{(k+1)}}\right) + ng\left(\frac{\gamma_2^{(k)}}{\gamma_2^{(k+1)}}\right) \geq ng\left(\frac{\gamma_1^{(k)}}{\gamma_1^{(k+1)}}\right) \quad ,$$

with $g(x) = x - 1 - log(x)$.

Let $\mu_1^{(k)} = \log(\gamma_1^{(k)})$. Hence

$$\Delta_k \quad \geq \quad g^*\left(\mu_1^{(k)} - \mu_1^{(k+1)}\right) \tag{3.39}$$

with $g^*(x) = n(e^x - 1 - x)$. The last function is continuous, null at 0 and strictly positive elsewhere.

Let define now $f^*(x) = \log f_{\mathbf{d}}(e^x, h(e^x))$. We can therefore rename the sequence $\log f_{\mathbf{d}}(\gamma_1^{(k)}, \gamma_2^{(k)})$ as $f^*(\mu_1^{(k)})$.

The sequence $f^*\left(\mu_1^{(k)}\right)$ is convergent (as EM can only increase the likelihood at each iteration, and the likelihood can be bounded in our case, this sequence is increasing and bounded). The sequence $(\mu_1^{(k)})$ can be bounded by construction (see the proof of Theorem 3.5.1). Finally, (3.39) shows that $f^*(\mu_1^{(k+1)}) - f^*(\mu_1^{(k)}) \geq \Delta_k \geq g^*(\mu_1^{(k)} - \mu_1^{(k+1)})$. $h$ and $f^*$ are continuous at any point greater than $\log \frac{n}{\sum d_i}$, which will be the case after the first iteration. Therefore, lemma 3.8.2 can be applied, the sequences $(\mu_1^{(k)})$ and $(\gamma_1^{(k)}) = (e^{\mu_1^{(k)}})$ converge. As $h$ is a continuous function, the sequence $\left(\gamma_2^{(k)}\right) = \left(h(\gamma_2^{(k)})\right)$ is also convergent, and this will be the case for the sequence $(\gamma_1^{(k)}, \gamma_2^{(k)})$.

We now prove that the limit is a solution of the likelihood equation. At any fixed point we have $\gamma_1^{(k+1)} = \gamma_1^{(k)} = \gamma_1^*$ and $\gamma_2^{(k+1)} = \gamma_2^{(k)} = \gamma_2^*$. Therefore, using (3.24) and (3.25):

$$\frac{\gamma_2^*}{(\gamma_2^* - \gamma_1^*)\gamma_1^*} = \frac{1}{n}\sum_{i=1}^{n} \frac{d_i e^{(\gamma_2^* - \gamma_1^*)d_i}}{e^{(\gamma_2^* - \gamma_1^*)d_i} - 1}$$

and

$$\frac{\gamma_1^*}{(\gamma_2^* - \gamma_1^*)\gamma_2^*} = \frac{1}{n}\sum_{i=1}^{n} \frac{d_i}{e^{(\gamma_2^* - \gamma_1^*)d_i} - 1} \quad ,$$

the same equations as the likelihood equation, which means that any fixed point of the E-M algorithm is also a solution of the likelihood equation.

**Proof of Lemma 3.8.1** Let $c$ be a point in $[a, b]$, and let construct a subsequence of $(x_n)$ that converges toward $c$. By definition, $a$ and $b$ are limit points of $(x_n)$, hence we can assume that $c \neq a$ and $c \neq b$.

Let $\epsilon = \min\left(\frac{c-a}{2}, \frac{b-c}{2}\right)$ be a positive number. $(x_{n+1} - x_n)$ converges towards zero. Hence, $\forall k, \exists N_k$ s.t. $\forall j > N_k, x_{j+1} - x_j < \frac{\epsilon}{k}$. By definition of a limit point, $\exists i_0 \leq N_1$, s.t. $x_{i_0} \in ]a - \epsilon, a + \epsilon[$. Similarly, $\exists j_0 > i_0$, s.t. $x_{j_0} \in ]b - \epsilon, b + \epsilon[$. Recursively, we can construct two integer sequences $(i_k)$ and $(j_k)$, such that $\forall k, N_{k+1} \leq i_k < j_k < i_{k+1}$, $i_k \in ]a - \epsilon, a + \epsilon[$ and $j_k \in ]b - \epsilon, b + \epsilon[$.

We are now in position to conclude. For all $k$, we have that $x_{i_k} < a + \epsilon \leq c - \frac{\epsilon}{k} < c + \frac{\epsilon}{k} \leq b - \epsilon < x_{j_k}$. Further more, $i_k < j_k$, and $\forall n > i_k, (x_{n+1} - x_n) \leq \frac{\epsilon}{k}$. This is enough to conclude that there exists $i_k < \phi(k) < j_k$ such that $x_{\phi(k)} \in ]c - \frac{\epsilon}{k}, c + \frac{\epsilon}{k}[$. Since $j_k < i_{k+1}$, the function $\phi(.)$ is strictly growing, and $(x_{\phi(n)})$ is a sub-sequence of $(x_n)$ convergent towards $c$. □

**Proof of Lemma 3.8.2** By assumption, the sequence $(x_n)$ is bounded. Therefore, it converges if and only if it admits one unique limit point.

Assume, by contradiction, that there is two distinct limit points $a$ and $b$, with $a < b$. The sequence $f(x_n)$ is convergent, therefore $(f(x_{n+1}) - f(x_n))$ converges toward zero. Using (3.38), we get that $g(x_{n+1} - x_n)$ is convergent toward zero. By contradiction, if $(x_{n+1} - x_n)$ admits one limit point $c \neq 0$, then the sequence $g(x_{n+1} - x_n)$ admits $g(c) > 0$ as limit point, which contradicts the fact that it converges to 0. Hence, $(x_{n+1} - x_n)$ admits no non-zero limit point, and as it is bounded, converges to 0.

Using lemma 3.8.1, we get that $\forall c \in [a, b]$, $c$ is a limit point of $(x_n)$, and hence $f(c)$ is a limit point of $f(x_n)$. As $f(x_n)$ converges towards $l$, it admits one unique limit point, and $\forall c \in [a, b], f(c) = f(a) = f(b) = l$. Let $\epsilon = \frac{b-a}{3}$ be a positive number, and let now $N$ be such that $\forall n \geq N, |x_{n+1} - x_n| < \epsilon$. As $a$ and $b$ are limit points of $(x_n)$, there exists $n_1 \geq N$ and $n_2 \geq n_1$ such that $|x_{n_1} - a| < \epsilon$ and $|x_{n_2} - b| < \epsilon$. Then $\exists n_3$ s.t. $n_1 \leq n_3 < n_3 + 1 \leq n_2$, $x_{n_3} \neq x_{n_3+1}$, $x_{n_3} \in ]a, b[$ and $x_{n_3+1} \in ]a, b[$. On one hand, (3.38) leads to $f(x_{n_3+1}) > f(x_{n_3})$. On the other hand, $f(x_{n_3}) = f(x_{n_3+1}) = l$. We get a contradiction. □

### 3.8.2 Proof of Lemma 3.5.4

For the three router case, for all $k \geq 0$, (3.23) is equivalent to

$$Q(\theta_1, \theta_2, \theta_3 | \gamma_1, \gamma_2, \gamma_3) = \sum_{i=1}^{n} \beta(d_i) \int_{l_1=0}^{d_i} \int_{l_2=0}^{d_i - l_1} (\log(\theta_1\theta_2\theta_3) - \theta_3 d_i + (\theta_3 - \theta_1) l_1 + (\theta_3 - \theta_2) l_2) e^{(\gamma_3 - \gamma_1) l_1} e^{(\gamma_3 - \gamma_2) l_2} dl_2 dl_1 \quad (3.40)$$

with $\beta$ defined in (3.31). We have

$$\int_{l_1=0}^{d} \int_{l_2=0}^{d-l_1} (a + bl_1 + cl_2)\, e^{\alpha l_1} e^{\beta l_2} dl_2 dl_1 = ac_a + bc_b + dc_d$$

with

$$
\begin{aligned}
c_a =& \frac{\beta(e^{\alpha d}-1)-\alpha(e^{\beta d}-1)}{\alpha\beta(\alpha-\beta)} \\
c_b =& \frac{\alpha\beta(\alpha-\beta)de^{\alpha d}-\beta(2\alpha-\beta)e^{\alpha d}+\alpha^2 e^{\beta d}-(\alpha-\beta)^2}{\alpha^2\beta(\alpha-\beta)^2} \\
c_c =& \frac{\alpha(\alpha-2\beta)e^{\beta d}-\alpha\beta(\alpha-\beta)de^{\beta d}+\beta^2 e^{\alpha d}-(\alpha-\beta)^2}{\alpha\beta^2(\alpha-\beta)^2} \quad .
\end{aligned}
$$

In order to evaluate (3.40) we have to take $\alpha = \gamma_3 - \gamma_1$, $\beta = \gamma_3 - \gamma_2$, $a = \log(\theta_1\theta_2\theta_3) - \theta_3 d_i$, $b = (\theta_3 - \theta_1)$ and $c = (\theta_3 - \theta_2)$ (note that $\alpha - \beta = \gamma_2 - \gamma_1$). In addition

$$
\beta(d_i) = \frac{\alpha\beta(\alpha-\beta)}{\beta e^{\alpha d} - \alpha e^{\beta d} + \beta - \alpha} = \frac{1}{c_a} \quad .
$$

Finally

$$
\begin{aligned}
Q(\theta_1,\theta_2,\theta_3|\gamma_1,\gamma_2,\gamma_3) =& \sum_{i=1}^{n} \alpha(d_i)\left[c_a \log(\theta_1\theta_2\theta_3) - c_b\theta_1 - c_c\theta_2 + (c_b + c_c - d_i c_a)\theta_3\right] \\
=& n\log(\theta_1\theta_2\theta_3) + \sum_{i=1}^{n}\left(\frac{c_b+c_c}{c_a} - d_i\right)\theta_3 - \frac{c_b}{c_a}\theta_1 - \frac{c_c}{c_a}\theta_2 \quad ,
\end{aligned}
$$

and therefore:

$$
\frac{\partial Q(\theta_1,\theta_2,\theta_3|\gamma_1,\gamma_2,\gamma_3)}{\partial\theta_1} = \frac{n}{\theta_1} - \sum_{i=1}^{n}\frac{c_b}{c_a} \quad .
$$

The expressions given in Lemma are then directly obtained from

$$
\frac{\partial Q(\gamma_1^{(k+1)},\gamma_2^{(k+1)},\gamma_3^{(k+1)}|\gamma_1^{(k)},\gamma_2^{(k)},\gamma_3^{(k)})}{\partial\gamma_1^{(k+1)}} = 0
$$

and other relations of the same type.

# Chapter 4

# Extension to Kelly Networks

## 4.1 Introduction

We have seen in chapter 3 that the distribution of probe packet end-to-end delays is sufficient to estimate the set of available bandwidths along a single path, when the delay distribution at each node belongs to a known parametric family. The goal of this chapter is to extend these results to the case of a network, exploiting the inherent path diversity.

In other words, we study a network inference problem with a firm foundation in queueing networks, thereby contributing simultaneously to network tomography, and to the area of inverse problems in queueing. We focus to the particular case of *point-to-multipoint* inverse problems, where a single source sends probe packets to multiple destinations over a feedforward network of nodes. The network can hence be considered as a tree, whose root is the sender and whose leaves are the receivers. We consider *multicast trees*, where each node of the tree copies its departing probes over all of its child links. Hence each probe sent from the root node effectively broadcasts over the entire tree until copies arrive at each leaf. Timestamps at the root and leaves can be compared, so that each multicast probe gives rise to a vector of delay values. Multicasting is supported by today's Internet protocols and represents an economical way to reach many receivers, and most works on delay tomography exploit it.

A typical delay model used in tomography over multicast trees is given as follows. To each node there is a random process controlling the delays imparted to packets. The node processes[53] are mutually independent (spatial independence) and are each individually i.i.d. (temporal independence). Thus the end-to-end delay of each probe at a given leaf is the sum of independent random variables, with (in general) different distributions, corresponding to its ancestor nodes in the tree, as shown in the example of Figure 4.1. The normal or *cross-traffic* packets in the network are taken to be responsible for the build up of node queues and hence the delays which are experienced by probes, however they do not enter explicitly in the description. Cross traffic is not assumed to be multicast, indeed

[53]Note that usually the processes are associated to links, not nodes, but as these are in 1-1 correspondence this is not essential.

the multicast tree is a construct of the probing experiment, whereas cross traffic traverses the full network and simply intersects the tree. Finally, it is assumed that probes are rare enough so as not to significantly perturb the normal traffic over the tree.

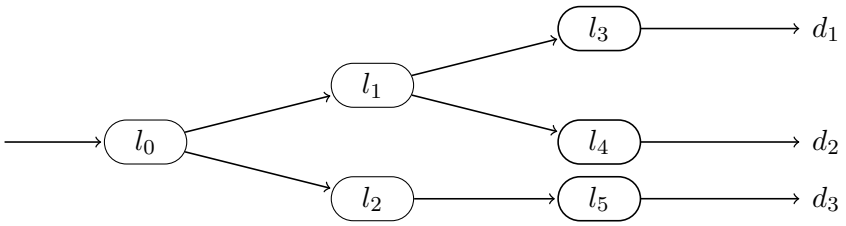



Figure 4.1: Example of a delay tomography problem over a tree: to estimate the means of the six internal random variables $l_0, l_1, l_2, l_3, l_4$ and $l_5$, just by observing samples of the three end-to-end delay variables $d_1$, $d_2$ and $d_3$, where $d_1 = l_0 + l_1 + l_3$, $d_2 = l_0 + l_1 + l_4$, and $d_3 = l_0 + l_2 + l_5$.

In this chapter we study the tomography problem as above in the case where the node delay variables are each exponentially distributed. We formulate a Maximum Likelihood Estimation problem for their parameters, implemented using the Expectation-Maximization algorithm. Our contributions are as follows. First we show how the tomography model described above corresponds to the delays experienced by probes in an appropriately defined queueing network also carrying cross traffic, thereby justifying the assumptions of a delay tomography problem over a tree in terms of queueing networks for the first time. Second, as a delay tomography problem, it is novel in that (see below for details) we do not focus on non-parametric estimation or alternatively with general but discretized delay, but instead work with the full MLE of a continuous density. In particular this involves dealing, both theoretically and practically, with the non-trivial combinatorics inherent in the conditional expectations over a general tree topology. We derive explicit solutions for the combined E and M steps. Finally, we provide a technique for convergence acceleration of the E-M algorithm, which is notoriously slow, allowing much larger trees to be considered than would otherwise be the case. The technique has some novel features and may be of broader interest.

Our work is the first to propose a delay tomography model based on exponential delays (see however [LMN07]). Given the accepted queueing origins of network delays, it is surprising that such a canonical choice has escaped attention until now. The chief reason for this omission, as argued for example in [PDHT02], is that there is no generally accepted model for the delays in Internet routers, so that flexibility is essential to match reality. While this point is well taken, our view is that realism also requires that node models be consistent with their purported queueing underpinnings, something which has never been shown previously, even in models which introduce, a priori, atomic components in an attempt to reproduce queue-like features [SH03, LMN07]. Although the exponential distribution is not considered to be a close fit to packet delays in the Internet today, is it a natural first choice when making a rigorous connection to queueing networks.

It is well known that the convergence of the E-M algorithm can be slow, and there is a considerable literature devoted to speed-up techniques. An element of our technique involves over-relaxation, that is inflating the jump size recommended by E-M. This idea is not new, for example it figures in [JJ93, SR03], and was explored by Lange and others in the context of E-M Gradient Algorithms (see §4.13, [MK08]). However, our jump size update rule, which does not bound the allowed increase at any step, is extremely aggressive, and qualitatively different to those we have seen elsewhere, although it shares with [HYH05] the principle that if a candidate step proves too aggressive, in particular if it leads to a decrease in likelihood, then a safer 'fallback' position can be taken (see also 'step decrementing' §4.14.1, [MK08]). The other core element of our technique involves using Principal Component Analysis (PCA) to efficiently exploit the information contained in prior evaluations of the likelihood, and to help counter the instability inherent in aggressive updates. This approach was inspired by recent work in robotics [DL08] in the quite different context of automated path finding. We know of no work which uses similar ideas to accelerate E-M or related algorithms.

To give an example of applications, our techniques could be used by service providers in order to monitor the quality of real-time services. In the case of ADSL 'triple-play boxes' providing IP TV services today, service providers own the end-user equipment, and so can run measurement software as well as operate the backbone and access networks. They therefore have the incentive and the ability to use multicast protocols.

The chapter is structured as follows. Section 4.2 describes the queueing inverse problem and how it maps to the delay tomography problem. Section 4.3 shows how these apply in the present case. Section 4.4 is a technical one showing how expressions for the conditional expectations over the tree which arise can be calculated explicitly. Section 4.5 exploits these solutions to provide the MLE for a number of example trees, using our E-M acceleration technique, which itself is described (and further illustrated) in Section 4.6. We conclude and comment on future work in Section 4.7.

## 4.2 A Delay Tomographic Queueing Inverse Problem

We begin with the model for cross-traffic only, and then consider how probes can be introduced.

Consider an open Kelly network of single server FCFS queueing stations connected in a tree topology. Routes corresponding to a given customer class can only move away from the root station (and are not multicast), but are otherwise general, entering the tree at any station and exiting either that same station, or any other further down the tree. The arrival process to route (or class) $r$ is Poisson of intensity $\lambda_r$. All packets have exponential size with mean 1, and the service rate of station $j$ is $\mu_j$. We consider only parameter values consistent with a stationary regime.

We first consider the special case of a tandem network of $K$ stations that we studied in the previous chapter, as it serves as a building block for what follows. Assume that route

$r = 0$ traverses the network from root to leaf, so that for each customer in class 0 we can associate an end-to-end system time, or *delay* $d$. We will call such a route a *path*. From (3.3) the marginal distribution for the number $N_0^j$ of customers of class 0 in station $j, 1 \leq j \leq K$, at a given time instant is

$$\mathbb{P}(N_0^j = n_0^j, j = 1, \ldots, K) = \prod_{j=1}^{K} \left(\frac{\lambda_0}{\gamma_j}\right)^{n_0^j} \left(1 - \frac{\lambda_0}{\gamma_j}\right) \quad ,$$

where $\gamma_j = \mu_j - \sum_{r \neq 0, j \in r} \lambda_r$ is the residual service capacity of station $j$ available to class 0. From lemma 3.2.1 (and its proof), we know that $d$ is the sum of $K$ independent exponential variables, one per station, where the mean parameter for station $j$ is just the reciprocal of the residual service capacity $\gamma_j - \lambda_0$. Furthermore, from corollary 1.2.11, we know that the departure processes of the classes exiting the system at station $K$ are Poisson and mutually independent, and that departures from any of these prior to some time $t$ are independent of the system state at time $t$.

Now consider a tree network. The above result for a tandem applies directly to any path, that is the end-to-end delay of each customer of a path is given by the sum of independent exponentials. Note however that this does not imply that the delays seen over different paths are independent. Now the set of stations in any two paths can be partitioned into three tandem subnetworks: a shared portion $S$ from the root down to some last shared station $A$, and two unshared portions $U_1$ and $U_2$ beginning from children of $A$, each terminating at a leaf.

The independence properties given above for the tandem network apply to customers exiting $A$. They imply that the arrival processes to each of $U_1$ and $U_2$ are independent not only of each other, but also of the states of $U_1$ and $U_2$, since the latter are functions only of the prior departures from $S$, which as noted above are independent of the state of $S$ at the departure instant of each probe. Since the service times of the stations in $U_1$ and $U_2$ are also mutually independent, it follows that the delays incurred over $U_1$ and $U_2$ are likewise independent both of each other, and of the delays incurred (by the customers of either path) over $S$. In summary, delays over the tandem subnetworks $S$, $U_1$, and $U_2$ are mutually independent, and inside each of these, delays experienced by customers of a given class (i.e. path) are given by a sum of independent exponentials. This argument extends naturally to the entire tree.

We now introduce multicast probe customers into the system, which behave as follows. The probes arrive as a Poisson process of intensity $\Lambda$ to the root station. Once a probe has arrived to a station it is treated exactly like a normal customer, but upon exiting, copies are instantaneously made which arrive simultaneously to each of its child stations. Hence each multicast probe traverses all paths (end-to-end routes) but no other routes.

Clearly the system consisting of cross-traffic classes plus the multicast probe class over the tree is not a Kelly network. However, as before the tandem analysis above applies, showing not only that probe delays over each path are distributed as a sum of independent

exponentials, but also that the probe delays on a given path can be analyzed as if the cross-traffic were absent, provided the appropriate reduced capacities are used. Furthermore, the above arguments concerning the decoupling of the delays experienced over the shared portion of paths from those below it continue to hold. However, the relationship between delays seen by customers of different paths *within* the shared or the un-shared portions is now substantially different.

To examine this question we revisit our two path example, but now consider the customers on each path to belong to the same multicast probe class.

**Shared part:** there is now only a single probe customer process rather than two. This can be interpreted as perfect station-by-station dependence of the delay components from each path, in contrast to the situation for cross traffic where the service times of customers, for example, were independent.

**Unshared part:** the arrival processes from $A$ to $U_1$ due to path 1, and $A$ to $U_2$ due to path 2, remain Poisson, but are now identical rather than independent, resulting in dependence between the delays of probes (and cross traffic) seen over $U_1$ and $U_2$.

To see why the delays of probes are now dependent on the unshared part, consider the following simple example without cross-traffic, where $U_1$ and $U_2$ each consist of a single node of capacity $\mu$. In other words, $U_1$ and $U_2$ are M/M/1 queues with independent service times, fed by the same Poisson Process of intensity $\Lambda$. Each queue has a marginal probability $(1-\rho) = (1-\frac{\Lambda}{\mu})$ to be empty. Now $U_1$ (resp. $U_2$) is empty at the arrival time $t_N$ of the $N^{th}$ probe packet if and only if the previous probe had a delay $D_1$ (resp. $D_2$) which is less than the inter-arrival time $t_N - t_{N-1}$. Assume for contradiction that spatial independence holds between $D_1$ and $D_2$, this leads to:

$$\mathbb{P}[\text{both queues empty}] = \int_0^\infty \mathbb{P}[D_1 \leq \tau, D_2 \leq \tau]\mathbb{P}[t_N - t_{N-1} = \tau]d\tau = 1 - 2\rho + \frac{\rho^2}{2\rho - \rho^2}$$

which is not equal to $(1-\rho)^2$ (unless $\rho = 1$), the result one would obtain if the waiting times were independent. But this is a contradiction, because the assumptions of independence between $D_1$ and $D_2$, and on the service times, clearly implies independence of waiting times. It follows that the delays must in fact be dependent.

Although multicast probes break the strict spatial independence property of path delays, we expect this dependence to be weak in most cases, since the arrival processes to $U_1$ and $U_2$ remain independent of the states of $U_1$ and $U_2$ (at arrival instants), the service times in $U_1$ and $U_2$ remain independent, and furthermore the cross-traffic arrivals (from paths or other routes) are independent as before. In particular, if we assume that $\Lambda$ is small, so that with high probability there is no more than a single probe in any given station, then the states of $U_1$ and $U_2$ are only slightly perturbed by probes and are thus approximately independent, and so the delays over $U_1$ are $U_2$ are likewise close to independent.

It is a general principle of network probing that $\Lambda$ be kept small, in order to avoid consuming network bandwidth, perturbing the system to be measured, and to prevent probes

being confused with network attacks. Since $\Lambda$ is under the control of the prober, it is quite reasonable to assume it is small. This same *rare probing* assumption justifies the assumption of temporal independence in the time series of probe delays associated to each path, used in the MLE formulation below.

In conclusion, the delays of rare multicast probes sent over a Kelly tree network of cross traffic closely hew to the assumptions of a spatially and temporally independent delay tomography problem over a tree with exponential delays. Namely, per-station delays experienced by probes obey a simple structure: perfect dependence over stations on the shared part of the path, and independence between the unshared parts. Cross traffic appears only through the values of the residual capacity parameters $\{\gamma_j - \Lambda\}$ to be estimated. Since $\Lambda$ is known, the residual capacities $\{\gamma_j\}$ relating to cross traffic only can subsequently be recovered. The actual intensities $\{\lambda_r\}$ and the server rates $\{\mu_j\}$ are not identifiable, however they can be recovered in principle by other means, for example using a prior measurement phase with fixed packet sizes, as discussed in the tandem case in chapter 2[54].

## 4.3 E-M for Exponential Tomography

In this section we apply the E-M algorithm to our delay tomographic problem.

Consider a tree $\mathcal{T}$, and call $T$ the set of its nodes and $V \subset T$ the set of its leaves. We introduce the fixed parameter vector $\alpha = (\alpha_j)_{j \in T}$ and the variable parameter vectors $\hat{\alpha}^{(k)} = (\hat{\alpha}_j^{(k)})_{j \in T}$. The complete data random vector of the E-M algorithm will correspond to the vector $l \in \mathbb{R}^T$ of the delays of each node, which are supposed independent and exponentially distributed with expected value $\alpha$, and the observed data vector $y$ will correspond to $d \in \mathbb{R}^V$, the vector of all end-to-end delays from the root to each leaf. We will have $d = f(l)$ for some linear function $f$ depending on the topology of the tree. We recall that the probability density function of an exponentially distributed variable of mean value $\alpha_j$ is $p_{\alpha_j}(z) = \frac{1}{\alpha_j} e^{-z/\alpha_j}$.

The fixed vector $\alpha$ will be referred as the *ground truth*, and the variables vectors $\hat{\alpha}^{(k)}$ as the current estimates (of E-M). We wish to estimate $\alpha$ via $\hat{\alpha}^{(k)}$, hoping that this last sequence will converge 'close to' $\alpha$. In networks context, $\alpha_j$ corresponds to the mean sojourn time of probes in server $j$.

*Remark.* Note that in Kelly networks, the mean sojourn time $\alpha_j$ on node $j$ is the inverse of the residual capacity $\gamma_j$ on this node. Both quantities are relevant in practice, and it is equivalent to estimate one or the other on a single occurence, due to the function invariance of maximum likelihood estimators. In this chapter, using the mean delays instead of available bandwidth leads to simpler equations to write. Recall however that, as the inverse function is not linear, the mean estimation of the mean delay is not the inverse of the mean estimate of the available bandwidth.

[54] An interesting discussion, in the context of a priori node models, of how the addition of atoms can assist in identifiability is given in [SH03].

The results given in this section actually hold more generally, for any set $T$ and $V$ with any random exponential vector $l \in \mathbb{R}^T$ (with independent coordinates) and any linear function $f : \mathbb{R}^T \to \mathbb{R}^V$. In particular they hold for delay tomography problems where the network topology is not tree-like.

### 4.3.1 Specialization of the Iterative Formula

Usually, each iteration of E-M can be computed in two steps: the E-step, where we compute the conditional expectation of the log-likelihood, and the M-step where we maximize it. But when the hidden data belongs to the regular exponential family, as it is the case here, it is well known [MK08] that the E- and M-steps can be solved directly in one step. In other words, the iteration can be made more explicit. Indeed, we have:

$$\begin{aligned} \hat{\alpha}^{(k+1)} &:= \operatorname*{argmax}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\hat{\alpha}^{(k)}}(\log p_\theta(l) | f(l) = d(i)) \\ &= \operatorname*{argmax}_{\theta} \sum_{i=1}^{N} \frac{\int_{\{l|f(l)=d(i)\}} \log(p_\theta(l)) p_{\hat{\alpha}^{(k)}}(l) dl}{\int_{\{l|f(l)=d(i)\}} p_{\hat{\alpha}^{(k)}}(l) dl} \quad , \end{aligned} \tag{4.1}$$

where in our case $p_\theta(l) = \prod_{j \in T} \frac{1}{\theta_j} e^{-\frac{l_j}{\theta_j}}$, and $\log p_\theta(l) = \sum_{j \in T} (\log \frac{1}{\theta_j} - \frac{l_j}{\theta_j})$ for every $\theta$.

