Large scale platform : Instantiable models and algorithmic design of communication schemes

Przemyslaw Uznanski

N° d'ordre : 4872

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Przemysław UZNAŃSKI**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Large Scale Platform: Instantiable Models and Algorithmic Design of Communication Schemes

---

Soutenue le : 11 octobre 2013

Après avis des rapporteurs :

Anne Benoit	Maître de Conférences HDR
Laurent Viennot	Directeur de Recherche INRIA

Devant la commission d'examen composée de :

Toufik Ahmed	Professeur	Président, Rapporteur
Anne Benoit	Maître de Conférences HDR ..	Examinateur
Fabien Mathieu	Chargé de Recherche HDR ...	Examinateur
Laurent Viennot	Directeur de Recherche INRIA	Examinateur
Olivier Beaumont	Directeur de Recherche INRIA	Directeur de Thèse
Lionel Eyraud-Dubois	Chargé de Recherche INRIA ..	Directeur de Thèse
Nicolas Bonichon	Maître de Conférences HDR ..	Directeur de Thèse (invité)

– 2013 –

Résumé

La popularité croissante des applications Internet très gourmandes en bande passante (P2P, streaming,...) nous pousse à considérer le problème suivant : *Comment construire des systèmes de communications collectives efficaces sur une plateforme à grande échelle ?*
Le développement de schéma de communications collectives dans le cadre d'un réseau distribué à grande échelle est une tâche difficile, qui a été largement étudiée et dont de multiples solutions ont été proposées. Toutefois, une nouvelle approche globale et systématique est nécessaire, une approche qui combine des modèles de réseaux et la conception algorithmique.

Dans ce mémoire nous proposons l'utilisation de modèles capables de capturer le comportement d'un réseau réel et suffisamment simples pour que leurs propriétés mathématiques puissent être étudiées et pour qu'il soit possible de créer des algorithmes optimaux.

Premièrement, nous considérons le problème d'évaluation de la bande passante disponible pour une connexion point-à-point donnée. Nous étudions la façon d'obtenir des jeux de données de bande passante, utilisant plateforme PlanetLab. Nous présentons aussi nos propres jeux de données, jeux obtenus avec bedibe, un logiciel que nous avons développé. Ces données sont nécessaires pour évaluer les performances des différents algorithmes de réseau. Bien qu'on trouve de nombreux jeux de données de latence, les jeux de données de bande passante sont très rares.

Nous présentons ensuite un modèle, appelé LastMile, qui estime la bande passante. En profitant des jeux de données décrits précédemment, nous montrons que cet algorithme est capable de prédire la bande passante entre deux noeuds donnés avec une précision comparable au meilleur algorithme connu de prédiction (DMF). De plus le modèle LastMile s'étend naturellement aux prédictions dans le scénario de congestion (plusieurs connexions partageant un même lien). Nous sommes effectivement en mesure de démontrer, à l'aide des ensembles de données PlanetLab, que la prédiction LastMile est préférable dans des tels scénarios.

Dans le troisième chapitre, nous proposons des nouveaux algorithmes pour résoudre le problème de diffusion. Nous supposons que le réseau est modélisé par le modèle LastMile. Nous montrons que, sous cette hypothèse, nous sommes en mesure de fournir des algorithmes avec des ratios d'approximation élevés. De plus nous étendons le modèle LastMile, de manière à y intégrer des artéfacts de connectivité, dans notre cas ce sont des firewalls qui empêchent certains nœuds de communiquer directement entre eux. Dans ce dernier cas, nous sommes également en mesure de fournir des algorithmes d'approximation avec des garanties de performances prouvables.

Les chapitres 1 à 3 forment les trois étapes accomplies de notre programme qui visent trois buts. Premièrement, développer à partir de zéro un modèle de réseau de communication. Deuxièmement, prouver expérimentalement sa performance. Troisièmement, montrer qu'il peut être utilisé pour

développer des algorithmes qui résolvent les problèmes de communications collectives.

Dans le 4e chapitre, nous montrons comment on peut concevoir des systèmes de communication efficaces, selon différents modèles de coûts, en utilisant des techniques combinatoires, tout en utilisant des hypothèses simplificatrices sur la structure du réseau et les requêtes. Ce travail est complémentaire au chapitre précédent puisque auparavant, nous avons adopté l'hypothèse que les connections étaient autonomes (i.e. nous n'avons aucun contrôle sur le routage des connexions simples). Dans le chapitre 4, nous montrons comment résoudre le problème du routage économe en énergie, étant donnée une topologie fixée.

Mots clés: PlanetLab, LastMile, diffusion, partage de bande passante, système de prédiction réseau, streaming, pare-feu, routage efficace en énergie, requêtes découpables

Discipline: Informatique

LaBRI,
Université Bordeaux 1,
351, cours de la libération
33405 Talence Cedex (FRANCE)

Abstract

The increasing popularity of Internet bandwidth-intensive applications prompts us to consider following problem: *How to compute efficient collective communication schemes on large-scale platform?*
The issue of designing a collective communication in the context of a large scale distributed network is a difficult and a multi-level problem. A lot of solutions have been extensively studied and proposed. But a new, comprehensive and systematic approach is required, that combines network models and algorithmic design of solutions.

In this work we advocate the use of models that are able to capture real-life network behavior, but also are simple enough that a mathematical analysis of their properties and the design of optimal algorithms is achievable.

First, we consider the problem of the measuring available bandwidth for a given point-to-point connection. We discuss how to obtain reliable datasets of bandwidth measurements using PlanetLab platform, and we provide our own datasets together with the distributed software used to obtain it. While those datasets are not a part of our model *per se*, they are necessary when evaluating the performance of various network algorithms. Such datasets are common for latency-related problems, but very rare when dealing with bandwidth-related ones.

Then, we advocate for a model that tries to accurately capture the capabilities of a network, named LastMile model. This model assumes that essentially the congestion happens at the edges connecting machines to the wide Internet. It has a natural consequence in a bandwidth prediction algorithm based on this model. Using datasets described earlier, we prove that this algorithm is able to predict with an accuracy comparable to best known network prediction algorithm (Distributed Matrix Factorization) available bandwidth between two given nodes. While we were unable to improve upon DMF algorithm in the field of point-to-point prediction, we show that our algorithm has a clear advantage coming from its simplicity, *i.e.* it naturally extends to the network predictions under congestion scenario (multiple connections sharing a bandwidth over a single link). We are actually able to show, using PlanetLab datasets, that LastMile prediction is better in such scenarios.

In the third chapter, we propose new algorithms for solving the large scale broadcast problem. We assume that the network is modeled by the LastMile model. We show that under this assumption, we are able to provide algorithms with provable, strong approximation ratios. Taking advantage of the simplicity and elasticity of the model, we can even extend it, so that it captures the idea of *connectivity artifacts*, in our case firewalls preventing some nodes to communicate directly between each other. In the extended case we are also able to provide approximation algorithms with provable performance.

The chapters 1 to 3 form three successful steps of our program to develop from scratch a mathematical network communication model, prove it experimentally, and show that it can be applied to develop algorithms solving hard problems related to design of communication schemes in networks.

In the chapter 4 we show how under different network cost models, using some simplifying assumptions on the structure of network and queries, one can design very efficient communication schemes using simple combinatorial techniques. This work is complementary to the previous

chapter in the sense that previously when designing communication schemes, we assumed atomicity of connections, *i.e.* that we have no control over routing of simple connections. In chapter 4 we show how to solve the problem of an efficient routing of network request, given that we know the topology of the network. It shows the importance of instantiating the parameters and the structure of the network in the context of designing efficient communication schemes.

Keywords: PlanetLab, LastMile, broadcast, bandwidth sharing, network prediction, streaming, firewalls, power aware routing, splittable requests

Discipline: Computer Science

Contents

Résumé étendu

Internet à haut débit est omniprésent dans notre vie quotidienne. Des connexions de plusieurs gigaoctet par seconde sont fréquentes, et des nouvelles applications gourmandes en bande passante sont apparues. Parmi ces services, on trouve les réseaux peer-to-peer, le streaming de vidéo, le stockage dans des clouds. Chacun de ces services, est utilisé par un grand nombre d'utilisateurs et manipule de grand volumes de données, laissent leurs créateurs avec un ensemble de défis à relever. Voici quelques exemples de questions à réoudre : comment modéliser le comportement du réseau ? Comment prédire le comportement du réseau ? Comment concevoir des réseaux de recouvrement efficaces ?

Même si le progrès technologique permet d'apporter des nouvelles solutions pour les utilisateurs, le facteur crucial dans leur développement réside dans le progrès algorithmique. C'était le développement d'architectures décentralisées, qui ont fait les solutions comme Gnutella [22] ou BitTorrent [54] un grand succès. La transition a eu lieu grâce à une meilleure utilisation des ressources, comme la bande passante, le temps CPU et la mémoire des machines connectées au réseau.

Ces solutions reposent sur la transmission de grandes quantités de données entre les nœuds du réseau. Le *Broadcast* est un scénario qui apparait souvent, où un nœud veut envoyer un (volumineux) message à tous les autres nœuds du réseau, ou plus généralement le *multicast*, où seul un sous-ensemble nœuds souhaite recevoir le bloc de données. Par exemple, dans un réseau peer-to-peer, où les données sont initialement présentes sur un seul nœud (la source) et puis distribuées à chaque nœud ces données. Dans un tel cadre de coopération, les nœuds qui ont obtenu des blocs de données peuvent commencer à les redistribuer eux-mêmes, accélérant ainsi le processus de diffusion.

Nous trouvons un autre exemple avec les sites de streaming vidéo. Les sites de streaming vidéo sont basés sur une architecture hautement centralisé, où le fournisseur doit investir dans une infrastructure énorme (comme c'est le cas de YouTube, en s'appuyant sur l'infrastructure de Google), et doit s'appuyer sur les solutions comme les serveurs géographiquement dispersés (afin de réduire la charge du centre de données). Il n'y a pas beaucoup d'espace pour la coopération entre les clients dans un tel service. Le nombre d'utilisateurs simultanés est massif, mais ils sont répartis sur un grand nombre de vidéos. Une situation tout à fait différente est présente pour les services de vidéo en streaming fonctionnant comme la télévision, (dédié aux événements en direct : sport, musique, jeux), puisqu'un très grand nombre de téléspectateurs peut coopérer de manière peer-to-peer. Cependant, même dans ce scénario, différentes solutions sont nécessaires. Ce n'est pas comme dans le cas de la diffusion classique. Dans le cas de video-streaming l'ordre de réception

des données possède un rôle crucial. Les données sont produites et consommées en direct. Des solutions théoriques ont été proposés : CoolStreaming [69], PPLive [65] ou SplitStream [17].

Comme nous l'avons vu, le développement des systèmes/méthodes efficaces est un problème complexe et repose sur plusieurs niveaux. Il est d'une grande importance de proposer des solutions en prenant en considération la bande passante disponible sur la plateforme sous-jacente. Beaucoup de travail a été fait pour développer ces solutions à un niveau algorithmique, qui consiste à présenter un modèle abstrait du réseau, puis à formaliser un problème et enfin à proposer une solution algorithmique pour la dernière. Internet lui-même est trop complexe pour permettre une approche "directe", d'où la nécessite d'introduire une couche d'abstraction.

Par conséquent, si nous représentons le réseau avec un modèle simple, par exemple, si nous faisons l'hypothèse que la structure sous-jacente du réseau n'intervient pas, nous permettons aux nœuds de communiquer librement et nous supposons qu'aucune interaction entre les différentes connexions ne prendra place, nous ne serons pas capable de capturer le comportement complexe et sophistiqué du réseau (par exemple l'apparition de contentions). Cependant, nous ne pouvons pas nous attendre à des résultats raisonnables des algorithmes reposant sur des modèles qui ne prennent pas en compte la contention.

D'un autre côté, en s'appuyant sur des modèles trop complexes, par exemple les réseaux représentés par un graphe pondéré, on risque de se retrouver incapable de développer des algorithmes d'optimisation raisonnables pour l'allocation des ressources sur ces réseaux (car déjà de nombreux problèmes de base sont trop difficiles à résoudre). Nous rencontrons également le problème d'être incapable d'instancier les paramètres du modèle, c'est-à-dire incapable de trouver une bonne représentation dans l'espace des paramètres d'un modèle donné pour un réseau donné. Les modèles trop précis sont également incapables de s'adapter aux différentes variations de réseaux que nous pourrions rencontrer, donc ils peuvent se révéler inutiles. En outre, les difficultés que nous rencontrons avec les opérations de base (telles que les mesures de point à point étant très variable au fil du temps) font que des modèles sophistiqués sont inutiles en pratique.

Le juste milieu que nous trouvons ici (nous devons sacrifier soit la force du modèle, soit la capacité de concevoir un algorithme efficace en utilisant le modèle) nous amène à un examen attentif des propriétés du réseau, que nous aimerions garder dans notre modèle et de développer des modèles en fonction de nos besoins. Dans le reste de cette thèse, nous aimerions (à partir des idées simples et en augmentant progressivement la complexité tout en montant l'échelle d'abstraction) construire méthodiquement des modèles et des algorithmes agissants à la base de ces modèles. Le sujet n'a pas été épuisé dans ce travail, mais nous avons tracé un cadre général qui pourrait être réutilisé dans les travaux futurs.

Motivation

La motivation principale de ce travail est le développementde schémas de communications collectives dans le cadre des réseaux distribués à grande échelle. Nous mettons la bande passante au centre de nos considérations. Les systèmes de diffusion provenant de la littérature ne sont pas satisfaisants dans notre contexte. Soit ils supposent une connaissance détaillée de la topologie détaillée du réseau (par exemple des solutions de Broadcast dans des réseaux maillés), soit les solutions

conçues sont compliquées, comme dans SplitStream [17].

Dans cette étude nous visons à trouver l'équilibre entre les deux approches décrites ci-dessus (à savoir des modèles simples rendant des solutions algorithmiques possibles, ou des modèles complexes et exact), et de proposer une approche nouvelle et globale. Nous allons commencer par l'approche très basique : la communication entre seulement deux nœuds sur Internet. Une telle communication simple peut déjà montrer un comportement complexe, que nous verrons en analysant le déroulement de l'envoi d'un message. Nous allons essayer de modéliser le réseau et de proposer un modèle qui représente fidèlement le réseau, même en cas de plusieurs communications simultanées.

Nous voulons abandonner l'idée d'une parfaite connaissance de la topologie du réseau . C'est pourquoi la prochaine étape est d'essayer de proposer des algorithmes afin d'instancier et d'évaluer les paramètres réels du modèle dans un vrai réseau. Nous sommes en mesure de le faire par l'usage de systèmes de prédictions (*Network Prediction Systems*), qui sont des systèmes conçus pour prévoir diverses mesures de performance du réseau. Ces systèmes attribuent généralement un nombre de paramètres pour les nœuds, et font une évaluation des paramètres du réseau (latence, bande passante) en fonction des paramètres des extrémités de la liaison. Les systèmes que nous trouvons dans la littérature dépendent soit sur le plongement du réseau dans un espace métrique, soit utilisent des approches plus synthétiques, comme la factorisation des matrices [42] ou le plongement sur un arbre (tree embedding) [57]. Nous plaidons pour l'utilisation de LastMile, où chaque nœud possède des paramètres associés ayant une interprétation dans la réalité (capacités des liaisons reliant les nœuds périphériques au centre d'Internet). Cette interprétation permet une simple instanciation des paramètres et rend naturel le développement des algorithmes base sur notre modèle.

Notre dernière étape sera de développer des algorithmes avec des bonnes garanties de performance (facteur d'approximation). Le but de ces algorithmes est de créer des systèmes de communications collectives. Notre objectif est de montrer qu'il n'est pas nécessaire de connaître la topologie complète du réseau, et que la possibilité d'instancier quelques paramètres suffit pour concevoir des systèmes de communication efficaces. Nous construisons ces algorithmes en supposant que rien de plus que la connaissance des liens extrêmes qui caractérisent les capacités du nœud à transférer des données dans le réseau. Grâce à notre approche, nous sommes en mesure de réduire le problème de la construction des réseaux logique à des problèmes combinatoires très simples et de le résoudre en conséquence. Le cadre que nous construisons est assez large, et à titre d'exemple, nous montrons comment intégrer la notion de *connectivy artefact*, sous la forme de firewall (nous classons les nœuds comme "open" ou "guared", et nous supposons que deux nœuds guared ne peuvent pas se communiquer directement).

Une fois le réseau logique construit (ensemble de connexions entre les nœuds) et les débits qui devraient être alloués à toutes les connexion sont déterminés, la solution peut être décomposée en un ensemble d'arbres de diffusion pondérés. Cette décomposition précise quelles données doivent être envoyées sur quelle connexion. Afin d'éviter cette étape de décomposition, qui est difficile à utiliser en pratique, nous nous appuyons sur un algorithme randomisé de Broadcast, proposé par Massoulié dans [47]. Cet algorithme est entièrement décentralisé et est capable de résoudre des cas avec des légères variations des ressources, grâce à son caractère aléatoire et dynamique. Cet

algorithme nécessite la connaissance de la topologie du réseau et des bandes passantes sur les arêtes et pas de contention sur les nœuds, ce qui est complètement irréaliste dans une plateforme à grande échelle. Cependant, le réseau logique que nous construisons possède exactement ces propriétés, par construction. Le réseau logique que nous construisons peut donc être utilisé comme une entrée directe dans l'algorithme de Massoulié.

L'originalité de cette thèse est d'étudier le problème d'un point de vue global, en partant des mesures pour arriver aux modèlex et aux algorithmes basés sur ces modèles. Toutes les étapes décrites ci-dessus peuvent être combinées en une plus grande image qui permet à notre solution d'avoir des applications pratiques :

- La méthodologie d'évaluation de la bande passante point-à-point,
- Le modèle de réseau, validé par ensemble de données expérimentales,
- L'instanciation des paramètres du modèle à partir des mesures,
- L'algorithme pour une construction d'un réseau logique efficace en termes d'énergie, ayant les paramètres du modèle de Broadcast en tant qu'entrée,
- L'algorithme de Massoulié pour une détermination randomisée du paquets routage réelle.

Comme un travail complémentaire, seulement partiellement lié au reste de la thèse, nous fournissons également une étude où nous analysons la façon de concevoir un système de routage efficace en termes d'énergie dans une classe spéciale de graphes. Ce travail est motivé par la nécessité de concevoir un nouveau type de solutions lorsqu'il s'agit de routage et des problèmes de flot dans les réseaux. Les solutions classiques analysent généralement des situations où les arêtes d'un réseau sont limitées de manière brutale. Nous avons des solutions de conception où à chaque lien est associée une fonction de coût convexe, et la forme de la fonction impose que les solutions à faible coût soient bien équilibrés et aucun arête ne recevra de bande passante élevée. Notre motivation provient du besoin d'assurer un routage des messages efficace dans un *system-on-chips*, où la fonction de coût représente la consommation d'énergie par rapport à un transfert de données sur l'arête.

Introduction

Recent years have made a high speed Internet ubiquitous. Connections of speed gigabytes per second are common, and new bandwidth-intensive applications have arisen. Examples of such services are peer-to-peer networks, video-streaming, cloud storage. Such services, massive in amount of users and data usage, put their creators with a set of challenges to overcome. Examples of such challenges are: how to model network behavior? How to predict network behavior? How to design efficient overlay networks?

Even though the technological progress makes possible to bring new solutions to users, the crucial factor in developing them lies in an algorithmic progress. It was the development of decentralized architectures that made the solutions like Gnutella [22] or BitTorrent [54] highly successful and made file-sharing a massive phenomenon. The transition occurred in better usage of given resources, *i.e.* bandwidth, CPU time and disk space of collective of machines connected to the network.

Such solutions rely on transmitting large chunks of data between nodes of the network. *Broadcast* is a common scenario, where one node wants to send a chunk of data to all other nodes in the network, or more general *multicast*, where there exists helper nodes but they do not need to receive the whole message. We can imagine, for example, a peer-to-peer network, where the data that is initially seeded by a single node (source) is finally distributed across every node wanting to obtain it. In such a cooperative setting, nodes that got some chunks of data can start redistributing them on their own, thus speeding the whole process.

Another example is provided by video-streaming sites. Popular video sites are based on highly centralized architecture, where the provider has to invest in a huge infrastructure (as it is the case of Youtube, relying on Google's infrastructure), and rely on the solutions like geographically distributed caching servers (to reduce the load of the data-center). There is not much space for cooperation between the clients in such a service: while the number of simultaneous users is massive, they are spread over huge number of different videos. Different situation is present for video-streaming services functioning tv-like, (dedicated for live events: sport, music, gaming), as even the massive number of viewers can cooperate in a peer-to-peer fashion. However even in this scenario different solutions are necessary, as in classical broadcast we do not care about the order of incoming data, as long as we finally get all of it, and in video-streaming, the data is produced and consumed live, and has short life-span. Theoretical solutions have been proposed: CoolStreaming [69], PPLive [65] or SplitStream [17].

As we saw, designing efficient schemes is a complex and a multi-level problem. It is of high importance to make such solutions bandwidth-aware, that is taking into consideration available

bandwidth of the underlying platform when constructing solution. Plenty of work has been done to develop such solutions on an algorithmic level, that is to present an abstract model of network, formalize a problem, and propose an algorithmic solution to it. Abstracting through a model is necessary, as the Internet itself is too complex to allow any "direct" approach.

Therefore, if we represent the network with a simple model, *e.g.* if we do not assume anything about the underlying structure, and allow nodes to communicate freely, and assume no interactions between various connections, it fails to capture complex and sophisticated behavior of the network (*e.g.* contention). However, we cannot expect any reasonable results from the algorithms relying on models that do not take contention into account.

On the other side, if relying on too complicated models, for example networks represented as a weighted graph, we are left unable to develop any reasonable optimization algorithms for resource allocation on those networks (as many basic problems became too hard to solve). We also encounter a problem of being unable to *instantiate the parameters* of the model, by which we mean being able to find a good representation in a parameter space of a given model for the given actual network. Too precise models are also unable to adapt to different variations of networks we could encounter, which may prove them useless. Also, the difficulties that we encounter with basic operations (such as point-to-point measurements being highly variable and changing over time) make sophisticated models useless.

The trade-off we are dealing here (we have to sacrifice either strength of the model, or ability to design efficient algorithm using the model) leads us to a careful examination of what network properties we would like to have in our model and develop models according to our needs. In the rest of this thesis we would like to (starting with simplest ideas and gradually increasing complexity while we move upward in abstraction layers) methodically construct models and algorithms working on them. The topic is not exhausted in this work, but a general framework that could be emulated in future work is provided.

Motivation for a study

The main motivation for this work is the design of collective communications in the context of large sale distributed networks. We put emphasis on being bandwidth-aware in our design. State-of-the-art content distribution systems are unsatisfactory. They either assume knowledge of the detailed topology of the network, *e.g.* broadcast solutions for meshes, or the designed solutions are complicated, as in SplitStream [17].

We aim in this study at finding the balance between the two approaches described above (namely simple models making algorithmic solutions possible, and complex and exact models), and to propose a new and comprehensive approach. We will start with the very basic, atomic approach of understanding the communication between just two nodes on the Internet. Such a simple communication can already show complex behavior, which we will see by analyzing bandwidth-time plots of communications. We will try to model the network and to propose a model that accurately captures the bandwidth-related capabilities of the network, *e.g.* contention, bottleneck links and bandwidth sharing.

We want to abandon the idea of perfect knowledge of the network topology. That is why our

next step is to try to propose algorithms in order to instantiate and evaluate the actual parameters of the model in a real network. We are able to do it by usage of *Network Prediction Systems*, which are systems built to predict various performance metrics of the network. Such systems usually assign a few parameters to the nodes, and evaluate network parameters (latency, bandwidth) as a function of parameters of the endpoints of the link. State-of-the-art systems either depend on embedding the network into a metric space, or use more synthetic approaches like matrix factorization [42] or tree embedding [57]. We advocate for the usage of LastMile, where each node has associated parameters having real-life interpretation (capacities of links connecting peripheral nodes to the core Internet). Such interpretation allows for simple instantiation of the parameters, as well as for natural usage in the design of algorithms operating on the model.

Our last step will be to design algorithms building collective communication schemes with good approximation guarantees. Our aim is to show that it is not necessary to know the full topology of the network, and that the ability to instantiate few parameters provides enough knowledge to design efficient communication schemes. We build those algorithms assuming nothing more than the knowledge of the end-links characterizing the capabilities of the node to transfer data in the network. In our approach we are able to reduce the problem of building overlay networks into a very simple combinatorial problem and solve it accordingly. The framework we build there is broad enough, and as an example we show how to incorporate the notion of connectivity artifact, in the form of firewalls (we classify the nodes into ”open” and ”behind firewall”, and we assume that two nodes behind firewalls cannot communicate directly).

Once the network is built (which nodes communicate together) and the bandwidths that should be allocated to each edge create an overlay are determined, the solution can be decomposed into a set of weighted broadcast trees. This decomposition specifies which data should be sent on which edge at a given time step. In order to avoid this decomposition step, which is difficult to use in practice, we rely on the randomized broadcasting algorithm proposed by Massoulié in [47]. This algorithm is fully decentralized and is even able to deal with small variations of resource performance due to its randomized and dynamic nature. This algorit hm requires the knowledge of the topology of the network with bandwidths on edges and no contentions on the nodes, that is in general not realistic. On the other hand, the overlay network that we build has exactly, by construction, these properties. The overlay network that we build in this chapter can therefore be used as direct input to Massoulié's algorithm.

The originality of this thesis is to consider the problem on the complete chain, from measurements to model and algorithms. All of the steps described above can be combined into a ”bigger picture” which allows our solution to be applicable in practice:

- methodology of point-to-point bandwidth measurements,
- network model, validated through experimental dataset,
- instantiation of model parameters from measurements,
- algorithm for construction of efficient broadcast overlay network having model parameters as an input,
- Massoulié's algorithm for randomized determining actual packet routing.

As a complementary work, only partially related to rest of the thesis, we also provide a study where we analyze how to design an efficient, power-aware routing scheme in a special class of graphs. This work is motivated by a necessity to design a new kind of solutions when dealing with routing and flow-related network problems. Classical solutions usually analyze situations where links on a network are capped in a hard way. We design solutions where with every link there is associated a convex cost function, and the shape of the function enforces that low-cost solutions will be well-balanced ones and no edge will receive high bandwidth. Our motivation comes from need of efficient routing of messages in system-on-chips, where cost function represents the power usage in relation to data transfer over edge.

Structure of the thesis

This thesis is structured as follow:

In Chapter 1, we discuss the measurements of network parameters, *e.g.* latency and available bandwidth. We discuss techniques for measuring available-bandwidth. We also describe *PlanetLab* platform, a large-scale, worldwide distributed platform for performing network-related experiments. We discuss how to obtain reliable datasets of bandwidth measurements using a PlanetLab platform. The S^3 project from HP aimed at monitoring the PlanetLab platform, provides us an example of such datasets. We also provide our own datasets, and we discuss them together with the distributed software used to obtain them. While those datasets are not a part of our model *per se*, such datasets prove necessary when evaluating performance of various network algorithms. We also discuss differences between bandwidth-related and latency-related datasets, and the problems we encountered when trying to obtain the former ones. In this chapter we also describe our experiments concerning bandwidth sharing measurements. We provide a detailed discussion of the setting, methodology, software and platform concerning those datasets. We believe such datasets are important to validate models when it comes to predicting congestion.

In Chapter 2, we introduce the idea of *Network Prediction Systems*. These systems are an algorithmic approach to the problem of predicting the values of various network metrics (for instance latency or available bandwidth between nodes). They are typically used in a situation when only partial measurements are available, to be able to infer value for every point-to-point communication. Each algorithm is discussed together with its pro and cons, and for each system we also discuss the network model that it is based upon.

We start by discussing earlier approaches, based on embeddings into metric space. Such embedding rely on assigning to each node in network an appropriate point in the parameter space, and network metric we are estimating is read as the distance between points in the metric space. We present various systems based on such an approach, like GNP (embedding into Euclidean high dimension space) or Vivaldi (embedding into Euclidean space + height). Another approach is basis of the Sequoia system, which tries to embed network into the virtual tree, trying to mimic the hierarchical structure of the network.

We also describe approaches based on a matrix factorization. This is a synthetic approach based on a treating the full set of measurements as an unknown, partially filled matrix of values, that we try to represent as a low rank matrix (a matrix product $P \times Q$ where P and

Q are respectively $n \times d$ and Q is $d \times n$ matrices, for some $d \ll n$). Our experimental comparison on datasets we obtained shows that this approach yields best estimation from known prediction systems. This is despite the fact that there is no underlying network model supporting this form of a calculation and effectiveness of the algorithm is only due to the matrix approximation as a low rank being very efficient.

Then, we advocate for a network model that tries to accurately capture the capabilities of the network, named LastMile model. This model assumes that essentially the congestions happen at the edges connecting machine to the wide Internet. It has a natural consequence in bandwidth prediction algorithm based on this. Using datasets described earlier, we prove that this algorithm is able to predict with an accuracy comparable to best known network prediction algorithm (Distributed Matrix Factorization) available bandwidth between two given nodes. While we were unable to improve upon DMF algorithm on the field of point-to-point prediction, we show that our algorithm naturally extends to the network predictions under the congestion scenario. We are actually able to show that LastMile predicts better in such scenarios, using PlanetLab datasets.

We also show the use of another strength of LastMile approach, *i.e.* the easiness of extending the model. We show how the model can be generalized to incorporate the notion of well-connected core Internet. The division into small, well-connected core and large, sparse, tree-like peripheral Internet is supported by various empirical evidences (for example measures of treewidth and hyperbolicity of the Internet). We show that LastMile is elastic enough to incorporate such notions into its model, and that this extension leads to similarly simple algorithms.

We end the chapter with description of a `bedibe` tool, that we developed as a response to a growing need to prototype and evaluate different algorithms over various datasets. We used this tool to obtain datasets, as well as analyzing performance of the algorithms with respect to this datasets.

Chapter 3 describes the next step of our work, of developing specific optimization algorithms for networking problems. We consider the classical problem of broadcasting a large message at an optimal rate in a large scale distributed network under the LastMile communication model. In this context, we are interested in both building an overlay network and providing an explicit algorithm for scheduling the communications. From an optimization point of view, we aim both at maximizing the throughput (*i.e.* the rate at which nodes receive the message) and minimizing the degree of the participating nodes (*i.e.* the number of TCP connections they must handle simultaneously). We propose new algorithms solving this massive scale broadcast problems. We show that under that type of assumption, we are able to provide algorithms with provable, strong approximation ratios.

Taking advantage of the simplicity and elasticity of the model, we can even extend it, so that it captures the idea of *connectivity artifacts*, in our case firewalls preventing some nodes to communicate directly between each other. In the extended case we are also able to provide approximation algorithms with provable performance.

In Chapter 4, we show how under different network cost models, using some simplifying assumptions on the structure of the network and of the queries, we can design very efficient communication schemes using simple combinatorial techniques. We investigate designing routing of the communication algorithms, using some simplifying assumptions on the environment (the graph where the communication takes place is a grid graph) and cost model (we investigate the power-aware model). This work is complementary to the previous chapter in the sense that previously when designing communication schemes, we assumed atomicity of connections, *i.e.* that we have no control over routing of simple connections. In this chapter we show that the knowledge of topology of the network or well-specified cost model allows us to solve efficiently the problem of routing the network requests. It shows the importance of being able to efficiently instantiate the parameters and the structure of the model, as such detailed instantiation gives us very strong tools to design efficient communication schemes.

Chapter 1

Measuring the Network

In large scale Internet platforms (*e.g.* peer-to-peer, streaming networks, clouds), measuring the available bandwidth between nodes of the platform is difficult and costly. In many Internet applications, network-awareness is an important part of achieving good performance or lowering resource usage. In the case of delivering video on demand [63], or performing peer-assisted streaming [44] for example, estimations of available bandwidth allow the construction of an efficient overlay topology. Efficient algorithms for broadcasting or for organizing master/slave communications have been proposed [8, 6]. Examples of such overlays will be the topic of Chapter 3.

Those algorithms rely on an accurate representation of the performance of the network. In all these applications, large amounts of data need to be exchanged between nodes, and hence the available bandwidth is the important metric for application performance. However, the naive approach, that is measuring available bandwidth between nodes which are geographically distributed over the Internet usually incurs a high cost, both in terms of measuring time and induced network load. It is thus not desirable to perform periodic explicit end-to-end path measurements. Another downside of this direct approach is that the number of possible scenarios we have to consider grows quickly with the size of the network. Taking into account isolated point-to-point communication, the number grows quadratically with the number of nodes (which already is impractical for large networks), that is a full matrix of values indexed by starting and ending points of communications. If we consider congestion scenarios (where one node can communicate with a set of nodes), the number can be even exponential. This forces us to provide other solutions than measuring in advance network performance in every possible communication scenario, and directly measuring every value of the communication matrix can be too much. Furthermore, we should consider that network environment is a highly dynamic one, which implies that such explicit path measurements provide a representation of the platform that is never up-to-date. We should be able to reevaluate our measures, and take into consideration the high volatility of the measures.

As we will see in section 1.5.1, having the measures over the same edge varying twofold or even larger is not that uncommon. We will discuss the methodology of bandwidth measurement in Section 1.2.

To remedy problems highlighted above, we should consider the following solution. Firstly, for each node, just keep track of partial data only, trying to capture the behavior of the node in the network with only few essential measurements. We should be able to perform those measures for

example when new nodes are introduced to the network (in a peer-to-peer networks), or periodically, to keep track of changes of network topology. Second, we should be able to extrapolate this small data to predict the behavior of connections in more complicated situations. We will discuss problem of predicting the network in Chapter 2. Having access to those predictions about the performance of the platform, and especially about the available bandwidth, allows us to design efficient algorithms that optimize resource usage for different types of collective communications. In fact, whenever designing broadcasting algorithms or organizing master/slave computations, it seems essential to obtain this type of information from the underlying network.

We will discuss point-to-point predictions with introduction of Network Coordinate Systems, and more generally, Network Prediction Systems in Chapter 2. We will cover them deeper further in the thesis, but the way both topics interact is worth noting. On the one hand, it is essential to have reliable datasets when designing, prototyping and testing various Network Prediction Systems. On the other hand, using reliable prediction algorithm, we can measure network only partially, for some landmark edges, and recover the rest of the data using predictions.

Available bandwidth datasets are quite rare in the literature. The S^3 project [67], which measured available bandwidth with the Spruce tool [62] between PlanetLab nodes, is now discontinued, and we are not aware of other similar projects which are still active. Several tools exist for measuring available bandwidth (*i.e.* the minimum remaining capacity on all links on the path) [55], while keeping the load incurred to the network low. Furthermore, available bandwidth is not identical to the available TCP throughput, which is the metric that actually influences the performance of applications. State-of-the art methods for measuring available TCP throughput (such as Iperf [25]) are much more intrusive, and generally rely on mimicking the behavior of TCP, by sending as much data as possible, and measuring how much data is actually received in a given amount of time, once steady-state has been reached.

In this chapter, we describe our effort to obtain complete and reliable datasets of two different types. First, we present datasets measuring point-to-point bandwidth over larger collection of nodes, then we follow with datasets measuring bandwidth sharing over smaller sets of nodes. Both datasets were collected on PlanetLab platform. We discuss the methodology we used for performing both experiments, what we aimed for and what we obtained. We also provide broad discussion of obstacles we encountered while obtaining datasets, originating from either networking environment characteristic, or being specific to PlanetLab. Finally, in this chapter we also provide statistical analysis of the datasets, followed with reasoning about possible models for the behavior of underlying network.

1.1 Context

In this section, we discuss our measurements in a wider context. We discuss platforms and software used for obtaining datasets. In contrary to latency datasets, bandwidth datasets are rare. Nevertheless we discuss already existing bandwidth dataset, *i.e.* S^3 [68]. Please note that we use a collective term *bandwidth* for both TCP throughput and available bandwidth.

1.1.1 PlanetLab

The measures have been performed on the PlanetLab platform[1] [19]. PlanetLab is a large-scale, worldwide distributed platform which provides an access to nodes on more than 500 sites across the world, with more than 1000 nodes. PlanetLab was created in order to support the design and evaluation of various Internet related applications, such as peer-to-peer file sharing or content distribution networks. PlanetLab is designed with a heavy emphasis on virtualization. The center of PlanetLab architecture is a *slice*, a horizontal cut of global PlanetLab resources. Each service runs in a slice of PlanetLab. Each slice is allocated a certain amount of resources (processing, memory, network resources) across a set of individual nodes, and can be viewed as a network of virtual machines, with resource bound to each virtual machine. Nodes of PlanetLab are geographically distributed, connected by a diverse collection of links, including edge sites, co-location and routing centers, and low-profile machines. It has also become a de facto standard for conducting large scale Internet experiments, and is thus well suited for our purpose. Furthermore, its accessibility makes it relatively easy to conduct the required measurements.

However, there are still several downsides associated to the choice of PlanetLab for performing measurements. The nodes are not the best representation of the Internet, with majority of the nodes connected to the Global Research and Education Network (GREN). While there were various efforts to improve the diversity, most of the nodes are still connected through the GREN (with only 26 nodes connected through the commercial Internet). The heavy load that the nodes of PlanetLab experience is another source of problems.

1.1.2 SPLAY

The measurements were performed using the SPLAY middleware[2] [40]. SPLAY was created with aim of simplifying the prototyping and development of large scale distributed applications and overlay networks. It provides a set of libraries and interfaces for writing applications. SPLAY also provides web and command-line interfaces for job management on platform, node selection tools, or for control of the behavior of testbed itself. We used SPLAY service on PlanetLab running on its own PlanetLab slice. Using SPLAY helped us to write really lightweight and error-less networking code, and concentrate on the logic of experiments and measures themselves. It is also a great tool for automatic deployment of code on tens or hundreds of PlanetLab nodes.

1.1.3 S^3

The S^3 project from HP [68] was aimed at monitoring the large scale distributed platform PlanetLab, and in particular at providing available bandwidth measurements between pairs of nodes of this platform, obtained with the Spruce tool [62]. These datasets have been used by the community to validate bandwidth estimation tools [35]. However, this project is now discontinued, and we are not aware of other similar projects which are still active. We decided not to use Spruce by ourselves, because it requires some extra knowledge on the measured link, that is capacities of

[1] `http://www.planet-lab.org`
[2] `http://www.splay-project.org`

individual links on the edge. We use S^3 datasets as a reference point for checking validity of our point-to-point measures.

1.2 Measurement methodology

In this chapter we focus on application-level measurements, in order to observe the platform as it would be accessible to the application. Hence, we measure available TCP throughput, which is the steady-state reachable throughput that can be achieved with a TCP connection.

1.2.1 Available bandwidth

It is a nontrivial problem to define the ”available bandwidth” for a fixed Internet path P. It is common to define it as the maximum rate of transmission that could be achieved without reducing rate of the rest of the traffic in P [30]. However, in our experimental setting, we have no control on other traffic that goes over the same path. But we can safely assume that the network we are operating on (PlanetLab) is coupled with the Congestion Control mechanisms appropriate for systems under heavy usage [53]. So we assume, that the bandwidth we achieve when we ”flood” the path with our transfer, is in fact the maximum achievable one that does not prevent other communications from taking place.

