Bandwidth Sharing and Control in High-Speed Networks: Combining Packet- and Circuit-Switching Paradigms

Sébastien Soudan

January 15, 2010

To the happy few

Remerciements

Mes remerciements vont en premier lieu à mes parents Henri et Véronique, Séverine et Sylvain, Marisie et Dominique, aux peuplades du Bugey et des Bauges et tous ceux qui ont fait que la rédaction de cette thèse a été possible.

Merci à Pascale, pour m'avoir laissé beaucoup de liberté dans mes travaux de thèse, tout en me fournissant un très bon soutien, à Paulo pour ses cours de traitement du signal qui m'ont redonnés l'envie de jouer avec des outils mathématiques.

Je remercie également les collègues sans qui ces dernières années n'auraient, évidemment, pas été pareil : Ludovic, Romaric, Dinil, Marcelo, Patrick, Augustin, Philippe, Fabienne, Guilherme, Hugo, Suleyman, Andrei, Binbin, Magi, ainsi que l'équipe RESO et le LIP. Finalement, je remercie mes co-auteurs et collaborateurs, en particulier : Julien Vanegue, Tomohiro Kudoh, Dominique Verchere et Eitan Altman.

Merci aux rapporteurs et membres du jury.

Contents

Abstract

As an offspring of ARPANET, the Internet has been designed as a packet network with robustness and survivability as first objectives. At that time, packet switching has been preferred over circuit switching. To cope with congestion due to an increasing popularity and the scarcity of resources, the Internet has then evolved with a fair-sharing objective implemented using the end-to-end principle. The global allocation performed by this system can be interpreted (*cf.* [Chiang et al., 2007]) as a distributed optimization procedure aiming to maximize the individual utilities summed in an α-fair manner—α depends on the TCP variant.

Concurrently, the usage—video-conferencing, video-on-demand, large-file sharing, remote backup, data-intensive applications, e-science—and expectations of users regarding the Internet have also evolved, making both users and providers unhappy: the former because of the quality of service obtained, and the latter because their profit margins are impacted. Indeed, their revenues are not directly linked to the provided resources.

Variety of requirements and associated utility functions together with renewal of circuit through optical circuit switching and development of the automation of circuit provisioning in transport networks with unified control plane leads to consider other allocations mechanisms. They are suited for these new utilities such as variable provisioning of access link over time or deadline-constrained transfers and combination of them.

The most promising is based on services where users can express their needs through request submission. Then, by taking advantage of the packet switching paradigm and its relative fluidity, and providing the advantage of knowing in advance if the resource will be sufficient for the users, the service optimizes resource allocation under the constraints of both users and network operators. The existence of this service does not preclude classical best-effort approaches for the remainder of the traffic.

As an example, if a user wants a dedicated circuit for high-quality video conferencing, the request will contain different constraints specifying the requested QoS, such as a low delay—*e.g.,* less than 50 ms—and a constant bandwidth—*e.g.,* 960 Mbit/s for 4096×2048 px at a frame rate of 30.0 $pict/s$—for the duration of the video conference. Alternatively, users or data-intensive applications might want to have a deterministic completion of a 40 GB transfer within a time window of 10 min, and get an allocation fulfilling the constraint. This allocation is a profile of bandwidth that, over time, delivers the requested volume. The rest of the time, the needed bandwidth is much less.

In short, the problem is for operators to have an interest in providing users with the expected QoS. My contributions are distributed in five different sub-problems:

(i) the evidence of the problem for bulk data transfers with the evaluation of transfer time predictability;
(ii) the algorithmics for the service that schedules requests of network utilization;
(iii) practical considerations on how to implement this service from the source application to the destination application, and its performance;
(iv) how a service provider can use a rented infrastructure to provide a transfer service;
(v) finally, is this approach justified, from the points of view of users and service providers?

Predictability of Transfer Completion Time Due to the dependence of the allocation to the number of participants, fair sharing shows a fundamental problem when it comes to predicting the allocation that one will get in capacity-constrained networks—although, even if the number of concurrent flows remains constant, it is still difficult. Using real experiments with different transport protocols, our first contribution shows that completion time under congestion depends on so many parameters that it is difficult—or impossible—for users to determine the completion time of their transfers [4, 5, 6, 8].

Explicit Request and Scheduling of Transfers To solve this, we propose a network-resource allocation service that allocates network bandwidth based on the users' utilities while taking care of the value of the network. The allocation mechanism proposed gives both satisfaction to users—seeking best-effort or guaranteed services—and flexibility to network operators, which can plan in advance the resource utilization of users with large needs [3]. The scheduler has been implemented [jBD09].

Data Plane and Conformance to Allocated Profile After the presentation of the bulk data transfer scheduling service on a static network, we present how bandwidth delivery can be articulated with the data plane to enable regular applications to efficiently use the allocated resources. For this purpose, we present the FLOC software [FLO09] and evaluate its performance regarding the original objective: being able to use this to enforce profiles of rates for flows [1, 9, 10, 18].

Dynamic Bandwidth Provisioning and Bit Mover Service Next, to benefit from dynamically reconfigurable networks, we introduce a tiered architecture: the clients or customers, the service provider, and the network provider. In this architecture, clients issue transfer requests to a bit-mover service that provisions the circuits of its dynamic infrastructure by issuing bandwidth requests to a network provider. We present the provisioning optimization problem as a linear program and simulations to highlight the benefit of mixing malleable requests with more constrained requests [2, 11, 12].

Extension of Routing Games to Flow over Time Finally, we propose an extension of routing games with convex cost functions to consider time, as a tool to quantify, study and propose solutions to overcome the inefficiency introduced by selfishness in Wardrop-like games. As a result, the service provider is shown to be able to improve the social cost of the allocation, compared to the social cost of the Wardrop equilibrium reached by selfish decisions. This makes the service provider both legitimate and sustainable [24].

Résumé

En tant que descendant du projet ARPANET, Internet a été conçu comme un réseau de commutation de paquets. Le principal objectif était alors la robustesse : le réseau devait continuer à être fonctionnel même en cas de perte d'une partie de ses éléments. Le paradigme de commutation par paquets s'avérait alors intéressant : les décisions sont prises pour chaque fragment d'information et non à l'établissement de la communication, comme cela est le cas dans le paradigme circuit.

Pour faire face à la congestion due à sa popularité grandissante et à la rareté des ressources, Internet a ensuite évolué et intégré le concept de partage équitable de la bande passante mis en oeuvre selon le principe de bout en bout. Selon ce principe, lorsque cela est possible, les opérations liées au contrôle des communications individuelles doivent être effectuées exclusivement par les machines d'extrémités. TCP s'inscrit strictement dans cette approche, aussi bien pour le contrôle des erreurs ou des pertes de paquets que pour la régulation du débit.

L'allocation globale qu'effectue ce système, très largement distribué, peut être interprétée[1] comme une procédure résolvant un problème d'optimisation qui maximise, sous contrainte, une combinaison des utilités individuelles. Les contraintes sont celles de capacités. La fonction de bien-être social utilisée pour agréger les utilités individuelles dépend de la variante de TCP utilisée, mais vise une certaine équité sous forme de somme pondérée de puissances de la bande passante allouée (α-*fairness*).

Parallèlement les utilisations et attentes des utilisateurs vis-à-vis d'Internet ont aussi évolué : Ce au niveau des caractéristiques de bas niveau, en passant du besoin de connectivité à des besoins de bande passante pour faire fonctionner des applications gourmandes, mais aussi qu'au niveau des besoins plus élaborés tels que la prédictabilité du temps de complétion d'un transfert afin de pouvoir enchaîner des opérations de différentes natures comme des transferts suivis par des phases de calcul sur des resources réservées en avance.

Parce que le modèle de qualité de service d'Internet repose sur le sur-dimensionnement du réseau de cœur, cette évolution des besoins se heurte à la fois au problème de la qualité de service qui n'est pas forcément atteinte ou adaptée, mais aussi au coût pour les opérateurs d'avoir une telle infrastructure pour laquelle, de plus, les ressources allouées ne sont pas directement liées aux recettes.

La diversité des besoins et les utilités individuelles associées, ainsi que le renouveau des approches circuit au travers de la commutation de circuits optiques et de l'automatisation de la configuration et de l'approvisionnement des réseaux de transport grâce aux plans de contrôle unifiés a conduit à étudier des mécanismes d'allocation différents de ceux en place dans Internet. Au lieu de laisser TCP déterminer le partage, ces mécanismes reposent sur les processus de mise en oeuvre des réseaux (routage, création de tunnel, reconfiguration) jusqu'alors utilisés par les opérateurs sur des agrégats de trafic. En les rendant plus dynamiques et plus fiables, l'automatisation permet de les rendre disponibles aux utilisateurs. Ce contrôle ne peut cependant pas être mis directement à disposition des utilisateurs et nécessite la mise en place d'intermédiaires assurant la planification et la correction de la configuration du réseau.

L'approche la plus prometteuse est basée sur des services qui permettent à l'utilisateur de spécifier ses besoins par la soumission de requêtes en avance à l'opérateur, qui effectue ensuite l'allocation de ressources et la mise en oeuvre de celles-ci pour le compte de l'utilisateur. En

[1]voir [Chiang et al., 2007] pour une synthèse

profitant de la relative fluidité des paradigmes de commutation par paquet, le service peut optimiser l'allocation des ressources sous les contraintes spécifiées d'une part par l'utilisateur et d'autre part par l'opérateur lui-même. Finalement, l'existence d'un tel service n'empêche pas celle du classique trafic *best-effort* qui a fait le succès d'Internet. Ainsi, si un utilisateur veut un circuit dédié pour de la diffusion de vidéo en haute définition, la requête spécifiera une borne sur le délai du chemin et sur la bande passante minimum. Similairement, un utilisateur voulant la garantie de transférer un certain volume de données entre deux dates, spécifiera le volume et ces dates à la soumission de la requête. Cependant, lors de l'allocation, l'opérateur pourra lui attribuer un profil de bande passante variable avec le temps à la condition qu'une fois celui-ci intégré entre les bornes temporelles, il donne le volume escompté. Finalement, un utilisateur n'ayant pas de contraintes particulières pourra simplement utiliser opportunément TCP et partager avec les autres flux *best-effort* la capacité restante.

Prédictabilité du temps de complétion de transferts massifs Au delà du problème fondamental du partage équitable dans les réseaux de capacité limitée qui font que l'allocation de bande passante qu'obtient un flux dépend du nombre de participants, nous montrons que les interactions entre flux ou entre couches rendent difficile l'estimation de la durée nécessaire pour terminer un transfert.

Grâce à des expériences effectuées sur un vrai réseau avec un nombre de flux fixé, nous montrons dans une première contribution que le temps de complétion dépend de tant de paramètres qu'il est difficile ou impossible pour un utilisateur de déterminer cette durée pour ses transferts. Ces expériences montrent en particulier qu'en présence de congestion, l'influence de la couche physique sur la couche transport a un impact sur les performances, et ce même dans les réseaux filaires commutés où le comportement de TCP change en fonction du choix du commutateur [4, 5, 6, 8].

Allocation de ressources basée sur une demande explicite et la planification Afin d'offrir aux utilisateurs des garanties de temps de complétion, nous proposons un service d'allocation de ressources qui alloue les ressources réseaux en fonction du besoin des utilisateurs tout en s'assurant de ne pas nuire à la valeur du réseau, qui repose en grande partie sur la disponibilité permanente de bande passante pour le trafic *best-effort*.

Le mécanisme proposé satisfait les deux types d'utilisateurs recherchant des garanties et recherchant de la connectivité, ainsi que l'opérateur qui peut planifier l'utilisation de son réseau en avance pour les transferts volumineux.

Ce service a été implanté [jBD09]. Cette approche a également été étendue au cas des transferts sur un *overlay*. Dans ce cas le *middleware* contrôlant celui-ci remplace l'opérateur et l'objectif des allocations est d'éviter autant que possible la congestion qui peut arriver du fait du trafic concurrent à l'*overlay* [3].

Articulation plan de données / plan de contrôle La planification des transferts telle que proposée dans la contribution précédente requiert que les émetteurs soient capables d'émettre leurs données en un flux qui se conforme à un profil de bande passante variable au cours du temps.

L'articulation du service et du plan de données est étudiée dans cette contribution afin de permettre la mise en oeuvre de la réservation de ressources pour une application et l'utilisation efficace de celles-ci. Une solution basée sur le contrôle du débit à la source a été proposée et implantée [FLO09]. Elle permet de respecter les allocations en contrôlant l'émission des paquets de sorte que l'application n'émette ni plus ni moins que prévu par l'ordonnanceur du service [1, 9, 10, 18].

Approvisionnement en bande passante pour fournir un service de transferts planifiés Alors que les infrastructures réseau considérées étaient statiques dans les contributions précédentes, nous proposons ensuite une architecture à plusieurs niveaux : clients, fournisseur de service, et opérateur réseau qui permet au fournisseur de service de manipuler une infrastructure réseau dynamique. Dans cette architecture, les clients soumettent les requêtes au service de transferts qui les ordonnance et approvisionne son infrastructure dynamiquement auprès de l'opérateur, en lui soumettant des requêtes de bande passante. Nous présentons le problème d'optimisation correspondant au choix de l'approvisionnement et de l'ordonnancement des transferts, en proposant une approche à base de programmes linéaires. Nous étudions numériquement son comportement lorsque des requêtes flexibles et non flexibles sont mélangées [2, 11, 12].

Extension des *routing games* aux transferts planifiés et aux flux au cours du temps Dans cette dernière contribution, dans le cas de coûts convexes, nous proposons une extension des *routing games*, de la notion d'équilibre de Wardrop, et de prix de l'anarchie en prenant le temps en compte.

Cet outil permet d'étudier et de quantifier l'inefficacité en terme de coût social introduite par le comportement égoïste des utilisateurs. Grâce à cette extension, en utilisant un modèle de coût simple qui prend en compte la préférence des utilisateurs pour un intervalle de temps plutôt qu'un autre et leur aversion à la congestion, nous montrons que sous certaines conditions de charge du réseau, par leur comportement égoïste, ceux-ci utilisent l'intervalle qu'ils préfèrent au détriment de la congestion ainsi créée et dont le coût sera supporté par tous.

Ainsi, l'existence d'un service réalisant l'ordonnancement permet dans ce cas de décider qui devra être sacrifié pour le bien de tous. Le coût total de cette allocation est alors inférieur à celui de l'allocation égoïste, ce qui rend l'existence de ce service légitime [24].

« *On ne connaît que les choses qu'on apprivoise.* »

—Le Petit Prince, *Antoine de Saint-Exupéry*

— One —
Introduction

The Internet is a Common Good The utilization of the Internet has become a *good*—actually, a service—in the economic meaning as it increases the utility of its users. Economics classifies goods in four categories depending on whether they are rival or nonrival, and *excludable* or non-excludable. Depending on the combination of these two characteristics, the goods have different properties. Their names are reported in Table 1.1.

	Excludable	Non-excludable
Rivalrous	Private goods	Common goods
Non-rivalrous	Club goods	Public goods

Table 1.1: Classification of goods

Network bandwidth is fundamentally *rivalrous*, since what is consumed by someone can not be by someone else. But as long as the network is over-provisioned—or not congested—, there is no rivalry. Everyone gets what they expect.

On the other side, when the network becomes congested, the cost of congestion is paid by everyone. This cost can be of different types, such as an increased queueing delay, more losses, less bandwidth than expected, or even a higher-level cost such as a deadline missed because of a longer completion time. There is rivalry in congested networks.

Unless networks are built as fat trees, there can always be congestion at a link aggregating traffic from different sources. In the Internet, usually, the last mile or local loop is the less provisioned and the more congested, since it is the most difficult and expensive part to upgrade.

When there is rivalry and one user cannot be excluded from the consumption, the good is referred to as a *common good* and has to be shared between the participants. The way the limited capacity of the network bandwidth will be shared has to be decided. But, as will be detailed later, it is not an easy task.

This model does fit the current Internet. Once someone has an access to the network, through an ISP for example, he can send data anywhere, and potentially create congestion on some of the links. Even if the destination can be random, this has to be mitigated because the exact path that will be taken by the data depends on the routing that has been decided by the network provider first, and then by the whole routing table resulting from the distributed routing protocol between domains. In some cases, congestion can be created by one user as at access links for example. There is rivalry in the network and it is a common good.

Non-rivalrous cases are not of much interest since the constraints on capacity are not tight. If the network can be congested and users or flows can be excluded, it is a *private good*. As any private good, as long as the law permits it, the rules for sharing are decided by the owner. For common goods, the rules can be decided by the owner or a regulatory agency targeting a specific objective.