We notice that $\log p_\theta(l)$ is easily differentiable according to $\theta$, giving:

$$\frac{\partial \log p_\theta(l)}{\partial \theta_j} = -\frac{1}{\theta_j} + \frac{l_j}{\theta_j^2} = -\frac{1}{\theta_j^2}(\theta_j - l_j) \quad ,$$

and therefore, $\mathbb{E}_{\hat{\alpha}^{(k)}}(\log p_\theta(l) | f(l) = d(i))$ is also differentiable, with:

$$\begin{aligned} \frac{\partial \mathbb{E}_{\hat{\alpha}^{(k)}}(\log p_\theta(l) | f(l) = d(i))}{\partial \theta_j} &= -\frac{1}{\theta_j^2} \frac{\int_{\{l|f(l)=d(i)\}} (\theta_j - l_j) p_{\hat{\alpha}^{(k)}}(l) dl}{q_{\hat{\alpha}^{(k)}}(d(i))} \\ &= -\frac{1}{\theta_j^2} \mathbb{E}_{\hat{\alpha}^{(k)}}(\theta_j - l_j | d(i)) = -\frac{1}{\theta_j^2}(\theta_j - \mathbb{E}_{\hat{\alpha}^{(k)}}(l_j | d(i))) \quad . \end{aligned}$$

We then have:

$$\frac{\partial \sum_{i=1}^{N} \mathbb{E}_{\hat{\alpha}^{(k)}}(\log p_\theta(l) | f(l) = d(i))}{N \partial \theta_j} = -\frac{1}{\theta_j^2} \left( \theta_j - \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\hat{\alpha}^{(k)}}(l_j | d(i)) \right) \quad .$$

Thus, setting this derivative to zero leads to $\theta = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\hat{\alpha}^{(k)}}(l | d(i))$, and so

$$\hat{\alpha}^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\hat{\alpha}^{(k)}}(l | d(i)) \quad . \tag{4.2}$$

*Remark.* Here we have generalized the conditional expectation to the multivariate case.

That is, in $\mathbb{E}_{\hat{\alpha}^{(k)}}(l|d(i))$, $l$ is a vector, where we have defined $\mathbb{E}_{\hat{\alpha}^{(k)}}(l|d(i)) := (\mathbb{E}_{\hat{\alpha}^{(k)}}(l_j|d(i)))_{j\in T}$, and the sum $\sum_{i=1}^{N}\mathbb{E}_{\hat{\alpha}^{(k)}}(l|d(i))$ is to be understood as a component-wise addition.

We just reduced the E- and M-steps to one, a significant simplification which in many contexts would almost constitute a 'solution' to the problem. However, computing the conditional expectation $\mathbb{E}_{\hat{\alpha}^{(k)}}(l|d)$ remains a challenge as it involves dealing with combinatorics over the tree, and is in fact a main part of our work. In the next section, we explain how it can be computed efficiently. First, we point out an interesting property which will be useful later.

**Proposition 4.3.1.** *Let $l \in \mathbb{R}^T$ be the vector of the delays of each node, and $d \in \mathbb{R}^V$ the vector of all end-to-end delays from the root to each leaf. Assume that for some linear function $f$ depending on the topology of the tree, we will have $d = f(l)$ for some . Assume finally that $l$ is an exponentially distributed random vector with mean $\alpha \in \mathbb{R}^T$, and that $(l_1, \dots, l_N)$ are $N$ i.i.d. random vectors (with the same distribution as $l$).*

*Let $(d(i))_{1\le i\le N} = (f(l(i)))_{1\le i\le N}$ be the measured end-to-end delays, $\hat{\alpha}^{(k)}_{k\in\mathbb{N}}$ be the sequence of successive estimates of $\alpha$ by the E-M algorithm based on these delays. We then have for all $k$:*

$$f(\hat{\alpha}^{(k+1)}) = \bar{d} = \frac{1}{N}\sum_{i=1}^{N} d(i) \quad , \tag{4.3}$$

where again $\bar{d}$ is a vector defined by averaging component-wise.

*Proof.* Thanks to the linearity of the conditional expectation and the linearity of $f$, we have in our case that $\mathbb{E}_{\hat{\alpha}^{(k)}}(f(l)|d) = f(\mathbb{E}_{\hat{\alpha}^{(k)}}(l|d))$. Therefore,

$$f(\hat{\alpha}^{(k+1)}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\hat{\alpha}^{(k)}}(f(l)|d(i)) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\hat{\alpha}^{(k)}}(d|d(i)) = \frac{1}{N}\sum_{i=1}^{N} d(i) \quad .$$

□

Because of this relation, we know that each term of the EM sequence $(\hat{\alpha}^{(k)})$ except the first will satisfy $f(\hat{\alpha}^{(k)}) = \bar{d}$. Therefore, the sequence stays in $f^{-1}(\bar{d})$ which, since $f$ is linear, is a linear subspace of $\mathbb{R}^T$.

## 4.4 Explicit Formula for $\mathbb{E}(l|d)$

In this section we compute the conditional expectation $\mathbb{E}_{\hat{\alpha}^{(k)}}(l|d)$, which is the key to the evaluation of the step function (4.2). Since

$$\mathbb{E}_{\hat{\alpha}^{(k)}}(l|d) = \frac{\int_{f^{-1}(d)} l p_{\hat{\alpha}^{(k)}}(l)dl}{\int_{f^{-1}(d)} p_{\hat{\alpha}^{(k)}}(l)dl} := \frac{\xi_{\hat{\alpha}^{(k)}}(l|d)}{q_{\hat{\alpha}^{(k)}}(d)} \quad , \tag{4.4}$$

the calculation can be divided in the computation of the two terms $q_{\hat{\alpha}^{(k)}}(d)$ and $\xi_{\hat{\alpha}^{(k)}}(l|d)$.

By their nature these calculations are detailed. This section is self-contained and could be skipped on a first reading.

### 4.4.1 Notations

In this section, we are interested in a single iteration of the EM algorithm. In order to simplify notations, we will therefore here (and only here) write $\alpha$ instead of $\hat{\alpha}^{(k)}$. Similarly, in order to have another point of view and nicer notations, we will also introduce the *rate* $\gamma_j := \frac{1}{\alpha_j}$ and use the notation $p_{\alpha_j}(z) = \gamma_j e^{-g_j z}$ instead.

We recall that $\mathcal{T}$, $T$, $V \subset T$, $l = (l_j)_{j\in T} \in \mathbb{R}^T$ and $d = (d_j)_{j\in V} \in \mathbb{R}^V$ denotes respectively the tree we consider, the set of its node, the set of its leaves, the vector of delays on each nodes and the vector of end-to-end delays in the tree. The variable vectors $\alpha = (\alpha_j)_{j\in T}$ and $\gamma = (\gamma_j)_{j\in T}$ is the current estimate of EM.

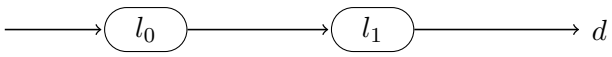
As in Section 4.3, the observed data are the end-to-end delay vectors $d(1), \ldots, d(N)$, and are the images under some linear function $f_{\mathcal{T}}$ of the unknown complete data $l(1), \ldots, l(N)$, where $f_{\mathcal{T}}$ captures the details of the tree topology.

We provide $T$ with the order $\prec$ defined by: for all $i, j$ in $T$, $i \prec j$ if $i$ is an ancestor of $j$. With these notations, the function $f_{\mathcal{T}} : \mathbb{R}^T \to \mathbb{R}^V$ such that $f_{\mathcal{T}}(l) = d$ is given by $\forall k \in V,\ (f_{\mathcal{T}}(l))_k = d_k = \sum_{\substack{j\in T \\ j\preceq k}} l_j$, and the two terms of the fraction (4.4) can be written:

$$q_\alpha(\mathcal{T}, d) = \int_{f_{\mathcal{T}}^{-1}(d)} \prod_{j\in T} \gamma_j e^{-\gamma_j l_j} dl \quad \text{and} \quad \xi_\alpha(\mathcal{T}, l|d) = \int_{f_{\mathcal{T}}^{-1}(d)} l \prod_{j\in T} \gamma_j e^{-\gamma_j l_j} dl \quad . \tag{4.5}$$

### 4.4.2 Some simple examples

**a) 2 Nodes Tree**




In this simple case from chapter 3, since $l_0$ and $l_1$ are linked to $d$ by $l_0 + l_1 = d$, there is only one unknown. Therefore $q_\alpha$ can be expressed as an integral over $l_0$ only.

$$q_\alpha(\mathcal{T}, d) = \gamma_0\gamma_1 \int_{l_0=0}^{d} e^{-\gamma_0 l_0} e^{-\gamma_1 (d-l_0)} dl_0 = \gamma_0\gamma_1 \left( \frac{e^{-\gamma_0 d}}{\gamma_1 - \gamma_0} + \frac{e^{-\gamma_1 d}}{\gamma_0 - \gamma_1} \right) \quad ,$$

and similarly:

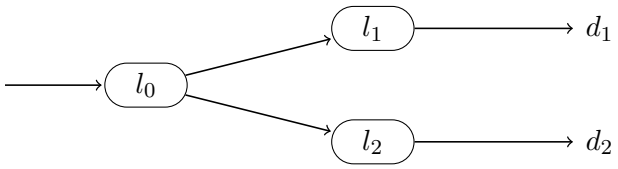
$$\xi_\alpha(\mathcal{T}, l_0|d) = \gamma_0\gamma_1 \left( \frac{e^{-\gamma_0 d}}{\gamma_1 - \gamma_0} \left( d - \frac{1}{\gamma_1 - \gamma_0} \right) + \frac{e^{-\gamma_1 d}}{(\gamma_0 - \gamma_1)^2} \right) \quad .$$

Although the figure does not suggest it, the problem is actually symmetric in the nodes 0 and 1. Indeed, what we observe being the sum of two delays, the tree $0 \to 1$ is equivalent

to the tree $1 \to 0$. Therefore, we have by symmetry:

$$\xi_\alpha(\mathcal{T}, l_1|d) = \gamma_0\gamma_1 \left( \frac{e^{-\gamma_0 d}}{(\gamma_1 - \gamma_0)^2} + \frac{e^{-\gamma_1 d}}{\gamma_0 - \gamma_1} \left( d - \frac{1}{\gamma_0 - \gamma_1} \right) \right) \quad .$$

**b) Root with 2 Leaves**




In this case, since $l_0 + l_1 = d_1$ and $l_0 + l_2 = d_2$, we can consider as before only one unknown $l_0$, and express $q_\alpha$ as an integral over $l_0$ between 0 and $d_0 := \min\{d_1, d_2\}$. Since $l_0, l_1$ and $l_2$ are nonnegative, $l_0$ has to be smaller than $d_1$ and $d_2$. We have:

$$\begin{aligned} q_\alpha(\mathcal{T}, d) &= \gamma_0\gamma_1\gamma_2 \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} e^{-\gamma_1(d_1 - l_0)} e^{-\gamma_2(d_2 - l_0)} dl_0 \\ &= \gamma_0\gamma_1\gamma_2 \left( e^{-\gamma_0 d_0} \frac{e^{-\gamma_1(d_1 - d_0)} e^{-\gamma_2(d_2 - d_0)}}{\gamma_1 + \gamma_2 - \gamma_0} + \frac{e^{-\gamma_1 d_1} e^{-\gamma_2 d_2}}{\gamma_0 - \gamma_1 - \gamma_2} \right) \quad , \end{aligned}$$

and similarly:

$$\begin{aligned} \xi_\alpha(\mathcal{T}, l_0|d) = \gamma_0\gamma_1\gamma_2 \Bigg( & e^{-\gamma_0 d_0} \frac{e^{-\gamma_1(d_1 - d_0)} e^{-\gamma_2(d_2 - d_0)}}{\gamma_1 + \gamma_2 - \gamma_0} \left( d_0 - \frac{1}{\gamma_1 + \gamma_2 - \gamma_0} \right) \\ & + \frac{e^{-\gamma_1 d_1} e^{-\gamma_2 d_2}}{(\gamma_0 - \gamma_1 - \gamma_2)^2} \Bigg) \quad , \\ \xi_\alpha(\mathcal{T}, l_1|d) = \gamma_0\gamma_1\gamma_2 \Bigg( & e^{-\gamma_0 d_0} \frac{e^{-\gamma_1(d_1 - d_0)} e^{-\gamma_2(d_2 - d_0)}}{(\gamma_1 + \gamma_2 - \gamma_0)} \left( d_1 - d_0 - \frac{1}{\gamma_0 - \gamma_1 - \gamma_2} \right) \\ & + \frac{e^{-\gamma_1 d_1} e^{-\gamma_2 d_2}}{\gamma_0 - \gamma_1 - \gamma_2} \left( d_1 - \frac{1}{\gamma_0 - \gamma_1 - \gamma_2} \right) \Bigg) \quad , \end{aligned}$$

and $\xi_\alpha(\mathcal{T}, l_2|d)$ can be deduced from $\xi_\alpha(\mathcal{T}, l_1|d)$ by symmetry between nodes 1 and 2.

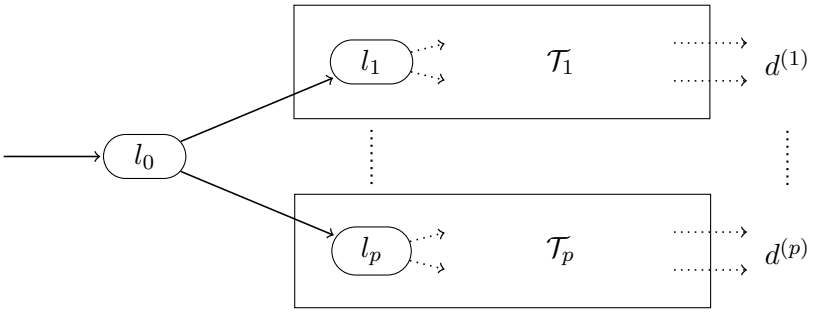
### 4.4.3 Inductive Expression

The last example above can readily be extended to more than two leaves. More generally, it suggests that it be possible to express $q_\alpha$ (resp. $\xi_\alpha$) for any tree as an integral over the delay in the root node from 0 to the minimum of the end-to-end delays, of some term using $q_\alpha$ (resp. $\xi_\alpha$ and $q_\alpha$) inductively applied to the child subtrees of the root. We now show how this can be done.

Let 0 denote the root of the tree, and $p$ the number of its children. In the case where the tree is a single node, *i.e.* $p = 0$, we have obviously $q_\alpha(d) = \gamma_0 e^{-\gamma_0 d_0}$. When $p \geq 1$ we

denote by $\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(p)}$ the associated child subtrees. Subtree $\mathcal{T}^{(i)}$ has nodes $T^{(i)}$ and leaves $V^{(i)} \subset T^{(i)}$.

We notice that $(V^{(1)}, V^{(2)}, \ldots, V^{(p)})$ forms a partition of $V$, and therefore any vector $d$ in $\mathbb{R}^V$ can be identified with a vector $d = (d^{(1)}, d^{(2)}, \ldots, d^{(p)})$ in $\mathbb{R}^{V^{(1)}} \times \mathbb{R}^{V^{(2)}} \times \cdots \times \mathbb{R}^{V^{(p)}}$. Similarly, any vector $l$ in $\mathbb{R}^T$ can be identified with a vector $l = (l_0, l^{(1)}, \ldots, l^{(p)})$ in $\mathbb{R} \times \mathbb{R}^{T^{(1)}} \times \cdots \times \mathbb{R}^{T^{(p)}}$.




**Theorem 4.4.1.** *Define* $d_0 := \min\{d_j \mid j \in V\}$. *The following inductive relation holds:*

$$q_\alpha(\mathcal{T}, d) = \int_{l_0=0}^{d_0} \gamma_0 e^{-\gamma_0 l_0} \left[ \prod_{i=1}^{p} q_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0)) \right] dl_0 \quad , \tag{4.6}$$

*where the slightly abusive notation* $d^{(i)} - (l_0)$ *denotes the vector* $(d_j^{(i)} - l_0)_{j \in V^{(i)}} \in \mathbb{R}^{V^{(i)}}$.

*Proof.* For a more convenient notation, we introduce for each $i \in \{1, \ldots, p\}$ the function $f^{(i)} := f_{\mathcal{T}^{(i)}}$ which is to the tree $\mathcal{T}^{(i)}$ what the function $f_\mathcal{T}$ is to the tree $\mathcal{T}$.

We notice that the following relation holds: for all $l$ in $\mathbb{R}^T$, let $d = f(l)$, then for all $k \in V$, there exist one unique $i \in \{1, \ldots, p\}$ such that $k \in V^{(i)}$, and:

$$d_k^{(i)} = d_k = \sum_{\substack{j \in T \\ j \preceq k}} l_j = l_0 + \sum_{\substack{j \in T^{(i)} \\ j \preceq k}} l_j^{(i)} = l_0 + \left( f^{(i)}(l^{(i)}) \right)_k \quad .$$

Which gives for all $i \in \{1, \ldots, p\}$, $d^{(i)} = (l_0) + f^{(i)}(l^{(i)})$, and therefore:

$$l^{(i)} \in (f^{(i)})^{-1}(d^{(i)} - (l_0)) \quad .$$

Therefore, the integral in (4.5) over $l \in f_\mathcal{T}^{(-1)}(d)$ can be sliced as an external integral over $l_0 \in [0, d_0]$ where $d_0 = \min\{d_k \mid k \in V\}$, and a product of internal integrals over $l^{(i)} \in (f^{(i)})^{-1}(d^{(i)} - (l_0))$ for each $i$, which gives

$$q_\alpha(\mathcal{T}, d) = \int_{l_0=0}^{d_0} \gamma_0 e^{-\gamma_0 l_0} \left[ \prod_{i=1}^{p} \int_{(f^{(i)})^{-1}(d^{(i)} - (l_0))} \prod_{j \in T_i} \gamma_j e^{-\gamma_j (l_j^{(i)})} dl^{(i)} \right] dl_0 \quad .$$

As we can see, the inner integral looks very similar to the initial integral in (4.5), and

indeed, we can finally write:

$$q_\alpha(\mathcal{T}, d) = \int_{l_0=0}^{d_0} \gamma_0 e^{-\gamma_0 l_0} \left[ \prod_{i=1}^{p} q_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0)) \right] dl_0 \quad . \qquad \square$$

The function $\xi_\alpha(\mathcal{T}, l|d)$ also satisfies an inductive relation, but its expression is slightly more complicated. From (4.5), a similar reasoning shows that the following inductive formula holds. For the root:

$$\xi_\alpha(\mathcal{T}, l_0|d) = \int_{l_0=0}^{d_0} \gamma_0 l_0 e^{-\gamma_0 l_0} \left[ \prod_{i=1}^{p} q_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0)) \right] dl_0 \quad , \qquad (4.7)$$

and for any node $j \in T \setminus \{0\}$, let $i \in \{1, \ldots, p\}$ be the unique child of the root such that $j \in T^{(i)}$, we have:

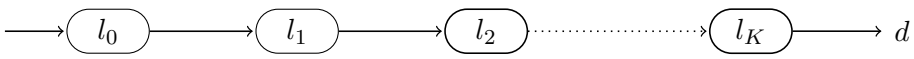
$$\xi_\alpha(\mathcal{T}, l_j|d) = \int_{l_0=0}^{d_0} \gamma_0 e^{-\gamma_0 l_0} \left[ \xi_\alpha(\mathcal{T}^{(i)}, l_j^{(i)}|d^{(i)} - l_0) \prod_{k \neq i} q_\alpha(\mathcal{T}^{(k)}, d^{(k)} - (l_0)) \right] dl_0 \quad . \quad (4.8)$$

Using this inductive formula, it is possible to deduce a recursive algorithm computing the expanded symbolic expression for the terms $q_\alpha$ and $\xi_\alpha$. However, we prefer to derive an alternative expression which, as we will see presently, is simpler.

### 4.4.4 More Examples

The following examples can be derived using the inductive expressions above. The first generalizes the case of a unary tree to any number of nodes, and the second is a simple tree for which the expanded expressions of $q_\alpha$ and $\xi_\alpha$ are already quite complicated. Here and below we recommend that the reader first focus on the expressions for $q_\alpha$.

**c) Unary Tree with K Nodes**

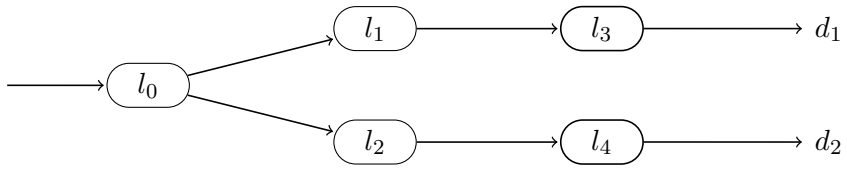



We have:

$$q_\alpha(\mathcal{T}, d) = \left( \prod_{j=1}^{K} \gamma_j \right) \sum_{j=1}^{K} \frac{e^{-\gamma_j d}}{\prod_{k \neq j} (\gamma_k - \gamma_j)} \quad \text{,and}$$

$$\xi_\alpha(\mathcal{T}, l_i|d) = \left( \prod_{j=1}^{K} \gamma_j \right) \left( \frac{e^{-\gamma_i d}}{\prod_{k \neq i} (\gamma_k - \gamma_i)} \left( d - \sum_{k \neq i} \frac{1}{\gamma_k - \gamma_i} \right) + \sum_{j \neq i} \frac{e^{-\gamma_j d}}{(\gamma_i - \gamma_j)^2 \prod_{\substack{k \neq i \\ k \neq j}} (\gamma_k - \gamma_j)} \right) \quad .$$

**d) One Root with two 2-Server Leaves**




As we did in Example **b)**, we introduce $d_0 := \min\{d_1, d_2\}$. We have:

$$
\begin{aligned}
&q_\alpha(\mathcal{T}, d) = \gamma_0\gamma_1\gamma_2\gamma_3\gamma_4 \\
&\Bigg( e^{-\gamma_0 d_0}\Bigg( \frac{e^{-\gamma_1(d_1-d_0)}e^{-\gamma_2(d_2-d_0)}}{(\gamma_1+\gamma_2-\gamma_0)(\gamma_3-\gamma_1)(\gamma_4-\gamma_2)} + \frac{e^{-\gamma_1(d_1-d_0)}e^{-\gamma_4(d_2-d_0)}}{(\gamma_1+\gamma_4-\gamma_0)(\gamma_3-\gamma_1)(\gamma_2-\gamma_4)} \\
&\qquad + \frac{e^{-\gamma_3(d_1-d_0)}e^{-\gamma_2(d_2-d_0)}}{(\gamma_3+\gamma_2-\gamma_0)(\gamma_1-\gamma_3)(\gamma_4-\gamma_2)} + \frac{e^{-\gamma_3(d_1-d_0)}e^{-\gamma_4(d_2-d_0)}}{(\gamma_3+\gamma_4-\gamma_0)(\gamma_1-\gamma_3)(\gamma_2-\gamma_4)}\Bigg) \\
&\quad + \frac{e^{-\gamma_1 d_1}e^{-\gamma_2 d_2}}{(\gamma_0-\gamma_1-\gamma_2)(\gamma_3-\gamma_1)(\gamma_4-\gamma_2)} + \frac{e^{-\gamma_1 d_1}e^{-\gamma_4 d_2}}{(\gamma_0-\gamma_1-\gamma_4)(\gamma_3-\gamma_1)(\gamma_2-\gamma_4)} \\
&\quad + \frac{e^{-\gamma_3 d_1}e^{-\gamma_2 d_2}}{(\gamma_0-\gamma_3-\gamma_2)(\gamma_1-\gamma_3)(\gamma_4-\gamma_2)} + \frac{e^{-\gamma_3 d_1}e^{-\gamma_4 d_2}}{(\gamma_0-\gamma_3-\gamma_4)(\gamma_1-\gamma_3)(\gamma_2-\gamma_4)}\Bigg) \quad ,
\end{aligned}
$$

and $\xi_\alpha(l_0|d) = \gamma_0\gamma_1\gamma_2\gamma_3\gamma_4$

$$
\begin{aligned}
&\Bigg( e^{-\gamma_0 d_0}\Bigg( \frac{e^{-\gamma_1(d_1-d_0)}e^{-\gamma_2(d_2-d_0)}}{(\gamma_1+\gamma_2-\gamma_0)(\gamma_3-\gamma_1)(\gamma_4-\gamma_2)}\Big(d_0 - \frac{1}{(\gamma_1+\gamma_2-\gamma_0)}\Big) \\
&\qquad + \frac{e^{-\gamma_1(d_1-d_0)}e^{-\gamma_4(d_2-d_0)}}{(\gamma_1+\gamma_4-\gamma_0)(\gamma_3-\gamma_1)(\gamma_2-\gamma_4)}\Big(d_0 - \frac{1}{(\gamma_1+\gamma_4-\gamma_0)}\Big) \\
&\qquad + \frac{e^{-\gamma_3(d_1-d_0)}e^{-\gamma_2(d_2-d_0)}}{(\gamma_3+\gamma_2-\gamma_0)(\gamma_1-\gamma_3)(\gamma_4-\gamma_2)}\Big(d_0 - \frac{1}{(\gamma_3+\gamma_2-\gamma_0)}\Big) \\
&\qquad + \frac{e^{-\gamma_3(d_1-d_0)}e^{-\gamma_4(d_2-d_0)}}{(\gamma_3+\gamma_4-\gamma_0)(\gamma_1-\gamma_3)(\gamma_2-\gamma_4)}\Big(d_0 - \frac{1}{(\gamma_3+\gamma_4-\gamma_0)}\Big)\Bigg) \\
&\quad + \frac{e^{-\gamma_1 d_1}e^{-\gamma_2 d_2}}{(\gamma_0-\gamma_1-\gamma_2)^2(\gamma_3-\gamma_1)(\gamma_4-\gamma_2)} + \frac{e^{-\gamma_1 d_1}e^{-\gamma_4 d_2}}{(\gamma_0-\gamma_1-\gamma_4)^2(\gamma_3-\gamma_1)(\gamma_2-\gamma_4)} \\
&\quad + \frac{e^{-\gamma_3 d_1}e^{-\gamma_2 d_2}}{(\gamma_0-\gamma_3-\gamma_2)^2(\gamma_1-\gamma_3)(\gamma_4-\gamma_2)} + \frac{e^{-\gamma_3 d_1}e^{-\gamma_4 d_2}}{(\gamma_0-\gamma_3-\gamma_4)^2(\gamma_1-\gamma_3)(\gamma_2-\gamma_4)}\Bigg) \quad ,
\end{aligned}
$$

and $\xi_\alpha(l_1|d) = \gamma_0\gamma_1\gamma_2\gamma_3\gamma_4$

$$\Bigg( e^{-\gamma_0 d_0} \Bigg( \frac{e^{-\gamma_1(d_1-d_0)} e^{-\gamma_2(d_2-d_0)}}{(\gamma_1+\gamma_2-\gamma_0)(\gamma_3-\gamma_1)(\gamma_4-\gamma_2)} \Big( d_1 - d_0 - \frac{1}{(\gamma_0-\gamma_1-\gamma_2)} - \frac{1}{(\gamma_3-\gamma_1)} \Big)$$
$$+ \frac{e^{-\gamma_1(d_1-d_0)} e^{-\gamma_4(d_2-d_0)}}{(\gamma_1+\gamma_4-\gamma_0)(\gamma_3-\gamma_1)(\gamma_2-\gamma_4)} \Big( d_1 - d_0 - \frac{1}{(\gamma_0-\gamma_1-\gamma_4)} - \frac{1}{(\gamma_3-\gamma_1)} \Big)$$
$$+ \frac{e^{-\gamma_3(d_1-d_0)} e^{-\gamma_2(d_2-d_0)}}{(\gamma_3+\gamma_2-\gamma_0)(\gamma_1-\gamma_3)^2(\gamma_4-\gamma_2)} + \frac{e^{-\gamma_3(d_1-d_0)} e^{-\gamma_4(d_2-d_0)}}{(\gamma_3+\gamma_4-\gamma_0)(\gamma_1-\gamma_3)^2(\gamma_2-\gamma_4)} \Bigg)$$
$$+ \frac{e^{-\gamma_1 d_1} e^{-\gamma_2 d_2}}{(\gamma_0-\gamma_1-\gamma_2)(\gamma_3-\gamma_1)(\gamma_4-\gamma_2)} \Big( d_1 - \frac{1}{(\gamma_0-\gamma_1-\gamma_2)} - \frac{1}{(\gamma_3-\gamma_1)} \Big)$$
$$+ \frac{e^{-\gamma_1 d_1} e^{-\gamma_4 d_2}}{(\gamma_0-\gamma_1-\gamma_4)(\gamma_3-\gamma_1)(\gamma_2-\gamma_4)} \Big( d_1 - \frac{1}{(\gamma_0-\gamma_1-\gamma_4)} - \frac{1}{(\gamma_3-\gamma_1)} \Big)$$
$$+ \frac{e^{-\gamma_3 d_1} e^{-\gamma_2 d_2}}{(\gamma_0-\gamma_3-\gamma_2)(\gamma_1-\gamma_3)^2(\gamma_4-\gamma_2)} + \frac{e^{-\gamma_3 d_1} e^{-\gamma_4 d_2}}{(\gamma_0-\gamma_3-\gamma_4)(\gamma_1-\gamma_3)^2(\gamma_2-\gamma_4)} \Bigg) \quad .$$

$\xi_\alpha(l_2|d)$,$\xi_\alpha(l_3|d)$ and $\xi_\alpha(l_4|d)$ can be deduced by symmetry.

