Evaluation et optimisation de la performance des flots dans les réseaux stochastiques à partage de bande passante

Henda Ben Cheikh

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)*

Présentée et soutenue le *22/05//2015* **par :**

Henda BEN CHEIKH

Évaluation et optimisation de la performance des flots dans les réseaux stochastiques à partage de bande-passante

---

JURY

André-Luc Beylot	Professeur, INP-ENSEEIHT	Président du Jury
Hind Castel-Taleb	Professeur, Télécom SudParis	Rapporteur
Abdelhamid Mellouk	Professeur, UPEC	Rapporteur
Jonatha Anselmi	Chargé de Recherche, INRIA	Examinateur
Olivier BRUN	Chargé de Recherche, CNRS	Directeur de thèse
Jean-Marie GARCIA	Chargé de Recherche, CNRS	Directeur de thèse

---

École doctorale et spécialité :
EDSYS : Informatique 4200018
Unité de Recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS)
Directeur(s) de Thèse :
Olivier BRUN et *Jean-Marie GARCIA*
Rapporteurs :
Hind Castel-Taleb et *Abdelhamid Mellouk*

À ma mère Sabeh

Remerciements

Cette thèse constitue une riche expérience qui ne peut s'achever sans remercier les personnes qui m'ont encadré, aidé et soutenu au cours de ces trois dernières années.

Mes premiers remerciements sont adressés à mes directeurs de thèse Olivier Brun et Jean-Marie Garcia. J'exprime ma profonde gratitude et ma reconnaissance à Olivier et Jean-Marie d'avoir bien voulu diriger mes travaux de thèse. Je remercie Olivier pour son soutien, sa disponibilité, son expérience et sa patience ainsi que pour sa bonne humeur. Ses conseils, tout au long de ce travail, toujours pertinents et sa rigueur scientifique m'ont été très utiles pour mener à bien ce travail. Je tiens aussi à remercier Jean-Marie pour la confiance qu'il m'a témoignée tout au long de ces années et pour tous ses conseils et remarques constructives.

Je dois aussi une grande partie de mon travail à Balakrishna J. Prabhu. Merci infiniment Bala pour ton aide, tes conseils, ta disponibilité et ta patience. Je remercie également Ahmad Al Sheikh pour ses conseils, sa disponibilité et son aide.

Mes remerciements vont ensuite aux membres du jury, qui m'ont fait l'honneur d'évaluer mon travail de thèse et ont accepté de faire le déplacement pour assister à ma soutenance. Je vous en suis très reconnaissante. Tout d'abord, mes remerciements vont à André-Luc Beylot, professeur à L'INP-ENSEEIHT pour avoir accepté de présider mon jury de thèse. Je remercie également Hind Castel-Taleb, professeur à Télécom SudParis et Abdelhamid Mellouk, professeur à Université Paris-Est Créteil Val de Marne pour leur long travail de rapporteur. Merci infiniment pour le temps que vous avez consacré à la lecture de mon manuscrit et à la rédaction de vos rapports éclairés. Je remercie aussi Jonatha Anselmi, chargé de recherche au centre INRIA-Bordeaux, pour avoir bien voulu faire partie de mon jury.

Je tiens aussi à remercier tous les membres de l'équipe SARA, chercheurs, doctorants et stagiaires, pour leur disponibilité, leur sympathie, et pour l'ambiance amicale qui règne au sein de l'équipe. Merci aussi à tous ceux du LAAS-CNRS qu'il m'a été donné de croiser pendant ces trois années. Je tiens à remercier particulièrement Khalil Drira de m'avoir accueilli au sein de l'equipe SARA dont il est le responsable.

Je remercie également Sonia De Sousa et Caroline Malé, secrétaires de l'équipe SARA pour leur aide précieuse, ainsi que les différents services techniques et administratifs du LAAS-CNRS qui m'ont permis de travailler dans de très bonnes conditions.

Pendant ces trois années, cela aura été un plaisir de les partager avec mes collègues déjà docteurs ou en voie de le devenir. Un merci tout particulier à mes collègues de bureau Josselin, Josu et Christopher pour les discussions qui ont étés très utiles et aussi pour leur agréable compagnie. Merci beaucoup pour votre bonne humeur, votre gentillesse et votre générosité tout au long de ces trois années ! Je ne peux que vous souhaiter beaucoup de bien et bonne chance ! Je tiens à citer également mes collègues, et non moins amis, pour leur aide, leur soutien et leur gentillesse :

Je te remercie Tom pour ta gentillesse et ta générosité ! je ne peux te souhaiter que le meilleur. Merci Amira pour ton soutien, les longues discussions et les inoubliables moments de rire qu'on a pu avoir ! Je ne peux que te souhaiter plein succès. Merci aussi à Ikbel, Maialen, Ane, Yassine, Ghada, Mehdi, Emna, Salma... à qui je souhaite bonne continuation.

Mes remerciements vont bien évidement à tous les gens autour de moi qui m'ont soutenu dans mon travail surtout pendant les moments délicats qui se reconnaîtront ici. Un très grand merci pour votre soutien. Je ne l'oublierais jamais.

Je terminerai mes remerciements en pensant à mes précieux parents Sabeh et Hamda à qui je dédie cette thèse. Merci papa, merci maman ! Merci pour tout ce que vous m'avez appris, d'avoir toujours été présents quand j'ai eu besoin de vous. Sans vous, je ne serais pas là ! Sans oublier aussi mes chers frères Abdelkrim et Mohamed qui me soutiennent et m'encouragent continuellement.

Merci à tous.
Henda

Résumé-Abstract

Résumé :

Nous étudions des modèles mathématiques issus de la théorie des files d'attente pour évaluer et optimiser les performances des mécanismes de partage de ressources entre flots dans les réseaux.

Dans une première partie, nous proposons des approximations simples et explicites des principales métriques de performance des flots élastiques dans les réseaux à partage de bande passante opérant sous le mode "équité équilibré". Nous étudions ensuite le partage de bande passante entre flux élastiques et flux de streaming en supposant que le nombre de ces derniers est limité par un mécanisme de contrôle d'admission, et proposons des approximations de performance basées sur une hypothèse de quasi-stationnarité. Les résultats de simulation montrent le bon niveau de précision des approximations proposées.

Dans une deuxième partie, nous étudions le compromis entre délai et énergie dans les réseaux à partage de bande passante dont les nœuds peuvent réguler leur vitesse en fonction de la charge du système. En supposant que le réseau est initialement dans un état de congestion, on s'intéresse à la politique optimale d'allocation de débit permettant de le vider à coût minimal. L'analyse de la politique stochastique optimale via la théorie des processus de décision markoviens étant extrêmement difficile, nous proposons de l' approximer en utilisant un modèle fluide déterministe qui peut être résolu grâce à des techniques de contrôle optimal. Pour le cas d'un seul lien partagé par plusieurs classes de trafic, on montre que la politique optimale correspond à la régle $c\mu$ et on propose une expression explicite de la vitesse optimale.

Enfin, dans une troisième partie, on s'intéresse aux plateformes de Cloud Computing dans le cadre du modèle SaaS. En supposant un partage équitable des ressources physiques entre machines virtuelles s'exécutant de manière concurrente, nous proposons des modèles de file d'attente simples pour prédire les temps de réponse des applications. Les modèles proposés prennent explicitement en compte le comportement des différentes classes d'application (tâches interactives, de calcul ou permanentes). Les expérimentations menées sur une plateforme réelle montrent que les modèles mathématiques obtenus permettent de prédire les temps de réponse avec une bonne

précision.

mots-clés : réseaux à partage de bande passante, trafic IP, théorie de files d'attente, estimation de performance, Équité équilbrée, théorie de contrôle, règle $c\mu$.

Abstract :

We study queueing-theoretic models for the performance evaluation and optimization of bandwidth-sharing networks.

We first propose simple and explicit approximations for the main performance metrics of elastic flows in bandwidth-sharing networks operating under balanced fairness. Assuming that an admission control mechanism is used to limit the number of simultaneous streaming flows, we then study the competition for bandwidth between elastic and streaming flows and propose performance approximations based on a quasi-stationary assumption. Simulation results show the good accuracy of the proposed approximations.

We then investigate the energy-delay tradeoff in bandwidth-sharing networks in which nodes can regulate their speed according to the load of the system. Assuming that the network is initially congested, we investigate the rate allocation to the classes that drains out the network with minimum total energy and delay cost. We formulate this optimal resource allocation problem as a Markov decision process which proves to be both analytically and computationally challenging. We thus propose to solve this stochastic problem using a deterministic fluid approximation. For a single link shared by an arbitrary number of classes, we show that the optimal-fluid solution follows the well-known $c\mu$ rule and give an explicit expression for the optimal speed.

Finally, we consider cloud computing platforms under the SaaS model. Assuming a fair share of the capacity of physical resources between virtual machines executed concurrently, we propose simple queueing models for predicting response times of applications. The proposed models explicitly take into account the different behaviors of the different classes of applications (interactive, CPU-intensive or permanent applications). Experiments on a real virtualized platform show that the mathematical models allow to predict response times accurately.

keywords : bandwidth sharing networks, IP traffic, queuing theory, performance estimation, Balanced Fairness, control theory, $c\mu$ rule.

Table des matières

Table des figures

1 Introduction

1.1 Contexte

À la fin des années 90, l'Internet a connu un succès planétaire avec le développement incroyable du World Wide Web, l'adoption par tout un chacun d'applications telles que le courrier électronique ou encore avec le déploiement des logiciels de partage de fichiers pair à pair. Ce succès n'a fait que s'amplifier depuis avec sa transformation en une architecture de communication globale supportant, en plus des services "traditionnels", de nouvelles applications basées sur la voix ou la vidéo qui ont des contraintes de qualité de service beaucoup plus strictes. Un autre fait marquant est la convergence des réseaux d'accès, qu'ils soient filaires ou basés sur des technologies radio (UMTS, WiFi, etc.), qui a permis aux utilisateurs d'accéder aux mêmes services, quel que soit le terminal utilisé et où qu'ils se trouvent. Enfin, on assiste actuellement à la prolifération des nouvelles applications, comme par exemple les réseaux sociaux ou le cloud computing.

Toutes ces évolutions conduisent à une multiplication des services offerts par les réseaux et à une croissance sans précédent du nombre d'utilisateurs et des volumes de trafic qu'ils génèrent. Aujourd'hui, avec plus de 2 milliards d'internautes dans le monde, les technologies de l'information et de la communication sont de plus en plus présentes dans nos activités quotidiennes [1], fondant ainsi une nouvelle économie dont le poids est devenu considérable. Au delà, elles sont devenues un composant critique dans tous les

1. Pour ne citer que quelques chiffres, le moteur de recherche de Google a plus de 620 millions de visites par jour, il y a eu plus de mille milliards de vidéos visionnées sur Youtube en 2011, et l'encyclopédie Wikipédia compte plus de 26 millions d'articles.

secteurs d'activité industriels. Dans ce contexte, l'interruption des services fournis par les réseaux, ou même une dégradation significative de la qualité de service, deviennent de moins en moins tolérables. Garantir la continuité et la qualité des services offerts est ainsi un enjeu majeur pour les opérateurs de réseaux qui doivent en permanence adapter leurs infrastructures.

Cependant, le contexte concurrentiel actuel, ne permet plus d'améliorer les performances d'un réseau par un sur-dimensionnement excessif des équipements. Pour les opérateurs, la solution est d'avoir un suivi plus régulier et plus fin de leurs infrastructures et d'utiliser des techniques d'ingénierie de trafic pour anticiper les phénomènes de congestion et les dégradations de qualité de service qui en résultent. L'utilisation de ces techniques supposent toutefois de disposer de modèles et de méthodes théoriques ainsi que des outils logiciels appropriés pour prédire et contrôler la qualité de service des flux.

1.2 Objectifs et contributions

Les travaux menés par Erlang et Engset au début du XX^e siècle, dont est issue la théorie des files d'attente, ont été utilisés pendant longtemps par les opérateurs téléphoniques pour planifier leurs réseaux. Notre conviction est qu'aujourd'hui encore, les outils mathématiques issus de la théorie des files d'attente ont un rôle essentiel à jouer dans l'ingénierie des réseaux et des systèmes informatiques au sens large. Le défi majeur est toutefois celui du passage à l'échelle, que ne permettent ni la simulation évènementielle, qui a des temps de calcul prohibitifs même pour de petits réseaux, ni les techniques de modélisation markovienne, du fait de l'explosion combinatoire de l'espace d'états.

Dans ce mémoire, nous utilisons des techniques de modélisation issues de la théorie des files d'attente pour étudier les trois problématiques détaillées ci-dessous.

1.2.1 Estimation des performances des flux Internet

Un effort de recherche important a été consacré depuis quinze ans au développement d'une théorie du trafic pour l'Internet. En comparaison avec son équivalent pour les réseaux téléphoniques, cette théorie du trafic Internet est encore très limitée. La principale difficulté de modélisation provient de la nature élastique de la plupart des trafics Internet dont le débit peut être modulé par le protocole TCP en fonction des conditions de trafic dans le réseau. En dépit des avancées significatives réalisées (cf. chapitre 2), la théorie du trafic Internet en est encore à ses balbutiements. Il n'existe pas à l'heure actuelle de modèles de performance satisfaisants permettant d'évaluer précisément les performances d'une architecture TCP/IP. Un des objectifs de nos travaux est de concevoir des schémas d'approximation permettant l'évaluation des performances des flux dans de très grands réseaux.

Plus précisément, nous proposons des approximations simples et explicites des principales métriques de performance des flots élastiques dans les réseaux à partage de bande passante opérant sous le mode "équité équilibré". Nous analysons également l'intégration des flux élastiques et flux de streaming en supposant que le nombre de ces derniers est limité par un mécanisme de contrôle d'admission, et proposons des approximations de performance basées sur une hypothèse de quasi-stationnarité.

1.2.2 Compromis entre énergie et performance dans les réseaux

L'usage croissant des smart-phones et des tablettes numériques, de même que le développement de nombreuses applications gourmandes en bande passante devraient amener le réseau Internet à devoir supporter des volumes de trafic sans précédent dans les années à venir [5]. Dans ce contexte, l'un des plus grands défis de l'industrie des technologies de l'information et de la communication (TIC) est de gérer cette croissance du trafic des données de manière durable et économiquement soutenable. La prise en compte de la consommation énergétique des réseaux devient ainsi incontournable.

En s'inspirant des techniques de speed-scaling utilisées en informatique, nous étudions le compromis entre délai et énergie dans les réseaux à partage de bande passante dont les nœuds peuvent réguler leur vitesse en fonction de la charge du système. L'enjeu est de caractériser la politique optimale d'allocation de débit permettant de faire passer le réseau d'un état de congestion à un état considéré comme souhaitable en minimisant le coût en termes d'énergie consommée et de performance des flux. Étant donnée la complexité du problème, nous proposons une approximation basée sur un modèle fluide déterministe qui peut être résolu grâce à des techniques de contrôle optimal. On montre, pour le cas d'un seul lien, que la politique optimale coïncide avec la fameuse règle $c\mu$ et nous proposons aussi une expression explicite de la vitesse optimale.

1.2.3 Temps de réponse des plateformes de cloud computing

Le Cloud Computing est aujourd'hui une solution très pratique pour externaliser ses activités et réduire ainsi ses coûts de gestion, d'achat et d'entretien des infrastructures informatiques. C'est une approche conjuguant de multiples technologies dont la virtualisation qui joue le rôle crucial de catalyseur pour les plateformes de cloud computing. Pour les modèles SaaS (pour *Software as service*) que nous considérons dans ce travail, les applications sont souvent exécutées dans des machines virtuelles, hébergées sur des machines physiques d'un data-center, dont les capacités de traitement sont partagées simultanément par plusieurs machines virtuelles. Ce partage pourrait éventuellement entrainer une baisse significative de performance en cas de forte demande. Les fournisseurs de services ont alors besoin d'outils leurs permettant de répondre de façon appropriée aux exigences des utilisateurs en termes de performance. Un autre objectif

de nos travaux, est de proposer des approximations permettant de prédire les performances des applications dans une plateforme de Cloud Computing dans le cadre du modèle SaaS. Plus précisément, en utilisant des modèles simples issus de la théorie de file d'attentes, on propose des approximations explicites du temps de réponse moyen des différents types d'applications (interactive, CPU-intensive et serveur web) s'exécutant dans un environnement virtualisé.

1.3 Organisation de ce document

La suite de ce document s'organise comme suit :

— Le chapitre 2 présente un état de l'art sur la modélisation du trafic Internet.

— Le chapitre 3 propose plusieurs approximations pour estimer la qualité de services des flux élastiques et des flux de streaming dans l'Internet.

— Le chapitre 4 étudie la politique de partage de bande passante permettant d'optimiser un compromis entre consommation énergétique et débit des flux.

— Le chapitre 5 propose des approximations simples et explicites du temps moyen de réponse des applications executées dans des plateformes de cloud computing.

À la fin de ce document, nous discutons brièvement l'apport de chacune des contributions présentées et dégageons quelques perspectives.

1.4 Notations

Avant de conclure ce chapitre, nous présentons quelques notations utilisées dans le reste de ce mémoire :

— Dans la suite, $\mathbb{R}_+$ désigne l'ensemble des réels positifs et $\mathbb{N}$ l'ensemble des entiers naturels.

— On note $\mathbf{e}_i$ le vecteur unitaire avec 1 dans la position i, et 0 ailleurs.

— Etant donnés $\boldsymbol{y} = (y_1, \ldots, y_M)^T \in \mathbb{R}_+^M$ et $\boldsymbol{x} = (x_1, \ldots, x_M)^T \in \mathbb{N}^M$, on note

$$|\boldsymbol{x}| = \sum_{m=1}^{M} x_m, \quad \boldsymbol{x}! = \prod_{m=1}^{M} x_m!, \quad \text{et } \boldsymbol{y}^{\boldsymbol{x}} = \prod_{m=1}^{M} y_m^{x_m}.$$

— Si $\pi()$ est une distribution de probabilités sur $\mathbb{N}^M$ et $\mathcal{B}$ un sous-ensemble de $\mathbb{N}^M$, on note $\pi(\mathcal{B}) = \sum_{\boldsymbol{x}\in\mathcal{B}} \pi(\boldsymbol{x})$.

— On utilisera la notation $B(N, x)$ pour représenter la formule d'Erlang C pour un système de N serveurs avec un trafic offert de x Erlangs.

La précision des résultats présentés dans ce manuscrit est exprimée en pourcentage d'erreur relative, i.e.,

$$RelErr(\%) = 100\ \frac{\|T_{meas} - T_{mod}\|}{T_{meas}} \tag{1.1}$$

où T_{meas} et T_{mod} représentent la valeur mesurée par les simulations (ou expérimentations) et la valeur théorique obtenue avec nos modèles, respectivement.

2

Modélisation du trafic : État de l'art

Résumé

Ce chapitre présente le contexte dans lequel s'inscrit notre étude, qui a pour objectif de modéliser mathématiquement les réseaux à partage de bande passante afin d'étudier des problématiques liées à l'évaluation des performances et au partage optimal des ressources dans ces systèmes. Au paragraphe 2.2, on introduit les systèmes auxquels on s'intéresse dans le cadre de cette thèse ainsi que nos hypothèses sur la nature du trafic circulant dans ces systèmes et en particulier le trafic IP. Finalement, un état de l'art sur la modélisation de ces systèmes au niveau flot est présenté dans le reste de ce chapitre.

Sommaire

2.1 Introduction

La modélisation mathématique consiste à représenter un système réel potentiellement très complexe par un modèle simplifié afin de mieux comprendre et analyser son fonctionnement et d'en déduire éventuellement certaines métriques de performance. Dans ce mémoire, on s'intéresse particulièrement à la modélisation des réseaux de communication à partage dynamique de ressources. Le trafic circulant dans ces systèmes étant aléatoire, la théorie des files d'attente [61, 58] fournit le cadre théorique naturel pour leur modélisation mathématique. Comme nous allons le voir dans la suite, la modélisation peut s'effectuer à différents niveaux en fonction des objectifs poursuivis.

Dans ce chapitre, nous rappelons brièvement au paragraphe 2.2 quelques principes de fonctionnement des réseaux IP. Nous abordons ensuite au paragraphe 2.3 la modélisation mathématique de ces réseaux, en présentant les avantages d'une modélisation au niveau flot plutôt qu'au niveau paquet et en discutant les hypothèses sous sous-jacentes. Afin de décrire le contexte des travaux présentés au chapitre 3, nous présentons au paragraphe 2.4 les principaux modèles de partage de bande passante entre flots élastiques proposés dans la littérature, puis considérons le partage de bande passante entre flots élastiques et flots de streaming au paragraphe 2.5. Finalement, nous décrivons au paragraphe 2.6 quelques résultats d'ordonnancement stochastique qui seront utilisés dans le chapitre 4.

2.2 Les Réseaux TCP/IP

Nous présentons ci-dessous quelques éléments sur les technologies utilisées dans l'Internet qui nous semble utiles pour la compréhension du trafic Internet et l'évaluation de ses performances. Nous commençons par présenter la différence entre les deux principaux modes de partage des ressources d'un réseau : la commutation de circuits et la commutation de paquets. Nous introduisons ensuite une classification des trafics dans l'Internet, avant de donner quelques éléments sur les principaux protocoles de transport qui y sont utilisés.

2.2.1 Mode de partage de ressources

Il existe deux modes pour partager les ressources d'un réseau (cf. figure 2.1) :

— **commutation de circuits :** ce mode de partage des ressources, utilisé en téléphonie, consiste à réserver des ressources à l'établissement de la connexion le long d'un chemin entre l'émetteur et le récepteur. Les ressources sont dédiées à la connexion durant toute la durée de la communication, même lorsqu'aucune donnée n'est échangée.

— **commutation de paquets :** ce mode de partage des ressources, utilisé dans les réseaux TCP/IP [1] [41], consiste à découper les données à transmettre en paquets de petit taille, qui peuvent ensuite être acheminés de manière indépendante jusqu'à la destination grâce aux informations contenues dans l'en-tête des paquets.

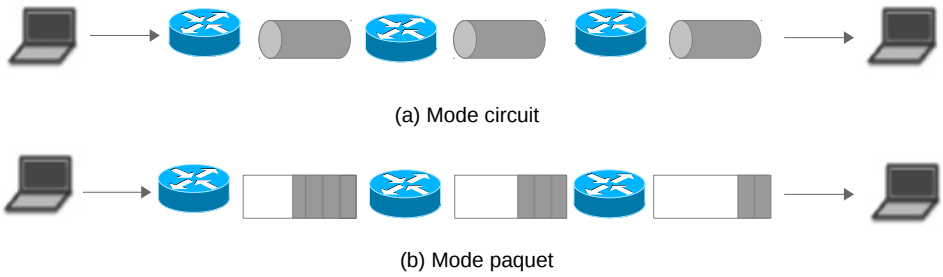


FIGURE 2.1 – Modes de communication.

La commutation de circuits permet, en réservant en permanence des ressources à la communication, de garantir la qualité de service des flux. La quantité de ressources disponibles dans le réseau étant toutefois limitée, il se peut qu'une connexion ne puisse pas être établie : c'est le phénomène de blocage des appels, qui constitue la métrique de performance principale des réseaux téléphoniques. À l'inverse, il est beaucoup plus complexe de garantir la qualité de service des flux dans les réseaux à commutation de paquets. Toutefois, un avantage évident de la commutation de paquets sur la commutation de circuits est une utilisation beaucoup plus rationnelle des ressources de communication : les ressources ne sont réservées que durant leur utilisation effective. Dans ce mémoire, nous nous intéressons uniquement aux réseaux à commutation de paquets basés sur la technologie TCP/IP.

2.2.2 Trafic IP

Le trafic IP peut être classifié en deux grandes catégories de flots ayant des propriétés et des besoins différents en termes de qualité de service [82] :

— **trafic élastique** : les flots élastiques sont ainsi nommés car leur débit est régulé par le protocole de transport TCP pour s'adapter aux ressources disponibles dans le réseau. Ces flots sont générés par des applications qui n'ont pas de contrainte temporelle et peuvent donc accepter de grandes variations de délai et compenser

1. famille de protocoles de communication conçus pour être utilisés par Internet.

les pertes éventuelles par des retransmissions. Les flots élastiques concernent ainsi toutes les fonctionnalités traditionnelles telles que le transfert de pages Web (protocole HTTP [46]), de messages électroniques (protocole SMTP [72]) ou de fichiers de données (protocole FTP [36]). Le maintien d'un débit à long terme est suffisant pour garantir un temps de réponse satisfaisant pour l'utilisateur final.

— **trafic non-élastique**[2] : les flots de cette catégorie correspondent aux applications interactives multimédias (la voix sur IP (VoIP), la vidéo conférence,...) qui utilisent généralement le protocole de transport UDP. Contrairement aux flots élastiques, ces flots sont générées par des applications qui ne peuvent pas s'adapter aux conditions de trafic dans le réseau car elles ont des contraintes du type temps réel et ont besoin de garanties de délai et/ou de débit pour les flots de données qu'elles génèrent. Ils peuvent également nécessiter un taux de perte faible. Différents types d'applications auront besoin de garanties plus ou moins strictes.

2.2.3 Protocoles de transport

Les protocoles de transport les plus utilisés dans les réseaux IP sont TCP (pour *Transmission Control Protocol*)[78] et UDP (pour *User Datagram Protocol*)[77]. TCP est un protocole qui permet la transmission fiable de paquets IP en mode connecté. TCP est généralement utilisé pour des applications qui n'ont pas de contrainte temporelle spécifique et assure un partage équitable des ressources. Contrairement à TCP, UDP est un protocole qui ne garantit pas la bonne livraison des paquets de données à destination, ni leur ordre d'arrivée. UDP est généralement utilisé pour le trafic de streaming.

Il existe également des protocoles de contrôle de congestion dits TCP-friendly [52] qui sont utilisés pour des applications sensibles au délai et/ou aux pertes mais capables d'adapter leur débit en fonction des conditions de trafic dans le réseau à la manière d'un flux TCP (grâce à un changement de codec par exemple). L'idée clef est d'éliminer les fluctuations drastiques de la fenêtre de congestion de TCP et d'ajuster le débit de manière beaucoup plus progressive. Pour être équitable avec les flux pilotés par TCP, le débit alloué aux flux TCP-friendly est celui que recevrait une connexion TCP persistante dans les mêmes conditions de trafic (d'où le nom TCP-friendly).

Néanmoins, la plupart des applications de streaming n'ont pas la capacité de s'adapter et attendent du réseau un débit de transmission constant. Ce sont ces applications de streaming que nous considérons dans cette thèse. Le service "au mieux" (best effort) proposé par l'Internet n'est clairement pas adapté à ces applications [38] . En s'inspirant des travaux réalisés pour l'ATM [91], des architectures comme Intserv ou Diffserv [92, 21] ont été proposées pour offrir un service différencié aux applications sensibles aux délais ou aux pertes. Elles sont toutefois parfois critiquées pour leur complexité ou

2. Dans la suite, ce trafic sera aussi appelé trafic de streaming

les problèmes de passage à l'échelle qu'elles posent.

2.3 Modélisation mathématique du trafic Internet

La complexité des protocoles de transport justifie l'utilisation de modèles simplifiés en vue de mieux analyser le fonctionnement du système. Les modèles doivent être suffisamment simples pour être analysables, et en même temps suffisamment détaillés pour pouvoir évaluer précisément les métriques d'intérêt tel que la durée moyenne de requêtes ou bien taux de pertes, etc. Une caractéristique fondamentale des réseaux IP est que le trafic généré par les utilisateurs est par nature imprévisible. On est ainsi amené à utiliser des modèles dans lesquels l'arrivée des communications et leurs volumes sont gouvernés par des processus stochastiques. La théorie des files d'attente fournit alors un cadre naturel pour la modélisation et l'analyse des réseaux. Comme expliqué ci-dessous, la modélisation peut toutefois être faite à différents niveaux, suivant les métriques que l'on cherche à évaluer. Nous discutons également les hypothèses qui peuvent être faites sur les processus stochastiques gouvernant les arrivées et la taille des connexions.

2.3.1 Modèles de niveau paquet et de niveau flot

On peut modéliser le trafic Internet à différentes échelles de temps et suivant différents niveaux de granularité (cf. figure 2.2).

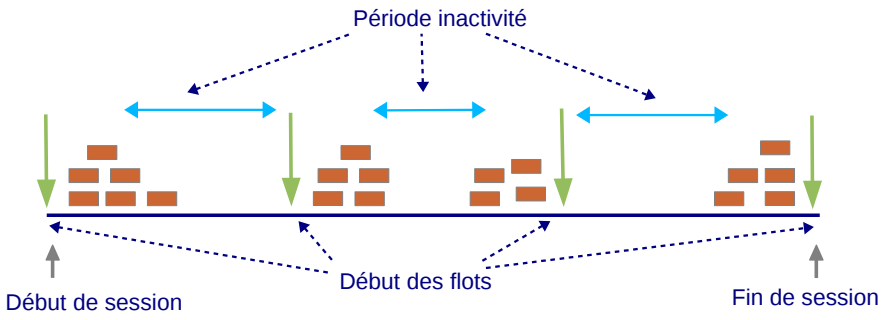


FIGURE 2.2 – Granularité du trafic.

Le niveau de granularité le plus fin est celui où l'unité élémentaire correspond au paquet IP. Un autre niveau est celui des flots, définis comme l'ensemble des paquets ayant les mêmes adresses source et destination. Le niveau supérieur groupe les flots en sessions, définis comme une succession de flots reflétant l'activité d'un utilisateur. Dans la littérature sur la modélisation du trafic Internet, on peut ainsi distinguer trois types de modèles : les modèle de niveau paquet, les modèles de niveau flot et les modèles de niveau session. Les sessions étant en pratique plus difficiles à identifier que les flots, la très grande majorité des travaux portent sur les deux premiers types de modèles.

Les modèles de niveau paquet intègrent de nombreux détails sur le système (Round Trip Times, taille des buffers, etc.) mais considèrent en général un nombre fixe de flots persistants [68, 76, 8, 35]. L'échelle de temps correspondante est de l'ordre de la microseconde ou de la milliseconde en fonction du débit des liens considérés. Si ces modèles peuvent être pertinents pour calculer des métriques de performance de niveau paquet (taux de perte ou délai de transmission par exemple), leur inconvénient majeur est de ne pas prendre en considération la dynamique au niveau flot, c'est-à-dire l'arrivée des flots à des instants aléatoires et les volumes aléatoires de données à transmettre.

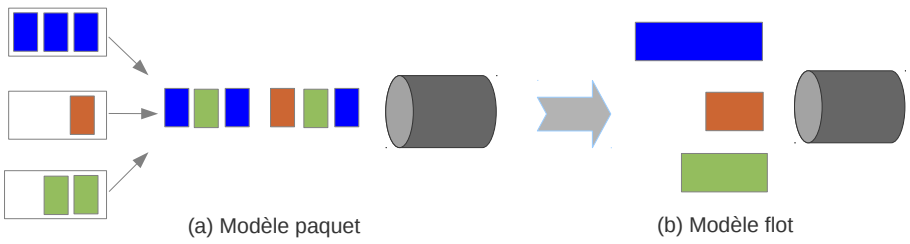


FIGURE 2.3 – Modèle de niveau flot.

Dans les modèles de niveau flot [84], les flots sont représentés par un fluide s'écoulant à un certain débit qui varie instantanément à chaque arrivée ou départ d'un autre flot (cf. figure 2.3). Les flots sont générés selon un certain processus stochastique et quittent le réseau une fois que leur volume aléatoire est écoulé. Contrairement aux modèles précédents, ces modèles ignorent les phénomènes de niveau paquet (accès au médium, propagation, pertes, etc.) et s'intéressent à la modélisation de la dynamique des flots. Ce sont donc des modèles idéalisés qui prennent en compte la dynamique aléatoire au niveau des flots (arrivées et départs de connexions), mais qui utilisent un modèle très

simplifié du partage de bande-passante réalisé par TCP (voir [85] pour un survey).

Dans le reste de ce mémoire, nous nous focaliserons sur des modèles de niveau flot. En effet, l'analyse des performances semble plus pertinente au niveau flot, car les métriques de performance critiques sont celles perçues par l'utilisateur final telles que la durée de transfert d'un fichier ou le débit moyen disponible. Ces métriques dépendent principalement de la dynamique des flots et de la manière dont les ressources sont partagées entre les flots.

2.3.2 Dynamique des flots

Comme expliqué précédemment, les instants auxquels les communications sont générés par les utilisateurs, et les volumes de données qu'elles transfèrent, sont représentés par des processus stochastiques. Comme expliqué dans [47], les arrivées de sessions suivent généralement un processus de Poisson. Ceci s'explique par le fait que les sessions sont généralement générées indépendamment par une large population d'utilisateur. Les arrivées de demande à ce niveau résultent donc de la superposition d'un nombre élevé de demandes élémentaires indépendantes entre elles [3].

Ce n'est généralement pas le cas pour les arrivées des flots. En effet, dans le cas où les flots d'une même session correspondent à des transferts de données successifs par un même utilisateur, comme dans le cas par exemple d'une session web, on peut s'attendre à une certaine corrélation entre les instants auxquels les flots sont générés. Toutefois, l'hypothèse d'un processus d'arrivée de flots suivant une loi de Poisson est couramment admise pour simplifier l'analyse des performances du système. En effet, avec cette hypothèse, de nombreuses quantités comme le débit moyen d'un flot ou bien la probabilité de rejet d'un flot peuvent être exprimés par des expressions mathématiques.

L'hypothèse que les volumes aléatoires des flots sont distribués selon une loi exponentielle est aussi considéré dans plusieurs travaux [45]. D'autre modèles plus réalistes reposent simplement sur le fait que les volumes sont des processus aléatoires indépendant et identiquement distribués.

Nous verrons dans la section suivante que sous certaines hypothèses sur le modèle de partage de bande passante, les métriques de performances sont insensibles aux caractéristiques détaillées du trafic. Elles ne dépendent alors que de l'intensité moyenne du trafic. Il s'agit évidemment d'une propriété essentielle pour l'utilisation pratique des formules obtenues. C'est notamment cette propriété qui a assuré le succès de la fameuse formule d'Erlang B.

3. C'est l'un des principaux invariants communément reconnus en modélisation du trafic Internet [47]

2.4 Partage de bande passante entre flots élastiques

Nous avons vu les hypothèses qui peuvent être faites concernant les processus stochastiques gouvernant l'arrivée des connexions et les volumes de données transférées. Un autre composant essentiel d'un modèle du trafic Internet concerne les hypothèses faites sur le partage de bande passante entre les flots. Dans ce paragraphe, nous passons en revue les principaux modèles proposés dans la littérature pour représenter le partage des ressources entre flots élastiques.

2.4.1 Modèles à un seul lien

Le modèle classique consiste en un seul lien isolé modélisé par une file d'attente avec un serveur de capacité C fixe partagé par des flots élastiques. La discipline de partage PS (pour *Processor Sharing*) est la plus utilisée dans ce cas pour modéliser le partage équitable de la capacité réalisé par TCP [48]. Les modèles à processeur partagé ont été initialement introduits par Kleinrock [60] pour modéliser des processeurs multi-tâche à temps partagé. Sous la discipline PS, la capacité est partagée équitablement par les flots actifs : en notant x le nombre de flots actifs, chaque flot reçoit le même débit C/x.