Current Sharing Model For the Internet, different level of rules apply but the most important one is its transport protocol: TCP [Jacobson, 1988]. Its sharing mechanism has been designed by V. Jacobson in 1988 when the Internet was still mainly a research network. Before that, TCP had no congestion control mechanism and was only retransmitting lost packets. Because of this and the increase of the number of flows, in 1986, the network got congested and collapsed. Only 40 bit/s of throughput was observed on a 32 kbit/s backbone due to losses and retransmissions. This led to the addition of congestion control. The congestion control algorithm for flows using TCP relies on the detection of loss events to reduce the sending rate. The sharing achieved by these mechanisms maximizes an *alpha*-fair combination of the flow rates under the capacity constraints [Chiang et al., 2007]. This is based on several assumptions:

(i) the objective is to share the benefits, and not the costs;
(ii) the fairness is between flows and not users;
(iii) flows all have the same utility functions (concave and increasing).

This problem is known as *Network Utility Maximization* (NUM).

In 2009, while the Internet has reached 1.59 billion of users, the same rules still apply. This one-fits-all solution is claimed to not be the universal solution anymore, for multiple reasons. A variety of new usages have emerged: some use a large amount of bandwidth, some need low delays, or a low loss rate, some have a high value/utility (for their owners) to volume ratio. Considering all the flows of the same importance is not appropriate anymore. Especially since the number of flows is not limited—the Internet is a not an excludable good—and a single user can have multiple flows where one would suffice to get a bigger share *in virtue* of the principle of fair sharing at flow-rate level. This sharing does not offer any guarantee on the allocation along the time. New flows can come and reduce the share making it difficult to forecast the completion time of a transfer.

Based on these observations, differentiation mechanisms have been proposed to add quality of service to the Internet's sharing model. Their deployment have failed. New sharing mechanisms are needed but they have to convince both users and providers that they are a solution to their concerns.

The Tragedy of the Commons Then, what are the specificities of resource sharing in networks? In the presence of *negative externalities*—which are cost payed by all, such as congestion—common goods are known to face a dilemma called "the Tragedy of the Commons" after the title of a 1968 paper of Garrett Hardin published in Science [Hardin, 1968].

When different agents, each having their own interest, share common goods, they have an incentive to increase their own income even if the negative externality will increase, and possibly lead to the destruction of the common goods when the cost due to the negative externalities becomes higher than earned income. This is because agents have an incentive to become unfair and, once unfair, no incentive to become fair again even if this reduces the total benefit.

More generally, this inefficiency is known to appear when selfish users can choose their routes—as in the class of game called *routing games* [Nisan et al., 2007, Chap. 18]—, the server they will use, or their sending rates [Kameda and Altman, 2008], and this affects the cost that everyone will pay.

A typical example of this kind of situation is "Pigou's example" [Nisan et al., 2007, Chap. 17]. An infinite number of users have to route a total demand of 1 on two alternate paths. The cost of the first path is 1 while the cost of the second is x where x is the amount of traffic that is routed through this path. Since x is less than 1 on $[0, 1]$, every infinitesimal agent chooses the second path. The total cost of this Wardrop equilibrium is $1 \cdot 1 + 0 \cdot 1 = 1$ while the optimal allocation routes one half of the traffic on each path and has a total cost of $1/2 \cdot 1/2 + 1/2 \cdot 1 = 3/4$. Selfish users created congestion by selecting the route that is cheaper for them. The optimal solution needs coordination and potentially redistribution of the cost between the agents since some have to use the link with the cost of 1.

The tragedy of the commons focuses on the total cost of realizing the allocation, instead of the utility or aggregate utility, as NUM does. To summarize the tragedy of the commons, there is no incentive for users to reduce their consumption. This can apply at different levels: choice of congestion control algorithm, use of parallel flows, or simply by consuming/downloading more than the average and then creating more than its "share" of congestion.

In 1999, S. Floyd observed that at that time, incentive to be fair was mainly a social incentive and thus, besides promoting the use of TCP's congestion control, she proposed router-based mechanisms to identify flows that either are not "TCP-friendly" or simply use "a disproportionate share of the bandwidth", and punish them [Floyd and Fall, 1999]. The former are flows that do not use more bandwidth than a regular TCP flow while the latter can be regular TCP flows that, for example, grab more bandwidth because of their small round-trip times. Since this proposal was based on the control of flow behaviors, it also raised the concern of connections split into multiple connections—in order to get more. This is actually what is done now in peer-to-peer file sharing applications that exchange a lot of small chunks of files using different connections. Even though they use regular TCP, the continuously-present demand and the multiple flows lead this kind of application to use a large share of the bandwidth of the Internet. In 2006, a study of residential user-to-user traffic in Japan shows that traffic distribution among users is highly skewed: the top 4% use 75% of the total inbound traffic [Cho et al., 2006].

Solutions to Overcome the Inefficiency Economics has a solution to overcome the inefficiency introduced by selfishness [Varian, 1992, Chap. 24]. Pigouvian taxes are one example of them. Just like the carbon tax that makes polluters pay for emitting carbon dioxide and greenhouse gases, Pigouvian taxes add to the cost that users will face to deter them from being inefficient. If the taxes are properly chosen, this leads the new equilibrium to be the optimal setting of the problem with the original cost.

Another solution is obviously to transform common goods into private goods by making users/flows excludable and the allocation decided by an agent aware of the demand and the available resources. The new service model can propose different services that users will request for their flows based on their needs. This approach has been advocated in 1995 by S. Shenker [Shenker, 1995] but the current evolution of the technologies available in the transport network makes this model particularly well-suited.

New Technologies One the challenges faced by network operators due to the increase of consumed bandwidth is packet processing at every router. This is costly in terms of time, energy, and equipments.

At the same time, optical networks are becoming mature. With the circuit-switching paradigm, they present the advantage of being able to carry data from end-to-end without any processing, at the cost of the setup of the circuit. The technical evolution of the available protocols and tools to manage such optical networks make it possible to consider configuring paths on-demand in a reasonable amount of time: from seconds to hours depending on the checks performed.

At the expense of management and bookkeeping for the operators, and explicit requesting for applications, this kind of circuit can provide high bandwidth and low delay to applications, new valuable and planned services for operators, and less traffic in the best-effort service.

To summarize the previous paragraphs, the change of sharing approach is driven by three forces: new needs, foreseen bad trends, and technology.

New Sharing Approaches As for any disruptive solution, its success depends on its ability to convince network operators that it will benefit them and be adopted by their customers [Clark et al., 2005]. Customers must, in turn, find an interest in switching to the new solution and not ignore it or change provider.

For customers, the value of a given network depends on the number of reachable participants: the network effect captured by Metcalfe's law, which states that the value of a network with n participants is related to the number of potential connections of the network $n\,(n-1)/2$. This is probably over-estimated but definitively more than linear in n.

The other part of the value of the network depends on the congestion that customers will face and the real cost—time, money, energy—it implies. Another important point regarding the value that customers associate with a service is the development of "everything as a service" and combinations of services that require time synchronization: it is useless to get the processing service before input files have reached the site where data will be processed. Similarly, bandwidth for a high-quality video conference is required at the time of the conference, not later. Considering time in advance can help for some applications.

For the operator, the benefit is two-fold: first, its users will be satisfied by their utilization of the network, and they will not move to another provider or decide to switch to their own private infrastructure based on dark-fibers. Optical fibers are not that scarce a resource since a lot had been deployed by the dot-com bubble. Second benefit is that operators have an interest in getting a part of the traffic as in-advance requests that they can schedule according to capacity planing and planned utilization of the network.

The constraints we derive from the analysis above are:

(i) since existing applications already use TCP and it performs reasonably well when there is no congestion, TCP has to be kept;
(ii) current best-effort service is valuable and mandatory;
(iii) guaranteed services are valuable but explicit specification of users constraints can be required;
(iv) the requests submitted by users have to be flexible so the network provider can exploit this to reduce the cost of the allocation while serving user needs;
(v) support of in-advance reservation is required.

Since circuits can be used to carry packets, the proposal combines the benefit of packets that are suited for web browsing, e-mail, and any application requiring a large number of small flows to different hosts, with circuits that are suited for bandwidth-demanding applications or applications requiring guarantees, as it shifts the rejection risk from every packet to the setup—or reservation in the case of explicit in-advance reservation. From the operator's point of view, it also reduces the packet-processing costs by moving traffic to end-to-end optical network. It is then required to define how bandwidth will be shared, and its utilization controlled to fulfill the requirements previously mentioned.

Throughout the chapters, we try to answer the following questions:

Q1 How does the current approach behave regarding massive transfer completion time?

Q2 How can bandwidth-scheduling services help to provide guarantees on transfer completion time?

Q3 How do services fit in the economical environment?

Q4 How do they help regarding the value of the network?

Q5 What kind of request is suitable for a malleable network service?

Q6 How can they be allocated?

Q7 How can the allocations be enforced?

Q8 How can dynamic provisioning of the network infrastructure be used to provide a bandwidth-scheduling service?

Q9 What is the benefit of the service approach? Is it sustainable?

Q10 What are the alternative approaches to reduce the inefficiency introduced by the agents' selfishness?

The dissertation is organized as follow. The first two chapters present the state-of-the-art on the Internet model and packet networks and on optical networks organization. In Chapter 4, we explore experimentally the performance of the current model regarding the predictability of transfer completion time. Chapter 5 contains the proposal of the sharing model while Chapter 6 presents the algorithmic for the scheduling of large file transfers specified by volume and time constraints. Chapter 7 shows the architecture of the service, as well as the interaction with the different planes down to the data plane and the allocation-enforcement mechanisms. The performance of the transport protocol under these mechanisms is verified. Based on an optimization problem, an extension of the service to benefit from the dynamic provisioning offered by optical networks is shown in Chapter 8. Finally, Chapter 9 is devoted to the new tool we developed, "routing games over time", to study the problem of selfishness and the price of anarchy of allocations of malleable requests over the time. The conclusion is in Chapter 10.

— Two —

Internet Model and Packet Networks

Introduction

Telephone networks are more than 100 years old. The Internet has been designed more than 40 years ago. While the former is a circuit network, the later uses—as of now—packets to carry pieces of information.

This design choice made at the early stage of its existence is responsible for the Internet's success thanks to its simplicity and its support of communications with numerous peers. But, it also has heavy implications on the performance of applications built on top of this network: *Quality of Service* (QoS), reliability, security.

At a time where convergence of telephone network and Internet has already been pursued for years, and will probably results in the disappearance of classical circuit-switched telephone network, the study of initial motivations as well as history of Internet and packet networks is of importance to understand what the limits of this model are and how attempts to overcome them have been pursued.

The first section presents packet networks, the Internet's history and model, and proposals that have been made regarding QoS control. The next section deals with the *Transmission Control Protocol* (TCP) and the end-to-end principle that has driven the development of Internet protocols. Finally, the last section details the lower layers—namely layer 2 and 3—, their architectures, and their sharing mechanisms.

2.1 Packet Networks and the Internet

2.1.1 Foundations of Packet Networks and the Internet

As mentioned in the introduction, Internet is about 40 years old [Zakon, 1997]. Its precursor, ARPANET, has been designed as a *data network* in 1968. At that time, this kind of network was new. Previous communication networks such as telephone networks were based on circuits—or calls—established for the whole duration of the communication, regardless of the actual transmission of information and silences. While circuits were suited to human-to-human conversations, they were not to machine-to-machine.

2.1.1.1 '60s: Packet Networks

The new concept used in data networks has been introduced by L. Kleinrock in early 1960s. The idea was to chop information in fixed-length blocks. In these data networks, information is first divided into small messages which are sent over the network and then reassembled at destination. In the network, the messages—or packets—are processed and forwarded based on the content of the message—most commonly destination written in the header—and algorithms located in the routers or switches. With packets, information only is sent in bursts while silent time periods can be used by others. Circuits were not suited since interconnection resources were scarce.

This proposal was motivated by an efficiency concern. According to L. Kleinrock's analysis of requirements of data networks, here are the major reasons why users of data networks are different from users of telephone networks [Kleinrock, 2002]:

(i) they don't warn you exactly when they will demand access;
(ii) you cannot predict how much they will demand;
(iii) most of the time they do not access;
(iv) when they ask for it, they want immediate access.

The ratio of time when the line is silent is claimed to be as high as 1,000:1 or 10,000:1 for data networks while being about 1/3 and tolerable for telephone lines. This analysis has led L. Kleinrock to propose a different paradigm for communications in data network.

Another thread in the development of packet network is the work of Baran. One particular, dated from 1962 and presented in volume P-2626 of RAND reports *On Distributed Communications Networks* [Baran, 1962], studies the impact of loss of some switches on the survivability of the communications in a network under an assumption of "perfect switching". This assumption states that as long as a path exists between two nodes, communication can be done. This is typically what packet networks tend to achieve by taking routing decisions for each packet.

2.1.1.2 '70s–'80s: Early Stage of TCP/IP and Ossification

The standard model for Internet is now the TCP/IP model. It is made of four abstract layers: Link layer, Internet layer, Transport layer, and Application layer. In this model, shown on Figure 2.1, on a given host of router, information is propagated vertically between the layers. Schematically, going down in the protocol stack means encapsulating data from the upper layer into an object of the lower layer. The reverse process is done when going up the stack. Packets are transmitted from one host to another using the medium accessed using the lowest layer—the link layer. End-to-End connection is realized at the transport layer on top of connectionless communications.

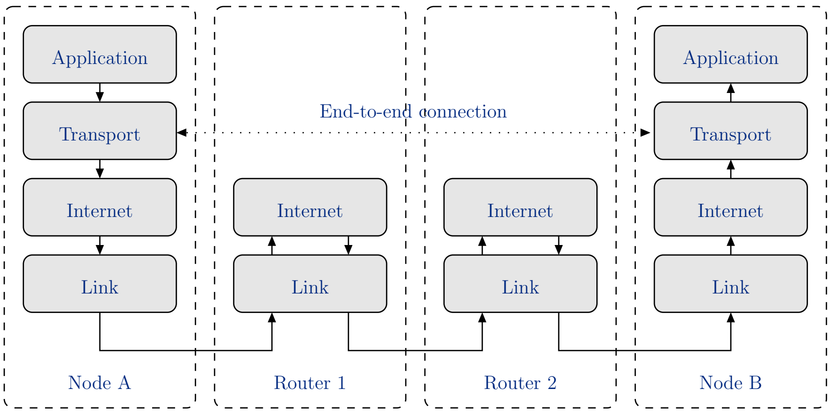


Figure 2.1: Interconnection of network stack in TCP/IP model.

The *International Organization for Standardization* (ISO) along with the *Telecommunication Standardization Sector* (ITU-T)—an activity of the *International Telecommunication Union* (ITU)—defined another layering model called the *Open Systems Interconnection* (OSI) model. This one has 7 layers, from bottom to top: Physical, Data-Link, Network, Transport, Session, Presentation, and Application. They usually are numbered from 1 to 7 and referred to as L1 to L7. L1 does not have an equivalent in the TCP/IP model, L2 is the Link layer, L3 is the Internet layer, L4 and L5 are the Transport layer, while L6 and L7 are the Application layer.

The *Internet Engineering Task Force* (IETF) maintains and develops protocols that fit this model through its working groups and the publication of RFCs. We can note some of the important protocols of the Internet defined by RFCs: the *Transmission Control Protocol* (TCP), the *Domain Name System* (DNS), the *Simple Mail Transfer Protocol* (SMTP)...

When discussions on the design of the Internet protocols started, the concept of packet previously explored in ARPANET was reused. The same applies for the store-and-forward concept. IP routers store packets in queues before processing and forwarding them to the next hop. The first goal retained for the Internet's architecture was to be able to interconnect already existing packet networks and to be able to multiplex their traffic. But then additional objectives have been added. D. Clark lists seven of them in [Clark, 1995]:

(i) robustness;
(ii) support for different types of service;
(iii) accommodation of a variety of networks;
(iv) distributed management;
(v) cost-effectiveness;
(vi) low level of effort to add a host;
(vii) accountability of the resources used in the Internet.

Not all of them have been addressed with the same attention.

Robustness has lead to routing decisions taken at each router based on the destination address to cope with the failure of some routers. When a packet reaches a router, the decision of where it must be sent to reach its final destination is determined by a set of rules. IP uses the concept of subnet to define routing. Subnets are sets of machines in the same network. They share a common part of their IP addresses. In its simplest form, IP routing consists in sending the packet to the next hop defined in the entry of the routing table of the router, the subnet field of which contains the destination IP. One special entry called the default route is used for destinations with no specific entry. The distribution of routing tables can be done statically or using routing protocols such as the *Routing Information Protocol* (RIP) [Malkin, 1998], *Open Shortest Path First* (OSPF) [Moy, 1998], or the *Border Gateway Protocol* (BGP) [Rekhter et al., 2006].

The support of a variety of networks has been solved by the common layer that IP offers. One of the implications of the choice of packets instead of circuits for Internet is that every packets can be dropped. In a circuit-switched network, once the call has been admitted, it is guaranteed not to face blocking. With packets, this can happen at any time. Since IP does not provide any additional mechanisms to protect from losses, it basically offers an unreliable data delivery. In addition, because of the routing mechanisms used, nothing prevents two consecutive packets from taking two different paths and reaching the destination in a different order.