### 4.4.5 Explicit Expression

One could use the previous inductive formulae with algebraic computation to generate the expressions of $q_\alpha$ and $\xi_\alpha$. However such a method is not efficient, since terms have to be merged for optimization. For instance, the inductive formula of $q_\alpha$ applied to Example **c)** would lead before simplification to a sum of $2^K$ terms, while the simplified expression has only $K$ summands. We therefore give here explicit, already simplified formulae.

#### Vocabulary for Tree Combinatorics

Example **d)** shows that the formulae for $q_\alpha$ and $\xi_\alpha$ can be expressed as a sum of terms with a distinct structure. These in fact correspond to particular 'slices' or 'cuts' of the tree. In this section we define cuts and related nomenclature (illustrated in Figure 4.2) which will be subsequently used to provide simplified symbolic expressions for $q_\alpha$ and $\xi_\alpha$.

The following definitions are given in the context of a *tree*, but they extend naturally to a *forest*, that is a set of trees.

**Definition 4.4.2.** A *cut* of a tree is a maximal unordered set of nodes for the order $\prec$ defined above. In other words, $C$ is a cut of a tree $\mathcal{T}$ if it satisfies:

**1)** $\forall\, i, j \in C,\ i \not\prec j$ and $j \not\prec i$.

**2)** $\forall\, i \in T \setminus C,\ \exists\, j \in C,\ i \prec j$ or $j \prec i$.

**Example 4.4.1:** In Example **d)** above, the tree has five possible cuts which are: $\{0\}$, $\{1;2\}$, $\{1;4\}$, $\{3;2\}$ and $\{3;4\}$.

**Definition 4.4.3.** We call *branch* every maximal sequence of nodes $(i_1, \ldots, i_n)$ such that for all $k \in \{1, \ldots, n-1\}$, the node $i_{k+1}$ is the unique child of $i_k$.

Every tree can then be visualized as a tree of *branches*, each having at least two children. However, a *cut* as defined above will still be a set of *nodes*, not *branches*. See Figure 4.2 for an example.

Finally, as shown in the figure, we will talk about the "past", the "present" and the "future" of a cut as follows: The "present" of a cut is the set of all nodes in the branches intersected by the cut. The "past" is the set of their ancestors in different branches, and the "future" the set of their descendants in different branches. More formally, we will adopt the following notations:

**Definition 4.4.4.** For each pair of nodes $(i, j)$ in $T$, we write:

**a)** $i \sim j$ if $i$ and $j$ belongs to the same branch and we say that $j$ belongs to the present of $i$ and $i$ belongs to the present of $j$.

**b)** $i \ll j$ if $i \prec j$ and $i \nsim j$, and we say that $i$ belongs to the past of $j$ and $j$ belongs to the future of $i$.

The past, (resp. present, future) of a node will be the set of all nodes belonging to the past (resp. present, future) of this node, and by extension, the past (resp. present, future) of a cut will be the set of all the nodes belonging to the past (resp. present, future) of at least one of the nodes of the cut. We will denote these by **past**$(i)$, **present**$(i)$, **future**$(i)$ for a node $i$ and **Past**$(C)$, **Present**$(C)$, **Future**$(C)$ for a cut $C$. It is important not to confuse *nodes* and *branches*.

*Remark.* The past, present and future of a cut forms a partition of $T$.

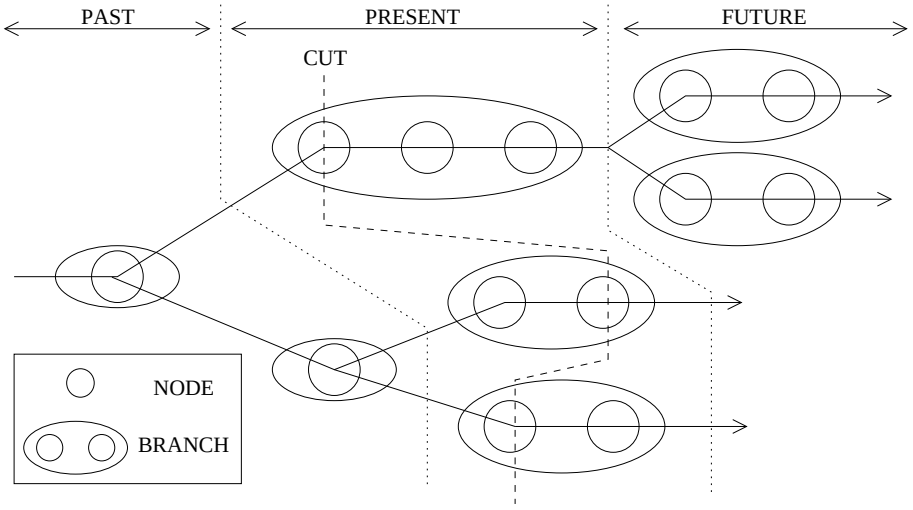



Figure 4.2: A tree with its branches and one of its cuts, with the corresponding past, present, and future of the cut.

Finally, we extend the vector $d = (d_j)_{j \in V} \in \mathbb{R}^V$ to $d = (d_j)_{j \in T} \in \mathbb{R}^T$ by introducing for each $j$ in $T \setminus V$, $d_j := \min \{d_k \mid k \in V \text{ and } j \prec k\}$.

**The Explicit Form of $q_\alpha$**

In this section we use the cut vocabulary to define an expression for $q_\alpha$ which is not only closer to closed form, but is also more compact and more efficient to evaluate than that produced by the inductive formula. The validity of this formula is proved in the appendix.

For any fixed tree $\mathcal{T}$ and delay $d$, we have $q_\alpha(\mathcal{T}, d) = \Gamma_T \sum_{C \text{ cut of } \mathcal{T}} h_\alpha(\mathcal{T}, d, C)$, where $\Gamma_T := \prod_{j \in T} \gamma_j$ and where each term $h_\alpha(\mathcal{T}, d, C)$ can be expressed as a product of three factors:

$$h_\alpha(\mathcal{T}, d, C) := r(C)s(C)t(C) \quad ,$$

where $r(C)$ depends only on the cut $C$ and its past, $s(C)$ depends only on $C$ and its present, and $t(C)$ depends only on $C$ and its future.

**i) Past and Present** The factors $r(C)$ and $s(c)$ are given by

$$r(C) := \prod_{k \in \mathbf{Past}(C)} \frac{1}{\gamma_k - \sum\limits_{\substack{j \in C \\ k \ll j}} \gamma_j} \quad \text{and} \quad s(C) := \prod_{j \in C} \frac{e^{-\gamma_j d_j}}{\prod\limits_{\substack{k \sim j \\ k \neq j}} (\gamma_k - \gamma_j)} \quad .$$

**ii) Future** The factor $t(C) = t(\mathcal{T}, d, C)$ is more complicated as it involves a recursion. To describe it, we regard $h_\alpha(\mathcal{T}, d, C)$ as functions of all its arguments to allow it to apply to subtrees with modified delays, and also extend its definition from a tree $\mathcal{T}$ to a forest $\mathcal{F}$. The set of nodes of a forest $\mathcal{F}$ is denoted by $F$.

If the future of each cut $C$ of $\mathcal{T}$ is empty, *i.e.* if the tree $\mathcal{T}$ is reduced to a single branch, we have $t(\mathcal{T}, d, C) = 1$. Recursively, we can then define:

$$t(\mathcal{T}, d, C) := \prod_{j \in C} t_j(\mathcal{T}, d, C) \quad ,$$

where

$$t_j(\mathcal{T}, d, C) := \sum_{C_j \text{ cut of } \mathcal{F}_j} \frac{h_\alpha(\mathcal{F}_j, d - (d_j), C_j)}{\left(\sum_{k \in C_j} \gamma_k\right) - \gamma_j} \quad ,$$

where $\mathcal{F}_j$ is the subforest of $\mathcal{T}$ containing all the nodes belonging to the future of j, and therefore $F_j := \{k \in T | j \ll k\}$, where $d - (d_j)$ is the vector $(d_k - d_j)_{k \in F_j}$.

We can interpret $d - (d_j)$ as the best information we have about the delay between $j$ and the leaves, since we know only that the delay between the root and $j$ has to be smaller than any delay between the root and the leaves in the future of $j$.

The formula $q_\alpha(\mathcal{T}, d)$ is now entirely defined. One can verify that it gives the correct expressions for the examples in the paper.

**The Explicit Form of $\xi_\alpha(l|d)$**

As we can see in the previous examples, the term $\xi_\alpha(l_i|d)$ has globally the same structure as $q_\alpha(d)$ with additional factors. Therefore it is relatively easy to modify any algorithm computing $q_\alpha(d)$ to compute $\xi_\alpha(l_i|d)$.

For any non-fixed tree $\mathcal{T}$ and any node $i$ in $T$, we have:

$$\xi_\alpha(l_i|d) = \Gamma_T \sum_{C \text{ cut of } \mathcal{T}} h^i_\alpha(\mathcal{T}, d, C, 0) \quad ,$$

where for any real number $x$:

$$h^i_\alpha(\mathcal{T}, d, C, x) := r^i(C) s^i(C) t^i(\mathcal{T}, d, C, x) \quad ,$$

where $r^i(C)$ (resp. $s^i(C)$) depends only from the cut $C$ and its past (resp. present), and where $t^i(\mathcal{T}, d, C, x)$ involves a recursion.

**a) Past and Present**

$$r^i(C) := \prod_{k \in \textbf{Past}(C)} \frac{1}{(\gamma_k - \sum\limits_{\substack{j \in C \\ k \ll j}} \gamma_j)^{1+\delta^i_k}} \quad \text{and} \quad s^i(C) := \prod_{j \in C} \frac{e^{-\gamma_j d_j}}{\prod\limits_{\substack{k \sim j \\ k \neq j}} (\gamma_k - \gamma_j)^{1+\delta^i_k}} \quad ,$$

where $\delta^i_k = \begin{cases} 1 \text{ if } k = i \\ 0 \text{ else} \end{cases}$ is the Kronecker delta.

**b) Future**

As for $t(\mathcal{T}, d, C)$, the definition of $t^i(\mathcal{T}, d, C, x)$ induce a recursion with the whole formula. First we introduce $\rho_i(C) := \sum\limits_{k \in \textbf{past}(i)} \frac{1}{\gamma_k - \sum\limits_{\substack{j \in C \\ k \ll j}} \gamma_j}$ and $\sigma_i(C) := \sum\limits_{\substack{k \sim i \\ k \neq i}} \frac{1}{\gamma_k - \gamma_i}$.
These are terms corresponding respectively to the past and the present of node $i$. Now

$t^i(\mathcal{T}, d, C, x) := \prod_{j \in C} t^i_j(\mathcal{T}, d, C, x)$, where:

$$t^i_j(\mathcal{T}, d, C, x) :=$$
$$\begin{cases} \Big(d_i - \sigma_i(C) - \rho_i(C) - x\Big) & \text{if } \textbf{future}(j) = \emptyset \text{ and } j = i \\ 1 & \text{if } \textbf{future}(j) = \emptyset \text{ and } j \neq i \\ \displaystyle\sum_{C_j \text{ cut of } \mathcal{F}_j} \frac{h^i_\alpha(\mathcal{F}_j, d - (d_j), C_j, 0)}{\big(\sum_{k \in C_j} \gamma_k\big) - \gamma_j} u^i(\mathcal{T}, d, C, C_j, x) & \text{if } \textbf{future}(j) \neq \emptyset \text{ and } j = i \\ \displaystyle\sum_{C_j \text{ cut of } \mathcal{F}_j} \frac{h^i_\alpha(\mathcal{F}_j, d - (d_j), C_j, \tau_j(C_j))}{\big(\sum_{k \in C_j} \gamma_k\big) - \gamma_j} & \text{if } \textbf{future}(j) \neq \emptyset \text{ and } j \neq i, \end{cases}$$

where $u^i(\mathcal{T}, d, C, C_j, x) := \Big(d_i - \sigma_i(C) - \rho_i(C) - x + \tau_i(C_j)\Big)$, and where for any node $j$ and cut $C$, $\tau_j(C) := \dfrac{1}{\gamma_j - \sum_{k \in C} \gamma_k}$.

*Remark.* It is important to remember in which tree each term is computed. When it is not specified, it is implicitly the tree called $\mathcal{T}$ on the current level of recursion.

We notice that the term $x$ is used only in the cases where $j = i$. Therefore, the term $\tau_j(C_j)$ in $h^i_\alpha(\mathcal{F}_j, d - (d_j), C_j, \tau_j(C_j))$ above is used only when $i \in C_j$.

In order to understand the different cases in the definition of $t^i_j(\mathcal{T}, d, C, x)$, we can have a look at the example **d)**. In this example, the term $(d_0 - \frac{1}{\gamma_1 + \gamma_2 - \gamma_0})$ in $\xi_\alpha(l_0|d)$ comes from $(d_i + \tau_i(C_j))$ in $u^i(C, C_j, x)$, while the term $(d_1 - d_0 - \frac{1}{\gamma_0 - \gamma_1 - \gamma_2} - \frac{1}{\gamma_3 - \gamma_1})$ in $\xi_\alpha(l_1|d)$ comes from $(d_i - x - \sigma_i(C))$ with $x = \tau_j(C_j)$ and $C = C_j$ (recursive call), and the term $(d_1 - \frac{1}{\gamma_0 - \gamma_1 - \gamma_2} - \frac{1}{\gamma_3 - \gamma_1})$ comes from $(d_i - \rho_i(C) - \sigma_i(C))$.

#### Alternative Informal Description

A less formal way to describe $\xi_\alpha(l_i|d)$ is to consider each term in the expanded expression of $q_\alpha(d)$, and each time a $\frac{e^{-\gamma_i d}}{coef}$ appears, multiply it by $(d - coef')$, where $coef'$ is the sum of each multiplicative factor appearing in $coef$ and containing $\gamma_i$ in its expression, and each factor being multiplied by $(-1)$ when $\gamma_i$ appears in it with a positive sign. Further, when a term in the expanded expression of $q_\alpha(d)$ is of the form $\frac{e^{-\gamma_{j_1} d} \ldots e^{-\gamma_{j_n} d}}{coef}$, replace $coef$ with $coef''$, being equal to $coef$ where all the factors containing $\gamma_i$ are squared.

### 4.4.6 Implementation

We can finally compute $\mathbf{E}_\alpha(l_i|d) = \dfrac{\xi_\alpha(l_i|d)}{q_\alpha(d)}$. Our implementation is in the programming language *Objective Caml*. Only the standard packages of the language were used.

We need to compute $\varphi(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{E}_\alpha(l|d(i))$ at each step of the EM algorithm. Therefore, the formula $\mathbf{E}_\alpha(l|d)$ has to be efficiently calculated for a fixed tree $\mathcal{T}$ and fixed $\alpha$, but for $N$ distinct values of $d$, $N$ being the number of probes used in the experiment which may not be fixed in advance. It follows that the best way to compute this formula efficiently is to generate the symbolic expression of $\mathbf{E}_\alpha(l|d)$ with $\alpha$ known and $d$ unknown

parameters, to simplify it as most as possible and then to compute it for each of the $N$ values $d(i)$.

Efficiency is further improved if $d = (d_j)$ is precomputed for all $j \in T$, and also if the differences $d_i - d_j$ are precomputed and kept easy to access, since they appear frequently and keep the same value at each different step.

Our program generated the symbolic expression of $\mathbb{E}_\alpha(l|d)$ as a symbolic tree. The factors depending only on $\alpha$ were evaluated and simplified during the generation of the tree. The tree was then reduced as much as possible and was finally evaluated for each $d(i)$ using the precomputed matrix $M(i)$.

A first sanity check for program correctness, which is easy to perform, is to exploit the relation $d_k = \mathbb{E}_\alpha(d_k|d) = \sum_{j \preceq k} \mathbb{E}_\alpha(l_j|d)$ for all $k$ in $V$. If the program is correct, this has to be verified for any value of $d$ and any trees $\mathcal{T}$.

### 4.4.7 Size of the expression and Complexity of the EM step

We will only consider the size of the explicit expression of $q_\alpha$ in the particular cases of a unary tree and of a binary tree. In the former case, the size is clearly linear, while in the latter it is exponential in the number of nodes, and thus doubly exponential in the height of the tree. Here, we measure the size of the expression in the number of exponentials appearing in it after reduction, since all the other factors, especially those involving $\gamma$, can be precomputed.

For a unary tree with $K$ nodes, the number of exponentials is exactly $K$, and the size is then linear. For a binary tree of height $h{+}1$, the size $S$ is given by $S(h{+}1) = 2{*}S(h)^2$, since the terms of the two subtrees of height $h$ are multiplied (giving $S(h)^2$) and are integrated, giving twice as many terms. Since $S(1) = 1$, we deduce that $S(h) = 2^{2^{h-1}-1}$. Since the number of node in a binary tree of height $h$ is $2^{h+1} - 1$, the size is then exponential in the number of nodes.

We get the complexity of one step of EM by multiplication of this size by the number of nodes in the tree and the length of data, since we need to compute $\mathbb{E}_\alpha(l_j|d(i))$ for all $j$ in $T$ and all $i$ in $\{1, \ldots, N\}$. It is, however, possible to factorize the computation of the exponentials shared by the expressions, but the number of multiplications will still be the same.

Finally, the number of steps to convergence is likely to grow with the number of nodes as well, which increase further the global complexity of the EM algorithm. This motivates the speed-up technique presented in Section 4.6.

## 4.5 Results

We conducted series of experiments on different kinds of trees. The data were generated by simulating random delays in a tree using the known ground truth. Except in Section 4.5.3, each experiment was conducted on a data set of $N = 10^4$ samples and repeated 200 times.

### 4.5.1 Unary Tree Case

This case was studied in the previous chapter, where two and three nodes cases were tested.

**a) Two Node Case:** In the simple case of a tree with two nodes, we have some additional results on the convergence of the E-M sequence. Property 4.3.1 becomes $\hat{\alpha}_1^{(k+1)} + \hat{\alpha}_2^{(k+1)} = \bar{d}$. This relation implies that there is only one unknown value, and thanks to this, it is possible to prove the following result by using the intermediate value theorem.

We recall lemma 3.5.3, which is specific of the two node case:

**Lemma 4.5.1.** *In the two node case, the sequence* $(\hat{\alpha}_1^{(k)}, \hat{\alpha}_2^{(k)})$ *converges to a finite limit which is a solution of the likelihood equation.*

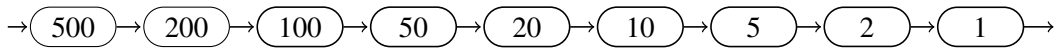
The proof uses the fact that, since (for all $k \geq 1$) $\hat{\alpha}_1^{(k)} + \hat{\alpha}_2^{(k)} = \bar{d}$, the likelihood $L_d(\hat{\alpha}^{(k)})$ can be expressed as a function of $\hat{\alpha}_1^{(k)}$ alone. Therefore, the proof cannot be generalized to more than two nodes. The following table gives some results obtained in this case.

| Gr. truth | Mean | 10% percentile | 90% percentile | Variance$^{1/2}$/Mean |
|---|---|---|---|---|
| $(1.1, 1)$ | $(1.114, 0.987)$ | $(1.044, 0.881)$ | $(1.220, 1.057)$ | $(0.0644, 0.0725)$ |
| $(2, 1)$ | $(1.999, 1.003)$ | $(1.933, 0.946)$ | $(2.063, 1.063)$ | $(0.0244, 0.0446)$ |
| $(10, 1)$ | $(10.003, 1.003)$ | $(9.866, 0.942)$ | $(10.148, 1.070)$ | $(0.0115, 0.0501)$ |
| $(100, 1)$ | $(100.044, 1.015)$ | $(98.782, 0.827)$ | $(101.368, 1.214)$ | $(0.0104, 0.1514)$ |

Table 4.1: Experimental results of the EM estimator $(\hat{\alpha}_1, \hat{\alpha}_2)$ for various ground truths in the 2-node case.

**b) Nine Nodes Case** $(U9)$

Table 4.2 shows the results for the following unary tree with 9 nodes.




| Gr. truth | 500 | 200 | 100 | 50 | 20 | 10 | 5 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| Mean | 498.42 | 194.98 | 99.59 | 49.88 | 26.10 | 10.55 | 5.38 | 4.25 | 3.79 |
| 10%-tile | 481.47 | 150.08 | 48.31 | 13.93 | 5.78 | 1.54 | 0.099 | 0.087 | 0.077 |
| 90%-tile | 512.30 | 236.78 | 155.35 | 79.58 | 56.20 | 24.94 | 12.19 | 10.86 | 9.99 |
| $\sigma$/Mean | 0.025 | 0.175 | 0.428 | 0.486 | 0.769 | 0.990 | 0.897 | 0.904 | 0.974 |

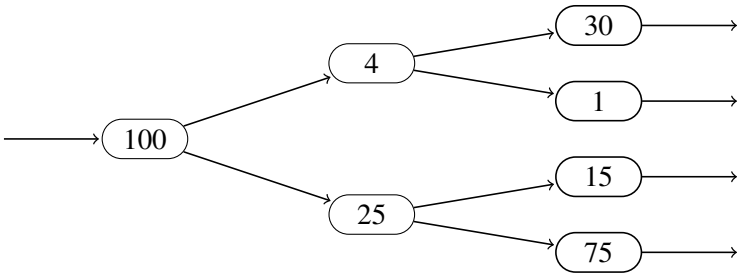
Table 4.2: Experimental results obtained for a unary tree with 9-nodes.

The main difficulty for estimating such unary trees comes from the fact that we get the same information for all the nodes. In particular, the estimation can only give the values of the mean delays modulo an unknown permutation. We will also see in Section 4.6 that the convergence of EM for this tree is very slow.

### 4.5.2 General Case

We present here some results obtained for other different trees.

**a) Binary Tree of Height 3** $(B1H3)$

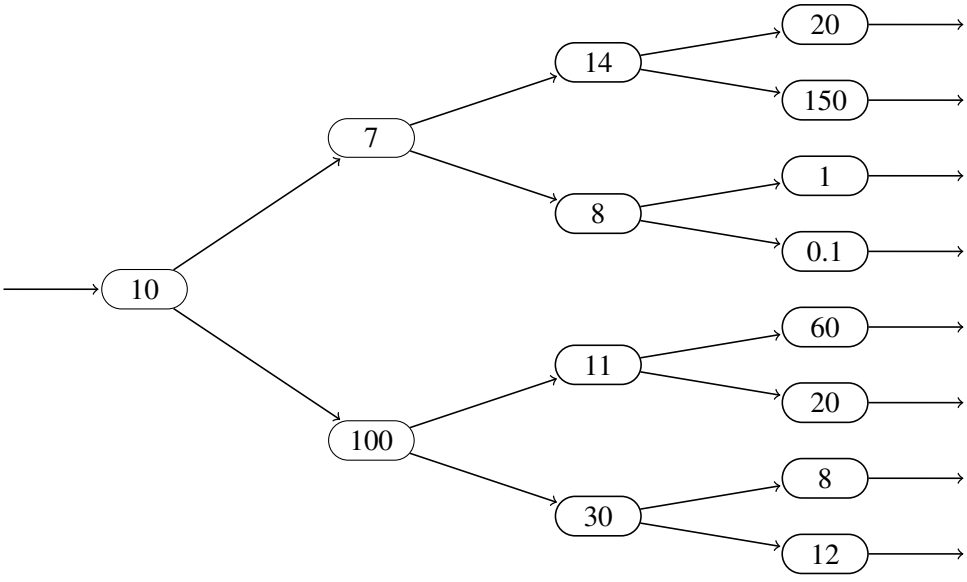



| Ground truth | 100 | 4 | 30 | 1 | 25 | 15 | 75 |
|---|---|---|---|---|---|---|---|
| Mean | 100.03 | 4.01 | 29.99 | 0.9999 | 25.007 | 14.996 | 74.995 |
| $10\%$ percentile | 99.62 | 3.95 | 29.86 | 0.98 | 24.85 | 14.88 | 74.70 |
| $90\%$ percentile | 100.42 | 4.04 | 30.12 | 1.02 | 25.14 | 15.11 | 75.35 |
| $\text{Var}^{1/2}/\text{Mean}$ | 0.0032 | 0.0092 | 0.0031 | 0.012 | 0.0045 | 0.0060 | 0.0034 |

Table 4.3: Experimental results obtained for a binary tree of height 3.

In the case of a binary tree, the estimation are accurate too. with again a slight bias on the smallest values. The estimations are in a sense easier for this tree than for the unary ones because we get more information and also because each node can be discriminated from the others, while in the unary trees all the nodes are equivalent.

**b) Binary Tree of Height 4** $(B1H4)$



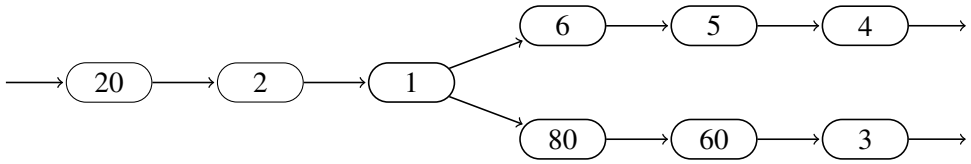
| Ground truth | 10 | 7 | 14 | 20 | 150 | 8 | 1 | 0.1 |
|---|---|---|---|---|---|---|---|---|
| Mean | 10.01 | 6.99 | 13.99 | 20.03 | 149.95 | 7.999 | 1.0004 | 0.0999 |
| $10\%$ percentile | 9.67 | 6.66 | 13.52 | 19.50 | 147.77 | 7.80 | 0.99 | 0.096 |
| $90\%$ percentile | 10.35 | 8.14 | 15.32 | 20.98 | 152.17 | 8.29 | 1.71 | 0.28 |
| Variance$^{1/2}$/Mean | 0.03 | 0.04 | 0.03 | 0.02 | 0.01 | 0.02 | 0.01 | 0.01 |

| Ground truth | | 100 | 11 | 60 | 20 | 30 | 8 | 12 |
|---|---|---|---|---|---|---|---|---|
| Mean | | 100.09 | 11.30 | 60.13 | 20.01 | 30.03 | 8.002 | 12.01 |
| $10\%$ percentile | | 99.83 | 10.56 | 59.8 | 19.59 | 29.58 | 7.86 | 11.85 |
| $90\%$ percentile | | 102.16 | 13.13 | 61.47 | 21.31 | 31.65 | 10.27 | 14.19 |
| Variance$^{1/2}$/Mean | | 0.01 | 0.03 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |

Table 4.4: Experimental results obtained for a binary tree of height 4.