1.2.2 Intrusive vs. non-intrusive measures

Efficiently measuring the available bandwidth on a path has been addressed in several works (see for example [30]). A simple and natural method consists in sending as much data as possible, and measuring how much data we are able to send in a given time. The major downside of this method is that it incurs a very heavy load on the network for the whole duration of the experiment, that needs to be long enough to ensure that steady-state has been reached. By necessity, several less intrusive methods for measuring available bandwidth have been developed, like pathload, IGI or pathchirp [55]. Pathload tool relies on a Self-Loading Periodic Streams (SLoPS) methodology. The methodology involves sending periodic packet streams of fixed size, and then monitoring variations in the one way delays of the probing packets. If the stream rate is larger than the path's available bandwidth, the stream will cause a short term overload in the queue of the tight link. On the other hand, if the stream rate is lower than the available bandwidth, the probing packets will go through the path without causing an increase in the delays. The sender can thus try and estimate the highest possible sending rate that will not increase the packet delays. IGI and pathchirp rely on Trains of Packet Pairs (TOPP) methodology, which is a modification of SLoPS, based on statistical analysis of behavior of pairs of packets. TOPP sends many packets pairs at gradually increasing rates from the source to the sink. If the offered rate of the packets (L/Δ_S, where L is size of packet, Δ_S is the dispersion rate, that is the time between first and second packet) exceeds end-to-end available bandwidth, the second probing packet will be queued behind the first probing one, and measured rate will smaller than offered rate. On the other hand, if offered rate does not exceed available bandwidth, the measured rate will be the same.

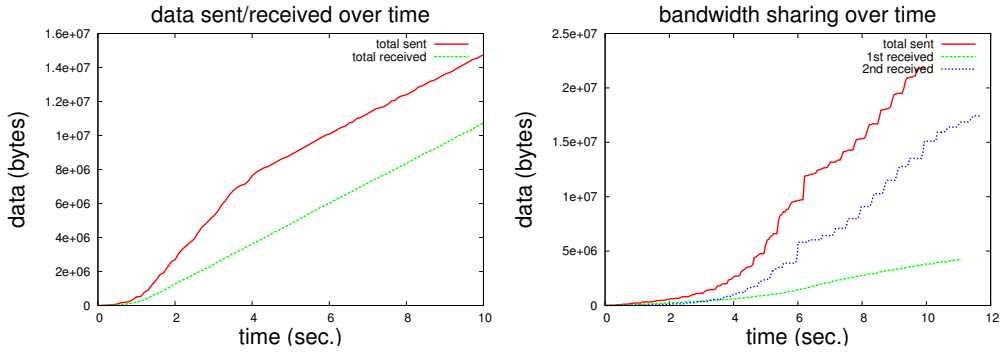

Figure 1: Plot of total cumulative data sent or received, with one node sending simultaneously: to one node (on the left) or to two nodes (on the right).

To summarize, those tools rely on sending a few packets along the path, and analyze the effects of intermediate nodes and cross traffic on these probe packets. Although they do not require privilege access, these tools require a fine grain access to the network. However, using them on PlanetLab is not easy and is not reliable [39], at least not more reliable than brute force approach, since the main source of failures of measurement between nodes is the inability to establish connection, without distinction for non-intrusive and intrusive measures. Even though these methods are much more efficient in terms of resource usage, we decided to rely on simple, yet intrusive, measures. This decision is motivated by two points. Firstly, since the goal of our study is to analyze several methods for predicting bandwidth from a limited number of measurements, it is important to introduce as few measurement errors as possible in the data. Furthermore, we could not find bandwidth datasets obtained with a direct method, since the S^3 dataset was obtained with Spruce. Secondly, and most importantly, our study of congestion with several connections requires direct intrusive measures.

1.2.3 Protocol limitations

The biggest obstacle to overcome is the slow start of a TCP connection [29]. TCP protocol contains Congestion Control mechanisms. As a result, during the initial phase of the connection the transmission rate increases exponentially with time, by doubling the congestion window with each received acknowledgment. So, when measuring bandwidth, we must make sure not to take into account the initialization period of connection, as it will make measures underestimated. On the left side of the Figure 1 we provide a plot of both total number of bytes send and received as a function of time, during a single transmission session between two nodes, one sending and another one receiving. It provides an example of behavior described above, where we can clearly see that all sending nodes achieve full available transmission speed a few seconds after the connection we established, as constant slope means constant throughput. After a while, the sending speed gets adjusted, also as a result of a Slow Start mechanism. The delay between sending and receiving the data can be attributed to the TCP buffers. Situation becomes more unpredictable, when there are multiple simultaneous connections. We observed that the bandwidth share allocated to a unique

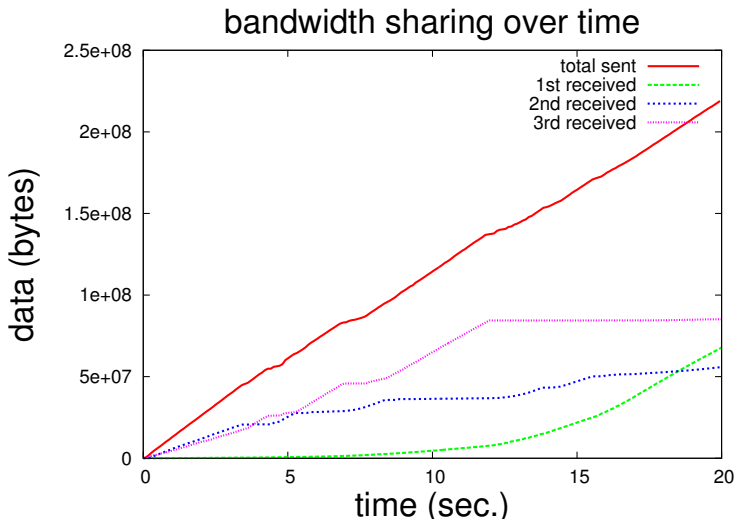

Figure 2: Plot of total cumulative data sent or received when one sending node transmits simultaneously to three receiving nodes.

connection, during a single transmission session is fluctuating heavily over time. Since the starting behavior is chaotic, we arrange the measures of the connection speed for sufficiently long enough to make averages overcome local fluctuations. However, even with the long snapshots, the behavior varies strongly, even when trying to reproduce the same scenario. Plot on the right of 1 provides an example of sharing a connection between two transmissions. The steps we see on the plot are another effect of TCP Congestion Control.

Figure 2 shows an example of such a behavior, where multiple outgoing connections compete over share in more or less constant stream of outgoing data in one sending node to three receiving ones. We observe that while the sender quickly saturates its outgoing link and transmission is performed at constant speed, the share that is assigned to each receiver varies in time. We see that one connection (only after 12s since start) saturates and effectively kills both other competing connections (their speed drops to zero).

Another obstacle to overcome comes from the fact that TCP is not stateless. When performing series of measures, closing and reopening the connection can be not enough to reset the connection. To avoid encountering traces of old connections in the network when repeating the same connection, we made sure to wait a sufficient amount of time between measurements on the same edge (5min wait was implemented).

1.2.4 PlanetLab limitations

Our choice to perform the experiments on PlanetLab influenced heavily how they were performed. The PlanetLab platform itself, because of the fact that it is shared among a large number of users, comes with a number of restrictions.

In order to avoid flooding, PlanetLab has a policy fixing for each node a daily data transfer

limit. Together with our measurement methodology which requires to send data for 20 seconds to perform one measurement, this means that it takes several days to gather an exhaustive and broad enough dataset, even for a small number of nodes. Hence, the number of nodes must be kept at reasonable size if we want our datasets to span relatively short periods of time.

PlanetLab nodes are under heavy usage, both in terms of CPU and bandwidth. Because of heavy CPU usage, and special process scheduler, time measures are unreliable (under 10ms error), which makes latency measures over network unpractical, and makes it very difficult to use state-of-the art available bandwidth estimation tools, that rely on precise timings of packet arrival times (see [55, 30]). Our bandwidth measures however do not suffer from this problem, since we measure time at a much coarser scale. Heavy bandwidth usage from other PlanetLab users yields a very high variability of the measures, even when performing one measure just after another. Example of how variable individual measurements over single edge can be seen on the Plot 5.

The PlanetLab platform is not a "typical" Internet platform, in the sense that it consists of servers hosted by universities or research institutions, often connected through high-speed and high-bandwidth academic networks. They are also usually close to the main Internet routes. It is important to keep in mind that this platform is thus not representative of a typical peer-to-peer situation. However, its size and its geographically distributed nature make it interesting to observe and to analyze. Furthermore, when we will introduce LastMile model in Chapter 2, we will see that this situation is a "worst-case" for that model, since typical peer-to-peer platforms with DSL connected nodes are more likely to follow the LastMile model than PlanetLab (due to their slower "last-mile" links more likely being a bottleneck of transmission, which is a central assumption for this model).

1.3 Experiment design

We performed two types of experiments: individual end-to-end throughput measurements, as well as contention experiments for measuring the performance achieved when multiple communications take place at the same time. To keep them as simple as possible, we concentrated on the particular situation with one sender and two receivers. On the considered platform, such a situation is enough to generate contention, and thus allows us to capture congestion and sharing mechanisms. The situation with two senders and one receiver would be interesting to observe as well, but we left it open as a possible extension to our work in the future. The original scenario already cost us much in terms of effort and time, and provided us with enough data to analyze extensively. It is also important to point that, due to asymmetry of Internet download vs. upload, the one-to-two scenario is much more likely to generate congestion than the two-to-one scenario.

In order to make sure that we observe steady-state, and thus avoid the slow-start mechanisms of TCP, the setting for measuring individual connection was as following: data is sent from the sending node to the receiving node on a TCP socket for 20 seconds . The first 15 seconds are not measured, and only used to "warm-up" the connection. The receiver measures how much data is received for the last 5 seconds, and uses this value to compute an average throughput over these 5 seconds. We present an example of such a measurement on Figure 3. Notice, that the sending node sends data only for 20 seconds, and even if receiving nodes still receive the data after 20

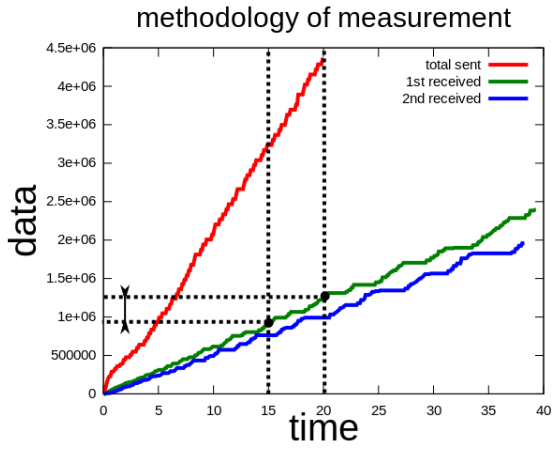

Figure 3: Plot of total cumulative data sent or received when one sending node transmits simultaneously to two receiving nodes.

seconds have passed, it doesn't count towards measured values. The length of measure was chosen as a compromise between usage of resources (daily cap on used bandwidth per node of PlanetLab) and precision of measurement. Contention experiments used a similar setting, with both receivers measuring how much data they receive on the last 5 seconds. This is a major difference from S^3 datasets, that were obtained using the Spruce tool [62].

With the dynamic nature of the PlanetLab platform, both kinds of measurements suffer from high variability. In order to overcome this variability, we perform at least 10 repetitions of the measurement for each configuration (the same sender-receiver pair for end-to-end measures, and the same sender-receivers triplet for contention measures). The variability of measures is analyzed in Section 1.5.

1.4 Datasets

The measurements performed are grouped into two datasets. The first dataset contains our end-to-end bandwidth measures. It was obtained by randomly selecting 50 nodes of PlanetLab, among which we performed measures over the course of one month, with the objective of having 10 measures for each sender/receiver pair. Due to PlanetLab limitations and in order to avoid exceeding daily limits for a node transfer, each node could be involved only in a few measurements daily. With a daily limits around 8GB of download, and one measurement taking even up to 200MB, or sometimes larger if the nodes involved happen to be in a network close proximity. The total transfer limitations of PlanetLab created a situation, where even with all nodes available all the time and with no connection problems, obtaining the full matrix would have taken months. Such a large timespan contradicts the idea of obtaining snapshot of network parameters at one precise moment. Even worse, because of unavailability and connection problems, we abandoned the starting target of the experiment, that is to obtain a complete matrix over a larger set of nodes. A subset of

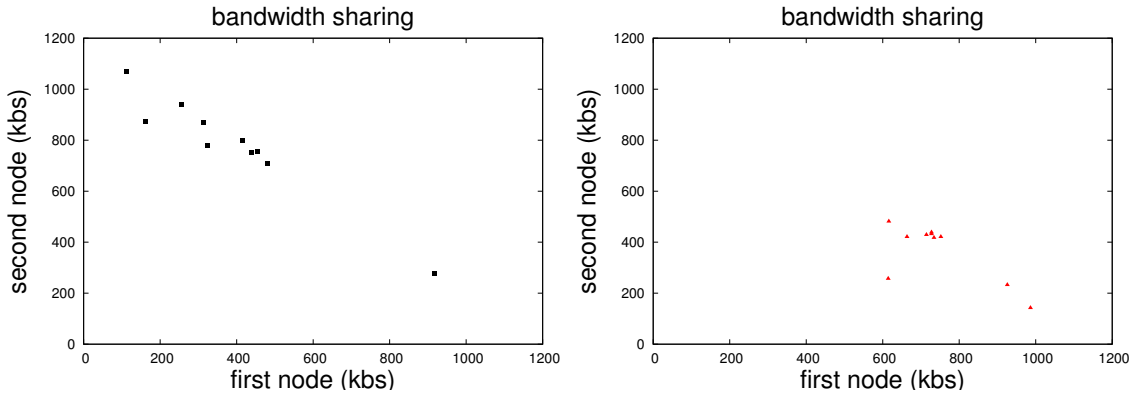

Figure 4: Two examples of shared bandwidth distributions when connecting repeatedly to the same two nodes.

15 most reliable nodes was selected (based on already obtained partial data), among which it was possible to obtain a complete set of measurements. This data was collected between December 20th, 2012 and January 16th, 2013.

The second dataset contains our congestion measurements. It consists of 87 measures of bandwidth shared between triplets of nodes (one sender and two receivers), where each measure over each triplet was performed 10 times in about 10 minutes of time. Triplets were selected at random among the set of 15 nodes which had been selected for the complete end-to-end measurements as described above. This data was collected between January 4th, 2013 and February 4th, 2013.

Examples of results of this type of measurements are on plots on Figure 4, where we plot measured source-to-first node bandwidth vs. measured source-to-second node bandwidth in a bandwidth sharing experiment. On both those plots the results seem to cluster on a $X + Y = const$ line, however we see strong variability of a distribution of individual values.

Both of these datasets are available as part of a larger project named `bedibe`, and can be downloaded at the following address: `https://gforge.inria.fr/frs/download.php/32092/data.zip`. We will discuss `bedibe` more thoroughly in a Chapter 2.

1.5 Data analysis

In this section, we perform statistical analysis of datasets mentioned above. We provide basic statistical analysis of point-to-point datasets, followed with analysis of bandwidth sharing datasets. We perform them to increase our confidence that we are in fact measuring network parameters, even when faced with noise and highly volatile environment on a large scale.

1.5.1 Point-to-point measurements

In order to evaluate the variability of our measurements, we focus on the subset of 15 nodes, for which we have performed a larger number of measurements. Between all pairs of nodes, a

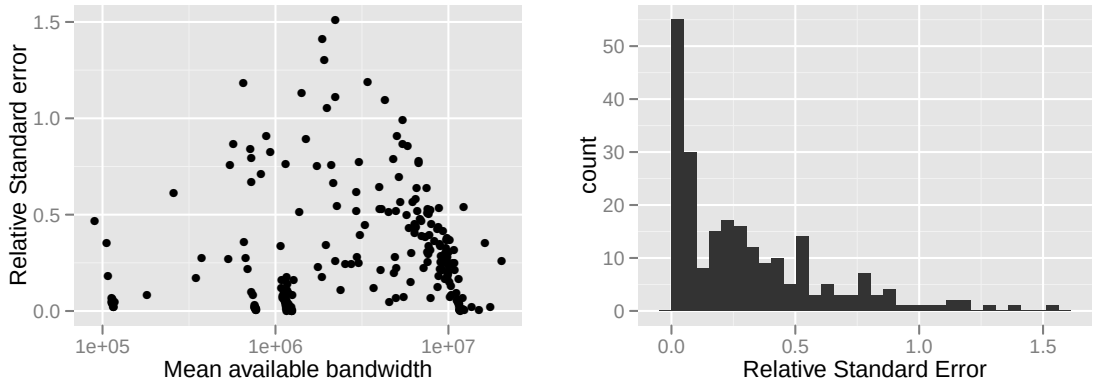

Figure 5: Plot of variability of individual measurements.

total of 10 individual bandwidth measurements were performed. This allows to compute mean and standard deviation for each given source/destination pair, and we express how variable the results of a measurement between two given nodes can be by computing the relative standard deviation (the standard deviation divided by the mean). On Figure 5, we plot relative standard deviation as a function of mean bandwidth, and distribution of relative standard deviation among all source/destination pairs. We can see that only a fraction of measures are stable. However, in significant number of cases the variability is high, with relative standard error around or above 0.5. This means that on average, we can be certain of the measure only up to a factor of 2. This variability is not a surprise, given that the PlanetLab platform is shared among many users, and this shows that providing estimates for the bandwidth is certainly challenging.

1.5.2 Bandwidth sharing measurements

In the bandwidth sharing experiments, we measure the throughput received by both receivers. Here also, we want to observe the variability of these measurements. For each configuration (*i.e.* for each choice of one sending and two receiving nodes), the measurement was performed 10 times during 1 hour, and reports the throughput received by each receiving node (b_1 and b_2), as well as the sum of these throughputs $b_{tot} = b_1 + b_2$. Similarly to the previous paragraph, for each configuration we compute the relative standard error for b_1, b_2 and b_{tot}, as an indication of how variable these measures are. The results are shown on Figure 6, where we plot distribution of the relative standard error of the throughput received by the first node, by the second node, and of the total throughput received by both of them. We can see on the plot, that the total throughput is much more stable than the individual throughput received by a given node. This high variability for individual values is the reason why, in Chapter 2, we will not attempt to predict how the sharing is done between the two receivers, but instead focus on predicting the total throughput.

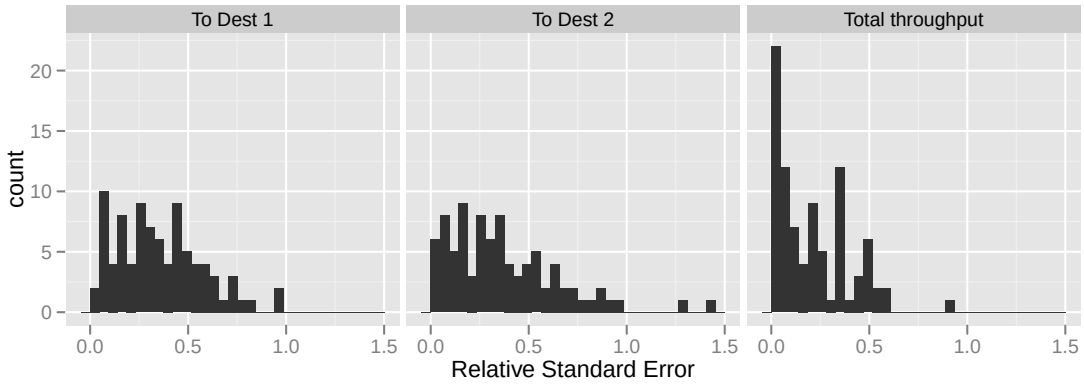

Figure 6: Plot of variability of sharing bandwidth measurements.

1.6 Conclusion

We have been able to get a consistent, reliable data in a reasonable time, using PlanetLab. Our initial aim to obtain data on how the bandwidth is shared between individual connections in a contention scenario proved to be too ambitious. As we saw on the results, even if the total throughput of a sending node is stable, the individual share of bandwidth is unpredictable when we repeatedly perform the same node connection configuration. However, higher stability of total sending throughput is a strong indicator of saturation of the sending node and applicability of the LastMile model in this scenario. Our point-to-point measurements provide better results, however scale of influence of various parameters is noticeable. Level of errors we obtain on this measurement is also a reason, that when we will be performing network predictions, we should not distinguish between perfect prediction, and prediction with a relative error of 50%, as that is usual error ratio the measurements come with.

Further work in this area is needed, as there is not enough bandwidth datasets publicly available to the research community. On the measurement side, it would be very useful to obtain larger datasets for available TCP throughput. But to achieve this, it seems necessary to use other platforms, since we believe we have achieved the limits of what is possible within PlanetLab. While it might be possible to use existing tools for obtaining point-to-point measurements, and thus obtain larger datasets, no tools for measuring bandwidth sharing is known to us. As there exists S^3 point-to-point dataset obtained on PlanetLab, using Spruce, we saw no point in duplicating that work. For congestion measurements, it would be interesting to perform measures of congestion/bandwidth sharing in more general settings. In this chapter, we concentrated on scenarios with one sender and two receivers. Scenarios with two (or more) senders and one receiver would allow to explore congestion on links to receiving nodes, instead of sending nodes. Our work allowed to observe congestion with no more than 3 nodes involved. Considering larger scenarios, *e.g.* one sender and many receivers or many senders and one receiver, would also be interesting, in order to confirm the results obtained in this chapter in more general settings.

Chapter 2

Evaluation of Bandwidth Network Prediction Algorithms

2.1 Introduction

In many Internet applications, network awareness is an important part of achieving good performance or lowering resource usage. In the case of delivering video on demand [63], or performing peer-assisted streaming [44] for example, estimations of available bandwidth allow the construction of an efficient overlay topology (*e.g.* SplitStream [17]). In this chapter, we cover the topic of estimating available bandwidth between the nodes located in the network. We present systems (centralized algorithms and their decentralized versions), that can be deployed onto the platform and allow us to ask for estimated available bandwidth between any two given nodes. Such a system should take into account the dynamic nature of the network, nodes entering or leaving.

In many applications (such as content delivery or video streaming), large amounts of data need to be exchanged between nodes, making available bandwidth the important metric for application performance. However, as we saw in Chapter 1, it is often not desirable to perform explicit end-to-end path measurements because of the high cost of such measurements. Single measurements are both time and resource consuming, and exhibit large discrepancy. Moreover the number of possible scenarios that should be covered by measures grows quickly with the size of the platform, typically as N^2, where N is the number of nodes in the platform. The solution provided should take care of a dynamic nature of the platform, *e.g.* nodes added or leaving. Also, the dynamic of the network itself is another challenge that a naive approach has trouble dealing with, as the edges can change its throughput in time or new edges can appear thus changing topology of the networks.

In the light of problems stated above, and in order to perform resource optimization in large scale platforms, it is thus necessary to summarize the network performance in the platform, in a way that providing such a summary can be done with a reasonable amount of measurements. In a topic of the latency-awareness, this idea has led to the design of Network Coordinate Systems (NCS), which embed the nodes of the platform in a metric space. Appropriate metric spaces and efficient algorithms have been proposed for estimating latency over the Internet (Vivaldi [20] is a good example). The idea of designating some nodes as a landmark nodes lead to the Global

Network Positioning (GNP) system, which embeds subset of nodes into a (high) dimension Euclidean space [51]. High dimension required for successful Euclidean embedding gave rise to the hyperbolic spaces embeddings [60]. Such systems should capture the large-core and long-tendrils structure of the Internet. A study measuring hyperbolicity, and treewidth of the Internet [49], resulting in a finding a large, well connected, inseparable core of the Internet (large treewidth), and high hyperbolicity (so the peripheral Internet tends to be separated from each other). Main problem with metric space embeddings are the triangle-inequality-violations: a situation where a direct distance measured between two nodes A and B is larger than sum of two distances between a third node C and both of nodes A and B. Works measuring the range and effect of the phenomenon have been done, and solution by weakening the distance function into a inframetric has been proposed [23]. Another worth noting solution is Sequoia [57], which tries a different approach of embedding network into tree structure. The tree Sequoia embeds is built from virtual nodes, that does not necessarily reflect physical network infrastructure, and only leaves of the tree reflect actual network nodes.

As we will see in Section 2.2, the metric space embedding is an approach that translates poorly into the bandwidth estimation, and a new approach is required. In order to predict bandwidth, our main focus in this chapter is on the LastMile model [10] and on Decentralized Matrix Factorization (DMF) [43]. Both are good candidates for available bandwidth prediction algorithms, mainly because they are able to give asymmetric estimation, which is impossible for all NCS based on a metric space embedding. In Section 2.2 we will discuss several other prediction algorithms (Vivaldi, Sequoia, Iterated LastMile).

In Section 2.3, we analyze the estimation precision of both DMF and LastMile algorithms, based on measurements datasets obtained on PlanetLab from S-cube [67]. We show that matrix factorization techniques are quite efficient at predicting point-to-point available bandwidth. LastMile provides slightly less accurate estimation, but we also provide a variant of the algorithm which improves the predictions quality and provides ones comparable to DMF ones. Both DMF and LastMile outperform previously existing algorithms.

Then, in Section 2.4, we go beyond estimation of point-to-point performance and address the problem of congestion. The challenge is: *is it possible to predict the performance obtained when several communications take place at the same time?* When several large communications take place at the same time, they are expected to interfere with each other. Being able to predict and model these interferences is crucial for optimizing the resource usage of an application. On the one hand, algorithms like DMF or Sequoia do not seem to easily deal with such scenarios. On the other hand, the LastMile model actually specifies that communications can happen in parallel, as long as both the outgoing and the incoming bandwidth limits of each node are fulfilled. In this chapter, we evaluate performance of bandwidth prediction algorithms on a data described in details in Chapter 1, that is bandwidth sharing data, a dedicated measurements performed on PlanetLab specifically to study this question. Using this data, we show that LastMile allows to perform this prediction with a reasonable accuracy.

The Chapter 3 will cover the topic of algorithms constructing an efficient overlay networks for such a platform, under the assumption that LastMile model applies.

2.2 Network Prediction Algorithms

In this section, we present different models for network predictions. Several models have been developed for latency estimations (Vivaldi, GNP, DMF). We also present models for general network distance estimation (Sequoia). In such a situation, one tries to estimate the general network distance function d, that can be defined either in terms of latency $d(C_i,C_j) = \text{latency}(C_i,C_j)$ or in terms of bandwidth $d(C_i,C_j) = \frac{1}{BW(C_i,C_j)}$ or $d(C_i,C_j) = \text{maxbw} - BW(C_i,C_j)$. With this transformation in mind, we can even try to apply latency estimation into the bandwidth estimation (for example we will do such with DMF). The only model designed purely for bandwidth is the LastMile model.

Network Coordinate Systems have received a lot of attention recently, especially in the context of latency estimation. Original systems, like GNP [50], relied on landmarks to make the predictions – landmarks are special nodes whose positions are computed first, and all nodes of the system compute their position with respect to these landmark nodes. Afterwards, more distributed systems, like Vivaldi [20], have been designed, in which all nodes have the same role, leading to more precise and robust estimations. The term Network Coordinate System comes from the fact that all those systems embed the nodes in a metric space, hence assigning coordinates to all nodes, and use the distance in this space as an approximation of latency.

A notable exception to this is the Matrix Factorization [43] technique, in which the rationale is to approximate the distance matrix by a low rank matrix by assigning a column and a row vectors to all the nodes of the system. Matrix Factorization has been originally designed for latency estimation [46], and later extended to estimate network performance classes [41]. However, in this chapter we are interested in estimating available bandwidth, and it seems natural to observe how well Matrix Factorization performs in this context.

Another exception to this is Sequoia, in which the algorithm tries to embed the nodes of the network into one or several (virtual) trees. Sequoia has been the first attempt to provide bandwidth estimation, but in fact it provides general network distance estimations. Plenty of work [49] has been done in the area of treeness, as well as the connectivity of the Internet, measuring various metrics of the Internet.

Hyperbolic space has been proposed as a replacement for the Euclidean space embeddings. The rationale behind this is that when trying to embed Internet into the Euclidean space, one has either to choose high dimension of the space, or suffer from the problem of having not enough space on the periphery to represent ending nodes of the network as having high distance between themselves.

2.2.1 Shape of the Internet

In this section we will present various results of experiments performed to define the ”shape” of the Internet. Those results, while not applicable in a small scale solutions (platforms deployed locally), are important when designing solutions working not in an arbitrary network, but in a geographically distributed hosts deployed over the Internet, *e.g.* peer-to-peer networks or live streaming.

An experimental evaluation measuring the treewidth of the Internet has been done [49]. An evaluation was performed on the experimental datasets, obtained as follow. For a router level, a set of Internet hosts (IP addresses) was chosen. For those hosts a connectivity graph was obtained, containing as vertices both hosts and intermediate nodes. Paths were discovered using traceroute utility tool. For Autonomous System (AS) level, the IP interconnections between two IP addresses can be used to infer an AS interconnection between two ASes who advertise IP prefixes (such links are observed at the BGP routing level or at IP level).

A *tree decomposition* of $G = (V,E)$ consists of a tree T and a subset $V_t \subset V$ associated with each node t of T (called a "bag"), that satisfies following properties

- every node of G belongs to at least one bag of T,
- for every edge $e \in E$, there is a bag V_t containing both ends of e,
- the collection of bags containing a given node of G induces a connected subtree of T.

The *width* of the tree decomposition $(T, \{V_t\})$ is defined to be one less than the maximum size of a bag. The *Treewidth* of the G is the minimum width of a tree decomposition of G. Intuitively, a treewidth of a graph answers the question of how far away the structure of the graph is from the tree one. Calculating of exact value is NP-complete, however heuristic approach can be applied to get bounds on the value. Treewidth can be seen as a measure of the global connectivity of the graph. Tree has treewidth of 1, while n vertices grid graph has a treewidth of $\sqrt{n}$.

Treewidth of connectivity graphs described as above show follows the growth of $\text{treewidth}(G) \approx \sqrt{n}$, which means that a rather large number of vertices need to be removed from the graph to disconnect it. That is a strong evidence for existing of a well connected core of the Internet, composed of a small set of very well connected set of hosts.

Another evaluation of the Internet graph (with distances defined as a number of hops), was performed as follow. *Hyperbolicity* of a graph G, denoted as $\delta(G)$ is defined as follow: first, for arbitrary quadruple of nodes x,y,z,t, we define:

$$\delta(x,y,z,t) = \frac{d_1 - d_2}{2},$$

where $d_1 \geq d_2 \geq d_3$ are sorted in a nondecreasing order values of $d(x,y)+d(z,t)$, $d(x,z)+d(y,t)$, $d(x,t)+d(y,z)$. Then, we simply define $\delta(G) = \max_{x,y,z,t} \delta(x,y,z,t)$. Hyperbolicity measures how far graph is from the tree.

Even stronger finding is that, if one takes not max, but the average over all quadruples of pairs, that the average hyperbolicity is relatively small. The conclusion of the measures is that the distances (in hop counts) between any four hosts in the Internet is on average behaving like they were vertices of a tree, with almost all quadruples having $\delta(x,y,z,t)$ either 0 or 1.

2.2.2 Symmetry violations

One of the major steps in improvement over bandwidth predictions was moving from *symmetrical* algorithms to *asymmetrical* ones. It is no longer a valid assumption, that $\text{BW}(C_i,C_j) \approx \text{BW}(C_j,C_i)$,

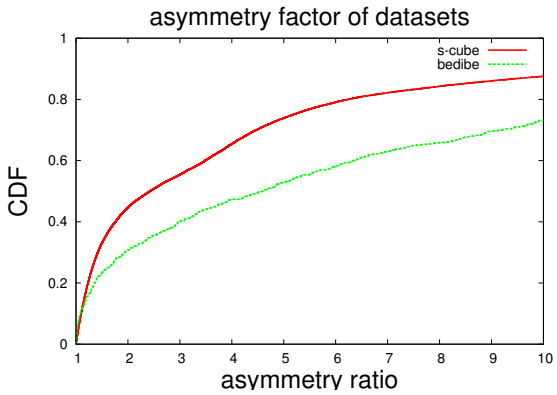

Figure 7: An *asymmetry ratio* cumulative distribution function (CDF), taken for all edges in the dataset. Calculated for S^3 dataset and `bedibe` dataset.

while it was a valid assumption for latencies. On Figure 7, we present a CDF plot of an *asymmetry ratio* (a relative error between two symmetrical measured bandwidth) taken for all possible pairs:

$$\max\left(\frac{BW(C_i,C_j)}{BW(C_j,C_i)},\frac{BW(C_j,C_i)}{BW(C_i,C_j)}\right).$$

So, we in fact plot what fraction of pairs have the error equal-or-less than a given value. As we can see, errors can be of a rather large magnitude. For example, in S^3 dataset, 40% of values measured differ by a factor 4 or larger from their symmetrical counterparts, and almost 20% have a factor 9 or larger. This means, that the best we can hope for when estimating using symmetrical algorithm, is that for 40% values measured we will have an error of a factor 2 or larger, and that for 20% values measured we will have an error of a factor 3.

2.2.3 Global Network Positioning

Global Network Positioning (GNP) was the first proposed network prediction system. It models the Internet as an Euclidean space, where any end host is assigned coordinates [51]. Small number of distributed hosts called *landmarks* measure inter-landmark distances. The system then computes the coordinates of each landmark in the Euclidean embedding, by minimizing the discrepancy between measured distances and computed distances. Each host in the platform knows the landmark nodes, and their coordinates. When localizing itself in the network (for example, node enters the network), node has to measure its actual distance to each landmark. Node then computes its own coordinates that minimize the overall discrepancy between measured distances to the landmarks and computed distances.

Actual performance of the algorithm depends on policy of choosing landmarks. Experimental evaluation shows that it achieves reasonable predictions on latencies. But the dimension of the Euclidean space has to be set to at least values of $d = 7$.

2.2.4 Vivaldi

Vivaldi was presented first in [20], and is a latency prediction algorithm. It is meant to be simple and lightweight algorithm for assigning a synthetic coordinates to the hosts. Basic idea that the algorithm was developed around is embedding the hosts into an Euclidean metric space, to mimic the geographic-related behavior of a large scale Internet.

Vivaldi works by searching for coordinates $x_1, \ldots, x_n$ minimizing the value of an error function:

$$E = \sum_{i,j} \left(L_{i,j} - \|x_i - x_j\| \right)^2$$

for a given L as an input matrix. Minimization is performed by mass-spring simulation. Based on Hooke's law, for each pair of nodes i, j, the force generated by j on i by the displacement of the spring connecting them is:

$$\overrightarrow{F}_{i,j} = \left(L_{i,j} - \|x_i - x_j\| \right) \times \mathbf{u}(x_i - x_j).$$

Note that $\mathbf{u}(\overrightarrow{v})$ is the unit vector in the direction of $\overrightarrow{v}$. The algorithm simulates the "movement" of the nodes in the parameter space, by calculating total force at each step of iteration

$$\overrightarrow{F}_i = \sum_j \overrightarrow{F}_{i,j}$$

and by simulating the small progression of time

$$x_i' = x_i + \delta t \cdot \overrightarrow{F}_i.$$

Algorithm is adapted to work with a partial input matrix, and it handles in an easy way updates of information. It can be also easily adapted into a decentralized environment, where each node computes its own coordinates, and performs its own set of measures. This way the algorithm can also handle the changing environment and other nodes joining/leaving the network.

The general algorithm presented here can be adapted to a various metric spaces. It works best, when the metric space is an Euclidean one paired with directionless *height* assigned to each node, that is included to capture latency overhead on the access links of single-homed hosts.

2.2.5 Sequoia

In general, algorithms specialized in latency predictions have poor performance when applied to bandwidth estimations. Sequoia was the first attempt to develop bandwidth estimation algorithms [57], but it can be used for latencies also. It uses the notion of *distance* between two nodes of the network, being it either the latency measure, inverse bandwidth measure, or constant minus bandwidth measure. Note, that the distance is not necessarily a metric distance, as it can *violate triangle inequality*, having

$$d(C_i, C_j) > d(C_i, C_k) + d(C_j, C_k)$$

In fact, various analysis of latency datasets show that between 10% and 40% of triplets violate the triangle inequality [57]. We also have seen previously that bandwidth distance also violates symmetry, having $d(C_i, C_j) \neq d(C_j, C_i)$.

Sequoia relies on embedding network latency and bandwidth onto trees, using the notion of *prediction trees*, where end hosts at the leaf level connected via a network of virtual inner nodes model latency or bandwidth. This work differs from system trying to reconstruct internal topology of the Internet, in the sense that it operates using virtual nodes, that doesn't necessarily reflect existing gateways and routers. Sequoia virtual trees could be used to construct a hierarchically organized distributed system if required.

For each virtual tree, Sequoia designates one node as a *lever*, that acts as a reference for the tree. It means that for every other node in this tree, the distance in the tree between this node and lever is the same as real-life measured metric (bandwidth or latency, depending which one we are estimating).

In each virtual tree Sequoia chooses another node, named *anchor*. Having lever R, anchor A, while adding node C_i, algorithm will try to preserve exact distances $d(R,A)$, $d(R,C_i)$ and $d(A,C_i)$.It achieves it by connecting C_i to the tree through new virtual node S on a path connecting R and A, such that it preserves all three distances. The node S should be selected that

$$d(s,R) = \frac{1}{2}\left(d(A,R) + d(C_i,R) - d(A,C_i)\right).$$

The virtual construction proposed above has one disadvantage, that is the trees constructed there are of very poor structure (path with attached single edges). That is why Sequoia itself uses another abstraction, called *anchor trees*. It represents exactly the same set of predictions as corresponding prediction tree, but is a well balanced structure. It does not use virtual nodes, and network nodes can be intermediate nodes in the tree. On those type of trees, every operation (joining/leaving of a node, new measurement) can be performed with a time efficient computation.

A different architecture version of the algorithm have been proposed [57], ranging from fully centralized one, through partially centralized (one machine builds prediction trees, machines can perform queries by itself), to fully decentralized one.

An advantage of Sequoia is that it mimics the treeness of the Internet in its prediction trees. Another strong point is that it does not assume much about the distance on the network, and works fine even on datasets with a lot of triangle-inequality-violations. The applicability of the algorithm to any form of network distance is an advantage, as the system is very generic, but it is also a source of disadvantage, as Sequoia does not use any specific properties of bandwidth measurements and bandwidth related metrics. This proves important when we compare the performance of Sequoia to other, more bandwidth-specific algorithms.

2.2.6 Decentralized Matrix Factorization

Matrix Factorization (MF) has recently been proposed as a novel approach for distance estimation in the Internet [46]. This approach strongly differs from other models, as it does not attempt to make any assumption on the underlying network. Rather, the objective is to approximate the bandwidth matrix (*i.e.* the n by n matrix $\mathscr{M}$ such that $\mathscr{M}_{i,j}$ is the measured available bandwidth

between C_i and C_j) by a low rank matrix. In MF, we thus search for matrices P and Q (of dimension $n \times d$ and $d \times n$ respectively, where d is a fixed parameter) such that the product $P \times Q$ is "close to" the measured matrix $\mathscr{M}$. If the closeness property is defined by the quadratic error

$$err(P,Q) = \sum_{i,j} (\mathscr{M}_{i,j} - (P \times Q)_{i,j})^2,$$

then optimal P and Q can be computed from $\mathscr{M}$ by SVD (Singular Value Decomposition), or by iterative optimization if some values are missing.