Regarding the support for different types of service, IP does it through the "type of service" field, which was supposed to allow senders to specify the objective that should be aimed when processing packets [Almquist, 1992]:

(i) minimize delay,
(ii) maximize throughput,
(iii) maximize reliability,
(iv) minimize monetary cost,
(v) normal service.

However, in actual use, only the normal service is used and reliability is provided by the transport protocol.

A lot of transport protocols exist, but only two are commonly used nowadays: TCP is reliable, connection-oriented, and provides in-order delivery, while the *User Datagram Protocol* (UDP) is unreliable and connectionless.

Even if version 6 of IP has been designed and implemented, until now, the unification by IP (version 4) has spread so much that it has become an obstacle to change.

2.1.1.3 '90s: Web and Ramp Up

In the '90s, the number of hosts in the Internet grew from 159,000 in October 1989 to 72M in January 2000. While the Internet was mostly exclusively used by early adopters—research community—at the beginning of the decade, it reached its main market before year 2000. This growth was accompanied by the "dot-com bubble" mainly based on new business models supported by this spreading network and its harnessing network effect.

In 1990, the first version of the *HyperText Transfer Protocol* (HTTP), the first version of the *HyperText Markup Language* (HTML), and the first browser and server were developed by Berners-Lee. The expansion of the *world wide web*—the web of nodes—interconnected by hypertext links made the success of the Internet.

2.1.1.4 2000s: Spread of High-Speed Internet and Ubiquity

In the current ending decade, Internet has spread up and, according to *Internet World Stats*, counted 1.59 billion users in March 2009 with 23.8 % of penetration of the global population.

At the same time, broadband connections get widely deployed. Different technologies are in use for this: cable, *Digital Subscriber Line* (xDSL), *Fiber to the "x"* (FTTx)—"x" being one of: neighborhood, curb, building, home... depending where the fiber ends. These technologies provide bandwidth ranging roughly from a few hundreds kbps to 100Mbps or 1 Gbps. This increase has come with the increase of bandwidth-consuming usages and all-over-IP trends. They include video streaming, peer-to-peer file sharing, video-on-demand, voice-, telephony- and video-conferencing-over-IP...

Internet is now ubiquitous in developed countries. Instant messengers have replaced earlier chatting applications such as *Internet Relay Chat* (IRC) and Usenet's newsgroups. Blogs, social networks, and virtual communities such as Facebook, Twitter, Orkut, MySpace, or LinkedIn have made the network effect available to the majority. Mobile phones now provide Internet access, e-mails, and are continuously connected.

The ubiquity of the Internet has made possible the development of *Service Oriented Architectures* (SOA) providing access to software or processing and storage facilities as a service in its economic acceptation. For example, Amazon Elastic Compute Cloud (Amazon EC2) proposes virtual machines as a service, Salesforce.com proposes *Customer Relationship Management* (CRM) software also as a service, others propose online storage or backup. Usage of these services make users/companies dependent on the network—Internet—that connects them to the service provider and the performance it is able to achieve.

2.1.2 Quality of Service of Packet Networks

These data networks made of packets have different performance metrics usually referred to as Quality of Service metrics.

2.1.2.1 Quality of Service Metrics

Measurement, control or even definition of parameters relative to QoS is a difficult topic. Depending on what the system will be used for, the relevant parameters measuring the QoS can be different. Various models have been used to quantify them, evaluate the performance of the systems, and build control mechanisms.

The performance of circuit-switched systems has been studied by A. K. Erlang. His well-known Erlang-B formula gives the *blocking probability*—probability of being rejected—when requesting a circuit from a group of circuits when the calls can not be queued and do not retry. Call attempts are supposed to be a Poisson process and independent with holding time exponentially distributed. The Erlang-C formula gives the *waiting probability*—probability that a call has to wait—under the same assumptions as Erlang-B formula except that calls can wait in an infinite queue if no circuit is available. Note that the probability that a call has to wait for more than a given time can also be obtained since there is a product form for it [Park, 2005, Chap. 3].

When dealing with packet networks, since blocking can happen for each packet, the metrics considered are different.

When the application initiating the communication is *delay-sensitive* or *real-time* as Voice over IP, several metrics regarding the QoS can be of interest: *bit-error rate*, *delay*, *jitter*, *packet loss probability*... They depend on different aspects of the whole system and can be the result of combined effects. A loss of packet can be due to a single bit error in a critical field leading the packet to be discarded by a router. It can also be caused by a congestion in the network. Delay can depend on the transmission delay of the packet as well as its queueing delay at intermediate hops.

Queuing theory was known before data networks appeared but L. Kleinrock was the first to apply it to packet networks to study their performance. Using it, he defined global QoS metrics—averaged transmit time and waiting time of packets over the network—to compare packet networks and design them using relations between the average delay experienced by packets crossing a network, capacity and load [Kleinrock, 1969].

As communications usually involve many packets, higher-order metrics can be used to aggregate the per-packet metrics. Then the specification of QoS constraints can be statistical or deterministic, and possibly aggregated at different timescales. Finally, QoS control can be applied with different granularities. It can be one every single packet [Braden et al., 1994], on flows identified by their 5-tuples (source and destination addresses, source and destination ports, and protocol) [Roberts and Oueslati-Boulahia, 2000], or on an aggregate of traffic [Firoiu et al., 2002].

As an example, timescale is an important point for the definition of throughput: in packet network, a single packet being always transmitted at the maximum throughput or line rate, the rate of information transmission is not constant and made of on-off periods. In order to get the actual long-term transmission rate, the throughput has to be averaged on a time interval leading to different metrics. When dealing with specification of needs, the token bucket model and network calculus can be used to specify the buffer size needed to accommodate transient *bursts* at *peak rate* or long-term behavior using cumulative arrivals and services curves formalism [Le Boudec and Thiran, 2001].

2.1.2.2 QoS Control Mechanisms

Different attempts have been made to improve or control the QoS of packet networks. As of now, none have succeed to be deployed in the Internet. According to Jon Crowcroft's analysis in [Crowcroft et al., 2003], some were too complicated and came before they were needed, others came too late with too little guarantees.

As an alternative to IP, *Asynchronous Transfer Mode* (ATM) has been proposed in the mid-'80s. Different services are proposed by ATM:

- *Constant Bit Rate* (CBR): peak rate is specified and constant.
- *Variable Bit Rate* (VBR): average rate, peak rate, and burst size are specified.
- *Available Bit Rate* (ABR): minimum rate is specified.
- *Unspecified Bit Rate* (UBR): designed to use the remaining capacity.

These services, similarly to *X.25* and *Frame Relay*, are based on connection-oriented communications: *virtual circuits*. ATM uses fixed-size packets called *cells* of 48 B. The ATM protocol suite provides functions of layers 1 to 3 of the OSI model.

Another QoS architecture that has been proposed is *Integrated Services* (IntServ) [Braden et al., 1994]. It is based on the specification of services as token buckets. Both the requested service and the traffic that is going to be sent are described as token buckets and sent through *Resource ReSerVation Protocol* (RSVP) messages. Depending on the type of service requested—*Controlled-Load Network Element* [Wroclawski, 1997] or *Guaranteed Quality of Service* [Shenker et al., 1997]—different guarantees apply. Each router on the path maintains states for each reservation.

IntServ provides a fine-grained QoS architecture acting on flows while *Differentiated Services* (DiffServ) is coarse-grained [Blake et al., 1998] and acts on classes of traffic. Different classes of services are defined and traffic is classified and processed according to the class it belongs to. These classes of services includes *Assured Forwarding* (AF) [Heinanen et al., 1999] and *Expedited Forwarding* (EF) [Davie et al., 2002], with *Best-Effort* (BE) being the default one.

While IntServ and DiffServ are layer 3 mechanisms, *Multiprotocol Label Switching* (MPLS) [Rosen et al., 2001] is called a layer 2.5 mechanism as it lies between layer 2 and layer 3. It provides a packet-forwarding mechanism that does not need to look at the packet's content. It is based on the *label switching* paradigm that consists in determining the next hop using an *ingress label* that is later replaced by an *egress label* before the packet is forwarded. Doing so, the forwarding process executed at each switch is made easier by skipping the *longest-prefix match* that determines which path must be taken. This also enables the creation of circuits in packet networks possibly using *source routing*. Source routing means the path that will be used by the data is determined at setup (possibly by the source). Circuits can be setup by distributing the labels over the path with RSVP, for example. Then, incoming traffic is put in the circuit at the *Label Edge Router* (LER) that assigns the first label to the traffic. An intermediate router is known as *Label Switching Router* (LSR). Depending on how the paths are computed and circuits accepted by routers, this allows offering QoS to traffic put in a MPLS tunnel. MPLS serves as a support for virtual wire or virtual LAN technologies such as *Virtual Leased Line* (VLL) or *Virtual Private LAN Service* (VPLS) [Lasserre and Kompella, 2007]. These services are respectively known as *E-Line* and *E-LAN* in Metro Ethernet terminology [MEF09].

Finally, the solution currently used to provide an acceptable QoS is to rely on *over-provisioning* of the core network. The capacity of links in the core of the network is greater than what access links can accept. For a simple M/M/1 queue, that is a queue served by one server, with arrivals as a Poisson process and exponential service times, the equation for the expected sojourn time at a queue is: $E[T] = 1/(\mu - \lambda)$ where μ is the mean service rate and λ the average arrival rate. It shows that over-provisioning of network capacities helps reducing the end-to-end delay experienced by packets. It also obviously helps with the obtained throughput. And under the assumption that losses are mainly due to congestion and not to transmission error—this assumption does not hold in wireless networks—, over-provisioning also reduces the number of packets queued and so the risk of being dropped at a buffer. Over-provisioning has been the means used to ensure the QoS of the Internet up to now.

2.1.3 Conclusion

The Internet was designed as a packet network for robustness and performance reasons: to get a better blocking probability under scarce communication resources and high silence to transmission period ratio. Under these assumptions, sharing the medium using packets similarly to what was done for time-shared computers makes sense.

The layering model that has emerged of the Internet's design makes the upper-layer depend on the lower. The attempts to bring QoS at layer 3 (IP) have failed. Current trends with MPLS, and virtual wires or LANs now try to address the problem of QoS and sharing of the resources at a lower layer, closer to the physical resources to be shared.

For now, the sharing is the result of the convergence of a distributed process implemented at end-host *in virtue* of end-to-end principle and on top of the lower layers, L1 to L3. In wired telecommunications, there is no problem with sharing at L1 since: (i) wired and optical communications are very reliable and do not face many transmission error; (ii) most media are made of switched and full-duplex segments and thus have no collisions since there is only one sender for one direction.

2.2 Internet L4 Sharing—End-to-End Principle

IP offered a unified layer for the routing and addressing of hosts on the Internet. Even though it supports different transport protocols, TCP is the most commonly used and almost all connections made in the Internet use it. As it provides reliable communications at an affordable cost, this protocol has become the one-fits-all solutions for almost any application. Its congestion control mechanism is responsible for the resource allocation reached by flows and the prevention of network collapse due to congestion.

2.2.1 End-to-End Principle

The *end-to-end principle* states that end-to-end functions such as ensuring the integrity of a transmission should be implemented in endpoints. This is motivated by the fact that endpoints have the knowledge required to do this [Saltzer et al., 1984]:

> The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible.

This principle has been applied to TCP and, more generally, to the Internet's architecture and protocol designs [Carpenter, 1996]. Regarding TCP, it has been applied for integrity and buffer-management functions as well as congestion control functions.

2.2.2 End-to-End Control—TCP's mechanisms

At his beginning in 1974, TCP stood for *Transmission Control Program* [Cerf et al., 1974]. This protocol provided support for different packet sizes, transmission failures, in-order delivery, receiver buffer management, multiplexing of communications between processes, and end-to-end error recovery on an unreliable communication channel.

2.2.2.1 Error Detection/Retransmission

Every transmitted byte in TCP has a known position in the flow of information. This information is sent from the sender to the receiver through the *sequence number* of the first byte of the payload. Since the receivers are supposed to acknowledge received bytes, the senders can figure out whether some have been lost. If the sender receives multiple times the same information for the last acknowledged byte, the sender will retransmit the missing bytes. Improvements of this scheme have since been proposed to help with some loss pattern: *selective acknowledgments* or *negative acknowledgments*.

2.2.2.2 Flow Control

In order not to overflow the receiver with data, in addition to the last sent byte and last acknowledged byte, the sender also keeps a number of bytes that can be sent to the receiver: the *receiver window*. This information is stored in the *TCP Control Buffer* (TCB) structure on the sender. This structure keeps the state information needed for end-to-end control of the communications. The receiver window is announced by the receiver to the sender through packets sent to acknowledge data.

2.2.2.3 Slow-start/Congestion Avoidance

Finally, TCP also manages the congestion of the network. It changes its average sending rate by modifying the number of *in-flight* packets. These are packets that have been sent but not yet acknowledged. The way this *congestion window* is managed depends on the exact variant of TCP and will be detailed in the next section. But, two different behaviors can be distinguished: the slow-start and congestion avoidance phases. In the first case, the transmission starts with a small sending rate—small congestion window—and quickly increases it to probe the state of the network. This phase ends with a loss due to congestion and then the congestion avoidance phase begins. In this phase, the congestion window is increased and decreased according to the congestion control algorithm implemented in the sender.

2.2.3 End-to-End Sharing—Congestion Control Algorithms

Currently, the deployed congestion control algorithms relies on an update scheme known as AIMD using feedback information—losses or delay variations—obtained by the end-to-end control between the endpoints of the communication.

2.2.3.1 Need for Congestion Control

Even though the collapse of the Internet due to congestion, as well as the need for congestion control, have been envisioned in 1984 by J. Nagle [Nagle, 1984], congestion control mechanisms for TCP appeared in 1988, after the collapse really happened in October 1986: the actual transmission rate dropped by 3 orders of magnitude due to retransmission caused by and causing congestion [Jacobson, 1988].

Different mechanisms can be used to control the congestion and the sharing of the resources. They can be open- or closed-loop control, use implicit or explicit and local or global feedbacks. We refer to [Yang and Reddy, 1995] for a taxonomy of the feasible scheme for congestion control. TCP, as currently defined by RFCs, uses a congestion window and reacts to different events: acknowledgments and losses [Allman et al., 1999]. TCP basically increases the congestion window when a packet has successfully been transmitted, and reduces it when it has been lost—assuming it is a signal for congestion. Other approaches have been proposed to detect congestion events such as the increase of the round-trip time.

2.2.3.2 Additive Increase, Multiplicative Decrease

The adopted scheme for congestion window update is termed as *Additive Increase, Multiplicative Decrease* (AIMD). Every time an acknowledgment is received, the congestion window *cwnd* is updated by the relation:

$$cwnd_{n+1} \leftarrow cwnd_n + \frac{\alpha}{cwnd_n},$$

when a loss is detected, the update equation is:

$$cwnd_{n+1} \leftarrow \beta\, cwnd_n$$

where both α and β depends on the variant of TCP, it is respectively 1 and 1/2 for V. Jacobson's version.

In case there is no congestion, this makes the congestion window increase by one every RTT since up to *cwnd* packets can be sent by RTT.

D.-M. Chiu & R. Jain have proved this mechanism to be stable and converge towards a fair allocation of the resources between two users—here, fair means both get the same amount [Chiu and Jain, 1989]. Using fluid models for TCP, L. Massoulie has given sufficient conditions for local stability in general network topologies and heterogeneous delays [Massoulie, 2002]. Using a microeconomics interpretation, the equilibrium reached by the mechanisms can be seen as the result of a distributed optimization procedure solving a utility-maximization problem under capacity constraints. We will go back on this in Chapter 5.

2.2.3.3 Variants

As seen in the congestion avoidance equations in the previous section, for regular TCP, the increase of the congestion window is of one every RTT. When RTT is large, it means that TCP will take time to recover from losses. For high capacity links, such as 1 Gbps links, and under large RTT, typically hundreds of milliseconds, the increase phase can take hours before reaching the maximum rate.

The maximum number of in-flight packets being the congestion window, it is required to be as large as the *bandwidth-delay product* (BDP) in order to achieve the maximum throughput. If it is, the congestion window can contain enough packets to permit the sender to send packets continuously. These high BDP networks have led to the proposal of TCP variants [Weltz et al., 2005]. These variants propose different α and β which can be made a function of the parameters of the TCB. For some of the proposed variants, the congestion events include both losses and increases of the RTT.