Kleinrock a également introduit la discipline DPS (pour *Discriminatory Processor Sharing*) [60] [6] comme généralisation de la discipline PS pour des systèmes multiclasses. Cette discipline de service affecte différents poids aux utilisateurs de différentes classes. Supposons à titre d'exemple qu'il y ait K classes dans le système et notons w_i (resp. x_i) le poids (resp. le nombre de flots) associé à la classe i. Chaque flot de la classe i reçoit alors le débit suivant :

$$\frac{w_i}{\sum_{k=1}^{K} x_k w_k} \tag{2.1}$$

La discipline GPS (pour *Generalized Processor Sharing*) a été aussi introduite dans [43] pour modéliser le partage de capacité dans un système multiclasse. Sous la discipline de service GPS, le débit agrégé de la classe i est

$$\frac{w_i 1_{x_i>0}}{\sum_{k=1}^{K} w_k 1_{x_k>0}} \tag{2.2}$$

Pour des modèles à plusieurs liens avec des flots hétérogènes, d'autres algorithmes (détaillés dans la section suivante) ont été proposés pour modéliser le partage de bande passante réalisé par TCP.

2.4.2 Modèles à plusieurs liens

Nous allons illustrer les principales notions en utilisant l'exemple d'un réseau linéaire tel que celui décrit sur la figure 2.4.

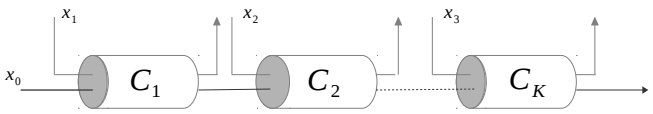


FIGURE 2.4 – Exemple de réseau linéaire.

Plus précisément, on considère un réseau linéaire constitué de L liens, le lien l ayant la capacité C_l. Dans ce réseau, il y a un ensemble $\mathcal{E} = \{0, 1, \ldots, L\}$ de classes de trafic. On note x_i le nombre de flots actifs de la classe $i \in \mathcal{E}$ et on prend le vecteur $\boldsymbol{x} = (x_0, x_1, ..., x_L)$ pour état du système. À chaque classe i est associée une route r_i. Pour le type de réseaux linéaires que nous considérons, les flots de classe $l = 1, \ldots, L$ traversent seulement le lien l tandis que les flots de classe 0 traversent tous les liens. On notera A la matrice d'incidence associée au routage des classes de trafic dans le réseau. C'est une matrice telle que $a_{i,l} = 1$ si les flots de classe i traversent le lien l, et 0 sinon. Enfin, on note $\phi_i(\boldsymbol{x})$ le débit alloué à la classe i dans l'état $\boldsymbol{x}$.

Nous supposons ici que le débit alloué à une classe est équitablement partagé entre les différents flots actifs de cette classe. Le système est alors équivalent à un réseau de files d'attente PS dont les taux de service évoluent en fonction de l'état $\boldsymbol{x}$ du système. A titre d'exemple, considérons le réseau linéaire de la figure 2.4 constitué de deux liens et trois classes. Comme illustré sur la figure 2.5, pour un état $\boldsymbol{x}$ donné, le système est équivalent à un réseau de trois files d'attente à processeur partagé : la file $i = 1, 2, 3$ de capacité $\phi_i(\boldsymbol{x})$ reçoit les flots appartenant à la classe i.

Il s'avère extrêmement difficile de déterminer une expression exacte du débit alloué à chaque classe. En effet, la manière dont TCP effectue le partage des ressources entre les flots élastiques ne peut probablement pas être connue exactement. Par rapport au cas d'un lien unique, ce cas s'avère bien plus complexe à modéliser.

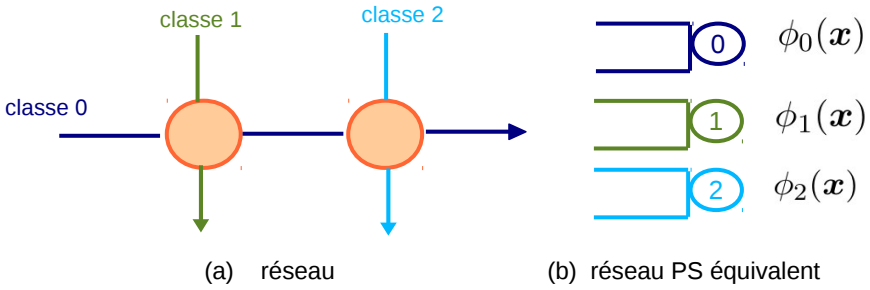


FIGURE 2.5 – Réseau PS.

De nombreux travaux [62, 71] ont cependant proposé d'approximer le partage de ressources réalisé par TCP par des allocations théoriques optimisant des fonctions dites d'utilité. Étant donné un paramètre $\alpha \geq 0$, une allocation de débit α-équitable dans l'état $\boldsymbol{x}$ correspond au vecteur $\boldsymbol{\phi}(\boldsymbol{x})$ solution optimale du problème suivant :

$$\text{maximiser}_{\boldsymbol{\phi}} \sum_{i \in \mathcal{E}: x_i > 0} x_i^{\alpha} \frac{\phi_i^{1-\alpha}}{1-\alpha} \qquad (\alpha\text{-fair})$$

sous les contraintes :

$$\sum_{i \in \mathcal{E}} a_{l,i} \phi_i \leq C_l, \quad \forall l = 1, \ldots, L \qquad (2.3)$$

$$\phi_i \geq 0, \quad \forall i \in \mathcal{E} \qquad (2.4)$$

On retrouve les allocations classiques : allocation maximisant le débit total ($\alpha = 0$), équité max-min ($\alpha \to \infty$) et équité proportionnelle ($\alpha = 1$). Nous détaillons ci-dessous ces allocations, toujours sur l'exemple d'un réseau linéaire tel que celui de la figure 2.4.

Allocation maximisant le débit total ($\alpha = 0$)

Cette politique vise à maximiser la performance globale du réseau en maximisant la somme totale des bandes passantes allouées aux flots actifs : $max_{\boldsymbol{\phi}} \sum_{i \in \mathcal{E}} \phi_i$. L'inconvénient majeur de cette politique de partage est qu'elle peut mener à des allocations non équitables. En effet, considérons l'exemple du réseau linéaire illustré sur la figure 2.4. L'objectif se réduit alors à maximiser $\phi_0 + \ldots + \phi_L$ avec $\phi_i \geq 0$ pour $i = 0, 1, \ldots, L$

sous les contraintes de capacité suivantes :

$$\phi_l + \phi_0 \leq C_l, \quad \forall l = 1, 2, \ldots, L. \tag{2.5}$$

Il est évident qu'en l'absence d'autres contraintes, l'algorithme utilise toute la bande passante disponible, et que par conséquent $\phi_0 + \phi_l = C_l$ pour tout lien l. On déduit que,

$$\sum_{i=0}^{L} \phi_i = \phi_0 + \sum_{l=1}^{L} (C_l - \phi_0) = (1 - L)\phi_0 + \sum_{l=1}^{L} \phi_l \tag{2.6}$$

Ainsi, pour tout réseau linéaire composé de $L \geq 1$ liens, maximiser le débit total revient à minimiser ϕ_0. La solution de ce problème dans ce cas est évidemment $\phi_0 = 0$ et $\phi_i = C_i$ pour $i = 1, \ldots, L$. On voit bien sur cet exemple que le critère de maximisation du débit total peut aboutir à des allocations loin d'être équitables.

Équité max-min ($\alpha \to \infty$)

L'équité max-min a été initialement utilisée en sciences économiques et sociales [57] et fut introduite dans les réseaux de télécommunications par Bertsekas et Gallager [22]. Contrairement aux politiques de partage de bande passante maximisant le débit total, la politique équité max-min vise à allouer aux flots actifs les allocations les plus proches possibles les unes des autres tout en restant optimal au sens de Pareto [56]. En d'autres termes, la vitesse de transmission d'un flot ne peut pas être augmentée sans réduire le débit d'un autre flot. L'allocation globale au niveau du réseau est évidemment inférieure à celle obtenue avec l'allocation maximisant le débit total, l'allocation équité max-min visant à atteindre une certaine équité entre les flots. L'algorithme permettant de calculer l'allocation max-min est un algorithme de type water-filling [9] que l'on peut résumer ainsi :

1. Initialiser la bande passante à zéro pour tous les flots : $\phi_i = 0$, $\forall i \in \mathcal{E}$,
2. Augmenter équitablement la bande passante allouée aux flots en faisant $\phi_i = r$ pour tout i jusqu'à ce qu'un lien soit saturé, c'est à dire jusqu'à ce que :

$$r = \min_l \frac{C_l}{\sum_{i \in \mathcal{E}} a_{i,l}}$$

3. Réitérer l'étape 2 pour les flots qui ne sont pas encore contraints en faisant

$$C_l \leftarrow C_l - \sum_{i \in \mathcal{E}} a_{i,l}\phi_i \quad \text{et} \quad \mathcal{E} \leftarrow \mathcal{E} \setminus \{i \ : \ \exists l \in r_i, \ C_l = 0\}$$

jusqu'à ce qu'on ne puisse plus augmenter la bande passante d'aucun flot.

Bien que la politique de partage équité max-min soit Pareto optimale, elle favorise trop les flots longs et n'utilise pas de façon assez efficace la bande passante. En effet,

si nous revenons à l'exemple d'un réseau linéaire et en supposant que tous les liens ont la même capacité C, chaque flot reçoit un débit égal à $C/2$. Ainsi, le flot 0 reçoit dans l'allocation max-min, la moitié de la bande passante disponible sur chaque lien, soit autant qu'un flot court, ce qui minimise le débit total. En effet, le débit total dans ce cas est égal à $(L+1)\,C/2$, soit approximativement la moitié du débit total $L\,C$ obtenu avec l'allocation optimisant le débit du réseau.

Équité proportionnelle ($\alpha = 1$)

Une allocation intermédiaire dite équité proportionnelle a été introduite pour garantir un certain compromis entre optimisation du débit du réseau et équité du partage de bande passante entre flots. L'équité proportionnelle est un cas particulier du concept d'arbitrage de Nash [74] et fut utilisée dans les réseaux de télécommunications par Kelly [44]. Cette allocation maximise la fonction d'utilité U suivante où pour chaque état $\boldsymbol{x}$, $\boldsymbol{\phi}(\boldsymbol{x})$ est définie comme l'unique vecteur $\boldsymbol{\phi}$ maximisant

$$U(\boldsymbol{x}, \boldsymbol{\phi}) = \sum_{i \in \mathcal{E}} \log(\phi_i(\boldsymbol{x})) \tag{2.7}$$

sous les contraintes (2.3)-(2.4). L'avantage clé de cette allocation est de garantir un bon compromis entre équité et optimalité. En effet, l'équité proportionnelle offre davantage de bande passante aux flots courts que l'équité max-min, donnant ainsi une meilleure utilisation de la bande passante tout en maintenant une certaine équité. Prenons à nouveau l'exemple du réseau linéaire avec des liens identiques de capacité C. L'objectif se réduit alors à maximiser $\log(\phi_0) + \ldots + \log(\phi_L)$ avec $\phi_i \geq 0$ pour $i = 0, 1, \ldots, L$ sous les contraintes de capacité $\phi_0 + \phi_l \leq C$ pour $l = 1, \ldots, L$. Il est évident qu'en l'absence d'autres contraintes, l'algorithme utilise toute la bande passante disponible : $\phi_l + \phi_0 = C$ pour tout l. Le problème revient alors à maximiser,

$$\max_{\phi_0} \left(log(\phi_0) + \sum_{i=1}^{L} \log(C - \phi_0) \right) \tag{2.8}$$

En posant

$$f(x) = \left(log(x) + \sum_{i} \log(C - x) \right), \tag{2.9}$$

on voit que la solution optimale est obtenue pour ϕ_0 tel que $f^{'}(\phi_0) = 0$, ce qui donne

$$\phi_0 = \frac{C}{L+1}. \tag{2.10}$$

À titre de comparaison, le tableau 2.1 récapitule les résultats obtenus pour les trois

	ϕ_0	Débit total
maximisation du débit total	0	L C
max-min	C/2	(1+L) C/2
équité proportionnelle	C/(L+1)	$(1+L^2)C/(L+1)$

TABLE 2.1 – Comparaison de 3 allocations α équitables dans le cas d'un réseau linéaire.

allocations sur l'exemple du réseau linéaire à liens identiques.

On voit sur cet exemple que l'équité proportionnelle favorise moins les flots longs que l'équité max-min, donnant ainsi un débit total plus élevé. En effet, le débit total obtenu avec l'équité proportionnelle $(1+L^2)C/(1+L)$ est supérieur au débit total obtenu avec l'équité max-min qui est de $(1+L)C/2$. D'autre part, cet exemple montre que l'équité proportionnelle permet d'obtenir pour ϕ_0 un débit $C/(L+1)$ supérieur au débit nul obtenu en optimisant le débit total du réseau, donnant ainsi une meilleure équité entre les flots. On en déduit que l'équité proportionnelle réalise un meilleur compromis entre équité de l'allocation de débit et optimisation du débit total.

Le problème majeur des allocations α-équitables, et en particulier de l'équité proportionnelle, est qu'elles sont sensibles aux caractéristiques détaillées du trafic comme par exemple les moments d'ordre $k \geq 2$ de la taille des fichiers échangés. Outre les difficultés d'analyse qu'elles posent, l'utilisation des résultats obtenus avec ces allocations est délicate car en général les opérateurs de réseaux n'ont qu'une connaissance imprécise du trafic et ne sont donc pas en mesure de fournir les paramètres du modèle. Il s'avère toutefois que l'équité proportionnelle peut être approximée par l'allocation dite d'équité équilibrée (BF, pour *balanced fairness*) [27], [30], [31] dont l'intérêt majeur est d'être insensible aux caractéristiques fines du trafic (en d'autres termes, elle ne dépend que du trafic moyen), permettant ainsi d'obtenir des métriques de performance robustes aux incertitudes sur les caractéristiques du trafic. Dans [67], Massoulié a notamment montré que l'équité équilibrée est asymptotiquement équivalent à l'équité proportionnelle.

Equité équilibrée

Bonald et Proutière ont introduit la notion d'équité équilibrée qui est l'unique allocation insensible optimisant un certain critère. L'insensibilité est une propriété très utile en pratique car elle permet de s'affranchir de la connaissance des caractéristiques fines du trafic. Il n'est même pas nécessaire de supposer que le processus d'arrivée des flots suit un processus de Poisson. Il suffit juste de supposer que les sessions arrivent selon un processus de Poisson [31]. Comme souligné plus haut, cette hypothèse est toujours valide pour le trafic Internet.

Sa formulation repose sur les réseaux de Whittle [87], qui sont une extension des réseaux de Jackson [55] où le taux de service dépend de l'état. Les réseaux de Whittle,

représentent la classe des réseaux PS la plus générale ayant la propriété d'insensibilité. Un réseau de files d'attente PS est un réseau de Whittle si et seulement si la condition d'équilibre ou de balance des débits est satisfaite. Les débits sont dits équilibrés si pour tout classe i, j, la condition suivante est vérifiée pour chaque état $\boldsymbol{x}$ tel que $x_i > 0$ et $x_j > 0$ pour $i, j = 1...L + 1$:

$$\frac{\phi_i(\boldsymbol{x} - e_j)}{\phi_i(\boldsymbol{x})} = \frac{\phi_j(\boldsymbol{x} - e_i)}{\phi_j(\boldsymbol{x})} \tag{2.11}$$

Cette condition d'équilibre implique que le changement relatif de la capacité allouée à la classe i si on enlève un flot de classe j est égal au changement relatif de la capacité allouée à la classe j si on enlève un flot de classe i.

L'allocation équilibrée est définie à l'aide de la fonction de balance Φ par

$$\phi_i(\boldsymbol{x}) = \frac{\Phi(\boldsymbol{x} - e_i)}{\Phi(\boldsymbol{x})}, \tag{2.12}$$

pour tout état $\boldsymbol{x} \in \mathbb{N}^{L+1}$ tel que $x_i > 0$.

L'équité équilibrée correspond au choix d'une fonction de balance respectant les contraintes de capacité et maximisant l'utilisation des ressources. Elle est définie de manière récursive avec $\Phi(0) = 1$, et

$$\Phi(\boldsymbol{x}) = \max \left\{ \max_{l \in \mathcal{L}} \frac{1}{C_l} \sum_{i \in \mathcal{E}} \Phi(\boldsymbol{x} - e_l) a_{i,l}, \max_{i \in \mathcal{E}} \frac{\Phi(\boldsymbol{x} - e_i)}{c_i x_i} \right\} \tag{2.13}$$

pour tout état $\boldsymbol{x} \in \mathbb{N}^{L+1}$ tel que $x_i > 0$ avec $\Phi(\boldsymbol{x}) = 0$ si $\boldsymbol{x} \notin \mathbb{N}^{L+1}$.

Malgré leurs différences, l'équité équilibrée et les allocations α-équilibrées donnent des résultats de performance très proches comme illustré par l'exemple suivant. On considère un réseau linéaire (cf. 2.4) avec 3 liens et 4 classes. On suppose que $C_1 = 20$, $C_2 = 15$ et $C_3 = 25$. On augmente progressivement le trafic total dans le réseau, en supposant la répartition de charge suivante : 10% pour la classe 0, 20% pour les classes 1 et 2 et 50% pour la classe 3. La figure 2.6 montre l'évolution du débit des différentes classes pour les allocations équité équilibrée, équité proportionnelle et équité max-min. On observe que les résultats sont assez proches pour les trois allocations de débit, l'équité équilibrée fournissant une bonne approximation de l'équité proportionnelle et de l'équité max-min.

L'avantage clé de l'équité équilibrée est de conduire à une évaluation analytique des performances. Cependant, comme nous le verrons au chapitre 3, l'évaluation numérique des performances n'est faisable que pour des réseaux de petite taille du fait de l'explosion combinatoire de l'espace d'états. Pour des réseaux plus complexes, de nouvelles approches sont nécessaires. C'est l'intérêt des travaux présentés au chapitre 3.

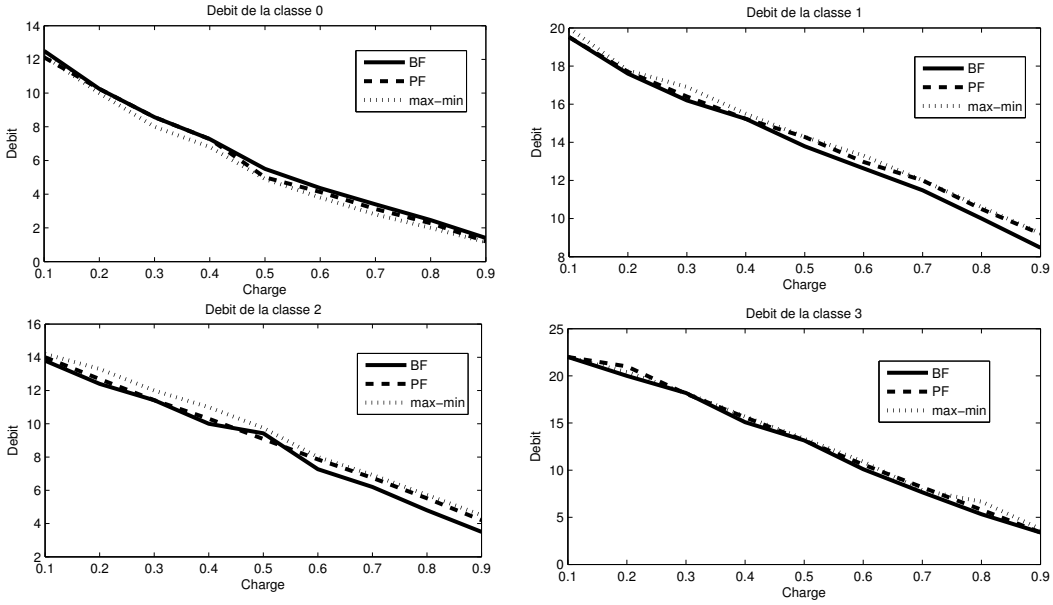


FIGURE 2.6 – Débit des 4 classes obtenus avec les allocations équité équilibrée (BF), équité proportionnelle (PF) et équité max-min.

2.5 Partage de bande passante entre flots de streaming et flots élastiques

Lorsque le trafic streaming partage la capacité d'un lien avec du trafic élastique, le trafic streaming réussit généralement à émettre à son débit intrinsèque, le trafic élastique s'adaptant à la présence du trafic élastique. Ceci a été vérifié avec des expérimentations réalisées avec le simulateur NS-2 [75] et détaillées ci-dessous.

Le modèle de simulation consiste en un seul lien de capacité C partagé par des flots élastiques et des flots de streaming. Le générateur de trafic FTP est utilisé pour simuler les flots TCP (élastiques) et le générateur de trafic CBR pour simuler les flots UDP (streaming). On note x_1 (resp. x_2), le nombre de flots élastiques (resp. flots de streaming). Soit d le débit des flots de streaming. On considère deux exemples. Dans le premier exemple, on fixe les paramètres suivants : C=2 Mbps, d=1 Mbps, $x_2 = 1$ et on varie le nombre de flots élastiques x_1. La figure 2.7 montre l'évolution du débit des flots en fonction du nombre de flots élastiques présents. On observe que le flot de streaming a toujours la priorité dans le sens où les flots élastiques s'adaptent pour se partager la bande passante résiduelle de 1 Mbps laissée disponible par le flot de streaming.

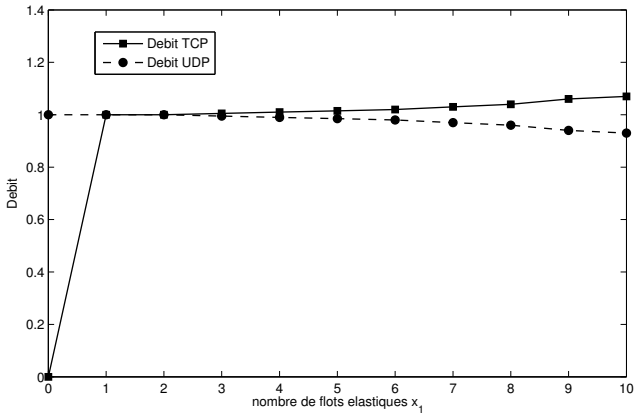


FIGURE 2.7 – Débit du flot de streaming et débit agrégé des flots TCP en fonction du nombre de flots élastiques.

Dans le deuxième exemple, on fixe les paramètres suivants : C=10 Mbps, $x_1 = x_2 = 1$ et on varie le débit d du flot de streaming. La figure 2.8 présente l'évolution du débit des deux flots en fonction de d. Comme le flot de streaming est toujours prioritaire, le débit du flot élastique diminue avec l'augmentation de d. De plus, on observe que si le débit du flot de streaming est proche de la capacité du lien, le débit des flots élastiques s'approche de zéro. Ce scénario met donc en évidence le besoin d'un contrôle d'admission approprié pour les flots de streaming afin d'assurer un débit minimum aux flots élastiques

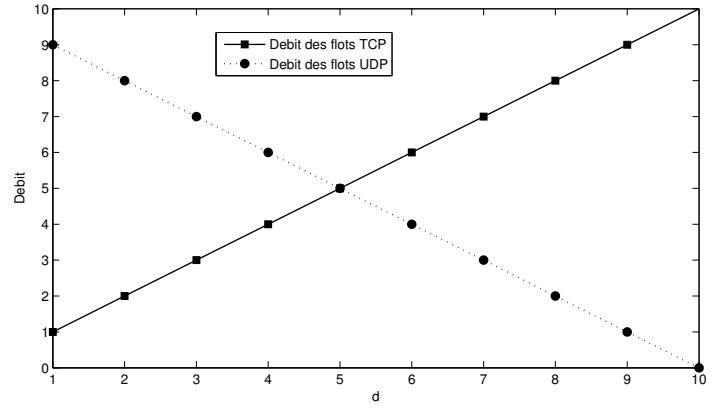


FIGURE 2.8 – Débit des deux flots en fonction du débit d du flot de streaming.

2.5.1 Obtention de bornes insensibles

L'analyse des réseaux à partage de bande passante peut généralement se ramener à l'étude d'un réseau de files d'attente PS couplées, la classe i étant servie au noeud i avec le taux de service $\phi_i(\boldsymbol{x})$ lorsque le système est dans l'état $\boldsymbol{x}$. Dans certaines situations, comme pour le partage de bande passante entre trafic élastique et trafic de streaming (cf. Chapitre 3), l'allocation de débit ne vérifie pas la condition d'équilibre (2.11), qui est une condition nécessaire d'insensibilité. Dans de telles situations, l'analyse des performances peut s'avérer potentiellement très complexe et l'utilisation pratique des résultats obtenus devient difficile car elle nécessite des informations détaillées sur les caractéristiques du trafic.

Toutefois, en suivant l'approche décrite dans [30], des bornes insensibles sur les performances peuvent être obtenues à condition que la *propriété de monotonicité* suivante soit vérifiée

$$\phi_i(\boldsymbol{x} - e_j) \geq \phi_i(\boldsymbol{x}), \tag{2.14}$$

pour tout i, j et pour tout état $\boldsymbol{x} = (x_1, ...x_k)$ tel que $x_i > 0$ et $x_j > 0$. La propriété de monotonicité implique que si on enlève un flot d'un nœud PS cela ne peut pas diminuer le débit d'un autre nœud. Cette propriété est clairement vérifiée dans de nombreux systèmes réels.

Sous cette condition, les auteurs de [30] propose d'encadrer l'état du système de la façon suivante

$$\boldsymbol{x}^-(t) \leq \boldsymbol{x}(t) \leq \boldsymbol{x}^+(t); \tag{2.15}$$

où $\boldsymbol{x}^-(t)$ et $\boldsymbol{x}^+(t)$ représentent les vecteurs d'état à l'instant t de deux systèmes de files d'attente PS pour lesquels la condition (2.11) est vérifiée, et donc pour lesquels la distribution stationnaire est insensible. Les bornes supérieure et inférieure $\boldsymbol{x}^-(t)$ et $\boldsymbol{x}^+(t)$ sont entièrement caractérisées par les fonctions de balance associées Φ^- and Φ^+ [30]. Il reste tout de même difficile de trouver des expressions explicites de ces fonctions sauf dans le cas de réseaux vérifiant la *propriété de biais* suivante :

$$\frac{\phi_i(\boldsymbol{x} - e_j)}{\phi_i(\boldsymbol{x})} \leq \frac{\phi_j(\boldsymbol{x} - e_i)}{\phi_j(\boldsymbol{x})}. \tag{2.16}$$

pour tout i, j tel que $i \leq j$. Pour un réseau de k nœuds vérifiant cette propriété, les expressions des fonctions de balance associées à la borne supérieure et à la borne inférieure s'écrivent respectivement

$$\Phi^-(\boldsymbol{x}) = \left(\prod_{i=1}^{x_k} \phi_k(i\mathbf{e}_k) ... \times \prod_{i=1}^{x_1} \phi_1(i\mathbf{e}_1 + \sum_{j=2}^{k} x_j \mathbf{e}_j) \right)^{-1}, \tag{2.17}$$

et,

$$\Phi^+(\boldsymbol{x}) = \left(\prod_{i=1}^{x_1} \phi_1(i\mathbf{e}_1) \times ... \times \prod_{i=1}^{x_k} \phi_k(i\mathbf{e}_k + \sum_{j=1}^{k-1} x_j \mathbf{e}_j) \right)^{-1}. \tag{2.18}$$

Les distributions stationnaires des vecteurs d'état $\boldsymbol{x}^-(t)$ and $\boldsymbol{x}^+(t)$ sont alors données par

$$\pi^-(\boldsymbol{x}) = \pi^-(0)\Phi^-(\boldsymbol{x}) \prod_{i=1}^{k} \rho_i^{x_i}, \tag{2.19}$$

et,

$$\pi^+(\boldsymbol{x}) = \pi^+(0)\Phi^+(\boldsymbol{x}) \prod_{i=1}^{k} \rho_i^{x_i}. \tag{2.20}$$

Nous verrons au Chapitre 3 comment on peut utiliser cette technique pour obtenir des bornes insensibles sur les performances des flots dans le cas où la bande passante est partagée entre flots de streaming et flots élastiques.

2.6 Ordonnancement optimal

Dans les paragraphes précédents, nous avons vu plusieurs modèles de partage de bande passante qui permettent de représenter plus ou moins fidèlement le partage réalisé par TCP. L'analyse stochastique de ces modèles se révèle plus ou moins complexe, et les résultats de cette analyse sont plus ou moins exploitables en pratique suivant qu'ils sont insensibles ou non. De manière plus générale, de nombreux travaux se sont intéressé à l'ordonnancement optimal des clients dans les réseaux de file d'attente.

La plupart des articles se focalisent sur le cas d'un seul serveur, en supposant que l'objectif est de minimiser le temps de réponse moyen du serveur [73], [37], [81], [7]. En général, la politique optimale est une politique prioritaire donnant la priorité à une classe choisie en fonction d'un certain critère. Une politique classique dite SRPT (pour *Shortest Remaining Processing Time*) [7] alloue la totalité de la capacité du serveur au client dont le temps de service résiduel est minimal. Schrage a montré l'optimalité de SRPT pour minimiser le temps moyen de réponse des clients pour le cas d'un serveur avec une seule classe de trafic [86] (voir également [15]).

Même si cette politique améliore la performance globale du système, son application à des systèmes réels est complexe car SRPT suppose la connaissance *a priori* des temps d'exécution. En pratique, l'information sur les temps d'exécution n'est généralement pas disponible. Par conséquent, d'autre travaux se sont intéressés aux politiques non-

anticipatives [4], comme par exemple la politique LAS (Least Attained Service) qui alloue toute la capacité à la tâche ayant été le moins servie. Pour une file d'attente $M/G/1$, la politique optimale parmi les politiques non-anticipatives pour minimiser le nombre moyen de clients correspond à une règle de priorité basée sur les indices de Gittins (voir par exemple [7]). Enfin, dans le cas d'un serveur multiclasse, plusieurs travaux [73] [37], [81] ont prouvé l'optimalité de la règle $c\mu$ pour minimiser l'espérance d'une somme pondérée du nombre de clients.

L'étude de l'ordonnancement optimal dans les réseaux de files d'attente est beaucoup plus complexe. En ce qui concerne les réseaux à partage de bande passante, très peu de travaux se sont intéressés à l'étude de la politique optimale permettant de minimiser le nombre de flots en cours. Les résultats obtenus sont valides pour des topologies particulières, comme les réseaux linéaires [90] ou bien les réseaux en étoile [88]. Dans le chapitre 4, nous nous intéressons à l'ordonnancement des flots dans un réseau, mais en supposant que l'objectif est d'optimiser un compromis entre énergie consommée et nombre de flots dans le système. A notre connaissance, cette problématique n'a été étudiée que pour un seul lien et une seule classe de trafic dans [10].

2.7 Conclusion

Dans le présent chapitre, nous n'avons donné qu'un aperçu sur les travaux existant sur la modélisation mathématique du trafic IP. Malgré les résultats déjà obtenus dans la littérature, l'étude n'en est qu'à ses débuts. Notre étude des travaux existants dans la littérature révèle notamment des insuffisances qui nous ont mené à proposer les résultats présentés dans les chapitres suivants.

4. qui n'ont aucune information *a priori* sur le temps d'exécution.

3

Évaluation de performance du trafic Internet

Résumé

Ce chapitre est consacré à l'évaluation des performances du trafic Internet. Nous considérons tout d'abord le cas où un ensemble de flots élastiques se partagent les ressources du réseau suivant le mécanisme d'équité équilibrée. Nous proposons des approximations simples et explicites de l'espérance du débit des flots élastiques. Nous étudions ensuite la situation dans laquelle les flots élastiques sont en compétition avec des flots de streaming pour l'accès aux ressources du réseau, en supposant qu'un mécanisme de contrôle d'admission est utilisé pour limiter la bande passante prise par les flots de streaming. Nous proposons dans ce cas des résultats de performance exacts pour les flots streaming et des approximations pour les flots élastiques. Les simulations montrent toutefois la bonne qualité des approximations proposées.

Sommaire

3.1 Introduction

Les réseaux de données tels qu'Internet assurent l'acheminement des messages d'une source vers une destination grâce à un ensemble de protocoles standardisés. Le transfert d'un message sur Internet est généralement réalisé par l'envoi de plusieurs paquets IP. Les paquets de données appartenant au même flot sont chacun acheminés à travers un chemin de bout en bout calculé grâce à un ou plusieurs protocoles de routage [25]. Tout au long de la route, les paquets traversent plusieurs routeurs et switchs qui sont liés par des liens physiques ou bien logiques. Les paquets de différents flots sont alors en concurrence pour l'accès aux ressources de transmission. On s'intéresse ici à la modélisation du partage de bande passante entre les différents flots de communication.

L'augmentation continue du trafic [5] (cf. figure 3.1) et sa forte variabilité peuvent entrainer la congestion de certains liens dont le résultat est une détérioration des performances perçues par les utilisateurs.

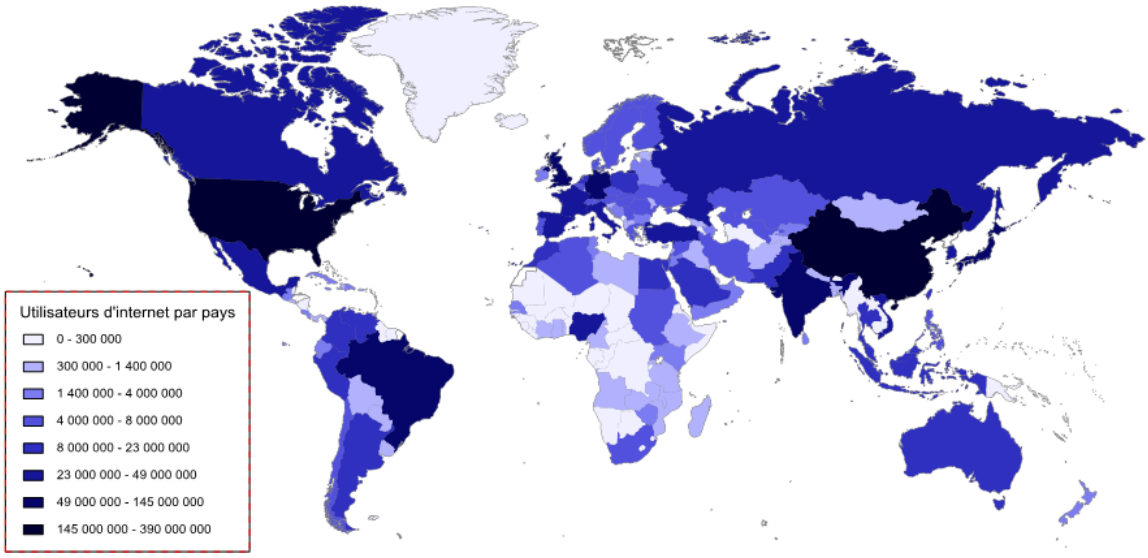


FIGURE 3.1 – Nombre d'utilisateurs Internet par pays en 2012.

Pour éviter que de tels évènements ne se produisent et pour fournir toujours le même niveau de service, les opérateurs ont besoin de méthodes rigoureuses pour adapter la capacité de leurs réseaux aux évolutions du trafic. La surveillance continue du réseau est également primordiale pour détecter le plus rapidement possible toute dégradation des performances et pouvoir reconfigurer le réseau pour y faire face. À ce niveau, les

mesures de bout en bout s'avèrent extrêmement complexes à mettre en œuvre. Les opérateurs de réseaux ont alors besoin des outils permettant d'estimer ou de prédire les performances de leurs infrastructures de communication en temps réel. C'est ce qui a motivé les travaux présentés dans le présent chapitre qui propose de modéliser au niveau flot le trafic internet afin d'évaluer ses performances.

Comme décrit au chapitre 2, les modèles de niveau flot ont été initialement considérés par Massoulié et Roberts pour évaluer les performances du trafic Internet [84]. De nombreux travaux [62, 71] ont proposé d'approximer le partage de ressources réalisé par TCP par des allocations théoriques α-équitables optimisant une fonction d'utilité : ce sont par exemple les notions d'équité max-min ($\alpha \to \infty$), d'équité proportionnelle ($\alpha = 1$) ou de maximisation du débit total ($\alpha = 0$). Comme on l'a vu, le problème majeur des allocations α-équitables est qu'elles sont sensibles aux caractéristiques détaillées du trafic et ne conduisent donc pas à des formules robustes directement utilisables par les opérateurs de réseaux. C'est l'avantage essentiel de l'allocation "équité équilibrée" (BF, pour balanced fairness) introduite par Bonald et Proutière [31] [26], qui permet en outre une évaluation analytique des performances.