A fully distributed algorithm has also been proposed [43],[42], named DMF, in which each node is in charge of its own values in matrices P and Q, and iteratively, nodes optimize their values based on the current values of their neighbors. The algorithm uses method named Stochastic Gradient Descent (SGD), founded on the stochastic optimization theory. The decentralized setting features neither requirement for explicit construction of matrices nor special landmark nodes and central servers. The network nodes exchange messages with each other, and matrix factorization is collaboratively and iteratively achieved at all nodes.

It has been shown experimentally that this algorithm converges [42], and gives actually better latency estimates that Vivaldi algorithm; this algorithm has also been used to perform classification of paths in either "good" or "bad" performance [41]. As far as we know, there is no experimental evaluation validation of DMF for bandwidth estimation.

2.2.7 LastMile

The LastMile model [10] was derived with aim at simplicity and easiness of derivation of algorithms operating on it. The LastMile model is based on the assumption that contention only happens at the periphery of the network (on the "last-mile" link that connects each participating node to the core network). This assumption is realistic in many scenarios (like for example when the system consists of home computers connected to the network by DSL connections), hence this model has been used in several studies to design or analyze communication algorithms for video-on-demand [16], peer-assisted streaming [44] or master-slave tasking [9].

The LastMile model assigns to each node C_i an outgoing bandwidth b_i^{out} and an incoming bandwidth b_i^{in}, and then the available bandwidth between two nodes $BW(C_i,C_j)$ can be estimated by $LM(C_i,C_j)$, which by definition is equal to

$$LM(C_i,C_j) = \min(b_i^{out}, b_j^{in}).$$

Below, we present experimental evidences for the correctness of the LastMile model. If we take a fixed node C_i (that has associated value of b_i^{out}), and look at all the predicted bandwidth values "to" all the possible nodes C_j (with associated values of b_j^{in}), the value treated as a function of b_j^{in}, is either equal to b_j^{in} (for values of b_j^{in} smaller than b_i^{out}), or is equal to a (constant) value b_i^{out} (for values of b_j^{in} greater than b_i^{out}).

On the Figure 8 we show evidence supporting that behavior described above takes place. We plot there, for a selected node, all the values of outgoing measured bandwidth, sorted. On the left plot, we see that smallest 50 values resemble even distribution of values. We also see that 50th

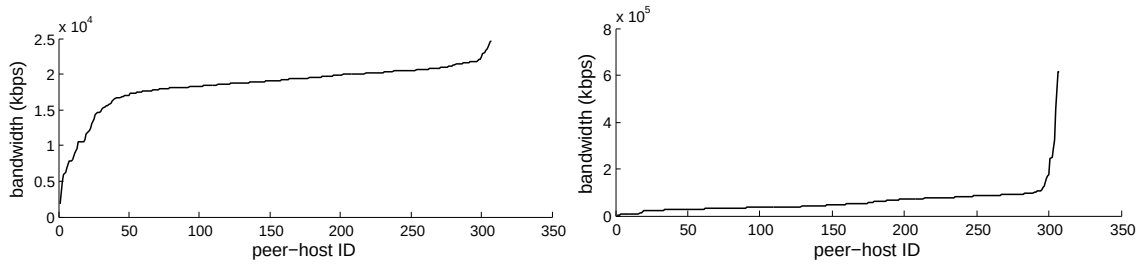

Figure 8: Plots showing values of measured outgoing bandwidth, sorted, for a given selected node from a S^3 dataset (planetlab7.millennium.berkeley.edu on the left,planetlab3.hiit.fi on the right). Plots taken from [10].

smallest value is very close to the 300th one, and only few values marginally exceed that value. We interpret it as the outgoing bandwidth of that node is limited by a value, and it determines the speed of connection for all connections except 50 smallest ones, where the other node limitations determine the speed.

The plot on the right of Figure 8 shows a different behavior, with few values greatly exceeding the speed limit we could otherwise infer. We can explain this fluctuation by a few nodes being in a very close network proximity, for example the same campus network, or other WAN.

Basic LastMile

LastMile bandwidth estimation algorithm is a simple approach to calculating the corresponding values of b^{out}, b^{in}. It relies on the fact, that in the scenario where measured values $BW(C_i, C_j)$ are perfectly predicted by $LM(C_i, C_j)$ calculated as above, we would have

$$b_i^{out} = \max_j LM(C_i, C_j),$$

$$b_j^{in} = \max_i LM(C_i, C_j).$$

This finding gives us a first initial assignment of

$$b_i^{out} = \max_j BW(C_i, C_j),$$

$$b_j^{in} = \max_i BW(C_i, C_j).$$

However, such a setting is potentially dangerous, as only one bogus measurement is enough to obtain wrong prediction for a node. Furthermore, the last-mile assumption is not always satisfied in practice, as it may happen that some nodes share a bottleneck link. This observation motivates the removal of a few extremal values before computing b^{out} and b^{in} values. The solution proposed is to define b_i^{out} an $(1-\alpha)$ percentile for some small α constant, of all measured values $BW(C_i, C_j)$. This solution is a generalization of previous approach, with $\alpha = 0$ reducing it to taking max.

Iterated LastMile

In order to improve on basic LastMile algorithm, a following approach, similar to Vivaldi approach, is proposed. We will iteratively change the values of b^{out} and b^{in}, to minimize the value of:

$$E = \sum_{i,j} \left(BW(C_i,C_j) - LM(C_i,C_j)\right)^2.$$

Particularly, substituting b_i^{out} as x, above takes the form:

$$E(x) = \text{const} + \sum_j \left(\min(x, b_j^{in}) - BW(C_i,C_j)\right)^2$$

for which calculating value of x minimizing above function can be done in $O(n)$ time.

By iterating above steps, we eventually end with a set of new values b^{out} and b^{in}, which give us optimal values of $LM(C_i,C_j)$ in the sense of square-sum-difference from $BW(C_i,C_j)$.

2.3 Evaluation of point-to-point Prediction Algorithms

In this section, we present an evaluation of the precision of LastMile (both in its plain and iterated versions, as described in 2.2.7) and DMF, obtained using a software named `bedibe` (which itself will be described in a Section 2.5).

2.3.1 Comparison methodology

There are many ways to measure the efficiency of a bandwidth prediction system. On the one hand, we should be able to compare the predictions to the actual data, and on the other hand, be able to simulate fact that algorithms are only given partial data. Our methodology involves taking a (possibly) full matrix of measurements M, and taking a partial measurement M'. Algorithm returns prediction matrix $\mathrm{P}(M')$. We can then calculate some metric $d(M, \mathrm{P}(M'))$. For example, we could use:

$$d(A,B) = \sum_{1 \le i,j \le n} (A_{i,j} - B_{i,j})^2$$

However, one downside of a single value metric is that it does not give us insight on detailed performance of an algorithm. To remedy this problem, we use different evaluation technique. To evaluate the precision of estimation algorithms, we use the standard *relative error* metric, which is defined as $e = \max(\frac{p}{v}, \frac{v}{p})$, where p is the predicted value and v is the actual measured value. This relative error is computed for every corresponding pair of matrix elements, and we are interested in the *distribution* of all relative error values on all pairs.

2.3.2 DMF and LastMile

In our comparisons there are two datasets used: the first one is a snapshot of the PlanetLab platform of April 2th, 2010 taken from the (now discontinued) S-cube project [67], which contains available

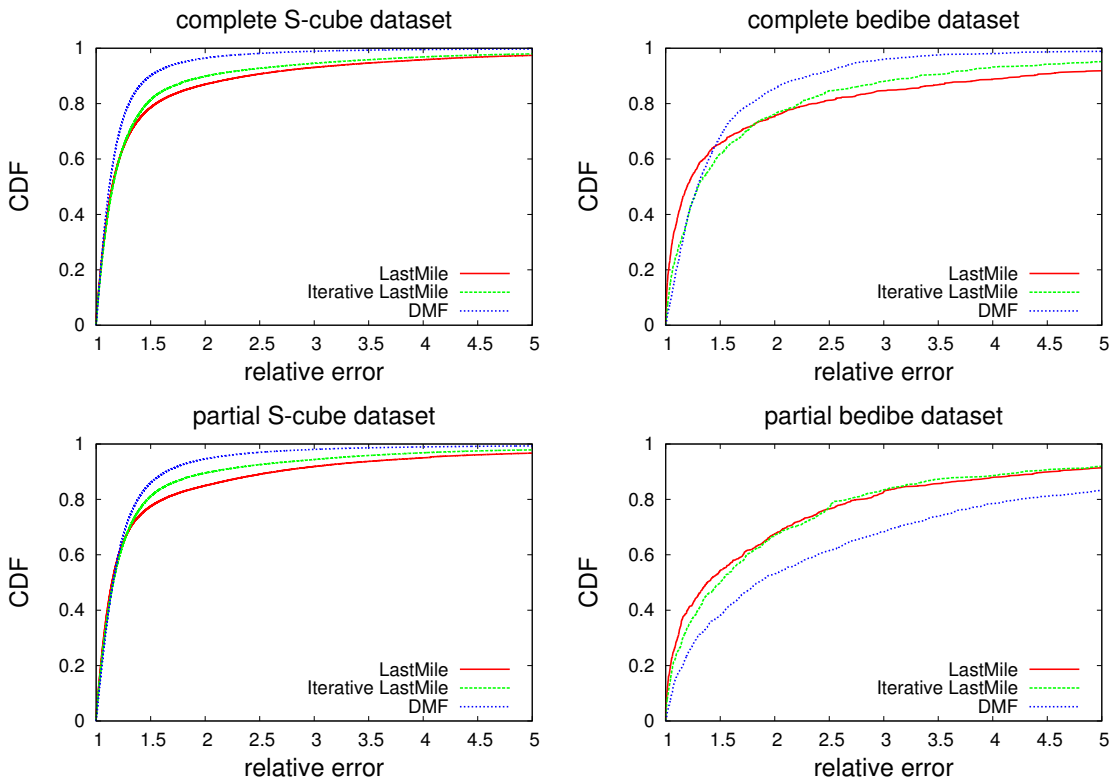

Figure 9: Comparison of performance of DMF ($d = 10$) and LastMile, done on S-cube dataset (on the left), `bedibe` dataset (on the right), full information (top) and partial information (bottom).

bandwidth measurements between 426 hosts, with many missing measurements, and we extracted a set of 308 hosts for which the complete measurement matrix was available. The second one is obtained with `bedibe` measurement methodology (described in Section 1.4), and contains achievable TCP throughput between 50 hosts of PlanetLab. In this dataset, about 17% of measurements are unavailable. In order to account for the fact that performing all end-to-end measurements is not reasonable in a practical setting, a subset of 20% of values are randomly chosen and given to the estimation algorithms. Note however that algorithms provide estimations for all values, and that the relative errors are computed on the whole matrix (for comparison, we also provide plots how algorithms perform given full matrix as an input).

These results are shown on Figure 9, where we plot the Cumulative Distribution Function of relative error for each estimation algorithm. For example, a point at coordinates $(1.5, 0.8)$ for DMF on the S-cube dataset means that for this dataset, 80% of all source/destination pairs are predicted with an error below 1.5. Hence, plots closer to the upper left corner of the graph represent better estimations.

The analysis of Figure 9 shows several facts. On the S-cube dataset, DMF (with $d = 10$) clearly outperforms other algorithms, and provides reasonably good predictions, even when partial information only is available. Matrix Factorization technique is actually a very strong approach,

as it packs the necessary information (measurement matrix) in the least available space. We expect DMF to consistently outperform other methods.

We can also see, as in [10], that the iterated improvement of LastMile helps to obtain better predictions, but the improvement is not that significant.

On the `bedibe` dataset, DMF is only able to provide reasonable predictions if given the full matrix, and its performance drops by a large factor when given only partial data. The precision of LastMile suffers a much smaller drop and outperforms DMF on partial data. We suspect it is the result of sensitivity to the parameter selection of DMF, as `bedibe` dataset is small in size in comparison to S^3 one.

The `bedibe` dataset is harder to estimate than the S-cube dataset: the 80th percentile relative error is less than 1.5 for all algorithms on the S-cube dataset, but is around 2 for the `bedibe` dataset. It is worth remembering that S^3 dataset measured available bandwidth, but `bedibe` one measured throughput. Also, S^3 one was obtained using external tools, while `bedibe` one using the brute-force approach. We believe it leads to less self-correlation, more noise and less smoothness of the `bedibe` dataset, making it overall harder to estimate. Size of the dataset also makes difference, as larger datasets should be more self-consistent.

2.4 Evaluation of one-to-many Prediction Algorithms

In this section, we study the possibility to predict the total achievable throughput when a sending node sends data to several receiving nodes. Being able to predict how the bandwidth is shared between several connections, or how the network behaves in the case of congestions is important, as single isolated connections are an artificial assumption. Congestion can happen in various different scenarios:

- at the first edge of connection, *i.e.* one node sending to several nodes at once,
- in the middle of connection (the general case),
- at the last edge of connection, *i.e.* one node receiving from several nodes at once.

In light of all the experimental evidences supporting LastMile, it seems reasonable to consider only first and last case, as those cases are most likely to cause congestion. Also, experimental evaluation of congestion happening in the middle seems out of our reach.

As we have seen in the Sections 2.2.2 and 2.2.1, Internet seems to be dominated by asymmetrical links connecting peripheral nodes to the core Internet, with download dominating over upload capacity. This way it seems reasonable to experimentally evaluate the case where the congestion is going to happen on the sender side, as two receivers are more likely to saturate one sender than in the opposing scenario. By similar argument we restrict ourselves to the case of two receiving nodes, because this is already enough to achieve congestion and thus obtain meaningful results.

For this study, we use both `bedibe` datasets described previously, in Section 1.5.2. The datasets were collected on the PlanetLab platform, by selecting at random three nodes, one as a sender, two as a receivers, and measuring the achieved bandwidth to both of the receiving nodes when

establishing connection simultaneously and trying to push as much as possible data. For all the nodes used in this experiment, we also have full matrix of point-to-point bandwidth measurements, so we are able to look at what LastMile algortihms or DMF can deduce from the matrix and compare it to actually achieved measurements.

We will try to predict the value $TT_{i,j,k}$, being the amount of data node i is able to transmit to nodes j and k simultaneously. The reasoning behind this is that, as seen in the Section 1.5.2, total throughput achieved by the sender has smaller variability than the individual values per each receiver.

For this prediction, we tried several possibilities:

- **LastMile**: we compute LastMile values (incoming and outgoing parameters) for all nodes, and we use the LastMile assumption to predict the available bandwidth: it is either limited by the sending capacity of the sender, or by the sum of the receiving capacities of the receivers: $\mathscr{P}^{LM}_{i,j,k} = \min(b^{out}_i, b^{in}_j + b^{in}_k)$.

- **Avg, Sum, Max**: we use end-to-end individual measurements between the sender and each of the receivers, and use the average, total, or maximum value as a prediction:

$$\begin{aligned}
\mathscr{P}^{avg}_{i,j,k} &= \frac{BW(C_i,C_j) + BW(C_i,C_k)}{2} \\
\mathscr{P}^{sum}_{i,j,k} &= BW(C_i,C_j) + BW(C_i,C_k) \\
\mathscr{P}^{max}_{i,j,k} &= \max(BW(C_i,C_j), BW(C_i,C_k))
\end{aligned}$$

- **AvgDMF, SumDMF, MaxDMF**: we compute the DMF predictions for the available bandwidth between the sender and each of the receivers, and similarly use the average, total, or maximum value as a prediction.

$$\begin{aligned}
\mathscr{P}^{avgDMF}_{i,j,k} &= \frac{DMF(C_i,C_j) + DMF(C_i,C_k)}{2} \\
\mathscr{P}^{sumDMF}_{i,j,k} &= DMF(C_i,C_j) + DMF(C_i,C_k) \\
\mathscr{P}^{maxDMF}_{i,j,k} &= \max(DMF(C_i,C_j), DMF(C_i,C_k))
\end{aligned}$$

  For this analysis, we have set the dimension value of DMF to $d = 5$.

We then compute the relative error of each prediction, with the same definition as in Section 2.2: $e^X_{i,j,k} = \max(\frac{TT_{i,j,k}}{\mathscr{P}^X_{i,j,k}}, \frac{\mathscr{P}^X_{i,j,k}}{TT_{i,j,k}})$. To evaluate all prediction techniques, we thus analyze the distribution of the error ratios for all the measured triplets in our experiments. The results are shown on Figure 10, and mean and median values are reported in Table 1.

The results show that **Sum** and **SumDMF** provide very bad estimations. This comes from the fact that in PlanetLab nodes, the congestion often takes place at the sending node, even with only

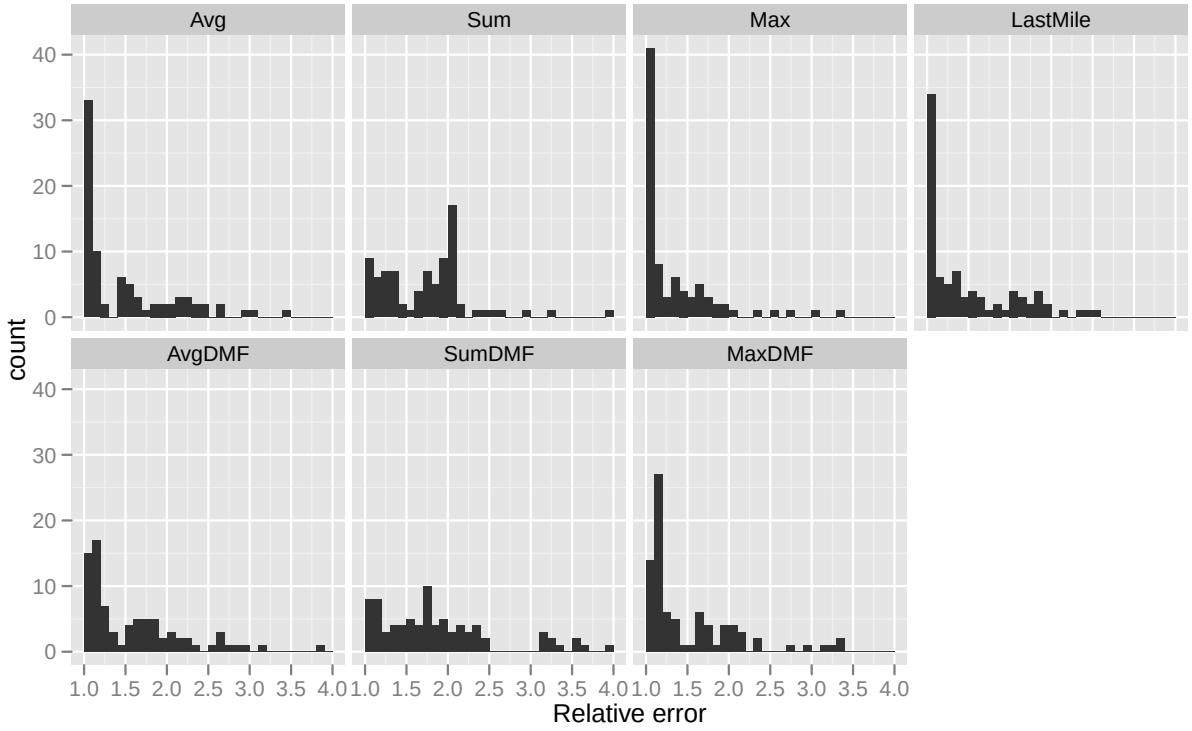

Figure 10: Distribution of error ratios for 7 estimates of total throughput when one sender sends to two receivers.

Est.	LM	Avg	Sum	Max	AvgDMF	SumDMF	MaxDMF
Mean	**0.851**	1.32	1.62	0.983	1.89	2.04	0.941
Median	0.236	0.217	0.823	**0.15**	0.564	0.755	0.218

Table 1: Mean and median relative error for 7 estimation techniques

two receivers. Hence summing the individual performance of both receivers yields a large overestimate of the actual total throughput. This explains why using average or maximum values give better predictions. Actually, using the **Max** estimate gives the best results in most cases, because the maximum measured individual throughput is very often close to the outgoing bandwidth of the sender. This explains that the median error ratio of **Max** is much lower than all other estimates. For the same reasons, **MaxDMF** also has a rather good median error ratio, not as good as **Max** because of the imprecisions incurred by DMF. However, in some cases **Max** and **MaxDMF** provide estimates which are off by a larger factor, whereas **LastMile** predictions are more stable, as can be seen by the lower mean error ratio.

It is important to note that in a practical setting, **Max** estimates can only be obtained by actually performing both individual end-to-end measures, whereas by design **LastMile** and **MaxDMF** can be computed for all possible triplets by using only a smaller number of measurements. Furthermore, from an algorithmic point of view, these results encourage the use of the LastMile model for the design of bandwidth allocation algorithms, even in settings where the "LastMile" assumption is not completely valid.

2.5 Bedibe

We aim at keeping our results accessible and reproducible. That is why we decided to make the datasets used in this work publicly available (`https://gforge.inria.fr/frs/download.php/32092/data.zip`). We also decided to make available the software package used for generating the results available.

`bedibe` is a tool for benchmarking bandwidth and latency estimations. We created it with a purpose of the development, testing, benchmarking and visualization of bandwidth estimation algorithms. It is written in Python, a very popular scripting language. It can be downloaded at `http://bedibe.gforge.inria.fr/`. Below in this section we provide motivation for creation of this tool, followed with technical details.

Our basic motivation for the creation of `bedibe` was to create an environment, where we could evaluate the performance of different network prediction algorithms in an automated way. When working on possible improvements to algorithms like LastMile, we found ourselves without either a unified collection of datasets, or implementations of algorithms that would work on datasets we had access to. Thus, the first step was to standardize the format of datasets (and convert existing other ones), and provide implementation of state-of-the art prediction algorithms. From then, we followed with providing ways to visualize results of prediction algorithm.

Data is read from CSV files (Comma Separated Values), with source and target hostnames, and measured values. For example we provide S^3 datasets (see Section 1.1.3) and our SPLAY datasets (see Section 1.4) in this format. Because network measures admit large variations, datasets often contain multiple values per pair of nodes. Because of this, we provide a set of functions for easily picking representative value or set of values (in case of fuzzy computation) from several of them.

We provide implementations of the state-of-the art estimation algorithms: DMF, LastMile, Sequoia, Vivaldi (described in Section 2.2). LastMile is provided in both its basic and iterated

versions. We also provide our implementation of two-level LastMile. To further simplify implementation of the algorithms, we also provide a library of decorators (function transformation, idea borrowed from a functional programming paradigm) that for example, allow us to transform functions that operate on matrices into ones working in our environment of CSV style data. Other decorators allow easy control over what type of preprocessing an algorithm supports.

Estimations done by algorithms can be stored into CSV files, or used in several comparison tools. Our module for comparing allows easy comparison of the error ratio between different measures or estimations, or calculation of various standard metrics (stress, 80th percentile error...). Visualization tools allow to create images out of the computed matrices (assigning different colors to high/low bandwidth, or accurate/inaccurate estimation). Additionally, a plotting module can be used to produce (using `gnuplot`) different types of plots, for example a CDF of the dataset, or a CDF of relative error of the estimates. Various parameters of the plots can also be customized from inside the code.

In summary, we created a tool that greatly simplifies the implementation of an algorithm, and with a few extra lines of code it allows the algorithm to be tested on datasets we provide, tweaked, analyzed or compared to other algorithms. Finally, the tool can be used to output plots useful for performance visualization. The plots provided in this thesis were generated using this tool.

Below we provide an example of a python code used to generate plots in a scientific paper. The code shows how in a few lines, a program can read the dataset, prepare the partial dataset (by choosing subset of measurements), run a set of prediction algorithms on partial dataset and then generate the CDF plots of comparison between original dataset and prediction results.

Listing 2.1: bedibe - example of usage

```
def main(args):
    [V] = inp.init([args[1]])
    M = make_complete(V)
    M = select(M, getfirst)
    A = neighbours(M, 32)
    LM = lastmile(A, logexp=False, iterated=True, alpha=0.1)
    DMF = dmf(A, logexp=False, d=10)
    SEQ = sequoia(M, theoretical=True, nb_trees = 15)

    with plot.Plot("Figure4a") as p:
        p.add_plot(plot.simple_comp(LM,M),"lastmile -0.1-32")
        p.add_plot(plot.simple_comp(DMF,M),"DMF-10-32")
        p.add_plot(plot.simple_comp(SEQ,M),"Sequoia")
```

2.6 Future works: improving LastMile

In this chapter we advocate for usage of the LastMile model when dealing with algorithmic problems related to bandwidth. One of the strengths of the LastMile model lies in its simplicity. This, and the fact that we are not predicting *generic* network distances, just bandwidth, gives us a lot of

room for improving and tweaking the model and algorithms.

Our aim is to have a model that can give reliable predictions. But we also would like the model to reflect somehow real structure of the network. Our belief is that by using a network model that is close to the real life structure of the network, we could obtain better predictions. As we saw, opposite implication is not true: DMF gives very synthetic and abstract representation of the network, but does not give us any "knowledge" on the structure of the network. For every node of the network i, the model encodes its parameters as a vector v_i. DMF states that prediction is the scalar product of the vectors,

$$\mathrm{DMF}(C_i, C_j) = v_i \cdot v_j.$$

It is hard to imagine a model of network that would lead us to such a prediction scheme, both for latencies and bandwidth.

Apart from DMF and LastMile, every other model we presented in this thesis is using somehow the ideas of metric, distance and embedding. This way of constructing predictions is a result of those models being created mainly with aim of predicting latencies. The intuition comes from the treating the latency as a measure of a distance signal has to travel in the network. This leads to the way we combine edges into the paths in those types of models:

$$\mathrm{dist}(p) = \mathrm{dist}(e_1) + \ldots + \mathrm{dist}(e_k)$$

for a path p composed of edges $e_1, \ldots, e_k$ and where dist() represents the latency incurred by this path or edge.

If we translate previous equation by the usual way the generic network distance prediction algorithms translate latency to bandwidth, that is by taking inverse, we get:

$$\frac{1}{\mathrm{BW}(p)} = \frac{1}{\mathrm{BW}(e_1)} + \ldots + \frac{1}{\mathrm{BW}(e_k)}.$$

LastMile is based on a simpler principle, which we believe to be closer to the actual network:

$$\mathrm{BW}(p) = \min\left(\mathrm{BW}(e_1), \ldots, \mathrm{BW}(e_k)\right).$$

Two-level LastMile

We discussed in Section 2.2.1 what structure the Internet exhibits when measuring various parameters, *i.e.* strongly interconnected core of the network, and larger peripheral network having more of a tree-like structure. LastMile model captures the second behavior (see Figure 11), but fails to take into account the structure of the core itself. Also, LastMile model fails to take into account the possibility of nodes sharing a bottleneck link (for example belonging to the same academic network).

In light of problems listed above, we would like to propose the following improvement upon the LastMile model, a *two-level* LastMile. The way the model works is to group existing nodes $C_1, \ldots, C_n$ into k distinct sets $G_1, G_2, \ldots, G_k$. Let us name the function assigning node to the corresponding sets containing it a *GID* (a Group ID):

$$\mathrm{GID}(C_i) = G_j \text{ iff } C_i \in G_j.$$

Our model can be represented as a graph of (see Figure 12):

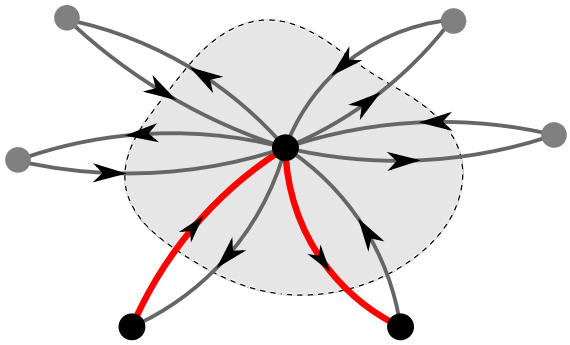

Figure 11: Picture of a graph representation of a LastMile model, with thick red edges used to communicate between nodes in black.

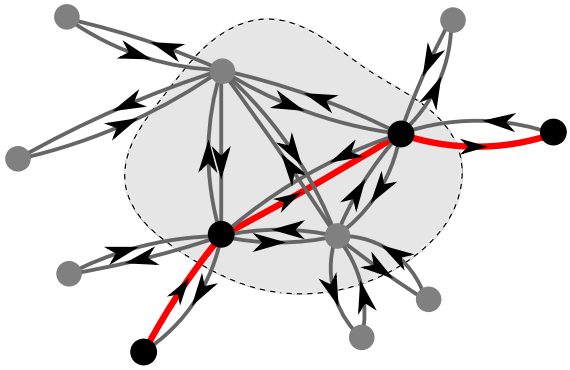

Figure 12: Picture of a graph representation of a Two-level LastMile model, with red edges used to communicate between nodes in black.

- virtual nodes $G_1, \ldots, G_k$ represent the core of the Internet, and they form a full clique graph with edges
$$(G_i, G_j) \text{ for } 1 \leq i, j \leq k,$$
- nodes $C_1, \ldots, C_n$ represent the peripheral Internet, and they are connected to the core by the edges
$$(C_i, \mathrm{GID}(i)) \text{ for } 1 \leq i \leq n.$$

Since each edge is weighted (in fact, it represents two directed weighted edges), our model reduces to a set of values $b_1^{in}, b_1^{out}, \ldots, b_n^{in}, b_n^{out}$ and a full $k \times k$ matrix $F[i, j]$ for $1 \leq i, j \leq k$. The prediction of bandwidth between nodes i and j is

$$LM(C_i, C_j) = \min(b_i^{out}, F[\mathrm{GID}(C_i), \mathrm{GID}(C_j)], b_j^{in}).$$

It is worth noticing that if we set $k = 1$, our model collapses back into LastMile model. On the opposite, if we choose $k = n$, we can then encode any $n \times n$ matrix into the G matrix. We suspect that setting $k \approx \sqrt{n}$ is a fine choice, making the number of parameters of both parts of model equal.

Notice that if we have been given in advance the values of cluster, finding a best match for a given set of measurements is solvable by an algorithm analogous to the one used in Iterative LastMile Algorithm (Section 2.2.7). However, we could not find a "good" method to perform the clusterization step. On the one hand, assigning nodes to clusters at random resulted in very poor performance. On the other hand, performing any non-random clusterization requires ability to decide if nodes are close (so they should be in the same cluster) or distant (so they should be in different clusters), which brings us back to the problem of predicting network distance. Probably a new approach is required if we want to develop this idea further.

2.7 Conclusion

In this chapter, we analyzed the possibility to provide estimations for available bandwidth in large scale platforms. Estimation techniques and models exist for the latency metric, but their extension to available bandwidth is not always possible. We focused on the LastMile model, widely used as a communication model in algorithmic works, and on Decentralized Matrix Factorization (DMF), originally designed for latency estimation. We show that DMF is able to provide very good estimations for available bandwidth as well, but that its precision drops when fewer measurements are available.

DMF provides better bandwidth predictions than any other algorithm. That should not be surprising, as DMF is in some sense optimal as an algorithm to encode matrix in a compressed form of a low rank matrix. However, this performance comes at a cost, that DMF has no intuitive model underlying. It does not rely at all on the fact that the estimated values are bandwidth, and exactly the same algorithm is used for latencies. The LastMile model, on the opposite, was developed with bandwidth in mind. It assumes a certain model of the underlying network. That will prove advantageous in Chapter 3, in which we develop an overlay network construction algorithm for broadcast on large-scale platforms.

Furthermore, we analyzed contention in the presence of multiple simultaneous communications. We used data obtained from the PlanetLab platform, and we observed that the total throughput can be predicted by the LastMile model with a precision almost comparable to estimations based on the complete set of measurements. That is another strong point of LastMile based algorithms, as the model of the network it is based upon takes into account naturally the multiple simultaneous communications, and that is not the case for other algorithms.

We already saw, in Chapter 1, how to perform measures over network parameters (bandwidth, latency). In this chapter we covered the topic of efficiency of measures, *e.g.* that we do not need to measure every single edge in case of point-to-point measures. We were able to perform predictions over the network when we assumed a certain model of a physical behavior of the network. By assuming LastMile model, we were able to predict both point-to-point communications, and to a certain degree also congestions. A natural step would be to try and apply this model in a more general setting. However, as we move up in the layers of networking, we lose control over certain details of communication (*e.g.* routing). When designing large scale communication scheme over large networks (*e.g.* Internet), we are unable to control routes used by point-to-point communications. This situation shows a clear advantage that LastMile model have in such setting over other

models, that is no matter what routing would be chosen for fixed point-to-point communications, links connecting node to a network are always used by this communication. Another advantage of the LastMile model is that it does not contain any assumption about the inner structure of network.

Chapter 3

Broadcasting over LastMile

3.1 Introduction

3.1.1 Context and Motivation

Data dissemination in distributed platforms has been the subject of a vast literature. The problem comes into two flavors, depending on the context. On the one hand, if the topology of the platform is known (in the case of computer networks or parallel machines for example), the goal is to organize data transfers so as to maximize the throughput (or minimize the makespan for a given message size). On the other hand, in the context of a large scale Internet level platforms, the goal is to find the topology (*i.e.* the overlay network) that maximizes the throughput.

The one-to-all broadcast, or single-node broadcast, is the most primary collective communication pattern: initially, only the source processor holds the data that needs to be broadcast; at the end, there is a copy of the original data residing at each processor. Parallel algorithms often require to send identical data to all other processors, in order to disseminate global information (typically, input data such as the problem size or application parameters). Numerous broadcast algorithms have been designed for parallel machines such as meshes, hypercubes, and variants (see among others [32, 66, 64, 47]).

The same framework applies for broadcasting a live stream of data, such as a movie or a TV show. In the context of content distribution systems, it is at the core of live streaming distribution systems such as CoolStreaming [69], PPLive [65] or SplitStream [17]. In this case also, we are interested in the distribution of a large message to all the nodes of a large scale platform, made of a large number of computers, geographically distributed, and interconnected by the Internet. In the context of this work, it is thus not possible to obtain the actual topology of the core of the network, and we are rather interested in application-level solutions. Thus, the goal is to build an overlay network that makes the best possible use of the communication capabilities of all participating nodes, so as to maximize the overall streaming rate (once steady-state has been reached).

In the context of large scale Internet platforms, it is common to assume that the communication between two nodes is only limited by the available outgoing bandwidth of the sender and by the incoming bandwidth of the receiver. This assumption is also very suited to the case where nodes are connected to the Internet with low bandwidth links, like DSL for example. In that case, the

bandwidth limitation is either physical (from the link capacity) or logically enforced at the user's request. In large scale platforms, it is also desirable to limit the number of connections that must be handled simultaneously at each node. Both of these assumptions are common in the context of data dissemination in large scale platforms. However, they fail to correctly model the behavior of the nodes located behind a NAT or a firewall. As we will see, adding this constraint on node connectivity capabilities strongly modifies both the algorithms and the theoretical results.

In previous chapters we justified LastMile as a network model. Because it deals with a problem of congestion, and discards any knowledge on inner structure of the network, LastMile model seems well suited for dealing with the problem of broadcasting over a network. In this setting, we will deal with an optimization problem, where each node has fixed limit on both incoming and outgoing bandwidth. We are interested in designing a communication scheme where we maximize the size of a message that one node is able to broadcast to all other nodes, in a steady streaming fashion (we discard in our analysis the notion of time, assuming that transmissions are defined by a rate of data transfer), while keeping limits imposed on bandwidth used by nodes. In our communication scheme, we try to use all available resources to a network, not only the upload of the source, by making other nodes exchange parts of original message in a peer-to-peer fashion. Secondary way to measure efficiency of the solution is to keep track of the number of nodes a given node has to communicate with, that is the degree of a node. We will both try to develop solutions with unconstrained degree, and with small degrees.

In this chapter, we use the notion of "acyclic" solution, meaning that the solution treated as a graph can be sorted topologically. By "cyclic" solution, we mean any solution, not just necessarily non-acyclic. The main novelty of our approach is the classification of the set of participating nodes into two parts: open nodes that stay in the open-Internet and "guarded" nodes that lie behind firewalls or NATs. Two guarded nodes cannot communicate directly, but rather need to use an open node as a gateway for transmitting a message. In the presence of guarded nodes our main contributions are a closed form formula for the optimal cyclic throughput and the proof that the optimal solution may require arbitrarily large degrees. In the acyclic case, we propose an algorithm that reaches the optimal throughput with almost optimal degree used. Then, we prove a worst case ratio between the optimal acyclic and cyclic throughput and show through simulations that this ratio is on average very close to 1, what makes acyclic solutions efficient both in terms of throughput maximization and degree minimization.

This broadcast problem was introduced in [7], however in a slightly different version, with nodes having both bandwidth and strict degree constraints, and open nodes only. The solutions provided are based upon resource augmentation, through relaxation of the degree constraint while still enforcing strict bandwidth constraint. Results from sections 3.3.1, 3.3.2 and Theorem 3.5.2 , where we prove that it is possible to reach the optimal throughput, at the price of a quasi-optimal (up to a small additive increase) degree of the participating nodes, are based upon [7], but reworked to a new setting without strict (independent from bandwidth) degree constraints.

In summary, our goal is first to design an overlay network and to determine the bandwidths associated to the edges of the overlay, such that both degree and capacity constraints are satisfied, such that nodes behind a NAT or a firewall use third party nodes to communicate and such that the overall throughput that can be reached using this overlay network is close to the optimal one. One

of the major contributions of this chapter is to study, under a realistic communication model and for a classic communication scheme, the impact on the complexity and on the performance of the algorithms of having nodes lying behind NATs and firewalls.

3.1.2 Outline

Below we summarize the results presented in this chapter, in the context of previous results. We either provide new results, or adapt results from [7] into a modified model of degree constraints.

1. In the presence of **open and guarded nodes** or with **open nodes only**, finding the best **acyclic** solution and finding the best **cyclic** solution are both NP-Complete in the strong sense (this is a new result, presented in Section 3.3.1).

2. In the presence of **open nodes only**, the optimal **acyclic** throughput can be achieved at the price of a small linear increase of **1** in the degree of the nodes (see Section 3.3.2, where we recall results from [7]).

3. In the presence of **open and guarded nodes**, the optimal **acyclic** throughput can be achieved at the price of a small linear increase of **3** in the degree of the nodes (see Section 3.4, this is a new result).

4. In the presence of **open nodes only**, the optimal **cyclic** throughput can be achieved at the price of a small linear increase of **2** in the degree of the nodes (in Section 3.5, we briefly recall the result from [7]).

5. In the presence of **open and guarded nodes**, the optimal **cyclic** throughput can be achieved only with an **unbounded increase** in the degree of the nodes (see Section 3.5, this is a new result).

6. For any instance, the optimal **acyclic** throughput is at least **5/7** of the optimal **cyclic** throughput (see Section 3.6, this is a new result).

7. On average (see Section 3.6), for a wide variety of realistic scenarios, the throughput of the **low degree acyclic solutions** proposed by our algorithms are very close to the **optimal cyclic** throughput (at most 5% decrease).

Therefore, except in the cyclic case with open and guarded nodes, and despite the strong NP-Completeness result, it is possible to build low degree solutions that achieve optimal throughput at the price of a small increase in the degree bound. Moreover, if the complexity of proofs dramatically increases from the acyclic open case to the cyclic open case and to the acyclic guarded case, all proposed algorithms are very efficient in time complexity and can therefore be used in practice.

The situation strongly differs in the cyclic guarded case: since arbitrarily large degrees are required in order to achieve the optimal throughput, so that we cannot rely on small degree increases to obtain optimal performance. Nevertheless, we prove in Section 3.6 that the algorithm that returns low degree solutions in the acyclic guarded case is a $\frac{5}{7}$-approximation algorithm for the cyclic guarded case.