2.2.4 Conclusion

TCP and its associated programming model—the socket API—, are used by almost all the applications of the Internet. They provide a reliable service and prevent congestion collapses. Although it is a completely distributed congestion control, TCP is proved—at least, models of TCP are—to have stable equilibrium allocations that are "fair". While the feedback information is coarse and provided by losses, different attempts have been carried to add an extra feedback from the lower layers of the network: *Random Early Drop* (RED) [Floyd and Jacobson, 1993], *Explicit Congestion Notification* (ECN) [Meyer, 2001], or to ask the network elements to determine the best congestion window along the path as in *eXplicit Control Protocol* (XCP).

2.3 Internet L2–L3 Sharing—Switching/Routing

The constitutive elements of a network are: links, queues, and elements doing packet switching. These elements exist at a different scale: from the packet classifier that decides where a packet will be queued, to the AS deciding where a packet will be sent to reach its destination.

2.3.1 Model

The Internet, and more generally the IP networks, use store-and-forward paradigms for transporting packets. Queues are where packets are stored, links are used to transport packets between queues, and switch fabrics decide which packet will be processed and sent to another queue using a link.

Figure 2.2 shows a macroscopic view of the Internet. We progressively have a closer look at its constitutive elements.

At the top of Figure 2.2, the macroscopic view of the Internet is represented. It is made of *Autonomous Systems* (AS) that are authoritative domains. The Internet has about 30,000 different advertised AS in 2009. Some AS host endpoints, such as the AS owned by *Internet Service Providers* (ISP) or content providers. Some are dedicated to the transit of traffic between other AS.

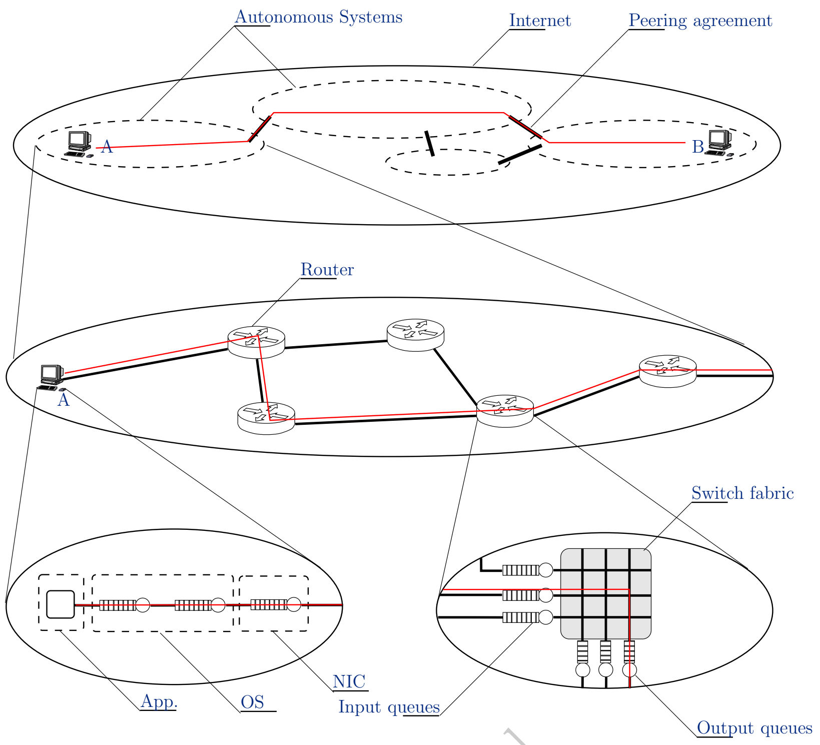


Figure 2.2: A macroscopic view of the Internet

A packet sent from one endpoint to another might cross different AS. Which AS will be used is determined by the routing protocols or, more specifically, by the exterior routing protocol. This will be the topic of Section 2.3.2.3.

Within an AS, as shown on the intermediate stage of the figure, the selection of the links—and hence routers—is done by the interior routing protocol depicted in Section 2.3.2.2.

Links are passive but routers contain memory and algorithms taking decisions. At the bottom of the figure, a switch or router is shown. It basically consists of input ports with queues, output ports with queues too, and a switch fabric that will move packets from one input to an output depending on the decision made by the arbitration algorithms.

In the endpoint, from the application to the wire, packets have to cross the *socket buffers* and *transmit queue* in the OS, before reaching the queue of the *Network Interface Card* (NIC). The same trip exists, in the reverse order, for the received packets, at the destination.

Finally, the endpoints are not always directly connected to a router but instead, there can be many hosts connected to one router through a Ethernet LAN. Between the NIC of the endpoints and the first router will lie Ethernet switches which can be assimilated to routers as once the path as been determined at layer 2, they both do the same task: switching packets between input and output ports.

2.3.2 L2–L3 Path Selection

The general architecture of the network is a generic graph and not a tree. Consequently, there is more that one path between two nodes. Some mechanisms and protocols are needed to decide which path will be followed by packets. This is done at different levels: in an Ethernet LAN, the *Spanning Tree Protocol* (STP) is used; within an AS, it is an *Interior*

Gateway Protocol such as *Open Shortest Path First* (OSPF); between the AS, the *Border Gateway Protocol* (BGP) is the norm.

2.3.2.1 Ethernet Spanning Tree

Since there is no time-to-live field in Ethernet frames, in order to prevent infinite looping of packets, STP is used to remove the loops in the physical topology.

This is done in two steps. First the root of the tree is selected. This selection is based on the ID of the switch—unique for each switch—and a priority combined together and ordered. Then for each node, the shortest path to the root is computed. The ports that are not on one of them are switched off. Variant of this exists to construct this spanning tree within each VLAN.

2.3.2.2 Interior Routing: Link State

Within an AS, since it is a network under a single authority and of limited size, broadcasting routing information between the routers is not an issue. OSPF does that. It broadcasts the state of the links—OSPF is called a *link-state routing protocol*—to every router, so that, after some time, they will have a complete knowledge of the topology and be able to compute shortest paths using Dijkstra's algorithm, or equivalent.

In addition to obvious parameters such as source and destination, announced link states can contain QoS metrics such as delay, bandwidth, or cost to be used while computing the shortest path, either to build the cost that will be used as the metric to be minimized or as a pruning criteria to remove links that are not suited.

2.3.2.3 Exterior Routing: Path Vector

The problem is different when routing between AS, since they are different economic agents. First, privacy concerns prevent AS from disclosing their internal topology and thus IGP does not fit this problem. Second, paths or routes for traffic are the results of business relations and contracts and thus need a policy-based mechanism to announce routes as defined by the contracts that apply.

BGP is a *path-vector protocol*. The idea of this protocol is for each AS to announce the reachability of other AS to the neighbors. Then when an AS A receives an announcement from a neighbor B about an AS C, A knows that traffic for C can be sent through B. Policies can easily be implemented since, in order to prevent an AS from sending traffic towards another, it is enough to not let it know that the AS is reachable. Policies can also be implemented for received routes.

2.3.3 Packet Switching and Scheduling

Switches have multiple inputs and outputs, and since packets take different paths, contentions have to be solved to decide which packet will be served first. Within a queue of packets going to the same destination, packets can also be served in a different order: this is the core of packet scheduling policy.

2.3.3.1 Architecture: Bus/Crossbar/Shared Memory

Looking at the history of switch designs, we can find three different approaches: (i) Shared-bus, (ii) Shared-memory, and (iii) Crossbar. In the first of these, the input/output ports communicate using a single shared bus. Consequently this sharing is a limitation, as no more than one pair of ports can communicate at a time.

The second prominent switch architecture is the shared-memory architecture. In a shared-memory switch, memory is shared by all input and output lines. The packets arriving at the input ports are written to the shared memory, which in due time, are read out and sent to the output lines. If R represents line rate, the characteristics of shared memory switches that make it difficult to scale to their capacity can be summed up as [Iyer and McKeown, 2001]:

- As line rates (R) increase, the memory bandwidth of the switch should also increase linearly ($2 \cdot N \cdot R$),
- Memory has to be accessed twice in every cell slot. For a cell size of 64 B, with $N = 32$ and $R = 10$ Gbps, the access time is just 800 ps,
- Memory size requirement increases with line rate.

Hence, shared memory architecture is not attractive for high-speed switches with a high port density.

The third important design, and the one we're focusing on here, is the crossbar architecture. The classical crossbar switch overcame the bottleneck imposed by the shared bus architecture that restricted the use of N input-output port pairs in parallel, as well as the shared-memory architecture that is impractical for switches with a high port density. The crossbar switch is a $N \times N$ matrix of $2N$ buses, connecting input/output ports. Each of the N^2 *crosspoints*, where the bus lines intersect, needs a control line from the scheduler to turn it on or off, thereby allowing the corresponding input/output port communication. Evidently, one input can communicate with only one output at any given time. For example, if an input line card wants to communicate with an output line card, then a scheduler, based on some criteria (which we will discuss later), has to *turn on* the crosspoint formed by the corresponding input and output lines [Varghese, 2004]. This operation is referred to as *scheduling* or *arbitration*. But since contentions occur, queues are needed to store packets.

2.3.3.2 Crossbar Switches

Depending on where the buffers are located within the switch fabric, there are different kinds of crossbar switch architectures, namely: *input queued* (IQ), *output queued* (OQ), *combined input/output queued* (CIOQ), *crosspoint queued*, etc. Each architecture is designed to meet the increasing performance demands. It also has its own disadvantages, either in performance or in complexity, that make it a challenge to implement and/or deploy [15]. In synchronous switches, packets are divided into fixed-size cells and operations are held synchronously.

If λ_{ij} represents the average cell arrival rate at input i to output j, then the incoming traffic is called admissible if $\sum_{i=1}^{N} \lambda_{ij} < 1$ and $\sum_{j=1}^{N} \lambda_{ij} < 1$, *i.e.,* if none of the input or output ports is over-subscribed. Traffic is uniform if all arriving processes have the same arrival rate, and destinations are uniformly distributed over all outputs. Otherwise, the traffic is non-uniform. An important performance metric of a switch, *throughput*, is the average rate at which bytes leave the switch per port, expressed as a percentage of the line's capacity. Thus, a 100 % throughput means the output ports are fully utilized. This depends not only on the switch's internals, but also on the arrival pattern. Hence, designers of scheduling algorithms not only try to achieve 100 % throughput, but also *stability*. For an admissible traffic, a scheduling algorithm is stable if the expected queue length is finite.

OQ is the reference model for performance. However, in order to be able to transfer all the cells within one slot, it requires the line rates to scale up by N, which makes it impractical.

IQ has been used for the early crossbar switches [Souza et al., 1994]. If an input line card wants to send a packet to an output line card, it first makes a *request* over a separate control bus to the output; whereupon the output sends a ticket to the input over the control bus. The input line then monitors the control bus. The output sends the current ticket number it is serving, as soon as it has finished processing a packet. When the input notices that its number is being served, it places its packet on the input data bus for the specific output line. At the same time, the output ensures that the corresponding crosspoint is turned on. This mechanism avoids contention at the output port. For each incoming packet in a time slot, an input buffer has to make two operations per time slot: (i) write the incoming packet, and (ii), copy a buffered packet to the switch fabric. The IQ architecture is thus attractive in high-speed networks, as the buffers that queue the incoming packets need run only twice as fast as the line rates.

IQ architecture as such is uninteresting due to a major disadvantage it holds: *head-of-line* (HOL) blocking. A packet at the head of the queue, destined to a busy output port, blocks all the packets behind it in the queue (even if they are destined to other free output ports). This reduces the throughput to 58.6 % for Bernoulli packet arrivals with uniformly randomly selected output ports [Karol et al., 1987].

Researchers proposed new methods to combat HOL blocking without resorting to output queueing [Tamir and Frazier, 1988, Obara, 1991, Obara and Hamazumi, 1992, Anderson et al., 1993]. The core idea is to decompose the single input queue in the IQ architecture into multiple queues at the input. More precisely, there will be N queues at each input port, one for each output. These are called *Virtual Output Queues* (VOQs). With the VOQs, a cell at the head of a queue can no longer block any cell destined to another output.

Parallel iterative matching, or PIM [Anderson et al., 1992], was one among those that came with the idea of using VOQ to achieve high throughput. It consists of three phases. In the *request* phase, every input port tells for each output port whether there is a cell destined this output port. The contention at the output port (arising when multiple input ports request the same output port) is solved by randomization; this forms the *grant* phase, where one of the inputs is given grant to send data. If an input port receives grants from more than one output port, it randomly accepts one and signals the same—and this forms the final phase, *accept* phase. To improve the throughput, the algorithm iterates converging to a maximal match, on an average, in $O(log(N))$ iterations. Note that, even if there is no centralized scheduler, the switching is still synchronized due to the iterations; in practice, after a fixed number of iterations, the matching is used to switch cells in the next time slot. As PIM can not totally avoid contention, it can not achieve 100 % throughput under an admissible uniform traffic. There is also a cost involved in generating random numbers at a very good pace.

iSLIP diverts from PIM when it comes to resolving contentions. iSLIP uses round-robin instead of randomness to choose one port among those contending. This permits simpler hardware implementations compared to PIM, besides making iSLIP faster [McKeown, 1999]. iSLIP achieves close to maximal matches after just one or two iterations. Despite being unstable, iSLIP is commonly implemented in commercial switches due to its simplicity.

Until now, we have presented queues that flows, following the same path in the sense of same input port and same output port, will encounter as if they were *First In, First Out* (FIFO) queues. But similarly to the splitting done for VOQ, the output queues can be split up into different queues for different classes of traffic in order to provide differentiated services. They can also be made more active, like RED, which drops packets before the queue is full.

2.3.3.3 Queue Management

Among the services policy, beyond the *First-Come First-Served* (FCFS) and its inherent tail dropping, there are other ways to schedule packets at a queue, such as:

- to improve packet-level QoS metrics, *e.g.,* *Strict Priority* (SP);
- to do *traffic policing* or *traffic shaping* using token bucket;
- to isolate flows, *e.g.,* *Weighed Fairness Queuing* (WFQ), *Stochastic Fairness Queuing* (SFQ) or *Deficit Round-Robin* (DRR);
- to improve the aggregate traffic's characteristics using dropping policies, *e.g.,* RED;
- to improve the flow-level response time *e.g.,* size-based scheduling.

With SP, packets have a priority, and are put in queues of the same priority. The queue with the highest priority is served and, once it is empty—if there is service remaining—, the queue of lower priority is served, and so on. This ensures that packets of the class with the highest priority will be served as if they were alone.

Token buckets are policing or shaping mechanisms parametrized by a rate and a bucket size. The bucket is continuously filled at that rate with tokens that are stored, as long as it does not excess the bucket size. Every time a packet has to be sent, it consumes a token. If there is none, the packet is not transmitted. In case of policing, it will be dropped, while in case of shaping, it will remain in the queue and wait for tokens to be served.

Fair queueing attempts to emulate a bit-by-bit round-robin by sending first the packet that would have the earliest finish time in the ideal sharing. Unfortunately, finish times have to be recomputed every time a packet is transmitted. WFQ, the weighted version, has a $O(n)$ time complexity per packet where n is the number of flows. $O(log(n))$ algorithms exist using *virtual clocks* [Zhang, 1990]. DRR [Shreedhar and Varghese, 1995] proposes a weighted round-robin scheduling policy that support packets of variable sizes by keeping a deficit counter up-to-date. Packets are sent when the counter has enough credits, and if some credits remain, they can be stocked to prevent a potential drift of unfairness in the long term. On a large time scale, both emulate well the ideal sharing in terms of bandwidth but they do not emulate isolated flows on a short time scale. Obviously, the difference between the packets' arrival and departure dates, the transmission delays, are not respected because of the transmission of entire packets.

Some schemes such as SFQ have been proposed to reduce the number of states to keep in routers by aggregating several flows in a single class of traffic and implementing a FQ policy between the classes.

RED assumes that the packets received by the queue belong to a TCP flow. By dropping some of them before the queue is full, it sends a congestion signal to the flows they belong to, which reduces their sending rates (congestion window). This helps avoiding synchronization of TCP flows due to tail-dropping of packets of different flows which will then be synchronized and grow together. This also helps with network utilization since not all the flows are loosing packets—and thus reducing their sending rates—at the same time, which could lead to under utilization of the capacity. The combination of RED and TCP has been proved to be stable [Low et al., 2003]. Even though RED is implemented in routers, it is rarely activated.

Finally other approaches that are being studied use the concept of flow and their ages or sizes. Scheduling based on flow size has been gaining importance in the recent times. Researchers have proposed different ways of scheduling based on size, ranging from *Shortest Remaining Processing Time* (SRPT) to *Least Attained Service* (LAS) to *Multi-level Pro-*

cessor Sharing (MPLS) scheduling mechanisms [Kleinrock, 1976, Avrachenkov et al., 2004, Sun et al., 2007]. The motivation to deviate from FCFS, and schedule flows based on size, is to give a better completion time to small flows. Strictly speaking, the aim has been to improve the conditional mean response time of small flows, at a negligible cost for large flows. LAS, for example, always gives the highest priority to the flow that has attained the least service. More details on size-based scheduling policies and the advantages they bring can be found in [Nuyens and Wierman, 2008] and [Avrachenkov et al., 2004].