Toutefois, l'équité équilibré reste complexe à utiliser dans un contexte pratique car elle requiert le calcul de la probabilité de chacun des états possibles du système, et est donc confrontée à l'explosion combinatoire de l'espace d'états pour de grands réseaux. Dans ce contexte, il est primordial de proposer des solutions permettant de calculer efficacement les métriques de performance (ou bien des approximations ou encore des bornes sur ces métriques) sans nécessiter l'évaluation des probabilités individuelles des états.

Dans [31], Bonald et al. proposent un algorithme récursif qui calcule efficacement les métriques de performance pour des réseaux particuliers dans lesquels il est possible d'identifier les liens saturés du réseau pour chaque état du système. Bien que l'algorithme permette de calculer d'une manière exacte les métriques de performance, il n'est applicable que sur des cas particuliers simples. Pour des réseaux plus complexes, l'identification des liens saturés n'est pas toujours faisable.

Une autre approche a été proposée dans [28] par Bonald et al. pour résoudre ce problème. Sous l'hypothèse que les flots n'ont pas de débit crête, les auteurs proposent des approximations explicites des principales métriques de performances dans des réseaux de topologie quelconque. En pratique, les flots ont généralement un débit crête qui est typiquement fonction de la ligne d'accès des utilisateurs.

Dans [26], Bonald et Proutière proposent des bornes stochastiques sur le débit moyen des flots élastiques lorsqu'ils ont un débit crête. Nous proposons dans la section 3.2 une approximation explicite simple pour évaluer les performances des flots élastiques. Les résultats numériques présentés au paragraphe 3.2 montrent que l'approximation proposée est plus précise que les bornes stochastiques de [26].

Nous nous intéressons également au partage de bande passante entre flux élastiques

et flux de streaming. Peu de travaux ont été consacrés à l'évaluation de performance dans ce contexte. Dans [30], Bonald et Proutière proposent des bornes insensibles sur les performances des flots dans un réseau où les flots de streaming sont TCP-friendly et partagent équitablement la bande passante avec les flots élastiques. Les auteurs de [42], [59], [80], [19] s'intéressent à l'évaluation de performance des flots dans un réseau où les flots de streaming sont non adaptatifs et prioritaires. Delcoigne et al. [42] justifie notamment la nécessité d'un mécanisme de contrôle d'admission approprié pour les flots de streaming afin de garantir un minimum de débit pour les flots élastiques. À notre connaissance, les travaux existants sur une telle intégration se focalisent uniquement sur le modèle de partage d'un lien unique. Nous proposons au paragraphe 3.3 une approximation pour étendre ces résultats au cas de plusieurs liens partagés par des flots hétérogènes (système multiclasse [32]).

3.2 Modèle de partage élastique

3.2.1 Hypothèses et notations

Le modèle consiste en un ensemble de liens $\mathcal{L} = \{1, ..., K\}$. On note C_l la capacité du lien l exprimée en bit/s et $\mathbf{C}$ le vecteur associé. Un certain nombre de flots sont en compétition pour le partage de la bande passante de ces liens. On suppose ici que le trafic est purement élastique. Soit $\mathcal{E} = \{1, ..., M\}$ l'ensemble des classes de flots élastiques. On note c_i le débit crête de la classe i et $\mathbf{c} = (c_1, \ldots, c_M)$ le vecteur associé. A chaque classe $i \in \mathcal{E}$ est également associée une route $r_i \in \mathcal{L}$. On désigne par A la matrice d'incidence tel que $a_{i,l} = 1$ si les flots de classe i utilisent le lien $l \in \mathcal{L}$, et 0 sinon. On suppose de plus que les flots de classe $i \in \mathcal{E}$ arrivent selon un processus de Poisson de moyenne λ_i et que leur volume suit une loi aléatoire de moyenne $\frac{1}{\mu_i}$. L'intensité du trafic de la classe i est alors donnée par le produit $\rho_i = \lambda_i/\mu_i$ du taux d'arrivée des flots par leur volume moyen, exprimée en bit/s. On note $\boldsymbol{\rho}$ le vecteur $\boldsymbol{\rho} = \{\rho_i\}_{i\in\mathcal{E}}$.

Soit x_i le nombre de flots de classe i en cours ; on note $\boldsymbol{x}$ le vecteur associé. On désigne par $\phi_i(\boldsymbol{x})$ le débit de la classe $i \in \mathcal{E}$ dans l'état $\boldsymbol{x}$ et par $\phi(\boldsymbol{x})$ le vecteur associé. Par ailleurs, nous considérons que les flots de classe $i \in \mathcal{E}$, se partagent les ressources de manière équitable, chacun recevant le même débit $\phi_i(\boldsymbol{x})/x_i$ dans l'état $\boldsymbol{x}$. C'est l'équivalent d'un réseau de files d'attente PS.

Le vecteur $\boldsymbol{x}$ définit un processus de naissances et de morts de dimension $|\mathcal{E}|$, avec taux de naissance λ_i et taux de mort $\mu_i\phi_i(\boldsymbol{x})$ pour la classe i dans l'état $\boldsymbol{x}$. La mesure invariante de ce processus est donnée par [29]

$$\pi(\boldsymbol{x}) = \pi(0)\Phi(\boldsymbol{x})\boldsymbol{\rho}^{\boldsymbol{x}}, \tag{3.1}$$

où Φ désigne la fonction de balance [cf. (2.13)], définie récursivement par $\Phi(0) = 1$ et

$$\Phi(\boldsymbol{x}) = \max\left\{\max_{l\in\mathcal{L}}\frac{1}{C_l}\sum_{i\in\mathcal{E}}\Phi(\boldsymbol{x}-e_l)a_{i,l}, \max_{i\in\mathcal{E}}\frac{\Phi(\boldsymbol{x}-e_i)}{c_i x_i}\right\}. \tag{3.2}$$

En désignant par $\theta_l = \sum_{i\in\mathcal{E}}\rho_i a_{i,l}$ le trafic offert à un lien $l \in \mathcal{L}$, une condition nécessaire de stabilité du système est $\theta_l \leq C_l$, pour chaque lien $l \in \mathcal{L}$. Nous supposerons cette condition satisfaite dans la suite.

Nous nous intéressons ici à la distribution du nombre de flots en cours pour chaque classe. Cette distribution permet le calcul des performances de chaque classe de trafic. En effet, la durée moyenne d'un flot de classe i s'obtient à partir du nombre moyen $E[x_i]$ de flots de classe i et de leur taux d'arrivée λ_i par la loi de Little : $t_i = E[x_i]/\lambda_i$; de même, le débit moyen des flots de classe i est donné par $\rho_i/E[x_i]$.

Notons que théoriquement, on peut calculer à partir de (3.1)-(3.2) la probabilité $\pi(\boldsymbol{x})$ de chaque état $\boldsymbol{x}$, et par conséquent les métriques de performance. Ces expressions ne sont hélas calculables que pour des cas simples. En effet, le calcul direct des métriques de performance, en utilisant ces formules, est confronté au problème de l'explosion combinatoire lorsque le nombre de flots devient important ou la charge du réseau est importante. Nous détaillons dans la section suivante, la solution que nous proposons pour résoudre le problème de l'explosion combinatoire de l'espace d'états du système.

3.2.2 Résultats pour un lien isolé

On se focalise ici sur un lien unique partagé par un ensemble de flots élastiques ; ce cas nous permettra par la suite d'obtenir des intuitions sur les performances de flots élastiques pour d'autre topologies. On considère donc un lien isolé de capacité C partagée par M classes de flots, (cf. figure 3.2). On notera $\theta = \sum_i \rho_i$.

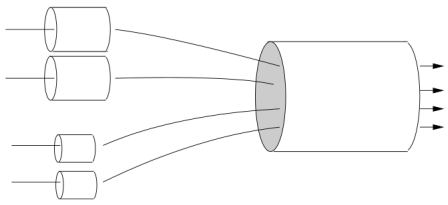


FIGURE 3.2 – Lien de capacité C.

Dans ce cas, la fonction de balance a l'expression suivante [32] :

$$\Phi(\boldsymbol{x}) = \begin{cases} \prod_{k\in\mathcal{E}}\frac{c_k^{-x_k}}{x_k!} & \text{si } \mathbf{c}.\boldsymbol{x} \leq C, \\ \frac{1}{C}\sum_{k\in\mathcal{E}}\Phi(\boldsymbol{x}-\mathbf{e}_k) & \text{sinon,} \end{cases} \tag{3.3}$$

d'où l'on déduit la distribution stationnaire de l'état via (3.1).

Comme démontré dans [51] (cf. Proposition 4.5.1), dans tout état $\boldsymbol{x}$ tel que $\mathbf{c}.\boldsymbol{x} > C$, aucun flot n'est servi à son débit crête, c'est-à-dire que $\phi_i(\boldsymbol{x}) < c_i x_i$ pour tout $i \in \mathcal{E}$. Nous définissons donc l'ensemble $\mathcal{B} = \left\{\boldsymbol{x} \in \mathbb{N}^M \ : \ \mathbf{c}.\boldsymbol{x} \geq C\right\}$ et remarquons que l'on a $\sum_i \phi_i(\boldsymbol{x}) = C$ pour tout état $\boldsymbol{x} \in \mathcal{B}$ et seulement dans ces états là. Par la suite, nous nous référerons donc à $\mathcal{B}$ comme l'événement de congestion du lien. En adaptant légèrement un résultat démontré par Bonald et Virtamo dans [32] (cf. Lemme 4.5.1 de [51]), on obtient le lemme suivant.

Lemme 1. *On a*

$$\pi\left(\mathcal{B}\right) = \sum_{i \in \mathcal{E}} \frac{\rho_i}{C - \theta} \pi\left(\mathcal{W}_i\right), \tag{3.4}$$

où $\mathcal{W}_i = \left\{\boldsymbol{x} \in \mathbb{N}^M \ : \ C - c_i \leq \mathbf{c}.\boldsymbol{x} < C\right\}$.

Démonstration. On remarque tout d'abord que pour tout $\boldsymbol{x}$ tel que $\mathbf{c}.\boldsymbol{x} \geq C$, on a $\sum_k \phi_k(\boldsymbol{x}) = C$. Avec (3.2) et (3.1), on a d'autre part $\pi(\boldsymbol{x}) \sum_k \phi_k(\boldsymbol{x}) = \sum_k \rho_k \pi(\boldsymbol{x} - \mathbf{e}_k)$. On en déduit que

$$\pi(\boldsymbol{x}) = \frac{1}{C} \sum_{k \in \mathcal{E}} \rho_k \pi(\boldsymbol{x} - \mathbf{e}_k), \quad \forall \boldsymbol{x} \in \mathcal{B}. \tag{3.5}$$

il s'ensuit que

$$\begin{aligned}
\pi(\mathcal{B}) &= \sum_{\mathbf{c}.\boldsymbol{x} \geq C} \frac{1}{C} \sum_{k \in \mathcal{E}} \rho_k \pi(\boldsymbol{x} - \mathbf{e}_k) \\
&= \sum_{k \in \mathcal{E}} \frac{\rho_k}{C} \sum_{\mathbf{c}.\boldsymbol{x} \geq C} \pi(\boldsymbol{x} - \mathbf{e}_k) \\
&= \sum_{k \in \mathcal{E}} \frac{\rho_k}{C} \left(\sum_{\mathbf{c}.\boldsymbol{x} \geq C} \pi(\boldsymbol{x}) + \sum_{C - c_k \leq \mathbf{c}.\boldsymbol{x} < C} \pi(\boldsymbol{x}) \right) \\
&= \sum_{k \in \mathcal{E}} \frac{\rho_k}{C} \left(\pi(\mathcal{B}) + \pi(\mathcal{W}_k) \right).
\end{aligned}$$

On conclut ainsi que $\pi\left(\mathcal{B}\right) = \sum_{k \in \mathcal{E}} \frac{\rho_k}{C - \theta} \pi\left(\mathcal{W}_k\right)$. □

Une conséquence évidente du Lemme 1 est formulée dans le Corollaire 1.

Corollaire 1. *Dans le cas d'un seul lien, l'expression de* $\pi(\mathbf{0})$ *est*

$$\pi(\mathbf{0}) = \left(\sum_{\mathbf{c}.\boldsymbol{x} < C} \Phi(\boldsymbol{x}) \boldsymbol{\rho}^{\boldsymbol{x}} + \frac{1}{C - \theta} \sum_{i \in \mathcal{E}} \rho_i \sum_{x \in \mathcal{W}_i} \Phi(\boldsymbol{x}) \boldsymbol{\rho}^{\boldsymbol{x}} \right)^{-1} \tag{3.6}$$

avec $\mathcal{W}_i = \left\{\boldsymbol{x} \in \mathbb{N}^M \ : \ C - c_i \leq \mathbf{c}.\boldsymbol{x} < C\right\}$.

Démonstration. Le résultat s'obtient à partir de $\sum_{\boldsymbol{x}\notin\mathcal{B}}\pi(\boldsymbol{x})+\sum_{\boldsymbol{x}\in\mathcal{B}}\pi(\boldsymbol{x})=1$ en utilisant (3.1) et le Lemme 1. □

Le Corollaire 1 permet de calculer la constante de normalisation du réseau à partir d'un nombre de termes restreint, pour lesquels (3.3) fournit une expression explicite de la fonction de balance.

Comme nous allons le voir, la probabilité de congestion joue un rôle important pour évaluer le débit moyen de chaque classe de trafic.

Débit crête commun

On suppose ici que $c_i = c$ pour tout $i \in \{1,\dots,M\}$. On note $N = C/c$ et on supposera pour simplifier que N est un entier. Dans ce cas, l'événement de congestion du lien est $\mathcal{B} = \left\{\boldsymbol{x}\in\mathbb{N}^M \ : \ |\boldsymbol{x}|\geq N\right\}$. Notons que dans l'état $\boldsymbol{x}$ la classe i obtient un débit donné par $\phi_i(\boldsymbol{x}) = x_i\,\min(c, C/|\boldsymbol{x}|)$. En d'autres termes, chaque flot a son débit crête tant que $|\boldsymbol{x}|\leq N$, alors que pour $|\boldsymbol{x}|>N$ les flots se partagent la capacité du lien suivant la discipline PS ordinaire. On peut alors démontrer le résultat suivant.

Proposition 1. *Le nombre moyen de flots de classe i est donné par*

$$E\left[X_i\right] = \frac{\rho_i}{c} \ + \ b\,\frac{\rho_i}{C-\theta} \tag{3.7}$$

où $b=\pi(\mathcal{B})$ est la probabilité de congestion du lien.

Démonstration. Notons tout d'abord qu'en utilisant (3.1) et (3.2), on obtient

$$\pi(\boldsymbol{x}-\mathbf{e}_i) = \pi(\mathbf{0})\Phi(\boldsymbol{x}-\mathbf{e}_i)\boldsymbol{\rho}^{\boldsymbol{x}-\mathbf{e}_i} = \frac{\phi_i(\boldsymbol{x})}{\rho_i}\,\pi(\mathbf{0})\Phi(\boldsymbol{x})\boldsymbol{\rho}^{\boldsymbol{x}} = \frac{\phi_i(\boldsymbol{x})}{\rho_i}\,\pi(\boldsymbol{x}), \tag{3.8}$$

pour tout $\boldsymbol{x}$ tel que $x_i>0$. A partir de (3.8), on a

$$\sum_{|\boldsymbol{x}|\leq N} x_i\pi(\boldsymbol{x}) = \sum_{|\boldsymbol{x}|\leq N} x_i\frac{\rho_i}{cx_i}\pi(\boldsymbol{x}-\mathbf{e}_i) = \frac{\rho_i}{c}\sum_{|\boldsymbol{x}|\leq N-1}\pi(\boldsymbol{x}) = \frac{\rho_i}{c}(1-\pi(\mathcal{B})). \tag{3.9}$$

En utilisant (3.5), il vient

$$\begin{aligned}
C \sum_{|\boldsymbol{x}|>N} x_i \pi(\boldsymbol{x}) &= \sum_{|\boldsymbol{x}|>N} x_i \sum_{k\in\mathcal{E}} \rho_k \pi(\boldsymbol{x}-\mathbf{e}_k), \\
&= \rho_i \sum_{|\boldsymbol{x}|>N} x_i \pi(\boldsymbol{x}-\mathbf{e}_i) + \sum_{k\neq i} \rho_k \sum_{|\boldsymbol{x}|>N} x_i \pi(\boldsymbol{x}-\mathbf{e}_k), \\
&= \rho_i \sum_{|\boldsymbol{x}|\geq N} (x_i+1)\pi(\boldsymbol{x}) + \sum_{k\neq i} \rho_k \sum_{|\boldsymbol{x}|\geq N} x_i \pi(\boldsymbol{x}), \\
&= \rho_i \sum_{|\boldsymbol{x}|\geq N} \pi(\boldsymbol{x}) + \theta \left(\sum_{|\boldsymbol{x}|=N} x_i \pi(\boldsymbol{x}) + \sum_{|\boldsymbol{x}|>N} x_i \pi(\boldsymbol{x}) \right),
\end{aligned}$$

d'où l'on déduit que

$$\begin{aligned}
\sum_{|\boldsymbol{x}|>N} x_i \pi(\boldsymbol{x}) &= \frac{\rho_i}{C-\theta} \sum_{|\boldsymbol{x}|\geq N} \pi(\boldsymbol{x}) + \frac{\theta}{C-\theta} \sum_{|\boldsymbol{x}|=N} x_i \pi(\boldsymbol{x}) \\
&= \frac{\rho_i}{C-\theta} \sum_{|\boldsymbol{x}|\geq N} \pi(\boldsymbol{x}) + \frac{\rho_i}{c} \frac{\theta}{C-\theta} \sum_{|\boldsymbol{x}|=N-1} \pi(\boldsymbol{x}), \qquad (3.10)
\end{aligned}$$

où la dernière inégalité est obtenue en utilisant le fait que $\pi(\boldsymbol{x}) = \rho_i \pi(\boldsymbol{x}-\mathbf{e}_i)/\phi_i(\boldsymbol{x}) = \rho_i \pi(\boldsymbol{x}-\mathbf{e}_i)/(cx_i)$ pour tout $\boldsymbol{x}$ tel que $|\boldsymbol{x}| \leq N$. Quand $c_i = c$ pour tout $i \in \mathcal{E}$, le Lemme 1 s'écrit simplement $\pi(\mathcal{B}) = \frac{\theta}{C-\theta} \sum_{|\boldsymbol{x}|=N-1} \pi(\boldsymbol{x})$. Par conséquent, on peut réécrire (3.10) sous la forme suivante

$$\sum_{|\boldsymbol{x}|>N} x_i \pi(\boldsymbol{x}) = \frac{\rho_i}{C-\theta} \pi(\mathcal{B}) + \frac{\rho_i}{c} \pi(\mathcal{B}), \qquad (3.11)$$

En sommant (3.9) et (3.11), on obtient

$$E[X_i] = \frac{\rho_i}{c} + \frac{\rho_i}{C-\theta} \pi(\mathcal{B}),$$

comme annoncé. □

L'équation (3.7) a une interprétation simple et intuitive. En régime de faible trafic, la probabilité de congestion du lien tend vers zéro. Par conséquent, le nombre moyen de flots de classe i tend vers ρ_i/c, c'est-à-dire ce qui aurait été obtenu si le lien avait une capacité infinie. En régime de fort trafic, la probabilité de congestion du lien tend vers 1 et le nombre moyen de flots de classe i est alors approximativement $\rho_i/(C-\theta)$.

Cette équation fait intervenir la probabilité de congestion $\pi(\mathcal{B})$ du lien, dont la valeur peut bien sûr être calculée à partir du Lemme 1 et en utilisant (3.3) et le Corollaire 1. On peut toutefois remarquer que sous des hypothèses markoviennes, la dynamique du

nombre total de flots correspond à celle du processus de naissances et de morts associé à la file d'attente $M/M/N/\infty$. Dans ce cas, il est clair que la probabilité de congestion $\pi(\mathcal{B})$ du lien est donné par la formule d'Erlang C, c'est-à-dire

$$\pi(\mathcal{B}) = B(N, \theta/c). \tag{3.12}$$

En fait, en utilisant la théorie des files d'attente GPS (*Generalized Processor Sharing* au sens de Cohen) [40], les auteurs de [34] ont démontré que la probabilité de congestion du lien est toujours donnée par la formule d'Erlang C, même lorsque le volume des flots n'est pas gouverné par une distribution exponentielle. Il s'ensuit que la Proposition 1 permet de calculer le nombre moyen de flots de chaque classe, et donc leur débit moyen, sans nécessiter le calcul des probabilités individuelles des états. Il suffit d'appliquer la formule d'Erlang C.

Débits crêtes hétérogènes

On considère maintenant le cas où les classes de trafic ont des débits crêtes différents. Notre démarche est la même que dans le cas d'un débit crête commun, mais au lieu de nous intéresser à la probabilité de l'ensemble $\mathcal{B} = \left\{\boldsymbol{x} \in \mathbb{N}^M \ : \ \mathbf{c}.\boldsymbol{x} \geq C\right\}$, nous allons nous intéresser, pour une classe i donnée, à celle de $\mathcal{B}_i = \left\{\boldsymbol{x} \in \mathbb{N}^M \ : \ \mathbf{c}.\boldsymbol{x} \geq C - c_i\right\}$. Un flot de classe i arrivant alors que le lien est dans un état $\boldsymbol{x} \in \mathcal{B}_i$ va l'amener dans un état $\boldsymbol{y} \in \mathcal{B}$ où il y a congestion du lien, c'est-à-dire tel que $\sum_k \phi_k(\boldsymbol{y}) = C$. En conséquence, dans la suite, $\pi(\mathcal{B}_i)$ sera appelée la probabilité de congestion de la classe i. Nous commençons par établir un résultat analogue au Lemme 1 et qui va nous permettre de calculer la probabilité de congestion $\pi(\mathcal{B}_i)$ de chaque classe i.

Lemme 2. *Etant donnée une classe $i \in \mathcal{E}$, on a*

$$\pi(\mathcal{B}_i) = \frac{1}{C-\theta} \sum_{k \in \mathcal{E}} \rho_k \pi(\mathcal{W}_k) + \pi(\mathcal{W}_i) \tag{3.13}$$

où $\mathcal{W}_k = \left\{\boldsymbol{x} \in \mathbb{N}^M \ : \ C - c_k \leq \mathbf{c}.\boldsymbol{x} < C\right\}$ pour tout $k \in \mathcal{E}$.

Démonstration. Il suffit de remarquer que $\mathcal{B}_i = \mathcal{B} \cup \mathcal{W}_i$. Vu que $\mathcal{B} \cap \mathcal{W}_i = \emptyset$, on a donc $\pi(\mathcal{B}_i) = \pi(\mathcal{B}) + \pi(\mathcal{W}_i)$. Le résultat découle alors directement de (3.4). □

Le Lemme 2 nous permet de calculer la probabilité de l'ensemble $\mathcal{B}_i$ à partir d'un nombre de termes restreint pour lesquels on a $\pi(\boldsymbol{x}) = \pi(\mathbf{0}) \frac{\boldsymbol{\rho}^{\boldsymbol{x}}}{\boldsymbol{x}! \mathbf{c}^{\boldsymbol{x}}}$, l'expression de $\pi(\mathbf{0})$ étant donnée par le Corollaire 1. Notre objectif est d'obtenir une expression du nombre moyen $E[X_i]$ de flots en cours d'une classe $i \in \mathcal{E}$ en fonction de cette probabilité de congestion. Pour cela, nous commençons par démontrer le résultat intermédiaire ci-dessous.

Lemme 3. *Pour une classe $i \in \mathcal{E}$ donnée, on a*

$$\frac{1}{C-\theta}\sum_{k\in\mathcal{E}}\rho_k \sum_{\boldsymbol{x}\in\mathcal{W}_k} x_i\pi(\boldsymbol{x}) \leq \frac{\rho_i}{c_i}\pi(\mathcal{B}_i). \tag{3.14}$$

Démonstration. On sait que pour tout $\boldsymbol{x} \in \mathbb{N}^M$, on a $\sum_{k\in\mathcal{E}} \phi_k(\boldsymbol{x}) = \sum_{k\in\mathcal{E}} \frac{\Phi(\boldsymbol{x}-e_k)}{\Phi(\boldsymbol{x})} \leq C$. Avec (3.1), on en déduit que

$$C\pi(\boldsymbol{x}) \geq \sum_{k\in\mathcal{E}} \rho_k \pi(\boldsymbol{x}-e_k), \tag{3.15}$$

d'où l'on obtient

$$\begin{aligned}
C\pi(\mathcal{B}_i) &\geq \sum_{k\in\mathcal{E}}\rho_k \sum_{\boldsymbol{x}\in\mathcal{B}_i} \pi(\boldsymbol{x}-e_k), \\
&\geq \sum_{\mathbf{c}.\boldsymbol{x}\geq C-c_i} \sum_{k\in\mathcal{E}} \rho_k\pi(\boldsymbol{x}-e_k) \\
&\geq \sum_{k\in\mathcal{E}}\rho_k \sum_{\mathbf{c}.\boldsymbol{x}\geq C-c_i-c_k} \pi(\boldsymbol{x}) \\
&\geq \sum_{k\in\mathcal{E}}\rho_k \sum_{C-c_i-c_k\leq \mathbf{c}.\boldsymbol{x}<C-c_i} \pi(\boldsymbol{x}) + \theta\pi(\mathcal{B}_i).
\end{aligned}$$

Par conséquent

$$\frac{\rho_i}{c_i}\pi(\mathcal{B}_i) \geq \frac{\rho_i}{c_i}\frac{1}{C-\theta}\sum_{k\in\mathcal{E}}\rho_k \sum_{C-c_i-c_k\leq \mathbf{c}.\boldsymbol{x}<C-c_i} \pi(\boldsymbol{x}). \tag{3.16}$$

D'autre part, avec (3.8), on a $\pi(\boldsymbol{x}) = \frac{\rho_k}{c_k x_k}\pi(\boldsymbol{x}-e_k)$ pour tout $k \in \mathcal{E}$ et tout $\boldsymbol{x}$ tel que $\mathbf{c}.\boldsymbol{x} \leq C$ et $x_k > 0$. Cela implique que

$$\sum_{\boldsymbol{x}\in\mathcal{W}_k} x_i\pi(\boldsymbol{x}) = \frac{\rho_i}{c_i}\sum_{\boldsymbol{x}\in\mathcal{W}_k}\pi(\boldsymbol{x}-e_i) = \frac{\rho_i}{c_i}\sum_{C-c_k-c_i\leq \mathbf{c}.\boldsymbol{x}<C-c_i}\pi(\boldsymbol{x}), \tag{3.17}$$

d'où l'on déduit le résultat grâce à (3.16). □

On utilise le Lemme 3 pour prouver le résultat suivant.

Proposition 2. *Soit $i \in \mathcal{E}$, on a*

$$E\left[X_i\right] \leq \frac{\rho_i}{c_i} \;+\; b_i\,\frac{\rho_i}{C-\theta} \tag{3.18}$$

où $b_i = \pi(\mathcal{B}_i)$ est la probabilité de congestion de la classe i.

Démonstration. On a

$$E[x_i] = \sum_{\boldsymbol{x}} x_i \pi(\boldsymbol{x}) = \sum_{\boldsymbol{x} \notin \mathcal{B}} x_i \pi(\boldsymbol{x}) + \sum_{\boldsymbol{x} \in \mathcal{B}} x_i \pi(\boldsymbol{x})$$

On sait d'après (3.8) que $\pi(\boldsymbol{x}) = \frac{\rho_i}{\phi_i(\boldsymbol{x})} \pi(\boldsymbol{x} - e_i)$ si $x_i > 0$, donc

$$\begin{aligned} \sum_{\boldsymbol{x} \notin \mathcal{B}} x_i \pi(\boldsymbol{x}) &= \sum_{\mathbf{c}.\boldsymbol{x} < C} x_i \frac{\rho_i \pi(\boldsymbol{x} - e_i)}{x_i c_i} = \frac{\rho_i}{c_i} \sum_{\mathbf{c}.\boldsymbol{x} < C} \pi(\boldsymbol{x} - e_i) = \frac{\rho_i}{c_i} \sum_{\mathbf{c}.\boldsymbol{x} < C - c_i} \pi(\boldsymbol{x}) \\ &= \frac{\rho_i}{c_i}(1 - \pi(\mathcal{B}_i)) \end{aligned} \tag{3.19}$$

D'autre part, on sait avec (3.5) que $C\pi(\boldsymbol{x}) = \sum_{k \in \mathcal{E}} \rho_k \pi(\boldsymbol{x} - e_k)$, $\forall \boldsymbol{x} \in \mathcal{B}$. Il s'ensuit que

$$\begin{aligned} C \sum_{\boldsymbol{x} \in \mathcal{B}} x_i \pi(\boldsymbol{x}) &= \sum_{\mathbf{c}.\boldsymbol{x} \geq C} \sum_{k \in \mathcal{E}} x_i \rho_k \pi(\boldsymbol{x} - e_k), \\ &= \sum_{k \in \mathcal{E}, k \neq i} \sum_{\mathbf{c}.\boldsymbol{x} \geq C - c_k} x_i \rho_k \pi(\boldsymbol{x}) + \rho_i \sum_{\mathbf{c}.\boldsymbol{x} \geq C - c_i} (x_i + 1) \pi(\boldsymbol{x}), \\ &= \rho_i \sum_{\boldsymbol{x} \in \mathcal{B}_i} \pi(\boldsymbol{x}) + \sum_{k \in \mathcal{E}} \rho_k \sum_{\mathbf{c}.\boldsymbol{x} \geq C - c_k} x_i \pi(\boldsymbol{x}), \\ &= \rho_i \sum_{\boldsymbol{x} \in \mathcal{B}_i} \pi(\boldsymbol{x}) + \sum_{k \in \mathcal{E}} \rho_k \left(\sum_{C - c_k \leq \mathbf{c}.\boldsymbol{x} < C} x_i \pi(\boldsymbol{x}) + \sum_{\boldsymbol{x} \in \mathcal{B}} x_i \pi(\boldsymbol{x}) \right). \end{aligned}$$

On en déduit que

$$\sum_{\boldsymbol{x} \in \mathcal{B}} x_i \pi(\boldsymbol{x}) = \frac{\rho_i}{C - \theta} \pi(\mathcal{B}_i) + \frac{1}{C - \theta} \sum_{k \in \mathcal{E}} \rho_k \sum_{C - c_k \leq \mathbf{c}.\boldsymbol{x} < C} x_i \pi(\boldsymbol{x}) \tag{3.20}$$

Avec (3.19) et (3.20), on conclut que

$$\begin{aligned} E[X_i] &= \frac{\rho_i}{c_i} + \frac{\rho_i}{C - \theta} \pi(\mathcal{B}_i) + \frac{1}{C - \theta} \sum_{k \in \mathcal{E}} \rho_k \sum_{C - c_k \leq \mathbf{c}.\boldsymbol{x} < C} x_i \pi(\boldsymbol{x}) - \frac{\rho_i}{c_i} \pi(\mathcal{B}_i), \\ &\leq \frac{\rho_i}{c_i} + \pi(\mathcal{B}_i) \frac{\rho_i}{C - \theta}, \end{aligned}$$

où la dernière inégalité résulte du Lemme 3.

□

La Proposition 2 fournit une borne supérieure sur le nombre moyen de flots en cours

de la classe i. On note que cette borne supérieure a une forme similaire à l'expression de $E[X_i]$ établie dans la Proposition 1 quand les classes ont un débit crête commun. Bien que nous n'ayons pas réussi à démontrer ce résultat, toutes nos observations numériques concordent sur le fait que cette borne supérieure fournit une très bonne approximation de $E[X_i]$ quand les débits crêtes sont hétérogènes. C'est ce qu'illustre l'exemple ci-dessous.

Exemple 1. *On considère un lien de capacité* 20 *Mbps avec 3 classes de flots et on note* $p_i = \frac{\rho_i}{\sum_k \rho_k}$ *la proportion des flots de classe* i*. On considère 5 scénarii définis par* $p_1 \in \{0.1, \ldots, 0.5\}$ *et* $p_2 = p_3 = 2p_1$*. La figure 3.3 illustre l'écart relatif de la borne supérieure par rapport à la valeur exacte du nombre moyen des flots de classes* 1 *en cours.*

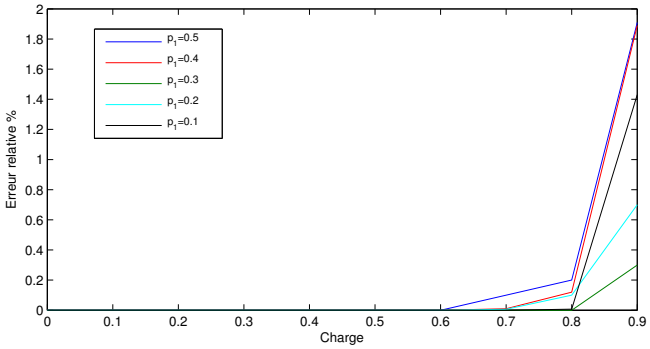


FIGURE 3.3 – Nombre de flots de classe 1 en fonction de la charge totale du lien.

On observe que l'écart relatif ne dépasse pas 2% *pour les flots de classe* 1 *(la même observation est vraie pour les deux autres classes). Cet exemple indique que la borne supérieure représente une très bonne approximation du nombre moyen de flots de chaque classe.*

3.2.3 Résultats pour plusieurs liens

On considère maintenant le cas où plusieurs liens sont partagés simultanément par un ensemble de flots élastiques. Comme précédemment, nous commençons par considérer le cas d'un débit crête commun, avant d'examiner le cas de débits crêtes hétérogènes.

Débit crête commun

On considère dans cette section que les flots ont des débits crêtes égaux, c'est-à-dire que $c_i = c$, $\forall\ i \in \mathcal{E}$. En s'inspirant de la forme de l'équation (3.7) ainsi que des observations numériques, on propose l'approximation suivante

Approximation 1. *Le nombre moyen des flots de classe i peut être approximé par*

$$E[X_i] \approx \frac{\rho_i}{c} + \sum_{j \in \mathcal{L}} a_{j,i}\, b_j\, \frac{\rho_i}{C_j - \theta_j}, \tag{3.21}$$

où b_j désigne la probabilité de congestion du lien j.

Comme on ne dispose pas d'une expression explicite de b_j, on considère l'impact de chaque lien séparément, en supposant l'indépendance des événements de congestion. Il s'avère que cette approximation est conservatrice au sens où elle surestime le taux de congestion (le taux réel est plus faible). Concrètement, on considère l'approximation suivante

$$b_j = B(N_j, \theta_j/c) \tag{3.22}$$

où $B()$ désigne la formule d'Erlang C.

On note tout d'abord que l'approximation (3.21) coïncide avec le résultat exact pour le cas d'un seul lien. On observe aussi que l'approximation est en accord avec les approximations proposées dans [28] pour les régimes de faible et de fort trafic. En régime de faible trafic, les probabilités de congestion tendent vers zéro et on obtient que le nombre moyen de flots de classe i tend vers ρ_i/c, c'est-à-dire exactement le résultat que l'on obtiendrait pour une capacité des liens infinie. En régime de fort trafic, les probabilités de congestion tendent vers 1 et l'approximation (3.21) coïncide avec la borne supérieure proposée dans [28].