For this tree, the estimations are still very good. The difficulty for bigger binary trees arise from the complexity of the EM step computation rather than the accuracy of the estimation. We made some simulation on a tree of height $5$ and obtained good estimations too, but they were very long to compute.

**c) Tree With Branches** $(B3H2)$




| Ground truth | 20 | 2 | 1 | 6 | 5 | 4 | 80 | 60 | 3 |
|---|---|---|---|---|---|---|---|---|---|
| Mean | 19.86 | 2.63 | 1.79 | 6.30 | 4.79 | 2.63 | 78.80 | 61.34 | 1.63 |
| $10\%$ percentile | 19.40 | 1.50 | 0.68 | 5.18 | 3.09 | 0.62 | 70.31 | 54.70 | 0.0052 |
| $90\%$ percentile | 20.35 | 3.92 | 2.98 | 7.58 | 6.03 | 4.86 | 84.89 | 70.28 | 3.74 |
| Var$^{1/2}$/Mean | 0.021 | 0.36 | 0.48 | 0.15 | 0.24 | 0.59 | 0.088 | 0.012 | 0.096 |

Table 4.5: Experimental results obtained with the tree above.

Trees such as the last one, with several parallel branches, are the hardest to estimate because they combine both difficulties seen in unary and binary trees: A difficult estimation in each branch where we have the same information for all nodes, and a high complexity of the EM step because of the multiple parallel subtrees.

### 4.5.3 Speed of convergence

All previous results were conducted with sample size $N = 10^4$. Figure 4.3 studies the speed of convergence of the maximum likelihood estimator with respect to $N$. We present results

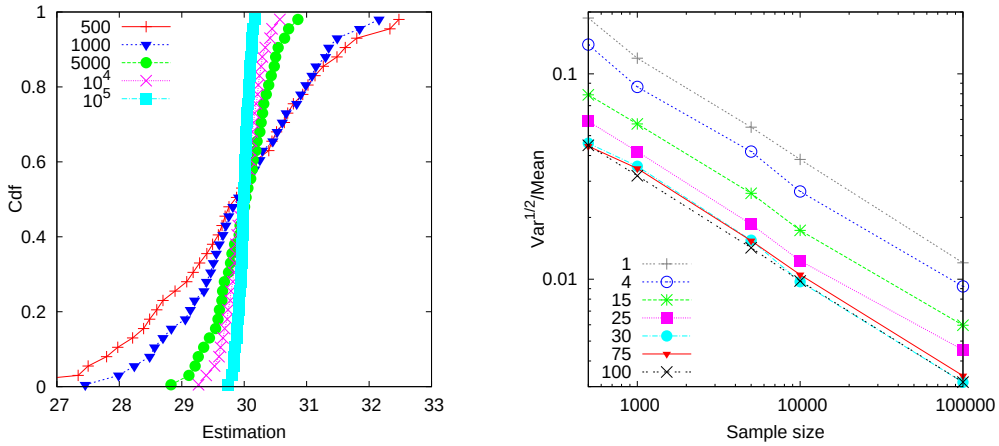



Figure 4.3: Left plot: the cumulative distribution function of the maximum likelihood estimator of node 30 for different sample sizes. The right-side shows the relative standard error for all nodes on the same tree B1H3, depending on the sample size.

only for the tree B1H3, but results are similar for the other nodes and trees. One might notice that smaller sample size can lead to good results: with as few as 1000 probes, 90% of the experiments estimate node 30 with less than 10% error. On the right plot, the relative standard error (ie. the square root of the variance, divided by the mean) are parallel lines of slope $-\frac{1}{2}$, which means that the standard error decreases as $\frac{1}{\sqrt{N}}$ and the variance as $\frac{1}{N}$.

### 4.5.4 Comparison to the Least Squares Method

In [LMN07], Lawrence *et al.* discard MLE based approaches due to their computation time in favour of moment based methods using least squares. Table 4.6 presents the results obtained by their method for 200 independent experiments with the same sample size $N = 10^4$, for the tree B1H3. This shall be compared with the results of MLE in Table 4.3. Other trees lead to similar results. As expected, the MLE approach yields better results, especially for nodes that have a small delay. However, the main advantage of moment based methods is their speed. The simulation of the 200 experiments took only about one minute for the least squares approach, whereas our algorithm needed 45 minutes. This difference increases for larger trees.

The complexity of an estimation technique can be expressed in two ways: the number of independent probes needed to reach a given precision, and the time needed to compute the estimator based on those probes. The relative interest of each method will depend on which of these two steps is the most crucial for the specific application considered.

### 4.5.5 Resilience to measurement noise and imperfect models

They are many ways to introduce model or measurement errors: a full chapter could be written on this topic. We will present only one case, which we hope is representative. Table 4.7 presents the results of the accelerated EM algorithm, when each end-to-end delay

| Ground truth | 100 | 4 | 30 | 1 | 25 | 15 | 75 |
|---|---|---|---|---|---|---|---|
| Mean | 99.98 | 4.04 | 29.93 | 0.93 | 24.77 | 15.18 | 75.07 |
| $10\%$ percentile | 98.22 | 2.42 | 28.73 | -0.19 | 22.58 | 13.92 | 73.54 |
| $90\%$ percentile | 101.91 | 5.46 | 31.20 | 2.21 | 26.77 | 16.31 | 76.65 |
| $\text{Var}^{1/2}/\text{Mean}$ | 0.014 | 0.32 | 0.033 | 1.009 | 0.060 | 0.062 | 0.017 |

Table 4.6: Experimental results obtained for B1H3, using the least squares method from [LMN07].

was sampled according to the model distribution, then multiplied by a i.i.d. uniform value between 0.95 and 1.05. The results are worse: the standard error is about three times higher for most nodes. However, the errors stay in a reasonable level, which might indicate that the algorithm is resilient to errors.

| Ground truth | 100 | 4 | 30 | 1 | 25 | 15 | 75 |
|---|---|---|---|---|---|---|---|
| Mean | 98.1 | 3.6 | 32.3 | 3.3 | 25.8 | 16.1 | 76.1 |
| $10\%$ percentile | 96.7 | 3.4 | 31.9 | 3.2 | 25.4 | 15.8 | 75.2 |
| $90\%$ percentile | 99.3 | 3.9 | 32.8 | 3.5 | 26.2 | 16.5 | 77.1 |
| $\text{Var}^{1/2}/\text{Mean}$ | 0.01 | 0.05 | 0.01 | 0.04 | 0.01 | 0.02 | 0.01 |

Table 4.7: Experimental results obtained for the tree B1H3, with imperfect measurements.

## 4.6 Steered Jumping for EM

In a number of cases, especially when the tree has long branches, the number of steps before converging to a fixed point can be very large, and since the complexity of each step is proportional to the length of data, the EM algorithm becomes very slow when $N$ becomes large. The complexity of each step rises also very quickly with the size of the tree, since the growth is quadratic in the number of nodes for an unary tree, but can be exponential for a binary tree (see Section 4.4.7). It is therefore important for this problem to significantly improve the convergence speed of the EM algorithm. We present a novel such method below.

### 4.6.1 Analysis of the iteration

We first illustrate some characteristics of the iteration through two examples.

**Example One** Consider a unary tree with ground truth $\alpha = (0.1, 1, 10)$. From Property 4.3.1, $\hat{\alpha}^{(k)} = (\hat{\alpha}_1^{(k)}, \hat{\alpha}_2^{(k)}, \hat{\alpha}_3^{(k)})$ obeys $\hat{\alpha}_1^{(k)} + \hat{\alpha}_2^{(k)} + \hat{\alpha}_3^{(k)} = \bar{d}$, so that the system has only two independent variables. We therefore plot the trajectories of the EM algorithm in a $(\hat{\alpha}_2^{(k)}, \hat{\alpha}_3^{(k)})$ plot.

Figure 4.4 shows some sequences of iterations of the algorithm from different initial conditions. We see that the trajectories all converge towards the same point, $\hat{\alpha} = (0.83, 0.83, 8.07)$. During this experiment, the mean delay was $\bar{d} = 9.73$, and Property 4.3.1

was respected. We also observe that the trajectories seems to converge quickly toward a straight line of equation $x = 1.7 - y$, and once on this line, the steps becomes very small and the convergence is much slower.

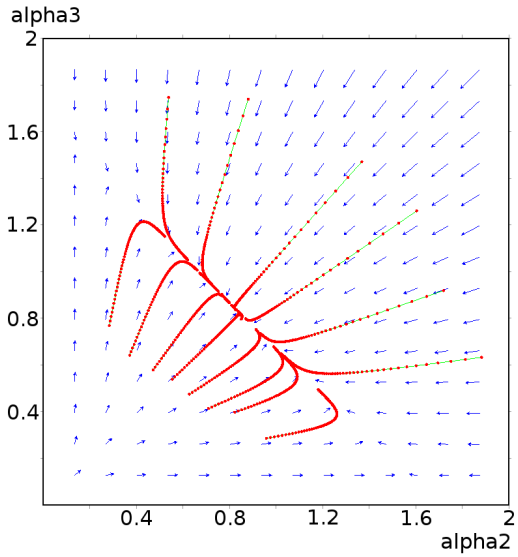



Figure 4.4: Plot of some trajectories of the step function for the ground truth $\alpha = (0.1, 1, 10)$ and for a sample of $N = 1000$ data. The sequences of point are sequences of iterations of the algorithm. The small arrows represents the directions of $\hat{\alpha}^{(k+1)} - \hat{\alpha}^{(k)}$.

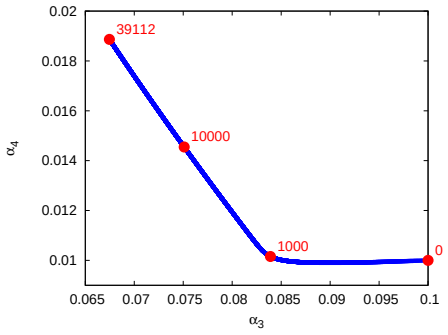



Figure 4.5: Plot of the values of the two smallest coordinate of each $\hat{\alpha}^{(k)}$ for the ground truth $\alpha = (1, 1/3, 0.1, 0.01)$.

| number of steps | log-likelihood |
|---|---|
| 0 | −1.240947 |
| 1 | −1.240910 |
| 10 | −1.240870 |
| 100 | −1.240831 |
| 1000 | −1.240803 |
| 10000 | −1.240787 |
| 20000 | −1.240783 |
| 39113 | −1.240782 |

Table 4.8: Evolution of the log-likelihood corresponding to the trajectory on the left side, it increases extremely slowly with $k$.

**Example Two** In this case $\alpha = (1, 1/3, 0.1, 0.01)$.

Figure 4.5 shows the trajectory of the two smallest coordinates of the sequence $\hat{\alpha}^{(k)}$ for a sample of $N = 10000$ data, with initial condition the ground truth itself. The algorithm stopped after 39113 steps. The red points show the trajectory after 0 steps (starting point), 1000 steps, 10000 steps and 39113 steps when the algorithm finally stopped.

As we can see, the trajectory again mostly falls on a straight line, and not unexpectedly, the size of each step drops while the trajectory approaches the final point. Table 4.8 shows

the evolution of the log-likelihood — it increases extremely slowly.

The extremely small rate of increase in likelihood as a function of $\hat{\alpha}^{(k)}$ along the trajectories means that the usual termination criteria, consisting in stopping when $\|\hat{\alpha}^{(k+1)} - \hat{\alpha}^{(k)}\| < \varepsilon$, or $L_d(\hat{\alpha}^{(k+1)}) - L_d(\hat{\alpha}^{(k)}) < \varepsilon$ for some small $\varepsilon$, or after a fixed number of iterations, can result in significant errors. The criterion we used to avoid these traps was to stop when $L_d(\hat{\alpha}^{(k+1)}) \leq L_d(\hat{\alpha}^{(k)})$, which in theory never happens but occurs in practice because of numerical errors very close to the fixed point.

These two examples illustrate two key characteristics of the EM algorithm (for this problem). First, the trajectory reaches an area relatively close to the final point relatively quickly, where it enters a 'glide path' which is relatively linear, corresponding in a sense to a valley of the function $-\log L_d(\theta)$, or 'reversed valley' of $\log L_d(\theta)$ (for the sake of simplicity, we will abusively refer to it as a 'valley'). Second, once in the 'valley', the speed of the trajectory becomes particularly slow. These two observations inspire the following strategy to accelerate EM: (i) reach a good first approximation as quickly as possible, (ii) once near the 'valley', increase the size of the steps to go faster.

### 4.6.2 The Sampling Method

This section addresses point (i) above. Our objective is to reach the 'valley' leading to the fixed point extremely quickly, using the intuition that even a rough method should be able to achieve this objective.

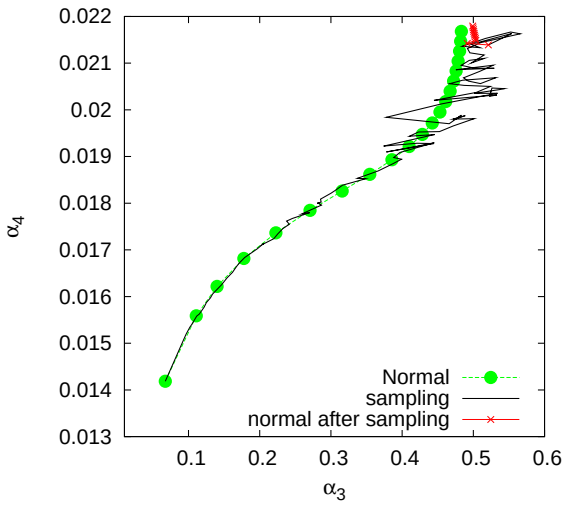



Figure 4.6: Comparison of the first 100 steps of a normal trajectory ("normal", every fifth step is shown) with 100 steps of the sampling method ("sampling"), followed by 10 normal steps ("normal after sampling"), from a random starting point. The ground truth $\alpha = (1, 1/3, 0.1, 0.01)$ and data length $N = 10000$ is as in Figure 4.5, but the data set is different.

The method consists in cutting the data (of length $N$) into $N/k$ subsets of equal length $k$ (for simplicity we assume that $k$ divides $N$). We then compute some iterations of the EM

algorithm using only one of the subsets as data, the subset being chosen randomly at each iteration.

More precisely, at each step, instead of computing the usual iteration $\hat{\alpha}^{(k+1)} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{E}_{\hat{\alpha}^{(k)}}(l|d(i))$ we compute one of the following iterations:

$$\hat{\alpha}^{(k+1)} = \frac{1}{k}\sum_{i=jk+1}^{(j+1)k} \mathbf{E}_{\hat{\alpha}^{(k)}}(l|d(i)) \quad ,$$

where at each step, $j$ is uniformly at random chosen among $\{0, \ldots, N/k-1\}$.

The advantage of this method is that $N/k$ steps based on subsets will cost only as much as 1 step using the full data, and yet gives a fair first approximation of the fixed point. This is a way to sacrifice precision for speed, but since we only want a first approximation here, low precision is acceptable. In particular, during these cheap steps, the likelihood of the parameters does not necessarily increase, but the parameters does get closer from the final point.

Several choices of $k$ are possible: in this paper, we used $k = \sqrt{N}$ ($N = 10000$ and $k = 100$), which worked well in practice. Any other choice (as long as $k$ is "large enough to be representative, but not too large to gain computation time" makes sense.

As a non-rigorous intuition that these cheap steps will still go in the right direction, note that since the integer $j$ is randomly chosen, the expectation of each random step is:

$$\mathbf{E}_j(\hat{\alpha}^{(k+1)}) = \frac{k}{N}\sum_{j=0}^{N/k-1}\frac{1}{k}\sum_{i=jk+1}^{(j+1)k} \mathbf{E}_{\hat{\alpha}^{(k)}}(l|d(i)) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{E}_{\hat{\alpha}^{(k)}}(l|d(i)) \quad .$$

So, in average, each cheap step goes in the same direction as a normal step.

Figure 4.6 shows an example of this method. As we can see the first iterations move in the same direction as the normal EM steps, and 100 iterations of the sampling method leads to a point close to the one obtained after 100 normal steps, but costing only as much as one normal step to get there.

### 4.6.3 The Steered Jumping Method

This section addresses point (ii) above. Namely, once we get a first approximation of the fixed point which is close to or within the 'valley', the steps usually become very small but the trajectory is linear. Our strategy is to exploit this linearity to increase step size dramatically. We will present the method in a more general context, since we believe it could be applied to other cases where the same kind of convergence issues are encountered.

#### General Context

We are given a function $F$ and we want to find a local maximum by using an iterative algorithm (*e.g.* the EM algorithm, gradient ascent) which can be expressed as follows: Starting

from some point $\hat{\alpha}^{(0)}$, construct the sequence $(\hat{\alpha}^{(0)}, \hat{\alpha}^{(1)}, \dots)$ defined by the recursive formula

$$\hat{\alpha}^{(k+1)} = \hat{\alpha}^{(k)} + \Delta_k \quad , \tag{4.9}$$

where the parameter $\Delta_k$ is chosen such that $F(\hat{\alpha}^{(k)}) \leq F(\hat{\alpha}^{(k+1)})$, with equality if and only if $\hat{\alpha}^{(k)}$ is a stationary point of $F$. The algorithm stops when the equality is reached.

The way to compute the parameter $\Delta_k$ depends on the chosen algorithm. In the case of the EM algorithm, we have $\Delta_k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\hat{\alpha}^{(k)}}(l|d(i)) - \hat{\alpha}^{(k)}$. In the case of a gradient ascent, we have $\Delta_k = \delta_k \nabla F(\hat{\alpha}^{(k)})$, where $\nabla F$ is the gradient of $F$, and where $\delta_k$ is some well chosen positive scalar number.

**Jumping method**

In a case where the behaviour of such an algorithm is as in Figure 4.5, namely linear and very slow, we would like to take much larger steps. More precisely, we would like to replace the last equation (4.9) by

$$\hat{\alpha}^{(k+1)} = \hat{\alpha}^{(k)} + \beta_k \Delta_k \quad , \tag{4.10}$$

with $\beta_k \geq 1$ and hopefully much bigger than 1, such that the relation $F(\hat{\alpha}^{(k)}) \leq F(\hat{\alpha}^{(k+1)})$ still holds at each iteration. In some sense, if $\beta_k = n \in \mathbb{N}$ we can interpret this as assuming that $\Delta_k \simeq \Delta_{k+1} \simeq \cdots \simeq \Delta_{k+n}$ and approximating all of them by $\Delta_k$, and then computing $n$ steps in one. We then say that we "jump" with a factor $\beta_k$. Figure 4.7 shows an example of this method applied to the example of Figure 4.5.

This idea is not new as such, and it has been applied to the EM algorithm in [JJ93] as a generalized conjugate gradient algorithm, and in [SR03] as an overrelaxed bound optimization.

The method as described above has two main flaws. First, we do not know how to choose the values of $\beta_k$ efficiently at each step. Second, and most importantly, it sometimes results in a behaviour similar to the one visible on Figure 4.7. If we try to jump too far from one side of the 'valley', we end up on the other side rather than reaching and tracking the 'valley' floor, resulting in an inefficient zigzag trajectory. In other words, increasing step size can cause instability. To counter this, we next modify not only the size of the steps, but also their direction.

**Steered Jumping Method**

We modify slightly the formula (4.10) of the jumping method to read:

$$\hat{\alpha}^{(k+1)} = \hat{\alpha}^{(k)} + \beta_k C_k \Delta_k \quad , \tag{4.11}$$

where $C_k$ will be a well chosen matrix such that $F(\hat{\alpha}^{(k)}) \leq F(\hat{\alpha}^{(k+1)})$ still holds. The choice of $C_k$ can depend on many parameters, like the current state of the algorithm but also its past iterations. It is important to notice that it is always possible to try different choices

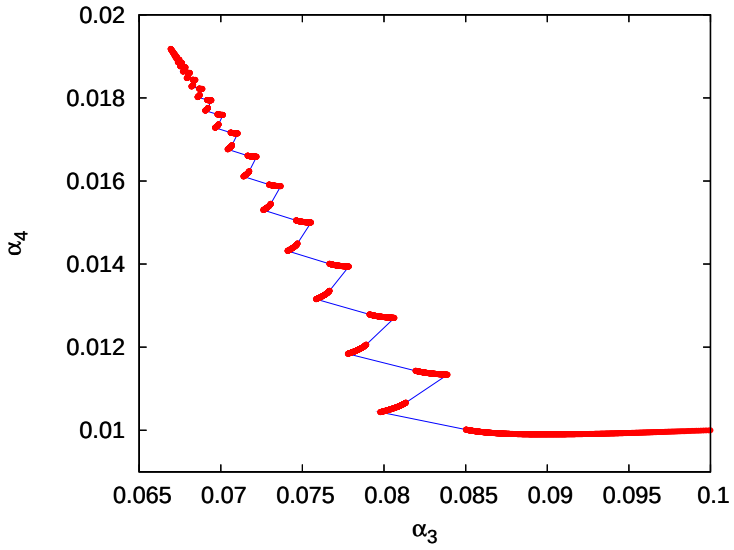



Figure 4.7: Trajectory using the jumping method, projected onto the two smallest coordinates of $\hat{\alpha}^{(k)}$, for the ground truth $\alpha = (1, 1/3, 0.1, 0.01)$ using the same data as in Figure 4.5. In this case we tried to jump every 100 steps, *i.e.* we had $\beta_k = 1$ except when $k = 0 \pmod{100}$ where we set $\beta_k = 1000$ if $F(\hat{\alpha}^{(k)}) \leq F(\hat{\alpha}^{(k)} + \beta_k \Delta_k)$, or $\beta_k = 1$ otherwise. The visible gaps between points reveal where jumps actually occurred. The total number of iterations before reaching the minimum was 14684, compared to 39112 for the normal EM.

of $C_k$ and $\beta_k$ and chose the one which gives the highest value of $F(\hat{\alpha}^{(k)} + \beta_k C_k \Delta_k)$.

In our case, we would like the jumps to be in the same direction as the 'valley'. For this, we use at each step the information given by the previous iterations of the algorithm about the global shape of the 'valley' to compute the matrix $C_k$, via Principal Component Analysis (PCA). This was inspired by a method recently developed in robotics to accelerate the growth of Rapidly-exploring Random Trees (RRT) for path finding [DL08].

### A Short Introduction to PCA

Principal Component Analysis is a method used to find the main axes of concentration of a set of points in a high dimensional space.

Consider a set of points $X = (x_1, \ldots, x_n)$, each $x_i$ being a point of $\mathbb{R}^d$. We construct the $d \times d$ matrix $C$ called the *covariance matrix* of the set of points X, whose element $(i, j)$ is $(x_i - \bar{x})(x_j - \bar{x})$, where $\bar{x} \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} x_i$ is the mean point of the set. The matrix is symmetric positive semi-definite, and has the property that it captures very well the repartition of the points $x_i$ in the space, since it is diagonalizable by the Cayley-Hamilton theorem. The eigenvectors $(e_1, \ldots, e_k)$ associated to its biggest eigenvalues $(\lambda_1, \ldots, \lambda_k)$ $(k \leq d)$ are the axes where the points are the most dispersed.

PCA usually consist in the computation of the $(e_1, \ldots, e_k)$, in order to restrict the space $\mathbb{R}^d$ to the subspace Vect$\{e1, \ldots, e_k\}$ with $k$ much smaller than $d$. To avoid the cost of computing these eigenvectors, we instead multiply directly by the covariance matrix, which

naturally flattens vectors along the main eigenvector axis.

### The PCA-jumping method

We now define the specific method we used based on the principles outlined above. Other variants are clearly possible, and we discuss some of these later.

The matrix $C_k$ is constructed as the covariance matrix of the set of the last $p_k$ iterations of the algorithm: $(\hat{\alpha}^{(k-1)}, \ldots, \hat{\alpha}^{(k-p_k)})$ for some well chosen $p_k$. For each iteration, we try three possible values for $C_k$: $C_k^1 = Id$ the identity matrix, $C_k^2 = \text{Cov}(\hat{\alpha}^{(k-1)}, \ldots, \hat{\alpha}^{(k-10)})$ the covariance matrix of the 10 last points, and $C_k^3$ the matrix of the 100 last points. We will say that $C_k^1$ corresponds to using *no memory*, $C_k^2$ to a *short memory*, and $C_k^3$ to a *long memory*. When there are insufficient points to fill the memory, we take as many as are available, for example when $k \leq 100$ we use $C_k^3 = \text{Cov}(\hat{\alpha}^{(k-1)}, \ldots, \hat{\alpha}^{(1)})$.

So that the matrices control the direction but not the size of jumps, at each step $k$ we renormalize to form the steered direction vectors $d_k^i = \dfrac{C_k^i \Delta_k}{\|C_k^i \Delta_k\|} \|\Delta_k\|$, $i = 1, 2, 3$.

We use the following aggressive algorithm to select the step size $\beta_k^i$ for each $i$:

**0)** Initialize with $\beta_k^i = 1$ ;

**1)** **If** $F(\hat{\alpha}^{(k)} + \beta_k^i d_k^i) < F(\hat{\alpha}^{(k)} + 2\beta_k^i d_k^i)$ **then** $\beta_k^i \leftarrow 2\beta_k^i$ and repeat **1)**
**else** return $\beta_k^i$ .

In other words, as long as doubling the jump size improves $F$, we double again.

Finally, we combine the step size and direction into three candidates: $\hat{\alpha}^i = \hat{\alpha}^{(k)} + \beta_k^i d_k^i$ for $i = 1, 2, 3$, and set $\hat{\alpha}^{(k+1)}$ to the one giving the highest value of $F$.

It is important to note that despite the opportunistic character of this algorithm, the crucial inequality $F(\hat{\alpha}^{(k)}) < F(\hat{\alpha}^{(k+1)})$ when a maximum is not yet reached is guaranteed, since $F(\hat{\alpha}^{(k)}) < F(\hat{\alpha}^{(k)} + \Delta_k)$ is guaranteed by the definition of $\Delta_k$, and because $F(\hat{\alpha}^{(k)} + \Delta_k) = F(\hat{\alpha}^{(k)} + 1 \times C_k^1 \Delta_k) \leq F(\hat{\alpha}^{(k+1)})$.

### The Cost of PCA-Jumping

Figure 4.8 gives an example of the EM algorithm with no speed-up, with the jumping method alone, and with PCA-jumping. Figure 4.9 shows the corresponding evolution of the log-likelihood for these trajectories.

The effect of the multiplication by the covariance matrix is that the direction of the basic EM step $\Delta_k$ is steered towards the axis where the previous iterations are concentrated. Thanks to this, the unwanted oscillation effect of Figure 4.7 is avoided, and the size of the jumps (*i.e.* the values of $\beta_k$) can become much larger.

We used three levels of memory to capture the shape of the trajectory on different 'spatial' scales. We noticed that usually all three alternatives are employed by the algorithm, and that on average the biggest jumps were made with the short memory.