2.3.4 Conclusion

In packet networks, the performance a user experiences depends on the the path its packets follow, and on the contention they meet along this path.

The path selection depends on different algorithms that manage aggregates of traffic—all the traffic coming from a given AS—and not flows. At the other end, contention algorithms take per-packet decisions. Both do not know about the user's concern: flows.

Queue-management policies manage packets, and some have knowledge of flows, like in size-based scheduling policies. Different packet-scheduling policies can be used to change the properties of the traffic going through it. Policies that allocate a fraction of the service rate to a given flow, and try to emulate isolated flows, need control decisions to decide how to share. Size-based scheduling does not provide guarantees on the flow completion times, even if it tries to reduce it for small flows.

Summary

Packet networks have been proposed for data networks at the beginning of the '60s. The Internet is based on the unification and interconnection of different packet networks thanks to their common layers, TCP/IP. In this network, the sharing of resources relies on routing and the end-to-end congestion control algorithm of TCP.

The currently implemented sharing is based on a "fair" sharing of the flow rates. All users are assumed to have the same utility, which is a function of the allocated rate. This allocation maximizes a mix of the utilities. These mixing functions—*welfare functions*—and utility functions depend on the exact variant of TCP that is used [Chiang et al., 2007]. Chapter 5 will discuss this in more details.

Guaranteed QoS is not provided by the Internet's architecture, and attempts to add it have failed: the proposed guarantees were on packet-level metrics such as delay or bandwidth, or they were on the traffic aggregate, but in all cases not requested in advance. It was not so interesting for users and costly for operators without a counterpart for them.

Bringing the guarantees to flow-level has been discussed for some years, like in flow-aware networking [Roberts and Oueslati-Boulahia, 2000], but are not deployed for now, neither with a fair approach nor another.

In order to provide high-level QoS guarantees that make sense for users, *e.g.,* transfer completion time, some control is needed to decide the allocation of resources and implement isolation, or simply do admission control and prevent new flows from entering. Without such mechanisms, the coming flows will take some bandwidth and delay the completion of the flows already present. Serving the whole flow at once, on a FCFS basis, reverts to circuit-switching and does not take into account the patience of users, who are not necessarily in a hurry, even if they have deadlines. Serving flows based on their needs and deadlines requires scheduling, in-advance planning, and allocation control: this is the core of our proposal, which relies on the scheduling of large transfers in-advance by the operator so that he can have an interest in doing this to shift large file transfers from peak to off-peak hours.

The advance of technology and control helps that, by making it possible to (re-)configure unified networks combining optical circuits, virtual circuits, and packet switching.

— Three —
From Optical Circuit Paradigm to Bandwidth On-Demand

Introduction

One of the reasons of the success of the Internet and communication networks is the ubiquity provided by radio communications such as Wi-Fi, *Universal Mobile Telecommunications System* (UMTS), or *High-Speed Downlink Packet Access* (HSDPA). Even if the bandwidth available for each customer is improving, it is less expandable than what can be obtained using copper cables and electrical signals, or optical fibers and optical signals. Another element of that success is the ability that the Internet had to increase the available bandwidth to support new users and new usages.

The development of lasers since the '60s, together with the discovery that attenuation of optical fibers can be reduced, have enabled optical communications. The signal attenuation is now less with optical fibers than it was with coaxial cables. The bandwidth capacity provided by fibers is also much higher. NTT transmitted at 14 Tbps over 160 km on a single fiber in September 2006 [NTT09]. In addition, while there are interferences between the electric signals carried in the different cables, thus limiting the number of cables that can be deployed together, there is none between optical signals. This permits to group and install together a large number of fibers between buildings, countries, or continents.

This huge amount of optical capacity raised some new concerns. Packet networks have been used for data networks because of the scarcity of resources. With the increase of the bit rate, and in turn packet rate, offered by fibers and optical technology, the processing of packets has become a challenge in terms of memory access speeds on the equipments, as well as in terms of power consumption.

Although optical packets are being studied—it is not feasible by current state-of-the-art, the current trend is to avoid, as much as possible, the processing of the data on intermediate network elements and to try to establish end-to-end optical connections. In the absence of optical packets and optical bursts, this approach implies going back to circuits for data networks, or at least for the support of data networks. In fact, the current transport networks are layered by transmission technologies and circuits can be used at the lower layers to provide provisionable resources to carry packets at the upper layers.

The first section presents the different switching paradigms that can be used for optical communications. The second presents the functional aspects of transport networks and their classification into planes. The unification of switching paradigms and communication technologies through the unified control planes are then presented. Finally, we survey the mechanisms that have been considered to deliver this bandwidth to users.

3.1 Optical Circuit/Burst/Packet Paradigms

Before describing the different optical switching paradigms, lets overview key components of optical communication technologies which enable the deployment of large capacity in today and future networks.

3.1.1 Optical Communications

Recent years have seen the development of two key transmission technologies in the data networks: wireless and optical. While wireless offers ubiquity to the network, the optical technologies give large bandwidth and low transmission delays.

3.1.1.1 Wavelength Multiplexing

As of 2009, data is processed in electronic devices and is thus available as electrical signals. In order to transmit data over optical fiber, one needs to convert them to an optical signal. This process is performed by a *transceiver* (transmitter and receiver). This is usually achieved using a laser that emits in a narrow wavelength spectrum. The reverse process is achieved through a photodiode that converts light into an electrical signal.

Data is encoded using a *channel code* which basically defines how a *bit* or a group of bits are represented by the optical signal.

Wavelength-division multiplexing (WDM) is a multiplexing method used to carry several different optical signals on one fiber. The frequency grid (central wavelength of each channel and channel spacing) is defined by ITU-T [G.6, 2002]. *Dense WDM* (DWDM) can have up to 160 channels. A less expensive version with a larger waveband and that tolerates more frequency dispersion, *Coarse WDM* (CWDM) has 18 wavelengths. Each wavelength can carry 2.5 Gbps.

Besides the huge capacity that WDM offers it also enables all-optical switching across a network. This contributes to reducing the latency by avoiding O-E-O (Optical-Electrical-Optical) conversions. This switching is enabled by *Optical Add-Drop Multiplexers* (OADM).

3.1.1.2 Optical Add-Drop Multiplexers

An OADM or *Reconfigurable OADM* is a device made of two kind of inputs and the same kind of output ports. These are fibers with WDM signals and isolated signals on wavelengths.

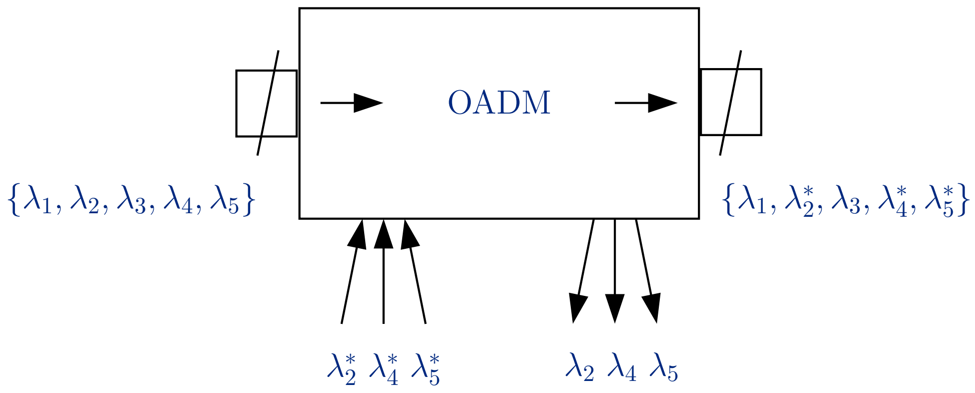

OADM
$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$
$\{\lambda_1, \lambda_2^*, \lambda_3, \lambda_4^*, \lambda_5^*\}$
$\lambda_2^*\ \lambda_4^*\ \lambda_5^*$
$\lambda_2\ \lambda_4\ \lambda_5$

Figure 3.1: Optical Add-Drop Multiplexer.

As its name suggests, an OADM has two functions: *add* and *drop*. Figure 3.1 shows an OADM extracting three lambdas (λ_2,λ_4, and λ_5) from a fiber and replacing them by λ_2^*,λ_4^*, and λ_5^* while the others are just transmitted. Its basic function is to extract some wavelengths from a WDM signal, and thus *drop* them from the WDM signal's viewpoint and replace them by some new signal (*add* function). This device is the cornerstone of all-optical networks. The extracted wavelengths can then be *terminated* on optical receivers and converted into an electrical signal or be injected into a different fiber.

3.1.1.3 Optical Cross-Connect

Optical Cross-Connect (OXC) is the optical switching equipment used by network operators to interconnect fibers and be able to control the routing of wavelengths in this network. Different sub-categories exist. If the OXC uses only OADM, it is called *transparent*: since there is no O-E-O conversion, the OXC is network-protocol agnostic. If the OXC is made of transceivers and an electrical switch fabric, the OXC is said to be *opaque*, since it can only forward signals of a protocol he understands. Figure 3.2 shows an OXC with 2 inputs and 2 outputs. The two OADM permit to extract wavelengths that are routed in the switch fabric before being injected in another fiber or terminated.

Because OXC are made of many input and output ports/fibers which in turn contain many wavelengths, these equipments require *control* to tell them which wavelength must be switched to a specific output wavelength.

Resulting configuration (actually a matching in the bipartite graph made of inputs and outputs) can be set for different timescales depending on the paradigm chosen: *Optical Packet Switching* (OPS), *Optical Burst Switching* (OBS), *Optical Circuit Switching* (OCS). They will be detailed in the next section.

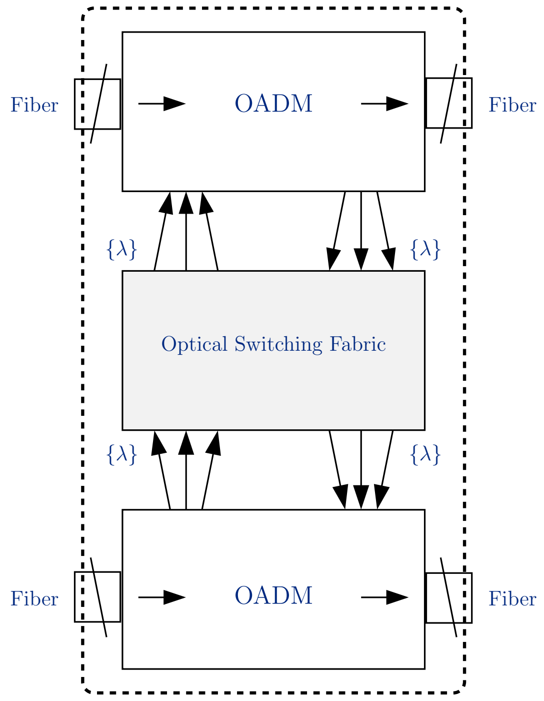


Figure 3.2: Optical Cross-Connect.

3.1.2 Optical Switching Paradigms

Three different switching paradigms have been proposed for optical communications.

3.1.2.1 Optical Packet Switching

With OPS, data is forwarded in a non-connected mode and routing decisions are made at each node [Yao et al., 2000]. This is similar to IP. The difference lies in that it is supposed to be completely done in the optical domain.

As of 2009, this paradigm does not have any implementation because optical memory does not exist and it is difficult to do the processing of the header without any way to store the payload. To overcome this, some of the proposed mechanisms for OBS use a signal message (a sort of header) sent some time before the data.

3.1.2.2 Optical Burst Switching

In the two following models—OBS and OCS—, signalization is used to tell the network equipments that data will follow.

OBS [Qiao and Yoo, 1999] has been proposed with a variety of signalization mechanisms. Some of them use *one-way signaling* and some also use a signaling with acknowledgment. In case of one-way signaling, the *blocking* can happen at any switch as nothing guarantees the availability of resources and/or capability of buffering the data between the sender and the receiver.

This has led [Zalesky, 2009] to propose to classify switching paradigms as either OBS if "blocking is possible [at a switch]", or OCS if "no blocking is possible". Note that signaling can be done *in-band* or *out-of-band* if the signalization message is sent on a different channel than data. Using this classification, the *tell-and-wait* [Widjaja, 1995] signalization mechanism for OBS as well as wavelength-routed OBS [Duser and Bayvel, 2002] fall in the OCS category. We will come back to OCS in the following section.

The *tell-and-go* [Varvarigos and Sharma, 1997] mechanism sends the signaling message just before the data and thus requires buffering at the routers. *Just-enough-time/just-in-time* [Acquaah et al., 2008] send the signal with an advance that depends on the number of hops, so that signalization and configuration can be done before the data reaches the equipments. Still, since the sender does not wait for the acknowledgment and sends data blindly, there is a risk of being blocked at each switch. Figure 3.3 shows this setting where the signal message is sent enough in advance to configure the switch fabric at each intermediate router. *Horizon* and *LAUC-VF* [Xiong et al., 2000] add the length of the burst and delay between the signal and the data so that routers can maintain a calendar of the future utilization of their resources.

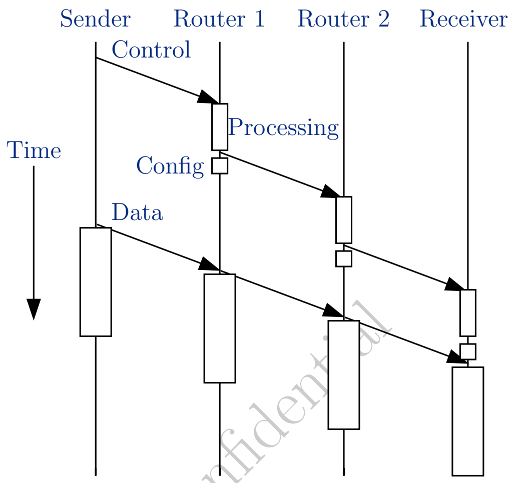


Figure 3.3: OBS control messages are sent in advance so that the data always reaches the router once configuration has been done.

3.1.2.3 Optical Circuit Switching

OCS systematically uses *two-way signaling* and thus waits for an acknowledgment before sending data. This acknowledgment can be either two-way and do a round-trip between source and destination, similarly to what RSVP does, or one-way from a centralized broker.

According to the classification presented in the previous section, optical circuit switching paradigm groups sharing solutions where blocking of data is not possible at a switch. Static optical circuits established manually belongs to this category and the same applies for dynamically established circuits since data can not be *blocked* at a switch.

In order to be sure that data will not be blocked at a switch, a two-way signalization mechanisms have to be used to ensure every switch is fine with this *circuit* before sending data. The configuration process is called lambda-path *setup* and *tear down* is used for the *release* of the circuit.

The selection of the switches used to establish a circuit can be done at every hop, like as in [Widjaja, 1995], or using source routing. This makes controlling this kind of network relatively convenient as exogenous sharing decisions can be taken into account and one can decide the path that will be used to create a circuit. This, in turn, permits *traffic engineering* or on-demand brokerage of bandwidth, depending on if the control is given to internal or

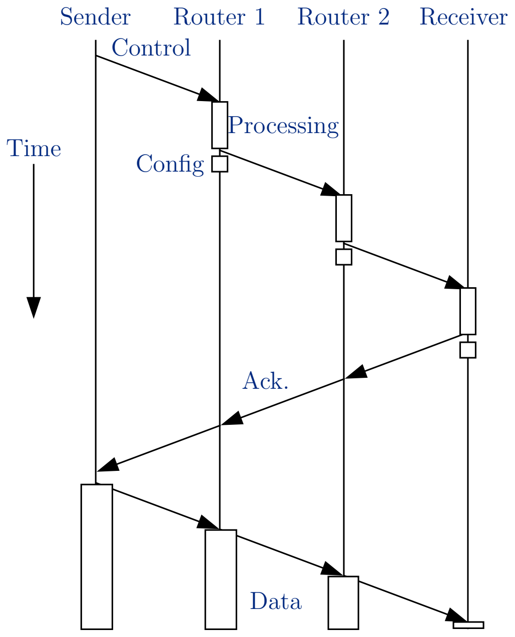


Figure 3.4: Two-way circuit setup and beginning of data forwarding.

	OCS	OBS	OPS
Suitable size of transfer	> GB	Tens of kB	An IP packet's size
Bandwidth guarantee	Yes	No	No
Possible blocking	At setup	At every switch	At every switch

Table 3.1: Brief comparison of the OPS, OBS, and OCS paradigms.

external customers. In addition, *call admission control* can be done on these requests.

This controllability makes optical circuits, together with *time-division multiplexing* such as *Synchronous Optical NETworking* (SONET) or *Synchronous Digital Hierarchy* (SDH), key components of today's *Transport Networks* [Ellanti et al., 2005]. But control has its drawbacks: need to gather/distribute information, to make decisions, to send signal messages. All of these functions are now well determined, and the complexity of managing such networks has been diluted in the functional planes of transport networks.