On remarque également que dans certains cas l'approximation est imprécise. Si on considère par exemple le cas d'une seule classe passant par k liens (sans être génée par d'autres classes), en régime de fort trafic l'approximation sera égale à k fois la valeur exacte obtenue avec (3.7). Cet exemple montre bien qu'il y a une hypothèse d'indépendance sous-jacente à l'approximation. On peut espérer que cette hypothèse soit satisfaite dans les réseaux réels, un peu à l'image de l'indépendance des probabilités de blocage en téléphonie.

Nous n'avons malheureusement pas réussi à obtenir des bornes sur l'erreur de l'approximation 1. Pour valider cette approximation, nous fournissons ci-dessous les résultats obtenus en simulation sur des réseaux de petite dimension pour lesquels on peut calculer la solution "exacte" (en tronquant l'espace d'états). On considère des exemples de topologies simples, du type réseau linéaire, réseau étoile, réseau parking-lot et réseau en arbre. Pour chaque exemple, on considère plusieurs scénarii et, pour chacun, on fait varier le trafic offert $\sum_k \rho_k$ en gardant fixe le ratio $p_i = \rho_i / \sum_k \rho_k$ de chaque classe i.

Exemple 2. *On considère l'exemple du réseau linéaire de la figure 4.2. Les capacités des liens sont $C_1 = 25$, $C_2 = 30$, $C_3 = 35$ Mbps et on suppose un débit crête commun aux quatre classes égal à $c = 5$ Mbps. On considère 4 scénarii définis par*

$p_0 \in \{0.01, 0.1, 0.2, 0.3\}$ *et* $p_2 = p_3 = \frac{1}{2}p_1$.

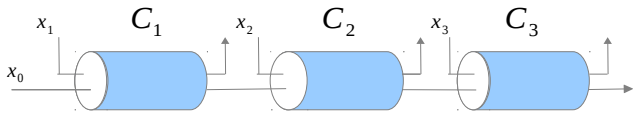


FIGURE 3.4 – example de réseau linéaire à 3 liens

Les figures 3.5 et 3.6 comparent le nombre moyen de flots de chaque classe pour $p_0 = 0.3$ *et* $p_0 = 0.2$ *avec l'approximation* (1) *et les bornes insensibles proposées dans [26]. La figure indique également la valeur "exacte" obtenue en tronquant l'espace d'états et en calculant avec* (3.1) *la probabilité de chacun des états possibles. On observe que les approximations proposées sont plus précises que les bornes de performance proposées dans [26].*

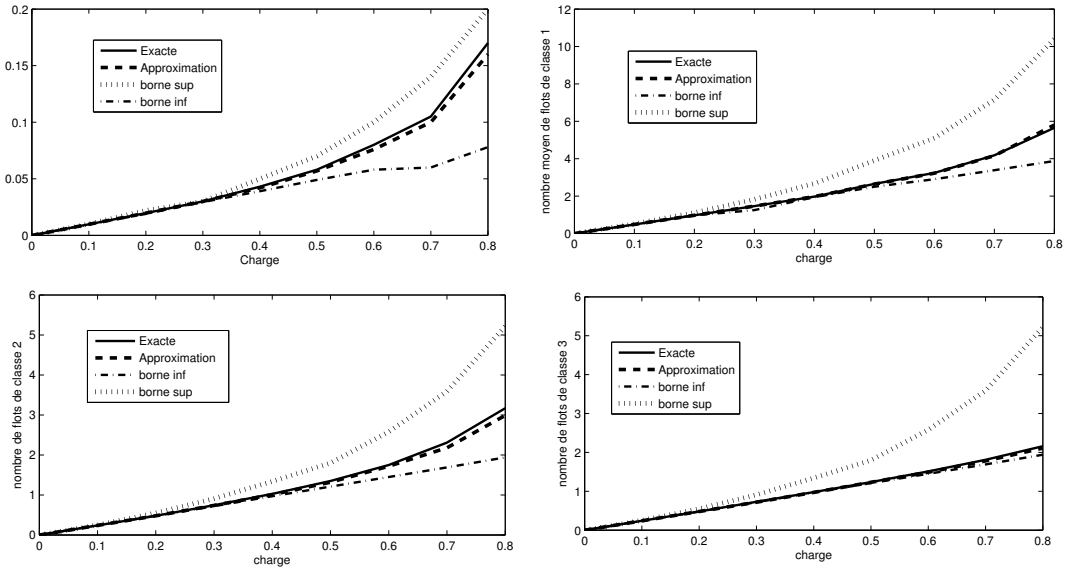


FIGURE 3.5 – Nombre de flots en cours pour $p_0 = 10^{-2}$.

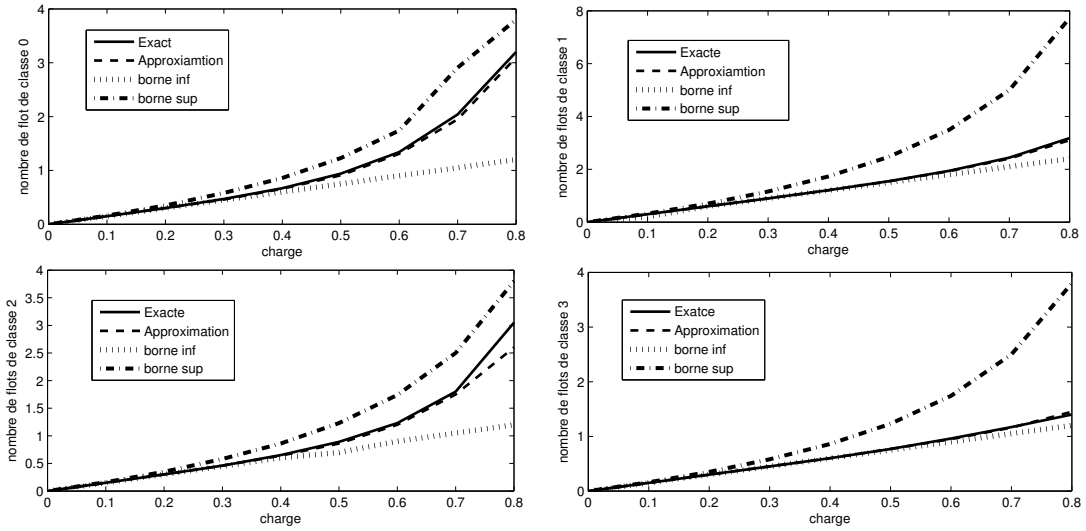


FIGURE 3.6 – Nombre de flots en cours pour $p_0 = 0.2$.

La figure 3.7 montre l'erreur relative de l'approximation en fonction de la charge du réseau. Comme on pouvait s'y attendre, l'approximation coïncide avec la valeur exacte lorsque le réseau est en régime de faible trafic. On observe également que l'erreur relative ne dépasse pas 6% pour des taux d'utilisation allant jusqu'à 80%.

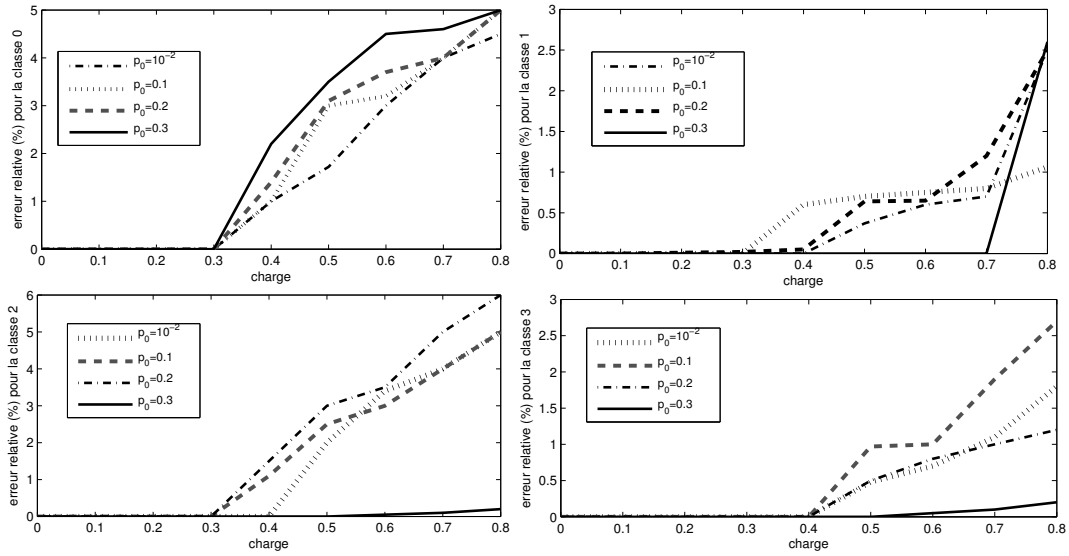


FIGURE 3.7 – Erreur relative dans l'exemple du réseau linéaire.

Exemple 3. *Considérons un deuxième exemple de réseau, du type topologie en étoile (cf. la figure 3.8). On suppose que* $c = 5$ *Mbps et on considère* 4 *scénarii définis par :* $p_1 \in \{0.01, 0.1, 0.2, 0.3\}$ *et* $p_2 = p_3$.

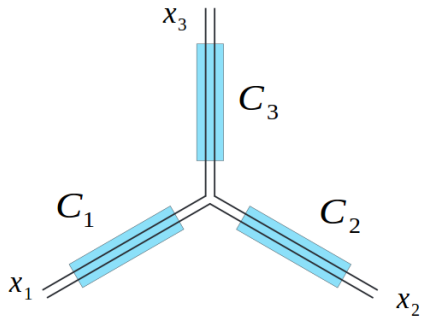


FIGURE 3.8 – Réseau étoile.

La figure 3.9 présente l'évolution de l'erreur relative en fonction de la charge des liens pour la classe 1. *On observe que nos approximations sont assez proches des valeurs exactes. L'erreur relative ne dépasse pas* 6%. *Signalons que nous avons observé une erreur relative inférieure à 5% pour les classes 2 et 3.*

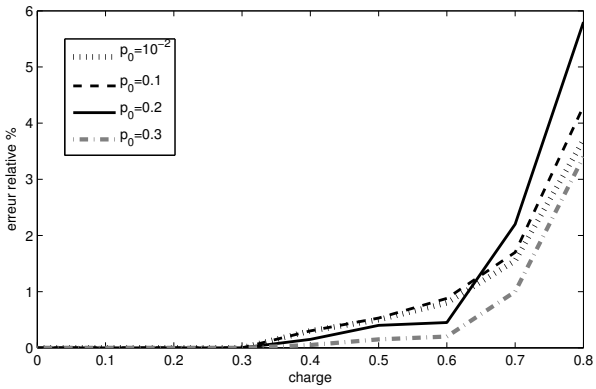


FIGURE 3.9 – Erreur relative pour la classe 1 dans un réseau étoile 3.8.

Exemple 4. *On considère maintenant le réseau de type "Parking Lot" de la figure 3.10, avec* $C_1 = 20$, $C_2 = 30$ *et* $C_3 = 25$ *Mbps. Le débit crête des flots est égal à* 5 *Mbps. On considère à nouveau* 4 *scénarii définis par* $p_1 \in \{0.01, 0.1, 0.2, 0.3\}$ *et* $p_2 = p_3$. *La figure 3.11 illustre l'erreur relative pour la classe 1 en fonction de la charge des liens.*

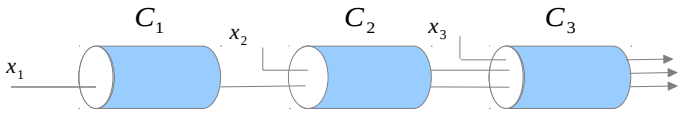


FIGURE 3.10 – Exemple de réseau *Parking Lot*

On observe que l'erreur relative ne dépasse pas 5% *pour tous les scénarii considérés. Nous avons observé que l'erreur relative ne dépasse pas 4% pour les autres classes.*

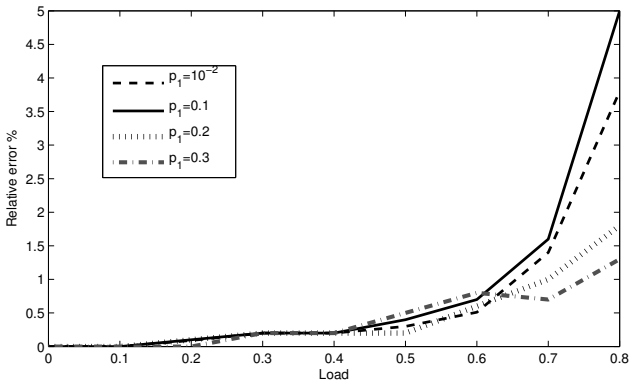


FIGURE 3.11 – Erreur relative pour les flots de classe 1 dans l'exemple de la figure 3.10

Exemple 5. *On considère aussi le réseau en arbre représenté sur la figure 3.12. Le réseau comprend 6 liens et 5 classes de trafic. La proportion des flots est donné par :* $p_1 \in \{10^{-2}, 0.1, 0.2, 0.3\}$ *et* $p_i = p_j \forall i, j \neq 1$ *La figure 3.13 présente l'évolution de l'erreur relative pour la classe 1 en fonction de la charge du réseau. On observe que dans tous les scénarios considérés l'erreur relative ne dépasse pas* 3%*. Notons avons observé que pour les autres classes l'erreur est inférieure à* 5%*.*

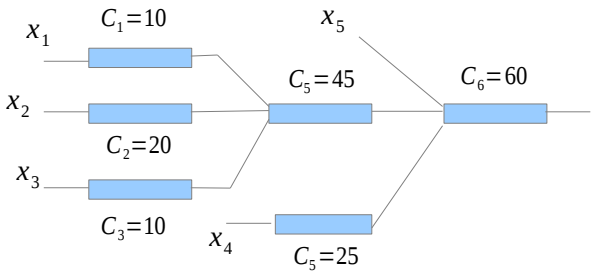


FIGURE 3.12 – Exemple de réseau arbre.

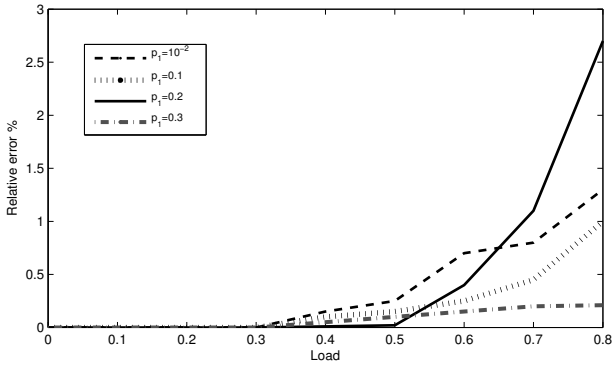


FIGURE 3.13 – Erreur relative pour la classe 1 dans le réseau arbre (cf. 3.12)

Exemple 6. *Nous considérons à présent un exemple plus complexe qui correspond au réseau en arbre de la figure 3.14. Le réseau est constitué de* 10 *liens et* 10 *classes de flots. La proportion des classes est :* $p_1 \in \{0.01, 0.1, 0.2, 0.3\}$ *et* $p_i = p_j \ \forall i, j \neq 1$.

La figure 3.15 montre l'évolution de l'erreur relative pour la classe 1 *en fonction de la charge du réseau. Pour tous le scénarii, l'erreur est inférieure à* 6.5 *%. De même l'erreur ne dépassent pas* 7% *pour les autres classes de flot.*

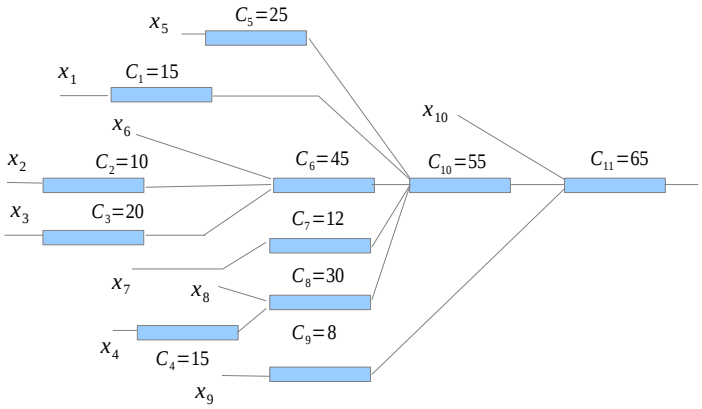


FIGURE 3.14 – Réseau arbre de 10 liens.

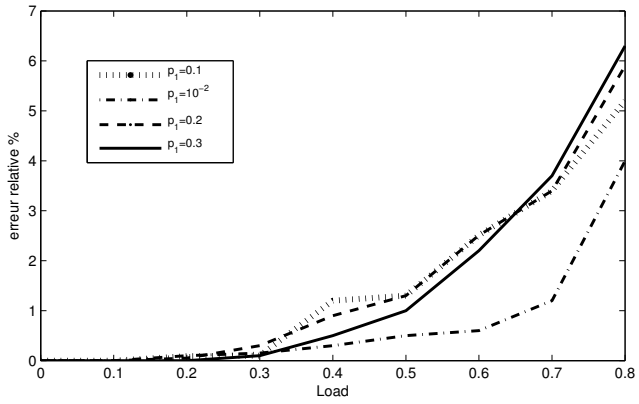


FIGURE 3.15 – Erreur relative pour la classe 1 dans l'exemple 3.14.

Débits crêtes hétérogènes

On peut simplement étendre l'approximation proposée pour le cas d'un débit crête commun au cas de débits crêtes hétérogènes en remplaçant b_j par b_j^i dans (3.21), b_j^i représentant la probabilité que les flots de classe i n'obtienne pas leur débit crête au lien j.

On obtient ainsi l'approximation suivante.

Approximation 2. *Le nombre moyen de flots de classe i est donné par*

$$E\left[X_i\right] \approx \frac{\rho_i}{c_i} \; + \; \sum_{j \in \mathcal{L}} a_{j,i}\, b_j^i \, \frac{\rho_i}{C_j - \theta_j}, \tag{3.23}$$

tel que $b_j^i = \frac{1}{C_j - \theta_j} \sum_{k \in \mathcal{E}} \rho_k \pi(\mathcal{W}_k) + \pi(\mathcal{W}_i)$ et
$\mathcal{W}_k = \left\{ \boldsymbol{x} \in \mathbb{N}^M \; : \; C - c_k \leq \mathbf{c}.\boldsymbol{x} < C \right\}$ pour tout $k \in \mathcal{E}$ (cf. Lemme 3).

Comme précédemment, nous utilisons des résultats de simulation pour valider cette approximation.

Exemple 7. *Nous considérons à nouveau l'exemple 2 basé sur le réseau linéaire de la figure 4.2. Les capacités des liens sont toujours* $C_1 = 25$, $C_2 = 30$, $C_3 = 35$ *Mbps, mais nous supposons maintenant que chaque classe de trafic a son propre débit crête :* $c_0 = 2$, $c_1 = 3$, $c_2 = 4$ *et* $c_3 = 5$ *Mbps. La figure 3.16 montre l'erreur relative obtenue pour la classe* 0 *en fonction de la charge du réseau. On observe que l'erreur relative est inférieure à* 3*%. Signalons que l'erreur relative obtenue pour les autres classes ne dépasse pas* 6 *%.*

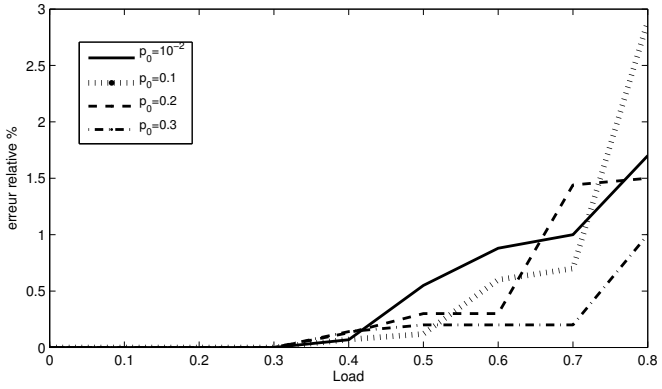


FIGURE 3.16 – Erreur relative pour la classe 0 dans un système multi-débit.

3.3 Modélisation de l'intégration des flots élastiques et de streaming

On étudie maintenant l'impact de la présence de flots de streaming sur les performances des flots élastiques, en supposant que le nombre de flots de streaming est régulé par un mécanisme de contrôle d'admission. Nous allons chercher à évaluer deux métriques de performance principales. Tout d'abord, pour les flots de streaming, nous allons nous intéresser à la probabilité qu'un flot de streaming soit bloqué par le contrôle d'admission (taux de blocage). Pour les flots élastiques, nous nous intéressons comme

précédemment au nombre moyen de flots élastiques en cours, car on peut en déduire la durée moyenne d'un transfert de fichier et le débit moyen des flots [17] .

On étudie tout d'abord le cas d'un seul lien partagé par un ensemble de flots de streaming et de flots élastiques. On généralise ensuite les résultats au cas où plusieurs liens sont partagés simultanément.

3.3.1 Hypothèses et notations

Nous considérons toujours un ensemble $\mathcal{L} = \{1, ..., K\}$ de liens , où chaque lien $l \in \mathcal{L}$ a une capacité finie C_l bit/s ; on note $\mathbf{C}$ le vecteur associé. Nous supposons qu'il y a maintenant un ensemble $\mathcal{S}$ de classes de flots de streaming, en plus de l'ensemble $\mathcal{E}$ des classes de flots élastiques. Chaque classe $i \in \mathcal{E} \cup \mathcal{S}$ est caractérisée par une route $r_i \subset \mathcal{L}$ et nous notons A la matrice d'incidence telle que $a_{i,j} = 1$ si $j \in r_i$, et 0 sinon.

On suppose que les flots élastiques de classe $i \in \mathcal{E}$ arrivent selon un processus de Poisson d'intensité λ_i et que leur volume aléatoire a pour moyenne $1/\mu_i$. De même, les flots de streaming de classe i arrivent selon un processus de poisson de paramètre λ_i et envoient des données à un débit fixe d_i pendant une durée aléatoire de distribution exponentielle dont la moyenne est $1/\mu_i$. On note $\rho_i = \lambda_i/\mu_i$ l'intensité du trafic de la classe i et $\boldsymbol{\rho}$ le vecteur associé. Pour tout $i \in \mathcal{E}$, on note c_i le débit crête des flots élastiques de la classe i.

Soit x_i le nombre de flots de classe i en cours. On note $\boldsymbol{x} = (\boldsymbol{x}^e, \boldsymbol{x}^s)$ l'état du système, où $\boldsymbol{x}^s = (x_i)_{i\in\mathcal{S}}$ et $\boldsymbol{x}^e = (x_i)_{i\in\mathcal{E}}$. On note également $\phi_i(\boldsymbol{x})$ le débit total de la classe $i \in \mathcal{E} \cup \mathcal{S}$ dans l'état $\boldsymbol{x}$ et $\boldsymbol{\phi}(\boldsymbol{x})$ le vecteur associé. Conformément aux observations faites en simulation (cf. Chapitre 2), on suppose que les flots de streaming sont servis en priorité, tandis que les flots élastiques s'adaptent pour partager la capacité résiduelle suivant l'équité équilibrée.

Pour garantir un débit minimum aux flots élastiques, on utilise un mécanisme de contrôle d'admission. On note C_s^l le débit maximum des flots de streaming sur le lien l, la bande passante résiduelle $C_l - C_s^l$ étant le minimum garanti aux flots élastiques sur le lien l. Un flot de streaming de classe $i \in \mathcal{S}$ arrivant est alors admis dans l'état $\boldsymbol{x}$ si et seulement si

$$\sum_{j\in\mathcal{S}} a_{j,l} x_j^s d_j \leq C_s^l - d_i, \quad \forall l \in r_i.$$

On notera $\mathbf{C}_s = \{C_s^l\}_{l\in\mathcal{L}}$ le vecteur décrivant le maximum de bande passante allouée au trafic de streaming sur chaque lien. En notant alors $\theta_l = \sum_{i\in\mathcal{E}} a_{i,l}\rho_i$ le trafic élastique total offert au lien $l \in \mathcal{L}$, on voit qu'une condition nécessaire et suffisante pour la stabilité du système est

$$\theta_l < C_l - C_s^l, \quad \forall l \in \mathcal{L}, \tag{3.24}$$

ce que nous supposerons par la suite. Notons aussi qu'en l'absence de flots élastiques le système est toujours stable.

3.3.2 Cas d'un seul lien

Ce paragraphe est consacré au partage d'un lien unique par un ensemble de flots élastiques et de streaming. On propose des résultats exacts pour les métriques de performance des flots streaming, ainsi que deux approches pour estimer les métriques de performances des flots élastiques. La première approche repose sur une *hypothèse de quasi-stationnarité* et permet d'obtenir des approximations simples et explicites des métriques de performance des flots élastiques [42, 80]. La deuxième approche repose sur l'utilisation de bornes insensibles sur les performances des flots élastiques. Nous commençons par considérer le modèle de base dans lequel les flots élastiques sont homogènes et ne sont pas limité en débit. Nous considérons ensuite le cas où les trafics élastiques ont un débit crête commun. Enfin, nous levons cette restriction pour obtenir le modèle le plus général.

3.3.2.1 Modèle de base

On considère un seul lien de capacité C partagée par deux classes de flots. On suppose que les flots élastiques appartiennent à la classe 1 tandis que les flots de streaming appartiennent à la classe 2. On note d le débit commun des flots de streaming et on suppose que les flots élastiques ne sont pas limité en débit ($c_1 \to \infty$)

Performance des flots streaming La performance des flots de streaming peut être analysée indépendamment. Le modèle est équivalent au modèle classique d'Erlang [83]. Les flots de streaming ne sont admis dans le système que si $x_2 < N_S = C_S/d$, et sont bloqués et perdus sinon. La distribution marginale du nombre de flots de streaming correspond alors tout simplement à celle d'une file d'attente $M/M/N_S/N_S$:

$$\pi_s(x_2) = \pi_s(0)\frac{\rho_2^{x_2}}{x_2!} \quad x_2 = 0, 1, \ldots, N_S. \tag{3.25}$$

On en déduit l'expression du taux de blocage des flots de streaming, qui est donné par la formule d'Erlang B :

$$B = \frac{\frac{\rho_2^{N_s}}{N_s!}}{\sum_{x_2=0}^{N_s} \frac{\rho_2^{x_2}}{x_2!}}. \tag{3.26}$$

On note que la formule d'Erlang B étant insensible, le taux de blocage ne dépend de la distribution sur la durée des flots de streaming qu'à travers sa moyenne.

Approche quasi-stationnaire pour les flots élastiques En l'absence de flots de streaming, le système se réduit à une file d'attente $M/G/1/PS$ et la probabilité stationnaire d'avoir x_1 flots élastiques est donnée par $(1-\rho_1)\rho_1^{x_1}$. Le nombre moyen de flots élastiques est alors

$$E[x_1] = \frac{\rho_1}{C-\rho_1}, \tag{3.27}$$

d'où l'on déduit le débit moyen des flots élastiques $\gamma_1 = C - \rho_1$.

Le calcul exact du débit moyen des flots élastiques s'avère compliqué lorsque le réseau contient des flots de streaming. On propose ici d'estimer ce débit sous une hypothèse simplificatrice de quasi-stationnarité (QS : Quasi Stationarity). Plus précisément, on suppose que le nombre de flots élastiques évolue rapidement par rapport au nombre de flots de streaming : le ratio λ_2/λ_1 est suffisamment petit pour que le nombre de flots élastiques atteigne un régime permanent avant que le nombre de flots de streaming n'ait évolué.

On peut utiliser l'hypothèse QS de la façon suivante. Étant donné le nombre x_2 de flots de streaming, on peut déduire le nombre moyen de flots élastiques $E[x_1|x_2]$ en remplaçant C par $C - \phi_2(\boldsymbol{x}) = C - dx_2$ dans l'équation (3.27). On obtient ainsi l'expression suivante du nombre moyen de flots élastiques

$$\begin{aligned} E[x_1] &= \sum_{x_2=0}^{N_s} E[x_1|x_2]\,\pi_s(x_2), \\ &= \frac{1}{\sum_{x_2=0}^{N_s} \frac{\rho_2^{x_2}}{x_2!}} \sum_{x_2=0}^{N_s} \frac{\rho_1}{C - dx_2 - \rho_1} \frac{\rho_2^{x_2}}{x_2!} \end{aligned} \tag{3.28}$$

et l'on en déduit immédiatement une expression du débit moyen des flots élastiques :

$$\gamma_1 = \frac{\rho_1}{E[x_1]} = \rho_1 \left(\frac{1}{\sum_{x_2=0}^{N_s} \frac{\rho_2^{x_2}}{x_2!}} \sum_{x_2=0}^{N_s} \frac{\rho_1}{C - dx_2 - \rho_1} \frac{\rho_2^{x_2}}{x_2!} \right)^{-1}. \tag{3.29}$$

Bornes insensibles pour les flots élastiques L'idée ici est de voir le système comme deux files d'attente PS couplées, la file d'attente servant le trafic de streaming ayant le taux de service $\phi_2(\boldsymbol{x}) = x_2 d$ dans l'état $\boldsymbol{x} = (x_1, x_2) \in \mathbb{N} \times \{0, 1, \ldots, N_S\}$, alors que celle servant le trafic élastique a un taux de service $\phi_1(\boldsymbol{x}) = (C - x_2 d)$ dans cet état.

L'analyse de ce système est complexe et n'a vraiment d'intérêt d'un point de vue pratique que si elle conduit à une distribution jointe de l'état insensible, c'est-à-dire une distribution qui ne dépend des lois des temps de service que par leurs moyennes. Comme

expliqué au paragraphe 2.5.1 du Chapitre 2, une condition nécessaire d'insensibilité est la propriété d'équilibre suivante

$$\frac{\phi_1(\boldsymbol{x}-e_2)}{\phi_1(\boldsymbol{x})} = \frac{\phi_2(\boldsymbol{x}-e_1)}{\phi_2(\boldsymbol{x})}, \tag{3.30}$$

pour tout état $\boldsymbol{x} = (x_1, x_2)$ tel que $x_1 > 0$ et $x_2 > 0$. Pour le système que nous considérons, on a

$$\frac{\phi_1(\boldsymbol{x}-e_2)}{\phi_1(\boldsymbol{x})} = \frac{C+d-x_2 d}{C-x_2 d} > 1 = \frac{x_2 d}{x_2 d} = \frac{\phi_2(\boldsymbol{x}-e_1)}{\phi_2(\boldsymbol{x})}, \tag{3.31}$$

ce qui implique que la condition (3.30) n'est pas vérifiée, et donc que la distribution stationnaire de l'état est sensible aux caractéristiques fines du trafic.

Pour obtenir des bornes insensibles sur les performances du trafic élastique, nous appliquons la technique proposée dans [30] et décrite au paragraphe 2.5.1. On vérifie aisément que la propriété de monotonicité est vérifiée, c'est-à-dire que si on enlève un flot d'une classe cela ne diminue pas le débit de l'autre classe. L'équation (3.31) implique d'autre part que la propriété de biais est vérifiée. On en déduit les résultats suivants :

— *Borne inférieure sur le débit moyen* : si $\rho_1 \leq C - C_s$, on a

$$\gamma_1 > \gamma_1^- = \rho_1 / E^-[\boldsymbol{x}_1], \tag{3.32}$$

où $E^-[\boldsymbol{x}_1] = \rho_1/(C-\rho_1)$.

— *Borne supérieure sur le débit moyen* : on a

$$\gamma_1 < \gamma_1^+ = \rho_1 / E^+[\boldsymbol{x}_1], \tag{3.33}$$

où $E^+[\boldsymbol{x}_1] = \sum_{x_2=0}^{N_s} \frac{\rho_1}{C-x_2 d-\rho_1}\alpha(x_2) / \left(\sum_{x_2=0}^{N_s} \alpha(x_2)\right)$ et $\alpha(x) = \left(1 - \frac{\rho_1}{C-xd}\right)^{-1} \frac{\rho_2^x}{d^x x!}$.

Nous fournissons ci-dessous un exemple qui permet de comparer les résultats obtenus avec les deux approches.

Exemple 8. *On considère maintenant l'exemple d'un lien de capacité $C = 30Mbps$ et deux classes de flots. La première classe correspond aux flots élastiques tandis que la deuxième correspond aux flots de streaming. On suppose que $c = 4Mbps$ et $d = 2Mbps$. On considère que les flots élastiques présentent une proportion de 90 % du trafic total et que les flots streaming présentent une proportion de 10 % du trafic. Soit $\alpha = \frac{\lambda_2\mu_2}{\lambda_1\mu_1}$.*

La figure 3.17 montre l'évolution du nombre moyen des flots élastiques en fonction de la charge du lien pour différentes valeurs de α.

On observe que les résultats obtenus avec l'approche quasi-stationnaire sont très

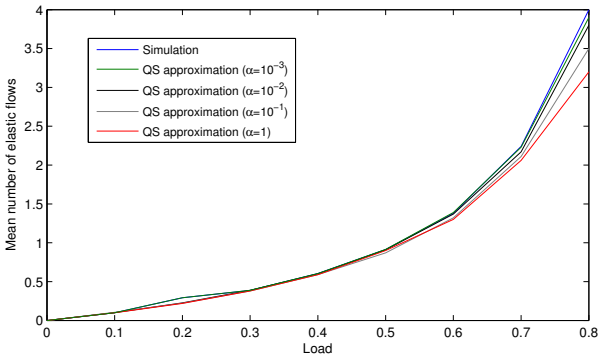


FIGURE 3.17 – Nombre moyen des flots élastique en cours en fonction α

proches de ceux obtenus par les simulations événementielles pour les différentes valeurs de α. Notons aussi que l'erreur relative diminue lorsque l'hypothèse quasi-stationnaire est vérifiée (α est assez petit).

La figure 3.18 compare les résultats obtenus avec les deux approches pour une valeur de α assez petite ($\alpha = 10^{-3}$) avec ceux obtenus par les simulations à évènement discret. On note que les approximations obtenues avec l'approche quasi-stationnaire sont très proches des simulations événementielles, l'erreur ne dépasse pas 3%. On remarque également que la précision des bornes insensibles diminue avec l'augmentation de la charge du lien.

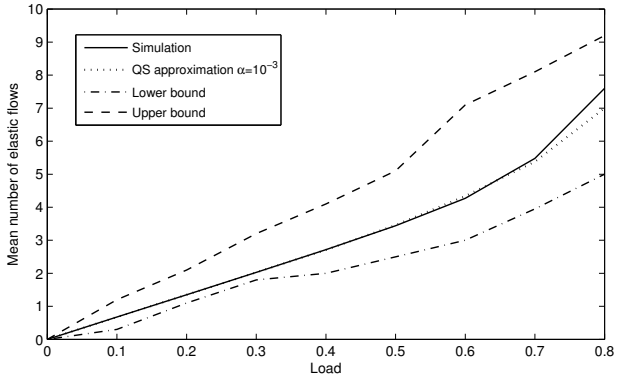


FIGURE 3.18 – Nombre moyen des flots élastique en cours en fonction de la charge du lien.

3.3.2.2 Modèle multiclasse

Nous considérons maintenant la situation dans laquelle les flots de streaming peuvent avoir des débits d_i différents. Nous supposons également que les flots élastiques sont limités en débit, mais avec un débit crête commun c (le cas de différents débits crêtes est traité au paragraphe suivant).

Performance du trafic de streaming Comme dans le cas monoclasse, on estime les performances des flots de streaming à travers leur taux de blocage. Pour simplifier, on suppose que C et les débits d_i, $i \in \mathcal{S}$, sont des multiples entiers d'une unité de débit arbitraire. On peut alors calculer le taux de blocage de chaque classe de trafic de streaming grâce à l'algorithme de Kaufman-Roberts [83]. Formellement, le résultat est le suivant.