3.2 Models and Related Works

3.2.1 Platform Modeling

The bounded multiport model has already been advocated by Hong et al. [27] for independent tasks distribution on heterogeneous platforms. In this model, node C_i can communicate with any number of nodes C_j simultaneously, each using a bandwidth $c_{i,j}$, provided that its outgoing bandwidth is not exceeded, *i.e.*, $\sum_j c_{i,j} \leq b_i^{\text{out}}$. Similarly, node C_i can receive messages from any number of nodes C_j simultaneously, each using a bandwidth $c_{j,i}$, provided that its incoming bandwidth is not exceeded, *i.e.*, $\sum_j c_{j,i} \leq b_i^{\text{in}}$. This corresponds well to modern network infrastructure, where each communication is associated to a TCP connection.

This model strongly differs from the traditional one-port model used in scheduling literature, where connections are made in exclusive mode: each node can communicate with a single node at any time step. But in the context of large scale platforms, in which the networking heterogeneity ratio may be high, it is unreasonable to assume that a 10GB/s server may be kept busy for 10 seconds while communicating a 10MB data file to a 1MB/s DSL node. Therefore, in our context, we will assume that all communications are directly handled at TCP level. Nevertheless, in order to keep the flavor of the one-port model, we will minimize the number of connections that need to be handled simultaneously at a given node. This constraint is particularly important in a context where QoS mechanisms are used to fix or bound the bandwidth associated to each communication (each TCP connection in practice). It is worth noting that at the operating system level, several QoS mechanisms enable a prescribed sharing of bandwidth [21, 1, 70]. In particular, it is possible to handle simultaneously several connections and to fix the bandwidth allocated to each connection. In our context, it has been proved in [12] that these mechanisms are necessary since the bandwidth allocated to the connection between C_i and C_j may be lower than both b_i^{out} and b_j^{in}. Therefore, the variant of the LastMile model we propose encompasses the benefits of both the bounded multi-port model and the one-port model. It enables several communications to take place simultaneously, which is compulsory in the context of large scale distributed platforms. Practical implementation is achieved by using TCP QoS mechanisms and by bounding the number of connections.

However, this model fails to correctly model the behavior of the nodes located behind a NAT or a firewall. This issue is crucial in the context of Peer-to-Peer applications running over the Internet. For instance, in distributed applications such as Skype [5, 26] or Bittorent [31], NATs play a crucial role, since in certain situations where "hole punching" techniques [61] fail, it can be impossible for a pair of nodes to communicate directly. In this case, the technique consists in using a third party node that acts as a relay for the packets. At a higher level, we can classify the nodes between open and guarded nodes, where open-open, open-guarded (and guarded-open) connections are possible, but not guarded-guarded. As we will see, adding this constraint on node connectivity capabilities strongly modifies the algorithms and the theoretical results.

3.2.2 Related works

Broadcast and streaming optimization have already been the subject of several studies in the literature. However, none of them has considered the constraint added by the presence of firewalls in

the system. The work closest to our approach is by Liu et al [45] in which they provide bounds for the streaming rate, the upload rate of the source needed to ensure a given stream rate, and the depth of the distribution trees produced. Degree constraints are also considered in their work, but with specific limitations. In particular, the degree of the source is not limited, and the degree constraint on nodes is considered separately for each tree of the solution, which means that the actual degree of each node is not limited.

More applied studies have been published, which focus on designing distributed algorithms to build the streaming overlay. For example, CoolStreaming [69] builds upon a gossip-based overlay to propose a distributed streaming algorithm. This algorithm inherently includes degree limitations, and provides a guarantee on the diameter of the overlay, but no guarantee about the streaming rate is available. On the other hand, SplitStream [17] is based on a distributed hash table and builds an overlay made of k different distribution trees, with a probabilistic guarantee on the streaming rate. Furthermore, SplitStream allows multicast (some nodes may only receive the data from a subset of the k trees – but they do not choose which part) and also includes a degree limitation, which is typically k times larger than the degree of our solutions.

3.2.3 Positioning

In this chapter, we assume that the network can be represented using the LastMile model. We already presented ways to obtain this representation in previous chapters.

Our contribution consist in computing, using this instantiated model, the overlay network (which nodes communicate together) and the bandwidths that should be allocated to each edge of the overlay in order to maximize the overall throughput of the collective communication scheme, given the bandwidth, the degree and the connectivity constraints of the network. The resulting weighted graph can be decomposed into a set of weighted broadcast trees [58, vol B, Chapter 53]. This decomposition specifies which data should be sent on which edge at a given time step.

In order to avoid this decomposition step, which is difficult to use in practice, we rely on the randomized broadcasting algorithm proposed by Massoulié in [47]. This algorithm is fully decentralized and is even able to deal with small variations of resource performance due to its randomized and dynamic nature. This algorithm requires knowledge of the topology of the network with bandwidths on edges and no contentions on the nodes, which is in general not realistic. On the other hand, the overlay network that we build has exactly, by construction, these properties, provided that bandwidth sharing mechanisms are used for the communications in order to limit the bandwidth of a communication to the weight of the edge, such as proposed in [21, 1, 70]. The overlay network that we build can therefore be used as direct input of Massoulié's algorithm.

Therefore, on the one hand by relying on measurements to instantiate the parameters of the LastMile model and on the other hand on Massoulié's algorithm to actually perform the broadcast operation, our algorithmic contribution provides a practical solution to the streaming problem whose approximations (due to the model, to the use of approximation algorithms for NP-Complete problems and to the decentralized and randomized implementation of the broadcast) can be rigorously analyzed and controlled. In this work, we focus on the approximation algorithms perspective.

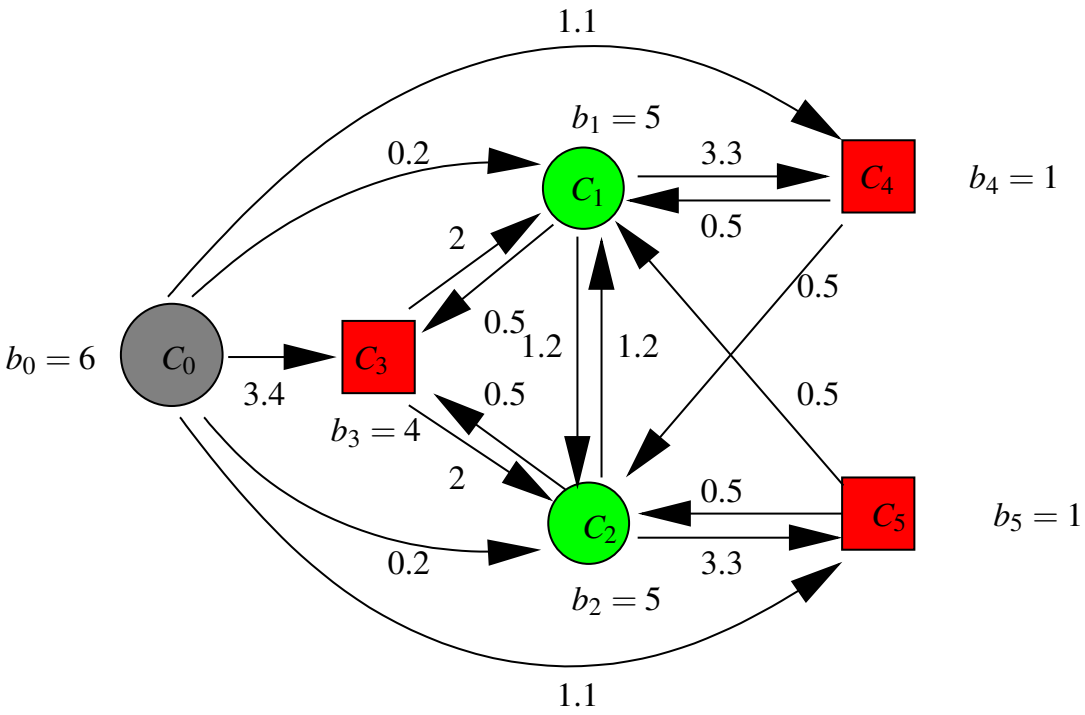

Figure 13: An instance with $n = 2$ open nodes (in green) plus the source (in gray), $m = 3$ guarded nodes (in red) together with an optimal broadcast scheme of throughput 4.4. In this broadcast scheme, the outdegree of the source is $o_0 = 5$, the outdegree of every guarded node is $o_3 = o_4 = o_5 = 2$ and the outdegree of the two open nodes is $o_1 = o_2 = 3$.

3.2.4 Model and notations

We consider a situation in which a *source* node, denoted as C_0, wants to broadcast a message. The recipients are partitioned into two sets: on the one hand some nodes belong to the *open-Internet*, and can communicate with each other freely (we call them *open* nodes); on the other hand some nodes can communicate only with nodes of the open-Internet, because they are behind a firewall or behind a NAT router (we call them *guarded* nodes). The source itself is supposed to be an open node.

An instance of our problem is specified by the number n and m of open and guarded nodes, and by the outgoing bandwidth b_i of each node C_i for $i \in [\![0, n+m]\!]$. The source node is C_0, nodes C_i for $i \in \mathscr{O} = [\![1, n]\!]$ are open nodes, and nodes C_i for $i \in \mathscr{G} = [\![n+1, n+m]\!]$ are guarded nodes.

The output of the problem is a broadcast scheme, defined by values $\{c_{i,j} | (i, j) \in [\![0, n+m]\!]^2\}$, where $c_{i,j}$ indicates the rate at which node C_i sends data to node C_j, subject to the following constraints:

- $\forall i \in [\![0, n+m]\!], \sum_j c_{i,j} \leq b_i$ (bandwidth constraint)
- $\forall (i, j) \in \mathscr{G}^2, c_{i,j} = 0$ (firewall constraint).

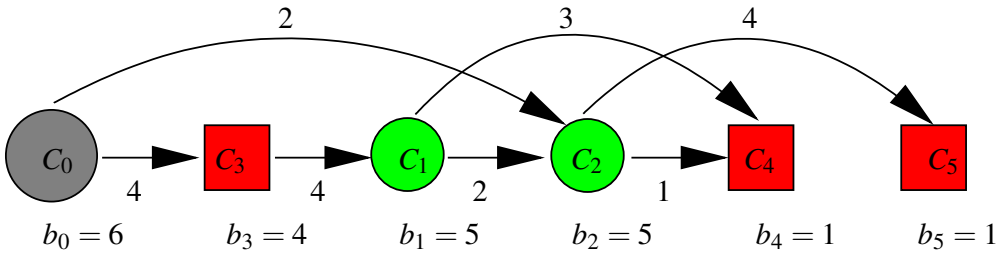

Figure 14: An acyclic broadcast scheme of throughput 4 on the instance of Figure 13. The order associated with this acyclic broadcast scheme is $\sigma = 031245$.

We implicitly assume that the input bandwidth of each participating node is large enough. The *throughput* of a broadcast scheme is given by $T = \min_{i \in [\![1,n+m]\!]} \{maxflow(C_0 \to C_i)\}$, where the flows are computed on the weighted graph described by the $c_{i,j}$s. Furthermore, given a broadcast scheme, we can define the *outdegree* of a node C_i as the number of nodes to which C_i actually sends some data, *i.e.* $o_i = |\{j,\ c_{i,j} > 0\}|$. Notations are illustrated in Figure 13.

As stated above, we want to provide broadcast schemes with small degree. To define what small degree may mean, let us note that in a solution of throughput T, the weight of any edge $c_{i,j}$ is at most T (indeed, receiving data at a rate larger than T is useless). Hence, if node i uses all of its outgoing bandwidth, then its outdegree o_i is at least $\left\lceil \frac{b_i}{T} \right\rceil$ and small degree therefore means o_i close to $\left\lceil \frac{b_i}{T} \right\rceil$. A solution that achieves (a fraction of) T^* by using an outdegree $o_i \leq \left\lceil \frac{b_i}{T^*} \right\rceil + d$ is thus a d-additive resource augmentation (approximation) algorithm. As a consequence, we do not consider strict degree constraints, but rather analyze the outdegrees used by our solutions in terms of $\left\lceil \frac{b_i}{T} \right\rceil$.

Computing a solution that achieves throughput T^* and such that the degree of each node is at most $\left\lceil \frac{b_i}{T^*} \right\rceil$ turns out to be an NP-complete problem (see Section 3.3.1), even for the special case where all nodes are open ($m = 0$). We are thus interested in this chapter in *approximate* solutions, both in terms of throughput (with respect to T^*) and additive resource augmentation on the degrees (with respect to $\left\lceil \frac{b_i}{T^*} \right\rceil$).

We prove that the situation differs if we concentrate on *acyclic* or more general *cyclic* solutions. A broadcast scheme is said to be *acyclic* if its communication graph (represented by the matrix c) is acyclic, which is equivalent to the existence of an order σ on the nodes such that

$$\forall i, j \in [\![0, n+m]\!], i > j \Rightarrow c_{\sigma(i),\sigma(j)} = 0.$$

This condition states that $\sigma(i)$, the node at position i in the ordering σ cannot feed $\sigma(j)$, the node at position j, if $i > j$. Figure 14 shows an example of an acyclic broadcast scheme associated with the order $\sigma = 031245$.

For a given instance and a given order σ, we denote by $T^*_{ac}(\sigma)$ the optimal acyclic scheme compatible with the order σ. For a given instance, we denote by T^*_{ac} the optimal acyclic throughput:

$$T^*_{ac} = \max_{\sigma} T^*_{ac}(\sigma).$$

3.3 Simple Case

As we have already noted, the problem comes into several flavors. Among the parameters that strongly influence the complexity of the problem are (i) the presence of guarded nodes (nodes behind firewalls) and (ii) the acyclicity of the solution, what leads to four different problems. In Section 3.3.1, we recall from [7] the proof that all four problems are NP-Complete. Because we use slightly different model, that leads to modified proofs, not using arbitrary degree constraints d_i (that are not present in our model, but were present in [7]). In Section 3.3.2, we concentrate on the easiest case, *i.e.* the case where we are looking for an acyclic solution with open nodes only.

3.3.1 Complexity Results

Introduction

In the case where guarded nodes are present, some of the communications are forbidden, what makes the combinatorial structure of the problem more complex. On the other hand, searching for an acyclic solution limits the search space and we will see throughout this chapter that it actually makes the problem easier. Of course, cyclic solutions can achieve higher throughput than acyclic ones, but we will see later in Section 3.6, that this only holds up to a (small) ratio $5/7$.

NP-Completeness

We prove in this section that the problem of finding an optimal allocation while satisfying the degree constraints (keeping $o_i \leq \left\lceil \frac{b_i}{T} \right\rceil$) is strongly NP-Hard. We prove this result by reduction to the **3 PARTITION** problem.

3 PARTITION: Let $a_i,\ 1 \leq i \leq 3p$ be $3p$ integers, such that $\sum_1^{3p} a_i = pT$ and $\forall i,\ \frac{T}{4} < a_i < \frac{T}{2}$. Is there a partition of the a_is into p disjoint sets $S_j,\ 1 \leq j \leq p$ containing exactly 3 elements and such that each set sums up to exactly T?

3 PARTITION is well-known to be NP-Hard in the strong sense [24]. Given a particular instance of **3 PARTITION**, let us consider the following instance $\mathscr{I}$ of our problem (see Figure 15), in which all nodes are open.

- The source (the upper node in Figure 15) has outgoing capacity $b_0 = 3pT$;
- $3p$ intermediate nodes (middle nodes in Figure 15), where $\forall 1 \leq i \leq 3p,\ b_i = a_i$;
- p final nodes (lower nodes in Figure 15), where $\forall 3p+1 \leq i \leq 4p,\ b_i = 0$;
- The target throughput to achieve is T.

If a solution to the **3 PARTITION** instance exists, then it is easy to build a solution to $\mathscr{I}$: the source serves all intermediate nodes with rate T, and intermediate nodes that correspond to the same set S_j serve a final node C_{3p+j} at their full capacity (see Figure 15).

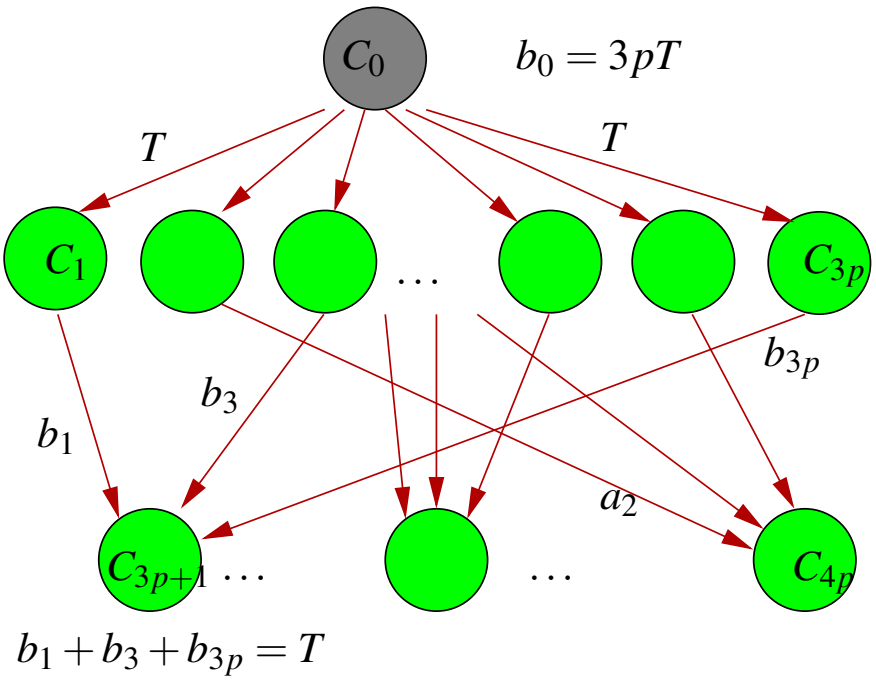

Figure 15: Solution used for the instances used in the reduction.

Conversely, let us assume that there exists a solution to $\mathscr{I}$. We first note that since the total outgoing bandwidth is exactly $4pT$ and $4p$ nodes need to receive the message at rate T, it is not possible to waste any bandwidth. Hence, the source necessarily sends data at rate exactly T (*i.e.* the maximal useful possible rate) to $3p$ nodes (*i.e.* the maximal number of clients since $\left\lceil \frac{b_0}{T} \right\rceil = 3p$), and each intermediate node C_i sends data at rate a_i (its maximal rate) to exactly another client (the maximal number of clients since $\lceil \frac{a_i}{T} \rceil = 1$). On the receiving side, at most $3p$ nodes are served by the source, and the intermediate nodes collectively serve the remaining p nodes (note that nodes served by intermediate nodes may be intermediate nodes themselves, so that the situation is slightly more complicated than depicted in Figure 15). Since no bandwidth is wasted, the sum of the weights of the incoming edges for such a node is exactly T. Furthermore, since $\forall i,\ \frac{T}{4} < a_i < \frac{T}{2}$, there are exactly 3 such incoming edges. It is thus possible to build a solution to the original **3 PARTITION** instance.

Note that this NP-Completeness result applies to all four different situations. Indeed, let us first remark that considered instances only contain open nodes, so that the NP-Completeness a fortiori holds in (the more complicated case in) presence of guarded nodes. Second, we a priori search for a general cyclic solution but we nevertheless prove that the optimal throughput (for our instances) can be reached using an acyclic solution (see Figure 15), so that above proof also shows that finding the optimal acyclic solution is NP-Hard.

The problem belonging in NP follows from the fact that **Maxflow** in arbitrary graphs is in P (and therefore in NP). This plus strong NP-Hardness shown above gives desired NP-Completeness.

3.3.2 Acyclic Solution with open nodes only

Introduction

We consider the simplest case, where all nodes are open and we search for an acyclic solution. As we have just proved it, despite its apparent simplicity, this problem is NP-Complete in the strong sense. Nevertheless, it is possible to achieve the optimal throughput at the price of a very small additive increase of the degree of the nodes. To establish this result, we recall the proof of an upper bound on the achievable throughput of any acyclic solution. Then, we recall exhibit an algorithm that achieves this throughput while keeping the degree of the nodes small. This algorithm is used as a starting point to build a cyclic solution for instances without guarded nodes in Section 3.5, and it introduces some of the ideas that will be later adapted in Section 3.4 for instances with guarded nodes.

Upper Bound

The first idea which will be used throughout the chapter is that nodes should be ordered by non-increasing order of bandwidth. In the remainder, we will thus consider that nodes are ordered so that $b_1 \geq \ldots \geq b_n$ and we will denote $S_k = \sum_{i=0}^{k} b_i$. In any acyclic solution, nodes can be sorted in topological order such that a node only feeds nodes with larger indexes. In particular, there exists at least one node that does not send data to any other node. Therefore, the overall throughput achieved by any acyclic solution T^* is upper bounded by $\frac{S_{n-1}}{n}$ since b_n denotes the smallest capacity and n nodes $C_1, \ldots, C_n$ must receive the message at a rate T^*. Additionally, it is clear that $T^* \leq b_0$.

Algorithm

Let us now describe an algorithm that provides an optimal acyclic solution for instances without guarded nodes.

The algorithm takes as input $T^* = \min(b_0, \frac{S_{n-1}}{n})$, and returns a broadcast scheme that achieves throughput T^*. Let us first remark that since the b_is are sorted in non-increasing order, and since $T^* \leq b_0$, then $\forall 0 \leq k < n, S_k \geq (k+1)T^*$.

The basic principle of the algorithm formalized in Algorithm 1 is to satisfy (*i.e.* send a complete message to) the nodes one after the other (considered in the previously defined sorting order), while maintaining the property that after each step, at most one node receives the message only partially, *i.e.* all previous nodes receive the message at rate T^* and all following ones do not receive anything yet. C_i thus sends data to a consecutive set of nodes, say from C_{α_i} to C_{β_i}. All intermediate nodes, except possibly α_i and β_i, will be served at rate T^*. Since the total bandwidth used by C_i is b_i, there are at most $\left\lceil \frac{b_i}{T^*} \right\rceil - 1$ such intermediate nodes. Hence the number of nodes served at least partially by C_i, *i.e.* its outdegree, is at most $\left\lceil \frac{b_i}{T^*} \right\rceil + 1$.

The behavior of Algorithm 1 is depicted in Figure 16.

Furthermore, this algorithm produces an acyclic graph. Indeed, before each step i, the property $S_{i-1} \geq iT^*$ ensures that the bandwidth available so far (S_{i-1}) is always large enough to satisfy all

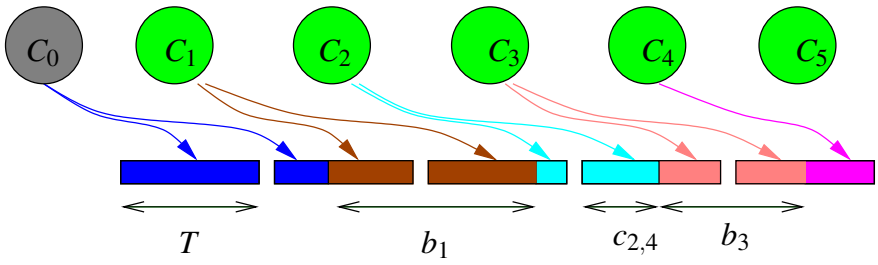

Figure 16: Solution returned by Algorithm 1. The upper part represents how the capacity of C_i is used (in column i) and the lower part describes which nodes the data sent to C_i (in column i) come from.

nodes from 1 to i. Hence, each C_i will only serve nodes with strictly larger indexes (*i.e.* $\alpha_i > i$ with above notations).

Algorithm 1 Acyclic Algorithm on open nodes only.

Set $t = 1$ and $\forall i, r_i = T^*$ and $\forall i, s_i = b_i$
for $i = 0$ to n **do**
  while $s_i > 0$ **do**
    $c_{i,t} := \min(r_t, s_i)$
    $s_i := s_i - c_{i,t}$; $r_t := r_t - c_{i,t}$
    if $r_t = 0$ **then**
      $t := t + 1$
    end if
  end while
end for

From above remarks, we can conclude that Algorithm 1 builds an acyclic communication graph such that each node receives exactly T^* from nodes with smaller indexes and that the degree of C_i is at most $\left\lceil \frac{b_i}{T^*} \right\rceil + 1$. Therefore, Algorithm 1 returns a solution of optimal throughput in which nodes have a degree that is larger by at most an additive factor of 1 with respect to the lower bound. Clearly, given the strong NP-Completeness of the problem, Algorithm 1 returns the best achievable result, with respect to the additive degree increase (provided, of course, that $P \neq NP$).

3.4 Acyclic algorithm with guarded nodes

In this section, we describe how to build an acyclic broadcast scheme with a small increase in the degree constraint in presence of guarded nodes:

Theorem 3.4.1. *Given an instance I and a throughput T, it is possible to decide in linear time if $T \leq T^*_{ac}$. Moreover if $T \leq T^*_{ac}$, it is possible to compute in linear time a broadcast scheme of throughput T such that*

- *for every guarded node $j \in \mathscr{G}$, outdegree o_j is bounded:* $o_j \leq \left\lceil \frac{b_j}{T} \right\rceil + 1$;
- *for at most one open node i,* $o_i \leq \left\lceil \frac{b_i}{T} \right\rceil + 3$;
- *for all other open nodes,* $o_i \leq \left\lceil \frac{b_i}{T} \right\rceil + 2$.

For instances with guarded nodes, there is no closed formula for T^*_{ac}, but the algorithm of Theorem 3.4.1 can be combined with a dichotomic search (on T) to find the optimal acyclic throughput.

The proof of Theorem 3.4.1 can be decomposed into three steps: we start by proving dominance relations in order to characterize optimal acyclic schemes (Lemma 3.4.3). We then provide an algorithm for testing if throughput T is achievable. If this is the case, a valid ordering is computed (Lemma 3.4.4). Then, we show how to compute a low degree solution from the computed valid ordering (Lemma 3.4.8).

3.4.1 Dominance relations

Before entering into the details of the algorithm, let us start with an intuitive property of ordering of nodes. An ordering σ is said to be *increasing* if its restriction to $\mathscr{O}$ is the identity on $\mathscr{O}$, and its restriction to $\mathscr{G}$ is the identity on $\mathscr{G}$. This means that nodes of the same color are ordered by non-increasing order on their bandwidth. The order $\sigma = 031245$ is an increasing order for the instance of Figure 14 whereas $\sigma = 041235$ is not increasing.

In Section 3.3, we have proved (in the open nodes only case) that good solutions can be built with increasing orderings, and the next lemma shows that this also holds in the general case.

Lemma 3.4.2.

$$T^*_{ac} = \max_{\sigma:increasing} \{T^*_{ac}(\sigma)\}.$$

Proof. Let c be an acyclic solution with order σ which is not increasing. Then, there exist two indices $x < y$ such that $p = \sigma(x) > q = \sigma(y)$ (and thus $b_p \leq b_q$). We will exhibit another acyclic solution c' with order $\sigma' = \sigma \circ (x,y)$ (where (x,y) denotes the transposition that exchanges x and y, which means that the nodes in position x and y are swapped) and whose throughput is not smaller than c.

The transformation is depicted on Figure 17. For most indices i, j, it is sufficient to set $c'_{\sigma'(i),\sigma'(j)} = c_{\sigma(i),\sigma(j)}$. However, this would break the bandwidth constraint of node $p = \sigma(x)$, and the solution is to give the connections in excess (denoted as E in Figure 17) to node $q = \sigma'(x)$. Since $x < y$, this does not break acyclicity.

Recursively, we can thus transform any acyclic solution into an increasing acyclic solution with at least the same throughput. □

An increasing order can be naturally encoded by a binary word π with n letters ◯ (corresponding to open nodes) and m letters □ (corresponding to guarded nodes): it is sufficient to specify if $\sigma(i)$ belongs to $\mathscr{O}$ or to $\mathscr{G}$. We denote by $|\pi|$ the length of the word π, and by $|\pi|_◯$ (resp. $|\pi|_□$) the

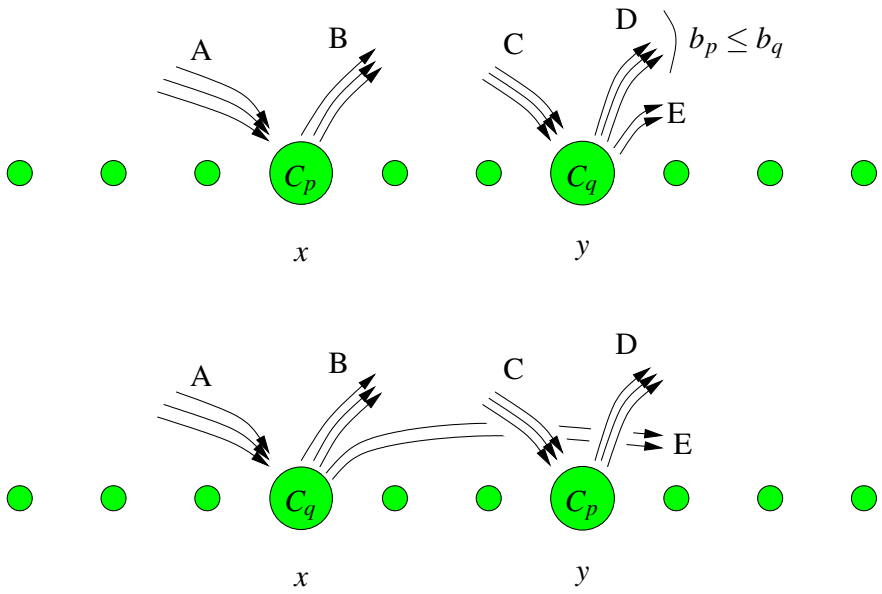

Figure 17: Exchange argument for dominance of increasing solutions

number of letters ◯ (resp. □) in π. For instance, the word $\pi = \square\bigcirc\bigcirc\square\square$ encodes the increasing order $\sigma = 031245$ for the instance of Figure 14.

The notation $\pi' \sqsubseteq \pi$ (resp. $\pi' \sqsubset \pi$) means that π' is a prefix (resp. a strict prefix) of π.

From now on, when no confusion is possible, π will be identified with its corresponding increasing order. For instance, $T^*_{ac}(\pi)$ corresponds to the optimal acyclic throughput associated with the order encoded by π. A word π is said to be *valid* (with respect to an instance I and a throughput T) if $T^*_{ac}(\pi) \geq T$.

A solution c is said to be *conservative* with respect to order σ, if there are no triplets of distinct indices i, j, k, such that $i < k$ and $j < k$, $\sigma(i) \in \mathscr{G}$, $\sigma(j), \sigma(k) \in \mathscr{O}$, and $c_{\sigma(j),\sigma(k)} > 0$ and $\sum_{l=i+1}^{k} c_{\sigma(i),\sigma(l)} < b_{\sigma(i)}$ simultaneously. The idea behind this definition is to consider solutions that feed the open nodes from guarded nodes as soon as possible. Indeed, the firewall constraint prevents transfer from guarded nodes to guarded nodes: transfer from open nodes is thus a valuable resource, and it is a "waste" to use it to feed open nodes when it is not necessary. Figure 14 shows an example of a conservative acyclic broadcast scheme and Figure 18 shows an example of a non-conservative one.

This means that when creating a conservative solution incrementally (by satisfying the nodes in a given order σ), there is no choice for the type of nodes that should feed the next node to add: a guarded node must be fed by open nodes (because of the firewall constraint), and an open node should be fed by guarded nodes as long as some of them have remaining outgoing capacity.

Lemma 3.4.3. *For every order σ there exists a* conservative *solution c that achieves $T^*_{ac}(\sigma)$.*

Proof. Let c be a solution that achieves $T^*_{ac}(\sigma)$. If there exists a triplet of indices i, j, k that violates *conservativeness*, we can build a solution c' that is conservative with respect to these indices. Let

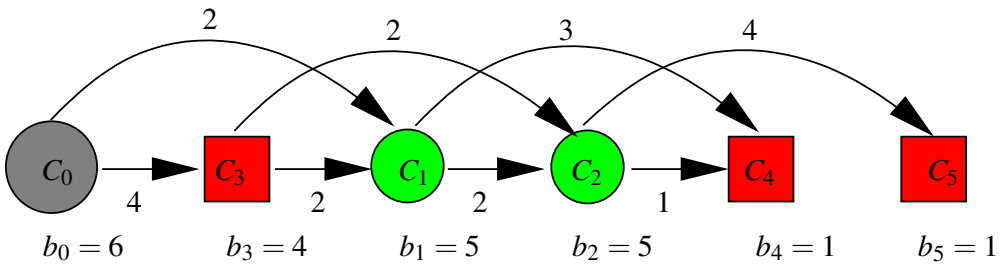

Figure 18: A non-conservative acyclic broadcast scheme: if we take $i = 1, j = 0, k = 2$, we see that node $C_1 = C_{\sigma(k)}$ could be totally fed by a guarded node $C_3 = C_{\sigma(i)}$, but it uses the open bandwidth of the source $C_0 = C_{\sigma(j)}$.

$\gamma = \min(b_{\sigma(i)} - \sum_{l=i+1}^{k} c_{\sigma(i),\sigma(l)}, c_{\sigma(j),\sigma(k)})$, and set:

$$c'_{\sigma(j),\sigma(k)} = c_{\sigma(j),\sigma(k)} - \gamma$$
$$c'_{\sigma(i),\sigma(k)} = c_{\sigma(i),\sigma(k)} + \gamma.$$

and as in the proof of Lemma 3.4.2, $\sigma(j)$ will be in charge in c' of the upload toward nodes $\sigma(l)$ with $l > k$ that the node $\sigma(i)$ will no longer be able to feed in c'; on all other indices c and c' coincide. It is easy to see that c' is a valid solution of the same throughput, and that the number of triplets of indices violating *conservativeness* is lower in c'. Recursively, we create a *conservative* acyclic solution with respect to order σ, with throughput $T^*_{ac}(\sigma)$. □

Given a throughput T, and a coding word π with $0 \leq i \leq n$ letters ◯ and $0 \leq j \leq m$ letters □, let $\mathscr{C}_\pi$ be the set of partial conservative solutions on the partial increasing order encoded by π (that feeds nodes $C_1, \ldots, C_i$ and $C_{n+1}, \ldots, C_{n+j}$).

All partial conservative solutions of $\mathscr{C}_\pi$ have the same amount of available throughput of each type. Let us denote by $O(\pi)$ (respectively $G(\pi)$) the open (respectively guarded) bandwidth available at the end of the partial solutions of $\mathscr{C}_\pi$. O and G satisfy the following recursive equations:

$$\begin{aligned}
O(\varepsilon) &= b_0, \\
G(\varepsilon) &= 0, \\
O(\pi\square) &= O(\pi) - T, \\
G(\pi\square) &= G(\pi) + b_{n+j+1}, \\
O(\pi\bigcirc) &= O(\pi) + b_{i+1} - \max(0, T - G(\pi)), \\
G(\pi\bigcirc) &= \max(0, G(\pi) - T).
\end{aligned}$$

The values O and G encompass all the capacity constraints of solutions in $\mathscr{C}_\pi$. Indeed, it is easy to see that a coding word π is valid for a throughput T if and only if

- for any prefix $\pi'\square$ of $\pi, O(\pi') \geq T$, and
- for any prefix $\pi'\bigcirc$ of $\pi, O(\pi') + G(\pi') \geq T$.

π	ε	$\square$	$\square\bigcirc$	$\square\bigcirc\square$	$\square\bigcirc\square\bigcirc$	$\square\bigcirc\square\bigcirc\square$
$O(\pi)$	6	2	7	3	5	1
$G(\pi)$	0	4	0	1	0	1
$W(\pi)$	0	0	0	0	3	3

Table 2: Execution of Algorithm 2 on the instance of Figure 13. Observe that the amount of open-open transfer ($W(\pi)$) is only 3 whereas in the acyclic scheme proposed in Figure 14 this amount is 4.

Another parameter that is common to each partial conservative solution of $\mathscr{C}_\pi$ is $W(\pi)$, the amount of transfer going from open nodes to other open nodes. This parameter satisfies the following recursive equations

$$\begin{aligned} W(\varepsilon) &= 0, \\ W(\pi\square) &= W(\pi), \\ W(\pi\bigcirc) &= W(\pi) + \max(0, T - G(\pi)). \end{aligned}$$

From above, we obtain

$$G(\pi) = b_{n+1} + \ldots + b_{n+j} - i \cdot T + W(\pi) \quad (1)$$

$$O(\pi) = b_0 + b_1 + \ldots + b_i - j \cdot T - W(\pi) \quad (2)$$

and $O(\pi) + G(\pi) = \sum_{k=0}^{|\pi|_\bigcirc} b_k + \sum_{k=n+1}^{n+|\pi|_\square} b_k - |\pi|T$.

3.4.2 Greedy algorithm

In this section, we present Algorithm 2 that decides whether a given throughput T is feasible. If T is feasible, Algorithm 2 also outputs a valid coding word. It works by iteratively building a partial conservative solution π, deciding at each step how to extend the partial solution (by $\bigcirc$ or by $\square$). This decision is made greedily, by choosing $\square$ if it is possible. The algorithm is forced to take $\bigcirc$ (see line 12):

- when it is not possible to choose $\square$ at the current step ($O(\pi) < T$);
- or when choosing $\square$ would make it impossible to continue afterwards ($O(\pi\square) + G(\pi\square) < T$).

Of course, if all guarded nodes have been used (line 6), the algorithm chooses $\bigcirc$. Another special case is when only one guarded node is left. In that case (see lines 8-11), the algorithm chooses at each step the node with the largest b_i (unless it is guarded and $O(\pi) < T$).

Table 3.4.2 shows an execution of Algorithm 2 on the instance of Figure 13. The generated scheme is shown in Figure 19.

The following lemma states that this algorithm is valid.

Algorithm 3 GreedyTest(T), Acyclic Algorithm on open/guarded nodes.

$\pi \leftarrow \varepsilon$
while $|\pi| < n+m$ **do**
  if $O(\pi)+G(\pi) < T$ **then return** FAIL
  $i \leftarrow |\pi|_{\bigcirc}; j \leftarrow |\pi|_{\square}; l \leftarrow \square$
  if $i \neq n$ **then**
    if $j = m$ **then**
      $l \leftarrow \bigcirc$
    else if $j = m-1$ **then**
      if $O(\pi) < T$ or $b_{n+j+1} < b_{i+1}$ **then**
        $l \leftarrow \bigcirc$
      end if
    else if $O(\pi) < T$ or $O(\pi\square)+G(\pi\square) < T$ **then**
      $l \leftarrow \bigcirc$
    end if
  end if
  $\pi \leftarrow \pi l$
  if $O(\pi) < 0$ **then return** FAIL
end while
return π

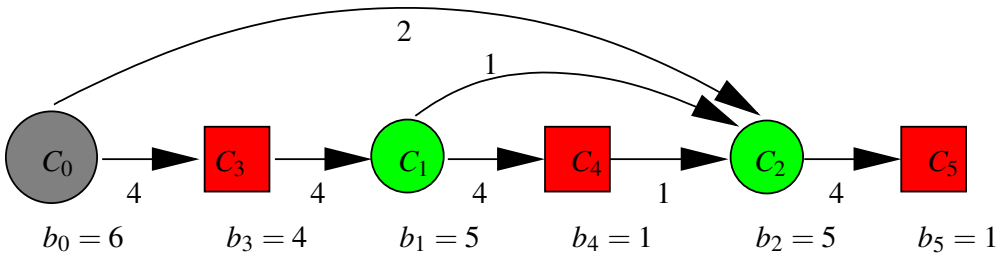

Figure 19: The acyclic broadcast scheme of throughput 4 built by Algorithm 2. The order associated with this scheme is $\sigma = 031425$.