3.1.3 Conclusion

Some figures and key properties of the three switching paradigms are shown in Table 3.1 as given in [Nord et al., 2003].

Today, commercially available equipments only support OCS. OBS (and consequently OPS) are foreseen technologies. As static optical network can be seen as OCS networks—since, ultimately, configuration at the switching point can be done by someone—, OCS is a common technology in transport networks. But dynamically reconfigurable OCS networks are being deployed. All in all, the OCS paradigm is very similar to the *Public Switched Telephone Networks* (PSTN) that have been in use for decades, except that it uses optical, instead of electrical, signals.

3.2 Functional Planes of Transport Networks

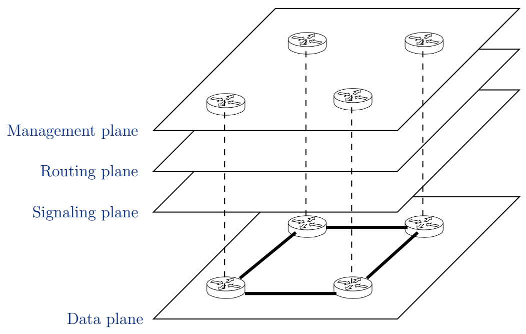


Figure 3.5: Network with its functional planes: Management plane, Control plane (Signaling and Routing planes), and Data plane.

ITU-T gives the following definition for *transport network* in [G.8, 2000]:

> The functional resources of the network which convey user information between locations.

Transport networks are divided along three axes: (i) authoritative boundaries, (ii) technology layers, and (iii) functional planes.

The authoritative boundaries limit the extent of the part of a network under a given authority: this is usually called a *domain*. Each network carrier can have several domains: with different technologies, different purposes, or different kind of customers. Domains are usually interconnected to other domains. This division is called *partitioning*.

A layer is defined by the technology used and groups the equipments or agents that are able to forward data to each other. Within a domain, different layers of technology can coexist, this is the *layering* of transport networks. One can think of an IP/MPLS-over-Ethernet network using SONET circuits on an optical physical layer (fibers), with CWDM as the multiplexing technology. In this example, the layers are IP, MPLS, Ethernet, SONET, CWDM, and L1 fibers.

Each technology brings its constraints and control and management features. In addition to this and in a functional vision of the transport network, functional planes define the logical organization of functions that the network realizes and that can be used to operate and control the network. This model is usually made of three planes: the *Data plane* (or *Forwarding plane*) is responsible for data forwarding, the *Control plane* provides the information and functions to configure the data plane by taking care of the routing aspects, and finally the *Management plane* serves supervision purposes, *e.g.,* the protection of some circuits or traffic engineering, to improve the operational performance or collect statistics. Figure 3.5 represents a domain with its functional planes. These planes can be seen as groups of functions concerned with operations of a different timescale. At one end of this scale is the data plane, which manages data forwarding operations: 12 μs to send a 1500-byte packet on a 1 Gbps Ethernet link. The time scale addressed by the control plane is larger since routing information is not so volatile, and used for aggregates of traffic made of many packets. Finally, management operations are in the time scale of minutes for recovery, to hours or months for network evolution planning.

The next three sections will describe these planes, before going into details of the control plane with GMPLS as an example of a unified control plane.

3.2.1 Data Plane

The data plane is responsible for *forwarding* the data along a *trail*. Trail is an end-to-end connection that can be used by a client. It can be made of *network connections*.

The forwarding process can possibly require to go from one layer to another, *e.g.,* IP packets pushed in SONET frames. This is done by *adaptation* and *termination*. Adaptation describes how to adapt the data from one layer to the other, and termination adds information so that the adapted data can be forwarded by the destination layer. The reverse process is done at the destination.

3.2.2 Control Plane

The data plane is configured using the control plane. This requires the execution of signaling and routing functions. They are sometimes referred to as the *signaling plane* and the *routing plane*, respectively.

Routing This refers to the process of distributing routing information, *i.e.,* connectivity and attributes of links, such as distance or capacity or any information that can be used to compute a path.

Signaling Once a path has been determined, signaling is the process of distributing the information required to be able to forward the data. In circuit switched approaches, this can be the distribution of the configuration of the switches of the selected path. This process can possibly inform the source that this configuration is not possible (*e.g.,* because of contention).

3.2.3 Management Plane

The *International Organization for Standardization* (ISO) proposed the *Fault, Configuration, Accounting, Performance, Security* (FCAPS) management model for telecommunication networks. FCAPS has later been refined by ITU-T in [M.3, 2000b].

Fault Management takes care of alarm handling, detection of fault and network recovery. Configuration Management is responsible for network provisioning—preparing the network to serve users—, resources discovery, backup and restoration. It coordinates addition, modification, or removal of equipments or resources. Accounting Management tracks services usage. Performance Management collects statistics for reporting and addresses the issues related to the overall performance of the network. Finally, Security Management is responsible for protecting the network from unauthorized users. It includes user authentication and authorization through management of permissions, admission rules and logs.

ITU-T G.872 [G.8, 2001b] focuses on optical transport networks. Among others, it defines the following management capabilities:

- *Connectivity supervision* to monitor the integrity of the routing of the connections;
- *Signal quality supervision* to ensure the performance of connections;
- *Protection control* to manage the protections ($1 + 1$ or $m : n$) of connections.

Finally we have to mention the *Operational Support Systems* (OSS) which are the architectural elements that handle management aspects in a telecommunication service provider. Closer to the customers is the *Business Support System* (BSS) which manages products, customers, revenues, and orders. Both operate closely together.

In the ITU-T *Telecommunications Management Network* (TMN) model [M.3, 2000a], Network Management such as previously defined is not the only component, it also contains Business Management, Service Management... The whole defines all the core functionalities needed to operate a network and supply customers.

The TeleManagement Forum (TMF), an international organization of service providers and suppliers to the communications industry provides a framework to build interoperable OSS.

3.2.4 Conclusion

As seen in this section, networks and the function representation which is transport networks have been functionally described in details by telecommunications industry and their standardization body as well as Internet standardization body. Configuration of equipments to deliver the service is part of this and is done in control plane but provisioning is a management function.

3.3 Unified Control Plane: GMPLS/ASTN

In the previous section, we showed the environment of the control plane and, to a limited extent, the functions it implements. This section describes the functions and protocols that are used in *unified control planes* such as GMPLS [Farrel and Bryskin, 2005] or ASTN. The first section presents the *generalized label switching* approach as an extension of MPLS, and the second section presents the protocols constitutive of the GMPLS architecture. Then the extension towards multi-domain is shown as well as architectural models of GMPLS networks.

3.3.1 Generalized Label Switching

We recall that MPLS is a forwarding mechanism that can be used to create circuits based on IP packets and so avoid the usual per-packet routing process. These circuits are called *Label Switched Paths* (LSP). Similarly to OCS presented before—and because this is also a circuit—the configuration of the switches has to be deployed. This configuration tells to each switch that MPLS packets coming with a given label must leave the switch on a predefined output port and with a given new label. The configuration is distributed along the path by RSVP [Braden et al., 1997]. MPLS supports label *stacking* which permits encapsulation of circuits in other circuits.

But it has been generalized to other technologies than IP. It can be noted that many things can be used as a label: wavelength, MPLS label, slot number in a *Time-Division Multiplexing* technology (*e.g.,* SONET), Ethernet VLAN tag... It also has to be noted that an order among these multiplexing solutions is possible. This ordering is given by the hierarchy of switching types: packet switched, layer 2 switched (*e.g.,* VLAN, ATM), TDM, wavelength switching and fiber. Observe that, in GMPLS, stacking can be done on different layers. In GMPLS, LSP have guaranteed bandwidth.

ITU-T proposes a similar architecture to enable automatic delivery of transport service: *Automatic Switched Transport Network* (ASTN)/ASON [G.8, 2001a].

The next sections will describe GMPLS control plane's protocols in details before presenting the extension towards multi-domain and finally the different architectural models that have been proposed using GMPLS.

3.3.2 Protocols: OSPF-TE/RSVP-TE/LMP/PCEP

GMPLS equipments are made of two parts, a switch fabric of one or several of the switching capabilities mentioned before and their instantiation of the following protocols. This realizes a part of the control plane. Routers exchange information through a traditional IP network.

GMPLS is made of three protocols. OSPF-TE [Kompella and Rekhter, 2005] is an extension of the routing information diffusion protocol OSPF. This extension is labeled as *traffic engineering* as it helps doing traffic management operations such as *traffic grooming*. This OSPF can announce two kind of elements: *TE-Links* and *TE-LSP*. The first describe resources that can be used to forward data: a fiber with the lambda it can transmit and how many are available, an OC-48 SONET line and its time-slots... The second, TE-LSP is a LSP announced as TE-Link. This means that they can be used by new LSP of a higher order in the hierarchy of switching capability. From the OSPF-TE database of link state, it is then easy to compute a path through *Constrained Shortest-Path First* (CSPF) by pruning the links that do not satisfy constraints and computing a shortest-path in the remaining graph. Other path computation methods can also be used.

RSVP-TE [Kompella and Lang, 2004] is the extension of RSVP which allows the creation of LSP in GMPLS through *Explicit Route Object* (ERO) describing the list of routers that must be crossed. Then a two-way signaling is performed using PATH/RESV messages.

Link Management Protocol (LMP) is used for parameter negotiation between neighboring routers that share a TE-Link. It also allows detecting faults such as *loss of light*. Actually each of these three protocols have an alternative defined in the GMPLS architecture but these are the more common.

Because of OSPF-TE, which broadcasts the routing information, GMPLS is only suited within a domain, for both privacy and scalability concerns. Routers that are on the border of this domain and adapt incoming or outgoing traffic to another network are known as *Label Edge Routers* (LER) while internal routers are *Label Switching Routers* (LSR).

To provide a path-computation service to entities out of the domain, *Path Computation Element* (PCE) [Farrel et al., 2006] has been proposed. This computation entity, being part of the domain, receives OSPF-TE messages and can thus maintain the OSPF-TE database. This PCE can be used to compute inter-domain path. *PCE Communication Protocol* (PCEP) [Vasseur and Roux, 2009] can be used to contact a PCE and ask him to compute a path.

PCE suffers from several drawbacks or imperfections. Due to the asynchronous update process of the databases, when several paths are requests from PCE at the same time, or if new LSP have been signaled and PCE has not yet been updated, paths given by PCE can interfere and not be feasible together. Another problem is the difficulty to compute a path to a node which is out of PCE's visibility.

In traditional IP networks, there is no PCE. Each domain uses OSPF for intra-domain routing and the *Border Gateway Protocol* (BGP) for inter-domain routing. BGP, as a *path-vector protocol*, announces routes and not link states. Then routes between domains can be used to determine where to go to reach a given destination. Intra-domain routing instances are used to determine paths within the domains.

3.3.3 GMPLS Service Models

A *service model* refers to the description of how a user, or another GMPLS domain, can use the resources of a GMPLS domain. GMPLS has been designed with three models in mind: the *peer model*, the *overlay model*, and the *hybrid model*.

In the peer model, the upper layer is supposed to have a complete access to the routing information of lower layers. Similarly, signaling messages are supposed to be able to go from end to end. In this model, the LSP that might have to be triggered in the lower layer, once established, is announced as a TE-LSP so that the remaining capacity can be used by new requests from the upper layers. The major drawback of this model is the privacy concerns. Networks have to expose their internal structure as well as resource availability to their peers.

The overlay model is closer to the customer/provider relationship model in the sense that domains are distinct and do not exchange information except through a *service interface*. In this model, LSP can be stitched at boundaries to make one end-to-end circuit.

The hybrid model is a mix of the previous two: privileged peers have access to some information and control facilities that normal users can not use; those have to use a different interface to mandate an agent to provision circuits on their behalf. This motivates the need for different interfaces to access the control plane of a domain: the *Network-to-Network Interface* (NNI) and the *User-to-Network Interface* (UNI) [Swallow et al., 2005].

3.3.4 Conclusion

We have seen in this section that control facilities that unify multiplexing technology through label switching have been proposed. They are domain-centric and do not scale as-is to the entire Internet, which anyhow is likely to remains a federation of domains, each with its own control facilities. For now, network operators mainly use the control plane for internal usage such as traffic engineering, *e.g.,* to improve the QoS of regular IP traffic. Finally, time is not managed by the control plane and planning has to be done in the management plane.

3.4 Bandwidth Delivery Services

In this chapter, service can refer to its economic meaning—a non-material good—or to the implementation pattern used to propose it. Traditional network services as offered by ISPs, *e.g.,* xDSL access, consist in providing an access link with a maximum rate, without any guarantee on the rate that can be achieved to a given endpoint. An alternative service, with a different specification, is a rate-guaranteed circuit between two endpoints of the network.

Transport networks and their functional planes can be used to provide on-demand network services to users. While business models for traditional Internet accesses are relatively well defined, a new business model needs to be defined to provide guaranteed resources on-demand through a service-oriented model. This business model is not clear for now, even though several approaches have been explored. Another issue related to the business model is the inter-domain, as it requires to clarify relationships between participants—users, domains. In a finite world where resources have finite capacity, defining this business model defines the sharing of resources, as it defines who is legitimate to get a given amount of resources.

For years, the trend has been to find a way to expose services based on this control plane, in short, to add a service plane to the functional model. Benefit of such guaranteed services have been argued from the perspective of the user's QoS, for example in Grids community [Jukan and Karmous-Edwards, 2007, De Leenheer et al., 2006].

In the next three section we will survey the different services that have been proposed for bandwidth delivery starting from the Internet model which we refer to as *network neutrality* as it is its main guiding principle, then the peer model which similarly to GMPLS service model exposes many information to customers, and the overlay model where customers have very limited information. Finally we will have a look at market-based models where both providers and customers publish their offers and needs—bids and asks—and a third entity—the market maker—realizes the matching.

3.4.1 Current Service Model: Network Neutrality

The current service model for broadband Internet access is mostly based on a flat-price charging. Customers pay for an access link of a given maximum capacity the same price whether they use it or not. The download, or upload, rates their flows will experience is not guaranteed and is determined by many parameters. The network neutrality principle states what should and should not be done to the user's traffic.

The definition of the network neutrality principle is not unique but it basically states that all Internet traffic should be treated equally. This subject—whether this principle must still be applied to ensure innovation on the Internet, mitigated to provide QoS, or dropped for the sake of the network provider's profitability—is highly debated.

Following this principle, flows get resources according to global mechanisms such as TCP congestion control and the queue management algorithms of IP routers. Some argue that packets must be served on a *First Come First Served* (FCFS) basis to statisfy the network neutrality principle. In this model, the QoS that users will experience—absence of loss, absence of severe congestions, bandwidth obtained—is provided by over-provisioning the core infrastructure.

The next section will present the architecture or ecosystem of the current Internet model in terms of agents and how they accept others' traffic. Then, some limits of this model will be reviewed.

3.4.1.1 Structure

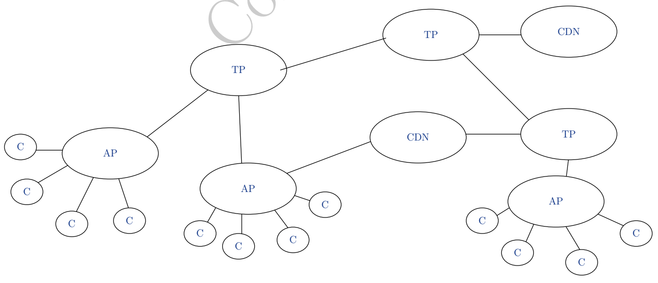


Figure 3.6: Network neutrality ecosystem: Customers, Access Providers, Transit Providers and Content Providers.

This ecosystem is made of: *Customers* (C) or users, *Access Providers* (AP) or *Internet Service Providers*, *Transit Providers* (TP), and *Content Providers* (CP). Distinction can be made between enterprise and home in customers group to push the analysis further [Dhamdhere and Dovrolis, 2008b, Xiao, 2008]. Figure 3.6 represents an example of such ecosystems.

The main characteristic of customers is that they mainly access content provided by content providers. Access providers are connected to customers and have to negotiate connection to content to attract users.

Interconnections of the aforementioned networks are done in *Internet eXchange Point* (IXP) [Awduche et al., 1998]. There are basically two kinds of relationship between providers:

peering and *transit*. While the peering relationship is a reciprocal access to each other's contacts, transit is a unilateral access against payment. These are business decisions and, as such, based on profitability or expected profitability [Baake and Wichmann, 1999, Chang and Jamin, 2006]. It can be noted that BGP, the inter-domain routing protocol used in the Internet, is particularly suited to implement such contracts. By not announcing routes to a neighbor domain, it makes this domain not able to use a given domain for transit as he does not know he can reach his destination through it.