Lemme 4. *Le taux de blocage de la classe $i \in \mathcal{S}$ est donné par*

$$B_i = \sum_{n > C - d_i} p_s(n), \tag{3.34}$$

où

$$p_s(n) = p_s(0) \sum_{i \in \mathcal{S}} \frac{\rho_i}{n} p_s(n - d_i), \tag{3.35}$$

avec $p_s(0) = \left(\sum_{k=0}^{C_s} \sum_{i \in \mathcal{S}} \frac{\rho_i}{k} p_s(k - d_i) \right)^{-1}$ *et* $p_s(n) = 0$ *pour* $n < 0$.

Approche quasi-stationnaire pour le trafic élastique Nous généralisons l'approche quasi-stationnaire décrite précédemment. L'idée est de supposer que $\boldsymbol{x}^s$ est fixé. En notant $n = \sum_{i \in \mathcal{S}} \phi_i(\boldsymbol{x}^s)$ la bande passante consommée par le trafic de streaming, on voit que les flots élastiques partagent la capacité résiduelle $C - n$ selon Balanced Fairness.

En posant $N = (C - n)/c$ et $\theta = \sum_{k \in \mathcal{E}} \rho_k < C - n$, on peut utiliser les résultats établis au paragraphe 3.2.2. On obtient que le nombre moyen de flots élastiques de classe i en cours est donné par

$$E\left[x_i^e | n\right] = \frac{\rho_i}{c} \; + \; B(n) \, \frac{\rho_i}{C - n - \theta}, \tag{3.36}$$

où $B(n)$ est la probabilité de congestion pour une capacité de $C - n$ Mbps et est donnée par la formule d'Erlang C

$$B(n) = \frac{\frac{1}{N!} \left(\frac{\theta}{c}\right)^N \frac{C_n}{C - n - \theta}}{\sum_{i=0}^{N-1} \frac{1}{i!} \left(\frac{\theta}{c}\right)^i + \frac{1}{N!} \left(\frac{\theta}{c}\right)^N \frac{C - n}{C - n - \theta}}. \tag{3.37}$$

On en déduit l'expression du débit moyen de la classe i :

$$\gamma_i = \rho_i / \sum_{n \leq C_s} E[x_i^e | n]\, p_S(n). \tag{3.38}$$

Bornes insensibles pour le trafic élastique Comme dans le cas monoclasse, le système correspond à un réseau de files d'attente PS de $|\mathcal{E}| + |\mathcal{S}|$ nœuds où le nœud i correspond à la classe i. Le taux de service du nœud i dans l'état $\boldsymbol{x}$ est donné par,

$$\phi_i(\boldsymbol{x}) = \begin{cases} x_i d_i & \text{si } i \in \mathcal{S}, \\ x_i \min\left(c, \frac{C - \sum_{k \in \mathcal{S}} \phi_k(\boldsymbol{x})}{|\boldsymbol{x}^e|}\right) & \text{sinon.} \end{cases}$$

On vérifie aisément que la propriété d'équilibre n'est pas satisfaite, et donc que la distribution stationnaire de l'état du système est sensible aux caractéristiques fines du trafic. En suivant la même approche que précédemment pour le cas monoclasse, on peut obtenir des bornes insensibles. Là encore, les propriétés de monotonicité et de biais sont vérifiées. On obtient les expressions suivantes des fonctions de balance

$$\Phi^-(\boldsymbol{x}) = \left(\prod_{i \in \mathcal{E}} \prod_{j=1}^{x_i} \phi_i(\sum_{k=1}^{i-1} x_k \mathbf{e}_k + j\mathbf{e}_i) \times \prod_{i \in \mathcal{S}} \prod_{j=1}^{x_i} \phi_i(\sum_{m \in \mathcal{E}} \mathbf{e}_m x_m + \sum_{k=1}^{i-1} x_k \mathbf{e}_k + j\mathbf{e}_i) \right)^{-1}. \tag{3.39}$$

et,

$$\Phi^+(\boldsymbol{x}) = \left(\prod_{i \in \mathcal{S}} \prod_{j=1}^{x_i} \phi_i(\sum_{k=1}^{i-1} x_k \mathbf{e}_k + j\mathbf{e}_i) \times \times \prod_{i \in \mathcal{E}} \prod_{j=1}^{x_i} \phi_i(\sum_{m \in \mathcal{S}} \mathbf{e}_m x_m + \sum_{k=1}^{i-1} x_k \mathbf{e}_k + j\mathbf{e}_i) \right)^{-1}. \tag{3.40}$$

Les bornes inférieure et supérieure sur le débit moyen de la classe i sont alors données, respectivement, par

$$\gamma_i^- = \rho_i / \left(\pi^-(0) \sum_{\boldsymbol{x}} x_i \Phi^-(\boldsymbol{x}) \rho^{\boldsymbol{x}} \right), \tag{3.41}$$

et,

$$\gamma_i^+ = \rho_i / \left(\pi^+(0) \sum_{\boldsymbol{x}} x_i \Phi^+(\boldsymbol{x}) \rho^{\boldsymbol{x}} \right), \tag{3.42}$$

où $\pi^-(0) = \left(\sum_{\boldsymbol{x}} \Phi^-(\boldsymbol{x}) \rho^{\boldsymbol{x}}\right)^{-1}$ et $\pi^+(0) = \left(\sum_{\boldsymbol{x}} \Phi^+(\boldsymbol{x}) \rho^{\boldsymbol{x}}\right)^{-1}$.

Comme dans le cas monoclasse, nous comparons les deux approches en utilisant l'exemple suivant.

Exemple 9. *La figure 3.19 illustre le résultat obtenu pour $C = 30Mbps$ en considérant 4 classes de flots ; les deux premières correspondent au trafic élastique tandis que les deux autres correspondent aux flots streaming. On considère les paramètres suivants : $c_1 = c_2 = 4Mbps$, $d_3 = 3$ et $d_4 = 4$ Mbps et une répartition de charge égale à 90% pour les flots élastiques et 10% pour les flots de streaming.*

On observe que les résultats obtenus par l'approche quasi-stationnaire sont très proches de ceux obtenus par les simulations à évènements discrets ; l'erreur relative est inférieure à 5%. De plus, on remarque que la précision des bornes insensibles diminue avec l'augmentation de la charge du lien.

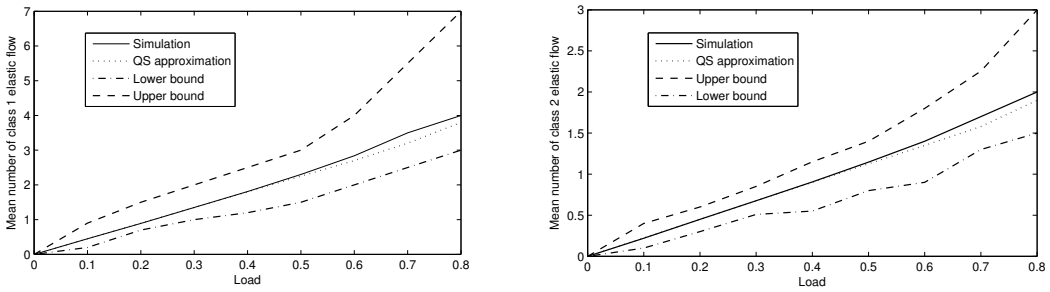


FIGURE 3.19 – Nombre moyen des flots élastique de classe 1 et 2 en cours en fonction de la charge du lien

3.3.2.3 Modèle avec débits crêtes hétérogènes

Nous considérons maintenant que non seulement les trafics de streaming peuvent avoir des débits différents, mais les différentes classes de trafic élastique peuvent avoir des débits crêtes hétérogènes. On note c_i le débit crête de la classe i et $\mathbf{c}$ le vecteur associé. Nous ne présentons ci-dessous que le modèle basé sur une hypothèse de quasi-stationnarité, sans chercher à établir des bornes insensibles sur la performance du trafic élastique.

En l'absence de trafic de streaming, on retrouve le modèle multi-débit étudié dans la section 3.2.2. En sa présence, on peut étendre l'approche quasi-stationnaire au cas multi-débit en procédant comme dans le cas d'un débit crête commun. Étant donné la bande passante occupée par le trafic de streaming $n = \sum_{i \in \mathcal{S}} \phi_s(\boldsymbol{x}_s)$, on calcule $E[\boldsymbol{x}_i^e | n]$ en utilisant le modèle multi-débit du paragraphe 3.2.2 dans lequel on remplace C par $C - n$. On obtient alors l'approximation suivante :

$$\gamma_i = \rho_i / \sum_{n \leq C_s} E[x_i^e | n]\, p_S(n). \tag{3.43}$$

Nous illustrons la qualité de cette approximation avec l'exemple suivant.

Exemple 10. *On considère à nouveau l'exemple 9, mais on suppose maintenant que* $c_1 = 2Mbps$ *et* $c_2 = 3Mbps$*. La figure 3.20 illustre l'évolution du nombre moyen des flots élastiques de classe* 1 *en fonction de la charge du lien. L'erreur relative est inférieure à 4 % pour les flots de classe* 1 *(elle ne dépasse pas 5 % pour les flots de classe* 2*).*

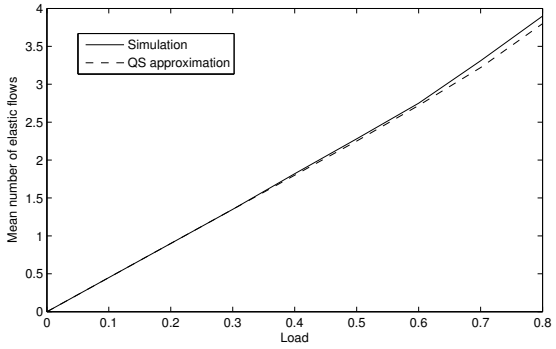


FIGURE 3.20 – Nombre moyen des flots élastiques de classe 1 en cours pour le cas multidébit.

3.3.3 Extension à un réseau

On considère maintenant le cas où plusieurs liens sont partagés par un ensemble de flots de streaming et de flots élastiques. On propose des résultats de performance exacts pour les flots de streaming et des approximations pour les flots élastiques. Les approximations sont basées sur la même approche quasi-stationnaire que celle considérée précédemment dans le cas d'un lien unique.

3.3.3.1 Performance des flots streaming

Comme pour le cas d'un seul lien, le taux de blocage des flux de streaming peut être analysé indépendamment. Le modèle approprié est celui des réseaux à commutation de circuits multidébits (*multirate loss networks* en anglais) [83]. La distribution stationnaire du nombre de flots de streaming de chaque classe a une forme produit

$$\pi_S(\boldsymbol{x}^s) = \pi_S(\mathbf{0}) \prod_{i \in \mathcal{S}} \frac{\rho_i^{x_i^s}}{x_i^s!}, \tag{3.44}$$

où $\boldsymbol{x}^s$ est un vecteur de $\mathbb{N}^{|\mathcal{S}|}$ tel que les contraintes de capacité sont vérifiées, c'est-à-dire $\sum_{j\in\mathcal{S}} a_{j,l}x_j^s d_j \leq C_s^l$ pour tout lien $l \in \mathcal{L}$. La constante $\pi_S(\mathbf{0})$ s'obtient par normalisation des probabilités.

Etant donnée une classe i, soit $\mathcal{X}_i^s$ l'ensemble des états tels qu'il existe au moins un lien $l \in r_i$ pour lequel on a

$$C_s^l - d_i \leq \sum_{j\in\mathcal{S}} a_{j,l}x_j^s d_j \leq C_s^l$$

En remarquant que les flots de classe i sont bloqués dans les $\boldsymbol{x}^s \in \mathcal{X}_i^s$, on obtient le taux de blocage des flots de classe i :

$$B_i = \sum_{\boldsymbol{x}^s\in\mathcal{X}_i^s} \pi_S(\boldsymbol{x}^s). \tag{3.45}$$

Évidemment, l'utilisation de la forme produit (3.44) pour en déduire la probabilité de blocage B_i de chaque classe n'est envisageable que pour un nombre de classes de trafic de streaming très réduit à cause de l'explosion combinatoire de l'espace d'états. Pour des réseaux de grande dimension, on pourra utiliser des algorithmes de type point fixe généralisant la méthode du point fixe de Kelly (cf. [83] pour plus de détails). Ces algorithmes supposent l'indépendance des probabilités de blocage des liens pour chaque classe de trafic, mais fournissent en général des approximations satisfaisantes pour de grands réseaux.

3.3.3.2 Performance des flots élastiques

On peut étendre l'approximation quasi-stationnaire introduite dans le cas d'un seul de la façon suivante. Etant donné le vecteur $\boldsymbol{x}_s$ décrivant le nombre de flots de streaming de chaque classe, on peut calculer le nombre de flots élastiques $E[x_i^e|\boldsymbol{x}^s]$ de la classe i en utilisant l'équation (3.23) dans laquelle on remplace la capacité C_l de chaque lien $l \in r_i$ par la capacité résiduelle $C_l - \sum_{k\in\mathcal{S}} a_{k,l}x_k^s d_k$. On obtient ainsi une approximation du nombre moyen de flots de classe i dans le réseau :

$$E[x_i^e] = \sum_{\boldsymbol{x}^s} E[x_i^e|\boldsymbol{x}_s]\pi_S(\boldsymbol{x}^s) \tag{3.46}$$

Évidemment, là encore, l'approximation (3.46) est basée sur l'utilisation de la forme produit (3.44) et n'est envisageable que pour des réseaux de petits taille. Des travaux supplémentaires doivent être menés pour développer une approximation qui passe à l'échelle. L'exemple ci-dessous montre néanmoins que, sur des petits réseaux, l'approximation (3.46) est de bonne qualité.

Exemple 11. *On reprend à nouveau l'exemple 2, et on suppose maintenant deux*

scénarios : (i) dans le premier scénario, on suppose qu'il y a deux classes de flots de streaming en plus, la classe 4 *et la classe* 5*. Les flots de classe* 4 *(resp.* 5*) traversent le lien* 1 *(resp.* 3*) avec* $d_4 = 3$ *et* $d_5 = 4$*. (ii) dans le deuxième scénario, on suppose qu'il y a trois classes de flots streaming de plus tel que la classe* 4 *traverse le lien* 0*, la classe* 5 *utilise le lien* 2 *et la classe* 6 *traverse le lien* 3 *avec* $d_4 = 2$*,* $d_5 = 3$ *et* $d_6 = 4$*. On considère pour les deux scénarii une répartition de charge égale à* 80% *pour les flots élastiques et* 20% *pour les flots de streaming.*

La figure 3.21 illustre l'évolution du nombre moyen de flots élastiques en fonction de la charge du réseau pour les deux scénarii considérés. On observe que les résultats obtenus avec l'approche quasi-stationnaire sont proches de ceux obtenus avec les simulations événementielles pour les deux scénarii considérés. L'erreur relative ne dépassent pas 6.5% *dans les deux scénarii.*

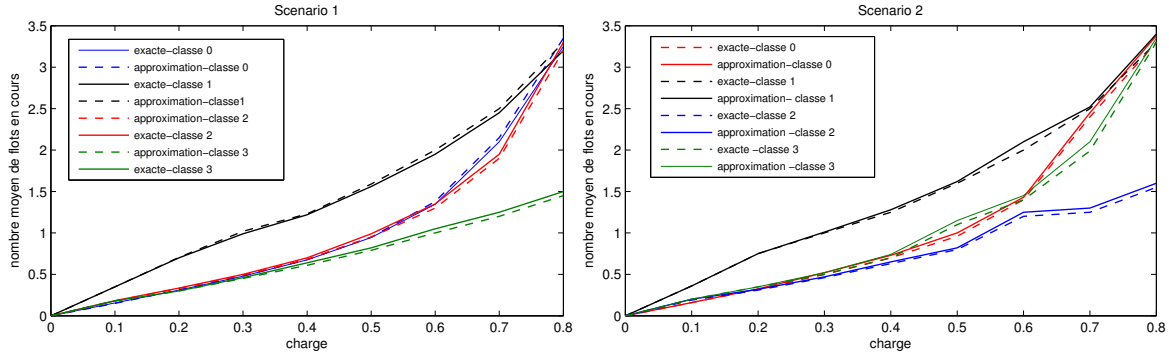


FIGURE 3.21 – Nombre moyen de flots élastiques en cours pour le scénario 1 et 2 en fonction de la charge du réseau.

3.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'évaluation de performance du trafic Internet. Nous nous sommes restreint au cas où les flots de streaming sont prioritaires et ont un débit fixe. Pour garantir un débit minimal pour les flots élastiques, nous avons considéré un mécanisme de contrôle d'admission. En l'absence de flux de streaming, des approximations simples et explicites ont été proposées.

En la présence de ces flots, des résultats de performance exacts ont été obtenus pour les flots streaming et des approximations ont été proposées pour les flots élastiques. Ces approximations sont basées soit sur une hypothèse de quasi-stationnarité soit sur des bornes insensibles. Dans le chapitre suivant, on considère toujours les réseaux à partage

de bande passante, mais on cherche cette fois à déterminer la politique optimisant un compromis entre énergie et performance.

4

Allocation de débit optimisant un compromis énergie/performance

Résumé

Dans ce chapitre, nous étudions le compromis entre consommation énergétique et performance dans les réseaux à partage de bande passante tel qu'Internet. On suppose ici que les nœuds du réseau peuvent réguler leur vitesse en fonction de la charge du système et on veut caractériser la politique de partage de bande passante optimale en termes de performance et de consommation énergétique. La politique stochastique optimale, solution d'un problème de décision Markovien, s'avère à la fois compliquée à analyser et à évaluer numériquement. On propose une approximation fluide déterministe du modèle stochastique. Pour le cas d'un réseau linéaire composé de deux liens, on vérifie que le modèle fluide donne une bonne approximation du modèle stochastique. Pour le cas d'un seul lien, on montre en utilisant le principe du maximum de Pontryagin que la politique optimale suit la loi $c\mu$ et on propose une expression explicite de la vitesse optimale.

Sommaire

4.1 Introduction

L'Internet ne cesse de voir son trafic augmenter. Selon une étude récente de Cisco [5], le trafic Internet a été multiplié par 4 de 2010 à 2015 et devrait atteindre 1 zetta-octet en 2015. Cette augmentation du trafic est susceptible d'être accompagnée d'une augmentation drastique de la consommation énergétique des réseaux. Proposer des solutions pour un compromis satisfaisant entre performance et consommation énergétique est donc important dans le sens ou cela réduira le coût de fonctionnement des réseaux. Dans ce chapitre, nous nous intéressons à l'allocation de débit aux flux permettant d'optimiser ce compromis.

Une solution possible pour réduire la consommation énergétique est de réguler d'une façon dynamique la capacité des liens en fonction de la charge du système de manière à respecter un certain critère de performance. Dans ce chapitre on suppose que la capacité est régulée en fonction de l'état du système. Notre objectif est de caractériser la politique de partage de bande passante optimale minimisant une certaine fonction coût, qui dépend à la fois de la consommation énergétique et de la performance ; la performance est exprimée ici en termes de délai. Pour cela, nous formulons le problème du partage de ressources entre les flux comme un processus de décision markovien (MDP pour *Markov Decision Processes*) [79].

La caractérisation de la politique stochastique optimale s'avère extrêmement complexe pour des réseaux généraux, même lorsque l'on ne prend pas en compte la consommation énergétique. En effet, les politiques minimisant le temps de séjour des flots dans le réseau ne sont connues que pour quelques topologies simples : un seul nœud [81, 73], réseau linéaire [90] ou bien réseau étoile [88]. Dans la plupart des cas, la politique optimale est une politique prioritaire. Prendre en considération le coût associé à la consommation énergétique et supposer que les capacités des liens sont variables et peuvent être régulées en fonction de l'état du système rend évidemment le problème encore plus complexe. En effet dans ce cas, il ne s'agit pas seulement de savoir dans quel ordre servir les flots, il faut aussi déterminer à quelle vitesse ils doivent être servis.

Si la caractérisation de la politique optimale est difficile d'un point de vue théorique, son calcul numérique est également complexe. En principe, on peut calculer la politique stochastique optimale à l'aide d'algorithmes classiques de la programmation dynamique comme par exemple *value-iteration* ou bien *policy-iteration* [79]. En pratique le calcul n'est hélas envisageable que sur des petits exemples. En effet, le calcul devient très couteux lorsque le nombre de classes de trafic est grand car le nombre d'états croît exponentiellement avec le nombre de classes de trafic. Ainsi, par exemple, pour un "petit" réseau avec 5 classes de trafic, si on suppose que chaque classe peut avoir au maximum 100 flots, on obtient une chaîne de Markov avec 1015 états.

Étant donné que la politique stochastique optimale du problème de départ est à la fois complexe à analyser et difficile à calculer numériquement, nous proposons de considérer la version fluide déterministe du problème originel. Nous suivons en cela l'approche proposée par Avram, Bertsimas et Richard dans [13]. L'approche fluide consiste à étudier tout d'abord un modèle déterministe "équivalent", également appelé modèle fluide, dont la dynamique est décrite par un ensemble d'équations différentielles. Le modèle fluide étant déterministe se prête plus facilement à l'analyse et au contrôle optimal. Dans une deuxième étape, il s'agit de définir une heuristique de résolution du problème stochastique originel en s'inspirant de la solution optimale du modèle fluide.

Cette technique a été appliquée dans plusieurs problèmes de contrôle optimal pour des chaines de Markov de grand dimension. Par exemple, Verloop utilise dans [90] cette technique pour construire une heuristique pour le partage de bande passante dans les réseaux linéaires. Son objectif est de minimiser le temps de séjour des flots dans le réseau et ne prend pas en compte la consommation énergétique. Il a été aussi démontré que la politique fluide et stochastique sont asymptotiquement équivalentes pour des systèmes à état élevés [70]. Dans notre contexte, un état élevé représente un état de congestion qui peut être du à une arrivée d'un nombre important de flots. Dans ce chapitre, nous supposerons que le réseau se trouve initialement dans un état de congestion suite, par exemple, à une foule subite. On peut donc s'attendre à ce que la politique optimal fluide pour vider le réseau à partir d'un état de congestion soit proche de la politique optimale stochastique.

L'avantage clef de cette approche est d'une part qu'elle permet d'obtenir une caractérisation analytique de la politique optimale fluide dans certains cas simples, et d'autre part qu'elle fournit une technique numérique passant à l'échelle pour le calcul de cette politique.

4.2 Description du modèle

4.2.1 Hypothèses et notations

On considère un réseau constitué d'un ensemble $\mathcal{L}$ de liens. On note C_l la capacité du lien l, et $\mathbf{C} = \{C_l\}_{l\in\mathcal{L}}$ le vecteur associé. Il y a K classes de trafic dans le système. On suppose que les flots de classe i arrivent selon un processus de poisson d'intensité λ_i et que leur volume suit une loi exponentielle de moyenne $\frac{1}{\mu_i}$. On note $\rho_i = \lambda_i/\mu_i$ l'intensité du trafic des flots de classe i. Ces flots sont acheminés via la route $r_i \subset \mathcal{L}$. Soit P la matrice d'incidence du réseau telle que l'entrée (i, j) est 1 si la classe i utilise le lien j et 0 sinon.

Soit $X_i(t)$ la variable aléatoire représentant le nombre de flots de classe i à l'instant t et $\boldsymbol{X}(t) \in \mathbb{N}^K$ le vecteur associé. On suppose par ailleurs que le lien $l \in \mathcal{L}$ peut servir le trafic avec une vitesse choisie dans l'intervalle $[0, C_l]$. En particulier, la vitesse d'un lien

peut dépendre du nombre de flots dans le système, c'est-à-dire de l'état du système $\boldsymbol{x}(t)$. On note $u_i(\boldsymbol{x})$ la capacité allouée aux flots de classe i dans l'état $\boldsymbol{x}$ et on suppose que ces flots se partagent équitablement la bande passante, c'est-à-dire que chacun reçoit le même débit $u_i(\boldsymbol{x})/x_i$. On note $\boldsymbol{u}(\boldsymbol{x})$ le vecteur $\boldsymbol{u}(\boldsymbol{x}) = (u_1(\boldsymbol{x}), u_2(\boldsymbol{x}), \ldots, u_K(\boldsymbol{x}))$ qui appartient à l'espace $\mathcal{U}$ des politiques de débit admissibles défini par

$$\mathcal{U} = \left\{ \boldsymbol{u} \in \mathbb{R}_+^K \ : \boldsymbol{u}P \leq C \right\}. \tag{4.1}$$

L'enjeu ici est de décider quelles classes servir en priorité et à quelle vitesse afin de minimiser les coûts liés au temps de présence des flots dans le réseau et à l'énergie consommée. Dans la suite, nous supposerons que l'énergie consommée pour opérer un lien à une vitesse x est proportionnelle à x^γ où $\gamma > 1$. L'énergie consommée liées au vecteur $\boldsymbol{u}$ est ainsi donnée par

$$\sum_{l \in \mathcal{L}} \left(\sum_{j \in \mathcal{F}} u_i P_{i,j} \right)^\gamma. \tag{4.2}$$

Etant donné une politique admissible $\boldsymbol{u}(\boldsymbol{x})$, le processus stochastique $\boldsymbol{X}(t)$ décrivant l'état du réseau est une chaîne de Markov (plus précisément, un processus de naissances et de morts multidimensionnel) dont les taux de transition sont donnés par

$$q(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \lambda_i & \text{si} \quad \boldsymbol{y} = \boldsymbol{x} + \mathbf{e}_i, \ i = 1, \ldots, K \\ \mu_i \, u_i(\boldsymbol{x}) & \text{si} \quad \boldsymbol{y} = \boldsymbol{x} - \mathbf{e}_i, \ i = 1, \ldots, K \end{cases}$$

4.2.2 Formulation du problème contrôle

Nous supposons qu'il existe un ensemble d'états $\mathcal{S}$ qui correspondent à des points de fonctionnement jugés souhaitables par l'opérateur du réseau, dans le sens où dans ces états la qualité de service des flots est satisfaisante pour les utilisateurs. Nous supposons de plus qu'à l'instant 0 le réseau se trouve dans un état initial $\boldsymbol{x}(0)$ éloigné de l'ensemble $\mathcal{S}$ (cf. figure 4.1) suite à un événement inattendu comme par exemple une foule subite.

Dans ce cas, l'objectif de l'opérateur est de ramener le réseau à un état acceptable $\boldsymbol{x}(t) \in \mathcal{S}$ avec un coût total minimal en termes de temps de séjour des flots dans le réseau et d'énergie consommée.

4.2.2.1 Modèle stochastique

Etant donnée une loi de commande $\boldsymbol{u} \in \mathcal{U}$, on note $T \in]0, +\infty[$ le premier temps où l'état du système se trouve dans un état appartenant à $\mathcal{S}$, c'est-à-dire

$$T = \inf\{t : \boldsymbol{X}(t) \in \mathcal{S} | \boldsymbol{X}(0) = \boldsymbol{x}(0)\}.$$

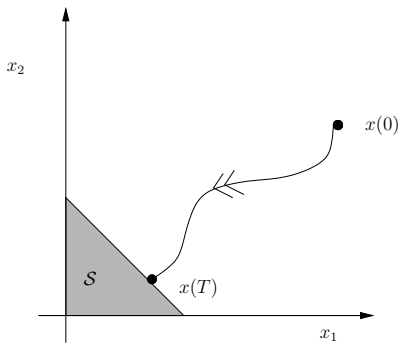


FIGURE 4.1 – Illustration du problème de contrôle à 2 classes.

Soulignons que T est une variable aléatoire qui dépend des réalisations des processus d'arrivée et de service, mais également de la loi de commande $\boldsymbol{u}$ appliquée. Pour simplifier les notations, nous ne rendons pas cette dépendance explicite. De plus, étant donnée un état $\boldsymbol{x}$ du système et une politique $\boldsymbol{u} \in \mathcal{U}$, on définit le coût $f(\boldsymbol{x}, \boldsymbol{u})$ du contrôle $\boldsymbol{u}$ appliqué dans l'état $\boldsymbol{x}$ comme

$$f(\boldsymbol{x}, \boldsymbol{u}) = \sum_{i \in \mathcal{F}} c_i x_i + \kappa \sum_{j \in \mathcal{L}} \left(\sum_{i \in \mathcal{F}} u_i P_{i,j} \right)^{\gamma}. \tag{4.3}$$

où c_i est le coût par unité de temps dans le système d'un flot de classe i. Le premier terme est une somme pondérée du nombre de flots dans le système, et est donc, en vertu de la loi de Little, directement lié au temps de séjour des flots dans le système. Comme détaillé au paragraphe précédent, le second terme représente le coût énergétique de l'allocation de débit $\boldsymbol{u}$. Dans (4.3), κ est un paramètre contrôlant les poids relatifs de la performance et de la consommation d'énergie.

L'opérateur souhaite calculer les vitesses des liens qui minimise le coût moyen pour ramener le système de l'état initial $\boldsymbol{x}(0)$ à un état appartenant à $\mathcal{S}$. Formellement, il s'agit de trouver une politique d'allocation de débit $\boldsymbol{u}^* : \mathbb{N}^N \to \mathcal{U}$ qui résout le problème d'optimisation suivant :

$$\text{Minimiser } \mathbb{E}_{\boldsymbol{x}(0)} \left\{ \int_0^T f\left(\boldsymbol{X}(t), \boldsymbol{u}(\boldsymbol{X}(t))\right) dt \right\}, \tag{4.4}$$

Remarque 1. *Le problème 4.4 est équivalent à un problème de plus court chemin stochastique (cf. [23]) où le coût dans les états appartenant à $\mathcal{S}$ est 0 et le taux de transition de ces états vers les autres états est 0. On peut donc supposer que l'horizon de temps est infini.*

Remarque 2. *Pour simplifier l'analyse, dans la suite on supposera que $\mathcal{S} = \{\mathbf{0}\}$, i.e.,*

notre objectif revient alors à vider le système avec un coût total minimal.

Le problème (4.4) peut être transformé en un problème à temps discret en uniformisant la chaine de Markov associée. Le temps de séjour dans l'état $\boldsymbol{x}$ est exponentiellement distribué de paramètre $\nu(\boldsymbol{x}) = \sum_i \lambda_i + \sum_i \mu_i\, u_i(\boldsymbol{x})$. En définissant

$$\nu = \sum_i \lambda_i + \max_{l\in\mathcal{L}}(C_l) \sum_i \mu_i, \tag{4.5}$$

on a $\nu(\boldsymbol{x}) \leq \nu$ pour tout $\boldsymbol{x} \in \mathbb{N}^N$. La version uniformisée de la chaine de Markov à temps continu est alors une chaine de Markov à temps discret dont les probabilités de transition sont données par

$$p(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} \frac{\lambda_i}{\nu} & \text{si} \quad \boldsymbol{y} = \boldsymbol{x} + \mathbf{e}_i,\ i \in \mathcal{F} \\ \frac{\mu_i}{\nu}\, u_i(\boldsymbol{x}) & \text{si} \quad \boldsymbol{y} = \boldsymbol{x} - \mathbf{e}_i,\ i \in \mathcal{F} \\ \frac{1}{\nu}\left[\nu - \sum_i(\lambda_i + \mu_i\, u_i(\boldsymbol{x}))\right] & \text{si} \quad \boldsymbol{y} = \boldsymbol{x}. \end{cases}$$

Dans sa version à temps discret, le problème (4.4) s'écrit :

$$\text{Minimiser } J_{\boldsymbol{\phi}}(\boldsymbol{x}(0)) = \frac{1}{\nu}\,\mathbb{E}\left\{\sum_{k=0}^{\infty} f(\boldsymbol{x}(k), \boldsymbol{u}(\boldsymbol{x}(k)))\right\}. \tag{4.6}$$

Notons que théoriquement la politique optimale peut être évaluée numériquement par des algorithmes classiques de programmation dynamique, comme par exemple l'algorithme *Value-Iteration* qui consiste à itérer plusieurs fois sur

$$J^{(k+1)}(\boldsymbol{x}) = \frac{1}{\nu} \min_{\boldsymbol{u}\in\mathcal{U}} \left[f(\boldsymbol{x},\boldsymbol{u}) + \sum_{\boldsymbol{y}} p(\boldsymbol{x},\boldsymbol{y}) J^{(k)}(\boldsymbol{y}) \right], \tag{4.7}$$

pour tout $\boldsymbol{x} \in \mathbb{N}^N$ et pour tout $k = 0, 1, 2, \ldots$. En pratique, ce calcul itératif nécessite le calcul à chaque itération du coût optimal à venir $J^{(k)}(\boldsymbol{x})$ pour chaque état initial possible $\boldsymbol{x}$. Ce calcul devient clairement inenvisageable lorsque le nombre de flots ou bien la charge du réseau deviennent important, à cause de l'explosion combinatoire de l'espace d'états.

4.2.3 Modèle fluide

La version stochastique du problème de contrôle optimal est à la fois difficile à analyser et compliquée à utiliser d'un point de vue numérique. On propose en conséquence d'étudier une approximation correspondante à la version fluide du problème stochastique originel. Plus précisément, on approxime le problème stochastique par sa version fluide dont la dynamique déterministe et à temps continu est donnée par

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\lambda} - \boldsymbol{\mu} \cdot \boldsymbol{u},$$

où $x_i(t)$ désigne la quantité de fluide associé à la classe i, $u_i(t)$ est le débit alloué a cette classe, et $\cdot$ désigne la multiplication composant par composant.

Le coût associé à la politique $\boldsymbol{u}$ à partir d'un état initial $\boldsymbol{x}_0$ dans un horizon $[0, T]$ est donné par

$$J(\boldsymbol{u}; \boldsymbol{x}_0) = \int_0^T f(\boldsymbol{x}(t), \boldsymbol{u}(t))dt.$$

Rappelons que T représente le premier instant où le système est vide, et donc que l'horizon T n'est pas fixe et dépend de la politique d'allocation de débit utilisée. Le problème de contrôle optimal devient dans sa version fluide :

$$\text{Minimiser } J(\boldsymbol{u}; \boldsymbol{x}_0) \tag{OPT}$$

sous les contraintes

$$\dot{x}_i(t) = \lambda_i - \mu_i u_i(t),\ i \in \mathcal{F}, \tag{4.8}$$

$$-x_i(t) \leq 0,\ i \in \mathcal{F}, \tag{4.9}$$

$$\boldsymbol{u}(t) \in \mathcal{U}, \tag{4.10}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{4.11}$$

$$\boldsymbol{x}(T) = \boldsymbol{0}. \tag{4.12}$$

4.2.4 Implémentation de la politique fluide

Dans ce paragraphe, on s'intéresse à l'implémentation de la politique fluide optimale $\boldsymbol{u}(t)$ minimisant (OPT). Il est essentiel ici de remarquer que le processus stochastique peut dévier de façon significative de sa limite fluide déterministe. Ainsi, appliquer la politique de contrôle en boucle ouverte sans prendre en considération l'état du réseau n'est clairement pas une approche appropriée. Plusieurs approches ont été récemment proposées pour implémenter la politique fluide optimale dans un réseau stochastique. On se restreint dans ce chapitre à l'approche dite "*discrete-review*" [53, 66, 24, 14, 89, 50].

L'idée globale de cette approche est d'aligner la trajectoire stochastique sur la trajectoire fluide optimale en adaptant périodiquement la politique optimale du modèle fluide. Puisque la trajectoire stochastique peut varier de manière imprévisible, elle s'éloignera de la trajectoire fluide au bout d'un certain temps. Afin de pallier les écarts dus à la variabilité des arrivées et de la taille des flots, la politique optimale fluide est recalculée périodiquement à des instants appelés instants de révision. A chaque instant de révision, le contrôleur calcule la politique optimale pour la version fluide avec pour état de départ l'état actuel du système stochastique. Cette politique est alors appliquée jusqu'au prochain instant de révision. Ainsi le contrôleur réagit et s'adapte aux écarts qui

Scénario	temps de calcul (s)
1	780s
2	1020s

TABLE 4.1 – temps de calcul de la politique optimale en utilisant l'algorithme *value-iteration* (la précision relative sur le coût optimal à venir est 10^{-3}).

sont induits par la dynamique stochastique.