Lemma 3.4.4. *Given an instance I and a throughput T, Algorithm 2 returns a valid word (a word π such that $T^*_{ac}(\pi) \geq T$) if and only if T is feasible for this instance ($T^*_{ac} \geq T$).*

In order to prove Lemma 3.4.4, we now need to state two preliminary lemmas. The first one shows that this algorithm uses open nodes as late as possible, and is as conservative as possible.

Lemma 3.4.5. *Let π_k be the value of π in Algorithm 2 when the k-th open node has just been added. ($|\pi_k|_\bigcirc = k$, and π_k ends with a $\bigcirc$).*

If $|\pi_k|_\square < m-1$, then for every π'_k ending with a $\bigcirc$ such that $|\pi'_k|_\bigcirc = k$, we have

$$W(\pi'_k) \geq W(\pi_k) \quad \textit{and} \quad |\pi'_k|_\square \leq |\pi_k|_\square.$$

Proof. We prove this lemma by induction on k. Clearly the lemma holds true for $k = 0$, since $\pi_0 = \varepsilon = \pi'_0$. (We treat source node as 0-th $\bigcirc$ node.)

Assume now that lemma holds true for $k-1$, and let us decompose the words maximally as follows

$$\pi_k = \pi_{k-1}\square^a\bigcirc \qquad \text{and note} \quad \delta = \pi_{k-1}\square^a,$$
$$\pi'_k = \pi'_{k-1}\square^{a'}\bigcirc.$$

Let $l = |\pi_k|_\square$ and $l' = |\pi'_k|_\square$. From (1) and (2), we get

$$O(\delta) = b_1 + \ldots + b_{k-1} - l \cdot T - W(\pi_{k-1}),$$
$$G(\delta) = b_{n+1} + \ldots + b_{n+l} - (k-1) \cdot T + W(\pi_{k-1}).$$

Since Algorithm 2 chooses $\bigcirc$ (after choosing δ), and $|\delta|_\square < m-1$, we have $O(\delta) < T$ or $O(\delta) + G(\delta) + b_{n+l+1} < 2T$.

Let us first prove by contradiction that $|\pi'_k|_\square \leq |\pi_k|_\square$. Assume that $|\pi'_k|_\square > |\pi_k|_\square$. In this case, there exists $\delta' \sqsubseteq \pi'_k$ such that $|\delta'| = |\delta|$. By induction assumption, $|\pi_{k-1}|_\square \geq |\pi'_{k-1}|_\square$, which implies that $|\pi'_{k-1}| \leq |\pi_{k-1}| \leq |\delta|$. Hence, $|\delta'|_\bigcirc = |\pi'|_\bigcirc - 1 = k-1$. We can thus compute

$$\begin{aligned} O(\delta') &= b_1 + \ldots + b_{k-1} - l \cdot T - W(\pi'_{k-1}) \\ &\leq b_1 + \ldots + b_{k-1} - l \cdot T - W(\pi_{k-1}) = O(\delta), \\ O(\delta') + G(\delta') &= \sum_{i=1}^{k-1} b_i - l \cdot T + \sum_{i=n+1}^{n+l} b_i - (k-1) \cdot T \\ &= O(\delta) + G(\delta). \end{aligned}$$

So, either $O(\delta') < T$ or $O(\delta') + G(\delta') + b_{n+l+1} < 2T$. Both lead to a contradiction when we try to continue δ' with $\square$. This proves that $|\pi'_k|_\square \leq |\pi_k|_\square$.

Let us now prove that $W(\pi'_k) \geq W(\pi_k)$. As π_k and π'_k end with $\bigcirc$,

$$\begin{aligned} W(\pi_k) &= W(\pi_{k-1}) + \max(0, T - G(\delta)) \\ &= \max(W(\pi_{k-1}), T \cdot k - (b_{n+1} + \ldots + b_{n+l})), \\ W(\pi'_k) &= \max(W(\pi'_{k-1}), T \cdot k - (b_{n+1} + \ldots + b_{n+l'})). \end{aligned}$$

Since $l' \leq l$ and $W(\pi'_{k-1}) \geq W(\pi_{k-1})$ (the inductive assumption), we have $W(\pi'_k) \geq W(\pi_k)$. $\square$

Lemma 3.4.6. *Let π_1, π_2 be two conservative partial solutions such that $|\pi_1|_{\bigcirc} = |\pi_2|_{\bigcirc}$ and $|\pi_1|_{\square} = |\pi_2|_{\square}$. If $W(\pi_1) \leq W(\pi_2)$, then $\forall \omega \in \{\bigcirc, \square\}^*, W(\pi_1\omega) \leq W(\pi_2\omega)$.*

Proof. To prove the lemma, we only have to consider the cases where $\omega \in \{\bigcirc, \square\}$. The case $\omega = \square$ is trivial since $W(\pi\square) = W(\pi)$.

Let us consider now the case $\omega = \bigcirc$.

$$\begin{aligned} W(\pi_1\bigcirc) &= \max(W(\pi_1), W(\pi_1) + T - G(\pi_1)) \\ &= \max(W(\pi_1), T + i.T - b_{n+1} - \ldots - b_{n+j}) \\ &\leq \max(W(\pi_2), T + i.T - b_{n+1} - \ldots - b_{n+j}) \\ &\leq W(\pi_2\bigcirc). \end{aligned}$$

□

Proof. of Lemma 3.4.4. The first implication is trivial, since the tests performed at each step of Algorithm 2 ensure that the returned word is always valid.

For the reverse implication, we prove that if Algorithm 2 fails to find a solution, then there does not exist a valid ordering of the nodes with respect to throughput T. According to Lemmas 3.4.2 and 3.4.3, we only consider encoding words.

Let ω be the partial solution built by Algorithm 2 (before it failed), and let $i = |\omega|_{\bigcirc}$ and $j = |\omega|_{\square}$.

There are four different cases to consider:

- $j < m-1$ and ω ends with $\bigcirc$.

  Since Algorithm 2 failed after ω, $O(\omega) + G(\omega) < T$. On the other hand, $O(\omega) \geq b_i$, what implies $b_i < T$ and $\forall k \geq i$, $b_k < T$.

  Let π be any encoding word, and let us consider the largest sub-word $\pi' \sqsubseteq \pi$ such that $|\pi'|_{\square} = |\omega|_{\square}$. If $|\pi'|_{\bigcirc} < |\omega|_{\bigcirc}$, then there exists a word $\rho \sqsubseteq \pi$ such that $|\rho|_{\bigcirc} = |\omega|_{\bigcirc}$ and $|\rho|_{\square} > |\omega|_{\square}$. Since this violates the conclusions of Lemma 3.4.5, π is not valid.

  If $|\pi'|_{\bigcirc} \geq |\omega|_{\bigcirc}$, then

  $$\begin{aligned} O(\pi') + G(\pi') &= O(\omega) + G(\omega) + \sum_{k=i+1}^{|\pi'|_{\bigcirc}} (b_k - T) \\ &\leq O(\omega) + G(\omega) < T. \end{aligned}$$

  In conclusion, $O(\pi') < T$ and thus π is not valid.

- $j \leq m-1$ and ω ends with $\square$.

  Because of the test at line 12, this implies that the last $\square$ was added by the instruction on line 4, and thus $|\omega|_{\bigcirc} = n$.

Let π be an encoding word. We can decompose ω and π as $\omega' \bigcirc \square^a$ and $\pi' \bigcirc \square^b$, and we can apply Lemma 3.4.5 to words $\omega'\bigcirc$ and $\pi'\bigcirc$

$$W(\omega) = W(\omega') \leq W(\pi') = W(\pi)$$

Since $|\omega|_\bigcirc = n$ and since Algorithm 2 failed, then either $O(\omega) + G(\omega) < T$ or $O(\omega\square) < 0$. In both cases, $O(\omega) < T$, and since $O(\omega) = O - jT - W(\omega)$, we get $O < mT + W(\pi)$, and thus $O(\pi) < 0$. Hence π is not valid.

- $j = m$. The main argument is that the bandwidth of the remaining open nodes is lower than that of the last guarded node. Since Algorithm 2 chose the last guarded node at some point (line 11), we have $b_{i+1} \leq b_{n+m}$.

  The failure of the algorithm implies $O(\omega) + G(\omega) < T$. Let $\omega = \omega'\alpha$. We know that $O(\omega') + G(\omega') \geq T$, and also:

$$\begin{aligned} O(\omega) + G(\omega) &= O(\omega') + G(\omega') - T + b_i && \text{if } \alpha = \bigcirc, \\ O(\omega) + G(\omega) &= O(\omega') + G(\omega') - T + b_{n+m} && \text{if } \alpha = \square. \end{aligned}$$

  So either $b_{n+m} < T$ or $b_i < T$. In both cases, we have

$$b_n \leq b_{n-1} \leq \ldots \leq b_{i+1} < T.$$

  Let $\pi = \pi'\beta$ be any encoding word. If $\beta = \bigcirc$, then

$$\begin{aligned} O(\pi') + G(\pi') &= b_0 + O - b_n - (n-1)T + G - mT \\ &= O(\omega) + G(\omega) + \sum_{k=i+1}^{n-1} (b_k - T) \\ &\leq O(\omega) + G(\omega) < T \end{aligned}$$

  Hence π is not valid. Otherwise, $\beta = \square$, and $O(\pi') + G(\pi') = b_0 + O - nT + G - b_{n+m} - (m-1)T$. Since $b_{n+m} \geq b_n$, we get the same conclusion.

- $j = m - 1$ and ω ends with $\bigcirc$. The main argument is that the last guarded node can be delayed: minimizing waste is not so important since only open nodes remain to be fed. Just like in the first case, we have $\forall k \geq i, b_k < T$. Let us decompose ω as $\omega = \omega'\bigcirc^a$ ($a \geq 0$).

  We begin by showing that words $\pi(x) = \omega'\square\bigcirc^x\square\bigcirc^{a-x}$ are invalid for throughput T. The following lemma shows that it is possible to consider only words where the last $\square$ is followed only by $\bigcirc$ with smaller bandwidth.

  Lemma 3.4.7. *If word $\pi = \pi_1\square\bigcirc\bigcirc^a$ is a valid word in which the last $\square$ has bandwidth g, the following $\bigcirc$ has bandwidth o, and $o \geq g$, then the word $\pi' = \pi_1\bigcirc\square\bigcirc^a$ is also valid.*

Proof. Let $G = O(\pi_1), R = G(\pi_1)$ and $\pi_2 = \bigcirc^a$. Since π is valid, we have $O \geq T$ and $O - T + G + r \geq T$. We can thus bound $O(\pi_1 \bigcirc)$

$$\begin{aligned} O(\pi_1 \bigcirc) &= O + o - \max(T - G, 0) \\ &= \min(O + o + G - T, O + o) \geq T. \end{aligned}$$

This ensures that $\pi_1 \bigcirc \square$ is a valid sequence.

Since π_2 is composed only of $\bigcirc$, and $O(\pi_1 \square \bigcirc) + G(\pi_1 \square \bigcirc) = O(\pi_1 \bigcirc \square) + G(\pi_1 \bigcirc \square)$, $\pi_1 \bigcirc \square \pi_2$ is a valid sequence. □

So if $\pi(x)$ is valid, we can iteratively use Lemma 3.4.7 to prove the existence of a valid $\pi(y)$ in which the last $\square$ is followed by a $\bigcirc$ with smaller upload. If $y < a$, since Algorithm 2 at that point chose $\bigcirc$ instead of $\square$, we know that $O(\omega' \square \bigcirc^y) < T$ and $\pi(y)$ is invalid. If $y = a$, then $\pi(y) = \omega \square$, which is invalid because Algorithm 2 failed.

Consider now any encoding word π. Let $\pi = \pi_1 \pi_2 \square \bigcirc^k$ be the decomposition with minimal π_1 having $|\pi_1|_\bigcirc = |\omega'|_\bigcirc$ (applying Lemma 3.4.5 we have $|\pi_1|_\square \leq |\omega'|_\square = m - 2$, so decomposing is always possible).

For any word δ we have

$$W(\delta \bigcirc \square) = W(\delta \bigcirc) = W(\delta) + \max(0, T - G(\delta)) \geq$$

$$\geq W(\delta \square) + \max(0, T - G(\delta \square)) = W(\delta \square \bigcirc).$$

We can apply it to word π

$$W(\pi_1 \pi_2) \geq W(\pi_1 \square^{|\pi_2|_\square} \bigcirc^{|\pi_2|_\bigcirc}).$$

Furthermore, since Lemma 3.4.5 applies to π_1 and ω'

$$W(\pi_1) \geq W(\omega').$$

so by Lemma 3.4.6, (since $|\pi_1|_\bigcirc = |\omega'|_\bigcirc$, $|\pi_1 \pi_2|_\square = m - 1 = |\omega' \square|_\square$)

$$W(\pi_1 \square^{|\pi_2|_\square} \bigcirc^{|\pi_2|_\bigcirc}) \geq W(\omega' \square \bigcirc^{|\pi_2|_\bigcirc}).$$

Composing it, we have

$$W(\pi_1 \pi_2) \geq W(\omega' \square \bigcirc^{|\pi_2|_\bigcirc}) \text{and}$$

$$\forall x, W(\pi_1 \pi_2 \square \bigcirc^x) \geq W(\omega' \square \bigcirc^{|\pi_2|_\bigcirc} \square \bigcirc^x).$$

So if $\pi = \pi_1 \pi_2 \square \bigcirc^k$ is valid, then $\omega' \square \bigcirc^{|\pi_2|_\bigcirc} \square \bigcirc^k$ is also valid. But we proved previously that no such solution can exist. Thus, we reached a contradiction.

□

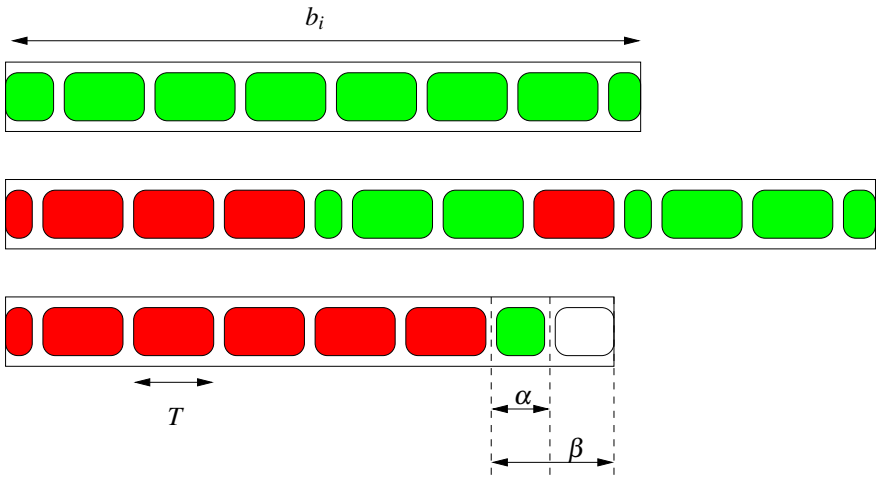

Figure 20: 3 examples of upload node repartition. A guarded node feeds at most 2 nodes partially (first example). An open node that is the first to feed the last guarded node (second example). General case for an open node (last example).

3.4.3 Low degree scheme for a word π

The output of Algorithm 2 is an encoding word and an ordering, together with the amounts of guarded or open bandwidths used for this purpose, but not the actual values of the $c_{i,j}$s. There are several possibilities for the $c_{i,j}$s. However, in order to prove bounds on the degree of the nodes, we will feed each node by the earliest possible nodes with unused upload bandwidth (as in the open nodes case described in Section 3.3.2).

Lemma 3.4.8. *From the word π given by Algorithm 2, it is possible to build a broadcast scheme such that*

- *for every guarded node $j \in \mathscr{G}$, outdegree o_j is bounded: $o_j \leq \left\lceil \frac{b_j}{T} \right\rceil + 1$;*
- *for at most one open node i, $o_i \leq \left\lceil \frac{b_i}{T} \right\rceil + 3$;*
- *for the other open nodes, $o_i \leq \left\lceil \frac{b_i}{T} \right\rceil + 2$.*

Proof. Since guarded nodes can only upload to open nodes, and open nodes always receive from the earliest guarded node available, every guarded node uploads to a consecutive interval of open nodes. So at most 2 nodes will be partially fed by a specific guarded node: the first and the last one of the interval (see first example of Figure 20).

Let us now consider an open node i. Because Algorithm 2 rather chooses guarded nodes when it is possible, as long as there is enough open bandwidth available, node i will feed a consecutive interval of guarded nodes. When the amount of open upload available gets low, there are two cases to consider:

- i is the earliest open node that feeds the last guarded node.

  The sequence of nodes fed by i first consists in a sequence of guarded nodes, then a sequence of open nodes, then the last guarded node and another sequence of open nodes (see second example of Figure 20). Since conservatism implies that $G(\pi\bigcirc) = 0$ after feeding an open node from node i, a partially fed open node can only take place as the first node after guarded nodes. Hence, the only nodes partially fed by i are the first one, the last one and the opening nodes of the 2 open sequences. In total, at most 4 nodes are partially fed by node i.

- Otherwise (see last example of Figure 20), Algorithm 2 feeds guarded nodes with the upload of i as long as there is enough bandwidth. At some point, $O+G+g_{\text{next}} < 2T$, where g_{next} is the bandwidth of the next guarded node to be fed. Let β be the remaining bandwidth of node i at that point. By the definition of O, $\beta \leq O$. At this moment, Algorithm 2 decides to switch to open nodes. Open nodes are fed using guarded bandwidth at first. If any open node is fed using $\alpha = T - G$ upload from i, the remaining upload of i is equal to $\beta - \alpha \leq O+G-T \leq T - g_{\text{next}} \leq T$. Thus, the next node fed by i uses all the remaining bandwidth of node i. Hence, node i feeds partially at most 3 nodes: the first node, one open node and the last node.

This concludes the proof of Theorem 3.4.1. □

3.5 Cyclic case

This section considers cyclic broadcast schemes. We start by giving an upper bound on the optimal cyclic throughput T^*:

Lemma 3.5.1.

$$T^* \leq \min\left(b_0, \frac{b_0+O}{m}, \frac{b_0+O+G}{n+m}\right),$$

where $O = \sum_{i=1}^{n} b_i$ *and* $G = \sum_{i=n+1}^{n+m} b_i$.

For the instance of Figure 13, $O = 10$, $G = 6$. Hence, from this lemma, we know that the throughput of the broadcast scheme of Figure 13 is optimal since $\min(6, 16/3, 22/5) = 4.4$.

Proof. Clearly $T^* \leq b_0$, since the whole message has to be sent at least once by the source. Then, the m guarded nodes have to receive the message at rate T^* and therefore consume mT^* bandwidth. Since this bandwidth must come from the source and the open nodes, then $mT^* \leq b_0 + O$. Finally, all $n+m$ nodes must receive the whole message at rate T^* and the bandwidth must come from the source, the open and the guarded nodes, so that $(m+n)T^* \leq b_0 + O + G$. □

As shown in Figure 21, it is not always possible to achieve a solution of optimal throughput with low degree in presence of guarded nodes. However, after [7]:

Theorem 3.5.2 ([7])**.** *There exists an algorithm which takes as input any instance without guarded nodes and a target value of* $T \leq T^* = \min\left(b_0, \frac{b_0+O}{n}\right)$*, and which builds a cyclic solution of throughput* T*, in which any node has outdegree* $o_i \leq \max\left(\left\lceil \frac{b_i}{T} \right\rceil + 2, 4\right)$.

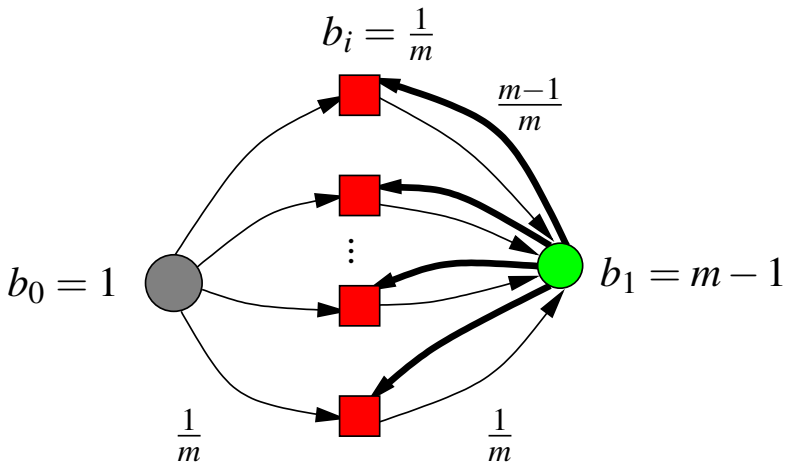

Figure 21: An instance with guarded and open nodes where the optimal cyclic throughput is $T^* = \min\left(b_0, \frac{b_0+b_1}{m}, \frac{b_0+b_1+mb_2}{m+1}\right) = 1$. In the optimal solution, the source has degree m, whereas $\left\lceil \frac{b_0}{T^*} \right\rceil = 1$.

Proof is a constructive one. It relies on incremental building of an acyclic solution (in a style resembling the Algorithm 1), and performing necessary local fixes when unable to perform another acyclic step.

3.6 Cyclic/Acyclic throughput comparison

In this section, we compare the optimal acyclic throughput with the optimal (cyclic) throughput. On the one hand we show that the ratio $\frac{T^*_{ac}}{T^*}$ can be as small as $\frac{5}{7} \approx 0.71$ for (small-size) instances and as small as $\frac{1+\sqrt{41}}{8} \approx 0.925$ for arbitrary large instances (by contrast, when there are only open nodes, this ratio tends to one when the number of nodes is large). On the other hand, we show that this ratio is larger than $\frac{5}{7}$ for any instance, so that this bound is tight. Finally we present experimental results on the ratio $\frac{T^*_{ac}}{T^*}$ on random instances, that prove that acyclic solutions achieve much better results than the $\frac{5}{7}$ bound in practice.

3.6.1 Worst cases

Without guarded nodes

Theorem 3.6.1. *For any instance I of size n and without guarded nodes,*

$$\frac{T^*_{ac}}{T^*} \geq 1 - \frac{1}{n}.$$

Proof. Let I be an instance of size n without guarded nodes. From Section 3.3 we know that

$$T^*_{ac} = \min\left(b_0, \frac{b_0 + O - b_n}{n}\right).$$

From Lemma 3.5.1 we have

$$T^* \leq \min\left(b_0, \frac{b_0 + O}{n}\right).$$

If $T^*_{ac} = b_0$ then it is also the case for T^* and the result holds. Else we have

$$\frac{T^*_{ac}}{T^*} \geq \frac{b_0 + O - b_n}{b_0 + O} \geq 1 - \frac{b_n}{b_0 + O}.$$

Because of the ordering of nodes we have $O \geq nb_n$. This concludes the proof. □

With guarded nodes

We start this section by characterizing a special class of instances which are the worst possible cases for the acyclic throughput. An instance is said to be *homogeneous* if all open nodes except the source have the same throughput o and all guarded nodes have the same throughput g. An instance is said to be *tight* if $b_0 = \frac{b_0+O+G}{n+m} = T^*$ (*i.e.* if no bandwidth can be wasted in the optimal cyclic solution). The instance of Figure 13 is tight but not homogeneous and the instance of Figure 21 is tight and homogeneous.

Lemma 3.6.2. *Let $\alpha > 0$. If for every tight homogeneous instance, $\frac{T^*_{ac}}{T^*} \geq \alpha$, then for every instance $\frac{T^*_{ac}}{T^*} \geq \alpha$.*

Proof. To prove this lemma we will show that given an instance, we can associate with it a tight homogeneous instance with the same optimal throughput T^* and with no greater optimal acyclic throughput T^*_{ac}.

First, if the instance is such that

$$\frac{b_0 + O + G}{n + m} > T^*,$$

by reducing the throughput of the guarded nodes it is possible to make this inequality an equality. This transformation does not change the optimal throughput T^* and any acyclic solution for the transformed instance is also an acyclic solution for the original one.

Consider now a non-homogeneous instance I such that

$$\frac{b_0 + O + G}{n + m} = T^*.$$

Let I' be the homogeneous instance obtained from I as follows:

$$b'_0 = T^*, b'_i = o = \frac{N + b_0 - T^*}{n} \quad \text{for } i \in [\![1, n]\!],$$

$$b'_i = g = \frac{M}{m} \quad \text{for } i \in [\![n+1, n+m]\!],$$

where b'_i is the throughput of the node C_i in I'. Clearly I and I' have the same optimal throughput T^* and I' is tight and homogeneous. Observe that since nodes of same type are ordered in the non-increasing order of their throughput,

$$\forall_{k \in [\![0,n]\!]} \sum_{i=0}^{k} b_i \geq \sum_{i=0}^{k} b'_i$$

and

$$\forall_{k \in [\![n+1,n+m]\!]} \sum_{i=1}^{k} b_i \geq \sum_{i=1}^{k} b'_i.$$

Hence, any acyclic scheme of I' can be turned into a scheme where I communications previously ensured by the k-th open (resp. guarded) node in I' are now ensured by the k first open (resp. guarded) nodes in I. The resulting scheme is acyclic and achieves the same throughput. □

Our first result states that the optimal acyclic (low degree) solutions achieve a throughput that is at least $\frac{5}{7}$ of the optimal cyclic solution (with possibly arbitrarily large degree) and that this $\frac{5}{7}$ bound is tight.

Theorem 3.6.3. *For any instance, $\frac{T^*_{ac}}{T^*} \geq \frac{5}{7}$. Moreover, there exists an instance such that this ratio is reached.*

Let us first show that the ratio 5/7 can be reached. For this purpose, let us consider the following instance (see Figure 22) consisting of one source of throughput 1, one open node of throughput $b_1 = 1 + 2\varepsilon$ and two guarded nodes with throughput of $b_2 = b_3 = 1/2 - \varepsilon$ each. For this instance, $T^* = 1$ (see Lemma 3.5.1). There also exist 3 increasing orderings $\sigma_1 = 0123, \sigma_2 = 0213$ and $\sigma_3 = 0231$. Ordering σ_1 achieves a throughput of $T^*_{ac}(\sigma_1) = (2/3).(1+\varepsilon)$ and ordering σ_2 achieves a throughput of $T^*_{ac}(\sigma_2) = 3/4 - \varepsilon/2$ (see Figure 22). The throughput of the last ordering is always smaller than the maximum of the two previous ones. When $\varepsilon = 1/14$, orderings σ_1 and σ_2 achieve the same throughput $T^*_{ac} = 5/7$.

Let us now prove that for any instance $\frac{T^*_{ac}}{T^*} \geq \frac{5}{7}$. Without loss of generality, we can consider only tight homogeneous instances. We can also assume that $n \geq 1$, $m \geq 2$ and $n + m \geq 4$ since other cases are trivial or have been considered above.

Let us consider the following two words

$$\omega_1(n,m) = \bigcirc \square^{\alpha_1} \bigcirc \square^{\alpha_2} \ldots \bigcirc \square^{\alpha_n},$$

$$\omega_2(n,m) = \square \bigcirc^{\beta_1} \square \bigcirc^{\beta_2} \ldots \square \bigcirc^{\beta_n}.$$

where $\alpha_i = \lfloor i \cdot \frac{m}{n} \rfloor - \lfloor (i-1) \cdot \frac{m}{n} \rfloor$ and $\beta_i = \lceil i \cdot \frac{n}{m} \rceil - \lceil (i-1) \cdot \frac{n}{m} \rceil$. Intuition of introduction of those two encoding words is as follow: to spread $\square$ or $\bigcirc$ (whichever is necessary) almost uniformly on the whole word. Such words encode almost optimal acyclic solutions, however they represent much more regular inner structure.

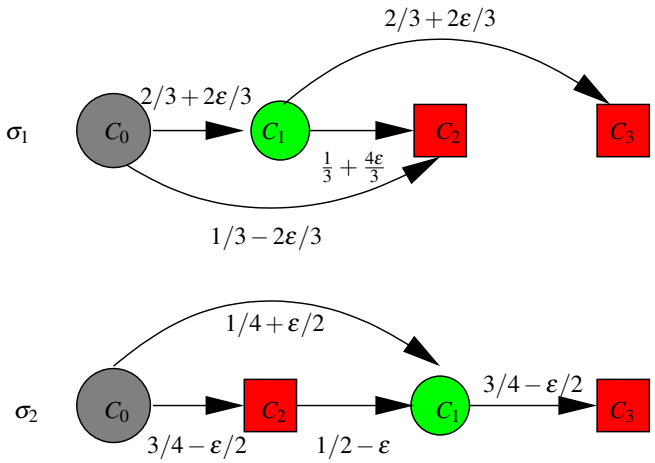

Figure 22: Optimal acyclic schemes of σ_1 and σ_2.

As observed in Section 3.4, these words encode increasing orders on vertices. To conclude the proof we only have to show that at least one of these two words encodes a valid scheme of throughput $5/7$

$$\max\left(T^*_{ac}(\omega_1(n,m)), T^*_{ac}(\omega_2(n,m))\right) \geq 5/7.$$

Recall that we consider tight homogeneous instances with $b_0 = 1$ such that $n \geq 1$, $m \geq 2$ and $n+m \geq 4$. Without loss of generality, we can also assume that $b_0 = 1, b_0 + O \geq m$ and $b_0 + O + G = n + m$. Hence for some $0 \leq \Delta \leq n$, the bandwidth of each open node is $o = \frac{m-1+\Delta}{n}$ and the bandwidth of each guarded node is $g = \frac{n-\Delta}{m}$. To show that $\max(T^*_{ac}(\omega_1(n,m)), T^*_{ac}(\omega_2(n,m))) \geq 5/7$, we will show a more precise statement

$$\begin{aligned} &\text{if } o \geq 1, T^*_{ac}(\omega_1(n,m)) \geq 5/7 \\ &\text{otherwise } T^*_{ac}(\omega_2(n,m)) \geq 5/7. \end{aligned} \tag{3}$$

Let us start with two additional technical lemmas.

Lemma 3.6.4. *For a tight homogeneous instance, a word ω is valid for throughput T if and only if*

- **(c1)** $\forall_{\pi \bigcirc \sqsubseteq \omega}\ b_0 + o \cdot |\pi|_\bigcirc + g \cdot |\pi|_\square - |\pi \bigcirc| \cdot T \geq 0$
- **(c2)** $\forall_{\pi' \bigcirc \sqsubseteq \pi \square \sqsubseteq \omega}\ b_0 + o \cdot |\pi|_\bigcirc + g \cdot |\pi'|_\square - |\pi \square|_\square \cdot T - |\pi' \bigcirc|_\bigcirc \cdot T \geq 0$

Proof. As shown in Section 3.4, a word ω is valid for a throughput T if and only if

$$\begin{aligned} &\text{for any prefix of } \omega \text{ of the form } \pi\square, G(\pi) \geq T, \\ &\text{for any prefix of } \omega \text{ of the form } \pi\bigcirc, G(\pi) + R(\pi) \geq T. \end{aligned}$$

For homogeneous instances, the second condition can be written as

$$\forall_{\pi \bigcirc \sqsubseteq \omega}\ b_0 + o \cdot |\pi|_\bigcirc + g \cdot |\pi|_\square - |\pi \bigcirc| \cdot T \geq 0$$

which is exactly (c1).

And the first condition is

$$\forall_{\pi\square\sqsubseteq\omega} b_0 + o\cdot|\pi|_{\bigcirc} - |\pi\square|_{\square}\cdot T - W(\pi) \geq 0.$$

From the recursive equations which define W and G, we can deduce

$$W(\pi\bigcirc) = \max(W(\pi), |\pi\bigcirc|_{\bigcirc}\cdot T - (b_{n+1} + \ldots + b_{n+|\pi|_{\square}})).$$

Together with $W(\pi\square) = W(\pi)$, this implies

$$W(\pi) = \max_{\pi'\bigcirc\sqsubseteq\pi} \{|\pi'\bigcirc|_{\bigcirc}\cdot T - (b_{n+1} + \ldots + b_{n+|\pi'|_{\square}})\},$$

hence the first condition can be rewritten to

$$\forall_{\pi'\bigcirc\sqsubseteq\pi\square\sqsubseteq\omega}, b_0 + o\cdot|\pi|_{\bigcirc} g\cdot|\pi'|_{\square} - |\pi\square|_{\square}\cdot T - |\pi'\bigcirc|_{\bigcirc}\cdot T \geq 0.$$

□

Lemma 3.6.5. *If ω is an encoding word for a valid solution of a homogeneous instance with $(b_0 = b_0', o = o', g = g')$ with throughput T, and ω is also encoding a valid solution of a homogeneous instance with $(b_0 = b_0'', o = o'', g = g'')$ with throughput T, and $0 \leq \lambda_1, \lambda_2 \leq 1$ are such that $\lambda_1 + \lambda_2 = 1$, then ω is also a valid solution for homogeneous instance with $(b_0 = \lambda_1\cdot b_0' + \lambda_2\cdot b_0'', o = \lambda_1\cdot o' + \lambda_2\cdot o', g = \lambda_1\cdot g' + \lambda_2\cdot g'')$ with throughput T.*

Proof. For fixed $\pi\bigcirc$, we can write (c1) as

$$b_0'' + o''\cdot|\pi|_{\bigcirc} + g''\cdot|\pi|_{\square} - |\pi\bigcirc|\cdot T = \lambda_1(b_0 + o\cdot|\pi|_{\bigcirc} + g\cdot|\pi|_{\square} - |\pi\bigcirc|\cdot T) +$$
$$+ \lambda_2(b_0' + o'\cdot|\pi|_{\bigcirc} + g'\cdot|\pi|_{\square} - |\pi\bigcirc|\cdot T) \geq 0.$$

Condition (c2) is proved analogously. □

Now let us go back to the proof of statement (3).

Since when $m > n$, it is impossible to have $o < 1$, we need to consider only 3 cases

- $m \geq n+1$ and $o \geq 1$,
- $m \leq n$ and $o \geq 1$,
- $m \leq n$ and $o \leq 1$.

Using Lemma 3.6.5, we can eliminate the parameter Δ from each of the cases, reducing each of them to two extreme cases

- $m \geq n+1$ and $o \geq 1$
  - **(A1)** $o = \frac{m-1}{n}$, $g = \frac{n}{m}$,

- **(A2)** $o = \frac{n+m-1}{n}$, $g = 0$.

- $m \leq n$ and $o \geq 1$
  - **(B1)** $o = 1$, $g = \frac{m-1}{m}$,
  - **(B2)** $o = \frac{n+m-1}{n}$, $g = 0$.

- $m \leq n$ and $o \leq 1$
  - **(C1)** $o = \frac{m-1}{n}$, $g = \frac{n}{m}$,
  - **(C2)** $o = 1$, $g = \frac{m-1}{m}$.

We now check for each case that the appropriate word ($\omega_1(n,m)$ or $\omega_2(n,m)$) satisfies the conditions **(c1)** and **(c2)** of Lemma 3.6.4.

Lemma 3.6.6. *In cases* **(A2)** *and* **(B2)**, $T^*_{ac} \geq 5/7$.

Proof. Merging cases **(A2)** and **(B2)** together, we consider the following instance

$$o = \frac{n+m-1}{n}, g = 0,$$

($o \geq 1$ obviously holds), and the encoding word $\omega_1(n,m)$.

It is enough to verify condition (c1), because open bandwidth is the only available bandwidth in this case. If we denote $|\pi|_{\bigcirc}$ as i, $0 \leq i < n$, (c1) becomes then

$$\forall_{0 \leq i < n} 1 + \frac{n+m-1}{n} \cdot i - \left(i + 1 + \left\lfloor \frac{m}{n} i \right\rfloor\right) \cdot \frac{5}{7} \geq 0.$$

Since $\lfloor \frac{m}{n} i \rfloor \leq \frac{m}{n} i$, it is enough to prove

$$1 + \frac{n+m-1}{n} \cdot i - \left(i + 1 + \frac{m}{n} \cdot i\right) \cdot \frac{5}{7} \geq 0$$

which simplifies to

$$\frac{2}{7} + \frac{2}{7} \cdot i + \frac{2}{7} \cdot \frac{m}{n} \cdot i - \frac{i}{n} \geq 0$$
$$2 + (2n + 2m - 7) \cdot i \geq 0$$

which holds, since $n + m \geq 4$. □

Lemma 3.6.7. *In cases* **(B1)** *and* **(C2)**, $T^*_{ac} \geq 5/7$.

Proof. Increasing n in those cases only results in adding open nodes with bandwidth $1 \geq \frac{5}{7}$. So it is enough to prove the two conditions for $n = m$.

So we now assume $n = m$. If $m \geq 4$, then $g \geq \frac{3}{4}$, and

$$\omega_1(n,n) = (\bigcirc\square)^n$$

$$\omega_2(n,n) = (\square\bigcirc)^n,$$

and every node is able to feed the next node.

If $m \leq 3$, we can easily verify that the words $\omega_1(2,2)$ and $\omega_2(2,2)$ are valid for the case $o = 1, g = \frac{1}{2}$ with throughput $T = \frac{5}{7}$, and that the words $\omega_1(3,3)$ and $\omega_2(3,3)$ are valid for the case $o = 1, g = \frac{2}{3}$ with throughput $T = \frac{5}{7}$. □

Lemma 3.6.8. *In case* **(A1)**, $T^*_{ac} \geq 5/7$.

Proof. Let us check condition (c1). For the sake of readability, let us denote by i the value $|\pi|_\bigcirc$ ($0 \leq i < n$). Condition (c1) can be rewritten to

$$1 + i \cdot \frac{m-1}{n} + \left\lfloor i \cdot \frac{m}{n} \right\rfloor \cdot \frac{n}{m} \geq \frac{5}{7} \cdot \left(1 + i + \left\lfloor i \cdot \frac{m}{n} \right\rfloor\right)$$

which is equivalent to

$$\frac{2}{7} + i \cdot \left(\frac{m-1}{n} - \frac{5}{7}\right) \geq \left\lfloor i \cdot \frac{m}{n} \right\rfloor \cdot \left(\frac{5}{7} - \frac{n}{m}\right).$$

Since $\frac{m-1}{n} \geq 1$, the left-hand side is always positive. We can safely assume that $\frac{5}{7} \geq \frac{n}{m}$ (otherwise, the right-hand side is negative). And since $\lfloor i \cdot \frac{m}{n} \rfloor < i \cdot \frac{m}{n}$, it is enough to prove

$$\frac{2}{7} + i \cdot \left(\frac{m-1}{n} - \frac{5}{7}\right) \geq i \cdot \frac{m}{n} \cdot \left(\frac{5}{7} - \frac{n}{m}\right).$$

Simplifying, we get

$$\frac{2}{7} + i \cdot \left(\frac{2}{7} + \frac{2}{7} \cdot \frac{m}{n} - \frac{1}{n}\right) \geq 0.$$

Since $\frac{2}{7} + \frac{2}{7} \cdot \frac{m}{n} \geq \frac{2}{7} + \frac{2}{7} \cdot \frac{7}{5} \geq \frac{1}{2} \geq \frac{1}{n}$, condition (c1) holds.