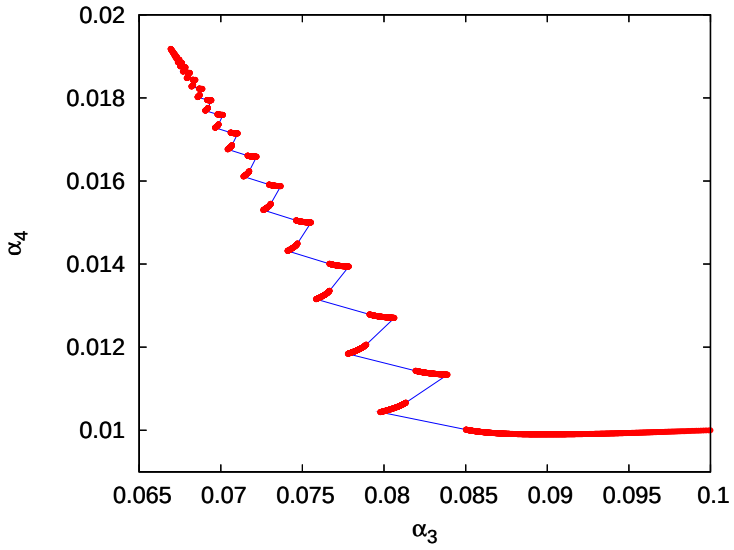



Figure 4.8: Trajectories of the two smallest coordinates of $\hat{\alpha}^{(k)}$, starting from the ground truth $\alpha = (1, 1/3, 0.1, 0.01)$ but with a different data set than that of Figure 4.5, using different methods. The "normal" trajectory (1 point per 1000 shown) is the EM algorithm without any speed-up, "jump" trajectory is the jumping method without PCA (choosing always $i = 1$), and "jump+PCA" uses the complete method. The "normal" trajectory has 47073 steps, "jump" has 1215, and "jump+PCA" only 59.

Each level of memory involves computing $F$, however the cost of this is not larger than that of computing the initial $\Delta_k$. In our problem, computing one normal EM step costs more than computing the log-likelihood $|T|$ times, since we need to compute $\frac{1}{N}\sum_{i=1}^{N} \mathbf{E}_{\hat{\alpha}^{(k)}}(l_j|d(i))$ for each $j \in T$, each $\mathbf{E}_{\hat{\alpha}^{(k)}}(l_j|d(i))$ being more complicated to compute than the likelihood. Therefore a single step of the PCA-jumping algorithm usually costs no more than 2 or 3 times a normal EM step, while the total number of steps is greatly reduced. The computation of the covariance matrices is, again for this problem, very cheap compared to the time needed to compute $F$ or $\Delta_k$.

Clearly, the above strategy has parameters which could be optimized. In particular the number of memory levels, and their durations, could be altered. It would also be possible to use the power $C^n$ of a covariance matrix instead of $C$ to increase the steering effect, or even to use alternative matrices. In [DL08], PCA (even the eigenvectors) was computed by using a recursive method allowing points to be added successively until the number of principal dimensions ceased dropping. Such a method could be employed here too.

More elaborate methods will always come at increased cost. The advantage of our particular strategy within the PCA-jumping class is its simplicity, since by selecting from only three possible steps, we capture the essence of traditional EM, as well as knowledge of the local and global trajectory shape, and by multiplying by the covariance matrix, we employ PCA without the usual costs of eigenvector evaluation. As for the choice of $\beta_k$, our strategy has the advantage of being both simple and very aggressive, allowing large values

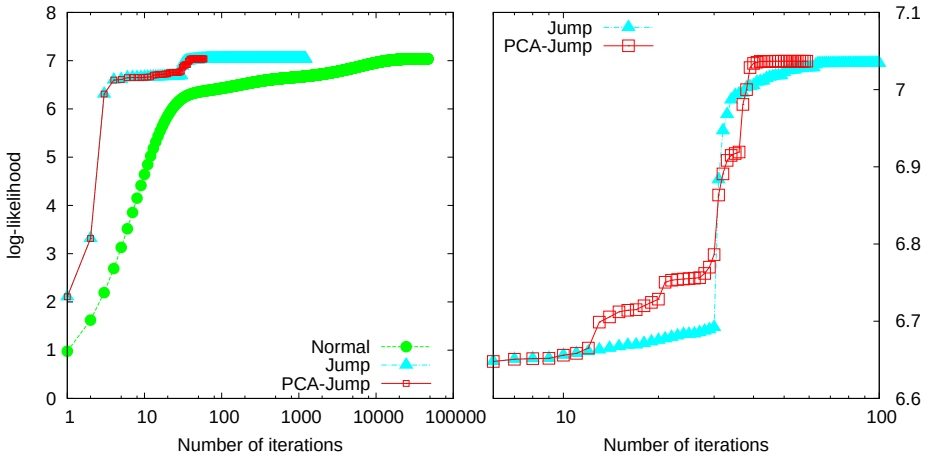



Figure 4.9: Evolution of the log-likelihood as a function of the number of iterations, corresponding to the trajectories of Figure 4.8. The right-side shows a zoom of the left-side.

of $\beta_k$ to be found quickly and in a single iteration, with no arbitrary upper limit imposed. (We investigated the possibility of selecting $\beta_k$ based on maximizing $F(\hat{\alpha}^{(k)} + \beta_k C_k \Delta_k)$, through a binary search, however, the overhead of the search was not compensated by the gain in jump size.)

### PCA-Jumping with Sampling Initialization: Results

Our final method consists of using a number of steps of the sampling method to get a rough approximation, which is then used to initialize the PCA-jumping method. To control the resources used by the sampling method phase, we set the number of sampling steps to be equivalent computationally to a single step of normal EM. This method was used for most of the experiments presented in Section 4.5.

Figure 4.10 shows a comparison against the normal EM, using the same ground truth and data as Figures 4.6 and 4.8, starting from a random point. The speed-up due to the method is very significant in this case. Table 4.9 shows a speed comparison against normal EM for one experiment for each of the $4$ main examples of Section 4.5, starting from the ground truth. All simulations were made on the same Intel Core2 Duo 2.40GHz laptop, but using only one CPU. Under the iterations column, "$100 + x$" means $100$ sampling method steps followed by $x$ steps of PCA-jumping. In all cases these $100$ steps cost as much as one normal step. The cost of the sampling steps was omitted when computing the average step time for PCA-jumping (last column).

We see that the acceleration method was particularly effective for the trees $U9$ and $B3H2$, where the convergence of the normal algorithm is extremely slow, but still produces a substantial gain for the binary trees $B1H3$ and $B1H4$ where the convergence of the normal EM was however already fairly good. The reason for such a difference is, we believe, the fact that $U9$ and $B3H2$ each contain long branches. The reason might be that we get exactly the same information for all the nodes in one same branch, and it becomes thus

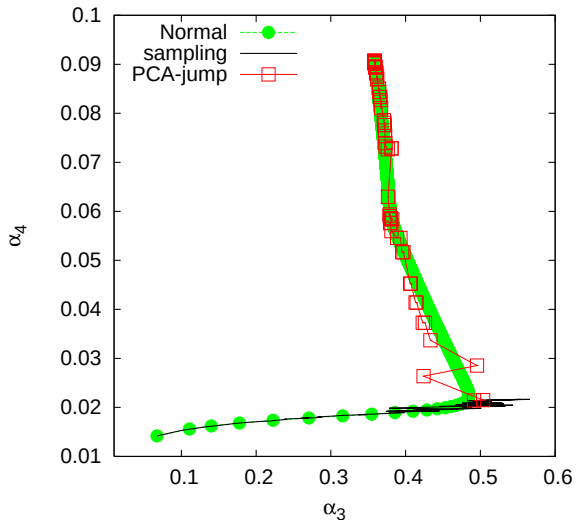



Figure 4.10: Trajectories of the two smallest coordinates of $\hat{\alpha}^{(k)}$, for the ground truth $\alpha = (1, 1/3, 0.1, 0.01)$, starting from a random point. The "normal" trajectory (1 point per 5 shown) is the EM algorithm without any speed-up. The "sampling" trajectory correspond to 100 iterations of the sampling method, (with cost equal to 1 step of the normal EM), and the "jump+PCA" trajectory was obtained by initializing the PCA-jumping method from the final point of the "sampling" trajectory. The "normal" trajectory has 58034 steps, while the "jump+PCA" has only 78. The computing time for the "sampling" and "jump+PCA" trajectories together was 195 times less than for "normal".

much harder to discriminate between them, resulting in slow convergence. This tendency has been also observed on other trees.

| Tree | Method | Final Log-L | # Iterations | CPU time | Av. step |
|---|---|---|---|---|---|
| $U9$ | Normal EM | $-7.517$ | 341845 | 1373m24s | 0.24s |
| | PCA-jump | $-7.517$ | $100 + 699$ | 6m46s | 0.58s |
| $B1H3$ | Normal EM | $-20.107$ | 153 | 1m21s | 0.53s |
| | PCA-jump | $-20.107$ | $100 + 14$ | 20s | 1.42s |
| $B1H4$ | Normal EM | $-35.614$ | 207 | 62m20s | 18.07s |
| | PCA-jump | $-35.614$ | $100 + 28$ | 14m12s | 30.4s |
| $B3H2$ | Normal EM | $-10.155$ | 122941 | 2853m16s | 1.4s |
| | PCA-jump | $-10.155$ | $100 + 429$ | 21m40s | 3.0s |

Table 4.9: Comparison between the normal EM and the sampling + PCA-jumping method for one estimation on the 4 trees in Section 4.5, starting from the ground truth.

| Tree | Method | Final Log-L | # Iterations | CPU time | Av. step |
|---|---|---|---|---|---|
| $U9$ | Normal | $-7.570$ | 3933 | 19m26s | 0.29s |
| | PCA-jump | $-7.517$ | $100 + 755$ | 6m53s | 0.55s |
| $B1H3$ | Normal EM | $-20.107$ | 216 | 1m57s | 0.54s |
| | PCA-jump | $-20.107$ | $100 + 14$ | 20s | 1.42s |
| $B1H4$ | Normal EM | $-35.614$ | 315 | 89m25s | 17.03s |
| | PCA-jump | $-35.614$ | $100 + 26$ | 12m44s | 29.38s |
| $B3H2$ | Normal EM | $-10.255$ | 2043 | 62m10s | 1.8s |
| | PCA-jump | $-10.155$ | $100 + 676$ | 41m41s | 3.7s |

Table 4.10: Comparison between the normal EM and the PCA-jumping method for one estimation on the 4 trees in Section 4.5, starting from a random point.

We used the ground truth as the initial condition here as we noticed that, in some cases, the normal EM converged towards a local maximum with a likelihood much lower than the one found by the accelerated method. When starting both methods from the ground truth they converged to the same fixed point, facilitating a direct speed comparison. Table 4.10 shows the comparison when starting from a random point. For the trees $U9$ and $H3B2$ the normal EM converged quite quickly to the final point (thought still less quickly than the accelerated EM), but more importantly, this point was **not** the MLE as its likelihood was smaller than the fixed point found by the accelerated EM. In all the experiments we performed starting from the same initial conditions, our method converged quicker than the normal EM, and always gave a likelihood as good as the normal method, if not better.

## 4.7 Summary

We have considered a network tomography problem based on a finite number of end-to-end delay measurements made of multicast probes sent over a tree, where each node of the tree imparts an exponentially distributed delay to each passing probe. We showed how its assumptions of spatial independence, and sum-of-exponentials delay marginals, follow naturally from the properties of Kelly networks in the case of rare probing, thereby firmly establishing for the first time a connection between a delay tomography problem and an inverse queueing problem over a network with non-trivial topology.

The problem was formulated as the search for a maximum likelihood estimator for the parameter, being the vector of mean delays for each node in the tree, which due to its complexity was solved using the E-M algorithm. We showed how the E and M steps could be solved explicitly and combined, reducing the problem to the evaluation of a set of conditional probabilities of internal node states, given the observed delays. We provided two solution methods for these with formal proofs, one based on a recursion beginning from the root node, the other an explicit expression (though with some recursive components).

The latter has far fewer terms and is amenable to efficient implementation, and was used to provide solutions for a number of examples.

The E-M algorithm is notoriously slow to converge, and moreover since the combinatorics of the tree make each step very expensive, only trivial trees can be solved in practice without acceleration techniques. We developed a new technique, *PCA-jumping with Sampling Initialization*, which provided a speed-up of between one and three orders of magnitude for our problem. Its novel features include the use of Principal Component Analysis (yet without the need to calculate eigenvectors) to efficiently mine local and global information about the E-M trajectory in order to control jump direction, and an efficient and aggressive geometric rule for the size of jumps which allows large steps to be made when profitable, as is the case for our problem. Initialization is performed using a 'Sampling' method which has very low and bounded cost, yet is capable of finding a starting point from which PCA-jumping can be effective. The speed of the method is compared to standard E-M in a variety of examples, and was also shown to provide better estimates in some cases.

The main directions for future work lie in a more formal analysis of the acceleration technique, its optimization with respect to a number of parameters, generalizations, and comparison against alternatives. Of particular interest is to understand to what extent the 'valley' phenomenon which inspired the technique holds for other problems, in particular non-linear ones where fixed points have small basins of attraction.

## 4.8 Appendix

We recall here that we use the notation simplification from Section 4.4. In particular, we use $\alpha$ in place of $\hat{\alpha}^{(k)}$ (as a single iteration is considered), and we set $\gamma = (\gamma_j)_{j\in T} = (1/\alpha_j)_{j\in T}$.

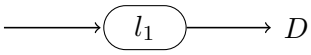
### 4.8.1 Proof of the Density Formula

We will now prove that the description we gave for the expanded expression of $q_\alpha(\mathcal{T}, d)$ is correct. For this, we will show that the following equality holds:

$$q_\alpha(\mathcal{T}, d) = \Gamma_T \sum_{C \text{ cut of } \mathcal{T}} h_\alpha(\mathcal{T}, d, C) \quad , \tag{4.12}$$

for $q_\alpha(\mathcal{T}, d)$ defined by its integral expression (4.5) and for the right-side terms as defined previously in 4.4.5.0. For this we will have an inductive reasoning over the tree $\mathcal{T}$, and we will use the inductive relation of Theorem 4.4.1.

#### Initialization of the Induction



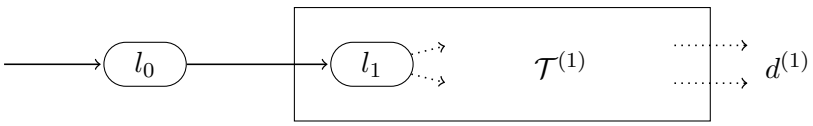

It is obvious that the formula holds for a single-node tree. In this case, we have $q_\alpha(d) = \gamma_1 e^{-\gamma_1 d}$, and it is easy to verify that it correspond to the formula given previously, since there is only one cut $C = \{1\}$ with no past nor future. Therefore, we have: $q_\alpha(d) = \gamma_1 r(C)s(C)t(C)$ with $r(C) = 1$, $t(C) = 1$, and $s(C) = e^{-\gamma_1 d_1}$, which proves the initial step of the induction.

#### Induction Step

We consider a general tree, assume by induction that the formula is true for each subtree of the root, and then prove that the formula is also true for the whole tree.

For this we will have to make a distinction between two cases: if the root has only one child and if it has at least two children. This distinction will explain why we introduced the notion of branch, since in the one-child case, some terms from the same branch can be combined.

#### a) Root with only one Child




We have $T^{(1)} = T \setminus \{0\}$, $V^{(1)} = V$, and $d^{(1)} = d$ and the inductive formula (4.6)

becomes:

$$q_\alpha(\mathcal{T}, d) = \int_{l_0=0}^{d_0} \gamma_0 e^{-\gamma_0 l_0} q_\alpha(\mathcal{T}^{(1)}, d - (l_0)) dl_0 \quad .$$

By induction, we suppose the formula true for the subtree $\mathcal{T}^{(1)}$, therefore:

$$q_\alpha(\mathcal{T}^{(1)}, d - (l_0)) = \Gamma_{T^{(1)}} \sum_{C \text{ cut of } \mathcal{T}^{(1)}} h_\alpha(\mathcal{T}^{(1)}, d - (l_0), C) \quad ,$$

and since $\Gamma_T = \gamma_0 \Gamma_{T^{(1)}}$:

$$q_\alpha(\mathcal{T}, d) = \Gamma_T \sum_{C \text{ cut of } \mathcal{T}^{(1)}} \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} h_\alpha(\mathcal{T}^{(1)}, d - (l_0), C) dl_0 \quad . \tag{4.13}$$

Let us consider a cut $C$ of $\mathcal{T}^{(1)}$. We have:

$$h_\alpha(\mathcal{T}^{(1)}, d - (l_0), C) = r(\mathcal{T}^{(1)}, d - (l_0), C)\ s(\mathcal{T}^{(1)}, d - (l_0), C)\ t(\mathcal{T}^{(1)}, d - (l_0), C) \quad .$$

*Remark.* For more convenient notations, we will denote $h_\alpha^{(1)}(C) := h_\alpha(\mathcal{T}^{(1)}, d - (l_0), C)$ and for $r, s, t$ as well, such that the last relation becomes $h_\alpha^{(1)}(C) = r^{(1)}(C) s^{(1)}(C) t^{(1)}(C)$ .

We now want to compute this integral: $\int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} r^{(1)}(C) s^{(1)}(C) t^{(1)}(C) dl_0$. As we can see in their definition, the variable $l_0$ does not appear in the terms $r^{(1)}(C)$ and $t^{(1)}(C)$. This fact is obvious for $r$, but the recursive nature of $t$ make it a little more difficult to see. But looking back at the definition, we notice that the recursion is made with "$d - (d_j) := (d_k - d_j)_{k \in F_j}$". Since here our whole formula is applied to $d' = d - (l_0)$, we see that the term $l_0$ is annihilated in the expression of $d' - (d'_j) = ((d_k - l_0) - (d_j - l_0))_{k \in F_j}$. And therefore, $l_0$ does not appear in $t^{(1)}(C)$.

Thanks to this, we have $r^{(1)}(C) = r(\mathcal{T}^{(1)}, d, C)$ and $t^{(1)}(C) = t(\mathcal{T}^{(1)}, d, C)$ and we can take $r^{(1)}(C)$ and $t^{(1)}(C)$ out of the integral, leaving us with this integral to expand:

$$\begin{aligned}
&\int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} s^{(1)}(C) dl_0 = \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \prod_{j \in C} \frac{e^{-\gamma_j (d_j - l_0)}}{\prod\limits_{\substack{k \sim j \\ k \neq j}} (\gamma_k - \gamma_j)} dl_0 \\
&= \left[ \frac{1}{\left(\sum_{j \in C} \gamma_j\right) - \gamma_0} \prod_{j \in C} e^{-\gamma_0 l_0} \frac{e^{-\gamma_j (d_j - l_0)}}{\prod\limits_{\substack{k \sim j \\ k \neq j}} (\gamma_k - \gamma_j)} \right]_{l_0=0}^{d_0} \\
&= \frac{1}{\left(\sum_{j \in C} \gamma_j\right) - \gamma_0} \prod_{j \in C} e^{-\gamma_0 d_0} \frac{e^{-\gamma_j (d_j - d_0)}}{\prod\limits_{\substack{k \sim j \\ k \neq j}} (\gamma_k - \gamma_j)} + \frac{1}{\gamma_0 - \sum_{j \in C} \gamma_j} \prod_{j \in C} \frac{e^{-\gamma_j d_j}}{\prod\limits_{\substack{k \sim j \\ k \neq j}} (\gamma_k - \gamma_j)} \\
&= e^{-\gamma_0 d_0} \frac{s(\mathcal{T}^{(1)}, d - (d_0), C)}{\left(\sum_{j \in C} \gamma_j\right) - \gamma_0} + \frac{s(\mathcal{T}^{(1)}, d, C)}{\gamma_0 - \sum_{j \in C} \gamma_j} = e^{-\gamma_0 d_0} K(C) + \frac{s(\mathcal{T}^{(1)}, d, C)}{\gamma_0 - \sum_{j \in C} \gamma_j} \quad .
\end{aligned}$$

As we will see, the exact value of $K(C)$ is in fact not really important for the proof. Once here, the hardest part remains: we have to put all these terms together to prove that the formula is true for the whole tree $\mathcal{T}$.

Putting the last equation back in (4.13), we get

$$q_\alpha(\mathcal{T}, d) = \Gamma_T \left[ e^{-\gamma_0 d_0} K' + \sum_{C \text{ cut of } \mathcal{T}^{(1)}} \frac{r^{(1)}(C) s(\mathcal{T}^{(1)}, d, C)}{\gamma_0 - \sum_{j \in C} \gamma_j} t^{(1)}(C) \right] \quad , \tag{4.14}$$

where $K' = \sum_{C \text{ cut of } \mathcal{T}^{(1)}} r^{(1)}(C) t^{(1)}(C) K(C)$.

As we noticed already, we have $r^{(1)}(C) = r(\mathcal{T}^{(1)}, d, C)$ and $t^{(1)}(C) = t(\mathcal{T}^{(1)}, d, C)$. But, since the term $t(\mathcal{T}^{(1)}, d, C)$ is a term depending only from the cut $C$ and its future in the tree $\mathcal{T}^{(i)}$, which are the same in the tree $\mathcal{T}$, we have $t(\mathcal{T}^{(1)}, d, C) = t(\mathcal{T}, d, C)$.

We will now prove that:

$$\frac{r(\mathcal{T}^{(1)}, d, C) s(\mathcal{T}^{(1)}, d, C)}{\gamma_0 - \sum_{j \in C} \gamma_j} = r(\mathcal{T}, d, C) s(\mathcal{T}, d, C) \quad .$$

It is relatively easy to see by looking at the definitions of $r$ and $s$, but two cases must be distinguished: when $C$ is a singleton, like $C = \{1\}$ here, or when $C$ contains 2 nodes or more. Indeed, if $C$ is a singleton, say $C = \{i\}$, then $i$ belongs to the present of 0 and 1 (note that here $0 \sim 1$), and we have $s(\mathcal{T}, d, C) = \dfrac{s(\mathcal{T}^{(1)}, d, C)}{\gamma_0 - \gamma_i}$ and $r(\mathcal{T}, d, C) = r(\mathcal{T}^{(1)}, d, C) = 1$ since the past of $C$ is empty in this case. In the other case where $C$ has 2 nodes or more, then we see that 0 and 1 belong to the past of $C$ and we have $s(\mathcal{T}, d, C) = s(\mathcal{T}^{(1)}, d, C)$ and $r(\mathcal{T}, d, C) = \dfrac{r(\mathcal{T}^{(1)}, d, C)}{\gamma_0 - \sum_{j \in C} \gamma_j}$. In both cases, the identity is then proved.

This results in:

$$q_\alpha(\mathcal{T}, d) = \Gamma_T \left[ e^{-\gamma_0 d_0} K' + \sum_{\substack{C \text{ cut of } \mathcal{T} \\ C \neq \{0\}}} h_\alpha(\mathcal{T}, d, C) \right] \quad , \tag{4.15}$$

since we notice that all the cuts of $\mathcal{T}^{(1)}$ plus the cut $\{0\}$ forms exactly all the cuts of $\mathcal{T}$.

Finally, in order to show that the formula is true for the tree $\mathcal{T}$, all we have to do left is to show that the term $e^{-\gamma_0 d_0} K'$ is equal to $h_\alpha(\mathcal{T}, d, \{0\})$. If we look back at the expression of $K' = \sum_{C \text{ cut of } \mathcal{T}^{(1)}} r^{(1)}(C) t^{(1)}(C) K(C)$, it seems difficult to show directly that this sum of terms combine into just one and is equal to $h_\alpha(\mathcal{T}, d, \{0\})$. Luckily, we will not have to do this, since a simple argument of symmetry will suffice. All we have to do is to notice that the function $q_\alpha(\mathcal{T}, d)$ stays the same if we exchange the nodes 0 and 1, *i.e.* if we exchange the values of $\gamma_0$ and $\gamma_1$. This becomes obvious if we take a look at the two station case, since $l_1 + l_2 = l_2 + l_1$, the two following trees are equivalent:

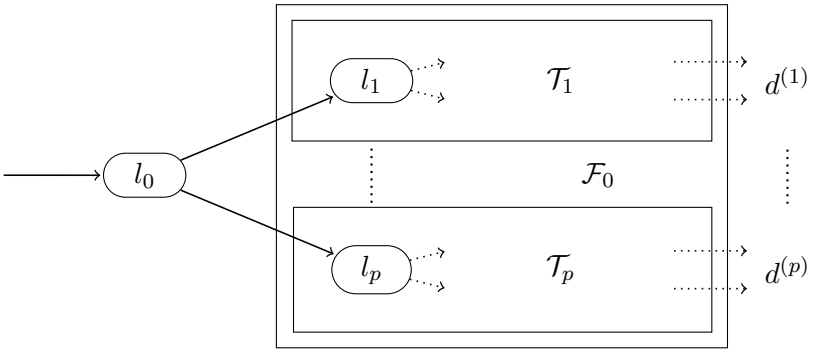

More generally, no permutation inside a branch of $\mathcal{T}$ will change $q_\alpha(\mathcal{T}, d)$. Thanks to this, we see that by exchanging the nodes 0 and 1, we get from (4.15):

$$q_\alpha(\mathcal{T}, d) = \Gamma_T \left[ e^{-\gamma_1 d_1} K'' + \sum_{\substack{C \text{ cut of } \mathcal{T} \\ C \neq \{1\}}} h_\alpha(\mathcal{T}, d, C) \right] \quad , \tag{4.16}$$

and by combining (4.15) and (4.16), we get $e^{-\gamma_0 d_0} K' + h_\alpha(\mathcal{T}, d, \{1\}) = e^{-\gamma_1 d_1} K'' + h_\alpha(\mathcal{T}, d, \{0\})$, and we can finally identify the two terms $e^{-\gamma_0 d_0} K'$ and $h_\alpha(\mathcal{T}, d, \{0\})$ since the term $h_\alpha(\mathcal{T}, d, \{0\})$ contains an exponential function of the form $e^{-\gamma_0 d_0}$ and $h_\alpha(\mathcal{T}, d, \{1\})$ does not.

Finally, we proved that in this first case the formula (4.12) is also true for the tree $\mathcal{T}$.

**b) Root with two Children or more**




The idea here is roughly the same as in the previous case, but the symmetry argument is no longer required, since here the new terms in $e^{-\gamma_0 d_0}$ will not combine as they did earlier.

By induction, we suppose the formula true for all the subtrees $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(p)}$. We therefore have for all $i \in \{1, \ldots, p\}$:

$$q_\alpha(\mathcal{T}^{(i)}, d - (l_0)) = \Gamma_{T^{(i)}} \sum_{C \text{ cut of } \mathcal{T}^{(i)}} h_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0), C) \quad .$$

In this case, the inductive formula (4.6) gives

$$q_\alpha(\mathcal{T}, d) = \int_{l_0=0}^{d_0} \gamma_0 e^{-\gamma_0 l_0} \left[ \prod_{i=1}^{p} q_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0)) \right] dl_0 \quad .$$

We already have that $\gamma_0 \prod_{i=1}^{p} \Gamma_{T^{(i)}} = \Gamma_T$. Therefore we get:

$$q_\alpha(\mathcal{T}, d) = \Gamma_T \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \left[ \prod_{i=1}^{p} \sum_{C \text{ cut of } \mathcal{T}^{(i)}} h_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0), C) \right] dl_0 \quad .$$

The product of sums can be easily expanded by noticing that each $p$-tuple $(C_1, \ldots, C_p)$ of cuts of the trees $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(p)}$ forms exactly a partition of the cut $C = \bigcup_{i=1}^{p} C_i$ of the forest $\mathcal{F}_0 = (\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(p)})$. We therefore have a bijection between the cuts $C$ of $\mathcal{F}_0$ and the $p$-tuple of cuts $(C_1, \ldots, C_p)$ of $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(p)}$. We can then write that:

$$\begin{aligned} q_\alpha(\mathcal{T}, d) &= \Gamma_T \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \left[ \sum_{C \text{ cut of } \mathcal{F}_0} \prod_{i=1}^{p} h_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0), C_i) \right] dl_0 \\ &= \Gamma_T \sum_{C \text{ cut of } \mathcal{F}_0} \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \prod_{i=1}^{p} h_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0), C_i) dl_0 \quad . \end{aligned}$$

Let us then consider a cut $C$ of $\mathcal{F}_0$. As we did in the first case, we introduce the lighter notation $h_\alpha^{(i)}(C_i) = h_\alpha(\mathcal{T}^{(i)}, d^{(i)} - (l_0), C_i)$ and for $r, s$ and $t$ as well, such that $h_\alpha^{(i)}(C_i) = r^{(i)}(C_i) s^{(i)}(C_i) t^{(i)}(C_i)$. Here again, we notice that the variable $l_0$ does not appear in $r^{(i)}(C_i)$ nor $t^{(i)}(C_i)$. We can then take these two terms out of the integral:

$$\int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \prod_{i=1}^{p} h_\alpha^{(i)}(C_i) dl_0 = \prod_{i=1}^{p} r^{(i)}(C_i) t^{(i)}(C_i) \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \prod_{i=1}^{p} s^{(i)}(C_i) dl_0 \quad .$$

By looking back at the definition of $t$, and since $l_0$ does not appear in $t^{(i)}(C_i)$, we have $t^{(i)}(C_i) = t(\mathcal{T}^{(i)}, d^{(i)} - (l_0), C_i) = t(\mathcal{T}^{(i)}, d^{(i)}, C_i)$, and therefore:

$$\begin{aligned} \prod_{i=1}^{p} t^{(i)}(C_i) &= \prod_{i=1}^{p} \prod_{j \in C_i} t_j(\mathcal{T}^{(i)}, d^{(i)}, C_i) = \prod_{i=1}^{p} \prod_{j \in C_i} t_j(\mathcal{F}_0, d, C) \\ &= \prod_{j \in C} t_j(\mathcal{F}_0, d, C) = t(\mathcal{F}_0, d, C) \quad . \end{aligned}$$

The fact that $t_j(\mathcal{T}^{(i)}, d^{(i)}, C_i) = t_j(\mathcal{F}_0, d, C)$ comes from the fact that $t_j(\mathcal{F}_0, d, C)$ depends only on the node $j$ and its future.