On utilise la politique *discrete-review* dans sa forme la plus simple. Soit $t_k = k\,\Delta T$ pour $k = 0, 1, 2, \ldots$ les instants de révision. A un instant t_k, la politique de contrôle fluide est calculée en considérant l'état courant du système stochastique comme état initial. En divisant la période de révision m fois, on affecte le débit $u_i(t_k + \frac{n}{m}\Delta T)$ à la classe i dans l'intervalle de temps $\left[t_k + \frac{n}{m}\Delta T, t_k + \frac{n+1}{m}\Delta T\right)$.

Pour valider cette approche, nous considérons un réseau linéaire de 2 liens et 3 classes de flots (cf. 4.2). La capacité des liens est : $C_1 = 10$ et $C_2 = 20$. On suppose que κ=1, γ=3 et on considère les scénarii suivants :

— Scenario 1 : $\rho_0 = 0.2$, $\rho_1 = 0.3$, $\rho_2 = 0.1$, $c_0 = 1$, $c_1 = 2$ et $c_2 = 2$.

— Scenario 2 : $\rho_0 = 0.4$, $\rho_1 = 0.1$, $\rho_2 = 0.5$, $c_0 = 2$, $c_1 = 1$, $c_2 = 6$.

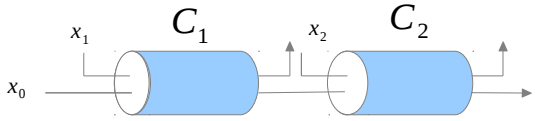


FIGURE 4.2 – réseau linéaire.

La politique optimale peut être calculée en utilisant l'algorithme *value iteration* ou bien avec *policy-interation* [23]. La politique optimale fluide peut être calculée en utilisant un solveur comme Bocop, une boîte à outils open-source pour résoudre les problèmes de contrôle optimal [1]. Comme indiqué dans les tableaux 4.1 et 4.2, le temps requis pour résoudre le problème de contrôle fluide est nettement inférieur au temps de calcul de la politique stochastique optimale.

La figure 4.3 illustre des exemples de trajectoires obtenues pour les 3 classes sous la politique stochastique optimale et la politique fluide optimale pour le premier scénario en supposant que l'état initial est $\boldsymbol{x}(0) = (3, 5, 5)$ et le second scénario avec un état initial $\boldsymbol{x}(0) = (4, 5, 3)$. On observe que les trajectoires obtenues sous la politique fluide en utilisant l'approche *discrete-review* sont très proches des trajectoires obtenues sous

Scénario	État initial	temps de calcul (s)
1	(3,5,5)	16 s
2	(4,5,3)	10 s

TABLE 4.2 – temps de calcul de la politique optimale en utilisant Bocop.

la politique stochastique optimale.

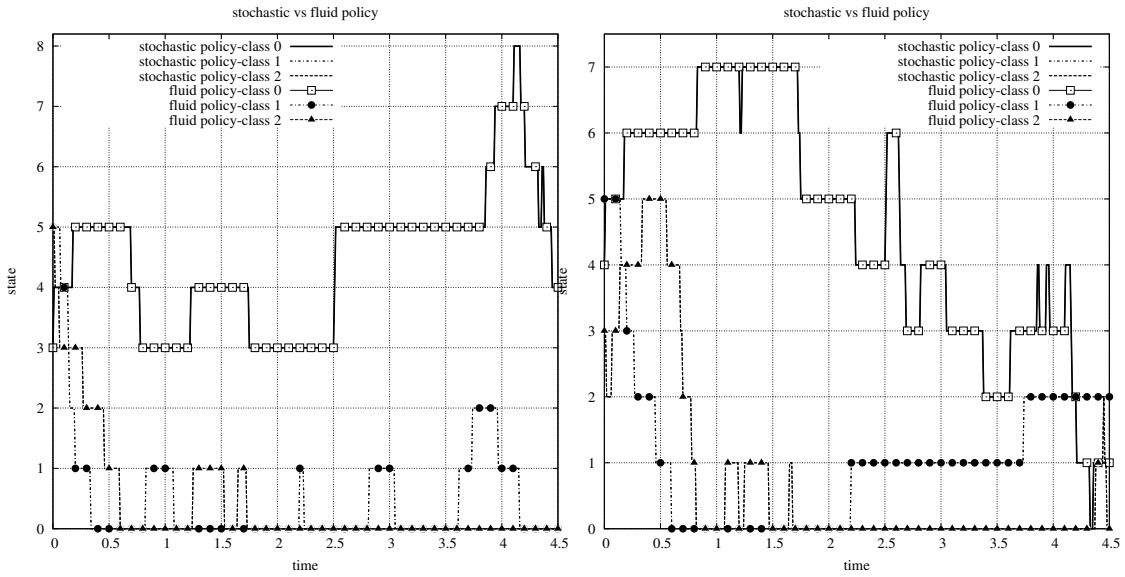


FIGURE 4.3 – Exemples de trajectoire obtenue sous la politique fluide (Bocop) et la politique stochastique $x_0(0)$=4, $x_1(0)$=5 et $x_2(0)$=3 (gauche) et pour $x_0(0)$=3, $x_1(0)$=5 et $x_2(0)$=5 (droite).

4.3 Lien isolé

Dans cette section, on considère le cas d'un seul lien de capacité C. L'enjeu ici est de caractériser la politique optimale minimisant le coût total. Notons qu'une solution a été déjà proposée dans [13] lorsque les coûts ne prennent pas en compte la consommation énergétique. Il a été démontré dans ce cas que donner la priorité à la classe i de plus grand indice $c_i\mu_i$ minimise le coût total.

On montre, tout d'abord, que dans le modèle stochastique originel, la politique optimale est prioritaire. Cependant, vue la difficulté du problème stochastique, il s'avère difficile d'identifier le critère de priorité.

En considérant la version fluide du problème, on détermine le critère de priorité : on montre que la politique optimale suit la règle $c\mu$. On propose en outre une expression explicite de la vitesse optimale.

Finalement, on vérifie numériquement que la politique fluide présente une bonne approximation de la politique stochastique et on observe aussi que la politique stochastique coïncide avec la règle $c\mu$ [33].

4.3.1 Modèle stochastique

Dans la suite, on notera $J(\boldsymbol{x})$ le coût correspondant à la politique $\boldsymbol{u}^* : \mathbb{N}^N \to \mathcal{U}$ minimisant (4.6). Dans ce cas, l'équation de Bellman est donnée par

$$\begin{aligned} J(\boldsymbol{x}) \quad = \quad & \frac{1}{\nu}\left[\sum_i c_i x_i + \nu J(\boldsymbol{x}) + \sum_i \lambda_i \Delta_{\boldsymbol{x}+\mathbf{e}_i}(i)\right. \\ & \left. + \min_{\boldsymbol{u}\in\mathcal{U}}\left[\kappa\left(\sum_i u_i\right)^\gamma - \sum_i \mu_i\, u_i \Delta_{\boldsymbol{x}}(i)\right]\right], \end{aligned} \tag{4.13}$$

tel que $\Delta_{\boldsymbol{x}}(i) = J(\boldsymbol{x}) - J(\boldsymbol{x}-\mathbf{e}_i)$ est le coût optimal partiel dans l'état $\boldsymbol{x}$. Une politique optimale $\boldsymbol{u}$ à l'état $\boldsymbol{x}$ minimise

$$\kappa\left(\sum_i u_i\right)^\gamma - \sum_i \mu_i\, u_i \Delta_{\boldsymbol{x}}(i).$$

Soit s tel que $\sum_i u_i = s$, le problème revient à maximiser la somme $\sum_i \mu_i\, u_i \Delta_{\boldsymbol{x}}(i)$. La politique optimale dans ce cas est clairement : $u_k = s$ pour tout k tel que pour tout i on a $\mu_k \Delta_{\boldsymbol{x}}(k) \geq \mu_i \Delta_{\boldsymbol{x}}(i)$, et $u_i = 0$ pour $i \neq k$. On en déduit que la politique optimale donne entièrement la priorité à une seule classe. Il reste tout de même difficile d'identifier la classe prioritaire. C'est pourquoi on propose dans le paragraphe suivant d'étudier la version fluide du problème afin d'identifier le critère de priorité.

4.3.2 Modèle fluide

Pour le cas d'un lien unique, le problème (OPT) devient

$$\text{minimiser} \int_0^T f(\boldsymbol{y}, \boldsymbol{u}) dt \tag{OPT-R}$$

sous contraintes

$$\dot{y}_i(t) = \rho_i - u_i(t),\ i \in \mathcal{F}, \tag{4.14}$$

$$-y_i(t) \leq 0,\ i \in \mathcal{F}, \tag{4.15}$$

$$\boldsymbol{u}(t) \in \mathcal{U}, \tag{4.16}$$

$$\boldsymbol{y}(T) = \boldsymbol{y}_0, \tag{4.17}$$

$$\boldsymbol{y}(T) = \boldsymbol{0}, \tag{4.18}$$

où $y_i = \frac{x_i}{\mu_i}$ et f est définie par

$$f(\boldsymbol{x}, \boldsymbol{u}) = \sum_{i \in \mathcal{F}} \tilde{c}_i x_i + \kappa \sum_{j \in \mathcal{L}} \left(\sum_{i \in \mathcal{F}} u_i P_{i,j} \right)^{\gamma}, \tag{4.19}$$

avec $\tilde{c}_i = c_i \mu_i$ pour tout $i \in \mathcal{F}$.

Notons que le problème (OPT-R) a des contraintes pures sur l'état ce qui rend le problème plus difficile à résoudre. On propose de résoudre le problème au moyen de l'*approche adjointe directe* [54] qui consiste en la relaxation des contraintes pures sur l'état pour déterminer les valeurs des multiplicateurs de Lagrange.

Le Lagrangien associé au problème (OPT-R) est

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\beta}) = f(\boldsymbol{y}, u) + \sum_i (\theta_i(\rho_i - u_i) - \eta_i y_i - \beta_i u_i),$$

où θ_i est la variable adjointe de la classe i, η_i désigne la contrainte pure sur l'état $x_i \geq 0$, et β_i associée à $u_i \geq 0$. Pour le moment, on néglige la contrainte associée à la somme des débits. Autrement dit, on supposera ici que la capacité du lien est infinie.

Une condition nécessaire pour l'optimalité de $\boldsymbol{u}$ est

$$\dot{\theta}_i(t) = -\dot{\mathcal{L}}_{y_i} \tag{4.20}$$

$$\dot{y}_i(t) = \dot{\mathcal{L}}_{\theta_i} \tag{4.21}$$

$$\boldsymbol{u}(t) = \arg\max_{\boldsymbol{u} \in \Phi} \mathcal{L} \tag{4.22}$$

$$\eta_i y_i = 0,\ \boldsymbol{\eta} \geq 0, \tag{4.23}$$

$$\beta_i u_i = 0,\ \boldsymbol{\beta} \geq 0, \tag{4.24}$$

Soit τ_k le premier instant où une contrainte est satisfaite en tant qu'égalité. On a,

$$\theta(t_k+) = \theta(t_k-) - \nu_k$$

De [54], on peut déduire que les variables adjointes sont continues si les trajectoires associées n'admettent pas de sommet ($\dot{y}_i(t_k-) \neq 0$ quand y_i), ce que nous supposerons dans la suite.

En remplaçant par l'expression exacte du coût (4.3), une condition nécessaire d'optimalité du contrôle (4.22) devient

$$\dot{\mathcal{L}}_{\boldsymbol{u}} = 0,$$

c'est-à-dire,

$$\kappa\gamma \left(\sum_k u_k(t) \right)^{\gamma-1} - (\theta_k(t) + \beta_k(t)) = 0, \ \forall k \in \mathcal{P}. \tag{4.25}$$

4.3.2.1 Optimalité de la règle $c\mu$

Soit $\mathcal{P}(t) = \{i : y_i(t) > 0\}$, i.e., l'ensemble des classes actives à l'instant t.

Theorem 3. *Pour tout $t > 0$,*

1. *$u_i(t) > 0$ si $y_i(t) = 0$ ou $i = \arg\max\{\tilde{c}_j : j \in \mathcal{P}(t)\}$;*
2. *si $y_i(t) = 0$, alors $u_i(t) = \rho_i$.*

Le Théorème 3 montre que la politique optimale dans ce cas coïncide avec la loi $c\mu$. Le même résultat a été déjà obtenu pour des coûts ne prenant pas en compte la consommation énergétique.

L'idée de la suite de la preuve est de commencer à partir de l'instant final T et déterminer la classe prioritaire à un instant antérieur t.

Proposition 3. *Pour chaque intervalle $[t_1, t_2]$ tel que $\mathcal{P}(t) = \mathcal{P}(t_1)$, $\forall t \in [t_1, t_2]$, $u_i(t)u_j(t) = 0$, sauf pour un nombre finis d'instants.*

Démonstration. Pour toute classe $k \in \mathcal{P}(t)$, les conditions de complémentarité (4.23) impliquent que $\eta_k(t) = 0$. D'après (4.20), on obtient

$$\dot{\theta}_k(t) = -\tilde{c}_k, \forall k \in \mathcal{P}(t),$$

En intégrant, on a

$$\theta_k(t) = -\tilde{c}_k t + d_k, \forall k \in \mathcal{P}(t), \forall t \in [t_1, t_2], \tag{4.26}$$

où les d_k sont les constantes d'intégration.

Soient i et j deux classes dans $\mathcal{P}(t)$ tel qu'on a $u_i(t) > 0$ et $u_j(t) > 0$. D'après les conditions de complémentarité (4.24), on a $\beta_i(t) = \beta_j(t) = 0$. On déduit de (4.25) que

$$\theta_i(t) = \theta_j(t).$$

À partir de (4.26), on voit que θ_i et θ_j sont deux fonctions linéaires dans $[t_1, t_2]$ de pentes distinctes, et ne peuvent donc se croiser qu'en un seul point dans cet intervalle. On en déduit que deux classes i, j dans $\mathcal{P}(t)$ ne peuvent avoir simultanément un débit non nul $(u_i > 0, u_j > 0)$ qu'en au plus un seul instant appartenant à $[t_1, t_2]$. Or, le nombre de classes est supposé fini, d' où le résultat. □

Corollaire 2. *Il existe τ_N tel que pour tout $t \in [\tau_N, T]$ il y a une seule classe dans $\mathcal{P}(t)$.*

Démonstration. On sait qu'à l'instant T le système est vide ($\mathcal{P}(t)$=$\{0\}$). Or d'après la proposition 3, on sait aussi que deux classes actives ne peuvent pas être servie simultanément. Donc l'unique solution pour avoir un système vide à l'instant T est que toutes les classes soient vides sauf une seule à partir d'un instant τ_N antérieur à T. □

Ce résultat montre que pour chaque intervalle appartenant à $[0, T]$, une seule classe dans l'ensemble $\mathcal{P}(t)$ est servie. De plus, l'intervalle $[0, T]$ peut être divisé en des sous-intervalles où seulement une seule classe est servie dans chacun de ces sous-intervalles.

Preuve du théorème 3. Soit $\tilde{c}_1 > \tilde{c}_2 > \ldots > \tilde{c}_N$. D'après le corollaire 2, il existe τ_N tel qu'une seule classe a une quantité de fluide non nulle dans $[\tau_N, T]$. On a aussi, pour tout $k \notin \mathcal{P}(t)$, $u_i = \rho_i$, d'où toutes les classes ont un débit strictement positive dans $[\tau_N, T]$. D'après (4.25) et les conditions de complémentarité (4.24), on a

$$\theta_i(t) - \theta_j(t) = 0, \forall i, j, \ \forall t \in [\tau_N, T], \tag{4.27}$$

c'est-à-dire

$$\dot{\theta}_i(t) - \dot{\theta}_j(t) = 0, \ \forall i, j, \ \forall t \in (\tau_N, T), \tag{4.28}$$

et d'après (4.20) et (4.23), on a

$$\dot{\theta}_i = -\tilde{c}_i + \eta_i(t), \ i \notin \mathcal{P}(t), \text{ et } \eta_i \geq 0, \tag{4.29}$$

$$\dot{\theta}_i = -\tilde{c}_i, \ i \in \mathcal{P}(t). \tag{4.30}$$

Les conditions (4.28), (4.29), et (4.30) sont satisfaites si et seulement si $\eta_N = 0$. D'où l'on déduit que la classe N, d'indice $\tilde{c}$ le plus petit, est la dernière à être servie.

On note τ_N le dernier temps où une classe parmi les $N-1$ autres n'est pas vide. Il existe alors un certain τ_{N-1} tel qu'il y a deux classes avec une quantité de fluide non nulle dans l'intervalle $[\tau_{N-1}, \tau_N]$.

D'après (4.29) et (4.30), l'autre classe avec une quantité de fluide non nulle ne peut être que la classe $N-1$. À partir de la proposition 3, on déduit qu'une seule classe peut être servie dans cet intervalle. Dans la suite, on montre que seulement la classe $N-1$ peut être servie.

De (4.20) et (4.23), on a $\forall t \in [\tau_{N-1}, \tau_N]$,

$$\theta_{N-1}(t) = \tilde{c}_N(\tau_{N-1} - t) + d_N, \tag{4.31}$$
$$\theta_N(t) = \tilde{c}_{N-1}(\tau_N - t) + d_{N-1}. \tag{4.32}$$

on déduit de (4.27) que les constantes d_N et d_{N-1} sont égales. On a alors,

$$\theta_{N-1}(t) - \theta_N(t) = (\tilde{c}_{N-1} - \tilde{c}_N)(\tau_N - t) \geq 0, \forall t \in [\tau_{N-1}, \tau_N].$$

On désigne par i la classe servie et par j l'autre classe non servie dans l'intervalle $[\tau_{N-1}, \tau_N]$. D'après (4.22) et les conditions de complémentarité (4.24),

$$\theta_i(t) = \theta_j(t) + \beta_j(t).$$

Sachant que $\beta_j(t) \geq 0$, cette équation est satisfaite si et seulement si $i = N-1$ et $j = N$. On en déduit que l'unique classe servie dans l'intervalle $[\tau_{N-1}, \tau_N]$ est la classe $N-1$.

Ces arguments peuvent être appliqués récursivement pour montrer que parmi les classes appartenant à $\mathcal{P}(t)$ seulement la classe ayant le plus grand indice $\tilde{c}$ reçoit un débit non nul. □

4.3.2.2 Vitesse optimale

Le Théorème 3 montre qu'il est optimal d'allouer $u_i(t) = \rho_i$ pour toutes les classes i tel que $y_i(t) = 0$, et de suivre la règle $c\mu$ pour les autres classes non vides, c'est-à-dire de servir la classe k avec l'indice $\tilde{c}_i$ le plus élevé parmi les classes actives. Cependant, le Théorème 3 ne caractérise pas entièrement la politique optimale tant qu'on n'a pas déterminé le débit alloué à la classe $k = \arg\max\{\tilde{c}_j : j \in \mathcal{P}(t)\}$. Notons que pour cette classe k, on a $\beta_k(t) = 0$. Pour tout $t \in [\tau_k, \tau_{k+1})$, on a $u_k(t) > 0$ $\beta_k(t) = 0$, on en déduit alors de (4.25),

$$u_k(t) = \left(\frac{\theta_k(t)}{\kappa\gamma}\right)^{\frac{1}{\gamma-1}} - \sum_{i \notin \mathcal{P}(t)} \rho_i, \tag{4.33}$$

D'après le Théorème 3, la politique optimale est donnée par

$$u_i(t) = \begin{cases} \left(\frac{\theta_k(t)}{\kappa\gamma}\right)^{\frac{1}{\gamma-1}} - \sum_{j \notin \mathcal{P}(t)} \rho_j & \text{if } i = k \\ 0 & \text{if } i \in \mathcal{P}(t) \setminus \{k\} \\ \rho_i & \text{if } i \notin \mathcal{P}(t) \end{cases} \tag{4.34}$$

On déduit alors de (4.25) que

Pour déterminer la politique optimale $(\boldsymbol{u}(t), T)$, il suffit de déterminer pour chaque

trajectoire $\boldsymbol{y}(t)$, le vecteur associé aux variables adjointes $\boldsymbol{\theta}(t)$ ainsi que le temps final T.

Notons que les deux équations (4.20) et (4.21) fournissent $2N$ equations différentielles, de telle sorte que $\boldsymbol{y}(t)$ et $\boldsymbol{\theta}(t)$ sont connus à chaque instant t à une constante près. En considérant le temps final T, on a en tout $2N+1$ constantes à déterminer. Notons que si l'on détermine une seule de ces constantes, les autres peuvent être déduites des $2N$ conditions de frontière $y_i(0)$ et $y_i(T) = 0$, $i = 1, \ldots, N$.

La valeur de θ_N s'obtient de la condition de transversalité suivante [39] :

$$\mathcal{L}(\boldsymbol{y}(T), \boldsymbol{u}(T), \boldsymbol{\theta}(T), \boldsymbol{\eta}(T), \boldsymbol{\beta}(T)) = 0 \tag{4.35}$$

On a $y_i(T) = 0$ pour tout i et $u_i(T) = \rho_i$ pour tout $i \neq N$. On en déduit

$$\kappa \left[\sum_{i \neq N} \rho_i + u_N(T) \right]^{\gamma} + \theta_N(T) \left[\rho_N - u_N(T) \right] = 0. \tag{4.36}$$

À partir de (4.33), l'équation (4.36) devient

$$\kappa \left[\sum_{i \neq N} \rho_i + u_N(T) \right]^{\gamma} + \kappa\gamma \left[\sum_{i \neq N} \rho_i + u_N(T) \right]^{\gamma-1} \left[\rho_N - u_N(T) \right] = 0, \tag{4.37}$$

On en déduit que $u_N(T)$ est l'unique racine de l'équation (4.37).

$$u_N(T) = \frac{\sum_{i \neq N} \rho_i + \gamma \rho_N}{\gamma - 1}. \tag{4.38}$$

On observe que $u_N(T)$ est insensible à T et aux conditions initiales, et dépend uniquement de l'intensité du trafic des classes et γ. On notera dans ce qui suit $\alpha = 1/(\gamma - 1)$, et $\tau_{N+1} = T$. À partir de (4.36) et (4.38), $\theta_N(T)$ s'écrit,

$$\theta_N(T) = \kappa\gamma \left(\frac{\gamma}{\gamma - 1} \sum_i \rho_i \right)^{\gamma-1}. \tag{4.39}$$

Dans la suite, on explique comment calculer la vitesse optimale à différents instants de temps (cf. Proposition 4).

Proposition 4. *Pour une valeur de T donnée et en utilisant l'expression de $\theta_N(\tau_{N+1})$ tel que détaillé dans* (4.39)*, on a les résultats suivants*

1. *En supposant que $\tau_{k+1} > 0$, la vitesse optimale d'une classe k, pour $k = 1, \ldots, N$,*

est donnée par

$$u_k(t) = \left(\frac{\tilde{c}_k(\tau_{k+1} - t) + \theta_k(\tau_{k+1})}{\kappa\gamma} \right)^{\frac{1}{\gamma-1}} - \sum_{i \notin \mathcal{P}(t)} \rho_i,\ t \in [\tau_k, \tau_{k+1}), \tag{4.40}$$

tel que τ_k est l'unique solution de

$$\begin{aligned} y_k(0) + \rho_k \tau_{k+1} + \sum_{i \notin \mathcal{P}(t)} \rho_i(\tau_{k+1} - \tau_k) &= \frac{\kappa\gamma}{(\alpha+1)\tilde{c}_k} \left(\left(\frac{\tilde{c}_k(\tau_{k+1} - \tau_k) + \theta_k(\tau_{k+1})}{\kappa\gamma} \right)^{\alpha+1} \right. \\ &- \left. \left(\frac{\theta_k(\tau_{k+1})}{\kappa\gamma} \right)^{\alpha+1} \right) \end{aligned} \tag{4.41}$$

dans l'intervalle $(-\infty, \tau_{k+1})$.

2. *Pour $k = 2, \ldots, N$, on a*

$$\theta_{k-1}(\tau_k) = \theta_k(\tau_{k+1}) + \tilde{c}_k(\tau_{k+1} - \tau_k). \tag{4.42}$$

3. *Pour $k = 1, 2, \ldots, N$, τ_k est une fonction croissante en T.*
4. *T est l'unique solution de $\tau_1 = 0$.*

Démonstration. 1. Soit $k < N$. Pour tout $t \in [\tau_k, \tau_{k+1})$, l'hypothèse de continuité et l'équation (4.27) peuvent être appliquées pour obtenir $d_k = d_{k+1} = \theta_{k+1}(\tau_{k+1})$. Il s'en suit,

$$\theta_k(t) = \tilde{c}_k(\tau_{k+1} - t) + \theta_{k+1}(\tau_{k+1}),\ t \in [\tau_k, \tau_{k+1}). \tag{4.43}$$

On déduit alors l'équation (4.40) à partir de (4.33) et (4.43).

Pour (4.41), on utilise les dynamiques (4.14) de y_k et les conditions de frontière $y_k(\tau_{k+1}) = 0$, c'est-à-dire,

$$y_k(\tau_{k+1}) = y_k(0) + \rho_k\, \tau_{k+1} - \int_{\tau_k}^{\tau_{k+1}} u_k(t)\, dt, = 0.$$

en combinant cette équation avec (4.40), on obtient (4.41).

On montre maintenant que τ_k est l'unique solution de (4.41). Pour une valeur de τ_{k+1} donnée, on note $\Delta_k = \tau_{k+1} - \tau_k$. L'équation (4.41) s'écrit

$$g(\Delta) = y_k(0) + \rho_k \tau_{k+1},$$

tel que g désigne les termes restants dans l'équation (4.41). Notons que le terme de droite de l'équation ci-dessous est strictement positive alors que $g(0) = 0$. Notons

aussi que, $g'(\Delta) = u_k(\tau_k) > 0$ pour tout $\Delta > 0$. Par conséquent, g est une fonction décroissante dans l'intervalle $[0, \infty)$ et il existe une unique valeur de Δ satisfaisant cette équation. Il existe donc une seule valeur de $\tau_k \in (-\infty, \tau_{k+1})$.

2. En intégrant (4.30), on obtient

$$\theta_k(\tau_k) = \tilde{c}_k(\tau_{k+1} - \tau_k) + \theta_k(\tau_{k+1}),$$

tel que $\theta_k(\tau_{k+1})$ est défini à partir de l'hypothèse 4.25. En remplaçant k par $k-1$ dans (4.43) et en combinant (4.43) avec l'équation précédente, on obtient

$$\theta_{k-1}(\tau_k) = \theta_k(\tau_{k+1}) + \tilde{c}_k(\tau_{k+1} - \tau_k).$$

3. On utilise la récursivité pour montrer cette propriété. On considère $k < N$ et on suppose que τ_i est croissante pour tout $i > k$. On note $\rho_{-i} = \sum_{j<i} \rho_i$. Notons que τ_k est définit à partir de (4.41) et que τ_i et θ_i, $\forall i$ dépendent de T. En dérivant (4.41) par rapport à T, on obtient

$$\rho_k \tau'_{k+1} + \rho_{-k}(\tau'_{k+1} - \tau'_k) = u(\tau_k)(\tau'_{k+1} - \tau'_k) + \frac{u(\tau_k) - u(\tau_{k+1})}{\tilde{c_k}} \theta'_k,$$

avec $u(t) = \sum_i u_i(t)$ désigne la vitesse totale du serveur à un instant t. Après un simple calcul élémentaire, on obtient,

$$\tau'_k = \tau'_{k+1} + (u(\tau_k) - \rho_{-k})^{-1} \left[-\rho_k \tau'_{k+1} + \frac{u(\tau_k) - u(\tau_{k+1})}{\tilde{c_k}} \theta'_k \right]. \tag{4.44}$$

Notons d'abord que pour $k = N$, $\theta_N(T)$ est une constante indépendante de T, et que $\tau_{k+1} = T$. D'où, (4.44) devient

$$\tau'_k = 1 - (u(\tau_k) - \rho_{-k})^{-1} \rho_k.$$

Notons que $u(\tau_k) - \rho_{-k}$ est la vitesse de la classe k dans l'intervalle de temps $[\tau_k, \tau_{k+1})$. Cette vitesse est supérieure à ρ_k, et donc le terme de droite de l'équation ci-dessus est positive. La propriété est donc vrai pour $k = N$.

Pour $k < N$, considérons d'abord le terme θ'_k dans (4.44). En développant la récursivité (4.42), on obtient

$$\theta_k(\tau_{k+1}) = \tilde{c}_N T - \tilde{c}_{k+1}\tau_{k+1} + \sum_{j=k+2}^{N} (\tilde{c}_{j-1} - \tilde{c}_j)\tau_j,$$

En différenciant donc l'équation ci-dessus par rapport à T, on a

$$\theta_k(\tau_{k+1})' = \tilde{c}_N - \tilde{c}_k\tau'_{k+1} + \sum_{j=k+2}^{N} (\tilde{c}_{j-1} - \tilde{c}_j)\tau'_j,$$

Étant donnée que $\tilde{c}_{j-1} - \tilde{c}_j > 0$ et τ'_j sont aussi positives (hypothèse de récursivité), $\theta'_k > -\tilde{c}_{k+1}\tau'_{k+1} > -\tilde{c}_k\tau'_{k+1}$. En substituant cette inégalité dans (4.44), on en déduit que

$$\begin{aligned}
\tau'_k &> \tau'_{k+1} + (u(\tau_k) - \rho_{-k})^{-1} \left[-\rho_k - u(\tau_k) + u(\tau_{k+1})\right] \tau_{k+1}{}' \\
&= \tau'_{k+1} + (u(\tau_k) - \rho_{-k})^{-1} \left[-(u(\tau_k) - \rho_{-k}) + u(\tau_{k+1}) - \rho_{-(k+1)})\right] \tau_{k+1}{}' \\
&= (u(\tau_k) - \rho_{-k})^{-1} \left[u(\tau_{k+1}) - \rho_{-(k+1)})\right] \tau_{k+1}{}' \\
&> 0 \quad \text{(étant donné que } u(\tau_{k+1}) - \rho_{-(k+1)} = u_{k+1}(\tau_{k+1}) > 0).
\end{aligned}$$

4. La preuve repose sur le fait que τ_1 est une fonction croissante en T et donc il existe une unique valeur de T tel que $\tau_1 = 0$.

□

Remarque 3. *On explique ici comment calculer récursivement les vitesses optimales pour les différentes classes à des différents instants. La récursivité ici s'effectue d'une façon décroissante de N à 1. Une fois que $\theta_N(\tau_{N+1})$ est déterminé, on peut déduire τ_N et $u_N(t)$, $t \in [\tau_N, \tau_{N+1})$. De (4.42), on peut calculer $\theta_{N-1}(\tau_N)$, et en déduire ainsi τ_{N-1} et $u_{N-1}(t)$, $t \in [\tau_{N-1}, \tau_N)$. On répète cette procédure jusqu'à $k = 1$. Finalement, pour calculer la valeur de T tel que $\tau_1 = 0$, on procède comme suit : notons d'abord que le calcul de T repose sur le fait que τ_k est croissante en T, pour $k = 1, 2, \ldots, N$ et donc il existe une valeur de T assez grande à partir de laquelle on a $\tau_k > 0$, pour $k = 1, 2, \ldots, N$. Plus précisément, l'idée est de recalculer plusieurs fois les valeurs de τ_k, pour $k = 1, 2, \ldots, N$ en augmentant à chaque fois la valeur de T jusqu'à ce qu'on obtient $\tau_k > 0$, pour $k = 1, 2, \ldots, N$.*

4.3.3 Résultats numériques

On décrit dans ce paragraphe les résultats numériques obtenus pour différents scénarii. Pour chaque scénario, on compare les résultats obtenus par la résolution du plus court chemin stochastique en utilisant l'algorithme *value-iteration* avec ceux obtenus par la politique fluide obtenue en utilisant les résultats de la Proposition 4 et aussi en utilisant Bocop.

Premier scénario

On considère comme premier exemple un lien partagé par trois classes de trafic. On suppose que κ=1 et γ=3. On considère les paramètres suivants : ρ_0=0.2, ρ_1=0.1, ρ_2=0.3, $\mu_0 c_0$=5, $\mu_1 c_1$=6 et $\mu_2 c_2$=4.

Dans la figure 4.4, on trace le débit optimal fluide alloué à chaque classe calculé analytiquement par les résultats de la Proposition 4 et Bocop. Ces débits sont comparés aux débits obtenus avec le modèle stochastique ; la quantité de fluide de chaque classe correspond à l'état du système [1]

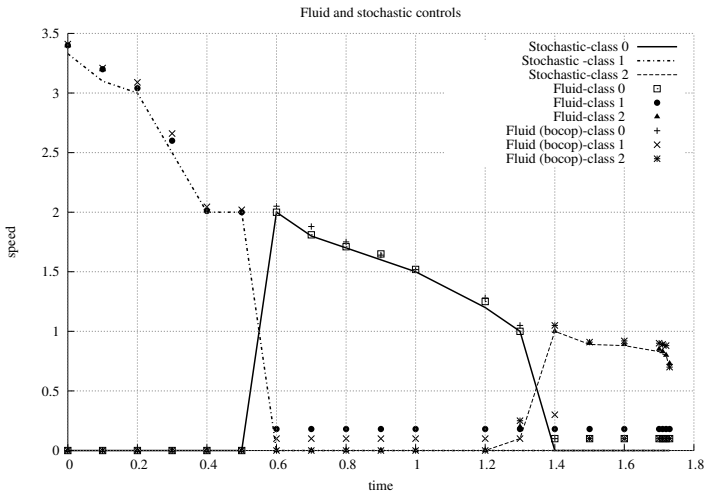


FIGURE 4.4 – Vitesse optimale fluide vs. Vitesse optimale stochastique pour le scénario 1 avec $x_0(0)$=10, $x_1(0)$=11 et $x_2(0)$=12.

Dans les figures 4.5 (resp. figure 4.6), on compare aussi des exemples de trajectoires obtenues sous la politique stochastique optimale et la politique fluide optimale en utilisant la méthode *discrete-review* pour un état initial : $\boldsymbol{x}(0) = (10, 11, 12)$ (resp. $\boldsymbol{x}(0) = (10, 11, 12)$). On observe que les trajectoires des classes 0, 1 et 2 obtenues sous la politique optimale fluide sont assez proches de celles obtenues avec la politique stochastique.

1. étant donné que la politique optimale stochastique est définie uniquement sur des nombres entiers, on utilise une interpolation bilinéaire pour calculer sa valeur en un point quelconque à partir des quatre nombres entiers les plus proches.