Let us now check condition (c2). We denote $|\pi|_\bigcirc = i$ and $|\pi'|_\bigcirc = j$ (it is enough to consider the longest such π'), $0 \leq j < i \leq n$.

$$1 + i \cdot \frac{m-1}{n} + \left\lfloor \frac{m}{n} \cdot j \right\rfloor \cdot \frac{n}{m} - \left\lfloor \frac{m}{n} \cdot i \right\rfloor \cdot \frac{5}{7} - (j+1) \cdot \frac{5}{7} \geq 0.$$

Simplifying, we get

$$\left(\frac{2}{7} + \left\lfloor \frac{m}{n} \cdot j \right\rfloor \cdot \frac{n}{m} - \frac{5}{7} \cdot j\right) + \frac{5}{7} \cdot \left(\frac{m}{n} \cdot i - \left\lfloor \frac{m}{n} \cdot i \right\rfloor\right) + \frac{i}{n} \cdot \left(\frac{2}{7} \cdot m - 1\right) \geq 0.$$

Now, we can also use this

$$\forall_{x \geq 0} \lfloor x \rfloor = \lfloor x \rfloor \cdot \frac{\lfloor x \rfloor + 1}{\lfloor x \rfloor + 1} \geq \frac{x \cdot \lfloor x \rfloor}{\lfloor x \rfloor + 1}.$$

So we have

$$\left\lfloor \frac{m}{n} \cdot j \right\rfloor \geq \frac{m}{n} \cdot j \cdot \frac{\lfloor \frac{m}{n} \cdot j \rfloor}{\lfloor \frac{m}{n} \cdot j \rfloor + 1} \geq \frac{m}{n} \cdot j \cdot \frac{j}{j+1}.$$

Using this, condition (c2) holds if

$$\left(\frac{2}{7}+j\cdot\frac{j}{j+1}-\frac{5}{7}\cdot j\right)+\frac{5}{7}\cdot\left(\frac{m}{n}\cdot i-\left\lfloor\frac{m}{n}\cdot i\right\rfloor\right)+\frac{i}{n}\cdot\left(\frac{2}{7}\cdot m-1\right)\geq 0.$$

The case $m\leq 3$ can be checked separately ($m=3,n=2,o=1,r=\frac{2}{3}$). When $m\geq 4$, we have $\frac{2}{7}m-1>0$, and it remains to prove that

$$\frac{2}{7}+j\cdot\frac{j}{j+1}-\frac{5}{7}\cdot j\geq 0$$

For $j\in\{0,1,2,3\}$ it can be checked, and for $j\geq 4$ the following inequality allows us to conclude

$$j\cdot\frac{j}{j+1}\geq\frac{5}{7}\cdot j.$$

□

Lemma 3.6.9. *In case* **(C1)**, $T^*_{ac}\geq 5/7$.

Proof. If we denote $i=|\pi|_\square$ (the case $i=0$ is trivial, so we can assume $0<i<m$), condition (c1) can be written as

$$1+\frac{n}{m}\cdot i+\frac{m-1}{n}\left\lceil i\cdot\frac{n}{m}\right\rceil\geq\frac{5}{7}\left(1+i+\left\lceil i\cdot\frac{n}{m}\right\rceil\right)$$

which simplifies to

$$\frac{2}{7}+\left(\frac{n}{m}-\frac{5}{7}\right)i\geq\left(\frac{5}{7}-\frac{m-1}{n}\right)\cdot\left\lceil i\cdot\frac{n}{m}\right\rceil.$$

We can safely assume that $\frac{5}{7}-\frac{m-1}{n}\geq 0$, because otherwise the right-hand side would be negative, while the left-hand side remains positive. Since $\lceil x\rceil\leq x+1$, the following equality implies (c1)

$$\frac{2}{7}+\left(\frac{n}{m}-\frac{5}{7}\right)i\geq\left(\frac{5}{7}-\frac{m-1}{n}\right)\cdot\left(i\cdot\frac{n}{m}+1\right)$$

And simplifies to

$$\left(\frac{2}{7}\cdot\frac{n}{m}+\frac{m-1}{m}-\frac{5}{7}\right)\cdot i+\frac{m-1}{n}\geq\frac{3}{7}.$$

We can observe that $\frac{2}{7}\cdot\frac{n}{m}+\frac{m-1}{m}-\frac{5}{7}\geq\frac{2}{7}+\frac{1}{2}-\frac{5}{7}>0$, so that the left-hand side of this inequality is minimized with $i=1$. It is thus enough to check that

$$\frac{2}{7}\cdot\frac{n}{m}+\frac{m-1}{m}+\frac{m-1}{n}\geq\frac{8}{7}.$$

This is what we do here:

$$\frac{2}{7}\cdot\frac{n}{m}+\frac{m-1}{m}+\frac{m-1}{n}\geq\frac{2}{7}\cdot\frac{n}{m}+\frac{1}{2}+\frac{m-1}{m}\cdot\frac{m}{n}\geq$$

$$\geq \frac{1}{2}+\frac{2}{7}\cdot\frac{n}{m}+\frac{1}{2}\cdot\frac{m}{n}\geq\frac{1}{2}+2\cdot\sqrt{\frac{2}{7}\times\frac{1}{2}}\geq\frac{8}{7}.$$

Let us now check condition (c2). Here, we denote $|\pi|_\square$ by i and $|\pi'|_\square$ by j ($0\leq j\leq i<m\leq n$) (We only have to consider the longest π'). Condition (c2) becomes $\forall 0\leq j\leq i<m\leq n$

$$1+\left\lceil i\cdot\frac{n}{m}\right\rceil\cdot\frac{m-1}{n}+j\cdot\frac{n}{m}-\frac{5}{7}\cdot\left(1+\left\lceil j\cdot\frac{n}{m}\right\rceil+i\right)\geq 0.$$

Simplifying, and substituting $\lceil x\rceil$ by x we get

$$\frac{2}{7}+i\cdot\frac{m-1}{m}+j\cdot\frac{n}{m}-\frac{5}{7}\cdot\left\lceil j\cdot\frac{n}{m}\right\rceil-\frac{5}{7}\cdot i\geq 0.$$

First, we solve the case when $j=0$

$$\frac{2}{7}+i\cdot\left(\frac{m-1}{m}-\frac{5}{7}\right)\geq 0$$

- $m=2$

$$\frac{2}{7}+i\cdot\left(\frac{1}{2}-\frac{5}{7}\right)\geq 0$$
$$i\leq\frac{4}{3}$$

  that holds true since $i<m=2$

- $m=3$

$$\frac{2}{7}+i\cdot\left(\frac{2}{3}-\frac{5}{7}\right)\geq 0$$
$$i\leq 6$$

  that holds true since $i<m=3$

- $m>3$

$$\frac{m-1}{m}\geq\frac{5}{7}.$$

Which solves case $j=0$.

Now we can safely assume $j\geq 1$. Using $\lceil\frac{a}{b}\rceil\leq\frac{a+b-1}{b}$, it is sufficient to prove that

$$\frac{2}{7}+i\cdot\frac{m-1}{m}+j\cdot\frac{n}{m}-\frac{5}{7}\cdot\left(j\cdot\frac{n}{m}+\frac{m-1}{m}\right)-\frac{5}{7}\cdot i\geq 0,$$

which simplifies to

$$\frac{2}{7}\cdot i+\frac{2}{7}\cdot j\cdot\frac{n}{m}+\frac{5}{7}\cdot\frac{1}{m}\geq i\cdot\frac{1}{m}+\frac{3}{7}.$$

Obviously, the left-hand side is minimized for $j=1$:

$$\frac{2}{7}\cdot i+\frac{2}{7}\cdot\frac{n}{m}+\frac{5}{7}\cdot\frac{1}{m}\geq i\cdot\frac{1}{m}+\frac{3}{7}.$$

We consider three cases:

- $m = 2$ (then, $i = 1$ must hold):

$$\frac{2}{7} + \frac{2}{7} \cdot \frac{n}{2} + \frac{5}{7} \cdot \frac{1}{2} \geq \frac{1}{2} + \frac{3}{7}$$

  which is equivalent to $n \geq 2$ (that holds).

- $m = 3$ (then, $1 \leq i < 3$ must hold)

$$\frac{2}{7} \cdot \frac{m}{3} + \frac{5}{7} \cdot \frac{1}{3} \geq \frac{1}{21} \cdot i + \frac{3}{7}$$

  which is equivalent to $2n \geq i + 4$ (that holds).

- $m \geq 4$

$$\left(\frac{2}{7} - \frac{1}{m}\right) \cdot i + \frac{2}{7} \cdot \frac{n}{m} + \frac{5}{7} \cdot \frac{1}{m} \geq \frac{3}{7}$$

  right-hand side minimizes for $i = 1$

$$\frac{2}{7} - \frac{1}{m} + \frac{2}{7} \cdot \frac{n}{m} + \frac{5}{7} \cdot \frac{1}{m} \geq \frac{3}{7}$$

  which simplifies to $2n \geq 2 + m$, that also holds.

□

We just proved that acyclic solutions have throughput at least $\frac{5}{7}$ of the optimal solution. Looking at Figure 23, we see that the worst case happens for small n and m, and ratio grows with size of instances. One could hope, that ratio goes arbitrarily close to 1, but unfortunately that is not the case.

The second result states that the optimal acyclic throughput does not get arbitrarily close to the optimal cyclic throughput when the size of the instances grows.

Theorem 3.6.10. *For every $\varepsilon > 0$ and every $K \in \mathbb{N}$, there exist instances with at least K open nodes and K guarded nodes such that*

$$\frac{T^*_{ac}}{T^*} \leq \frac{1 + \sqrt{41}}{8} + \varepsilon \approx 0.925 + \varepsilon.$$

Proof. of Theorem 3.6.10. For a given $\alpha = \frac{p}{q} < 1$, (p and q have integer values), and for any k, let us consider the instance $I(\alpha, k)$ such that:

- $b_0 = 1$;
- $n = kq$ open nodes have bandwidth α; and
- $m = kp$ guarded nodes have bandwidth $\frac{1}{\alpha}$.

The first observation is that for all α and k, Lemma 3.5.1 implies that the optimal throughput T^* is equal to 1.

For the second observation, let S be any acyclic solution to $I(\alpha,k)$ and x be the number of open nodes before the second guarded node in S. In other words, S starts with a prefix $\pi = \bigcirc^u \square \bigcirc^v \square$ with $u+v=x$. The throughput T achievable by S is bounded by two constraints

- the source and the first x open nodes should be able to feed the first two guarded nodes, ie. $\alpha x + 1 \geq 2T$, and
- the bandwidth of the source and of the first $x+1$ nodes should be enough to feed the $x+2$ nodes, ie. $\alpha x + \frac{1}{\alpha} + 1 \geq (x+2)T$.

Hence $T \leq \frac{\alpha x+1}{2} = f_\alpha(x)$ and $T \leq \frac{\alpha x+\frac{1}{\alpha}+1}{x+2} = g_\alpha(x)$. Since any optimal acyclic scheme must satisfy these two constraints for some x, we have $T^*_{ac} \leq \max_{x\in\mathbb{N}} \min(f_\alpha(x), g_\alpha(x))$.

Observe now that the function f_α is increasing, and g_α is decreasing (since $\alpha < 1$), and that they coincide (with value 1) for $x = \frac{1}{\alpha}$. The minimum is thus achieved by f_α for $x < 1/\alpha$, and by g_α for $x > 1/\alpha$, and this minimum is maximized for $x = 1/\alpha$. However, $\frac{1}{\alpha}$ is not necessarily an integer, so the maximal value is achieved for $x = \lfloor \frac{1}{\alpha} \rfloor$ or $x = \lceil \frac{1}{\alpha} \rceil$:

$$T^*_{ac} \leq \max\left(f_\alpha\left(\left\lfloor \frac{1}{\alpha} \right\rfloor\right), g_\alpha\left(\left\lceil \frac{1}{\alpha} \right\rceil\right)\right).$$

If $\alpha = \frac{\sqrt{41}-3}{8}$, simple computations show that $\lfloor \frac{1}{\alpha} \rfloor = 2$, $\lceil \frac{1}{\alpha} \rceil = 3$, and $f_\alpha(2) = g_\alpha(3) = \frac{\sqrt{41}+1}{8}$. Since this value of α can be approximated arbitrarily close with a rational number, and since the expressions $f_\alpha(2)$ and $g_\alpha(3)$ are continuous in α, we get the claimed result. □

To conclude this subsection, we show an exhaustive exploration of all possible tight and homogeneous instances, for n and m between 0 and 100. For each of them we compute the ratio T^*_{ac}/T^* (see Figure 23).

On the one hand, we can observe the result of Theorem 3.6.10: when $m \simeq \frac{\sqrt{41}-3}{8} n$ (for example $n = 100$ and $m = 42$), the ratio remains below 1, even for large values of n and m. On the other hand we can observe that except for few small instances, the ratio T^*_{ac}/T^* is larger than 0.8.

3.6.2 Average cases

In addition to this worst-case analysis, we also analyze the average ratio between acyclic and cyclic throughput of randomly generated instances. In order to explore the performance of our algorithms in different heterogeneity conditions, we consider several probability distributions for the bandwidths of the nodes

1. a uniform distribution between 1 and 100 (**Unif100**);
2. power-law (Pareto) distributions with average value 100 and standard deviation 100 (**Power1**) or 1000 (**Power2**);

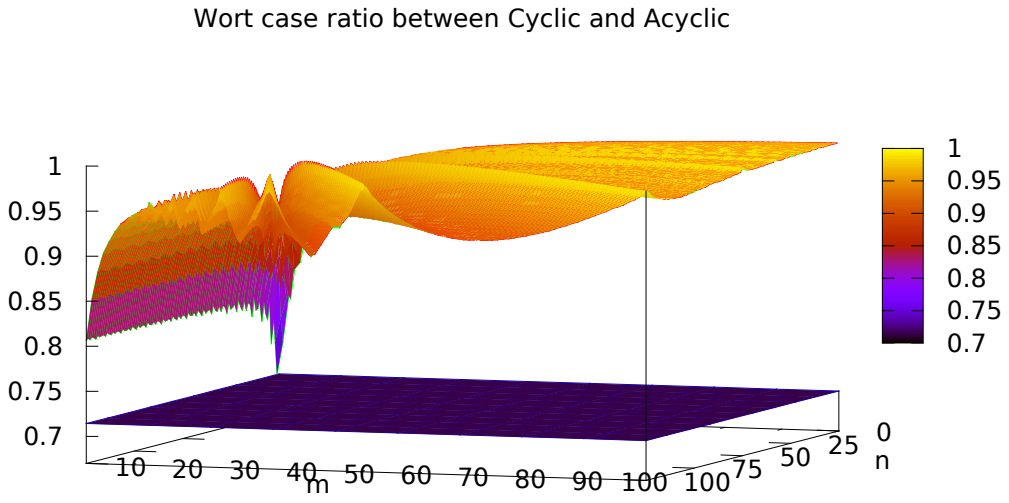

Figure 23: Worst case ratio between cyclic and acyclic optimal solutions on tight homogeneous instances. The bottom plane is $\frac{5}{7} \simeq 0.714$.

3. log-normal distributions with average value 100 and standard deviation 100 (**LN1**) and 1000 (**LN2**);

4. a uniform sampling from outgoing bandwidth values that were computed from measurements performed on the PlanetLab platform [11] (**PLab**).

In each case, each node is independently chosen to be an open node with probability p (and a guarded with probability $(1-p)$). In order to concentrate on difficult instances, the bandwidth of the source node is chosen equal to the optimal cyclic throughput – what ensures that the source is not a strong limiting bottleneck, and that it is also not sufficient by itself to feed all nodes. The results are shown on Figure 24, for different numbers of nodes and different values of p. For each set of parameters, 1000 random instances were generated, and the figure shows average values (connected by black lines) and boxplots with median, quantiles, and confidence intervals at 5% (the black dots are outliers, outside these confidence intervals).

The first conclusion of these simulations is that the average behavior of acyclic solutions is very close to the optimal cyclic throughput, and that this is true in a wide variety of scenarios. Furthermore, the results are very stable. We can note that more open nodes and moderate heterogeneity (with the **Power1** and **Power2** distributions) make the problem slightly more difficult for small size instances. Overall, however, we can see that even in these cases, producing low degree solutions comes at very little cost (at most 5%) with respect to the achievable throughput.

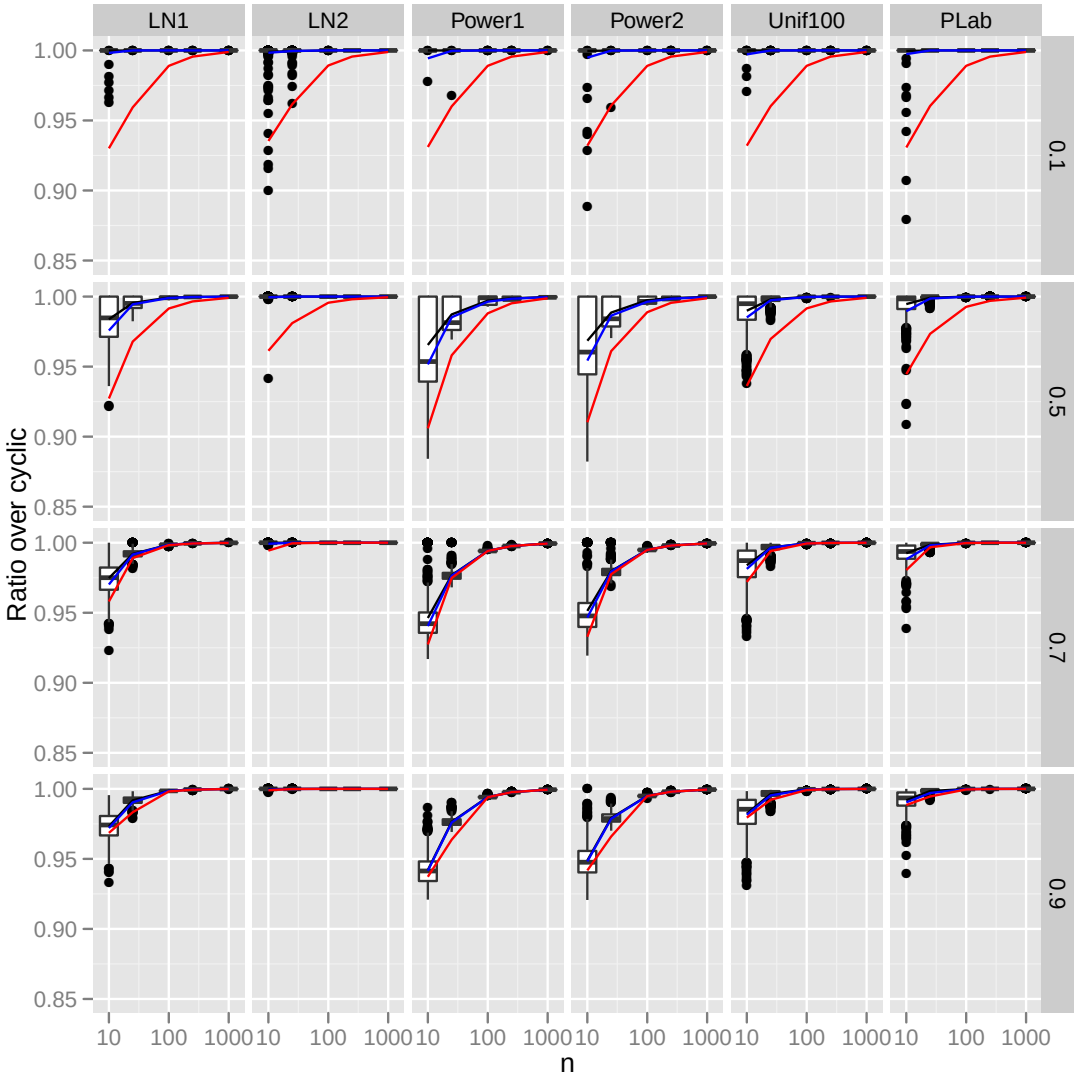

Figure 24: Average ratio between cyclic and acyclic optimal solutions of randomly generated instances.

The second conclusion is related to the acyclic throughput obtained considering only the best solution among those encoded by words ω_1 and ω_2 (blue lines on Figure 24). In all cases, these solutions are almost as competitive as the best acyclic ones and for all large instances they are as competitive. From a practical point of view these simpler schemes are of interest since they are easier to build in a distributed context once nodes have been ordered according their bandwidth. For comparison, the average throughput obtained by the word (either ω_1 or ω_2) used in the case analysis of the proof of Theorem 3.6.3 is shown by the red lines on Figure 24. We can see that there is a significant gap for smaller instances, hence it can be actually worthwhile to compute the best throughput among both words.

3.7 Conclusion

We have considered the classical problem of broadcasting a large message at an optimal rate in a large scale distributed and heterogeneous network. We have advocated the use of the LastMile model, that encompasses the advantages of both the bounded multiport model and the 1-port model.

The main originalities of our work are that we consider the case where some broadcast nodes lie in the open Internet whereas other broadcast nodes are guarded (behind NATs or firewalls), and that we search for either cyclic or acyclic solutions.

We propose algorithms that provide solutions with low degree (optimal up to a small constant additive term) and that achieve optimal throughput for the acyclic case. For the problem of finding cyclic solution, we prove that reaching the optimal throughput may require arbitrarily large degree at some nodes, but on the other hand, we prove a tight worst-case bound of 5/7 for the ratio between acyclic and cyclic cases. We also show, using a large set of simulations, that this ratio is even closer to 1 in practice, so that acyclic low degree solutions can be used for the cyclic case also.

Therefore, we provide low complexity algorithms (even if their proofs of correctness are sometimes sophisticated) that achieve optimal throughput (or quasi-optimal in the cyclic case) and where each node is connected to a minimal number of neighbors (up to a small additive constant ranging from 1 to 4 depending on the case), what makes implementation more efficient. We believe that this makes the proposed algorithms efficient both theoretically and practically.

Work presented in this chapter opens many perspectives. On the practical side, we rely on the LastMile model whose parameters can be easily evaluated at runtime, as shown in previous chapter, and we provide low degree acyclic solutions. Therefore, we expect that the theoretical results proved here can indeed be achieved in practice. On the theoretical side, since the use of the LastMile model enables to design (quasi-)optimal solutions with respect to both degree and throughput, we can introduce new objectives, such as dealing with the dynamicity of the platform (changing dynamically the set of participating nodes, or their communication abilities) or optimizing the depth of produced schemes in order to minimize delays.

Chapter 4

Power efficient communication over grid topologies

4.1 Introduction

4.1.1 Context

In previous chapters, we saw how the problem of instantiating the network parameters can be solved, when we assume certain models of networks. Also, when given such a model, we saw how we can solve certain combinatorial problems connected with efficient communication allocation over the network. That is the typical problem we will encounter when trying to model large scale communication over an unknown platform, *e.g.* peer-to-peer networks. However, when we try to move to different type of platforms, *e.g.* cloud computing, there arises a dual problem, concerning how to allocate transmission for just a known set of communications, or even a single communication. Previously we were free to decide on the amount of flow going through a given node, or the structure of point-to-point communication, without control of the actual routing of node to node communication. Now we will fix the endpoints of communication, and actually try and find efficient routing. We are free to do so, because it is possible to know the topology of the underlying network, or the infrastructure, when previously we could only assume the minimal knowledge. If we are going to analyze just the structure of communication over known network, we usually know in advance the transfer we want to achieve. Also, the maximal possible transfer in the flow related models is usually easy to compute (if we know bandwidth constraints over the edges), or infinite (when there are no constraints). So we have to provide a realistic cost function that will evaluate efficiency of the solution.

4.1.2 Motivation

In this chapter, we present results from our research on cost efficient routing over high performance computing platforms. Our point of interest is the search for power-efficient routing on rectangular grid topologies. We investigate the problem of routing communications between CMP (Chip

Multiprocessors) cores using shortest paths, in a model where the power cost associated with activating a communication link at a transmission speed of f bytes/second is proportional to f^α, for some constant exponent $\alpha > 2$. Power usage is a natural factor to optimize when investigating chip multiprocessors (CMP). For them, a significant part of power consumption is attributed to communications between cores, and power inefficiency is one of main limits preventing from further miniaturization of CMP [52]. Grid topology was chosen because it reflects well high performance platforms topologies. It can also be a basis for more complicated ones, and we hope for possible extensions of our results (for example planar topologies).

The starting point for our work in this chapter was the publication [13], that provided the setting (topology with power–transmission rate function) with an extensive theoretical and practical analysis of the problem. Key elements of the setting were already known and analyzed before. In [13], those ideas were for the first time put together and used to examine a wide category of routing algorithms. However, we felt that the part regarding efficient splitting of messages into atomic parts to be sent over single paths was underdeveloped, and we chose it as a way to improve this work.

With this research, we aimed at finding a balance between the simplicity of a model (that allows a deep mathematical analysis), and the complexity of actual processes (captures the real life setting of the problem). The setting we found allowed us for a very efficient theoretical analysis, while still reflecting the real life setting of high performance computing power efficiency.

Our main result is a trade-off showing how the power required for communication in CMP grids depends on the ability to split communication requests between a given pair of nodes, and then route each such request along multiple paths. For a pair of cores in a $n \times n$ grid, the number of available communication paths between them grows exponentially with n. By contrast, we show that the optimal power consumption (up to constant factors) can be achieved by splitting each communication request into k paths, starting from a threshold value of $k = \Theta(n^{1/(\alpha-1)})$. This threshold is much smaller than n for typical values of $\alpha \approx 3$, and may be considered practically feasible in routing schemes on the grid. More generally, we provide efficient algorithms for routing multiple k-splittable communication requests between any two cores in the grid, providing solutions within a constant approximation of the optimum cost. We support our results with simulations, showing that for practical instances, our approach using k-splittable requests leads to a power cost close to the one of the optimal solution with arbitrarily splittable requests, starting from the stated threshold value of k.

4.1.3 Setting

The increase in the level of integration of single chip multiprocessors (CMPs) creates demand for high-speed communications on-chip, which in turn increases the power consumption on CMP. This trend is expected to continue in the future [15]. Numerous studies concern the optimization of power cost in integrated chip designs, taking into account that both processors and communication buses may operate at variable frequency, determining the speed of computations or transmissions (cf. [28, 37, 48, 56]). The increase of power cost P as a w^α for $\alpha \approx 3$ (where w is a workload) in such designs is a well-established relation (cf. *e.g.* [56, 4, 18]).

A significant part of power in CMPs is consumed by maintaining communications within the chip, and that makes efficient allocation of communication routes a very important issue [52]. On CMP grids, links with dynamic frequency and/or voltage scaling are used ([38, 59]), and power P dissipated on a link is related to the frequency f and voltage V on it by the following relation supported by both theory and experiments $P \sim f \cdot V^2$ (cf. *e.g.* [3]). However, for most designs, an increase in operating frequency also results in an increase in voltage, roughly according to the relation $V \sim f$ (cf. *e.g.* [59]), which results in the relation between power cost and transmission speed given as $P \sim f^3$ ([13, 14]).

Such a model of power consumption was recently studied in the context of splittable Manhattan-path routing by Benoit *et al.* [13]. They introduced several routing schemes in an effort to minimize the overall power cost, but observed that this may require splitting each communication request, and routing its fragments along a potentially very large number of communication paths. Splitting a request, taking care of the route for each part, and merging it at the target imposes additional time and power overhead.

In this work, our goal is to show how to *limit path splitting as much as possible*, without increasing too much the communication power cost. Specifically, we consider the problem of optimizing the power consumption cost of a communication between two given cores, that may sometimes require the routing of multiple requests. Our power consumption model assumes that if an edge is transmitting at rate v, the power cost of maintaining the frequency over an edge is proportional to v^α for a given constant $\alpha > 2$, identical for every edge. We make the practically-motivated assumption [38, 59] that only the dynamic part (associated with transmission) is dominant for high communication rate, and static effects do not have to be considered in optimization. We studied more general case of v^α instead of fixed v^3 one, to provide much stronger argument for our analysis, that obtained results are not related to any 'special' properties of cubic function.

4.1.4 Outline and results.

Our study concerns routing between a single source-sink pair of nodes using Manhattan paths on a grid CMP. Communication between these nodes is assumed to be static, *i.e.* constant over time, and the cost of a transmission along an edge is assumed to be proportional to a fixed power function of the transmission rate. The considered model, power cost function, and rules of routing are formally presented in Section 4.2. We briefly outline the theory of Manhattan-path routing with arbitrarily splittable requests (Max-MP). We provide an optimal convex programming formulation of the problem, leading to a routing scheme denoted as OPT, and recall the properties of the routing scheme $\mathcal{C}$ introduced in [13]. We also provide a convenient formulation of Manhattan routings in terms of transmission through nodes.

Our main results are given in Section 4.3. They concern the variant of the Manhattan routing problem in which each request can be satisfied by at most k communication paths, where k is a parameter of the model (k-MP). We study the value of the ratio of the cost of the optimal solution in this case, denoted OPT_k, to the cost of the routing scheme OPT with arbitrarily splittable paths. We establish that in general, $\text{cost}(\text{OPT}_k)/\text{cost}(\text{OPT}) = O(1 + \frac{n}{k^{\alpha-1}})$, whereas for the special case of $d \geq 1$ identical requests of the same size, this ratio is given precisely as $\Theta(1 + \frac{n}{(kd)^{\alpha-1}})$. This means

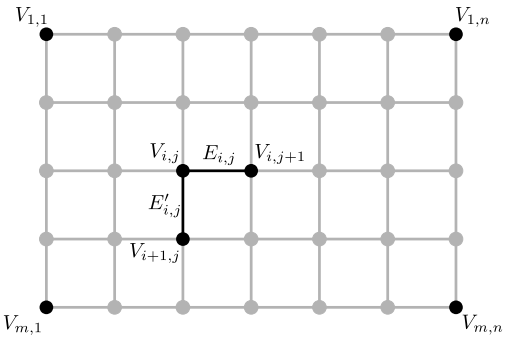

Figure 25: Notations used in the CMP grid model.

that for $k = o(n^{1/(\alpha-1)})$, the requirement that requests can be split into at most k paths impacts the cost of the routing scheme asymptotically, *i.e.*, increases the cost by an unbounded factor for sufficiently large n. On the other hand, for k larger than the threshold value of $\Theta(n^{1/(\alpha-1)})$, the obtained k-splittable routings are within a constant factor of the optimal solution to Max-MP routing problem.

The proposed bounds are obtained through the analysis of three efficiently implementable algorithmic schemes for solving k-MP: $\mathcal{F}_k$ routing and $\mathcal{D}_k$ routing for uniform requests (requests of same size), and $\mathcal{A}_k$ routing for non-uniform requests. The latter two are shown to have a constant approximation ratio with respect to the cost of OPT_k for all k, while the former converges to the cost of OPT as k goes to infinity. The design of such approximate techniques results from the observation that solving optimally the non-uniform k-MP routing problem is NP-hard.

Finally, in Section 4.4, we perform an experimental validation through simulations, of the determined threshold value of $k = \Theta(n^{1/(\alpha-1)})$, showing the effect of smaller and larger values of k on the routing cost. We also experimentally compare the performance of $\mathcal{F}_k$ routing and $\mathcal{D}_k$ routing, studying their convergence to asymptotic behavior for increasing values of k and different values of the power cost exponent $\alpha \approx 3$.

4.2 Framework

4.2.1 Platform and power consumption model.

We model the platform as a grid graph on a set of $m \times n$ uniform nodes $V_{i,j}$, with $1 \leq i \leq m$ and $1 \leq j \leq n$. Without loss of generality, we assume that $m \geq n$. We will also assume for the purpose of analysis that the sides of the grid are of the same order of magnitude, *i.e.*, $m = O(n)$. Nodes are connected by bidirectional edges. The horizontal edge $E_{i,j}$ connects $V_{i,j}$ and $V_{i,j+1}$ (for $1 \leq i \leq m$, $1 \leq j \leq n-1$), and the vertical edge $E'_{i,j}$ connects $V_{i,j}$ and $V_{i+1,j}$ ($1 \leq i \leq m-1$, $1 \leq j \leq n$), see the Figure 25 for an illustration.

The power consumed on each edge is closely related to the amount of data sent through this edge in a unit of time. To simplify the analysis of the model, we discard constant factors, and (following [13]) set the cost of transmission at rate x as $C(x) = x^\alpha$, where $\alpha > 2$ is an absolute

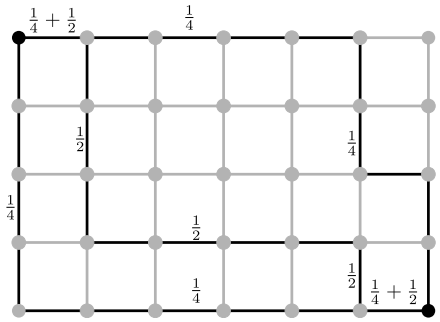

Figure 26: Example of 3-MP routing on 6×4 grid with a single request of size 1 being split into 3 paths of sizes $s_1 = \frac{1}{2}, s_2 = \frac{1}{4}$ and $s_3 = \frac{1}{4}$

constant of the model.

4.2.2 Communication and routing rules.

The study of routing with Manhattan-type paths (of shortest length) is motivated by practical concerns, in particular, the need to minimize communication latency, and to confine communications between nearby processors to a local area of the grid. For the purpose of the study of single source-sink communications, it is assumed that the source and target are placed in the opposite corners of the grid; for communications between a different pair of nodes, considerations can be restricted to the respective rectangular sub-grid.

A *routing* R of a single communication request of size s is a weighted set of paths, $\{(w_1, p_1), \ldots, (w_k, p_k)\}$, where each path p_i starts at the same source vertex $V_{1,1}$, and ends at the same target vertex $V_{m,n}$ in the opposite corner of the grid. The real-valued weights w_i satisfy $w_i \geq 0$ and $\sum_i w_i = s$. This definition of a routing naturally extends to a set of $d \geq 1$ requests, which may be *uniform* (with identical request size $s = K/d$, where K is total size of requests), or *non-uniform* (with possibly distinct request sizes $s_1, \ldots, s_d$).

Given a routing R, we define $R(e)$ as the size of the transmission going through an edge e, *i.e.*: $R(e) = \sum_{i:\ e \in p_i} w_i$. We will adapt this notation accordingly for routings denoted by letters different from R.

The *routing policy* is expressed by the bound k on the splitability of each request:

- In 1-*Paths Manhattan Routing (*1-MP*)*, single communications are atomic and we are not allowed to divide them into smaller parts. However, there may be several requests between the same pair of endpoints, so in fact this model doesn't differ much from the following ones.

- In *k-Path Manhattan Routing (*k-MP*)*, communication for each request can be split into any number of $k' \leq k$ (partially overlapping) source-sink paths, where k is a parameter of the model. Example of such a routing is on the Figure 26.

- In *Max-Paths Manhattan Routing (*Max-MP*)*, the number of paths allowed for each request is unbounded ($k = +\infty$).

4.2.3 Problem definition.

For a given routing policy with parameter k and power coefficient α, we define our optimization problem as follows: *Given a $m \times n$ grid and a set of requests of sizes $(s_1, \ldots, s_d)$, with $\sum_{i=1}^{d} s_i = K$, find a routing R of this set of requests minimizing the total power cost of transmission through all the edges of the grid, expressed by the cost function:*

$$\text{cost}(R) = \sum_{i=1}^{m} \sum_{j=1}^{n-1} R\left(E_{i,j}\right)^{\alpha} + \sum_{i=1}^{m-1} \sum_{j=1}^{n} R\left(E'_{i,j}\right)^{\alpha}.$$

4.2.4 Solution to Max-MP Routing.

For Max-MP, the routing policy does not impose a bound on k. We will denote the optimal solution to Max-MP by OPT and use it as a reference for k-splittable routing algorithms. The adopted definition of routing cost leads directly to a convex-programming formulation of Max-MP routing (see Algorithm 4), and thus applications of convex programming algorithms lead to polynomial-time schemes with arbitrarily good approximation of OPT (cf. *e.g.* [2, 34] for a discussion of convex programming in the context of finding min-cost flows).

Algorithm 4 OPT routing {for Max-MP}

Input: A set of arbitrarily splittable requests of total size K in a $m \times n$ grid.
Solve:

$$\text{Minimize} \left(\sum_{i,j} \text{OPT}\left(E_{i,j}\right)^{\alpha} + \sum_{i,j} \text{OPT}\left(E'_{i,j}\right)^{\alpha} \right)$$

Subject to:

$$\text{OPT}\left(E_{i,j}\right) \geq 0$$
$$\text{OPT}\left(E'_{i,j}\right) \geq 0$$
$$\text{OPT}\left(E_{i,j-1}\right) + \text{OPT}\left(E'_{i-1,j}\right) = \text{OPT}\left(E_{i,j}\right) + \text{OPT}\left(E'_{i,j}\right)$$
$$\text{OPT}\left(E_{1,1}\right) + \text{OPT}\left(E'_{1,1}\right) = K$$

Output: values of $\text{OPT}\left(E_{i,j}\right), \text{OPT}\left(E'_{i,j}\right)$

We remark on the following lower bound on the cost of the OPT routing. Consider any Max-MP routing that transmits requests of total size K. The edges adjacent to node $V_{1,1}$, *i.e.*, $\{E_{1,1}, E'_{1,1}\}$, have to transmit requests of size K in total. It follows that:

$$\text{cost}(\text{OPT}) \geq \left(\tfrac{K}{2}\right)^{\alpha} = \Theta\left(K^{\alpha}\right). \tag{4}$$

Remarkably, as shown in [13], this lower bound is tight regardless of the size of the grid, since it can be achieved using a specific routing scheme. We will provide a definition of a scheme

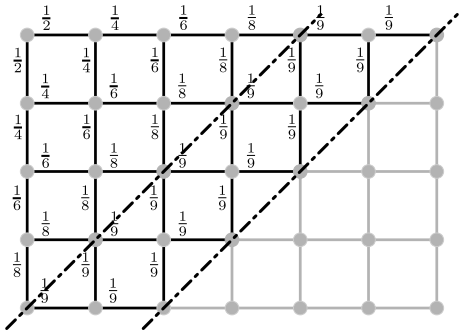

Figure 27: Visualization of lower bound argument, where we also clearly see infeasibility of the construction. For example, for vertex $V_{2,2}$, there is incoming transmission of $\frac{1}{2}$, while outgoing of $\frac{1}{3}$.

called $\mathcal{C}$ which has equivalent properties, but is described from a different perspective, based on load balancing on so-called *vertex diagonals*. We will then use this scheme as a starting point for schemes solving k-MP.

A better lower bound (in terms of constant) for max-MP routing can be obtained by a carefull estimation. Based on a remark that every diagonal DE_k has to transmit K in total.

$$\begin{aligned}
lower(K) &= \sum_{i=1}^{n+m-2} |DE_i| \cdot \left(\frac{K}{|DE_i|}\right)^{\alpha} \\
&= (m-n)\cdot(2n+1)\cdot\left(\frac{K}{2n+1}\right)^{\alpha} + 2\cdot\sum_{i=1}^{n-1} 2i\cdot\left(\frac{K}{2i}\right)^{\alpha} \\
&\geq (m-n)\cdot(2n+1)\cdot\left(\frac{K}{2n+1}\right)^{\alpha} + 2\cdot\int_1^n \frac{K^{\alpha}}{(2x)^{\alpha-1}}dx \\
&= K^{\alpha}\left(\frac{m-n}{2n+1}\cdot\frac{1}{(2n+1)^{\alpha-2}} + \frac{1}{2^{\alpha-2}(\alpha-2)}\left(1-\frac{1}{n^{\alpha-2}}\right)\right) \\
&\geq (m-n)\cdot(2n+1)\cdot\left(\frac{K}{2n+1}\right)^{\alpha} + 2\cdot\int_1^{\infty} \frac{K^{\alpha}}{(2x)^{\alpha-1}}dx \\
&= K^{\alpha}\left(\frac{m-n}{2n+1}\cdot\frac{1}{(2n+1)^{\alpha-2}} + \frac{1}{2^{\alpha-2}(\alpha-2)}\right).
\end{aligned}$$

However, one can easily see that this construction is unfeasible in Manhattan Paths, even for small values of n (it violates network flow property of equal incoming and outgoing transmission for vertices). We give example of the construction in the Figure 27.