By looking at the definition of $r$, we also deduce that:

$$\prod_{i=1}^{p} r^{(i)}(C_i) = \prod_{i=1}^{p} \prod_{j \in \mathbf{Past}(\mathcal{T}^{(i)}, C_i)} \frac{1}{\gamma_j - \sum\limits_{\substack{k \in C_i \\ j \ll k}} \gamma_k} = \prod_{j \in \mathbf{Past}(\mathcal{F}_0, C)} \frac{1}{\gamma_j - \sum\limits_{\substack{k \in C \\ j \ll k}} \gamma_k} = r(\mathcal{F}_0, d, C) \quad ,$$

where we recall that $\mathbf{Past}(\mathcal{T}^{(i)}, C_i)$ is the past of the cut $C_i$ in the tree $T^{(i)}$, and $\mathbf{Past}(\mathcal{F}_0, C)$ is the past of the cut $C$ in the sub-forest $\mathcal{F}_0$. We especially pay attention to the fact that indeed $\{k \in C_i \mid j \ll k\} = \{k \in C \mid j \ll k\}$.

We then focus on the integral:

$$\int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \prod_{i=1}^{p} s^{(i)}(C_i) dl_0 = \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \prod_{i=1}^{p} \prod_{j \in C_i} \frac{e^{-\gamma_j (d_j - l_0)}}{\prod_{\substack{k \sim j \\ k \neq j}} (\gamma_k - \gamma_j)} dl_0$$

$$= \int_{l_0=0}^{d_0} e^{-\gamma_0 l_0} \prod_{j \in C} \frac{e^{-\gamma_j (d_j - l_0)}}{\prod_{\substack{k \sim j \\ k \neq j}} (\gamma_k - \gamma_j)} dl_0 = e^{-\gamma_0 d_0} \frac{s(\mathcal{F}_0, d - (d_0), C)}{\left(\sum_{j \in C} \gamma_j\right) - \gamma_0} + \frac{s(\mathcal{F}_0, d, C)}{\gamma_0 - \sum_{j \in C} \gamma_j} \quad ,$$

the last equality having been already seen in the first case. Putting all back together we get

$$q_\alpha(\mathcal{T}, d) = \Gamma_T \sum_{\substack{C \text{ cut of } \mathcal{T} \\ C \neq \{0\}}} \left[ e^{-\gamma_0 d_0} \frac{h_\alpha(\mathcal{F}_0, d - (d_0), C)}{\left(\sum_{j \in C} \gamma_j\right) - \gamma_0} + \frac{r(\mathcal{F}_0, d, C) s(\mathcal{F}_0, d, C) t(\mathcal{F}_0, d, C)}{\gamma_0 - \sum_{j \in C} \gamma_j} \right] \quad .$$

since the cuts of $\mathcal{T}$ are the cuts of $\mathcal{F}_0$ plus the cut $\{0\}$, and because we already noticed in the first case that $r(\mathcal{F}_0, d, C) = r(\mathcal{F}_0, d - (d_0), C)$ and $t(\mathcal{F}_0, d, C) = t(\mathcal{F}_0, d - (d_0), C)$. We then notice that for any cut $C \neq \{0\}$, we have $\dfrac{r(\mathcal{F}_0, d, C)}{\gamma_0 - \sum_{j \in C} \gamma_j} = r(\mathcal{T}, d, C)$, $s(\mathcal{F}_0, d, C) = s(\mathcal{T}, d, C)$, and $t(\mathcal{F}_0, d, C) = t(\mathcal{T}, d, C)$. Therefore:

$$q_\alpha(\mathcal{T}, d) = \Gamma_T \left[ e^{-\gamma_0 d_0} \sum_{C \text{ cut of } \mathcal{F}_0} \frac{h_\alpha(\mathcal{F}_0, d - (d_0), C)}{\left(\sum_{j \in C} \gamma_j\right) - \gamma_0} + \sum_{\substack{C \text{ cut of } \mathcal{T} \\ C \neq \{0\}}} h_\alpha(\mathcal{T}, d, C) \right] \quad .$$

Finally, since $s(\mathcal{T}, d, \{0\}) = e^{-\gamma_0 d_0}$ and $r(\mathcal{T}, d, \{0\}) = 1$, we see that:

$$e^{-\gamma_0 d_0} \sum_{C \text{ cut of } \mathcal{F}_0} \frac{h_\alpha(\mathcal{F}_0, d - (d_0), C)}{\left(\sum_{j \in C} \gamma_j\right) - \gamma_0} = e^{-\gamma_0 d_0} t(\mathcal{T}, d, \{0\}) = h_\alpha(\mathcal{T}, d, \{0\}) \quad ,$$

and we finally prove that the formula (4.12) holds also for the tree $\mathcal{T}$.

To conclude, the induction being now proved in both cases, we conclude that our formula is true for any tree $\mathcal{T}$. The proof of the expression of $\xi_\alpha(l|d)$ follows a similar but slightly more complicated reasoning and is spared (or left) to the reader.

# Chapter 5

# Inverse Problems in Bandwidth Sharing Networks

## 5.1 Introduction

The previous chapters studied some inverse problems in queueing theory. In this chapter, we explore inverse problems in bandwidth sharing networks.

Queueing theory studies networks at a detailed packet level mechanism, and predicts packet level statistics (*e.g.* the distribution of packet delays). When seen as inverse problems in queueing theory, active Internet probing uses these predicted statistics to infer quantities of interest. Whilst fruitful, this approach has two main weaknesses:

1. it requires to measure these packet level statistics, which (depending on the nature of the statistic) can be difficult or require specific equipment (*e.g.* DAG cards);

2. it is difficult to introduce the natural feedback from TCP into queueing theory. This means in particular that the sending of probes can't follow the TCP protocol, and the probes hence are packets dedicated to the measure, which carry no useful data on the network.

From a practical point of view, it would be hence ideal to perform network tomography based on TCP flow measurements. It would allow in some cases to use already existing TCP flows, reaching hence the least intrusive possible measure. In the other case, the advantage of TCP flows is that they are easy to set up, and that flow statistics are easier to collect than packet level statistics.

In this chapter, we explore one possible way to perform such a tomography. We assume that the network behaves as a bandwidth sharing network, allocating (in a distributed or centralized way) the bandwidth to each flow such as to maximize a known utility function (see section 1.3). In particular, it has been shown that the bandwidth sharing resulting from TCP protocol is approximately an oscillation around the allocation of a bandwidth sharing network with given parameters.

What can be an inverse problem in bandwidth sharing networks? In the communication network context of bandwidth sharing networks, consider one user that starts downloading a file. This user can measure its long-term bandwidth, which is proportionally fair. Is it possible from this long-term bandwidth to estimate the server capacities along the path, and the number of competitors, *i.e.* the number of competing flows? Assume that the user knows the topology along the path (*e.g.* using Traceroute), and knows the utility function used to share the bandwidth. The inversion is then possible, if the user is allowed to open several TCP connections and measure the bandwidth allocated to individual connection for any number of connections. In the context of governmental budget (see section 1.3.2), the same question arises in a natural form: assume that you have some "union" of people of the same class. This could of course be a union of workers or managers, but the model would also make sense if some of your classes were firms or research labs, and one class could then be the association of all firms of one type (as the Medef or UIMM in France, or the association "Sauvons la Recherche"). Letting their class population change, this association could observe the "wealth" increase its members have access to in each case. Its aim is then to deduce the specific budget allocated to each levers (the capacities $(C_1, \ldots, C_k)$, as well as the population of competing classes for the welfare. The advantage in that context is that one might consider rational population numbers: welfare has a long tradition of fractional shares, where some people might have right to only half the allocation and count only as half a person.

We restricted the prober actions in the two previous examples. He can only change his population number (*i.e.* open new connection), and observe his allocated bandwidth (*i.e.* measure his long-term rate). With these restrictions, there is no natural inverse problem for some embodiment of bandwidth sharing networks. For example, considering the network as in the production context, there is no analogy for class populations, which are considered as 1 for all classes. However, one could consider different inverse problems in that case. Assume that one might be able to change the resources (manpower, supplies and equipment), and observe then the total production. Is it possible then to deduce the needs for each class of production, *i.e.* the route of the class? Another inverse problem could be as follows: assume that the "prober" is a client of one company, which keeps secret commercial agreements with its clients, and has limited unknown resources for production. The prober can then change the utility of different products by changing the prices he is willing to pay for them, and can then observe how much "products" the company is willing to sell him, *i.e.* his allocated rate. Assuming the the company maximizes its gains in all cases, is it possible for the prober to deduce the resources of the company and the utility of other products (*i.e.* the prices that competitors are willing to pay)? These last two examples showed that inverse problems for bandwidth sharing networks have a different meaning for different models. In order to focus on our primary context of communication networks, we will voluntarily restrict as above the allowed action of the prober, such as to keep a framework that corresponds to end-to-end measurement in communication networks.

With this restriction in mind, we can describe inverse problems as follows: bandwidth

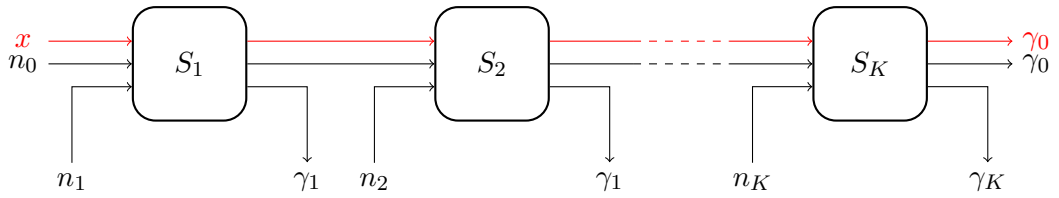



Figure 5.1: An example of path.

sharing networks describe the bandwidth allocation or the welfare allocation (which we will denote as the evolution of the system), depending on the input of the problem (network topology, utility function, capacities, flow numbers, etc.). Assume that one person, called the "prober", can observe (part of) the bandwidth allocation, possibly under different class populations, the topology and utility being static and known by the prober. What can the prober deduce about the system, from his only end-to-end observations? Is it possible to infer the server capacities and flow numbers, without any knowledge of the internal network, except the topology? When this is possible, how can the prober proceed? Is there a best way to do it? What is the minimum set of measurement that are needed to proceed to such an inference? Which are the needed observations? How many (and which repartition) probing intensities are required? Which parameters can't be deduced from observations? They are many problems that arise. We won't answer to all these questions, but focus on the identifiability question: on two simple, but generic examples, we will specify which quantities can be inferred, and present a method for the inference when it is possible.

This chapter is organized as follows: in section 5.2, we investigate the case of a network with $K$ servers in tandem. Section 5.3 studies the case of the simplest network that does not consist of a single path. This "triangle" network consists of three servers, which are pair-wise connected. Cross-traffic is considered in the most general way. We summarize these early results in section 5.4.

## 5.2 The static single path case

In this section, we consider only static networks, where the number of users in each class is fixed. Users may not enter or leave the system.

Consider a single static path, as depicted in Fig. 5.1. The path consists of $K$ servers ($S_1$, $\ldots$, $S_k$) in series, where the server $S_j$ has capacity $C_j$. There are $K+1$ class of users: users of class 0 use the whole path, and users of class $i$ ($1 \leq i \leq K$) enter the path just before server $S_i$, and exit the path after server $S_i$. Each class $i$ has $n_i$ users, and each of these users receives a bandwidth equal to $\gamma_i$. In addition to this static system, there are $x$ probes using the whole path, which receive a bandwidth equal to $\gamma_0$.

### 5.2.1 Direct equation

This setup is similar to the case of section 1.3.3, where the class 0 contains now $n_0 + x$ users. We recall here briefly the results from section 1.3.3.

Denoting by $U_{\boldsymbol{w},\alpha}$ the $(\boldsymbol{w}, \alpha)$ fairness[55], as defined in (1.19), we have the following utility:

$$U_\alpha(\boldsymbol{\gamma}) = \frac{1}{1-\alpha}\left(w_0(n_0+x)\gamma_0^{1-\alpha} + \sum_{i=1}^{K} w_i n_i \left(\frac{C_i - (n_0+x)\gamma_0}{n_i}\right)^{1-\alpha}\right) \quad , \tag{5.1}$$

where for simplicity of notation, we abusively define when $n_i = 0$

$$n_i\left(\frac{C_i - (n_0+x)\gamma_0}{n_i}\right)^{1-\alpha} = \begin{cases} 0 & \text{if} C_i - (n_0+x)\gamma_0 \geq 0 \\ -\infty & \text{otherwise} \end{cases} .$$

#### Maximum throughput

If $\sum_{i=1}^{K} w_i \mathbb{1}_{n_i>0} > w_0$, the maximum utility allocation in such a case is

$$\begin{cases} \gamma_0 = 0 \\ \gamma_i = \mathbb{1}_{n_i>0}\frac{C_i}{n_i} \end{cases} . \tag{5.2}$$

Otherwise, it is

$$\begin{cases} \gamma_0 = \min_{1\leq j\leq K} \frac{C_j}{n_0+x} \\ \gamma_i = \frac{C_i - \min_{1\leq j\leq K} C_j}{n_i} \end{cases} . \tag{5.3}$$

#### Max-min allocation

The only max-min allocation for this network is

$$\begin{cases} \gamma_0 = \min_{1\leq i\leq K} f_i(x) \\ \gamma_i = \frac{C_i - (n_0+x)\gamma_0}{n_i} \end{cases} , \tag{5.4}$$

where we define for all $i$ $f_i(x) = \frac{C_i}{n_0+x+n_i}$.

#### Other $\alpha$-fair allocations

The parameter $\alpha$ is now strictly positive and finite. The $(\boldsymbol{w}, \alpha)$-fair allocation hence verifies the following stationary equation:

$$\frac{w_0}{\gamma_0^\alpha} = \sum_{i=1}^{K} w_i \mathbb{1}_{n_i>0}\left(\frac{n_i}{C_i - (n_0+x)\gamma_0}\right)^\alpha \quad , \tag{5.5}$$

[55] We will abusively write $U_1(\boldsymbol{\gamma}) = \sum_{s\in\mathcal{S}} \frac{\gamma_s^{1-\alpha}}{1-\alpha}$ with $\alpha = 1$, instead of $U_1(\boldsymbol{\gamma}) = \sum_{s\in\mathcal{S}} \log(\gamma_s)$. This is not correct formally, but computations (and in particular differentiation) remains valid, as well as the final results.

which we are not able to solve in a general case. However, a solution can be found in the particular case when all servers have the same capacity $C$. The $(\boldsymbol{w}, \alpha)$-fair allocation then reads

$$\begin{cases} \gamma_0 = \frac{C}{n_0+x+\tilde{n}} \\ \gamma_i = \frac{\tilde{n}}{\tilde{n}+n_0+x}\frac{C}{n_i} \end{cases}, \tag{5.6}$$

where $\tilde{n}$ is the "weighted $\alpha$ sum" $\left(\frac{\sum_{i=1}^{K} w_i n_i^\alpha}{w_0}\right)^{\frac{1}{\alpha}}$.

### 5.2.2 The inverse problem

#### Maximum throughput allocation

If $w_0 < \sum_{1\leq i\leq K} w_i \mathbb{1}_{n_i>0}$, the inverse problem in this case is severely ill-posed. The bandwidth allocation is independent of the probing intensity, and the bandwidth allocated to the probing path is null. Hence, it is easy to identify such an allocation policy, as it is the only one among the considered policies that verifies any of the two above properties. Unfortunately, the fact that the probing path gets a null bandwidth allocation doesn't allow to deduce any parameter. When observing the bandwidth allocated to users of class $i$, we can deduce the ratio $\frac{C_i}{n_i}$, but we can't identify individually these values. This is the worst case scenario for inference: all allocated bandwidths are independent of each other and of the probing intensity.

In the other case, the allocation is as in (5.3), and two observation points $(x_i, \gamma_0(x_i))$ are sufficient to determine $n_0$ and $\min_{1\leq j\leq K} C_j$ as follows (assuming $x_1 < x_2$):

$$\begin{aligned} n_0 &= \frac{x_2\gamma_0(x_2) - x_1\gamma_0(x_1)}{\gamma_0(x_1) - \gamma_0(x_2)} \\ \min_{1\leq j\leq K} C_j &= \frac{(x_2 - x_1)\gamma_0(x_1)\gamma_0(x_2)}{\gamma_0(x_2) - \gamma_0(x_1)} \end{aligned}.$$

#### Max-min allocation

One can observe $\gamma_0(x) = \min_{1\leq i\leq K} f_i(x)$. Several quantities are therefore immediately non-identifiable. First, one can identify only the sums $n_0 + n_i, i \geq 1$, and not individually all $n_i, i \geq 0$. Second, only the servers which are the bottleneck for some probing intensity can have their capacity and cross-traffic intensity identified.

Before going further, it will be useful to state the two following lemmas:

**Lemma 5.2.1.** *Two functions $f_i(x)$ and $f_j(x)$ intersecting in more than 2 points on the real positive line are identical, and $C_i = C_j$ and $n_0 + n_i = n_0 + n_j$.*

*Proof.* Note that $f_i(x) - f_j(x) = \frac{C_i(n_0+n_j)-C_j(n_0+n_i)+x(C_i-C_j)}{(n_0+n_i+x)(n_0+n_j+x)}$, hence $f_i(x) = f_j(x)$ is equivalent to $C_i(n_0 + n_j) - C_j(n_0 + n_i) + x(C_i - C_j)$. This system admits at most one solution unless it is degenerate, which means $C_i = C_j$ and $C_i(n_0 + n_j) - C_j(n_0 + n_i)$. The last equality is easily deduced from these 2 last equations. □

**Lemma 5.2.2.** *Two functions $f_i(x)$ and $f_j(x)$ are tangent on the real positive line if and only if they are identical.*

*Proof.* Assume that $f_i$ and $f_j$ are tangent at $x$. Then $f_i(x) = f_j(x)$ and $f_i^{'}(x) = f_j^{'}(x)$, which we can rewrite $\frac{C_i}{n_0+n_i+x} = \frac{C_j}{n_0+n_j+x}$ and $\frac{-C_i}{(n_0+n_i+x)^2} = \frac{-C_j}{(n_0+n_j+x)^2}$. It follows directly that $n_0 + n_i + x = n_0 + n_j + x$, hence $n_0 + n_i = n_0 + n_j$, $C_i = C_j$ and both functions are identical. □

Assume now that server $j$ is at the minimum when probing at the intensities belonging to the "minimum set" $X_j$, *i.e.* $\forall x \in X_j, \gamma_0(x) = f_j(x)$. Then straightforward computations leads to

$$\forall (x_1, x_2), \quad (x_1, x_2) \in X_j^2 \Leftrightarrow \begin{cases} \frac{x_2\gamma_0(x_2) - x_1\gamma_0(x_1)}{\gamma_0(x_1) - \gamma_0(x_2)} & = n_0 + n_j \quad \text{and} \\ \frac{(x_2 - x_1)\gamma_0(x_1)\gamma_0(x_2)}{\gamma_0(x_1) - \gamma_0(x2)} & = C_j \end{cases} . \tag{5.7}$$

We can show that these minimum sets $X_j$ are convex sets. Let $x_1 < x_3$ be two elements of $X_j$. Let $x_2 \in [x_1, x_3]$. Assume that $x_2 \notin X_j$. Then $\exists k \neq j$ s.t.:

$$\begin{aligned} f_j(x_1) \leq & f_k(x_1) \\ f_j(x_2) > & f_k(x_2) \\ f_j(x_3) \leq & f_k(x_3) \quad . \end{aligned}$$

The functions $f_j(.)$ and $f_k(.)$ are continuous, which implies from the intermediate value theorem that $\exists x_1 \leq x_4 \leq x_2$ and $x_2 \leq x_5 \leq x_3$ such that $f_j(x_4) = f_k(x_4)$ and $f_j(x_5) = f_k(x_5)$. This means that they intersect in two points and must be identical functions according to lemma 5.2.1, and hence $x_2 \in X_j$.

The following theorem finally allows us to conduct the inversion step:

**Theorem 5.2.3.** *Assume a set of observation points $(x_i, \gamma_0(x_i)), i = 1, \ldots, N$, stemming from a max-min bandwidth sharing path, with capacities $(C_1, \ldots, C_K)$ and number of flows $(n_0, n_1, \ldots, n_K)$. If there is a subset $Y$ of $X$ such that $|Y| \geq 3$ and $\exists (C, n) \in \mathbb{R}^2, \forall x \in Y, \gamma_0(x) = \frac{C}{n+x}$, then there exists a server $S_j$ such that $Y \subset X_j$, and hence $C_j = C$ and $n_0 + n_j = n$, and any observation point $(x_i, \gamma_0(x_i))$ such that $\min Y \leq x_i \leq \max Y$ verifies also $\gamma_0(x_i) = f_j(x)$.*

*Proof.* Let $x_1 < x_2 < x_3$ be three points of $Y$. By assumption of the max-min allocation, there exists a server $j$ with capacity $C_j$ and number of flows $n_0 + n_j$ such that $\gamma_0(x_2) = f_j(x_2)$ and $\forall i \neq j, f_i(x_2) \geq \gamma_0(x_2)$. By Lemma 5.2.2, if the functions $f_j(.)$ and $\frac{C}{n+x}$ are tangent, the results is immediate. Otherwise, there exists $a$ such that $f_j(x_2 + a) < \frac{C}{n+x_2+a}$. Assume for simplicity that $a > 0$. But since $x_2$ and $x_3$ belongs to $Y$, we have by definition that $f_j(x_2) = \gamma_0(x_2) = \frac{C}{n+x_2}$ and $f_j(x_3) \geq \gamma_0(x_3) = \frac{C}{n+x_3}$. It follows from the intermediate value theorem that $\exists x_4 \in [x_2 + a, x_3]$ s.t. $f_j(x_4) = \frac{C}{n+x_4}$, and we have two intersection points $x_2$ and $x_4$. Lemma 5.2.1 then concludes that $C = C_j$ and $n = n_0 + n_j$.

The last part follows immediately from the fact that the minimum sets are convex. □

The inversion step can hence be summarized as follows: increase the set of measurements until you can find such subset $Y$ of cardinality greater than 3 ($C$ and $n$ can be found using (5.7) ). It is useless to try additional probing intensities that are already bounded by two elements of such a subset. Since there is a finite number of servers and an infinite number of available probing intensities, there will be finally at least one such set. Unfortunately, we can't know in advance whether there will more than one $Y$ set, and we have no information for measurement points that can't be linked to at least two other measurement points. Pairs of these "single" point might be two points belonging to the same minimum set $X_j$, and therefore allow us to retrieve the values of $C_j$ and $n_0 + n_j$. But they could also belong to two different minimum sets $X_j$ and $X_k$.

### Other $\alpha$-fair allocations

**Identical capacities along the path:** We first consider the case where all servers have the same capacity $C$, as it is the only "generic" case where we can fully solve the direct problem. We recall the solution (5.6): $\gamma_0(x) = \frac{C}{n_0+\tilde{n}+x}$, where $\tilde{n}$ is defined as $\tilde{n} = \left(\sum_{i=1}^{K} n_i^\alpha\right)^{\frac{1}{\alpha}}$. Using (5.7), we can fully inverse the problem iff we have two different measure intensities $x_1$ and $x_2$, leading to:

$$\begin{cases} n_0 + \tilde{n} & = \frac{x_2\gamma_0(x_2) - x_1\gamma_0(x_1)}{\gamma_0(x_1) - \gamma_0(x_2)} \\ C & = \frac{(x_2 - x_1)\gamma_0(x_1)\gamma_0(x_2)}{\gamma_0(x_1) - \gamma_0(x2)} \end{cases} . \tag{5.8}$$

**General capacities:** In this case, we are not able to predict what the bandwidth allocation is. However, the identifiability problems disappear, and we will be able to infer all capacities and flow numbers with mild assumptions. Recall that (5.5) is valid in this case: we can rewrite it as

$$w_0 \prod_{i=1}^{K} (C_i - (n_0 + x)\gamma_0(x))^\alpha = \gamma_0(x)^\alpha \sum_{i=1}^{K} w_i n_i^\alpha \prod_{j \neq i} (C_j - (n_0 + x)\gamma_0(x))^\alpha \quad . \tag{5.9}$$

From this equation, it is obvious that the flow numbers $n_i$ and weights $w_i$ are not independently identifiable. They appear only within their product $w_i n_i^\alpha$, and as neither $n_i$, nor $w_i$ nor $\alpha$ can be changed by the prober, the best that one will be able to identify is this product $w_i n_i^\alpha$. As weights are defined up to a (common) multiplicative constants, we can assume without loss of generality that $w_0 = 1$.

We assume that $\alpha$ is an integer, and we can define constant values $(a_{i,j})_{0 \leq i \leq j \leq \alpha K}$, which depend only on the capacities $(C_1, \ldots, C_K)$, the flow numbers $(n_0, n_1, \ldots, n_K)$ and the weights $\boldsymbol{w}$, such that the equation now reads:

$$\sum_{0 \leq i \leq j \leq \alpha K} a_{i,j} x^i \gamma_0(x)^j = 0 \quad . \tag{5.10}$$

There are strong relations between the capacities $C_i$ and flow numbers $n_i$ on one side, and the polynomial coefficients $a_{i,j}$ on the other side. We will see how the knowledge of the polynomial coefficients determines the capacities and flow numbers. First, the right-hand side of (5.9) can only have terms with degrees in $\gamma_0(x)$ strictly greater that degrees in $x$. On the left-hand sides, the only terms that will have identical degrees in $\gamma_0(x)$ and $x$ are the one stemming from the expansion of $\prod_{i=1}^{K}(C_i - x\gamma_0(x))^\alpha$. Hence, we have that

$$\prod_{i=1}^{K}(C_i - x\gamma_0(x))^\alpha = \sum_{i=0}^{\alpha K} a_{i,i}x^i\gamma_0(x)^i \quad ,$$

and $(C_1, \ldots, C_n)$ are the roots of the polynomial $\sum_{i=1}^{\alpha K} a_{i,i}X^i$. The (identical) weight of the probes and flows of class 0 can then be identified (this is for example the leading coefficient $a_{\alpha K,\alpha K}$, or the ratio between $a_{0,0}$ and $\prod C_i$).