The result is that the structure of the global Internet is made of domains (AS) interconnected differently depending on which kind of provider they belong to [Magoni and Pansiot, 2001]. Access networks are connected to customers but have an asymmetric need for bandwidth from other providers as their customers receive more than they send data/content. User traffic is primarily directed to content providers. They might not be accessible directly for the access provider. In this case, the access provider has to buy a transit agreement from a transit provider. Alternatively, the access provider can convince the content provider to establish a direct connection. The customers of transit providers are access providers and content providers, with the first willing to be able to reach the second. Content providers also want to be reachable to increase their visibility.

3.4.1.2 Limits and Concerns

Network neutrality by treating equally all traffic would forbid QoS improvement of part of the traffic. Network neutrality is supposed to ensure a "fair" treatment of traffic by doing the same for all. Since different traffic can have different needs, it would penalize applications having higher needs such as high-quality video conferencing. Then, depending on the utility obtained for a given value of a QoS parameter (bandwidth for example) and the purpose this flow serves, a fair allocation of rate can result in an unfair allocation of real user utility. As almost any kind of utility can be envisioned and can vary from customer to customer, this debate is pointless and endless: one system being fair for some setting will not be for another. The same applies for social welfare.

In that sense, [Dhamdhere and Dovrolis, 2008a] moves this debate to the question of ISP profitability. They propose a study of this profitability under an assumption of heavy-tail traffic distribution and highly-skewed popularity of CP, and different pricing models labeled as network neutral, as they are not based on the sources of content. These schemes are: charging of heavy hitters, limiting volume for heavy hitters, or charging CP. They conclude from a quantitative study that strategies based on charging are rarely profitable or are highly sensitive to factors out of the control of the AP and that direct peering with major content providers can be made profitable even if it is not network neutral.

3.4.1.3 Conclusion

In this model, bandwidth provided by the addition of new fibers, or by the setup of a new optical circuit, is not directly delivered to users, but to the network.The bandwidth is then redistributed among flows by the sharing mechanisms of the TCP/IP protocol suite.

To conclude this section, we refer to [Crowcroft, 2007] where Jon Crowcroft tries to list the different aspects of the issues and concerns related to network neutrality. He concludes that network neutrality never existed and that this must remain an ideal and should not become a static constraint limiting innovation.

After this overview of the current service models and how bandwidth is made available to users, we move to the models for on-demand network services: peer and overlay models for bandwidth-delivery services and market-based models.

3.4.2 Peer Model for Bandwidth Delivery Services

In the peer model, resource descriptions are made available to the users. These descriptions can include all the fibers and internal network elements such as routers or switches, or can be made of abstract services that can be composed by users.

Users can then decide what they will use based on their needs and the resource availability published by network owner. Users then inform the network that they will use the resources.

In the next sections, we present some projects and proposals revolving around this model. Some are new architectural elements providing functions that can be grouped in a new functional plane: *service plane*, some are infrastructures, testbeds, or research projects.

3.4.2.1 UCLP/Ca*net

User Controlled LightPath (UCLP) [Wu et al., 2002] is a solution to configure, partition, and expose lightpaths and network elements of a physical network. It manipulates lightpath as an object and allow users to create their own topologies. This solution has been proposed for Ca*net, the Canadian NREN. UCLP has since been developed as a commercial product under the name Argia. Manticore extends this model to layer 3 and provides IP slices.

3.4.2.2 GLIF

Besides being an organization promoting lambda networking and proposing technical solutions through its working groups, *Global Lambda Integrated Facility* (GLIF) is a federation of optical network owners who propose optical inter-connections for research purpose and e-science. The topology of this network is made of exchange points known as *GLIF Open Lightpath Exchanges* (GOLEs) inter-connected by lambdas. As this topology is publicly available (Figure 3.7), and since connection across this network can be requested, GLIF falls in the peer model for bandwidth on-demand.

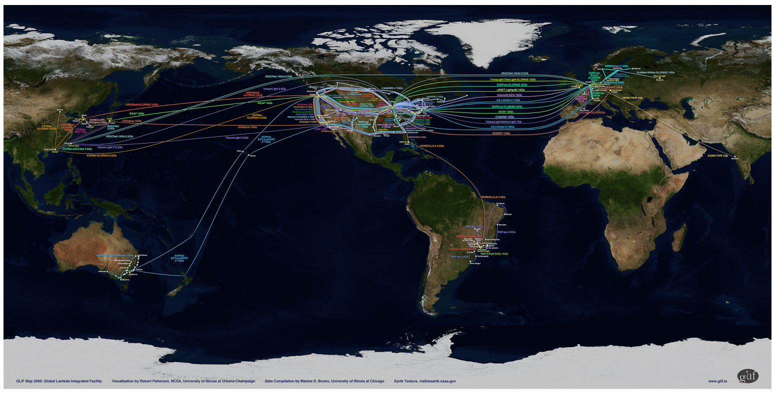


Figure 3.7: GLIF's exchange points and lambda (source http://www.glif.is).

3.4.2.3 Service-Oriented Peer Model

The peer model has also been investigated with service-oriented architecture.

Service Oriented ASTN (SO-ASTN) [Baroncelli et al., 2005] is a proposition of a service plane on top of the ASTN architecture. Its applications-oriented interface proposes to query information such as the delay, BER, or available and utilized bandwidth of a set of resources, matching the requested virtual topology.

Similarly [Verdi et al., 2007] proposes a service-oriented architecture where domains publish the services they propose in a directory, and the users select which service suits their needs and invoke it.

3.4.2.4 Conclusion

In this peer model, descriptions and information on resources are made available to users who can then select the ones they want to use before telling the broker and using them. In this model, the service is composed by users with the knowledge of what can be supported by the provider. The major concern regarding this model is privacy. It might be suited for NRENs but, for now, still raises objections from network providers. It can be noted that GÉANT2, which interconnects the European NRENs, proposes through *Automated Bandwidth Allocation across Heterogeneous Networks* (AutoBAHN) [Campanella et al., 2006] a provisioning service which uses the overlay model.

3.4.3 Overlay Model for Bandwidth Delivery Services

Another model for resource reservation is possible. In this model, users specify their constraints, pack them in requests, and send those to a broker. Then, based on resource availability, policy, and constraints expressed by the user, the broker will assign resources to serve the request. In this model, there is no need to leak any extra information about what is available or not. Just like in PSTN, when a user calls someone, he implicitly issues a request for a voice circuit between two endpoints. If the network has enough resources (and the callee is available), the call will succeed and resources will be dedicated to this. If not, the call will simply be rejected. Similarly, the establishment of leased line connections follows a similar process, except that it involves manual operations.

3.4.3.1 GNS-WSI

Grid Network Service - Web Services Interface (GNS-WSI) is an interface proposal to perform bandwidth reservation in networks. Using this interface, users can requests in-advance end-to-end paths, by specifying the two end-points. The underlying network is abstracted as a cloud. This interface can be used between clients and *resource managers*—they own resources—, or between clients and *resource coordinators*—they do not own resources but retail the resources requested from RM or other RC, still using the same interface. Figure 3.8 shows this stacking through the interface represented by the arrows.

This interface has been proposed by the G-Lambda [G-L09] project, which is a joint research of the *National Institute of Advanced Industrial Science and Technology* (AIST), KDDI R&D Laboratories, NTT, and the *National Institute of Information and Communications Technology* (NICT).

3.4.3.2 Carriocas

In the Carriocas project [2, Audouin et al., 2007], the *Scheduling and Reconfiguration Service* accepts requests from clients (which can possibly be a higher-order service provider) and serve them based on its internal representation of the network and computing resources. Network resources are assumed to be supplied by a network operator as a connection service described by endpoints and capacity. The same is done for computing resources, which are published in the SRV by the owner.

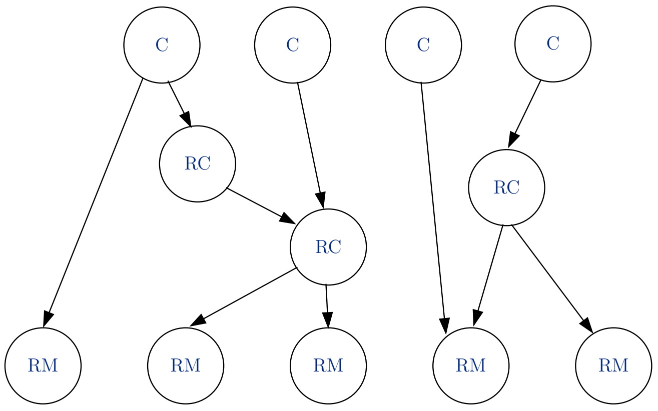


Figure 3.8: Clients, Resource Coordinators, and Resource Managers.

3.4.3.3 StarPlane

The StarPlane project provides a way to dynamically configure the optical network supporting DAS-3, the Dutch grid. The network control plane is based on DRAC. This interface provides both information about resource availability, and a mean to reserve resources. The *StarPlane Management Plane* (SPMP) [Grosso et al., 2009] provides applications with an interface to specify constraints on the abstracted services that are feasible.

3.4.3.4 Phosphorus

European project Phosphorus proposes a service plane for grid that manages both computing resources and network. The *Network Service Plane* (NSP) is this service plane and receives requests from the grid middleware. Based on the knowledge this service plane has, it determines an end-to-end path accross the domains [Figuerola et al., 2007]. This service plane is south-bounded to *Network Resource Provisioning Systems* (NRPS) such as UCLP, DRAC, or ARGON. In this context *Grid-GMPLS* (G^2MPLS) [Ciulli et al., 2008], an extension of GMPLS, is proposed to act as a control plane taking computing resources into account.

3.4.3.5 DRAGON

The *Dynamic Resource Allocation via GMPLS Optical Networks* (DRAGON) project [Yang et al., 2006] aims to develop technologies required to perform dynamic and in-advance reservation of networks resources in an heterogeneous and multi-domain network. Domains show an abstracted vision of their internal topology to other domains. DRAGON provides advance reservation of circuits set through a client-side API. This API communicates with the *Application Specific Topology Builder* (ASTB), which finds the resources suited for this request. DRAGON can manage multi-domain networks. Each network has its own *Network Aware Resource Broker* (NARB) that can exchange routing and signalization information with other domains.

3.4.3.6 Conclusion

In the overlay model, no information is leaked to users except for the one they get with the request acceptance notification, but this can not be avoided. Still, information needs to be exchanged between domains in the case of multi-domain requests, if this in implemented in a *chain model* where domains forward requests to the next one for acceptance—see Figure 3.9(a).

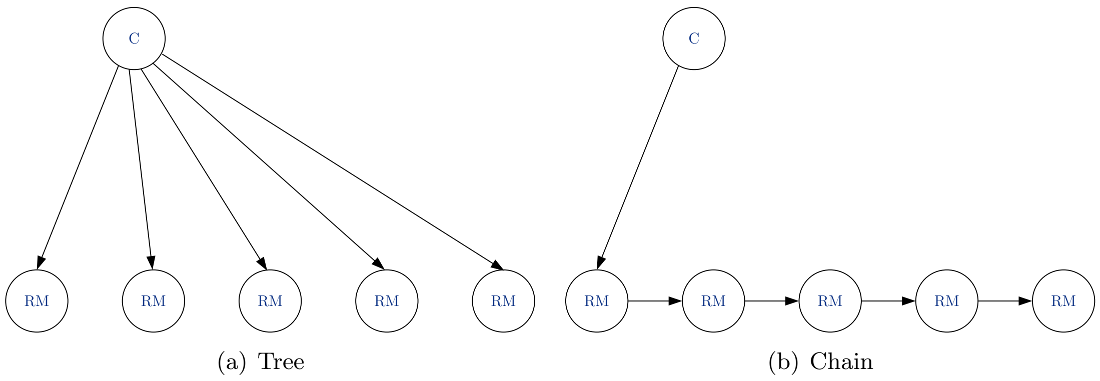


Figure 3.9: Tree- and Chain-model for reservation.

On the opposite, if multi-domain is implemented using the *tree model*, domain composition can be done by a third entity having a customer/provider relation with both domains. This requires a minimum of topological information to be available, such as the domains' peering points. Alternatively, it can also be done by the customers if they know the resource managers as shown on Figure 3.9(b).

3.4.4 Commodity Market—Bandwidth Market

In the previous two section, we have seen two different models. One in which users select the resources they want to use, and one where the network provider selects the resources based on users' constraints. Another solution to resource allocation is commodity markets where both side publish offers: offers to buy and offers to sell [Cheliotis, 2001].

In such markets, sellers propose bandwidth at a given price (ask) and buyers requests bandwidth for a price too (bid). Then the market maker finds matching offers. Different kinds of auction are feasible, but usually markets are built on double auctions. With this kind of auctions, both buyers and sellers proposes a price together with the description of the commodity, then the market maker sets the price of a commodity so that it clears the market—all the sellers who asked less than this price sell it to all the buyers who bid more than the price.

Commodity market supposes that bandwidth has been commoditized by defining standardized contracts to describe offers in order to have a liquid market where commodities are not tailored too much, and thus too specific [Fusaro, 2002]. In case the traded commodity is a segment (point-to-point connection), this implies specifying the source and destination of the segment, the QoS attributes, the duration of the "reservation", and the date at which it will start.

Segments can then be combined and possibly resold, creating a possible arbitrage if all the equivalent paths do not have the same spot price. Being able to combine segments and ask for them as a whole has led R. Jain and P. Varaiya [Jain and Varaiya, 2005] to propose combinatorial double auctions with an integer linear program to realize the matching.

In-advance bandwidth reservation can be seen as a derivative of the commodity as futures contract. Other derivatives, such as options, could also be used. This market would enable risk transfer, hedging, and speculation on future needs or shortage of bandwidth between some exchange points. All this would make the spot price of bandwidth similar to the spot price of crude oil or cotton and varying with both supply and demand ratio and speculation.

As bandwidth is not a storable good, it makes it similar to power market. Actually, in the beginning of 2000, some energy firms—*e.g.,* Enron—have created a bandwidth market. But it failed to mature and disappeared with Enron. The reasons why it did so are not clear but some of the proposed elements of explanation include the relative low price of bandwidth and technical difficulties to deliver on-demand bandwidth in early 2000.

3.4.5 Conclusion

Because of the increase of demand from some users, the current Internet model is said to not be sustainable by network operators. In addition, as the current model is not able to provide guaranteed bandwidth to customers willing to pay for it, new sharing models are being studied and/or have been deployed in some specific context—such as NRENs. These reservation models basically fall in three categories, depending on who takes the allocation decisions: clients, resource managers, or coordinators/market makers. All the presented models deal with bandwidth requests even though it is mentioned in Phosphorus' deliverable that malleable requests can also be proposed in the interface. In the *mechanism design* theory, there are two typical solutions to the resource allocation problem [Hurwicz and Reiter, 2006]: *parameter transfer*—where the client submits a request to the operator, or the operator exposes resource availability to users—and *direct revelation*—both give their parameters to a third agent. Among them, it is likely that network operators will prefer the ones where they do not have to leak any information.

Summary

Optical networks provide a huge amount of bandwidth. The currently-available switching paradigm is optical circuit switching, and requires configuration and control. For this purpose, the functions required to operate a network have been grouped in different functional planes.

Unified control planes such as GMPLS offer a set of functions to control these networks and establish circuits for different technologies. On top of it, management planes provide the management of configuration, monitoring, and performance-related functions. These technologies have been designed for the operator itself.

But, for some time, research and networking communities have been concerned by providing users with on-demand provisioning. Different approaches have been studied for different contexts: NRENs, private networks, telcos, the Internet.

Among them, the on-demand service approach for bandwidth delivery gives a clear interface between the customer and the provider, where the first expresses his constraints in a request submitted to the second. If it is accepted, this specification serves as a contract between them, like a *Service Level Agreement* (SLA). This kind of solution can be of interest for users since this can move the risk of blocking, that exists for every packets in packet networks, to the reservation procedure. In this scheme, provided the request is not rejected, users would be able to plan their utilization of the network and to obtain the resources they need to complete their tasks as planned.

For now, the network neutrality model prevails and the allocation of bandwidth is defined by the TCP/IP protocol suite, TCP's congestion control algorithms, loss events, and the arrivals and departures of users. The service consists of the right to use the access link provided by the network operator, but is not based on the bandwidth that a particular flow gets. There is no agreement between the user and the network provider about the bandwidth allocation since the network provider only manages the capacity of the network, and not precisely how it is shared. In this context, as we will highlight in the next chapter with the study of transfer completion time, predictability of performance is not given to users. Predictability is of importance when the network is not the only resource used. As an example, if the network is used to interconnect different sites for a video conference, in the absence of good enough network performance, the meeting is affected. If a dataset has not reached the machine where it is supposed to be processed at the time the reservation starts, CPU time or money is wasted.