4.2.5 Our approach: load balancing on vertex diagonals.

In all of the routing schemes we propose in this chapter, we will attempt to perform "load balancing" of paths with respect to transmission through vertices rather than edges. Hence, in a similar

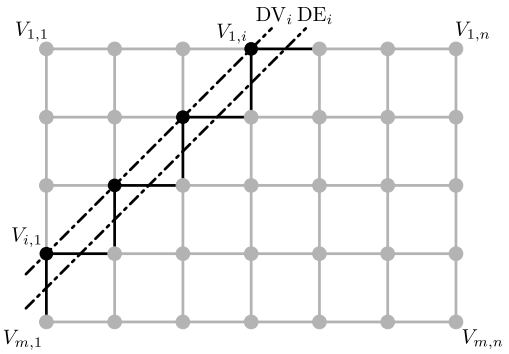

Figure 28: Vertex diagonal DV_i and edge diagonal DE_i

fashion to the notation $R(e)$ for an edge e, we define $R(v)$ as the total transmission size going through a vertex v in routing R.

We introduce the notion of the l-th *vertex diagonal*, denoted as DV_l ($1 \leq l \leq n+m-1$) by splitting the set of vertices according to their distance from the source, as follows (see Fig. 28 for an illustration): $V_{i,j} \in \mathrm{DV}_l$, iff $i+j = l+1$. Likewise, by the l-th *edge diagonal*, denoted DE_l ($1 \leq l \leq n+m-2$), we mean the set of edges connecting vertices from DV_l and DV_{l+1}, namely: $E_{i,j}, E'_{i,j} \in \mathrm{DE}_l$, iff $i+j = l+1$.

It is worth noting that

$$|\mathrm{DV}_l| = \begin{cases} l & \text{if } 1 \leq l \leq n, \\ n & \text{if } n \leq l \leq m, \\ n+m-l & \text{if } m \leq l \leq n+m-1 \end{cases} \tag{5}$$

and

$$|\mathrm{DE}_l| = \begin{cases} 2 \cdot l & \text{if } 1 \leq l < n, \\ 2 \cdot n - 1 & \text{if } n \leq l < m, \\ 2 \cdot (n+m-1-l) & \text{if } m \leq l < n+m-1. \end{cases} \tag{6}$$

We start by observing that the values of $R(v)$ uniquely determine the values of $R(e)$. This property will allow us to design routing schemes simply by setting $R(v)$ for all nodes.

Lemma 4.2.1. *Given arbitrary Manhattan-Paths routing R, we can compute the values of $R(e)$ from the values of $R(v)$.*

Proof. We will look at DE_j for various values of j. We denote $\mathrm{DV}_j = \{v_1, v_2, \ldots\}$ and $\mathrm{DV}_{j+1} = \{u_1, u_2, \ldots\}$, with vertices reverse-ordered by first coordinate. Also, the edges in $\mathrm{DE}_j = \{e_1, \ldots\} \cup \{e'_1, \ldots\}$ are assumed to be reverse-ordered by first coordinate respectively horizontal and vertical edges.

1. Let $1 \leq j < n$, and $1 \leq i \leq j$.

$$R(e_i) = (R(v_1) + \ldots + R(v_i)) - (R(u_1) + \ldots + R(u_i)),$$
$$R(e'_i) = (R(u_1) + \ldots + R(u_i)) - (R(v_1) + \ldots + R(v_{i-1})).$$

2. For $n \leq j < m$, we can perform similar reasoning

$$R(e_i) = (R(v_1) + \ldots + R(v_i)) - (R(u_1) + \ldots + R(u_{i-1})),$$
$$R(e'_i) = (R(u_1) + \ldots + R(u_i)) - (R(v_1) + \ldots + R(v_i)).$$

3. For $m \leq j < n+m$

$$R(e_i) = (R(v_1) + \ldots + R(v_i)) - (R(u_1) + \ldots + R(u_{i-1})),$$
$$R(e'_i) = (R(u_1) + \ldots + R(u_i)) - (R(v_1) + \ldots + R(v_i)).$$

□

Furthermore, there are close connections between the costs measured in terms of nodes and edges. For any Manhattan routing R we have

$$R(E_{i,j-1}) + R(E'_{i-1,j}) = R(V_{i,j}) = R(E_{i,j}) + R(E'_{i,j}).$$

Since the cost function $f(x) = x^\alpha$ is convex, we get

$$R(V_{i,j})^\alpha \geq R(E_{i,j})^\alpha + R(E'_{i,j})^\alpha,$$

what leads to useful inequalities between the costs related to vertices and edges:

$$\sum_{v \in \mathrm{DV}_l} R(v)^\alpha \geq \sum_{e \in DE_l} R(e)^\alpha \quad \text{and} \tag{7}$$
$$\sum_{v \in \mathrm{DV}_l} R(v)^\alpha \geq \sum_{e \in DE_{l-1}} R(e)^\alpha. \tag{8}$$

4.2.6 Routing scheme $\mathcal{C}$ for Max-MP.

We define the routing scheme $\mathcal{C}$ for Max-MP by putting a limit on the transmission going through vertices. Since each diagonal of vertices has a total transmission of exactly K, we set an equal value of transmission for all vertices in the layer:

$$\forall_{v \in \mathrm{DV}_j} \mathcal{C}(v) = \tfrac{K}{|\mathrm{DV}_j|}. \tag{9}$$

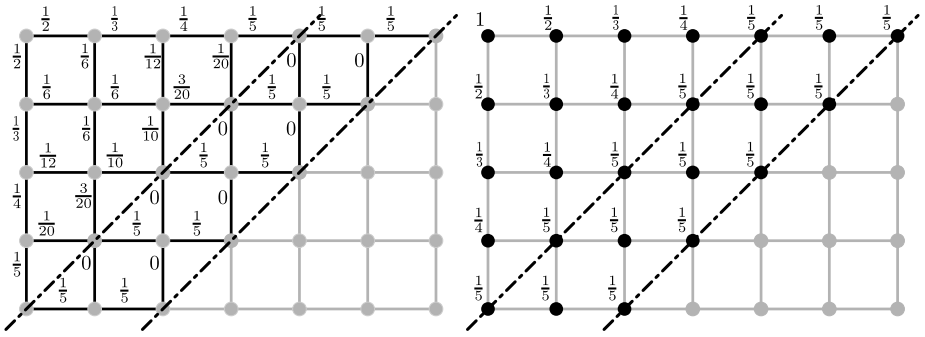

Figure 29: Example of $\mathcal{C}$ routing on a 5×7 grid, with highlighted values of transfer on edges (on the left) and vertices (on the right)

Taking into account equation (5), this gives us the following value of transmission for vertices of the grid:

$$\mathcal{C}\left(V_{i,j}\right) = \begin{cases} \frac{K}{i+j-1} & \text{if } 1 \leq i+j-1 \leq n, \\ \frac{K}{n} & \text{if } n \leq i+j-1 \leq m, \\ \frac{K}{n+m+1-i-j} & \text{otherwise.} \end{cases} \tag{10}$$

To verify that this routing is well-defined, we compute transfers over each edge based on the transfers on vertices. Examples of transfers obtained using this algorithm are shown in Fig. 29.

Algorithm 5 $\mathcal{C}$ routing {for Max-MP, cf. [13]}

Input: A set of arbitrarily splittable requests of total size K in a $m \times n$ grid.
Solution: For each diagonal DE_j of the grid, $1 \leq j < n+m$, set the flow on its successive horizontal edges e_i and vertical edges e'_i, $1 \leq i \leq j$, as follows:

- If $1 \leq j < n$, set

$$\mathcal{C}(e_i) := K\tfrac{i}{j} - K\tfrac{i}{j+1} \quad \text{and} \quad \mathcal{C}(e'_i) := K\tfrac{i}{j+1} - K\tfrac{i-1}{j}.$$

- If $n \leq j < m$, set

$$\mathcal{C}(e_i) := K\tfrac{i}{n} - K\tfrac{i-1}{n} = \tfrac{K}{n} \quad \text{and} \quad \mathcal{C}(e'_i) := 0.$$

- If $m \leq j < n+m$, set

$$\mathcal{C}(e_i) := K\tfrac{i}{n+m-j} - K\tfrac{i-1}{n+m-j-1} \quad \text{and} \quad \mathcal{C}(e'_i) := K\tfrac{i}{n+m-j-1} - K\tfrac{i}{n+m-j}.$$

The scheme $\mathcal{C}$ is a reformulation of the algorithm studied in [13], where it was shown that it admits a constant (depending only on α) approximation ratio for Max-MP.

Theorem 4.2.2 ([13]). $\mathrm{cost}(\mathcal{C}) = \Theta(K^{\alpha}) = \Theta(\mathrm{cost}(\mathrm{OPT}))$.

Proof. To analyze the cost of $\mathcal{C}$, we have:

$$\text{cost}(\mathcal{C}) = \sum_{k=1}^{n+m-2} \sum_{e \in \text{DE}_k} \mathcal{C}(e)^\alpha = \sum_{k=n}^{m-1} \sum_{e \in \text{DE}_k} \mathcal{C}(e)^\alpha + 2 \sum_{k=1}^{n-1} \sum_{e \in \text{DE}_k} \mathcal{C}(e)^\alpha$$

$$\leq (m-n) \cdot n \cdot \left(\frac{K}{n}\right)^\alpha + 2 \sum_{k=2}^{n} \sum_{v \in \text{DV}_k} \mathcal{C}(v)^\alpha = (m-n) \frac{K^\alpha}{n^{\alpha-1}} + 2 \sum_{k=2}^{n} \frac{K^\alpha}{k^{\alpha-1}}$$

$$\leq (m-n) \frac{K^\alpha}{n^{\alpha-1}} + 2 \int_1^\infty \frac{K^\alpha}{x^{\alpha-1}} dx = K^\alpha \cdot \left(\frac{m-n}{n} \frac{1}{n^{\alpha-2}} + \frac{2}{(\alpha-2)}\right) = \Theta(K^\alpha).$$

□

Although such a solution has (up to a constant factor) optimal power cost, it can result in a single request being split into a very large number of paths. Indeed, for a given graph $G = (V,E)$ and any flow f on G, f can be represented as the union of at most $|E|$ weighted paths. It follows that both OPT routing (computed through convex optimization) and $\mathcal{C}$ routing require $O(nm)$ splits per request. We conjecture that those routings require in fact $\Theta(nm)$ requests. In the next section, we will show that it is possible to preserve a constant approximation ratio of the optimal cost, while using a much smaller number of splits, sublinear in the dimensions of the grid.

4.3 Schemes for k-Splittable Routing

In this section, we present three schemes for solving the k-Path Manhattan Routing problem (k-MP). The first two, denoted $\mathcal{F}_k$ and $\mathcal{D}_k$, are designed for uniform sets of requests. As the bound k on the number of allowed paths per request tends to infinity, these approaches will be shown to converge to the performance of schemes OPT and $\mathcal{C}$ for Max-MP, respectively. The third scheme, denoted $\mathcal{A}_k$, is an extension of $\mathcal{D}_k$ which also works for non-uniform sets of requests.

4.3.1 1-splittable routing with uniform requests

Let us start by considering the 1-MP routing policy, meaning that requests cannot be split. We can treat this problem as a discrete version of a continuous Max-MP problem. First, we will consider uniform requests (of equal sizes); without loss of generality, we can assume that the input consists of d requests of size 1, each.

This considered problem can be solved by the flow-based $\mathcal{F}_1$ routing approach presented in Algorithm 6. The obtained solution is optimal, *i.e.* for uniform instances, we have $\text{cost}(\text{OPT}_1) = \text{cost}(\mathcal{F}_1)$. Moreover, using a classical min-cost flow algorithm, a $\mathcal{F}_1$ routing can be found in polynomial time with respect to parameters n, m, and d. It is worth noting, that while designing $\mathcal{F}_1$ algorithm, we did not use either the topology of the graph, or Manhattan Paths restriction. Thus, this approach (based on a folklore knowledge) will work in more general setting.

We will now provide asymptotic bounds on the size of the (optimal) solution to the uniform 1-MP problem. We obtain the lower bound by combining the lower bound for problem Max-MP (formula (4) with $K = d$), with an additional factor resulting from the discrete nature of 1-MP.

Algorithm 6 $\mathcal{F}_1$ routing scheme {optimal solution to uniform 1-MP}

Input: A set of d unsplittable requests of size $s = 1$ in a $m \times n$ grid.
Solution:

1. Construct a multigraph G' such that $V(G') = V(G)$.
2. For every directed edge $e \in E(G)$, add d weighted directed edges to G', having the same endpoints as e, and weights given as: $1^\alpha, 2^\alpha - 1^\alpha, \ldots, d^\alpha - (d-1)^\alpha$.
3. Return the min-cost flow of size d in G', using the two opposite corners of the grid as the source and sink.

Lemma 4.3.1. *For every $R \in$ uniform* 1-MP*:* $\mathrm{cost}(R) = \Omega(d^\alpha) + \Omega(nd)$.

Proof. The bound $\Omega(d^\alpha)$ holds since we cannot get a better solution than the one of Max-MP with the same set of requests. Since each message of size 1 induces a cost at least $n+m-2$, by the convexity of the cost function, we get a total cost of $\Omega(nd)$ for d of them. □

To provide a complementary upper bound on the size of 1-MP routings, we do not analyze the optimal scheme $\mathcal{F}_1$, but instead propose an approximation scheme called $\mathcal{D}_1$ routing, which turns out to be easier to analyze.

We design the $\mathcal{D}_1$ routing through a discretization of the construction of $\mathcal{C}$ routing proposed in the previous section for Max-MP. Similarly to equation (9), we will place limits on the size of the transfer going through vertices. Consider the vertex diagonal DV_p with $1 \le p \le n+m-1$, and let $i = \left|\mathrm{DV}_p\right|$. Suppose that the vertices of DV_p are ordered by decreasing first coordinate, as $\mathrm{DV}_p = \{v_1, \ldots, v_i\}$. Then, for $1 \le j \le i$, we successively set $\mathcal{D}_1(v_j)$ so that at each step, the following condition holds: $\mathcal{D}_1(v_1) + \ldots + \mathcal{D}_1(v_j) = \lfloor d \cdot \frac{j}{i} \rfloor$. This is achieved by setting

$$\mathcal{D}_1(v_j) = \left\lfloor d \cdot \tfrac{j}{i} \right\rfloor - \left\lfloor d \cdot \tfrac{j-1}{i} \right\rfloor. \tag{11}$$

To verify the correctness of this construction, we deduce transfer values over vertical and horizontal edges from values over vertices; a formal implementation of $\mathcal{D}_1$ routing is provided in Algorithm 7. An exemplary comparison of the vertex and edge transfers for $\mathcal{C}$ routing and $\mathcal{D}_1$ routing is shown in Fig. 30.

We start the analysis of the cost of $\mathcal{D}_1$ routing with the following lemma.

Lemma 4.3.2. *Let* DV *be an arbitrary vertex diagonal, and let* $|\mathrm{DV}| = i$*. Then*

$$\sum_{v \in DV} \mathcal{D}_1(v)^\alpha = \begin{cases} i\left(\left(\frac{d}{i}\right)^\alpha + O\left(\left(\frac{d}{i}\right)^{\alpha-2}\right)\right), & \text{for } i < d \\ d, & \text{for } i \ge d. \end{cases}$$

Proof. By equation (11) we get (substituting $d' = d \bmod i$)

$$\sum_{v \in \mathrm{DV}} \mathcal{D}_1(v)^\alpha = \sum_{j=1}^{i} \left(\left\lfloor d \cdot \frac{j}{i} \right\rfloor - \left\lfloor d \cdot \frac{j-1}{i} \right\rfloor \right)^\alpha = \left\lfloor \frac{d}{i} \right\rfloor^\alpha \cdot (i - d') + \left(\left\lfloor \frac{d}{i} \right\rfloor + 1 \right)^\alpha \cdot d'.$$

Algorithm 7 $\mathcal{D}_1$ routing scheme {for uniform 1-MP}

Input: A set of d unsplittable requests of size $s = 1$ in a $m \times n$ grid.
Solution: For each diagonal DE_j of the grid, $1 \leq j < n+m$, set the flow on its successive horizontal edges e_i and vertical edges e'_i, $1 \leq i \leq j$, as follows:

- If $1 \leq j < n$, set:

$$\mathcal{D}_1(e_i) = \left\lfloor d\frac{i}{j} \right\rfloor - \left\lfloor d\frac{i}{j+1} \right\rfloor \quad \text{and} \quad \mathcal{D}_1(e'_i) = \left\lfloor d\frac{i}{j+1} \right\rfloor - \left\lfloor d\frac{i-1}{j} \right\rfloor.$$

- If $n \leq j < m$, set

$$\mathcal{D}_1(e_i) = \left\lfloor d\frac{i}{n} \right\rfloor - \left\lfloor d\frac{i-1}{n} \right\rfloor \quad \text{and} \quad \mathcal{D}_1(e'_i) = 0.$$

- If $m \leq j < n+m$, set

$$\mathcal{D}_1(e_i) = \left\lfloor d\frac{i}{n+m-j} \right\rfloor - \left\lfloor d\frac{i-1}{n+m-j-1} \right\rfloor \quad \text{and} \quad \mathcal{D}_1(e'_i) = \left\lfloor d\frac{i}{n+m-j-1} \right\rfloor - \left\lfloor d\frac{i}{n+m-j} \right\rfloor.$$

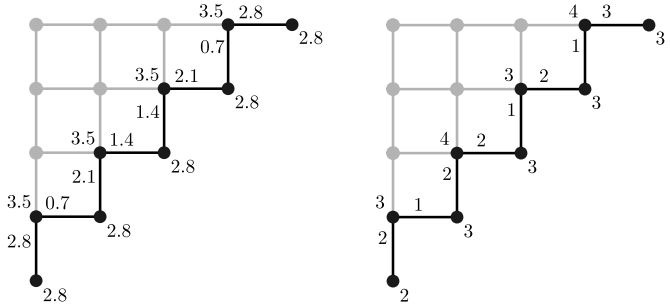

Figure 30: Comparison of transfer values over one diagonal for a $\mathcal{C}$ routing with $K = 14$ (on the left) and a $\mathcal{D}_1$ routing with $d = 14$ (on the right)

We consider two cases:

- ($i < d$) Substituting $x = \frac{d}{i}$, $\lambda = \frac{d}{i} - \left\lfloor \frac{d}{i} \right\rfloor = \frac{d'}{i}$, we obtain

$$\sum_{v \in \mathrm{DV}} \mathcal{D}_1(v)^\alpha = (x-\lambda)^\alpha (1-\lambda)\, i + (x+1-\lambda)^\alpha \lambda i$$

(using Taylor's series theorem, for some $x-\lambda \le u_1 \le x \le u_2 \le x+1-\lambda$)

$$= \left(x^\alpha - \lambda\alpha x^{\alpha-1} + \frac{\lambda^2}{2}\alpha(\alpha-1)u_1^{\alpha-2}\right)(1-\lambda)i +$$
$$+ \left(x^\alpha + (1-\lambda)\alpha x^{\alpha-1} + \frac{(1-\lambda)^2}{2}\alpha(\alpha-1)u_2^{\alpha-2}\right)\lambda i$$

(since $u_1 \le x$ and $u_2 \le x+1-\lambda \le 2x$)

$$\le ix^\alpha + i\alpha(\alpha-1)\left(\frac{\lambda^2}{2}(1-\lambda) + \frac{(1-\lambda)^2}{2}\lambda 2^\alpha\right)x^{\alpha-2}$$
$$= i\left(x^\alpha + O\left(x^{\alpha-2}\right)\right) = i\left(\left(\frac{d}{i}\right)^\alpha + O\left(\left(\frac{d}{i}\right)^{\alpha-2}\right)\right).$$

- ($i \ge d$) The sum simplifies to

$$\sum_{v\in \mathrm{DV}} \mathcal{D}_1(v)^\alpha = 0\cdot(i-(d \bmod i)) + 1\cdot(d \bmod i) = d.$$

□

Using the above lemma, we compute the cost of a $\mathcal{D}_1$ routing as $\mathrm{cost}(\mathcal{D}_1) = \Theta(d^\alpha) + \Theta(nd)$. By Lemma 4.3.1, this cost is asymptotically the best possible for 1-MP.

Theorem 4.3.3. *For a uniform set of d requests (with total size $K = d$):*

$$\mathrm{cost}(\mathcal{D}_1) = \Theta(d^\alpha) + \Theta(nd)$$

and

$$\mathrm{cost}(\mathcal{F}_1) = \Theta(d^\alpha) + \Theta(nd).$$

Proof. The lower bound follows from Lemma 4.3.1. We observe that $\mathrm{cost}(\mathcal{F}_1) \le \mathrm{cost}(\mathcal{D}_1)$. Now, to provide an upper bound on cost of the $\mathcal{D}_1$ routing scheme, we will consider two cases:

- ($d \ge n$) Then, using Lemma 4.3.2 and inequalities (7-8), we have

$$\mathrm{cost}(\mathcal{D}_1) = \sum_{i=1}^{n+m-2}\sum_{e\in \mathrm{DE}_i}\mathcal{D}_1(e)^\alpha = \sum_{i=n}^{m-1}\sum_{e\in \mathrm{DE}_i}\mathcal{D}_1(e)^\alpha + 2\sum_{i=1}^{n-1}\sum_{e\in \mathrm{DE}_i}\mathcal{D}_1(e)^\alpha$$
$$\le (m-n)\sum_{v\in \mathrm{DV}_n}\mathcal{D}_1(v)^\alpha + 2\sum_{i=2}^{n}\sum_{v\in \mathrm{DV}_i}\mathcal{D}_1(v)^\alpha$$
$$= (m-n)n\left(\left(\frac{d}{n}\right)^\alpha + O\left(\left(\frac{d}{n}\right)^{\alpha-2}\right)\right) + 2\sum_{i=2}^{n} i\left(\left(\frac{d}{i}\right)^\alpha + O\left(\left(\frac{d}{i}\right)^{\alpha-2}\right)\right).$$

Moreover, we have

$$(m-n)\cdot n\cdot\left(\frac{d}{n}\right)^{\alpha}+2\sum_{i=2}^{n} i\left(\frac{d}{i}\right)^{\alpha}=(m-n)\frac{d^{\alpha}}{n^{\alpha-1}}+2\sum_{i=2}^{n}\frac{d^{\alpha}}{i^{\alpha-1}}$$
$$\leq (m-n)\frac{d^{\alpha}}{n^{\alpha-1}}+2\int_{1}^{n}\frac{d^{\alpha}}{x^{\alpha-1}}dx$$
$$=d^{\alpha}\cdot\left(\frac{m-n}{n}\frac{1}{n^{\alpha-2}}+\frac{2}{(\alpha-2)}\left(1-\frac{1}{n^{\alpha-2}}\right)\right)=\Theta(d^{\alpha}).$$

It follows that

$$\begin{aligned}\operatorname{cost}(\mathcal{D}_1) &\leq \Theta\left(d^{\alpha}\right)+(m-n)\,n\cdot O\left(\left(\frac{d}{n}\right)^{\alpha-2}\right)+O\left(\sum_{i=2}^{n} i\left(\frac{d}{i}\right)^{\alpha-2}\right)\\ &=\Theta\left(d^{\alpha}\right)+O\left(d^{\alpha-2}n^{4-\alpha}\right)+O\left(n\cdot\max\left(d^{\alpha-2},d^{\alpha-2}n^{3-\alpha}\right)\right)\\ &=\Theta\left(d^{\alpha}\right)+O\left(d^{\alpha-2}n^{4-\alpha}\right)+O\left(d^{\alpha-2}n\right)+O\left(d^{\alpha-2}n^{4-\alpha}\right)=\Theta\left(d^{\alpha}\right),\end{aligned}$$

where the last step holds since $1\leq n\leq d$ and $\alpha>2$.

- ($d<n$) In this case, we have:

$$\operatorname{cost}(\mathcal{D}_1)=\sum_{i=1}^{n+m-2}\sum_{e\in \mathrm{DE}_i}\mathcal{D}_1(e)^{\alpha}=\sum_{i=d}^{n+m-d+1}\sum_{e\in \mathrm{DE}_i}\mathcal{D}_1(e)^{\alpha}+2\sum_{i=1}^{d-1}\sum_{e\in \mathrm{DE}_i}\mathcal{D}_1(e)^{\alpha}\leq$$
$$\leq (n+m-2d+2)\,d+2\sum_{i=2}^{d}\sum_{v\in DV_i}\mathcal{D}_1(v)^{\alpha}=O(nd)+2\sum_{i=2}^{d} i\left(\left(\frac{d}{i}\right)^{\alpha}+O\left(\left(\frac{d}{i}\right)^{\alpha-2}\right)\right)=$$
$$=O(nd)+\Theta\left(d^{\alpha}\right)+O\left(d\max\left(d^{\alpha-2},d\right)\right)=O(nd)+\Theta\left(d^{\alpha}\right).$$

□

4.3.2 *k*-splittable routing with uniform requests

We now extend the results of the previous section to the case of k-MP uniform routing. We will consider uniform sets of d requests of total size K, *i.e.*, of size K/d each. A natural generalization of $\mathcal{D}_1$ routing, called $\mathcal{D}_k$ routing, is presented in Algorithm 8.

Since in a $\mathcal{D}_k$ routing, we split the transmission of each request equally along its k paths, the cost of such a routing is the same as that of a $\mathcal{D}_1$ routing on the extended set of kd requests of size $\frac{K}{kd}$ each. Hence, the following result follows directly from Theorem 4.3.3 by a scaling argument: $\operatorname{cost}(\mathcal{D}_k)=\Theta\left(K^{\alpha}\right)+\Theta(K^{\alpha}\frac{n}{(kd)^{\alpha-1}})$. Next, we show that although $\mathcal{D}_k$ only splits requests into paths

Algorithm 8 $\mathcal{D}_k$ routing scheme {for uniform k-MP}

Input: A set of d k-splittable requests, of size K/d each, in a $m \times n$ grid.
Solution: Split each of the requests into k smaller ones, each of size $\frac{K}{kd}$. Return the $\mathcal{D}_1$ routing of this new set of requests.

of equal weight, one cannot achieve a better asymptotic result by using unequal splits, *i.e.* for any $R \in$ uniform k-MP : $\text{cost}(R) = \Omega(K^\alpha) + \Omega(K^\alpha \frac{n}{(kd)^{\alpha-1}})$. Combining these results, we obtain the following theorem, stating the optimality of $\mathcal{D}_k$ in the class of k-splittable routings.

Theorem 4.3.4. *For a uniform set of d requests with total size K,* $\text{cost}(\mathcal{D}_k) = \Theta(K^\alpha) + \Theta\left(K^\alpha \frac{n}{(kd)^{\alpha-1}}\right)$, $\text{cost}(\text{OPT}_k) = \Theta(K^\alpha) + \Theta\left(K^\alpha \frac{n}{(kd)^{\alpha-1}}\right)$, *where OPT_k denotes the optimal cost solution of the considered set of requests for k-MP.*

Proof. We have already established that for $R \in$ Max-MP, $\text{cost}(R)$ is in $\Omega(K^\alpha)$, which also holds for k-MP. For the second part of the lower bound, we use the convexity of the cost function and the fact that on any diagonal, the stream of the routing of the d requests can be split into at most kd distinct paths, *i.e.*

$$\sum_{e \in \text{DE}} R(e)^\alpha \geq (kd)\left(\frac{K}{kd}\right)^\alpha.$$

Thus

$$\text{cost}(R) = \sum_{i=1}^{n+m-2} \sum_{e \in \text{DE}_i} R(e)^\alpha \geq (n+m-2)(kd)\left(\frac{K}{kd}\right)^\alpha = \Omega\left(K^\alpha \frac{n}{(kd)^{\alpha-1}}\right).$$

□

Combining the bound on $\text{cost}(\text{OPT}_k)$ in the above Theorem with the bound on $\text{cost}(\text{OPT})$ in Theorem 4.2.2 for Max-MP routing, we obtain our main result: the threshold value of k for which imposing a limit of k into which each request can be split does not affect the asymptotics of power cost.

Theorem 4.3.5. *For uniform requests, imposing a routing policy with a split limit of $k = \Theta\left(\frac{1}{d} \cdot n^{\frac{1}{\alpha-1}}\right)$ does not affect the power cost, i.e.* $\text{cost}(\text{OPT}_k) = \Theta(\text{cost}(\text{OPT}))$, *and $\frac{1}{\alpha-1}$ is the smallest possible exponential.*

Proof. To prove that $\Theta\left(\frac{1}{d} \cdot n^{\frac{1}{\alpha-1}}\right)$ is enough, we set $k = \Omega(\frac{1}{d} \cdot n^{\frac{1}{\alpha-1}})$ and by Theorem 4.3.4 we obtain:

$$\text{cost}(\text{OPT}_k) = \Theta(K^\alpha) + \Theta\left(K^\alpha \frac{n}{(kd)^{\alpha-1}}\right) = \Theta(K^\alpha) + o(K^\alpha) = \Theta(K^\alpha).$$

To prove that $\Theta\left(\frac{1}{d} \cdot n^{\frac{1}{\alpha-1}}\right)$ is required, we observe that with $k = o(\frac{1}{d} \cdot n^{\frac{1}{\alpha-1}})$, the lower bound on cost becomes:

$$\text{cost}(\text{OPT}_k) = \Omega(K^\alpha) + \Omega\left(K^\alpha \frac{n}{o(n)}\right) = \omega(K^\alpha).$$

□

We end this subsection with a remark on the asymptotic behavior of the considered routing schemes for uniform instances, when $k \to +\infty$. Taking into account Theorems 4.2.2 and 4.3.4, we obtain the following proposition.

Proposition 4.3.6. *For a grid of fixed dimension:* $\lim_{k\to+\infty} \text{cost}(\mathcal{D}_k) = \text{cost}(\mathcal{C})$.

Proof. Let us denote by $\mathcal{C}'$ an assignment of transfer sizes to edges of G obtained by rounding up the value of the transfer size in an $\mathcal{C}$ routing to the nearest integer multiple of $\frac{2K}{kd}$, *i.e.*, for all edges e, $\mathcal{C}'(e) = \frac{2K}{kd}\lceil \frac{kd}{2K}\text{OPT}(e)\rceil$. By the properties of the rounding operation used in the design of $\mathcal{D}_k$, the assignment $\mathcal{C}'$ dominates the cost of the $\mathcal{D}_k$ routing for the considered instance. Thus, we have,

$$\text{cost}(\mathcal{D}_k) \leq \text{cost}(\mathcal{C}') \leq \text{cost}(\mathcal{C}) + O\left(nm \cdot \left(\frac{K}{kd}\right)^{\alpha-1}\right).$$

Taking the limit when $k \to +\infty$, the claim follows. □

Since in general, $\text{cost}(\mathcal{C}) > \text{cost}(\text{OPT})$, it is natural to ask for a different routing scheme for k-MP with improved limit behavior. A natural candidate is $\mathcal{F}_k$ routing, obtained by a natural generalization of $\mathcal{F}_1$ routing, as given by Algorithm 9.

Algorithm 9 $\mathcal{F}_k$ routing scheme {for uniform k-MP}

Input: A set of d k-splittable requests, of size K/d each, in a $m \times n$ grid.
Solution: Split each of the requests into k smaller ones, each of size $\frac{K}{kd}$. Return the $\mathcal{F}_1$ routing of the new set of requests.

This algorithm turns out to by asymptotically optimal when $k \to +\infty$.

Proposition 4.3.7. *For a grid of fixed dimension,* $\lim_{k\to+\infty} \text{cost}(\mathcal{F}_k) = \text{cost}(\text{OPT})$.

Proof. Let us denote by OPT' an assignment of transfer sizes to edges of G obtained by rounding up the value of the transfer size in an OPT routing to the nearest integer multiple of $\frac{K}{kd}$, *i.e.*, for all edges e, $\text{OPT}'(e) = \frac{K}{kd}\lceil \frac{kd}{K}\text{OPT}(e)\rceil$. By the properties of the min-cost flow used in the design of $\mathcal{F}_k$, the assignment OPT' dominates the cost of the $\mathcal{F}_k$ routing for the considered instance. Thus, we have,

$$\text{cost}(\mathcal{F}_k) \leq \text{cost}(\text{OPT}') \leq \text{cost}(\text{OPT}) + O\left(nm \cdot \left(\frac{K}{kd}\right)^{\alpha-1}\right).$$

Taking the limit when $k \to +\infty$, the claim follows. □

4.3.3 k-splittable routing with non-uniform requests

We close our considerations with a discussion of the general (non-uniform) case, where no assumptions are made about the sizes of the routed requests. We first observe that the considered problem is computationally hard.

Theorem 4.3.8. *The following decision version of non-uniform* 1-MP *routing is (weakly-)NP-complete: "Given* $(n,m,K=(K_1,\ldots,K_i),C,\alpha)$*, decide if it is possible to perform 1-MP routing with cost* $\leq C$*."*

Proof. The problem is obviously in NP, as one can easily calculate the cost of any solution in polynomial time.

The proof proceeds by reduction from PARTITION PROBLEM [33]: "Given a set I of integers, decide if it is possible to partition I into subsets I_1 and I_2 such that $\sum I_1 = \sum I_2$."

For such an instance, we select the instance of 1-MP routing as follows: $n=2$, $m=2$, $K=I$ and $C=4\left(\frac{1}{2}\sum_{s\in I} s\right)^{\alpha}$.

Observe that by identifying the sets of requests routed along the 2 different paths of the grid with I_1 and I_2, we have:

$$\operatorname{cost}(I_1,I_2) = 2\left(\sum_{s\in I_1} s\right)^{\alpha} + 2\left(\sum_{s\in I_2} s\right)^{\alpha} \geq 4\left(\frac{1}{2}\sum_{s\in I} s\right)^{\alpha} = C$$

and equality in the bound is achieved only if $\sum_{s\in I_1} s = \sum_{s\in I_2} s$. □

Despite the hardness of the considered problem, one can try to look for approximate solutions. Note that applying $\mathcal{D}_k$ routing naively to a set of non-uniform requests could lead to excessive additional cost. However, by applying a careful modification of $\mathcal{D}_k$ routing, called the $\mathcal{A}_k$ routing scheme (Algorithm 10), we obtain a good tool for routing non-uniform requests on the grid.

For example, in the case of unsplittable requests (1-MP) of sizes $\{\underbrace{1,\ldots,1}_{t},\underbrace{\varepsilon,\ldots,\varepsilon}_{n\cdot t}\}$, $\mathcal{D}_1$ treats each request equally, and so it could lead to grouping all t large requests into one path, giving:

$$\text{cost} = \Theta(n\cdot t^{\alpha})$$

while the following cost is achievable by routing t requests of size $1+n\cdot\varepsilon\approx 1$:

$$\text{cost} = \Theta(n\cdot t + t^{\alpha})$$

However, the Algorithm 10 gives an approximate solution for non-uniform k-MP routing.

Theorem 4.3.9. *For non-uniform requests,* $\mathcal{A}_k$ *finds a solution to* k-MP *whose cost is within a constant factor of the optimum k-splittable routing:* $\operatorname{cost}(\mathcal{A}_k) \leq \left(2^{4\alpha-2}\right)\cdot\operatorname{cost}(\textsc{Opt}_k)$.

Proof. It is enough to prove that for each edge diagonal DE, the cost induced by $\mathcal{A}_k$ on this diagonal is bounded by a constant with relation to the possible cost of the optimal solution $\textsc{Opt}_k \in k$-MP on this diagonal.

Algorithm 10 $\mathcal{A}_k$ routing scheme {for non-uniform k-MP}

Input: A set of d k-splittable requests, of given sizes $S = (s_1, s_2, \ldots, s_d)$ (with $1 = s_1 \leq s_2 \leq \ldots \leq s_d$), in a $m \times n$ grid.
Solution:

1. Partition the set of request sizes into the union of disjoint subsets, $S = S_0 \cup S_1 \cup \ldots$, such that $\forall_{s \in S_i}\ 2^i \leq s < 2^{i+1}$.
2. For all non-empty sets S_i:
   - Find a $\mathcal{D}_k$ routing for the uniform instance consisting of $|S_i|$ requests of size 2^{i+1} each.
   - For all $1 \leq j \leq |S_i|$, route the j-th input request belonging to S_i using the paths assigned to the j-th request in the corresponding $\mathcal{D}_k$ routing.