Second, one can realize with careful computations that:

$$\begin{aligned} a_{l,l} &= (-1)^l \sum_{\substack{j_1+j_2+\cdots+j_K=l \\ j_k\leq\alpha}} \prod_{k=1}^{K} \binom{\alpha}{j_k} C_k^{\alpha-j_k} \\ a_{l,l+\alpha} &= n_0^\alpha \binom{l+\alpha}{\alpha} a_{l+\alpha,l+\alpha} \\ &\quad + (-1)^{1+l}\sum_{i=1}^{K} w_i n_i^\alpha \sum_{\substack{j_1+\cdots+j_{i-1}+j_{i+1}+\cdots+j_K=l \\ j_k\leq\alpha}} \prod_{k\neq i} \binom{\alpha}{j_k} C_k^{\alpha-j_k} \quad . \end{aligned} \tag{5.11}$$

Remark first that in the expansion of $\prod(C_i-(n_0+x)\gamma_0(x))$, each term has a power in $\gamma_0(x)$ greater (or equal) to the power in $x$. Hence, the contribution to $a_{l,l}$ comes only from terms in the left-hand side of (5.9). On top of that, we are interested only in terms that choose $C_k$ in all factors except $l$ (which explains the sum, as each $C_k$ can be not chosen only $\alpha$ times). Now, there is $\binom{\alpha}{\alpha-j_k} = \binom{\alpha}{j_k}$ possibilities to choose $\alpha - j_k$ times $C_j$ among $\alpha$. Similarly, the contribution of the left-hand term of (5.9) to $a_{l,l+\alpha}$ contains only terms that choose $l$ times the factor $x\gamma_0(x)$, $\alpha$ times the factor $n_0\gamma_0(x)$ and $C_k$ the rest of the times. Hence, there are $a_{l+\alpha,l+\alpha}$ possibilities to choose all the $C_k$, and then $\binom{l+\alpha}{\alpha}$ possibilities to choose the $n_0\gamma_0(x)$ terms among the remaining terms. The contribution of the right-hand terms of (5.9) is as follows: $x\gamma_0(x)$ is chosen $l$ times in the expansion, and all other choices are $C_k$. Hence, the minus appears $l$ times, and the sum is equivalent to the one for $a_{l,l}$, except that the factor $C_i - (n_0 + x)\gamma_0(x)$ is not present.

The set of equations (5.11) consists of $\alpha K-\alpha+1$ linear equations with $K+1$ unknown values $(n_0^\alpha, w_1n_1^\alpha, \ldots, w_Kn_K^\alpha)$. For $\alpha > 1$ and $K > 1$, it is hence possible to determine the values of the flow numbers from these equations by solving the associated linear system. It is easy to see that the system will be regular if all capacities are pairwise different. When capacities of server $S_i$ and $S_j$ are equal, the only information we will be allow to determine will be the "weighted $\alpha$-power sum" $\left(w_in_i^\alpha + w_jn_j^\alpha\right)$, which is similar to the identical

capacities case. The case $K = 1$ falls into the "identical capacity" case. Finally, in the case when $\alpha = 1$, we have $K$ equations with $K + 1$ unknowns. It allows us to express all the flow capacities as an affine function of $n_0$. We can then compute the coefficient $a_{0,2}$ which is:

$$a_{0,2} = n_0^2 \sum_{\{i,j\}} \prod_{\substack{k\neq i \\ k\neq j}} C_k + \sum_{i=1}^{K} w_i n_i \sum_{j\neq i} n_0 \prod_{\substack{k\neq i \\ k\neq j}} C_k \quad . \tag{5.12}$$

Using the previous affine functions, we rewrite it as a second order polynomial, which we can solve to determine $n_0$, and hence all the flow numbers.

It remains to show how the polynomial coefficients $a_{i,j}$ can be estimated. Recall that equation (5.10) holds, for all probing intensities. Assume now that we have access to $N$ measurement points $(x_i, \gamma_0(x_i))_{i=1,\ldots,N}$ for $N$ different probing flow numbers $x_1, \ldots, x_N$. We can now rewrite (5.10) in a vector form, as follows:

$$P \times A = (-1)^{1+\alpha K} Y \quad , \tag{5.13}$$

where $P$ is the $N \times \left(\frac{(\alpha K+1)(\alpha K+2)}{2} - 1\right)$ matrix, whose element $k, (i,j)$ is $x_k^i \gamma_0(x_k)^j$, $A$ the $\left(\frac{(\alpha K+1)(\alpha K+2)}{2} - 1\right) \times 1$ column matrix whose element of line $(i,j)$ is $a_{i,j}$ (except $a_{K,K}$), and $Y$ the $N \times 1$ column matrix whose element $k$ is $x_k^{\alpha K} \gamma_0(x_k)^{\alpha K}$ (we force here the normalization $a_{K,K} = 1$). If $N = \frac{(\alpha K+1)(\alpha K+2)}{2} - 1$ and $P$ is full-rank, there is unique solution $A$ satisfying (5.13). We don't have any proof that $P$ is full-rank; however, in all practical case we simulated, $P$ was nearly-singular, but regular. If $N > \frac{(\alpha K+1)(\alpha K+2)}{2} - 1$ and $P$ is full-rank, one can have a more robust estimation of $A$ by minimizing the error $\|P \times A + (-1)^{\alpha K} Y\|$ for some well-chosen norm $\|.\|$ (a classical choice is the norm $\|.\|_2$, which minimizes the quadratic error and leads to linear regression).

**Summary:** The inversion is possible when enough measurement points are available:

1. If $K = 1$ or if all capacities are known to be identical, use (5.8) to identify the capacity and the aggregated flow number. Individual flow numbers can't be determined.

2. If $\alpha > 1$, use (5.13) to determine the polynomial coefficient $a_{i,j}$. Find $(C_1, \ldots, C_k)$ as the roots of the polynomial $\sum_{i=0}^{\alpha K} a_{i,i} x^i \gamma_0(x)^i$. If all capacities are pairwise different, use (5.11) to estimate the individual flow numbers. Otherwise, estimate the "aggregated" flow numbers $\sum_{i:C_i=C} n_i^\alpha$ for identical capacities servers, and individual flows with (5.11).

3. If $\alpha = 1$, use (5.13) to determine the polynomial coefficient $a_{i,j}$. Find $(C_1, \ldots, C_k)$ as the roots of the polynomial $\sum_{i=0}^{\alpha K} a_{i,i} x^i \gamma_0(x)^i$. Use then (5.11) to express all cross-traffic flow numbers $(n_1, \ldots, n_K)$ (or their aggregated sum in case of identical capacities) as affine functions of $n_0$. Use these functions and (5.12) to obtain a second-degree equation, solve it to get $n_0$, hence the other flow numbers as well.

### 5.2.3 Numerical application

In this section, we focus on the case of general integer $\alpha$, which seems the most interesting from a practical point of view and the simplest from a numerical point of view.

The simulation were performed using Matlab. For any number of flows, the bandwidth allocation is computed using convex optimization tools. Based on these measurement points $(x_k, \gamma_0(x_k))$, we then estimate the coefficient $a_{k,(i,j)}$ (with a special care for the matrix inversion step), then use the Matlab "root" routine to find the roots $(C_1, \ldots, C_K)$ of the polynomial $\sum_{i=0}^{K} a_{i,i} X^i$. For $k > 1$, the estimation of $\boldsymbol{n}$ is performed with the Matlab right-side division, a routine designed for solving matrices equations of the type $AX = B$. If $\alpha = 1$, the system is not linear, and we use the "Fsolve" function, where the objective function is the vector of the equations (5.11) and, for the $\alpha = 1$ case, (5.12). In particular, "Fsolve" finds the real variables that minimizes the absolute value of the objective function, and it is not possible to restrict the solution space to integer variables (the problem is then different, much more complicated, and most likely NP-hard).

To keep things simple, we will assume that all weights $w_i$ are equal to 1 (or equivalently, that the flow population $n_i$ is rescaled).

**Proportional fairness** The case of proportional fairness is slightly singular, because the system (5.11) is lacking one equation, and (5.12) is used to determine $n_0$. Table 5.1 presents a few numerical results.

| **C** | **n** | # add. | est. A | est. **C** | est. **n** |
|---|---|---|---|---|---|
| $\begin{pmatrix}2\\1\end{pmatrix}$ | $\begin{pmatrix}0\\3\\1\end{pmatrix}$ | 0 | $\begin{bmatrix}2.0001 & -4.9996 & -0.0015\\0 & -3.0001 & 3.9997\\0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}2.0001\\1\end{pmatrix}$ | $\begin{pmatrix}-0.0004\\3.0001\\1.0002\end{pmatrix}$ |
| $\begin{pmatrix}30\\1\end{pmatrix}$ | $\begin{pmatrix}2\\5\\1\end{pmatrix}$ | 0 | $\begin{bmatrix}-1.1101 & 0.4909 & 8.8867\\0 & 0.1102 & 5.8391\\0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}-1.1102\\1\end{pmatrix}$ | $\begin{pmatrix}2.97\\-0.08\\0.08\end{pmatrix}$ |
| $\begin{pmatrix}30\\20\end{pmatrix}$ | $\begin{pmatrix}2\\5\\1\end{pmatrix}$ | 0 | $\begin{bmatrix}600.0258 & -230.272 & 16.0772\\0 & -49.9989 & 10.0095\\0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}29.99\\20.00\end{pmatrix}$ | $\begin{pmatrix}2.01\\4.99\\0.997\end{pmatrix}$ |
| $\begin{pmatrix}30\\1\end{pmatrix}$ | $\begin{pmatrix}0\\200\\0\end{pmatrix}$ | 0 | $\begin{bmatrix}51.82 & -568.6 & 1487\\0 & -31.256 & 204.2\\0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}29.49\\1.76\end{pmatrix}$ | $\begin{pmatrix}7.56\\189\\-0.01\end{pmatrix}$ |
| $\begin{pmatrix}30\\1\end{pmatrix}$ | $\begin{pmatrix}0\\200\\0\end{pmatrix}$ | 5 | $\begin{bmatrix}29.86 & -199.16 & 0.25\\0 & -30.86 & 199.06\\0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}29.86\\1.0001\end{pmatrix}$ | $\begin{pmatrix}0.0013\\199.06\\-0.0001\end{pmatrix}$ |
| $\begin{pmatrix}30\\1\end{pmatrix}$ | $\begin{pmatrix}0\\50\\0\end{pmatrix}$ | 5 | $\begin{bmatrix}0.3085 & 0.4756 & 0.0\\0 & -1.309 & -0.4756\\0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}1.0\\0.31\end{pmatrix}$ | $\begin{pmatrix}-0.4756\\0.4756\\0.0000\end{pmatrix}$ |

Table 5.1: Proportional fairness: two-server cases. The two first columns show the groundtruth. The third column indicates how many additional measurement points on top on the minimum 5 required where. The column "est. A" is a matrix whose coefficient (i,j) is the estimated $a_{i,j}$. The fifth or sixth columns show the estimation using the technique proposed.

The estimation is reasonable in the cases 1, 3, 4 and 5. It fails by a large margin in the cases 2 and 6. Comparing cases 4 and 5, we can see that additional measurement points allow a better precision in some cases. In both "failed" cases, the matrix $P$ of equation (5.13) was nearly singular. For a simple intuition, these were also the cases where a single server is the clear bottleneck for the probe path. This means that the bandwidth is shared almost as $\gamma_0(x) = \frac{C}{n+x}$, where $C$ is the capacity bottleneck server and $n$ the number of flows (outside probes) which cross that server. As $x$ grows large, the coefficient $k, (i,j)$ of $P$ is $x_k^i \gamma_0(x_k)^j \to \frac{C^j}{x^{i-j}}$, and the coefficients $a_k, (i,i) \to C^i$ are almost independent of the line index $k$. This means that the columns $(i,i)$ are almost all equivalent, and the matrix $P$ is near singular.

As shown in table 5.2, the instability is worse for the three-server case. It is still possible to get "correct" estimation as in the line 2, 3 or 4, but additional measurement points are now required in order to correctly estimate the coefficient $a_{i,j}$. The fact that the estimation is harder when one server is the clear bottleneck or is clearly overprovisioned, remains.

| **C** | **n** | # add. | est. A | est. **C** | est. **n** |
|---|---|---|---|---|---|
| $\begin{pmatrix}2\\1\\3\end{pmatrix}$ | $\begin{pmatrix}0\\3\\1\\7\end{pmatrix}$ | 0 | $\begin{bmatrix}-1.74 & 8.28 & 5.24 & -22.68\\ 0 & 5.04 & -19.63 & -3.68\\ 0 & 0 & -4.3 & 10.67\\ 0 & 0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}1.\\0.66\\2.64\end{pmatrix}$ | $\begin{pmatrix}-0.28\\0.89\\-0.29\\9.22\end{pmatrix}$ |
| $\begin{pmatrix}2\\1\\3\end{pmatrix}$ | $\begin{pmatrix}0\\3\\1\\7\end{pmatrix}$ | 5 | $\begin{bmatrix}-5.72 & 25.5 & 10.39 & -0.706\\ 0 & 10.62 & -33.87 & -9.12\\ 0 & 0 & -5.896 & 10.17\\ 0 & 0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}1.93\\1\\2.97\end{pmatrix}$ | $\begin{pmatrix}-0.32\\1.70\\0.68\\6.80\end{pmatrix}$ |
| $\begin{pmatrix}2\\1\\3\end{pmatrix}$ | $\begin{pmatrix}0\\3\\1\\7\end{pmatrix}$ | 10 | $\begin{bmatrix}-5.98 & 26.9 & 9.63 & -0.005\\ 0 & 10.98 & -35.15 & -8.46\\ 0 & 0 & -6 & 10.23\\ 0 & 0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}1.98\\1\\3.02\end{pmatrix}$ | $\begin{pmatrix}-0.29\\2.25\\0.7\\6.4\end{pmatrix}$ |
| $\begin{pmatrix}2\\5\\3\end{pmatrix}$ | $\begin{pmatrix}0\\3\\1\\7\end{pmatrix}$ | 10 | $\begin{bmatrix}-25.68 & 99.76 & 18.52 & -13.14\\ 0 & 27.45 & -67 & -9.40\\ 0 & 0 & -69.31 & 10.23\\ 0 & 0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}2.01\\4.39\\2.91\end{pmatrix}$ | $\begin{pmatrix}-0.29\\2.71\\0.5\\6.15\end{pmatrix}$ |
| $\begin{pmatrix}2\\20\\3\end{pmatrix}$ | $\begin{pmatrix}0\\3\\1\\7\end{pmatrix}$ | 10 | $\begin{bmatrix}-13.13 & 99.21 & -147.9 & -150.62\\ 0 & 17.04 & -85.15 & 62.77\\ 0 & 0 & -7.24 & 17.8\\ 0 & 0 & 0 & 1\end{bmatrix}$ | $\begin{pmatrix}2.00\\3.16\\2.08\end{pmatrix}$ | $\begin{pmatrix}1.53\\2.11\\7.8\\12.49\end{pmatrix}$ |

Table 5.2: Proportional fairness: three-server cases. The columns are organized as in table 5.1.

**Other fairnesses** The method is absolutely unstable in that case and leads to meaningless (complex) results. The reason is that it requires the at least $\frac{(\alpha K+1)(\alpha K+2)}{2} - 1$ measurement points, and the inversion of a square matrix of the same size. Note that even for the smallest case with $K = 2$ and $\alpha = 2$, the minimum size is already 14! In all cases we have tried, the matrix had a few near zero eigenvalues, and the inversion lead to unexploitable estimation of the coefficient $a_{i,j}$.

## 5.3 The static triangle network

Previous section focused, in a detailed manner, on the single source-destination path, which can hardly be called a network. We extend here the results to a non-trivial (but small) network topology: a "triangle network", as depicted in Fig. 5.2.

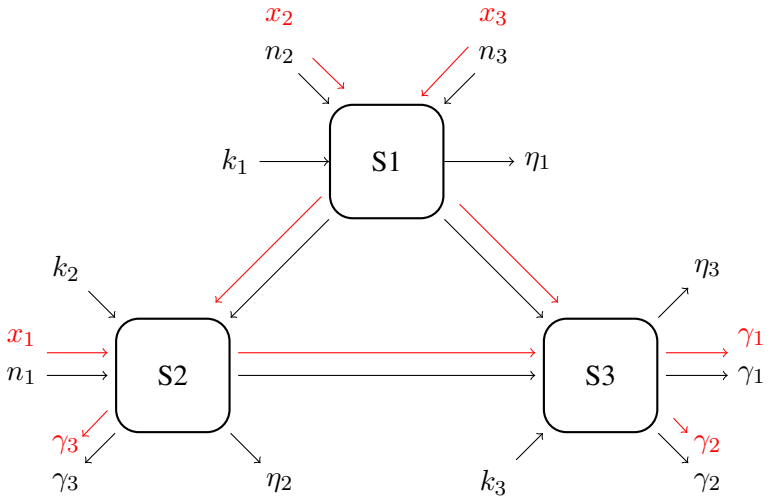



Figure 5.2: The triangle network.

The networks consists of 3 servers, each server being connected to both other servers. Server $S_i$ has a capacity $C_i$. $k_i$ flows cross the server $S_i$, and each of them gets an allocated bandwidth $\eta_i$. There are also flows using 2 servers: $n_3$ (resp. $n_2$ and $n_1$) flows use the route $(S_1, S_2)$ (resp. $(S_1, S_3)$ and $(S_2, S_3)$) and each of them gets an allocated bandwidth $\gamma_3$ (resp. $\gamma_2$ and $\gamma_1$).

Additionally, the prober can add $x_1$ (resp. $x_2$ and $x_3$) flows on the route $(S_2, S_3)$ (resp. $(S_1, S_3)$ and $(S_1, S_2)$). These flows get each the same bandwidth allocated to the route, *i.e.* $\gamma_1$ (resp. $\gamma_2$ and $\gamma_3$).

This is the setup of section 1.3.4: we recall equation (1.30), which reads:

$$\begin{aligned} U_\alpha(\boldsymbol{\gamma}, \boldsymbol{\eta}) =& (x_1 + n_1) w_1 \frac{\gamma_1^{1-\alpha}}{1-\alpha} + (x_2 + n_2) w_2 \frac{\gamma_2^{1-\alpha}}{1-\alpha} + (x_3 + n_3) w_3 \frac{\gamma_3^{1-\alpha}}{1-\alpha} \\ &+ k_1 v_1 \frac{(C_1 - (x_2 + n_2)\gamma_2 - (x_3 + n_3)\gamma_3)^{1-\alpha}}{(1-\alpha) k_1^{1-\alpha}} \\ &+ k_2 v_2 \frac{(C_2 - (x_1 + n_1)\gamma_1 - (x_3 + n_3)\gamma_3)^{1-\alpha}}{(1-\alpha) k_2^{1-\alpha}} \\ &+ k_3 v_3 \frac{(C_3 - (x_1 + n_1)\gamma_1 - (x_2 + n_2)\gamma_2)^{1-\alpha}}{(1-\alpha) k_3^{1-\alpha}} \quad . \end{aligned}$$

The partial derivatives of the utility with respect to $\gamma_i$ are null at the maximum utility, hence:

$$\frac{(x_1+n_1)w_1}{\gamma_1^\alpha} - \frac{x_1+n_1}{k_2^{1-\alpha}} \frac{k_2 v_2}{(C_2-(x_1+n_1)\gamma_1-(x_3+n_3)\gamma_3)^\alpha} - \frac{x_1+n_1}{k_3^{1-\alpha}} \frac{k_3 v_3}{(C_3-(x_1+n_1)\gamma_1-(x_2+n_2)\gamma_2)^\alpha} = 0$$

$$\frac{(x_2+n_2)w_2}{\gamma_2^\alpha} - \frac{x_2+n_2}{k_1^{1-\alpha}} \frac{k_1 v_1}{(C_1-(x_2+n_2)\gamma_2-(x_3+n_3)\gamma_3)^\alpha} - \frac{x_2+n_2}{k_3^{1-\alpha}} \frac{k_3 v_3}{(C_3-(x_1+n_1)\gamma_1-(x_2+n_2)\gamma_2)^\alpha} = 0$$

$$\frac{(x_3+n_3)w_3}{\gamma_3^\alpha} - \frac{x_3+n_3}{k_1^{1-\alpha}} \frac{k_1 v_1}{(C_1-(x_2+n_2)\gamma_2-(x_3+n_3)\gamma_3)^\alpha} - \frac{(x_3+n_3)}{k_2^{1-\alpha}} \frac{k_2 v_2}{(C_2-(x_1+n_1)\gamma_1-(x_3+n_3)\gamma_3)^\alpha} = 0$$

We can rewrite these as follows:

$$w_1 (C_2-(x_1+n_1)\gamma_1-(x_3+n_3)\gamma_3)^\alpha (C_3-(x_1+n_1)\gamma_1-(x_2+n_2)\gamma_2)^\alpha - \gamma_1^\alpha \times [k_2^\alpha v_2 (C_3-(x_1+n_1)\gamma_1-(x_2+n_2)\gamma_2)^\alpha + k_3^\alpha v_3 (C_2-(x_1+n_1)\gamma_1-(x_3+n_3)\gamma_3)^\alpha] = 0$$

$$w_2 (C_1-(x_2+n_2)\gamma_2-(x_3+n_3)\gamma_3)^\alpha (C_3-(x_1+n_1)\gamma_1-(x_2+n_2)\gamma_2)^\alpha - \gamma_2^\alpha \times [v_1 k_1^\alpha (C_3-(x_1+n_1)\gamma_1-(x_2+n_2)\gamma_2)^\alpha + v_3 k_3^\alpha (C_1-(x_2+n_2)\gamma_2-(x_3+n_3)\gamma_3)^\alpha] = 0 \quad (5.14)$$

$$w_3 (C_1-(x_2+n_2)\gamma_2-(x_3+n_3)\gamma_3)^\alpha (C_2-(x_1+n_1)\gamma_2-(x_3+n_3)\gamma_3)^\alpha - \gamma_3^\alpha \times [k_1^\alpha v_1 (C_2-(x_1+n_1)\gamma_1-(x_3+n_3)\gamma_3)^\alpha + k_2^\alpha v_2 (C_1-(x_2+n_2)\gamma_2-(x_3+n_3)\gamma_3)^\alpha] = 0$$

From these equations, one can realize that only the products $v_i k_i^\alpha$ will be identifiable. We show now how to identify these products, the server capacities $C_1$, $C_2$ and $C_3$, and the class population for the classes whose route crosses two servers.

Similarly to (5.10), it is possible to define constant values $(b_{i,j,k,l,m,n})$ (resp. $(c_{i,j,k,l,m,n})$ and $(d_{i,j,k,l,m,n})$), depending only on the flow numbers $n_i$ and $k_i$, the weights

$w_i$ and $v_i$ and the capacities $C_i$, such that the first (resp. second and third) equation reads:

$$\sum_{i+j+k\leq 2\alpha} \sum_{l\leq i} \sum_{m\leq j} \sum_{n\leq k} b_{i,j,k,l,m,n} \gamma_1^i \gamma_2^j \gamma_3^k x_1^l x_2^m x_3^n = 0 \quad . \tag{5.15}$$

The second and third equation are similar, with $b_{i,j,k,l,m,n}$ replaced by $c_{i,j,k,l,m,n}$ and $d_{i,j,k,l,m,n}$.

The knowledge of these coefficients is sufficient to estimate the server capacities $(C_1, C_2, C_3)$, the flow numbers $(n_1, n_2, n_3)$ and the products $(v_1 k_1^\alpha, v_2 k_2^\alpha, v_3 k_3^\alpha)$.

From (5.14), it is clear that

$$\sum_{i=1}^{\alpha} b_{0,0,i,0,0,i} Y^i = w_1 C_3^\alpha \left(C_2 - Y\right)^\alpha \quad .$$

and one can identify $C_2$ as the root of the polynomial $\sum_{i=1}^{\alpha} b_{0,0,i,0,0,i} Y^i$. As in the previous section, the coefficient $b_{i,j,k,l,m,n}$ are defined up to a multiplicative shift. We can hence always chose them such as the leading coefficient is $C_3^\alpha$, and consider that $w_1 = 1$.

Similarly, we have

$$\begin{aligned}
\sum_{i=1}^{\alpha} b_{0,i,0,0,i,0} Y^i &= w_1 C_2^\alpha \left(C_3 - Y\right)^\alpha \quad , \\
\sum_{i=1}^{\alpha} c_{i,0,0,i,0,0} Y^i &= w_2 C_1^\alpha \left(C_3 - Y\right)^\alpha \quad , \\
\sum_{i=1}^{\alpha} c_{0,0,i,0,0,i} Y^i &= w_2 C_3^\alpha \left(C_1 - Y\right)^\alpha \quad , \\
\sum_{i=1}^{\alpha} d_{i,0,0,i,0,0} Y^i &= w_3 C_1^\alpha \left(C_2 - Y\right)^\alpha \quad , \\
\sum_{i=1}^{\alpha} d_{0,i,0,0,i,0} Y^i &= w_3 C_2^\alpha \left(C_1 - Y\right)^\alpha \quad ,
\end{aligned}$$

which are sufficient to identify the server capacities $(C_1, C_2, C_3)$.

Moreover, it is also clear that $b_{0,1,0,0,0,0} = \alpha w_1 \times C_2^\alpha \times C_3^{\alpha-1} n_2$, $b_{0,0,1,0,0,0} = \alpha w_1 \times C_3^\alpha \times C_2^{\alpha-1} n_3$ and $c_{1,0,0,0,0,0} = \alpha w_2 \times C_1^\alpha \times C_3^{\alpha-1} n_1$, hence

$$\begin{aligned}
n_1 &= \frac{b c_{1,0,0,0,0,0}}{\alpha} C_1^{-\alpha} C_2^{1-\alpha} \\
n_2 &= \frac{b_{0,1,0,0,0,0}}{\alpha} C_2^{-\alpha} C_3^{1-\alpha} \\
n_3 &= \frac{b_{0,0,1,0,0,0}}{\alpha} C_2^{1-\alpha} C_3^{-\alpha} \quad .
\end{aligned} \tag{5.16}$$

Finally, we can also see that

$$b_{\alpha,0,0,0,0,0} = w_1 \sum_{i=0}^{\alpha} \binom{\alpha}{i}^2 C_2^i C_3^{\alpha-i} n_1^\alpha - k_2^\alpha v_2 C_3^\alpha - k_3^\alpha v_3 C_2^\alpha$$
$$c_{0,\alpha,0,0,0,0} = w_2 \sum_{i=0}^{\alpha} \binom{\alpha}{i}^2 C_1^i C_3^{\alpha-i} n_2^\alpha - k_1^\alpha v_1 C_3^\alpha - k_3^\alpha v_3 C_1^\alpha$$
$$d_{0,0,\alpha,0,0,0} = w_3 \sum_{i=0}^{\alpha} \binom{\alpha}{i}^2 C_1^i C_2^{\alpha-i} n_3^\alpha - k_1^\alpha v_1 C_2^\alpha - k_2^\alpha v_2 C_1^\alpha \quad .$$

The determinant of this linear system is $2C_1C_2C_3$, and the system is full-rank iff all capacities are non-null. Hence, we get:

$$\begin{bmatrix} v_1 k_1^\alpha \\ v_2 k_2^\alpha \\ v_3 k_3^\alpha \end{bmatrix} = \begin{bmatrix} 0 & C_3^\alpha & C_2^\alpha \\ C_3^\alpha & 0 & C_1^\alpha \\ C_2^\alpha & C_2^\alpha & 0 \end{bmatrix}^{-1} \times \begin{bmatrix} \sum_{i=0}^{\alpha} \binom{\alpha}{i}^2 C_2^i C_3^{\alpha-i} n_1^\alpha - b_{\alpha,0,0,0,0,0} \\ \sum_{i=0}^{\alpha} \binom{\alpha}{i}^2 C_1^i C_3^{\alpha-i} n_2^\alpha - c_{0,\alpha,0,0,0,0} \\ \sum_{i=0}^{\alpha} \binom{\alpha}{i}^2 C_1^i C_2^{\alpha-i} n_3^\alpha - d_{0,0,\alpha,0,0,0} \end{bmatrix} \quad . \tag{5.17}$$

Finally, it remains to show how one can estimate the polynomial coefficient values $(b_{i,j,k,l,m,n})_{j\leq\alpha,k\leq\alpha,i+j+k\leq\alpha,l\leq i,m\leq j,n\leq k}$. There are $M = \frac{5\alpha^6+51\alpha^5+209\alpha^4+441\alpha^3+506\alpha^2+300\alpha+72}{72}$ such coefficients for the single $b_{i,j,k,l,m,n}$ collection. For example, we have $M = 22$ for $\alpha = 1$ and $M = 160$ for $\alpha = 2$. Assume that the prober has access to $N \geq M - 1$ measurement points $(x_1(p), x_2(p), x_3(p), \gamma_1(p), \gamma_2(p), \gamma_3(p))_{1\leq p\leq N}$, where for the ease of notations, we abusively write $\gamma_i(k)$ for $\gamma_i(x_1(k), x_2(k), x_3(k))$. Recall that (5.15) is valid for all probing intensities $(x_1, x_2, x_3)$, and that from (5.14), we have $b_{2\alpha,0,0,2\alpha,0,0} = 1$. (5.15) now defines a linear system

$$X \times B = Y \quad , \tag{5.18}$$

where $B$ is the $M-1 \times 1$ column vector with row $(i, j, k, l, m, n)$ equal to $b_{i,j,k,l,m,n}$, Y the $N \times 1$ column vector with row $p$ equal to $-x_1(p)^{2\alpha}\gamma_1(p)^{2\alpha}$, and $B$ the $N \times M - 1$ matrix whose element $p, (i, j, k, l, m, n)$ is $x_1(p)^l x_2(p)^m x_3(p)^n \gamma_1(p)^i \gamma_2(p)^j \gamma_3(p)^k$. If $N = M - 1$ and $X$ is invertible, we can estimate $B = X^{-1} \times Y$. Otherwise, we can add points until $X$ is a full-rank matrix, and then there is a single vector $B$ which minimizes the distance $\|X \times B - Y\|_2$. In all cases that we simulated, we were able to add enough points such that $X$ is full-rank, and we conjecture that it is possible in all non-degenerate cases. However, we don't have any proof of it.