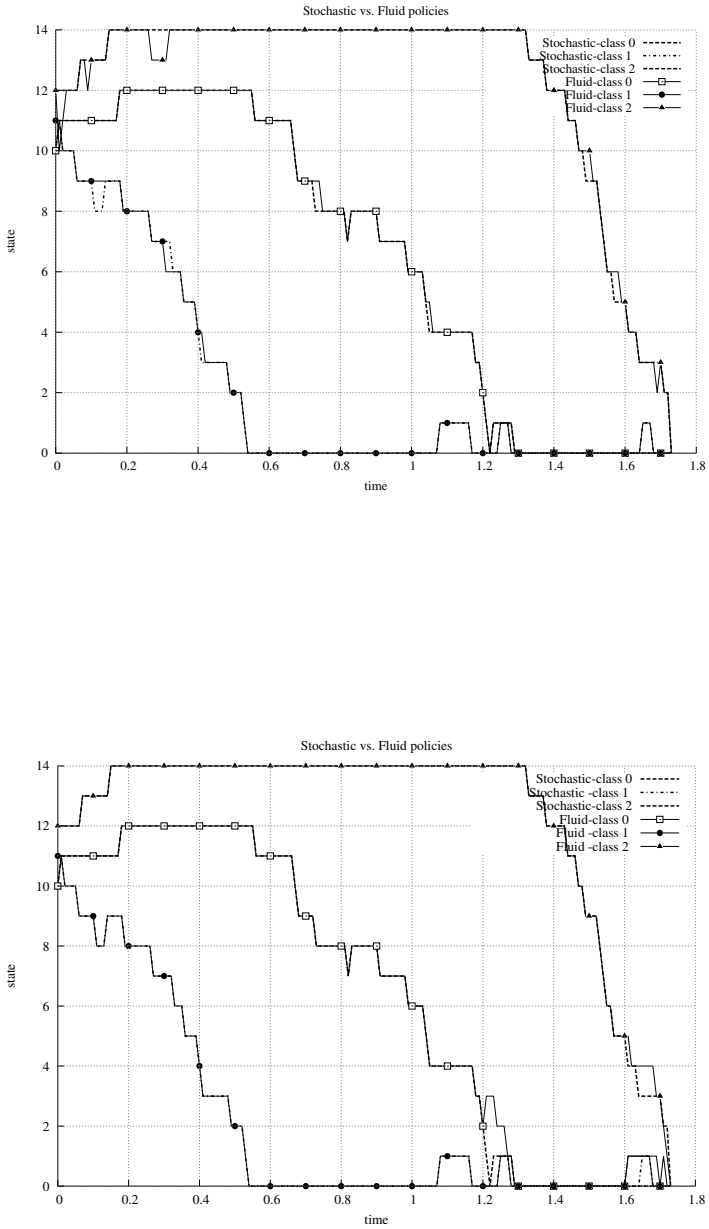
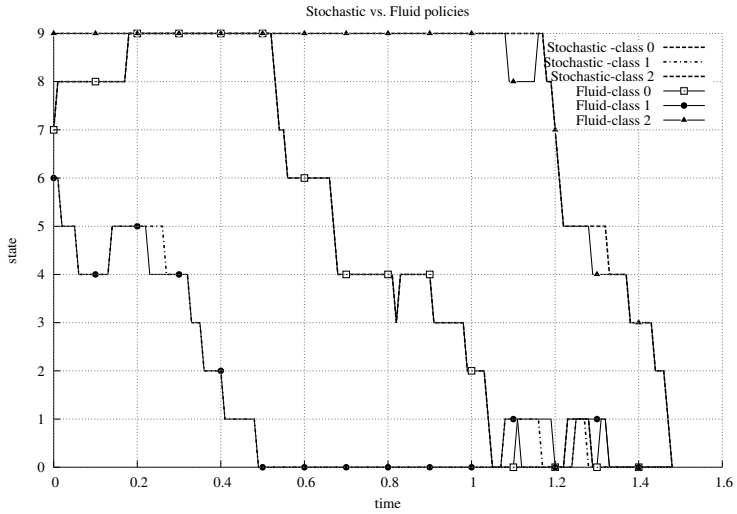


FIGURE 4.5 – Exemple de trajectoire obtenue sous la politique stochastique et la politique fluide (calcul analytique (haut) et Bocop (bas)) pour $x_0(0)$=10, $x_1(0)$=11 et $x_2(0)$=12.



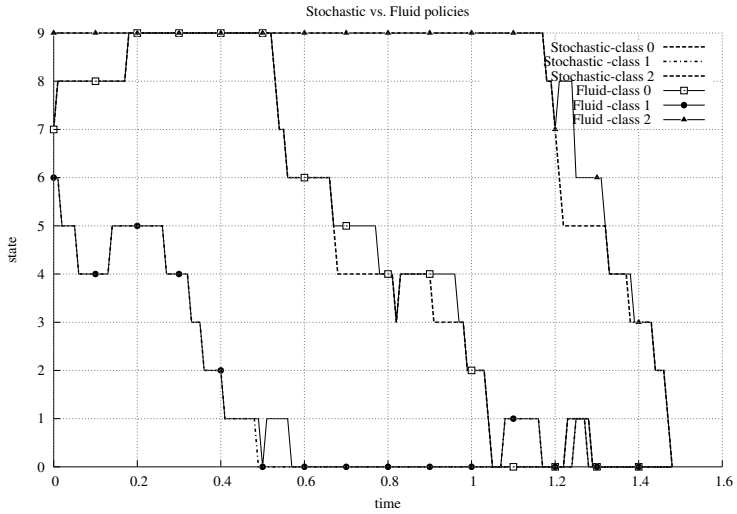


FIGURE 4.6 – Exemple de trajectoire obtenue sous la politique stochastique et la politique fluide (calcul analytique (haut) et Bocop (bas)) pour $x_0(0)$=7, $x_1(0)$=6 et $x_1(0)$=9.

Comme prévu, la politique fluide coïncide avec la règle $c\,\mu$, c'est-à-dire que la priorité est donnée à la classe 1 qui a l'indice $c_i\mu_i$ le plus élevé ($c_1\,\mu_1 = 6 > 5 = c_0\,\mu_0 > 4 = c_2\,\mu_2$). Notons que la politique stochastique optimale coïncide elle aussi avec la loi $c\,\mu$; en effet la priorité est donnée à la classe 1.

Deuxième scénario

Comme deuxième scénario, on considère à nouveau trois classes. La valeur de κ (resp. γ) est toujours 1 (resp. 3). On considère les paramètres suivants : ρ_0=0.5, ρ_1=0.6, ρ_2=0.4, $\mu_0 c_0$=4, $\mu_1 c_1$=2 et $\mu_2 c_2$=8.

La figure 4.7 présente le débit optimal fluide alloué à chaque classe pour un état initial $\boldsymbol{x}(0) = (11, 12, 13)$ obtenues sous la politique fluide et la politique stochastique.

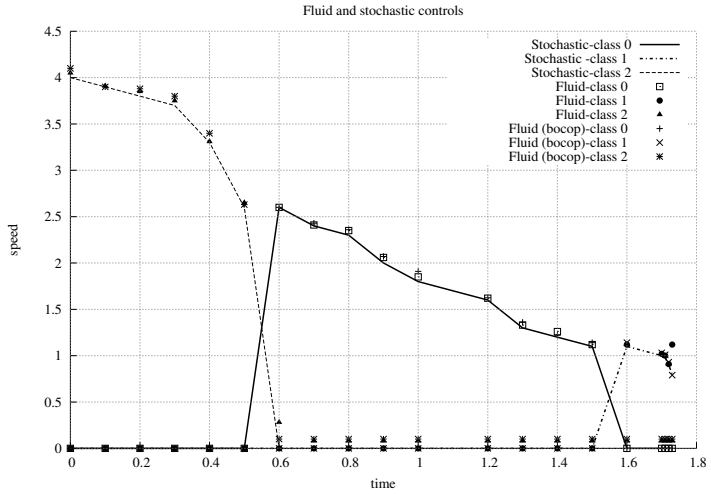


FIGURE 4.7 – Vitesse optimale fluide vs. Vitesse optimale stochastique pour le scénario 2 pour $x_0(0)$=11, $x_1(0)$=12 et $x_1(0)$=13.

Les figures 4.8 et 4.9 montrent des exemples de trajectoires des classes 1 et 2 sous la politique optimale fluide et la politique optimale stochastique pour un état initial $\boldsymbol{x}(0) = (11, 12, 13)$ et $\boldsymbol{x}(0) = (9, 6, 5)$, respectivement. De même, on observe que les trajectoires obtenues sous les deux politiques sont assez proches et que les deux politiques coïncident avec la loi $c\mu$. En effet, la priorité est donnée à la classe 2 ayant l'indice $c_i\mu_i$ le plus élevé ($c_2\,\mu_2 = 8 > 4 = c_0\,\mu_0 > 2 = c_1\,\mu_1$).

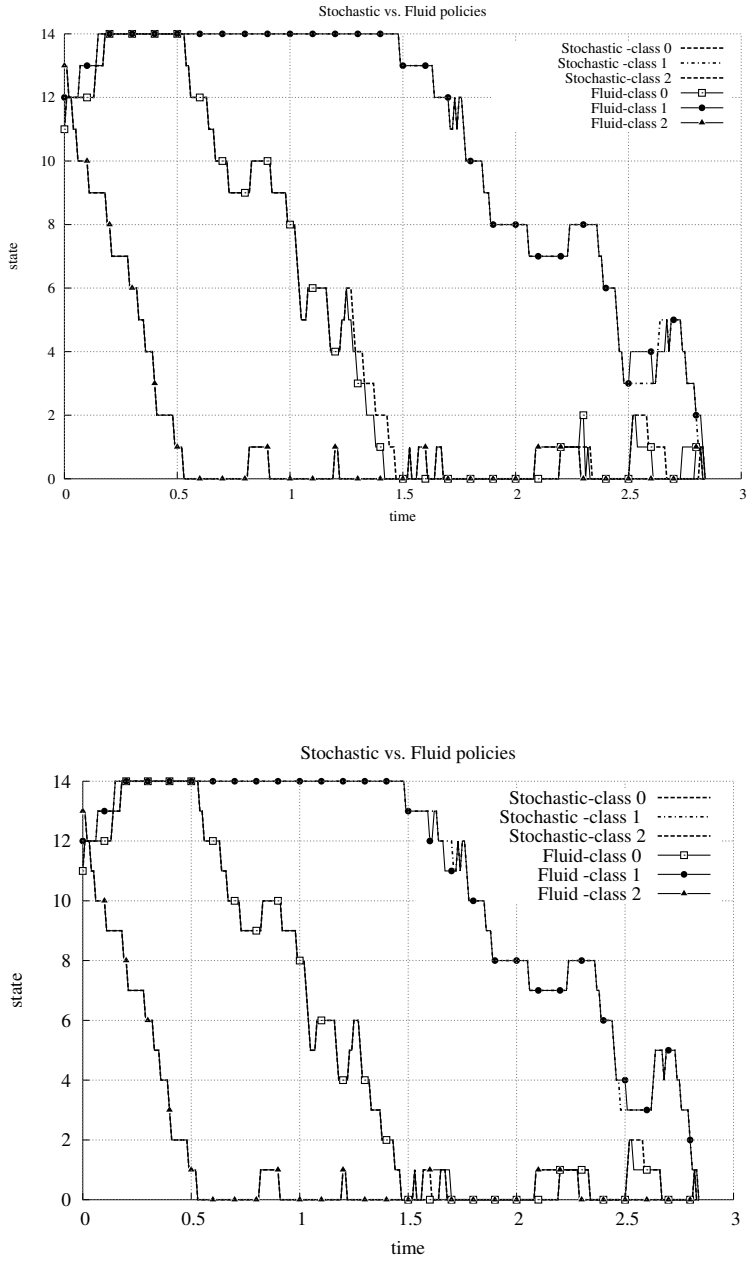


FIGURE 4.8 – Exemple de trajectoire obtenue sous la politique stochastique et la politique fluide (calcul analytique (haut) et Bocop (bas)) pour $x_0(0)$=11, $x_1(0)$=12 et $x_2(0) = 13$.

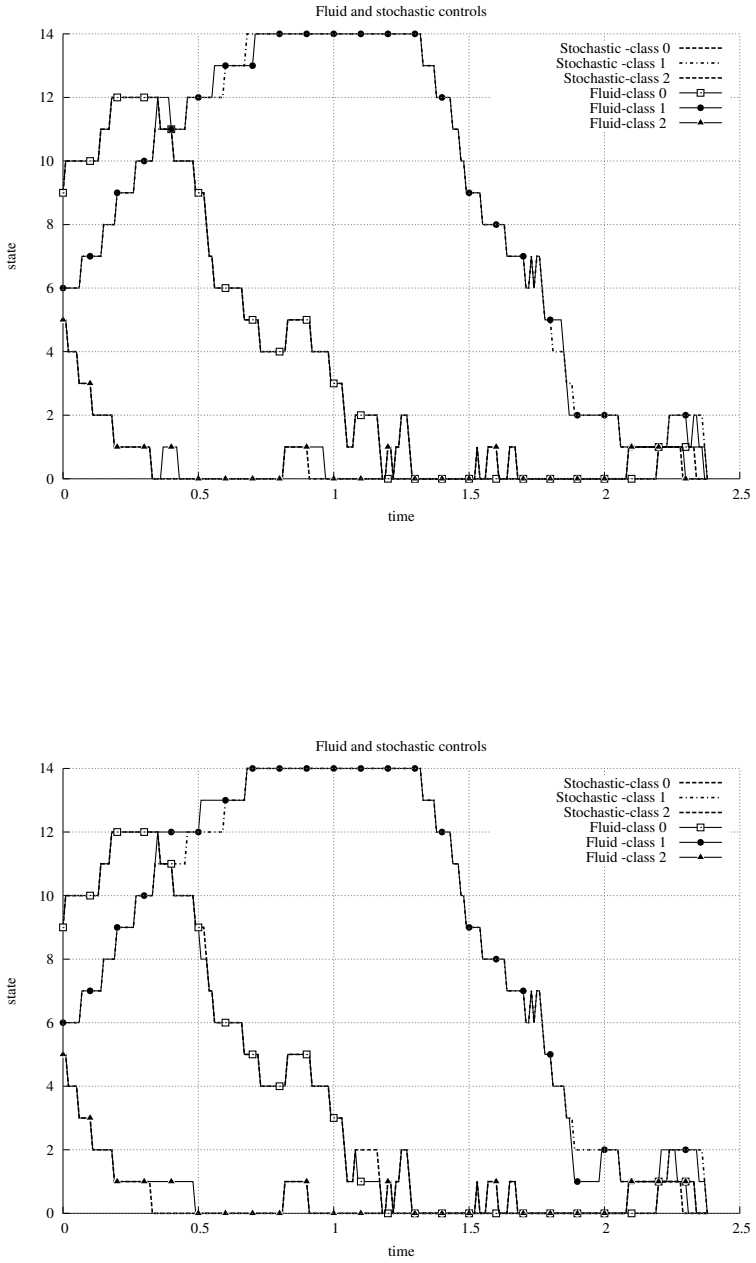


FIGURE 4.9 – Exemple de trajectoire obtenue sous la politique stochastique et la politique fluide (calcul analytique (haut) et Bocop (bas)) pour $x_0(0)$=9, $x_1(0)$=6 et $x_2(0) = 5$.

4.4 Conclusion

Dans ce chapitre, on a étudié le compromis performance/énergie dans les réseaux à partage de bande passante dont les nœuds peuvent réguler leur vitesse en fonction de la charge du système. L'objectif est de caractériser la politique optimale permettant de vider le système à partir d'un état de congestion. On a proposé une heuristique basée sur la version fluide du modèle stochastique et la méthode "*discrete-review*". On l'a validée numériquement sur un exemple de réseau linéaire et on a montré que pour le cas d'un seul lien la politique optimale suit la règle $c\mu$. Jusqu'à présent, nous nous sommes intéressés à l'évaluation et à l'optimisation des performances des réseaux à partage de bande de passante. Dans le prochain chapitre, on s'intéresse aux plateformes de Cloud Computing, et on cherche à évaluer les performances des applications distribuées exécutées dans ces environnements virtualisés.

5

Temps de réponse moyen des applications dans les plateformes de cloud computing

Résumé

L'objectif principal des travaux présentés dans ce chapitre est de montrer qu'il est possible de prédire correctement le temps de réponse moyen des applications exécutées sur une plateforme de cloud computing avec des modèles mathématiques extrêmement simples. On s'intéresse au modèle SaaS (pour *Software as a Service*), et l'on suppose que chaque application est exécutée dans une machine virtuelle VM (pour *Virtual Machine*) séparée. Nous prenons en compte le comportement spécifique de chaque type d'applications (interactive, CPU-intensive ou applications permanente) et proposons des approximations du temps moyen de réponse pour chacun de ces types d'application. Ces approximations sont ensuite validées grâce à des expérimentations réelles.

Sommaire

5.1 Introduction

Le Cloud Computing fournit des services ou des applications informatiques en ligne, accessibles partout, à tout moment, et depuis n'importe quel terminal (smartphone, ordinateur portable, tablette). Plus précisément, le Cloud Computing permet de partager les ressources (une infrastructure, une solution applicative ou encore une plateforme) mises à disposition par un fournisseur d'offres Cloud avec tout utilisateur qui en fait la demande via un simple site internet. Le cloud computing permet ainsi d'éliminer le coût de l'achat et de l'entretien des infrastructures informatiques, et permet aux utilisateurs de se concentrer sur ce que le service leur fournit plutôt que sur la façon dont le service est mis en œuvre ou hébergé. Les services offerts peuvent être classés en trois catégories, suivant le modèle utilisé :

— **Le modèle IaaS (Infrastructure as a Service)** : dans ce modèle, les services offerts permettent l'accès à des ressources informatiques dans un environnement virtualisé à travers Internet. Les ressources fournies correspondent à du matériel informatique virtualisé ou, en d'autres termes, à une infrastructure informatique.
— **Le modèle Paas (Platform as a Service)** : ce modèle fournit la plateforme et l'environnement informatique nécessaires aux développeurs d'applications pour mettre au point leurs propres services et applications sur Internet. Les services PaaS sont hébergés dans le Cloud et les utilisateurs y accèdent simplement, par leur navigateur web.
— **Le modèle SaaS (Software as a Service)** : ce modèle permet aux clients d'avoir accès à des applications logicielles sur Internet. Ces applications sont hébergées dans le Cloud et peuvent être utilisées via n'importe quel appareil disposant d'une connexion à Internet. Chaque application s'exécute sur une machine virtuelle séparée. Les hyperviseurs sont en charge du partage des ressources physiques entre les multiples machines virtuelles s'exécutant sur le même nœud physique.

Dans ce chapitre, nous nous intéressons particulièrement à la modélisation du partage des ressources physiques (CPU) (pour *Central Processing Unit*) entre les différents types d'applications dans le cadre du modèle SaaS. On suppose que chaque instance d'application est exécutée dans une machine virtuelle. La virtualisation joue un rôle capital dans les plateformes de Cloud Computing. En effet, la virtualisation a l'avantage de faire tourner plusieurs systèmes d'exploitation différents sur un même nœud physique (PC, Serveur,...).

Toutefois, les machines virtuelles tournant sur la même machine hôte doivent partager les ressources physiques disponibles. Cela peut éventuellement provoquer une

dégradation imprévisible des performances lors des pics de demande. Les fournisseurs de services ont besoin d'outils permettant d'estimer la quantité de ressources nécessaire pour répondre aux attentes des utilisateurs en termes de performance. C'est ce qui a motivé les travaux présentés dans ce chapitre.

5.2 État de l'art

Peu de travaux se sont intéressés à l'évaluation de performance dans les plateformes de Cloud Computing. Dans [64], les auteurs proposent un modèle de file d'attente pour évaluer les performances et l'énergie consommée dans un environnement virtualisé. En dépit de sa simplicité, le modèle requiert de nombreux paramètres d'entrée. En pratique, ces informations ne sont souvent pas disponibles, ce qui rend son utilisation difficile.

Les auteurs de [69] se sont intéressés à la consolidation de serveur dans un environnement virtualisé. Ils montrent que des modèles de files d'attente peuvent prédire efficacement les principales métriques de performance. La théorie de file d'attente a aussi été considérée dans [20] pour définir des modèles de performance pour les environnements virtualisés sous Xen Vms. Des modèles similaires ont égalemment été proposés dans [20, 69].

Dans [11], les auteurs analysent le compromis entre performance et consommation énergétique dans les environnements virtualisés. Il suppose dans leur étude un nombre fixe de machine virtuelle exécutant des serveurs web. Des modèles similaires ont été proposé dans [12] et [49].

Contrairement aux travaux cités ci-dessus, on s'intéresse dans ce chapitre à l'estimation du temps moyen de réponse des différents types d'application s'exécutant dans un environnement virtualisé. Notre contribution ici est de prendre en compte le comportement spécifique de chaque type d'applications (interactive, CPU-intensive et permanente). En utilisant la théorie de file d'attente, on propose des approximations simples du temps moyen de réponse des applications. Les expérimentations réalisées montrent le bon niveau de précision des approximations proposées.

5.3 Description du modèle

On considère un ensemble d'utilisateurs soumettant des requêtes aux serveurs d'un data center (cf. figure 5.1). Nous supposerons dans la suite que la population des utilisateurs est infinie. Plus précisément, la population des utilisateurs est suffisamment élevée pour considérer que le taux d'arrivée des requêtes est indépendant du nombre d'utilisateurs actifs. Notons que dans le cas contraire (cloud privé par exemple) le système peut être analysé à l'aide de la théorie des réseaux fermés de files d'attente [87].

On distingue trois types d'applications :

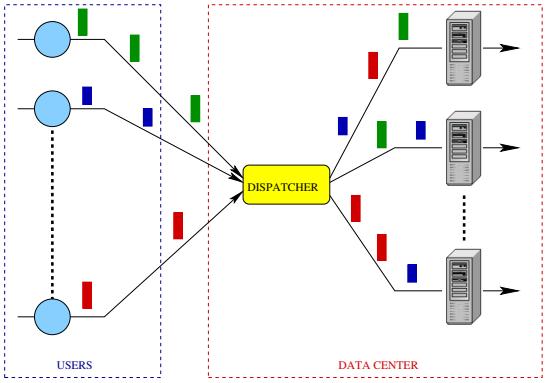


FIGURE 5.1 – Ensembles d'utilisateurs partageant les ressources d'un Data-center.

— *Application interactive* : Les applications appartenant à cette catégorie requièrent des ressources pour des durées finies qui peuvent être séparées par des périodes d'inactivité. Prenons l'exemple d'un utilisateur souhaitant éditer un texte. L'utilisateur va tout d'abord instancier un éditeur de texte. Pendant l'édition du texte, l'utilisateur peut aussi prendre un peu de temps pour réfléchir (temps de réflexion). Durant ces périodes de réflexions qu'on désigne ici par période d'inactivité, l'application requiert très peu de ressources et peut être ainsi considérée comme inactive.
— *Application CPU-intensive* : On désigne par application CPU-Intensive, toute application non interactive qui requiert des ressources de calcul pour une durée continue et finie de temps. L'exemple typique est celui du calcul scientifique et notamment des applications de simulation. Une fois instancié, un simulateur demeure actif jusqu'à la fin des simulations.
— *Application permanente* : Les applications de cette classe requièrent des ressources pour une durée infinie de temps et correspondent aux applications tel que les serveurs Web ou bien les serveurs de base de données qui s'exécutent sans interruption sur des périodes de temps extrêmement longues par rapport aux autres applications. En effet, une fois que la machine virtuelle hébergeant un serveur web est instanciée, elle demeure toujours active, soit en attente des requêtes ou bien entrain de les traiter. Même si la machine virtuelle pour cette classe est toujours instanciée, ses besoins en ressources peuvent varier en fonction de la demande. Par exemple, un serveur web requiert des ressources uniquement lorsqu'il est en train d'exécuter des requêtes. En dehors de ces périodes d'activité, les besoins en ressources du serveur sont négligeables.

Dans la suite, on notera $\mathcal{B}$ l'ensemble des applications CPU-intensive, $\mathcal{I}$ l'ensemble des applications interactives, et $\mathcal{P}$ l'ensemble des applications permanentes. On désigne

par $\mathcal{R} = \mathcal{B} \cup \mathcal{I} \cup \mathcal{P}$ l'ensemble des applications auxquelles les utilisateurs peuvent accéder. On suppose que les applications sont numérotées de 1 à R.

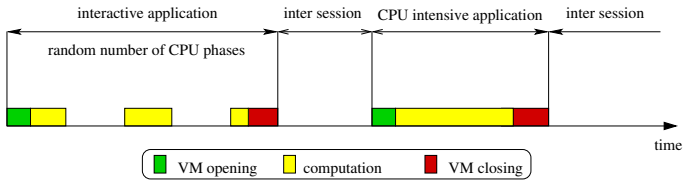


FIGURE 5.2 – Activité d'un utilisateur au cours du temps.

On supposera, par la suite, que les sessions sont initiées par les utilisateurs arrivent dans le système selon un processus de poisson d'intensité λ ; cette hypothèse est valable dès que le trafic de chaque utilisateur est relativement faible par rapport au trafic total. Une fois soumise, la requête est reçue par un dispatcher qui l'acheminera vers l'un des N serveurs de la plateforme selon une politique de routage probabiliste (*Bernouilli routing*) de type Round-Robin [65]. Par conséquent, les sessions arrivent sur chacun des serveurs selon un processus de poisson indépendant d'intensité λ/N. On peut par conséquent analyser chaque serveur de manière isolée. Dans ce qui suit, on considèrera donc un seul serveur en isolation, disons le serveur i.

Lorsqu'une application va être exécutée, le serveur lance tout d'abord une machine virtuelle. Soit $1/\mu_0(i)$ le temps moyen nécessaire pour ouvrir une VM dans le serveur i. Une fois que la machine virtuelle est prête, l'application commence son exécution dans la VM. Une nouvelle session utilise l'application r avec la probabilité γ_r. Chaque cycle d'exécution commence par une période de calcul. Soit $1/\mu_r(i)$ la durée moyenne de cette période pour l'application r. Les nombres de périodes de calcul durant une session sont des variables aléatoires indépendantes, de moyenne $1/p_r$. Par définition, $p_r = 1$ si r est une application CPU intensive, i.e., $r \in \mathcal{B}$.

On suppose que le nombre de phases de calcul pour une application interactive $r \in \mathcal{I}$ est distribué selon une variable aléatoire géométrique et on note $1/\beta_r$ la durée moyenne d'une période d'inactivité de l'application r. Une fois que l'application est terminée, la machine virtuelle est fermée et les ressources utilisées par cette machine virtuelle sont libérées. On désigne par $1/\mu_{R+1}(i)$ le temps moyen nécessaire pour la fermeture d'une VM dans le serveur i.

Dans la suite, on supposera que toutes les variables aléatoires sont indépendantes.

5.4 Évaluation de performance

Le modèle défini dans la section 5.3 est très général et se révèle assez complexe à analyser. On se restreint dans la suite à l'étude de deux cas particuliers. Au para-

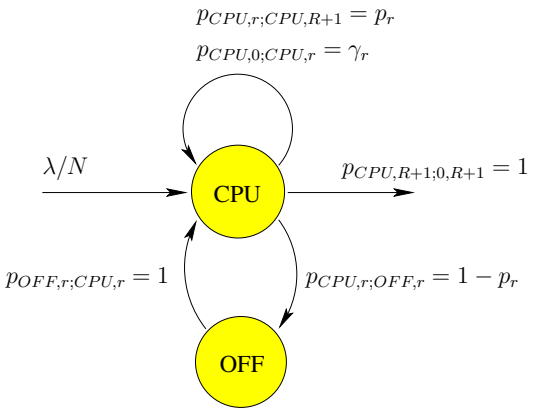


FIGURE 5.3 – Réseau de files d'attente ouvert où $p_{s,c;s',c'}$ désigne la probabilité de transition des clients de la classe c dans le noeud s à la classe c' dans le nœud s'.

graphe 5.4.1, on montre qu'en présence d'applications interactives et CPU-intensives, des expressions explicites simples de performance peuvent être obtenues en utilisant des résultats classiques de la théorie des files d'attente. Au paragraphe 5.4.2, on s'intéresse au scénario dans lequel une seule machine virtuelle exécutant une application permanente est lancée sur un hôte physique sur lequel elle est en concurrence avec d'autres machines virtuelles exécutant des applications CPU-intensives. Bien que ce cas se révèle beaucoup plus complexe que le cas précédent, on obtient là aussi des expressions simples et explicites des performances [18] .

5.4.1 Présence des applications CPU-intensives et interactives

En la présence d'applications CPU-intensives et interactives, on observe que le système est équivalent au réseau de files d'attente multiclasse décrit sur la figure 5.1.

Dans ce réseau, les clients présents dans les files d'attente correspondent aux applications en cours. On distingue $R + 2$ classes de clients :

— La classe 0 représente les VMs en cours d'ouverture ; tous les clients arrivant dans le système appartiennent à cette classe.

— La classe r tel que $1 \leq r \leq R$ correspond aux VMs exécutant l'application r.

— La classe $R + 1$ correspond aux VMs en cours de fermeture.

Chaque nœud dans le réseau est associé à un état possible d'une application en cours d'exécution. Par exemple, les clients présents dans le nœud CPU correspondent aux VMs en cours d'exécution (c'est-à-dire en cours d'ouverture, exécutant une application ou bien en cours de fermeture) tandis que les clients dans la phase OFF correspondent aux VMs exécutant des applications interactives qui sont en période d'inactivité. Sous l'hypothèse que le serveur partage la capacité CPU équitablement entre les machines

virtuelles, la discipline de service adéquate est alors la discipline PS. On suppose aussi que le nœud OFF est une file d'attente avec un nombre infini de serveurs et introduit un délai aléatoire de moyenne $1/\beta_r$ pour l'application interactive r (durée moyenne d'inactivité).

On notera dans la suite $\rho_{cpu}(i)$ et ρ_{off} le trafic offert dans les deux nœuds du système. Notons que le taux d'arrivée des clients appartenant à la classe 0 et à la classe $(R+1)$ au nœud *cpu* est λ/N, tandis que celui de la classe r est $\lambda\gamma_r/(N\,p_r)$ pour tout $r \in \mathcal{R}$. Le taux d'arrivée de la classe $r \neq 0, R+1$ au noeud OFF est $\lambda\gamma_r(1-p_r)/(N\,p_r)$, et 0 pour les deux autres classes. On obtient $\rho_{cpu}(i) = \sum_{r=0}^{R+1} \rho_{cpu}^r(i)$, où

$$\begin{aligned}
\rho_{cpu}^0(i) &= \frac{\lambda}{N}\,\frac{1}{\mu_0(i)} \\
\rho_{cpu}^r(i) &= \frac{\lambda}{N}\,\frac{\gamma_r}{p_r\mu_r(i)} \\
\rho_{cpu}^{R+1}(i) &= \frac{\lambda}{N}\,\frac{1}{\mu_{R+1}(i)},
\end{aligned}$$

et $\rho_{off} = \sum_{r=1}^{R} \rho_{off}^r$ avec $\rho_{off}^r = \lambda\gamma_r(1-p_r)/(N\,p_r\,\beta_r)$.

5.4.1.1 Distribution de probabilité

Soit n_s^r le nombre de sessions ouvertes de l'application r qui sont dans la phase $s \in \{cpu, off\}$. On définit les vecteurs suivants : $\mathbf{n}_{cpu} = \left(n_{cpu}^r\right)_{r=0,\ldots,R+1}$ et $\mathbf{n}_{off} = \left(n_{off}^r\right)_{r=1,\ldots,R}$. Soit $\pi(\mathbf{n})$ la probabilité stationnaire que le système soit dans l'état $\mathbf{n} = (\mathbf{n}_{cpu}, \mathbf{n}_{off})$. On note N_s (resp. N_s^r) le nombre total de sessions (resp. de sessions de l'application r) dans la phase s. Ce système est équivalent à un réseau BCMP [16]. On en déduit les résultats suivants en appliquant le théorème BCMP.

Proposition 5. *La distribution stationaire π existe si et seulement si $\rho_{cpu}(i) < 1$, et a la forme produit*

$$\pi(\mathbf{n}_{cpu}, \mathbf{n}_{off}) = \pi_{cpu}(\mathbf{n}_{cpu})\,\pi_{off}(\mathbf{n}_{off})$$

où

$$\begin{aligned}
\pi_{cpu}(\mathbf{n}_{cpu}) &= (1-\rho_{cpu}(i))\;|\mathbf{n}_{cpu}|!\;\prod_{r=0}^{R+1}\frac{(\rho_{cpu}^r(i))^{n_{cpu}^r}}{n_{cpu}^r!}, \\
\pi_{off}(\mathbf{n}_{off}) &= e^{-\rho_{off}}\;\prod_{r=1}^{R}\frac{(\rho_{off}^r)^{n_{off}^r}}{n_{off}^r!},
\end{aligned}$$

avec la notation $|\mathbf{n}_{cpu}| = \sum_{r=0}^{R+1} n_{cpu}^r$. *De plus,* $\Pr[N_{cpu} = k] = (1-\rho_{cpu}(i))\;(\rho_{cpu}(i))^k$

et $\Pr[N_{off} = k] = \frac{(\rho_{off})^k}{k!} e^{-\rho_{off}}$.

Soulignons que ces résultats, comme tous ceux présentés ci-dessous, sont insensibles aux caractéristiques détaillées des applications. En d'autres termes, ils ne dépendent des distributions aléatoires utilisées pour la durée des phases de calcul et des périodes d'inactivité qu'à travers leur moyenne.

5.4.1.2 Métriques de performance

On s'intéresse tout d'abord à l'estimation du nombre moyen de sessions. La durée moyenne d'exécution est ensuite déduite par la loi de Little. Soit D (resp. D^r) la durée moyenne d'exécution (resp. d'un application r). On a le résultat suivant :

Proposition 6. *Le nombre moyen de sessions dans la phase CPU est donnée par*

$$E[N_{cpu}] = \frac{\rho_{cpu}(i)}{1 - \rho_{cpu}(i)} \tag{5.1}$$

On a de plus $E[N_{off}] = \rho_{off}$. *Le nombre moyen de sessions d'une application* r *en cours d'exécution est* $E[N^r_{cpu}] = \frac{\rho_{cpu}(i)^r}{\rho_{cpu}(i)} E[N_{cpu}]$ *pour tout* $r \in \mathcal{R}$.

Avec la loi de Little, on en déduit le temps moyen de réponse des applications : $D = N \frac{E[N_{cpu}]}{\lambda}$. *De même, le temps moyen d'exécution de l'application* r *est* $D^r = N \frac{E[N^r_{cpu}]}{\lambda \gamma_r}$.

Remarque 4. *Pour simplifier, on a supposé que la probabilité qu'une nouvelle requête soit acheminée vers le serveur* i *est* $p_i = \frac{1}{N}$, $\forall i = 1, \ldots, N$. *Cette hypothèse est naturelle dans le cas de serveurs homogènes. Dans le cas contraire (les serveurs ont des vitesses différentes), il peut être intéressant de considérer d'autres stratégies d'équilibrage de charge. Tant que le routage restera probabiliste, les formules ci-dessus resteront valides : il suffit juste de remplacer* $1/N$ *par la nouvelle valeur de* p_i *dans l'expression de* $\rho^r_{cpu}(i)$ *et* ρ^r_{off}.

5.4.2 Présence d'applications permanentes et CPU-intensives

Dans ce paragraphe, on s'intéresse au cas où des applications CPU-intensives et permanentes coexistent sur le même nœud physique. Contrairement au cas précédent, il s'avère difficile de trouver des expressions des métriques de performance sans prendre des hypothèses simplificatrices.

Pour simplifier, on considère une seule VM exécutant une application permanente, disons un serveur web. Cette machine virtuelle est en concurrence sur le même nœud physique avec plusieurs VMs exécutant des applications CPU-intensives. On suppose aussi qu'il y a un nombre maximum M de machines virtuelles qui peuvent être exécutées

sur la même machine physique (le nombre maximal de processeurs virtuels par système mono-cœur est 8, $M = 8$).

Dans la suite, les VMs exécutant une application CPU-intensive seront appelées jobs de classe 1, tandis que les requêtes HTTP soumises au serveur web seront appelées jobs de classe 2.