We keep the notation $c_i = |S_i|$, and recall that DV denotes the vertex diagonal adjacent to DE. Also, let $p = |DV|$, and let $\mathcal{D}[S_i]$ refer to the discrete $\mathcal{D}_1$ routing scheme applied inside S_i. We have

$$\text{cost}(\text{DE}, \mathcal{A}_k) = \sum_{e \in \text{DE}} \mathcal{A}_k(e)^\alpha \leq \sum_{v \in \text{DV}} \mathcal{A}_k(v)^\alpha$$

(taking into account that for $s \in S_i$ we have $s < 2 \cdot 2^i$)

$$\leq \sum_{v \in \text{DV}} \left(\sum_i \mathcal{D}[c_i](v) \cdot \frac{2 \cdot 2^i}{k} \right)^\alpha = 2^\alpha \sum_{v \in \text{DV}} \left(\sum_{i:\ c_i \cdot k \leq p} \left(\mathcal{D}[c_i](v) \cdot \frac{2^i}{k} \right) + \sum_{i:\ c_i \cdot k > p} \left(\mathcal{D}[c_i](v) \cdot \frac{2^i}{k} \right) \right)^\alpha$$

(taking into account that $\forall_{a,b \geq 0} (a+b)^\alpha \leq 2^{\alpha-1}(a^\alpha + b^\alpha)$)

$$\leq 2^{2\alpha-1} \sum_{v \in \text{DV}} \left(\left(\sum_{i:\ c_i \cdot k \leq p} \mathcal{D}[c_i](v) \cdot \frac{2^i}{k} \right)^\alpha + \left(\sum_{i:\ c_i \cdot k > p} \mathcal{D}[c_i](v) \cdot \frac{2^i}{k} \right)^\alpha \right).$$

Observe that $\mathcal{D}[c_i](v) \leq \lceil \frac{c_i \cdot k}{p} \rceil$.
Consequently, for $c_i \cdot k \leq p$, we have $\mathcal{D}[c_i](v) \in \{0, 1\}$, and since $2^d + 2^{d-1} + \ldots < 2 \cdot 2^d$:

$$\sum_{v \in \text{DV}} \left(\sum_{i:\ c_i \cdot k \leq p} \mathcal{D}[c_i](v) \cdot \frac{2^i}{k} \right)^\alpha \leq \sum_{v \in \text{DV}} \left(2 \cdot \max_{i:\ c_i k \leq p} \left\{ \mathcal{D}[c_i](v) \cdot \frac{2^i}{k} \right\} \right)^\alpha$$

$$\leq 2^\alpha \sum_{v \in \text{DV}} \sum_{i:\ c_i \cdot k \leq p} \left(\mathcal{D}[c_i](v) \cdot \frac{2^i}{k} \right)^\alpha = 2^\alpha \sum_{i:\ c_i \cdot k \leq p} k \cdot c_i \left(\frac{2^i}{k} \right)^\alpha$$

For $c_i k > p$, we have $\mathcal{D}[c_i](v) \leq \lceil \frac{c_i \cdot k}{p} \rceil \leq 2\frac{c_i k}{p}$:

$$\sum_{v \in \text{DV}} \left(\sum_{i:\ c_i \cdot k > p} \mathcal{D}[c_i](v) \cdot \frac{2^i}{k} \right)^\alpha \leq \sum_{v \in \text{DV}} \left(\sum_{i:\ c_i k > p} 2 \cdot \frac{c_i \cdot k}{p} \cdot \frac{2^i}{k} \right)^\alpha = 2^\alpha \cdot \left(\frac{1}{p} \sum_{i:\ c_i k > p} c_i \cdot 2^i \right)^\alpha \cdot p.$$

Adding the two sums, we can write in general:

$$\operatorname{cost}(\mathrm{DE},\mathcal{A}_k) \le 2^{3\alpha-1}\cdot\left(\sum_{i:\ c_i k\le p} c_i\left(\frac{2^i}{k}\right)^{\alpha} + p\cdot\left(\frac{1}{p}\sum_{i:\ c_i k>p} c_i\cdot 2^i\right)^{\alpha}\right)$$

Now we proceed to bound the cost induced on diagonal DE by OPT_k routing. By $\mathrm{OPT}_k[S_i]$, we will denote OPT_k restricted to requests from S_i, and by $\mathrm{OPT}_k[s]$ we will understand OPT_k restricted to request s. We have

$$\begin{aligned}
&\operatorname{cost}(\mathrm{DE},\mathrm{OPT}_k) = \sum_{e\in\mathrm{DE}} \mathrm{OPT}_k(e)^{\alpha} \ge \frac{1}{2^{\alpha-1}}\sum_{v\in\mathrm{DV}} \mathrm{OPT}_k(v)^{\alpha}\\
&= \frac{1}{2^{\alpha-1}}\sum_{v\in\mathrm{DV}}\left(\sum_{i:\ c_i k\le p}\mathrm{OPT}_k[S_i](v) + \sum_{i:\ c_i k>p}\mathrm{OPT}_k[S_i](v)\right)^{\alpha}\\
&\ge \frac{1}{2^{\alpha-1}}\sum_{v\in\mathrm{DV}}\left(\left(\sum_{i:\ c_i k\le p}\mathrm{OPT}_k[S_i](v)\right)^{\alpha} + \left(\sum_{i:\ c_i k>p}\mathrm{OPT}_k[S_i](v)\right)^{\alpha}\right).
\end{aligned}$$

We put bounds on both parts of sum:

$$\sum_{v\in\mathrm{DV}}\left(\sum_{i:\ c_i k\le p}\mathrm{OPT}_k[S_i](v)\right)^{\alpha} = \sum_{v\in\mathrm{DV}}\left(\sum_{i:\ c_i k\le p}\sum_{s\in S_i}\mathrm{OPT}_k[s](v)\right)^{\alpha} \ge \sum_{i:\ c_i k\le p}\sum_{s\in S_i}\sum_{v\in\mathrm{DV}}\mathrm{OPT}_k[s](v)^{\alpha}$$

(observing that $\sum_{v\in\mathrm{DV}}\mathrm{OPT}_k[s](v) = s$, and s can be split into at most k parts)

$$\ge \sum_{i:\ c_i k\le p}\sum_{s\in S_i} k\cdot\left(\frac{s}{k}\right)^{\alpha} \ge \sum_{i:\ c_i k\le p} c_i k\left(\frac{2^i}{k}\right)^{\alpha}$$

In the second part, we obtain:

$$\begin{aligned}
&\sum_{v\in\mathrm{DV}}\left(\sum_{i:\ c_i k>p}\mathrm{OPT}_k[S_i](v)\right)^{\alpha} \ge p\cdot\left(\frac{1}{p}\sum_{v\in\mathrm{DV}}\sum_{i:\ c_i k>p}\mathrm{OPT}_k[S_i](v)\right)^{\alpha} =\\
&= p\cdot\left(\frac{1}{p}\sum_{i:\ c_i k>p}\sum_{s\in S_i} s\right)^{\alpha} \ge p\cdot\left(\frac{1}{p}\sum_{i:\ c_i k>p} c_i\cdot 2^i\right)^{\alpha}.
\end{aligned}$$

Merging both results, we get

$$\operatorname{cost}(\mathrm{DE},\mathrm{OPT}_k) \ge \frac{1}{2^{\alpha-1}}\left(\sum_{i:\ c_i k\le p} c_i k\left(\frac{2^i}{k}\right)^{\alpha} + p\cdot\left(\frac{1}{p}\sum_{i:\ c_i k>p} c_i\cdot 2^i\right)^{\alpha}\right).$$

Combining the lower-bound on the cost of OPT_k and the upper bound on the cost of $\mathcal{A}_k$, we finally obtain

$$\operatorname{cost}(\mathrm{DE},\mathcal{A}_k) \le 2^{4\alpha-2}\operatorname{cost}(\mathrm{DE},\mathrm{OPT}_k),$$

which proves that $\mathcal{A}_k$ is a $(2^{4\alpha-2})$-approximation algorithm for non-uniform k-MP. □

We end this section with a similar threshold theorem as Theorem 4.3.5 for the uniform case, obtaining bounds on value of k for which a split limit of k does not affects the asymptotics of the routing cost. However, in this case, the threshold depends on the structure of the set of requests, hence we only provide lower and upper bounds.

Theorem 4.3.10. *For non-uniform requests, imposing a routing policy with a split limit of k*

1. *always increases the asymptotic power cost (i.e.* $\text{cost}(\text{OPT}_k) = \omega(\text{cost}(\text{OPT}))$*) if* $k = o\left(\frac{1}{d} \cdot n^{\frac{1}{\alpha-1}}\right)$.

2. *does not affect the asymptotic power cost (i.e.* $\text{cost}(\text{OPT}_k) = \Theta(\text{cost}(\text{OPT}))$*) as long as* $k = \Omega\left(n^{\frac{1}{\alpha-1}}\right)$,

Proof.

1. The optimal-cost routing for a set of d requests of total size K, each of which can be split into k paths, cannot be better than the optimal-cost routing on a single request of size K, which can be split into kd parts. The latter routing problem belongs to uniform $(k \cdot d)$-MP and so, using Theorem 4.3.4, we can write

$$\text{cost}(\text{OPT}_k) = \Omega\left(K^\alpha + K^\alpha \frac{n}{(kd)^{\alpha-1}}\right) = \Omega\left(K^\alpha + K^\alpha \frac{n}{o(n)}\right) = \omega(K^\alpha) = \omega(\text{cost}(\text{OPT})).$$

2. Only the upper bound needs to be shown. Observe that the cost of an optimal k-MP routing cannot decrease if we replace a pair of requests of size s_1, s_2 by a single request of size $s_1 + s_2$. By iterating the argument, we can upper-bound the value of OPT_k for a given instance of d requests of total size K by the value of OPT_k for an instance consisting of a single request of size K. Once again, the claim follows by an application of Theorem 4.3.4.

□

4.4 Experimental results

In this section we provide the results of experimental evaluation of the algorithms presented in the previous section. We analyze the effect of n, k and α on the efficiency of solutions found for k-MP routing of instances with uniform (identical-size) requests. Throughout the section, we choose the number of requests as $d = 1$ (for uniform instances, other values result only in a scaling factor for k in k-MP, and do not affect Max-MP).

We focus on the approximation ratio, looking at the cost of the routing obtained using the two schemes designed for uniform k-MP ($\mathcal{D}_k$, $\mathcal{F}_k$), relative to the cost of the optimal solution OPT to Max-MP, which is treated as the reference solution. In some graphs, we also provide the cost of the sub-optimal Max-MP routing $\mathcal{C}$ as an additional reference.

We recall that the cost of the optimal solution to Max-MP, $\text{cost}(\text{OPT}_k)$, is bounded from below by $\text{cost}(\text{OPT})$, and from above by both $\text{cost}(\mathcal{D}_k)$ and $\text{cost}(\mathcal{F}_k)$.

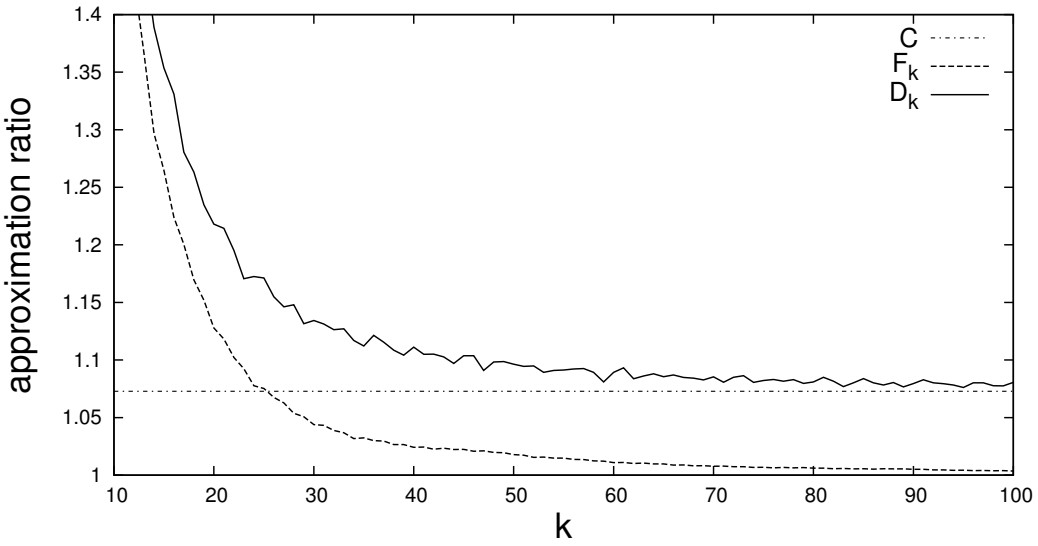

Figure 31: Effect of k on the cost of $\mathcal{F}_k$ and $\mathcal{D}_k$ routing for a 30×30 grid.

The implementation and tests were implemented in GNU C++. The min-cost flow subroutines were implemented using the standard cycle-canceling method [36].The results of the tests are deterministic and fully reproducible, independent of the test environment and the details of the implementation of the flow algorithms.

4.4.1 Impact of k on the routing cost.

We start by studying the approximation ratio of algorithms $\mathcal{F}_k$ and $\mathcal{D}_k$ for increasing values of k, the allowed number of splits of each requests. In the first plot (Fig. 31), we fix the dimensions of the grid $n, m = 30$, model power cost exponent $\alpha = 2.5$, plotting the values of $\text{cost}(\mathcal{F}_k)/\text{cost}(\text{OPT})$ and $\text{cost}(\mathcal{D}_k)/\text{cost}(\text{OPT})$ for k in the range $k \in [10, 100]$. For reference, we also provide the approximation ratio of $\mathcal{C}$ routing for the studied instance.

We observe that, as predicted by theory (Theorems 4.3.6 and 4.3.7), $\lim_{k\to+\infty} \text{cost}(\mathcal{F}_k) = \text{cost}(\text{OPT})$ and $\lim_{k\to+\infty} \text{cost}(\mathcal{D}_k) = \text{cost}(\mathcal{C})$, and the respective costs converge to their limits quickly, reaching a point 10% over the respective limit already for $k < n$. In general, the convergence need not be monotone for either of the approximation algorithms, since partitioning a request into $k+1$ equally-weighted paths may give worse results than partitioning it into k equally-weighted paths.

In our second plot (Fig. 32), we present more compelling evidence of the relation $k = \Theta\left(n^{1/(\alpha-1)}\right)$ for the threshold split value resulting in asymptotically-optimal cost, derived theoretically as Theorem 4.3.5. Once again, we choose model parameter $\alpha = 2.5$. In the experiment, we consider square grids of increasing size in the range $n = m \in [10, 120]$, testing three different relations between n and k ($k = \left\lfloor 2n^{1/2} \right\rfloor$, $k = \left\lfloor \frac{3}{2} n^{2/3} \right\rfloor$, $k = n$). For each of these relations, we plot the approximation ratios $\text{cost}(\mathcal{F}_k)/\text{cost}(\text{OPT})$ and $\text{cost}(\mathcal{D}_k)/\text{cost}(\text{OPT})$. Based on the plot, we can presume that:

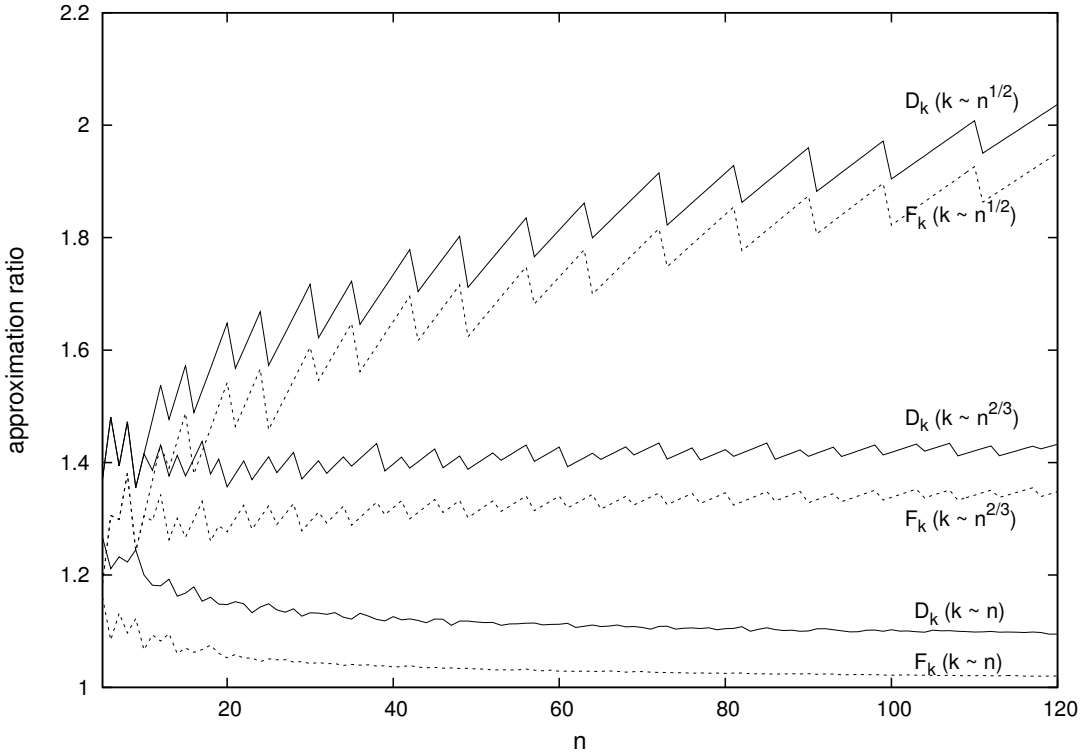

Figure 32: Approximation ratio for $\mathcal{F}_k$ and $\mathcal{D}_k$ routing schemes in a $n \times n$ grid. We considered split parameter $k \sim n^{\beta}$, for β greater, equal and smaller than $1/(\alpha-1)$, $[\alpha = 2.5]$.

- For the relation $k \sim n^{1/2}$, we have asymptotically:

$$\text{cost}(\mathcal{F}_k)/\text{cost}(\text{OPT}) \to +\infty, \qquad \text{cost}(\mathcal{D}_k)/\text{cost}(\text{OPT}) \to +\infty.$$

- For the relation $k \sim n^{2/3}$, we have asymptotically:

$$\text{cost}(\mathcal{F}_k)/\text{cost}(\text{OPT}) \to const, \qquad \text{cost}(\mathcal{D}_k)/\text{cost}(\text{OPT}) \to const.$$

- For the relation $k \sim n$, we have asymptotically:

$$\text{cost}(\mathcal{F}_k)/\text{cost}(\text{OPT}) \to 1, \qquad \text{cost}(\mathcal{D}_k)/\text{cost}(\text{OPT}) \to const.$$

We remark that the relation $k \sim n^{2/3}$ precisely corresponds to the threshold exponent $1/(\alpha-1) = 2/3$ for the considered value of α. Thus, the limit behavior of all the algorithms is consistent with the theory derived in the previous section. We note that the cost achieved by both $\mathcal{F}_k$ routing and $\mathcal{D}_k$ routing is highly satisfactory, and that the performance of $\mathcal{F}_k$ routing shows to be better than of $\mathcal{D}_k$ in all of performed tests.

4.4.2 Effect of power exponent α.

In auxiliary experiments, we studied the effect of the power exponent α (which is a constant of the model) on the required threshold value of split parameter k. We tested the rate of convergence of the approximation ratio $\text{cost}(\mathcal{F}_k)/\text{cost}(\text{OPT})$ to 1 in a grid of dimensions $n = m = 30$ for three different values of the power exponent, $\alpha \in \{2.5, 3, 3.5\}$. It was observed that the convergence is faster for larger values of α. This is consistent with the theoretical threshold, $k = \Theta\left(n^{1/(\alpha-1)}\right)$, whose growth rate decreases with the increase of α. (Results are depicted in Figure 33.)

4.5 Conclusion

The contribution of our study is twofold. On the one hand, we advance the theory of splitting of requests in Manhattan routing on the grid, and point out that in practice, only a relatively small number of splits per request will be beneficial from a power-cost perspective. On the other hand, we propose efficient approximation schemes for such a k-path routing problem. Experimental evidence corroborates the theoretical results, showing that the designed algorithms lead to routings with a cost which is, in practice, even superior to that resulting from our theoretical bounds.

In future work, it would be beneficial to improve the constant bounds on the approximation ratios of our algorithms, establishing more tightly their dependence on the power exponent α.

Another promising direction of study would consist in extending the model to be more general. We have put several limitations on grid topology, requests, routing policies and cost function, in order to simplify the problem. However, lifting or weakening some of the restrictions would make the model more realistic. One of possible ways of extending is allowing routing requests between multiple sources and targets on the grid. In such a study, we could try to repeat the framework

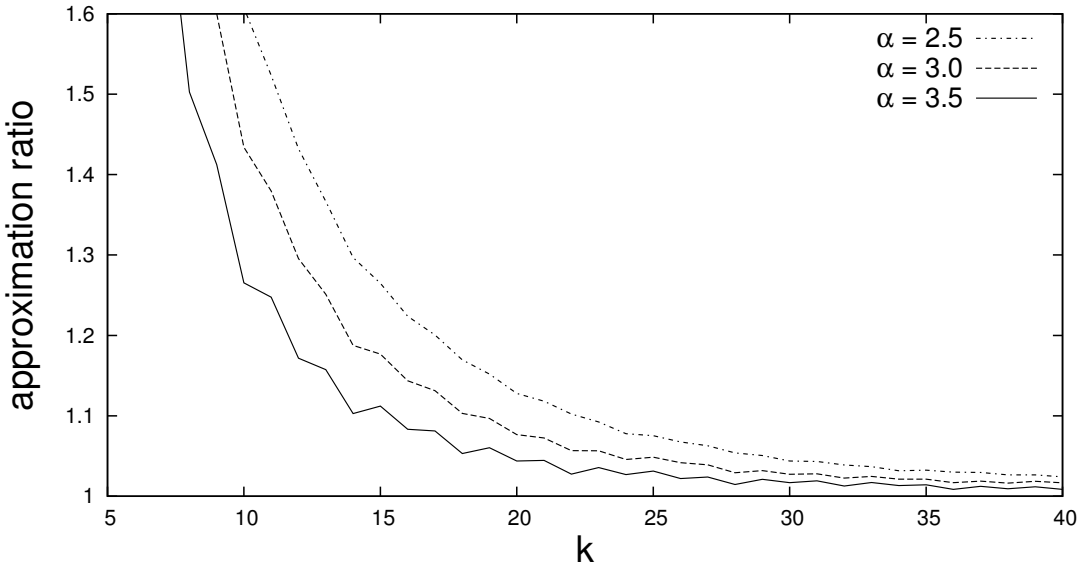

Figure 33: Effect of α on $\text{cost}(\mathcal{F}_k)/\text{cost}(\text{OPT})$, working on a 30×30 grid.

used here, that is, start with continuous solutions and try a discretization. Starting point could be multi-commodity flows, or convex programming formulation. Another way of extending the model is by taking in consideration more general network topologies. The techniques used here could be translated into planar graphs.

Conclusion

Summary of the Results

In this work we present results concerning various topics.

In Chapter 1, we discuss bandwidth measurements between hosts over the Internet. First, we present *point-to-point* measurements. Such measurements are available in the form of datasets. We provide our dataset, named `bedibe`, together with measurement methodology that was used to obtain it. The platform used was PlanetLab, together with SPLAY middleware used to deploy Lua code.

We also present a new kind of dataset, a *bandwidth sharing* dataset. We discuss why is it enough to collect data in the form of one-to-two experimental setting. This dataset was also obtained using PlanetLab and SPLAY.

Our bandwidth sharing dataset shows very high variability, when looking at the repeated measures over the same configuration of nodes, of individual values over the edges. This variability is greatly reduced when looking at the sum of bandwidth achieved to both receiving nodes. We conjecture that this is due to the fact that the output bandwidth of the sending node is the limiting factor in those scenarios.

Chapter 2 introduces the notion of *Network Prediction Systems*. We describe various existing algorithms used for latency and bandwidth predictions. We show that DMF algorithm, based on matrix factorization, can be successfully adapted from predicting latencies to bandwidth predictions. We also describe LastMile model, together with several algorithms to instantiate this model, varying in form of complexity.

We also provide in this chapter an evaluation of these prediction algorithms using two datasets, S^3 (obtained by HP project on PlanetLab) and `bedibe`. We analyze the differences between the performance of the algorithms (in general DMF outperforms other algorithms), and also we analyze the differences between these datasets (S^3 proved easier to predict).

We also discuss the topic of predicting bandwidth in a scenario with congestion. We show that the LastMile model naturally extends its usability to such scenarios. We also provide validation of such predictions using one-to-two datasets described above.

Another contribution presented in this chapter is `bedibe`, a software created to automatize prototyping, testing and evaluation of various prediction algorithms. This software package, in the form of Python libraries, was extensively used when preparing this work.

In Chapter 3, we describe algorithms for the construction of efficient broadcast overlay networks. We consider the setting under LastMile model, where each node has strict incoming and outgoing connection capabilities. This broadcast problem is time-independent, *i.e.* we consider steady-state solutions. While incoming limits can be trivially discarded from the model, we name the outgoing capabilities of node i by b_i. We also put restriction, that out-degree of each node d_i is limited by $\left\lceil \frac{b_i}{T} \right\rceil$, where T is the achieved throughput. Such a limitation of the out-degree is justified by assuring that on average the outgoing edges of each node are close to T.

We show, that under such constraints, it is in general NP-Complete to decide if a given instance is feasible for given value of T. If we consider the problem of finding maximal possible T, our results are as follow. On the one hand, by dropping either degree constraints we can give simple expression for maximal possible T without those constraints, thus putting upper limit on the value of solution to the original problem. On the other hand, by restricting solutions to be represented by *acyclic* graph, we reduce original problem to one solvable by polynomial time algorithms, if we allow *resource augmentation*, that is $d_i \leq \left\lceil \frac{b_i}{T} \right\rceil + O(1)$. Such a solution is in a constant ratio from our upper bound (factor of 2), thus giving us a provable constant ratio approximation algorithm under degree augmentation.

We also consider the original problem with an extended model, by introducing a classification of nodes into *open* and behind *firewalls*. The nodes behind firewalls are prohibited to communicate between each other, and can only communicate with open ones, which are under no restriction whatsoever. We repeat the same reasoning, providing on the one hand simple upper bound by analyzing unrestricted communication, and on the other hand a greedy algorithm for finding best acyclic solution. We end the chapter with a lengthy proof of a upper bound on ratio between both of them equal to $\frac{7}{5}$, thus proving that the greedy acyclic algorithm provides a constant approximation ratio to the original problem when considering firewalls.

Chapter 4 considers the problem of finding optimal routing strategies in a platform corresponding to a grid graph, with costs being power-usage aware (in the sense of integrated circuits scale). Thus, we assume that power usage on each edge is proportional to f^α (for $\alpha \approx 3$), where f is the data transmission rate. Such assumption is justified by physical model of power usage and data transmission.

We are able to show, that in such a model, when considering simple model of routing connections between two endpoints only, and considering only *Manhattan Paths*, one can solve the underlying combinatorial problem, and achieve simple and elegant solutions when considering just one highly splittable request. We were able to reduce problem of unsplittable requests to solved problem of routing one splittable request, thus getting similar combinatorial schemes for routing. For the problem of routing several different requests, we provide an approximation algorithm.

Concluding Remarks

Based on our findings, we conclude that LastMile is a proper model to be used for construction of bandwidth-aware algorithms. The experimental evaluation done in Chapter 2 shows that LastMile performs worse than Matrix Factorization based algorithms, when we compare prediction ability for point-to-point scenarios. However, by sacrificing strength of prediction, we can gain a model with an intuitive interpretation of parameters, that can be easily instantiated. and that is much better suited for algorithmic design.

Experimental data described in Chapter 1 exhibit high variability (in point-to-point case), and prove it impossible to predict the bandwidth sharing scenario. Still, we can derive prediction models from LastMile able to predict with ratio similar to variability of original data, the value of total bandwidth used by the sender.

We believe that this is advantageous, as it makes algorithmic analysis as presented in Chapter 3 possible. The model coupled with a method of instantiation of its parameters leads us to an algorithm for constructing an efficient broadcast overlay network. This overlay does not specify which chunk of data to be transferred when, and gives us only information on usage of edges between any two given nodes of the network. We resolve the issue by using Massoulié's distributed algorithms ([47]), which by a simple randomized algorithm achieves optimal broadcast throughput.

We believe that this framework, where we start with proposing a model, then we design algorithms to instantiate a parameters of the model, use them to construct efficient overlay broadcast network, and finally apply distributed broadcast algorithm to achieve optimal broadcast rate, proves to be an important contribution in creation of bandwidth-aware solutions deployable over the Internet.

Future works

There are many topics left for future work.

Our datasets described in the Chapter 1 are relatively small. We feel that it would be very valuable to obtain larger ones. It might be hard to use highly limited platform as PlanetLab and still use brute-force method of measures. We consider two possible routes, either usage of different measure methodology, or moving to a different platform. The former solution, while still feasible when obtaining point-to-point measures (in fact S^3 dataset was obtained this way), may prove impossible when performing one-to-many measures. The latter one may prove a huge organizational problem, as the machines used in the platform should be spread across the globe to provide insightful measures over the Internet.

Chapter 2 leaves open question of designing improved version of LastMile prediction algorithm. We hope that the model we presented (2-level LastMile) or other models (one can think for example of trying to merge LastMile with a multi-level structures) can be proven to be easily instantiable and provide significantly better predictions than those based on basic LastMile model.

We encountered similar problem in the Chapter 3, in the sense of finding extension to the LastMile model. We would hope to develop further algorithmic approach derived there. We hope that either another extension to the LastMile model can be made (similar to the *firewalls* one), or

a stronger model can be analyzed algorithmically. Our various attempts to solve similar problems to the ones considered in this chapter, but for a more general problem of trees (instead of star-like topology) proved unsuccessful: analogous approach of considering acyclic solutions does not lead to problems being easily solvable in polynomial time.

The model described in Chapter 4 is simplistic, as it allows only a very limited cost function, limited routing policies, and considers only one pair of endpoints for requests. However there are plenty of possible routes to improve it. One can either try to solve similar problems, but for a more general class of graphs, for example planar graphs instead of grids. We believe this can be achieved using the same "analyze partial sums on diagonals" approach. A different approach would be to consider more general cost functions, for example with the incorporation of constant factors, or with weights varying between different edges. This seems necessary if we want the model and solutions derived here to better reflect real-life behaviors, furthermore the solution proposed considers pairs of endpoints as isolated and not interfering. This seems restrictive, and we hope to be able to derive solutions that can handle multiple arbitrary communications.

Bibliography

[1] D. Abendroth, H. van den Berg, and M. Mandjes. A versatile model for tcp bandwidth sharing in networks with heterogeneous users. *AEU - International Journal of Electronics and Communications*, 60(4):267 – 278, 2006.

[2] R. K. Ahuja, D. S. Hochbaum, and J. B. Orlin. Solving the convex cost integer dual network flow problem. In G. Cornuéjols, R. E. Burkard, and G. J. Woeginger, editors, *IPCO*, volume 1610 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 1999.

[3] A. Andrei, M. T. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi. Simultaneous communication and processor voltage scaling for dynamic and leakage energy reduction in time-constrained systems. In *International Conference on Computer-Aided Design*, pages 361–367. IEEE, 2004.

[4] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *IPDPS*, page 113. IEEE Computer Society, 2003.

[5] S. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. *Arxiv preprint cs/0412017*, 2004.

[6] O. Beaumont, N. Bonichon, L. Eyraud-Dubois, and P. Uznanski. Broadcasting on large scale heterogeneous platforms with connectivity artifacts under the bounded multi-port model. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 173–180, 2011.

[7] O. Beaumont, L. Eyraud-Dubois, and S. Agrawal. Broadcasting on large scale heterogeneous platforms under the bounded multi-port model. In *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–11. IEEE, 2010.

[8] O. Beaumont, L. Eyraud-Dubois, C. T. Caro, and H. Rejeb. Heterogeneous resource allocation under degree constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(5):926–937, 2013.

[9] O. Beaumont, L. Eyraud-Dubois, H. Rejeb, and C. Thraves. Allocation of clients to multiple servers on large scale heterogeneous platforms. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, pages 3 –10, feb. 2010.

[10] O. Beaumont, L. Eyraud-Dubois, and Y. Won. Using the last-mile model as a distributed scheme for available bandwidth prediction. In *Proceedings of the EuroPar 2011 conference*, 2011.

[11] O. Beaumont, L. Eyraud-Dubois, and Y. Won. Using the last-mile model as a distributed scheme for available bandwidth prediction. In *Proceedings of the EuroPar 2011 conference*, 2011.

[12] O. Beaumont and H. Rejeb. On the importance of bandwidth control mechanisms for scheduling on large scale heterogeneous platforms. In *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–12. IEEE, 2010.

[13] A. Benoit, R. G. Melhem, P. Renaud-Goud, and Y. Robert. Power-aware manhattan routing on chip multiprocessors. In *IPDPS*, pages 189–200. IEEE Computer Society, 2012.

[14] A. Benoit, P. Renaud-Goud, and Y. Robert. On the performance of greedy algorithms for power consumption minimization. In *ICPP*, pages 454–463. IEEE, 2011.

[15] G. Blake, R. Dreslinski, and T. Mudge. A survey of multicore processors. *Signal Processing Magazine*, page 26(6):26 –37, 2009.

[16] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: optimal performance trade-offs. *ACM SIGMETRICS Perform. Eval. Rev.*, 36:325–336, June 2008.

[17] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: high-bandwidth multicast in cooperative environments. *ACM SIGOPS Operating Systems Review*, 37(5):298–313, 2003.

[18] J.-J. Chen and T.-W. Kuo. Multiprocessor energy-efficient scheduling for real-time tasks with different power characteristics. In *ICPP*, pages 13–20. IEEE Computer Society, 2005.

[19] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Comput. Commun. Rev.*, 33:3–12, July 2003.

[20] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26, New York, NY, USA, 2004. ACM.

[21] A. B. Downey. Tcp self-clocking and bandwidth sharing. *Computer Networks*, 51(13):3844 – 3863, 2007.

[22] S. Ertel. Unstructured p2p networks by example: Gnutella 0.4, gnutella 0.6.

[23] P. Fraigniaud, E. Lebhar, and L. Viennot. The inframetric model for the internet. In *INFOCOM*, pages 1085–1093. IEEE, 2008.

[24] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman San Francisco, 1979.

[25] M. Gates, A. Tirumala, J. Ferguson, J. Dugan, F. Qin, K. Gibbs, and J. Estabrook. Iperf. `http://www.iperf.fr`.

[26] S. Guha, N. Daswani, and R. Jain. An experimental study of the skype peer-to-peer VoIP system. In *Proceedings of IPTPS*, volume 6, 2006.

[27] B. Hong and V. Prasanna. Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. *International Parallel and Distributed Processing Symposium*, 2004.

[28] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, pages 197–202, Aug. 1998.

[29] V. Jacobson. Congestion avoidance and control. In *SIGCOMM*, pages 314–329, 1988.

[30] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. In *Proceedings of ACM SIGCOMM*, pages 295–308, 2002.

[31] R. Jimenez, F. Osmani, and B. Knutsson. Connectivity properties of mainline bittorrent DHT nodes. In *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*, pages 262–270. IEEE, 2009.

[32] S. Johnsson and C. Ho. Optimum broadcasting and personalized communication in hypercubes. *IEEE Transactions on Computers*, 38(9):1249–1268, 1989.

[33] N. Karmarkar and R. M. Karp. The differencing method of set partitioning. Technical Report UCB/CSD-83-113, EECS Department, University of California, Berkeley, 1983.

[34] A. V. Karzanov and S. T. McCormick. Polynomial methods for separable convex optimization in unimodular linear spaces with applications. *SIAM J. Comp.*, 26(4):1245–1275, 1997.

[35] P. B. Key, L. Massoulié, and D.-C. Tomozei. Non-metric coordinates for predicting network proximity. In *INFOCOM*, pages 1840–1848. IEEE, 2008.

[36] M. Klein. A primal method for minimal cost flows. *Management Sci.*, 14:205–220, 1967.

[37] P. Langen and B. Juurlink. Leakage-aware multiprocessor scheduling. *Journal of Signal Processing Systems*, 57:73–88, 2009.

[38] S. E. Lee and N. Bagherzadeh. A variable frequency link for a power-aware network-on-chip (noc). *Integration*, 42(4):479–485, 2009.

[39] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca. Measuring bandwidth between planetlab nodes. In C. Dovrolis, editor, *Passive and Active Network Measurement*, volume 3431 of *Lecture Notes in Computer Science*, pages 292–305. Springer Berlin Heidelberg, 2005.

[40] L. Leonini, E. Rivière, and P. Felber. Splay: distributed systems evaluation made simple. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 185–198, Berkeley, CA, USA, 2009. USENIX Association.

[41] Y. Liao, W. Du, P. Geurts, and G. Leduc. Decentralized prediction of end-to-end network performance classes. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 14. ACM, 2011.

[42] Y. Liao, W. Du, P. Geurts, and G. Leduc. Dmfsgd: A decentralized matrix factorization algorithm for network distance prediction. *CoRR*, abs/1201.1174, 2012.

[43] Y. Liao, P. Geurts, and G. Leduc. Network distance prediction based on decentralized matrix factorization. In M. Crovella, L. Feeney, D. Rubenstein, and S. Raghavan, editors, *NETWORKING 2010*, volume 6091 of *Lecture Notes in Computer Science*, pages 15–26. Springer Berlin Heidelberg, 2010.

[44] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang. Performance bounds for peer-assisted live streaming. *ACM SIGMETRICS Perform. Eval. Rev.*, 36:313–324, June 2008.

[45] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang. Performance bounds for peer-assisted live streaming. *SIGMETRICS Perform. Eval. Rev.*, 36(1):313–324, June 2008.

[46] Y. Mao and L. K. Saul. Modeling distances in large scale by matrix factorization. In *IMC '04*, pages 278–287. ACM, Oct. 2004.

[47] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez. Randomized decentralized broadcasting algorithms. In *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pages 1073–1081, 2007.

[48] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, and R. Melhem. Energy aware scheduling for distributed real-time systems. In *In International Parallel and Distributed Processing Symposium*, page 21, 2003.

[49] F. d. Montgolfier, M. Soto, and L. Viennot. Treewidth and hyperbolicity of the internet. In *Proceedings of the 2011 IEEE 10th International Symposium on Network Computing and Applications*, NCA '11, pages 25–32, Washington, DC, USA, 2011. IEEE Computer Society.

[50] T. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *IEEE INFOCOM '02*, pages 170–179, June 2002.

[51] T. S. E. Ng and H. Zhang. Global network positioning: a new approach to network distance prediction. *SIGCOMM Comput. Commun. Rev.*, 32(1):61–61, Jan. 2002.

[52] J. D. Owens, W. J. Dally, R. Ho, D. J. Jayasimha, S. W. Keckler, and L.-S. Peh. Research challenges for on-chip interconnection networks. *IEEE Micro*, 27:96–108, 2007.

[53] L. Peterson, V. Pai, N. Spring, and A. Bavier. Using planetlab for network research: Myths, realities, and best practices. *PlanetLab, Design Note*, pages 05–028, 2005.

[54] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In M. Castro and R. Renesse, editors, *Peer-to-Peer Systems IV*, volume 3640 of *Lecture Notes in Computer Science*, pages 205–216. Springer Berlin Heidelberg, 2005.

[55] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *Network, IEEE*, 17(6):27–35, 2003.

[56] K. Pruhs, R. van Stee, and P. Uthaisombut. Speed scaling of tasks with precedence constraints. In T. Erlebach and G. Persiano, editors, *WAOA*, volume 3879 of *Lecture Notes in Computer Science*, pages 307–319. Springer, 2005.

[57] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella. On the treeness of internet latency and bandwidth. In *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 61–72, New York, NY, USA, 2009. ACM.

[58] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003.

[59] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *HPCA*, pages 91–102. IEEE Computer Society, 2003.

[60] Y. Shavitt and T. Tankel. Hyperbolic embedding of internet graph for distance estimation and overlay construction. *IEEE/ACM Trans. Netw.*, 16(1), Feb. 2008.

[61] P. Srisuresh, B. Ford, and D. Kegel. State of peer-to-peer (P2P) communication across network address translators (NATs), 2008.

[62] J. Strauss, D. Katabi, and M. F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Internet Measurement Comference*, pages 39–44. ACM, 2003.

[63] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello. Push-to-peer video-on-demand system: Design and evaluation. *IEEE JSAC*, 25(9):1706 –1716, Dec. 2007.

[64] Y. Tseng, S. Wang, and C. Ho. Efficient broadcasting in wormhole-routed multicomputers: a network-partitioning approach. *IEEE Transactions on Parallel and Distributed systems*, 10(1):44–61, 1999.

[65] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt. Mapping the PPLive network: Studying the impacts of media streaming on P2P overlays. Technical Report UIUCDCS-R-2006-2758, Department of Computer Science, University of Illinois at Urbana-Champaign, 2006.

[66] J. Watts and R. Geijn. A pipelined broadcast for multidimensional meshes. *Parallel Processing Letters*, 5(2):281–292, 1995.

[67] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee. S3: a scalable sensing service for monitoring large networked systems. In *Proceedings of the 2006 SIGCOMM workshop on Internet network management*, INM '06, pages 71–76, New York, NY, USA, 2006. ACM.

[68] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee. S3: a scalable sensing service for monitoring large networked systems. In *ACM SIGCOMM workshop on Internet network management*, pages 71–76, Pisa, Italy, Sept. 2006.

[69] X. Zhang, J. Liu, B. Li, and Y. Yum. CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In *Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, 2005.

[70] Y. Zhu, A. Velayutham, O. Oladeji, and R. Sivakumar. Enhancing tcp for networks with guaranteed bandwidth services. *Computer Networks*, 51(10):2788 – 2804, 2007.