## 5.4 Summary

We have present two simple cases of bandwidth sharing networks, where the inference of parameters is theoretically possible from measurement points. The practical interest of this is not obvious: however, as bandwidth sharing networks are a common "simplified model" for real networks, it remains interesting to know what can (and can not) be deduced from

measured bandwidth allocation. In this aspect, they are more a theoretical toy example from which we might be able to deduce interesting conclusions than a rigorous analysis of the practical problem.

Whenever inference is possible, we presented a way to conduct it. These methods are most likely not the best efficient ones; they are numerically unstable , and it is obvious that for $\alpha-$ fairness with finite positive $\alpha$, they require too much measurement points (they need at least $\frac{K^2}{2}$ measurement points in the single path network to determines only $2K+1$ unknowns). It remains to find better method, that would use less measurement points and be more stable.

It is unexpected that inversion is less ambiguous in the cases where one can't compute an explicit formula for the bandwidth allocation. The "simplification" of having all capacities equal, which allows one to compute an explicit bandwidth allocation, is responsible for the ambiguity of the inverse problem.

Finally, this work is somehow preliminary. We conjecture that similar solution could be found for other topologies, the best candidate being regular trees. A general formula dealing with any network topology would be the best we can wish, but trees are a far-easier candidates because their routes are easily described.

# Bibliography


[ABC+00] A. Adams, T. Bu, T. Caceres, N.G. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S.B. Moon, V. Paxson, and D. Towsley. The use of End-to-end Multicast Measurements for Characterising Internal Network Behavior. *IEEE Communications Magazine, special issue on "Network Traffic Measurements and Experiments*, 38(5):152–158, May 2000.

[ADV07] V. Arya, N.G. Duffield, and D. Veitch. Multicast Inference of Temporal Loss Characteristics. In *IFIP Performance*, Special Issue, Performance Evaluation, Oct. 2007.

[ANO96] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase-type distributions via the em algorithm. *Scandinavian Journal of Statistics*, 23(4):419–441, Dec 1996.

[ANT01] S. Alouf, P. Nain, and D. F. Towsley. Inferring Network Characteristics via Moment-Based Estimators. In *Proc. of IEEE INFOCOM*, pages 1045–1054, 2001.

[AW87] J. Abate and W. Whitt. Transient behavior of the m/m/l queue: Starting at the origin. *Queueing Systems*, 2:41–65, 1987. 10.1007/BF01182933.

[AW88] J. Abate and W. Whitt. Transient behavior of the m/m/1 queue via laplace transforms. *Advances in Applied Probability*, 20(1):145–178, 1988.

[AW94] J. Abate and W. Whitt. Transient behavior of m/g/1 workload process. *Operations Research*, 42(4):750–764, 1994.

[BB03] F.B. Baccelli and P. Bremaud. *Elements of Queueing Theory*. Springer Verlag, Applications of Mathematics, second edition, 2003.

[BDF+09] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho. Seven years and one day: Sketching the evolution of internet traffic. In *Proc. of IEEE INFOCOM*, pages 711–719, Apr. 2009.

[BDLPT02] T. Bu, N. Duffield, F. Lo Presti, and D. Towsley. Network tomography on general topologies. *SIGMETRICS Perform. Eval. Rev.*, 30(1):21–30, 2002.

[Bey95] P. Beyssac. Pchar. `http://fgouget.free.fr/bing/index-en.shtml`, 1995.

[BJP04] T. Bonald, M. Jonckheere, and A. Proutiére. Insensitive load balancing. In *Proc. of ACM SIGMETRICS/Performance*, pages 367–377, 2004.

[BKV09] F. Baccelli, B. Kauffmann, and D. Veitch. Inverse Problems in Queueing Theory and Internet Probing. *Queueing Systems*, 63(1–4):59–107, 2009.

[BLFF96] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0. RFC 1945 (Informational), May 1996.

[BM00] T. Bonald and L. Massoulié. Impact of fairness on internet performance. In *Proc. of ACM SIGMETRICS*, pages 82–91, 2000.

[BMSV00] F. Brichet, L. Massoulié, A. Simonian, and D. Veitch. Heavy load queueing analysis with lrd on/off sources. In K. Park and W. Willinger, editors, *Self-Similar Network Traffic and Performance Evaluation*, pages 115–142. Wiley, 2000.

[BMT89] F. Baccelli, W. Massey, and D. Towsley. Acyclic fork-join queuing networks. *Journal of the ACM (JACM)*, 36(3):615–642, 1989.

[BMVB06] F. Baccelli, S. Machiraju, D. Veitch, and J. Bolot. The Role of PASTA in Network Measurement. *Proc. of ACM SIGCOMM*, 36(4):231–242, 11-15 Sep 2006.

[BMVB07] F. Baccelli, S. Machiraju, D. Veitch, and J. Bolot. On Optimal Probing for Delay and Loss Measurement. In *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 291–302, 23–26 October 2007.

[Bol93] J.-C. Bolot. End to end packet delay and loss behavior in the Internet. In *Proc. of ACM SIGCOMM*, pages 289–298, Sep. 1993.

[Bor98] Borovkov. *Mathematical Statistics*. Gordon and Breach, 1998.

[CB97] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 1997.

[CC96] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27-28:297–318, 1996.

[CCB07] A. Chen, J. Cao, and T. Bu. Network Tomography: Identifiability and Fourier Domain Estimation. In *Proc. of IEEE INFOCOM*, pages 1875–1883, 6-12 May 2007.

[CCL+04] R. Castro, M.J. Coates, G. Liang, R. Nowak, and B. Yu. Network Tomography: Recent Developments. *Statistical Science Magazine*, 19(3):499–517, August 2004.

[CDHT99] R. Caceres, N.G. Duffield, J. Horowitz, and D. Towsley. Multicast-Based Inference of Network-Internal Loss Characteristics. *IEEE Transactions on Information Theory*, 45:2462–2480, 1999.

[CFS08] R. Chertov, S. Fahmy, and N.B. Shroff. A device-independent router model. In *Proc. of IEEE INFOCOM*, pages 1642–1650, Apr. 2008.

[CHNY02] M. Coates, A. Hero, R. Nowak, and B. Yu. Internet tomography. *Signal Processing Magazine*, 19(3):47–65, May 2002.

[CMT98] K. Claffy, G. Miller, and K. Thompson. The nature of the beast: recent traffic measurements from an Internet backbone. In *INET 98*. Internet Society, July 21–24 1998.

[CN00] M.J. Coates and R. Nowak. Network Loss Inference using Unicast End-to-End Measurement. In *ITC Conference on IP Traffic, Modelling and Management*, Sep. 2000.

[CN01] M.J. Coates and R. Nowak. Network Tomography for Internal Delay Estimation. In *Proc. of IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, May 2001.

[dag] Endace Measurement Systems. `http://www.endace.com/`.

[DHLP01] N.G. Duffield, J. Horowitz, and F. Lo Prestis. Adaptive multicast topology inference. In *Proc. of IEEE INFOCOM*, volume 3, pages 1636 –1645, 2001.

[DHPT00] N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Multicast topology inference from end-to-end measurements. In *In ITC Seminar on IP Traffic, Measurement and Modelling*, 2000.

[DHPT02] N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Multicast Topology Inference from Measured End-to-End Loss. *IEEE Transactions in Information Theory*, 48(1):26–45, 2002.

[DHT+02] N.G. Duffield, J. Horowitz, D. Towsley, W. Wei, and T. Friedman. Multicast-based loss inference with missing data. *IEEE Journal on Selected Areas of Communications*, 20(4):700–713, 2002.

[DL08] S. Dalibard and J-P. Laumond. Control of probabilistic diffusion in motion planning. *8th International Workshop on the Algorithmic Foundations of Robotics (WAFR 2008)*, december 2008.

[DLM+07] L. Denbya, J. M. Landwehr, C. L. Mallows, J. Meloche, J. Tuck, B. Xi, G. Michailidis, and V. N. Nair. Statistical Aspects of the Analysis of Data Networks. *Technometrics*, 49(3):318–334, August 2007.

[DP00] N.G. Duffield and F. Lo Presti. Multicast Inference of Packet Delay Variance at Interior Network Links. In *Proc. of IEEE INFOCOM*, pages 1351–1360, March 2000.

[DP04] N. G. Duffield and F. Lo Presti. Network tomography from measured end-to-end delay covariance. *IEEE/ACM Transaction on Networking*, 12(6):978–992, 2004.

[DPPT01] N.G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring Link Loss Using Striped Unicast Probes. In *IEEE INFOCOM*, pages 915–923, April 22–26 2001.

[DRM01] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *Proc. of IEEE INFOCOM*, pages 905–914, 2001.

[DRM04] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transaction on Networking*, 12(6), Dec 2004.

[Duf06] N. Duffield. Network tomography of binary network performance characteristics. *IEEE Transactions on Information Theory*, 52(12):5373 –5388, Dec. 2006.

[ENW96] A. Erramilli, O. Narayan, and W. Willinger. Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM Transactions on Networking*, 4:209–223, 1996.

[Erl09] A. K. Erlang. The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Mathematik B*, 20, 1909.

[Erl17] A. K. Erlang. Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *Elektrotkeknikeren*, 13, 1917.

[ESM09] A. Es-Saghouani and M. R. H. Mandjes. On The Correlation Structure Of A Levy-Driven Queue. *Journal of Applied Probability*, 45:940 – 952, 2009.

[FDL+01] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, D. Papagiannakia, and F. Tobagi. Design and deployment of a passive monitoring infrastructure. In Sergio Palazzo, editor, *Evolutionary Trends of the Internet*, volume 2170 of *Lecture Notes in Computer Science*, pages 556–575. Springer Berlin / Heidelberg, 2001.

[FGAMS06] G. Faÿ, B. González-Arévalo, T. Mikosch, and G. Samorodnitsky. Modeling teletraffic arrivals by a poisson cluster process. *Queueing Systems*, 54(2):121–140, 2006.

[FGM+97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2068 (Proposed Standard), January 1997. Obsoleted by RFC 2616.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785.

[FM05] C. Fragouli and A. Markopoulou. A network coding approach to overlay network monitoring. In *Proc. of Allerton*, 2005.

[GL80] G. H. Golub and C. Van Loan. An Analysis of the Total Least Squares Problem. Technical report, Ithaca, NY, USA, 1980.

[GM09] P. Glynn and M. R. H. Mandjes. Simulation-Based Computation Of The Workload Correlation Function In A Levy-Driven Queue. In *Proceedings of Winter Simulation Conference 2009*, 2009.

[GR00] L.S. Gradshteyn and L.M. Ryzhik. *Table of Integrals, Series and Products*. Academic Press, sixth edition, 2000.

[gre] grenouille. `http://grenouille.com/cest_quoi.php`.

[GS98] P. Glynn and K. Sigman. Independent sampling of a stochastic process. *Stochastic Processes and their Applications*, 78:151–164, 1998.

[HA03] J. D. Horton and López-Ortiz A. On the number of distributed measurement points for network tomography. In *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 204–209, 2003.

[HcJS03] F. Hernández-campos, K. Jeffay, and F. Donelson Smith. Tracking the evolution of web traffic: 1995-2003. In *Proc. of IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems (MASCOTS)*, pages 16–25, 2003.

[HFGC98] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. RTP Payload Format for MPEG1/MPEG2 Video. RFC 2250 (Proposed Standard), January 1998.

[HL04] G. Hartl and B. Li. Loss inference in wireless sensor networks based on data aggregation. In *Proc. of the IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN*, pages 396–404, 2004.

[HS02] N. Hu and P. Steenkiste. Estimating available bandwidth using packet pair probing. Technical report, 2002.

[HS03] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE journal on Selected Areas in Communications*, 21:879–894, 2003.

[HST07] M. Hasib, J. Schormans, and T. Timotijevic. Accuracy of packet loss monitoring over networked cpe. *IET Communications*, 1(3):507–513, june 2007.

[HVA03] N. Hohn, D. Veitch, and P. Abry. Cluster processes, a natural language for network traffic. *IEEE Transactions on Signal Processing, special issue "Signal Processing in Networking"*, 51(8):2229–2244, Aug. 2003.

[HVPD04] N. Hohn, D. Veitch, K. Papagiannaki, and C. Diot. Bridging router performance and queuing theory. In *Proc. of ACM SIGMETRICS*, pages 355–366, June 2004.

[HVY05] N. Hohn, D. Veitch, and T. Ye. Splitting and merging of packet traffic: Measurement and modelling. *Performance Evaluation, Proc. of IFIP International Symposium on Computer Performance, Modeling, Measurements, and Evaluation*, 62(1-4):164–177, Oct. 3-7 2005.

[HYH05] Han-Shen Huang, Bou-Ho Yang, and Chun-Nan Hsu. Triple jump acceleration for the EM algorithm. *IEEE International Conference on Data Mining (ICDM'05)*, 2005.

[iet] Internet engineering task force. `www.ietf.org`.

[Jac87] V. Jacobson. Traceroute. `ftp.ee.lbl.gov/traceroute.tar.gz`, 1987.

[Jac97] V. Jacobson. Pathchar: A tool to infer characteristics of internet paths., 1997.

[JD02] M. Jain and C. Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *Proc. of Passive and Active Measurements (PAM) Workshop*, 2002.

[JD05] H. Jiang and C. Dovrolis. Why is the internet traffic bursty in short time scales. In *Proc. of ACM SIGMETRICS*, pages 241–252, 2005.

[JJ93] Mortaza Jamshidian and Robert I. Jennrich. Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Association*, 1993.

[JYCA01] G. Jin, G. Yang, B. Crowley, and D. Agarwal. Network characterization service (NCS). In *Proc. of IEEE Symposium on High Performance Distributed Computing*, 2001.

[KBV09] Bruno Kauffmann, François Baccelli, and Darryl Veitch. Towards Multihop Available Bandwidth Estimation. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):83,84, September 2009.

[Kel79] F. Kelly. *Reversibility and Stochastic Networks*. Wiley, New York, 1979.

[Kel97] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 1997.

[Kes95] S. Keshav. A control-theoretic approach to flow control. *SIGCOMM Comput. Commun. Rev.*, 25(1):188–201, 1995.

[Kle75] L. Kleinrock. *Queueing Systems*, volume I: Theory, II: Computer Applications. John Wiley and Sons, 1975.

[Kle08] J. Klensin. Simple Mail Transfer Protocol. RFC 5321 (Draft Standard), October 2008.

[KMFB04] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A nonstationary poisson view of internet traffic. In *Proc. of IEEE INFOCOM*, 2004.

[KMT98] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks:shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[KSC+02] K.Papagiannaki, S.Moon, C.Fraleigh, P.Thiran, F.Tobagi, and C.Diot. Analysis of Measured Single-Hop Delay from an Operational Backbone Network. In *Proc. IEEE Infocom*, New York, June 2002.

[Lar90] R. Larson. The queue inference engine: deducing queue statistics from transactional data. *Management Science*, 36(5):586–60, 1990.

[LB01] K. Lai and M. Baker. Nettimer: a tool for measuring bottleneck link, bandwidth. In *Proc. of USENIX Symposium on Internet Technologies and Systems*, pages 11–11, 2001.

[LC06] P. Laskowski and J. Chuang. Network monitors and contracting systems: competition and innovation. In *Proc. of ACM SIGCOMM*, 2006.

[LMN06] E. Lawrence, G. Michailidis, and V. N.Nair. Network delay tomography using flexicast experiments. *J. Roy. Statist. Soc. (series B)*, 68:785–813, 2006.

[LMN07] E. Lawrence, G. Michailidis, and V. N.Nair. Statistical inverse problems in active network tomography. In *Complex Datasets and Inverse Problems: Tomography, Networks and Beyond, IMS Lecture Notes-Monograph Series*, volume 54, pages 24–44. IMS, 2007.

[LRL05] X. Liu, K. Ravindran, and D. Loguinov. Multi-Hop Probing Asymptotics in Available Bandwidth Estimation: Stochastic Analysis. In *Proc. of ACM/USENIX Internet Measurement Conference*, October 2005.

[LRLL04] X. Liu, K. Ravindran, B. Liu, and D. Loguinov. Single-Hop Probing Asymptotics in Available Bandwidth Estimation: Sample-Path Analysis. In *Proc. of ACM Internet Measurement Conference (IMC)*, October 2004.

[LTWW94] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transation on Networking*, 2(1):1–15, 1994.

[LY03] G. Liang and B. Yu. Maximum Pseudo Likelihood Estimation in Network Tomography. *IEEE Transaction on Signal Processing (Special Issue on Data Networks)*, 51(8):2043–2053, 2003.

[MACM05] M.Tariq, A.Dhamdhere, C.Dovrolis, and M.Ammar. Poisson versus Periodic Path Probing (or, Does PASTA Matter)? In *Proc. of ACM Internet Measurement Conference (IMC)*, pages 119–124, Oct. 2005.

[Mah99] B. A. Mah. Pchar. `http://www.kitchenlab.org/www/bmah/Software/pchar/`, 1999.

[MBG00] B. Melander, M. Björkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Proc. of Globecom*, pages 415–421, Nov 2000.

[MFPA09] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband internet traffic. In *Proc. of ACM SIGCOMM Internet measurement conference*, pages 90–102, 2009.

[MIP+06] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iplane: an information plane for distributed services. In *Proc. of the symposium on Operating systems design and implementation (OSDI)*, pages 367–380. USENIX Association, 2006.

[MK08] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics. John Wiley and Sons, second edition, 2008.

[MKLP05] Y. Mao, F. R. Kschischang, B. Li, and S. Pasupathy. A factor graph approach to link loss monitoring in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 23:820–829, 2005.

[Mor55] P. Morse. Stochastic properties of waiting lines. *Journal of the Operations Research Society of America*, 3(3):255–261, 1955.

[MPAM98] J. Mahdavi, V. Paxson, A. Adams, and M. Mathis. Creating a scalable architecture for internet measurement. In *Proc. of INET*, July 1998.

[MSWA03] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level internet path diagnosis. *SIGOPS Oper. Syst. Rev.*, 37(5):106–119, 2003.

[Muu83] M. Muus. Ping. `http://ftp.arl.mil/~mike/ping.html`, 1983.

[MVBB07] S. Machiraju, D. Veitch, F. Baccelli, and J. Bolot. Adding definition to active probing. *ACM Computer Communication Review*, 37(2):17–28, April 2007.

[MvdM09] M. R. H. Mandjes and R. van de Meent. Resource Dimensioning Through Buffer Sampling. *IEEE/ACM Transactions on Networking*, 17:1631 – 1644, 2009.

[MW00] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transaction on Networking*, pages 556–567, 2000.

[MZ09] M. Mandjes and P. Zuraniewski. A queueing-based approach to overload detection. In *Network Control and Optimization*, volume 5894 of *Lecture Notes in Computer Science*, pages 91–106. Springer Berlin / Heidelberg, 2009.

[NT04] H. C. Nguyen and P. Thiran. Active measurement for multiple link failures diagnosis in ip networks. In *Passive and Active Network Measurement*, volume 3015 of *Lecture Notes in Computer Science*, pages 185–194. 2004.

[NT07a] H. X. Nguyen and P. Thiran. Network loss inference with second order statistics of end-to-end flows. In *ACM SIGCOMM Internet measurement conference (IMC)*, pages 227–240. ACM, 2007.

[NT07b] H.X. Nguyen and P. Thiran. The boolean solution to the congested ip link location problem: Theory and practice. In *IEEE INFOCOM*, pages 2117 –2125, May 2007.

[NTV06] A. Novak, P. Taylor, and D. Veitch. The distribution of the number of arrivals in a subinterval of a busy period in a single server queue. *Queueing Systems*, 53(3):105–114, 2006.

[Ott77] T. Ott. The covariance function of the virtual waiting time process in an M/G/1 queue. *Adv. App. Prob.*, 9, 1977.

[Par09] B. Parker. *Design of Experiments for Packet Networks*. PhD thesis, Queen Mary, University of London, School of Mathematical Sciences, 2009.

[Pax94] V. Paxson. Empirically derived analytic models of wide-area tcp connections. *IEEE/ACM Transaction on Networking*, 2(4):316–336, 1994.

[Pax97] V. Paxson. End-to-end routing behavior in the internet. In *Proc. of ACM SIGCOMM*, 1997.

[Pax99] V. Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, 1999.

[PDHT02] F. Lo Presti, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal delay distributions. *IEEE/ACM Transactions on Networking*, 10(6):761–775, 2002.

[Pet83] Karl Petersen. *Ergodic Theory*. Cambridge University Press, Cambridge England, 1983.

[PF95] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, pages 226–244, 1995.

[PGS09] B.M. Parker, S.G. Gilmour, and J. Schormans. Measurement of packet loss probability by optimal design of packet probing experiments. *IET Communications*, 3(6):979 –991, june 2009.

[Pla] Planetlab. `http://www.planet-lab.org/about`.

[PQW02] V. N. Padmanabhan, L. Qiu, and H. J. Wang. Server-based inference of internet performance. In *IEEE INFOCOM*, 2002.

[PSHC+06] C. Park, H. Shen, F. Hernández-Campos, J. S. Marron, and D. Veitch. Capturing the elusive poissonity in web traffic (extended version). In *Proc. of IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Sep. 11-13 2006.

[PV02a] A. Pásztor and D. Veitch. Active Probing using Packet Quartets. In *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, pages 293–305, Nov 6-8 2002.

[PV02b] A. Pásztor and D. Veitch. On the scope of end-to-end probing methods. *IEEE Communications Letters*, 6(11):509–511, November 2002.

[PVK10] F. Pin, D. Veitch, and B. Kauffmann. Statistical estimation of delays in a multicast tree using accelerated em. *Queueing Systems*, 66(4):369–412, 2010.

[RCD+09] F. Ricciato, A. Coluccia, A. D'Alconzo, D. Veitch, P. Borgnat, and P. Abry. On the role of flows and sessions in internet traffic modeling: an explorative toy-model. In *Proc of. IEEE Globecom 2009*, Nov. 30 - Dec. 4 2009.

[RCR+00] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, and R. G. Baraniuk. Multifractal cross-traffic estimation. In *Proc. of ITC Specialist Seminar: IP Traffic Measurement, Modelling and Management*, Sep 2000.

[Res08] P. Resnick. Internet Message Format. RFC 5322 (Draft Standard), October 2008.

[RM99] S. Ratnasamy and S. McCanne. Inference of Multicast Routing Trees and Bottleneck Bandwidths Using End-to-end Measurements. In *IEEE INFOCOM'99*, pages 353–360, 1999.

[RNC04] M. Rabbat, R. Nowak, and M. Coates. Multiple source, multiple destination network tomography. In *Proc. of IEEE INFOCOM*, 2004.

[Rou05] M. Roughan. Fundamental bounds on the accuracy of network performance measurements. In *Proc. of ACM SIGMETRICS*, pages 253–264, 2005.

[Rou06] M. Roughan. A Comparison of Poisson and Uniform Sampling for Active Measurements. *IEEE J. Selected Areas in Communication*, 24(12):2299–2312, Dec 2006.

[RRB+03] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Workshop*, volume 4, 2003.

[Sav99] S. Savage. Sting: a TCP-based network measurement tool. In *Proc. of USENIX Symposium on Internet Technologies and Systems*, pages 71–79, 1999.

[SB93] D. Sanghi and S. Banerjee. Netdyn. `http://www.cs.umd.edu/~suman/netdyn/index.html`, 1993.

[SBDR05] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving accuracy in end-to-end packet loss measurement. In *Proc. of ACM SIGCOMM'05*, pages 157–168, 2005.

[SBDR07] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and efficient sla compliance monitoring. *SIGCOMM Comput. Commun. Rev.*, 37(4):109–120, 2007.

[SCFJ03] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. Updated by RFCs 5506, 5761.

[SH03] M.-F. Shih and A. O. Hero. Unicast-Based Inference of Network Link Delay Distributions With Finite Mixture Models. *IEEE Transaction on Signal Processing (Special Issue on Data Networks)*, 51(8):2219–2228, 2003.

[She95] M. J. Shervish. *Theory of Statistics*. Springer Series in Statistics. Springer Verlag, first edition, 1995.

[SKK03] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 39–44, 2003.

[SM98] Vinod Sharma and Ravi Mazumdar. Estimating traffic parameters in queueing systems with local information. *Performance evaluation*, 32:217–230, 1998.

[SR03] Ruslan Salakhutdinov and Sam Roweis. Adaptive overrelaxed bound optimization methods. *International Conference on Machine Learning (ICML-2003)*, 2003.

[Tak62] L. Takács. *Introduction to the Theory of Queues*. Oxford University Press, New York, 1962.

[TCN03] Y. Tsang, M. Coates, and R. Nowak. Network Delay Tomography. *IEEE Transaction on Signal Processing (Special Issue on Data Networks)*, 51(8):2125–2136, 2003.

[tra] Traceroute.org. `http://traceroute.org/`.

[W3C] World wide web consortium. `http://www.w3.org/`.

[WP98] W. Willinger and V. Paxson. Where mathematics meets the internet. *Notices of the American Mathematical Society*, 45:961–970, 1998.

[WTSW97] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. *IEEE/ACM Transaction on Networking*, 5(1):71–86, 1997.

[XMN06] B. Xi, G. Michailidis, and V. N.Nair. Estimating Network Loss Rates Using Active Tomography. *Journal the American Statistical Association*, 101:1430–1448, 2006.

[ZCB09] Y. Zhao, Y. Chen, and D. Bindel. Towards unbiased end-to-end network diagnosis. *IEEE/ACM Transactions on Networking*, 17(6):1724 –1737, Dec. 2009.