Les jobs appartenant à la classe 1 arrivent selon un processus de poisson d'intensité λ_1 et ont une durée moyenne égale à $1/\mu_1$ exponentiellement distribuée (la durée inclue le temps d'ouverture et de fermeture de la VM). Les jobs sont soumis au serveur web selon un processus de poisson d'intensité λ_2 ; leur temps de service suit une loi exponentielle de moyenne $1/\mu_2$. L'intensité de trafic pour chaque classe i est donnée par $\rho_i = \lambda_i/\mu_i$. On note (n_1, n_2) l'état du système, où n_i désigne le nombre de jobs de classe i en cours d'exécution. En supposant que les VMs se partagent la CPU d'une manière équitable et donc selon la discipline PS, le taux de service de la classe 1 dans l'état (n_1, n_2) est

$$\phi_1(n_1, n_2) = \frac{n_1}{n_1 + \mathbb{1}_{\{n_2>0\}}},$$

alors que le taux de service agrégé des jobs de classe 2 est $\phi_2(n_1, n_2) = 1 - \phi_1(n_1, n_2)$. On observe que la dynamique de l'état du système est celle d'un processus markovien dont les taux de transition sont

$$q(\boldsymbol{n}, \boldsymbol{n}') = \begin{cases} \lambda_i & \text{si } \boldsymbol{n}' = \boldsymbol{n} + \mathbf{e}_i,\ i = 1, 2 \\ \mu_i \phi_i(\boldsymbol{n}) & \text{si } \boldsymbol{n}' = \boldsymbol{n} - \mathbf{e}_i,\ i = 1, 2 \end{cases}$$

Notons que la distribution stationnaire peut être calculée en résolvant numériquement cette chaine de Markov. Cependant, pour éviter un temps de calcul élevé et avoir des expressions explicites, nous proposons d'analyser le système sous une hypothèse de quasi-stationnarité (QS). C'est la même approche que celle considérée dans le chapitre 3. Nous supposons ici que le taux d'arrivée des jobs de classe 1 est assez petit par rapport à celui des jobs de classe 2 ($\lambda_1 \ll \lambda_2$) pour que le nombre de jobs de classe 2 atteigne un régime permanent avant que le nombre de jobs de classe 1 n'ait évolué.

Sous cette hypothèse, on peut déduire la distribution stationnaire $\pi(n_2|n_1)$ du nombre de jobs de classe 2 en présence de n_1 jobs de classe 1. On peut aussi calculer indépendamment la distribution stationnaire du processus stochastique $n1(t)$ en supposant que quand il y a n_1 jobs de classe 1, ils sont servis à la vitesse moyenne de $\tilde{\phi}_1(n_1) = \sum_{n_2} \phi_1(n_1, n_2)\pi(n_2|n_1)$. Le résultat est formellement énoncé dans la Proposition 7 ci-dessous.

Proposition 7. *Si $\rho_2 < \frac{1}{1+M}$, la distribution stationnaire du nombre de jobs de classe 2 existe et est donnée par*

$$\pi(n_2|n_1) = (1 - (1 + n_1)\rho_2)\ (1 + n_1)^{n_2}\, \rho_2^{n_2}, \quad n_2 = 0, 1, \ldots \tag{5.2}$$

La probabilité stationnaire d'avoir $n_1 \in \{0, 1, \ldots, M\}$ *jobs de classe* 1 *en cours d'exécution s'écrit*

$$\pi_1(n_1) = \begin{cases} \frac{1}{1+M} & \text{if } \nu = 1, \\ \frac{1-\nu}{1-\nu^{1+M}}\ \nu^{n_1} & \text{if } \nu \neq 1, \end{cases} \tag{5.3}$$

où $\nu = \rho_1/(1-\rho_2)$.

À partir de la Proposition 7, nous obtenons le résultat suivant.

Corollaire 3. *Le nombre moyen de jobs de classe* 1 *est donné par*

$$\mathbb{E}(N_1) = \begin{cases} \frac{M}{2} & \text{if } \nu = 1, \\ \frac{\nu}{1-\nu} - (1+M)\frac{\nu^{1+M}}{1-\nu^{1+M}} & \text{if } \nu \neq 1, \end{cases} \tag{5.4}$$

alors que le nombre moyen de jobs de classe 2 *est*

$$\mathbb{E}(N_2) = \sum_{n_1=0}^{M} \pi_1(n_1)\, \frac{\rho_2(1+n_1)}{1-\rho_2(1+n_1)}. \tag{5.5}$$

On en déduit évidemment le temps moyen d'exécution des applications par application directe de la loi Little.

5.4.2.1 Validité des approximations

On compare maintenant les résultats obtenus par l'approche QS avec ceux obtenus par la résolution numérique exacte de la chaine de Markov associée $(n_1(t), n_2(t))$. On note α le ratio $\frac{\lambda_1}{\lambda_2}$ et β le ratio $\frac{\mu_1}{\mu_2}$. On supposera ici que $M = 10$. On considère trois scénarii, chaque scénario étant défini par le produit $\alpha\beta$. La figure 5.4 illustre l'évolution de l'erreur relative de l'approximation en fonction du produit $\alpha\beta$.

On observe que les résultats obtenus avec l'approche quasi-stationnaire sont très proches de ceux obtenus en résolvant numériquement la chaîne de Markov. Notons aussi que l'erreur relative diminue quand le produit $\alpha\beta$ est assez petit, c'est-à-dire lorsque l'hypothèse de quasi-stationnarité est bien vérifiée.

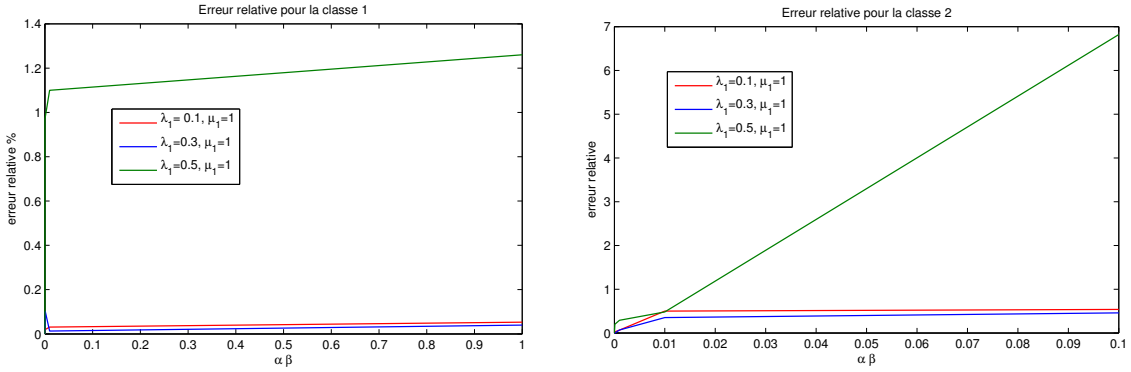


FIGURE 5.4 – Erreur relative pour les flots de classe 1 et 2.

5.5 Environnement d'expérimentation

Cette section décrit les expérimentations réalisées pour valider les approximations proposées aux paragraphes précédents. Plus précisément, ces expérimentations consistent à lancer en parallèle plusieurs applications interactives, CPU-intensives ou permanentes sur une machine hôte et à en mesurer les durées d'exécution. L'environnement d'expérimentation est décrit ci-dessous.

5.5.1 Machine hôte

L'environnement virtualisé dans lequel les applications ont été exécutés a été installé sur un serveur avec un processeur à quatre cœurs (Intel Xeon 2,3 GHz), 4 Go de RAM, un disque dur d'une capacité de 500 Go et un système d'exploitation Ubuntu 10.04.4 LTS. Pour la virtualisation, nous avons choisie VirtualBox [4] et nous avons installé 8 VMs sur 6 ordinateurs hôte, fonctionnant sous la même version d'Ubuntu. La mémoire allouée pour chaque machine virtuelle est de 512 MB (mémoire minimale requise par un système d'exploitation).

5.5.2 Émulation d'un processus de poisson

Pour émuler le processus de poisson d'arrivée des sessions, nous avons utilisé un programme Matlab. Plus précisément, le programme génère les instants d'arrivée $T_0, T_1, \ldots$ des sessions selon la formule récursive suivante :

$$T_{k+1} = T_k - (1/\lambda) * (\log(u)), k = 0, 1, \ldots \tag{5.6}$$

où $u \in [0, 1]$ est une variable aléatoire uniforme et $T_0 = 0$.

Les instants auxquels s'exécutent les applications servent ensuite de paramètres pour l'utilitaire cron [1] [2]. Lorsque le script d'exécution d'une application est lancée, une nouvelle VM s'ouvre, exécute l'application en question et se ferme à la fin de l'exécution.

5.5.3 Scilab : application CPU-intensive

Nous avons choisi un programme scilab comme exemple d'application CPU-intensive. Scilab est un logiciel Open source de calcul numérique fournissant un puissant environnement de développement pour les applications scientifiques et contenant des centaines de fonctions mathématiques et graphiques 2D et 3D, ainsi qu'un environnement de programmation pour le traitement du signal, l'optimisation, les statistiques et la simulation. Notre programme scilab résout un problème d'optimisation mathématique en utilisant la programmation dynamique (cf. [63]).

5.5.4 OpenOffice : application interactive

Nous avons choisi l'application Spreadsheet d'openOffice [3] comme exemple d'application interactive. Ce logiciel nous permettra de nous connecter depuis un ordinateur distant à un serveur qui exécute le traitement de texte d'une façon interactive et simple. On utilise un programme JAVA qui utilisent des composants UNO du kit de développement du logiciel Apache OpenOffice [2]. Ce programme ouvre une feuille de calcul, insère des données et trace ensuite quelques graphiques 3D. Afin d'émuler l'interactivité d'un utilisateur, le programme alterne entre des périodes ON, correspondant à la saisie des données et des périodes OFF, lorsque l'application est inactive. Le nombre de cycles ON-OFF suit une distribution géométrique de moyenne $\frac{1}{p} = 125$. La durée des périodes ON et OFF sont exponentiellement distribuées de moyennes 800 ms et 200 ms, respectivement. La durée moyenne d'un cycle de ON-OFF est donc de 1 seconde, et la durée moyenne d'une exécution est de 125 secondes.

Quand une nouvelle application interactive commence, un script est lancé pour ouvrir une VM, exécuter ensuite le programme puis fermer la VM à la fin de l'exécution. L'un des paramètres d'entrée du programme Java est le nom d'un fichier, choisie au hasard parmi 25 dossiers que nous avons générés. Chaque fichier donne le nombre de cycles ON-OFF, ainsi que la durée de chaque phase ON ou OFF. Le tableau 5.5 donne un exemple d'un tel fichier dans le cas de 4 cycles ON-OFF.

1. Un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiées à l'avance.

2. Le Kit de développement logiciel Apache OpenOffice est un add-on pour la suite bureautique de ZO. Il fournit les outils et la documentation pour la programmation des API IFP et de créer propres extensions (composants ONU) pour Apache OpenOffice nécessaires.

Phase	Temps (ms)
ON	153
OFF	36
ON	1986
OFF	43
ON	1612
OFF	561
ON	751
OFF	183

FIGURE 5.5 – Exemple de fichier à 4 cycles ON-OFF.

5.5.5 Web Service : application permanente

Pour simuler le serveur web, on utilise un simple programme client/serveur communiquant via les sockets. On utilise en outre le multi-threading pour traiter les requêtes des clients simultanément. Les requêtes sont générées selon un processus de Poisson. Lorsqu'un serveur reçoit une requête, un thread est généré pour traiter la requête en question. Le thread effectue un calcul itératif. Un paramètre d'entré permet de contrôler le nombre d'itérations et donc la durée d'exécution. Dans la suite, on choisira ce paramètre de manière à ce que la durée d'exécution des requêtes soit exponentiellement distribuée avec une moyenne de 5 secondes.

5.6 Résultats

Ce paragraphe présente un ensemble de résultats expérimentaux. On compare les résultats des expérimentations avec les prédictions des modèles présentés au paragraphe 5.5. Pour ce faire, nous avons mesuré le temps moyen de réponse des applications et nous l'avons comparé avec celui obtenu avec le modèle analytique.

5.6.1 Estimation des paramètres

Pour calculer le temps moyen de réponse théorique, on doit, tout d'abord, déterminer les valeurs du temps moyen , $1/\mu_i$. Pour ce faire, on propose de les déterminer empiriquement. Plus précisément, on lance une machine virtuelle en isolation et on exécute un script permettant d'ouvrir une VM, exécuter l'application en question (Scilab ou OpenOffice) et fermer ensuite la VM. On répète cette procédure plusieurs fois, puis on calcule le temps moyen de service total empirique de chaque type d'application correspondant à l'ouverture de la VM, l'exécution de l'application en question et de la fermeture de la VM.

Les résultats de ces expérimentations sont les suivants :

— *Machine virtuelle* : le temps moyen pour ouvrir et fermer une VM, i.e., $\frac{1}{\mu_0}$ et $\frac{1}{\mu_{R+1}}$, sont respectivement de 65 et 11 secondes.
— *Applications CPU-intensives* : le temps d'exécution du programme scilab est 122 secondes. En prenant en considération les valeurs obtenues pour l'ouverture et la fermeture d'une VM, on en déduit que le temps moyen de réponse d'une application de type CPU-intensive exécutée dans une VM est de 198 secondes.
— *Applications interactives* : Comme déjà mentionné au paragraphe 5.5.4, le nombre de cycles ON-OFF est $\frac{1}{p} = 125$ ms, la durée moyenne d'une phase ON est $\frac{1}{\mu_r} = 800$ ms, et la durée d'une phase OFF est $\frac{1}{\beta_r} = 200$ ms. Ainsi, le temps d'exécution moyen d'une application interactive est au total de 125 secondes. En prenant en considération le temps nécessaire pour ouvrir et fermer une VM, on obtient un temps de réponse moyen d'une application interactive de 201 secondes.
— *Application permanente* : comme déjà précisé au paragraphe 5.5.5, les requêtes ont un temps de service exponentiellement distribué de moyenne 5 secondes. Notons que pour varier la charge, il suffit de varier le taux d'arrivée des requêtes.

Les valeurs expérimentales des paramètres du modèle sont indiquées dans le tableau 5.1.

Paramètre	Valeur
$1/\mu_0$	65 sec
$1/\mu_{R+1}$	11 sec
$1/\mu_r$, $r \in \mathcal{B}$	122 sec
$1/\mu_r$, $r \in \mathcal{I}$	0.8 sec
$1/\beta_r$, $r \in \mathcal{I}$	0.2 sec
$1/p_r$, $r \in \mathcal{I}$	125 cycles

TABLE 5.1 – Paramètre d'exécution des applications dans un VM

Après avoir déterminé ces paramètres, on procède à la phase d'expérimentation et de validation du modèle proposé. Les expérimentations ainsi que les résultats sont détaillés dans les paragraphes ci-dessous.

5.6.2 Applications en isolation

Ce paragraphe décrit le cas où un seul type d'application (CPU-intensive, interactive, ou permanente) existe dans le système, i.e., $R = 1$. Contrairement aux expérimentations présentées au paragraphe 5.6.1, on suppose ici que plusieurs instances de la même application peuvent être exécutées simultanément. Les instants d'exécution ont été générés selon un processus de Poisson. On exécute 100 instances de l'application grâce à l'utilitaire cron. Chaque instance ouvre une VM, exécute l'application, et puis ferme la VM.

Charge	CPU Intensive		Interactive		Permanente	
	Temps moyen	Erreur relative	Temps moyen	Erreur relative	Temps moyen	Erreur relative
0.2	247 sec	1.2%	244 sec	1.21%	6.25 sec	3.2%
0.5	396 sec	2.2%	376 sec	2.58%	10.0 sec	3.9%

TABLE 5.2 – Temps moyen de réponse des applications CPU-intensive, Interactive et permanente.

Charge CPU-intensive	Charge Interact.	Charge totale	Temps moyen	Erreur relative
0.1	0.1	0.2	246 sec	1.99%
0.2	0.3	0.5	383 sec	3.03%

TABLE 5.3 – Erreur relative sur le temps moyen des applications interactives et CPU-intensives.

On compare les résultats pour les deux scénarii suivants : $\rho_{cpu} = 0.2$ et $\rho_{cpu} = 0.5$. Le temps moyen de réponse théorique ainsi que l'erreur relative pour les trois types d'application sont indiqués dans le tableau 5.2. On observe que le modèle prédit assez bien les résultats mesurés, l'erreur relative étant inférieure à 4%.

5.6.3 Applications CPU-intensives et Interactives

On s'intéresse dans cette section au scénario où des applications interactives et CPU-intensives coexistent dans le système. On génère selon le processus de Poisson et ordonnance les premières 100 exécutions pour chaque application (Scilab et OpenOffice). On compare les résultats théoriques et expérimentaux pour les deux scénarii suivants : (i) la charge due à chaque type d'application est 0.1 ; et (ii) la charge due aux applications CPU-intensives est 0.2 et celle due aux applications interactives est 0.3. Dans le tableau 5.3, on indique le temps moyen de réponse théorique ainsi que l'erreur relative pour les deux scénarii. On observe que le modèle prédit assez bien les résultats mesurés avec une erreur relative inférieure à 4%.

5.6.4 Applications CPU-intensives et permanentes

Finalement, on considère le cas où une seule application permanente coexiste avec plusieurs applications CPU-intensives dans le système. Le taux d'arrivé des requêtes pour le serveur web est généré de manière à ce que l'approche quasi-stationnaire soit satisfaite.

On considère deux scénarii : (i) la charge due à l'application permanente est égale 0.1, tandis que celle due aux applications CPU-intensives est égale à 0.4, et (ii) la

Charge CPU-intensive	Charge permanent	Temps CPU-intensive	Erreur relative	Temps permanent	Erreur relative
0.4	0.1	386.1 sec	8.8%	12.5 sec	6.4%
0.6	0.1	551.5 sec	5.53%	26.01 sec	4.9%

TABLE 5.4 – Erreur relative sur le temps moyen des applications permanentes et CPU-intensives.

charge due à l'application permanente est égale 0.2, alors que celle due aux applications CPU-intensives est 0.5.

Les résultats expérimentaux sont présentés dans le tableau 5.4. Les ressources sont alors utilisées à 20 % pour le premier scénario et à 70 % pour le deuxième scénario. Comme dans le cas précédent, on observe que le modèle prédit assez bien les valeurs mesurées avec une erreur relative inférieure à 8.8% pour les deux scénarii considérés.

Conclusion

Dans ce présent chapitre, on s'est intéressé à l'estimation du temps moyen de réponse des applications dans une plateforme de Cloud Computing fonctionnant sous le modèle SaaS. Les modèles de performance proposés ici sont applicables au cas où les applications sont exécutées dans une machine virtuelle elle même exécutée sur un hôte physique de la plateforme. Notre principale contribution est de prendre en compte le comportement spécifique des différentes applications (CPU-intensives, permanentes ou interactives) et de proposer pour chacune des expressions explicites simples du temps moyen de réponse. En dépit de leur simplicité, les approximations prédisent assez bien les valeurs mesurées dans un environnement réel.

6
Conclusion

Dans ce mémoire de thèse, nous avons étudié la performance des réseaux et des architectures distribuées en utilisant des techniques issues de la théorie des files d'attente.

Les chapitres 3 et 4 s'intéressent aux performances des réseaux à partage de bande passante. Les modèles étudiés dans ces chapitres sont des modèles de niveau flot, qui permettent d'évaluer les performances du réseau du point de vue des utilisateurs.

En utilisant ces modèles, on évalue tout d'abord au chapitre 3 les performances du trafic circulant dans les réseaux à partage de bande passante. On propose dans une première partie des approximations simples et explicites des principales métriques de performance des flots élastiques sous l'hypothèse que le partage de bande passante est conforme à l'équité équilibré. L'avantage clé de ce mode de partage est son insensibilité aux caractéristiques fines du trafic. Malheureusement, nous n'avons pas réussi à obtenir des bornes sur l'erreur des approximations proposées, ni même à montrer qu'elles peuvent être exactes sous certaines conditions asymptotiques. La validation de ces approximations s'est donc faite pour l'instant sur la base de simulations et d'observations numériques. À l'avenir, il serait évidemment très intéressant de démontrer des bornes sur l'erreur relative des approximations proposées.

On étudie égalemment au chapitre 3 le cas où des flots élastiques sont en compétition avec des flots de streaming pour le partage de la bande passante. On suppose dans cette partie que les flots de streaming sont toujours prioritaires et qu'un mécanisme de contrôle d'admission est utilisé pour en limiter le nombre. Nous avons obtenu des expressions exactes du taux de blocage des flots de streaming et des approximations sur le débit moyen des flots élastiques en utilisant une hypothèse de quasi-stationnarité. Pour

les réseaux de grande dimension, les approximations proposées ne sont pas utilisables du fait de l'explosion combinatoire de l'espace d'états. Il serait donc utile dans des travaux futurs de proposer une solution permettant le calcul du débit moyen des flots élastiques sans nécessiter le calcul de la distribution marginale du nombre de flots de streaming.

Dans le chapitre 4, on s'est intéressé au compromis entre énergie et performance dans les réseaux à partage de bande passante. L'objectif de ce chapitre est de caractériser la politique d'allocation de débit permettant de ramener le système dans un état acceptable à partir d'un état initial de congestion, tout en optimisant un compromis entre performance des flots et consommation énergétique. Nous avons formulé le problème comme un problème de contrôle optimal stochastique. La politique stochastique optimale étant difficile à déterminer numériquement et analytiquement, nous avons proposé d'étudier le modèle fluide associé. Nous avons montré comment la politique fluide peut être utilisée pour contrôler le réseau et vérifié sur un exemple simple que la politique fluide ainsi utilisée fournit une très bonne approximation de la politique stochastique optimale. Pour un seul lien, nous avons montré que la politique fluide correspond à la régle $c\mu$ et caractérisé complètement la vitesse optimale du lien. Deux pistes sont intéressantes pour des travaux futurs. Tout d'abord, nous avons l'intention d'étudier avec la même approche l'allocation de débit dans un réseau linéaire. Ensuite, nous proposons également de considérer le cas où des utilisateurs impatients sont présents dans le système et peuvent le quitter si leur performance n'est pas satisfaisante.

Finalement, nous nous sommes intéressé au chapitre 5 à l'évaluation de performance des applications exécutées dans une plateforme de Cloud fonctionnant sur le modèle SaaS. En supposant un partage équitable de la capacité CPU entre les machines virtuelles exécutées sur un même hôte physique, nous avons proposé des approximations simples et explicites du temps moyen de réponse des applications. Les modèles proposés prennent en compte le comportement spécifique des différentes applications (interactives, de calcul intensif ou permanente). Les expérimentations et mesures réelles montrent que nos modèles mathématiques prédisent correctement le temps moyen de réponse des applications. Les travaux futurs incluent notamment la généralisation des résultats obtenus au cas où les trois types d'applications coexistent en même temps sur un même nœud physique, ainsi que l'étude des performances lorsque les différentes applications ont des priorités relatives.

Bibliographie

[1] Bocop - the optimal control solver, http ://bocop.saclay.inria.fr.

[2] Cron, www.linuxmanpages.com/man5/crontab.5.php.

[3] Openoffice, www.openoffice.org/.

[4] Virtualbox, www.virtualbox.org/.

[5] Visual networking index : Global mobile data traffic forecast update, 2012-2017. *tech.rep Cisco.*, February 2013.

[6] S. Aalto, U. Ayesta, S. Borst, V. Misra, and R. Núñez Queija. Beyond processor sharing. *SIGMETRICS Perform. Eval. Rev.*, 34(4) :36–43, March 2007.

[7] S. Aalto, U. Ayesta, and R. Righter. On the gittins index in the m/g/1 queue, 2009.

[8] E. Altman, K. Avrachenkov, and C. Barakat. Tcp network calculus : The case of large delay-bandwidth product. In *Proceedings of IEEE Infocom*, 2002.

[9] E. Altman, K. Avrachenkov, and A. Garnaev. Closed form solutions for water-filling problems in optimization and game frameworks. In *Proceedings of the 2Nd International Conference on Performance Evaluation Methodologies and Tools*, ValueTools '07, pages 5 :1–5 :8, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[10] L.L.H. Andrew, A. Wierman, and A. Tang. Optimal speed scaling under arbitrary power functions. *SIGMETRICS Perform. Eval. Rev.*, 37(2) :39–41, October 2009.

[11] J. Anselmi and I.M. Verloop. Energy-aware capacity scaling in virtualized environments with performance guarantees. *Perform. Eval.*, 68(11) :1207–1221, November 2011.

[12] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. Energy-aware autonomic resource allocation in multitier virtualized environments. *IEEE Trans. on Services Computing*, 5(1), January-march 2012.

[13] F. Avram, D. Bertsimas, and M. Ricard. An optimal control approach to optimization of multiclass queueing networks. *Proceedings of Workshop on Queueing Networks*, 1994.

[14] N. Bambos and J. Walrand. Scheduling and stability aspects of a general class of parallel processing systems. *Adv. in Appl. Probab.*, 25 :176–202, 1993.

[15] N. Bansal and M. Harchol-Balter. Analysis of srpt scheduling : Investigating unfairness. In *Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '01, pages 279–290, New York, NY, USA, 2001. ACM.

[16] F. Baskett, K. Mani Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2) :248–260, April 1975.

[17] H. Ben Cheikh, O. Brun, and J.M. Garcia. Simple approximations of performance metrics for a link integrating streaming and elastic traffic. In *Proceedings of International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks PEMWN 2014*, 2014.

[18] H. Ben Cheikh, J. Doncel, O. Brun, and B. Prabhu. Predicting response times of applications in virtualized environments. *In IEEE NCCA 2014*, 2014.

[19] N. Benameur, S. Ben Fredj, F. Delcoigne, S. Oueslati-boulahia, J. W. Roberts, France Telecom R, and Issy Les Moulineaux. Integrated admission control for streaming and elastic traffic. In *Quality of Future Internet Services, Lecture Notes in Computer Science 2156*. Springer.

[20] F. Benevenuto, C. Fernandes, M. Santos, V. Almeida, J. Almeida, G. Janakiraman, and J. Santos. Performance models for virtualized applications. In G. Min, B. Martino, L.T. Yang, M. Guo, and G. Rünger, editors, *ISPA 2006 Workshops*, volume 4331 of *Lecture Notes in Computer Science*, pages 427–439. Springer Berlin Heidelberg, 2006.

[21] Y. Bernet, S. Blake, D. Grossman, and A. Smith. An informal management model for diffserv routers, 2002.

[22] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1987.

[23] D.P. Bertsekas. *Dynamic Programming and Optimal Control I*. Athena Scientific, Belmont, Massachusetts, 1995.

[24] D. Bertsimas and G. Van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Oper. Res.*, 39 :601–615, 1991.

[25] U. Black. *IP Routing Protocols : RIP, OSPF, BGP, PNNI and Cisco Routing Protocols*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[26] T. Bonald. Throughput performance in networks with linear capacity contraints. In *Proceedings of the 40th Annual Conference on Information Sciences and Systems*, pages 644–649, March 2006.

[27] T. Bonald, L. Massoulié, A. Proutière, and J. T. Virtamo. A queueing analysis of max-min fairness, proportional fairness and balanced fairness. *Queueing Syst.*, 53(1-2) :65–84, 2006.

[28] T. Bonald, A. Penttinen, and J. T. Virtamo. On light and heavy traffic approximations of balanced fairness. In *Proceedings of ACM SIGMETRICS/Performance*, pages 109–120, 2006.

[29] T. Bonald and A. Proutière. Insensitive bandwidth sharing in data networks. *Queueing Syst.*, 44(1) :69–100, 2003.

[30] T. Bonald and A. Proutière. On performance bounds for the integration of elastic and adaptive streaming flows. In *Proceedings of ACM SIGMETRICS '04/Performance '04*, pages 235–245, New York, NY, USA, 2004.

[31] T. Bonald and J. T. Virtamo. Calculating the flow level performance of balanced fairness in tree networks. *Perform. Eval.*, 58(1) :1–14, 2004.

[32] T. Bonald and J.T Virtamo. A recursive formula for multirate systems with elastic traffic. *IEEE Communications Letters*, 9(8) :753–755, Aug 2005.

[33] O. Brun, H. Ben Cheikh, and B. Prabhu. Optimal speed scaling for multiclass fluid queues. *Research report, LAAS-CNRS*, December. 2014.

[34] O. Brun, A. Al Sheikh, and J.M. Garcia. Flow-level modelling of tcp traffic using gps queueing networks. In *International Teletraffic Congress (ITC21)*, pages 1–8. IEEE, 2009.

[35] T. Bu and D. Towsley. Fixed point approximations for tcp behavior in an aqm network. In *Proceedings of ACM SIGMETRICS '01*, pages 216–225, New York, NY, USA, 2001.

[36] B. Butscher and W. Heinze. A file transfer protocol and implementation. *SIGCOMM Comput. Commun. Rev.*, 9(3) :2–12, July 1979.

[37] C. Buyukkoc, P. Varaiya, and J.Walrand. The μc-rule revisited. *Adv. Appl.Prob*, 17 :237–238, 1985.

[38] J. Cao, W. S. Cleveland, and D. X. Sun. Bandwidth estimation for best-effort internet traffic. *Statistical Science*, 19 :518–543, 2004.

[39] B. Chachuat. Nonlinear and dynamic optimization - from theory to practice. Technical report, Ecole Polytechnique Federale de Lausanne, 2006.

[40] J.W. Cohen. The multiple phase service network with generalized processor sharing. *Acta Informatica*, 12(3) :245–284, 1979.

[41] S. Deering and R. Hinden. Internet protocol, version 6 (ipv6) specification, 1998.

[42] F. Delcoigne, A. Proutière, and G. Régnié. Modeling integration of streaming and data traffic. *Perform. Eval.*, 55(3-4) :185–209, February 2004.

[43] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *SIGCOMM Comput. Commun. Rev.*, 19(4) :1–12, August 1989.

[44] D K H Tan F P Kelly, A K Maulloo. Rate control for communication networks : shadow prices, proportional fairness and stability, 1998.

[45] G. Fayolle, I. Mitrani, and R. Iasnogorodski. Sharing a processor among many job classes. *J. ACM*, 27(3) :519–532, July 1980.

[46] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1, 1999.

[47] S. Floyd and V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Trans. Netw.*, 9(4) :392–403, August 2001.

[48] S. Ben Fred, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts. Statistical bandwidth sharing : A study of congestion at flow level. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 111–122, New York, NY, USA, 2001. ACM.

[49] A. Gandhi, V. Gupta, M. Harchol-Balter, and M.A. Kozuch. Optimality analysis of energy-performance trade-off for server farm management. *Perform. Eval.*, 67(11) :1155–1171, November 2010.

[50] N. Gans and G. Van Ryzin. Optimal control of a muticlass, flexible queueing system. *Oper. Res.*, 45 :677–693, 1997.

[51] J-P. Haddad. *Bounds and Approximations for Stochastic Fluid Networks.* PhD thesis, University of Waterloo, Ontario, Canada, 2011.

[52] M. Handley, S. Floyd, J. Padhye, and J. Widmer. Tcp friendly rate control (tfrc) : Protocol specification, 2003.

[53] J.M. Harrisson. The bigstep approach to flow management in stochastic processing networks. *Stochastic Networks : Theory and Applications (F. Kelly, S. Zachary and I. Ziedins eds))*, pages 57–90, 1996.

[54] R.F. Hartl, S.P. Sethi, and R.G. Vickson. A survey of the maximum principles for optimal control problems with state constraints. *SIAM Review*, 37(2) :181–218, June 1995.

[55] J.R. Jackson. Networks of waiting lines. *Operations Research*, 5 :518–521, 1957.

[56] C. Joe-wong, S. Sen, T. Lan, and M. Chiang. Multi-resource allocation : Fairness-efficiency tradeoffs in a unifying framework. In *in Proc. IEEE INFOCOM*, 2012.

[57] J.Rawls. The theory of justice. *BelknapPress*, 1971.

[58] F. P. Kelly. *Reversibility and stochastic networks.* Wiley series in probability and mathematical statistics. Wiley, New York, Chichester, 1979.

[59] P. Key, L. Massoulié, A. Bain, and F. Kelly. Fair internet traffic integration : network flow models and analysis. *Annals of Telecommunications*, 59 :1338–1352, 2004.

[60] L. Kleinrock. Time-shared systems : A theoretical treatment. *J. ACM*, 14(2) :242–261, April 1967.

[61] L. Kleinrock. *Queueing Systems*, volume I : Theory. Wiley Interscience, 1975.

[62] S. Kunniyur and R. Srikant. End-to-end congestion control schemes : Utility functions, random losses and ecn marks. In *In Proceedings of IEEE Infocom*, pages 1323–1332, 2000.

[63] M. De Lara. Available online : http ://www.ing-mat.udec.cl/, 2005.

[64] R. Lent. Evaluating the performance and power consumption of systems with virtual machines. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 778–783, 2011.

[65] T. Li, D. Baumberger, and S. Hahn. Efficient and scalable multiprocessor fair scheduling using distributed weighted round-robin. In *Proceedings of the 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '09, pages 65–74, New York, NY, USA, 2009. ACM.

[66] C. Maglaras. Discrete-review policies for scheduling stochastic networks : trajectory tracking and fluid-scale asymptotic optimality. *The Annals of Applied Probability*, 10(3) :897–929, 2000.

[67] Laurent Massoulié. Structural properties of proportional fairness : Stability and insensitivity. *Annals of Applied Probability*, page 2007.

[68] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the "tcp" congestion avoidance algorithm. *SIGCOMM Comput. Commun. Rev.*, 27(3) :67–82, July 1997.

[69] D. A. Menascé. Virtualization : Concepts, applications, and performance modeling. In *The Computer Measurement Groups'2005 International Conference*, Orlando, FL, USA, 2005.

[70] Sean P. Meyn. Feedback regulation for sequencing and routing in multiclass queueing networks. *SIAM J. Control and Optimization*, 2000.

[71] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM TRANS. ON NETWORKING*, 8(5) :556–567, 2000.

[72] K. Moore. Simple mail transfer protocol (smtp) service extension for delivery status notifications (dsns), 2003.

[73] P. Nain and D. Towsley. Optimal scheduling in a machine with stochastic varying processing rate. *IEEE Transactions on Automatic Control*, 39 :1853–1855, 1994.

[74] J. Nash. The bargaining problem. *Econometrica*, 18(2) :155–162, April 1950.

[75] The Network Simulator NS-2. `http://www.isi.edu/nsnam/ns/`.

[76] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput : a simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM '98*, SIGCOMM '98, pages 303–314, New York, NY, USA, 1998. ACM.

[77] J. Postel. User datagram protocol. RFC 768, Internet Engineering Task Force, August 1980.

[78] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168.

[79] Martin L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[80] R. Núñez Queija, H. Van Den Berg, and M. Mandjes. Performance evaluation of strategies for integration of elastic and stream traffic.

[81] R. Righter and J. G. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Probability in the Engineering and Informational Sciences*, pages 323–333, 1989.

[82] J. Roberts. Realizing quality of service guarantees in multiservice networks. in : Proceedings of ifip seminar pmccn. *Performance Evaluation*, T.97, 1998.

[83] J. Roberts, U. Mocci, and J. virtamo. Broadband network teletraffic. *Springer*, 1996.

[84] J. W. Roberts, L. Massoulié, France Telecom Cnet, and Issy moulineaux Cedex. Bandwidth sharing and admission control for elastic traffic. In *Telecommunication Systems*, pages 185–201, 1998.

[85] J.W. Roberts. A survey on statistical bandwidth sharing. *Computer Networks*, 45(3) :319 – 332, 2004.

[86] L.E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, pages 16 :687–690, 1968.

[87] R. Serfozo. *Introduction to stochastic networks*. Applications of mathematics. Springer, New York, 1999.

[88] B. Tan, L. Ying, and R. Srikan. Short-term fairness and long-term qos in the internet. *Performance Evaluation*, 67 :406–414, 2010.

[89] L. Tassiulas and S. Papavassiliou. Optimal anticipative scheduling with asynchronous transmission opportunities. *IEEE Trans. Automat. Control*, 40 :2052–2062, 1995.

[90] I.M Verloop and S.C. Borst. Heavy-traffic delay minimization in bandwidth-sharing networks. *In : Proc. IEEE Infocom*, 2007.

[91] R. J. Vetter. Atm concepts, architectures, and protocols. *Commun. ACM*, 38(2) :30–ff., February 1995.

[92] J. Wroclawski. The use of rsvp with ietf integrated services, 1997.