— Four —
Exploration of Current Bandwidth Sharing Model

Introduction

When using the classical TCP/IP protocol stack, the sharing between flows depends on the mechanisms of each layer, as presented in chapter 2. In the presence of multiple flows, since no resources are dedicated to flows, they are not isolated from each other in terms of performance. In this chapter, we illustrate this using the transfer completion time as a metric, since it is assumed to be what interests users the most [Dukkipati and McKeown, 2006].

In the first section, the predictability of transfer completion time is studied. The second section has a deeper look at the interactions between layers and the behaviors of flows observed on the output port of a layer-2 switch under congestion.

4.1 L4—Unpredictability of Transfer Completion Time

In this section we study the impact on transfer time predictability of the interactions of transfers using different variants of the TCP congestion control algorithm. Experiments are conducted under different conditions, including the presence of reverse traffic and variations of the number of contending flows. This high-level performance metric is of interest for users since, in the end, it is what they will have to wait for their transfers to complete.

The lack of predictability of network performance makes it difficult to utilize combined resources to realize a task, when the resources have to be reserved in advance and the task involves utilization of the network. In order to plan the start time of a task, one has to know when the preceding tasks will be finished. If it is not possible, then users cannot precisely predict their resource needs.

4.1.1 Motivation

The goal of this work is to explore this issue and to examine how recent transport protocol enhancements could benefit to high-end applications, in terms of data transfer efficiency and predictability in the absence of any access control and reservation mechanisms. It is centered on elephant-like bulk data transfers in high-capacity (1 Gbps, 10 Gbps) networks and on the new variants of TCP that are currently available on end nodes. The systematic evaluation of the protocols in Grid'5000 and in a fully controlled testbed provides a set of measurements of the transfer time in a broad range of conditions. We explore mainly three factors: synchronization of start time, congestion level, and reverse traffic.

The work is organized as follows. Section 4.1.2 describes our experimental methodology and testbed. Experimental results are given and analyzed in Section 4.1.3. We systematically study three factors influencing protocol behavior and affecting the predictability of data transfers. Finally, we conclude this study in Section 4.1.4.

4.1.2 Methodology

The enhancement of TCP/IP has been intensively pursued to tackle limits encounter in large bandwidth-delay product environment. Different TCP variants have been proposed to improve the response function of the AIMD congestion control algorithm in large-bandwidth delay product networks. All these protocols are not equivalent and behave differently according to network and traffic conditions. In this work we concentrate on the TCP variants available in recent GNU/Linux kernels: High Speed TCP, Scalable TCP, Hamilton TCP, BIC TCP, and CUBIC.

To analyze the acquired data, several metrics can be used to synthetically characterize the behavior of different TCP variants [TMR, 2007]. This work is focused on the transfer time metric, which belongs to the throughput metric family. Indeed, throughput can be measured as a router-based metric of aggregate link utilization, as a flow-based metric of per-connection transfer times, and as user-based metrics of utility functions or user waiting times.

The Grid'5000 platform allows dynamic deployment of network stacks. The network infrastructure is an interconnection of LANs (*i.e.,* sites) and a 10 Gbps lambda-based private network [Bolze et al., 2006]. We use *iperf*, version 2.6.16 of the GNU/Linux kernel with the *Web100* patch and the CUBIC patch, to perform our experiments.

Figure 4.1 presents the topology used in our experiments. It is a classical dumbbell, with N pairs of nodes able to send at 1 Gbps on each side. One flow by node pair is used to perform a file transfer. N is subdivided into two parts, according to the function assigned to the nodes. N_f refers to the number of flows on the forward path (from A to B) and N_r the number of flows on the reverse path (from B to A). The bottleneck is the L2 switch. Here, Grid'5000's backbone could be the 10 Gbps link between Rennes and Toulouse (experiments at 19.8 ms RTT) or a 1 Gbps link between Rennes and Lyon (experiments at 12.8 ms RTT).

Definition 4.1.1 (Multiplexing Level). The *multiplexing level* N_f is the number of flows of the forward path.

Definition 4.1.2 (Congestion Level). The *congestion level* is defined as the ratio between the sum of the N_f nodes' nominal capacity and the bottleneck capacity.

Definition 4.1.3 (Reverse Traffic Level). The *reverse traffic level* is the ratio between the sum of the N_r nodes' nominal capacity and the bottleneck capacity.

We explore the following parameters: starting time, congestion level, and reverse traffic

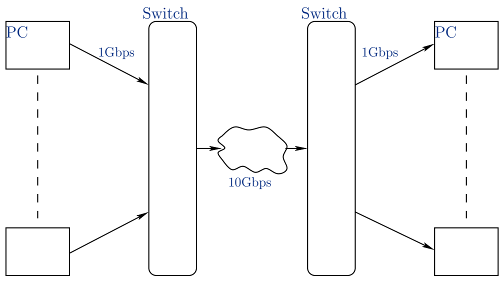


Figure 4.1: Experiment topology

level. We are considering several metrics. The primary metrics that will be used is the mean completion time, as defined in Def. 4.1.4:

Definition 4.1.4 (Mean Completion Time). $\overline{T} = \frac{1}{N_f} \sum_{i=1}^{N_f} T_i$ where T_i is the completion time of the i^{th} of the N_f file transfers.

The transferred files are 30 GB each. Additionally, we also use the following metrics that are more suited than just mean values to characterize the variability of completion time:

Definition 4.1.5 (Maximum Completion Time). $T_{max} = max(T_i)$

Definition 4.1.6 (Minimum Completion Time). $T_{min} = min(T_i)$

Definition 4.1.7 (Standard Deviation of Completion Time).

$$\sigma_T = \sqrt{\frac{1}{N_{forward}} \sum_{n=1}^{N_{forward}} (T_i - \overline{T})^2}$$

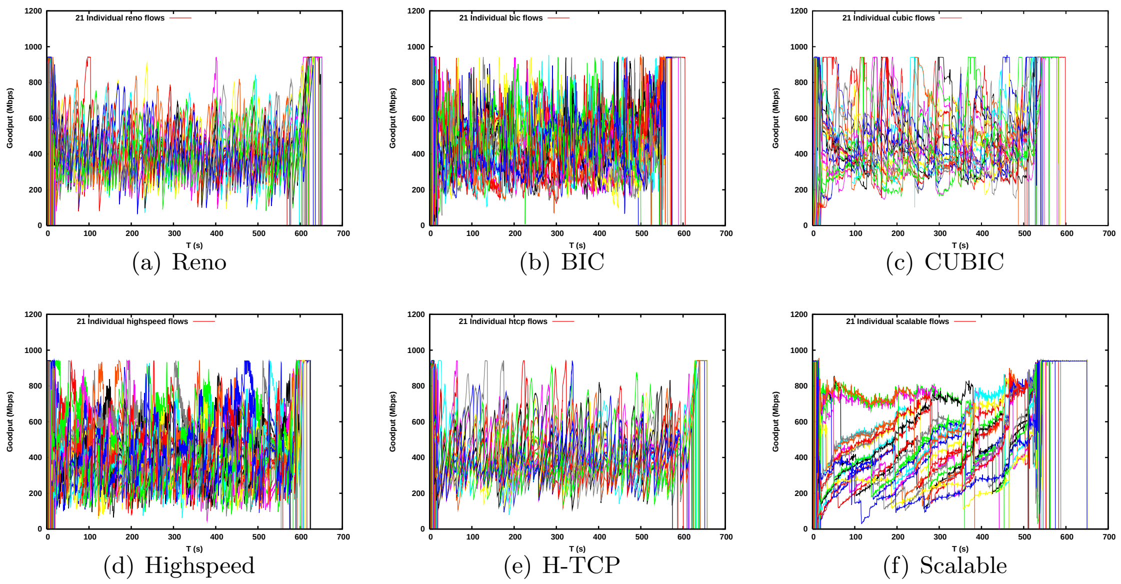
Definition 4.1.8 (Coefficient of Variation (CoV) of Completion Time). $CoV = \frac{\sigma_T}{\overline{T}}$

Each experiment for a given TCP variant, N_f, and N_r, was executed at least three times to ensure that our measures were consistent. By choice of the volume to transfer, an experiment would last an average of 400 s.

4.1.3 Transfer Completion Time Predictability of TCP on High-Speed Networks with Low Congestion Level

The experiments that were conducted on a 10 Gbps bottleneck in Grid'5000, were all performed between the Toulouse cluster and the Rennes cluster, Parasol. Both used Sun Fire V20z machines. The bottlenecks along the path are the access links of the sites. It is the output port of a 6500 Cisco in the Toulouse cluster and a 6509 Cisco in the Rennes cluster Parasol. The size of the buffers of the edge switches are not known. The cross traffic on the forward path is considered to be negligeable: background traffic of grid'5000 was around 400 kbps.

As preliminary experiments show, having multiple starts of flow at the same time causes some of them to face severe congestion because of the slow start. For the rest of our experiments, we chose to set the starting delay between transfers to 1 s to avoid potential harm from this parameter since, at best, slow start takes $(log_2(N) - 1) \cdot \text{RTT}$, for a congestion window of N packets. For a 19.7 ms RTT, N is close to 1600 and slow start lasts about 200 ms.



(a) Reno (b) BIC (c) CUBIC

(d) Highspeed (e) H-TCP (f) Scalable

Figure 4.2: Impact of a high congestion level: 2.1 congestion level (21 flows), 19.8 ms RTT, on CUBIC and Scalable

4.1.3.1 Congestion Level

In this section, we are considering the impact of the congestion level alone on different TCP variants. Congestion only takes place on the forward path. There is no significant reverse traffic except acknowledgements.

General Behavior Figure 4.2 shows the impact of a high congestion level on every TCP variant. For example, we observe that the predictability of a transfer time with Scalable is bad, as there is more than 300 s between the first and the last completion time. Even though each protocol is able to completely fill the link, they all have a different behavior. Bandwidth sharing with Reno, BIC, CUBIC, Highspeed, and H-TCP is fair among the various transfers leading to a smaller variance in the completion time.

Figure 4.3 presents the impact of the congestion level on mean completion time for several TCP variants. The ideal TCP represented on the same figure corresponds to a TCP able to send continuously over 1 Gbps links, without slow start phase, without losses or retransmissions, and with equal sharing of the bottleneck link. All protocols, except Scalable, behave similarly: one of our works [4] has shown that for the RTT used in our experiments here, most TCP variants tend to perform similarly.

Predictability Figure 4.4 presents the completion time distribution of all the TCP variants. Scalable is somewhat remarkable as it often displays the shortest and the longest completion time for a given N_f. Even though both distributions are roughly Gaussian-shaped, Scalable is more spread out (294 s *vs* 114 s for the 2.1 congestion level case) than CUBIC for instance. It makes Scalable a poor choice if we need to wait for all transfers to complete. But if we can start computation on a limited dataset, we might be able to increase the usage of the computation nodes. It might not be the case in other applications like astronomy interferometry that will

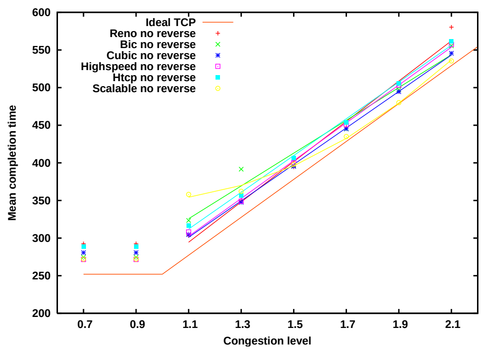
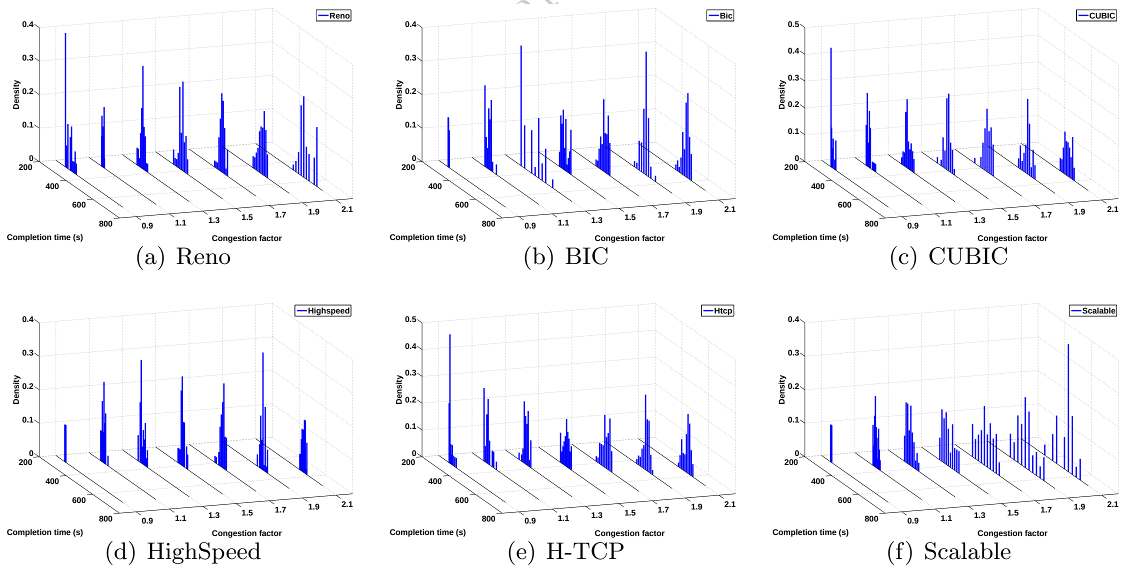


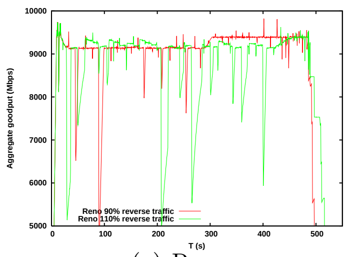
Figure 4.3: Impact of the congestion level on the mean completion time for all TCP variants



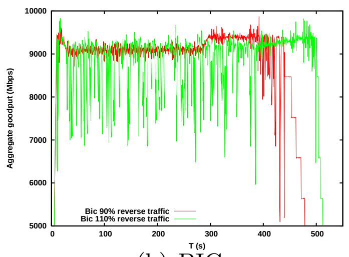
(a) Reno (b) BIC (c) CUBIC

(d) HighSpeed (e) H-TCP (f) Scalable

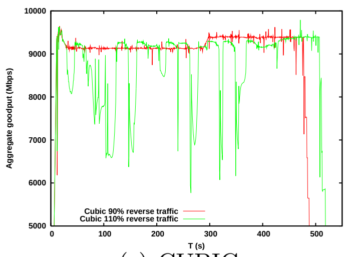
Figure 4.4: Completion time distribution for several TCP variants, 19.8 ms RTT



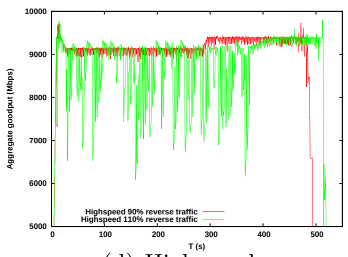
(a) Reno



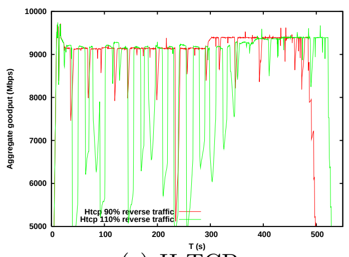
(b) BIC



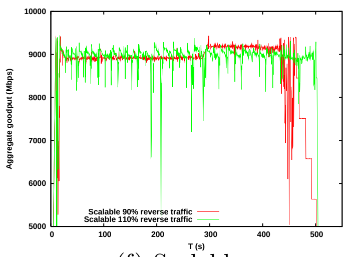
(c) CUBIC



(d) Highspeed



(e) H-TCP



(f) Scalable

Figure 4.5: Comparison of aggregate goodput for 1.8 congestion level according to the level of reverse traffic

need the full transfer of all images before the start of a computation phase. It seems that Highspeed TCP is the best choice if we are interested in good predictability, as it has the lowest dispersion of all protocols for high congestion levels.

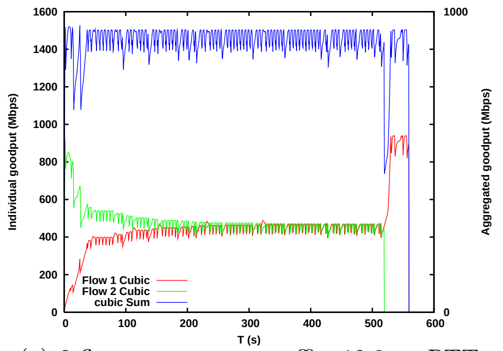
For most TCP variants, the CoV stays below 6 %. If we assume that the distribution of the completion time is indeed Gaussian, using this as a margin would provide a 68 % confidence interval for the completion of our transfers. If we want a more precise (say 95 % confidence interval), we would need to push the margin up to 12 %.

4.1.3.2 Impact of Reverse Traffic

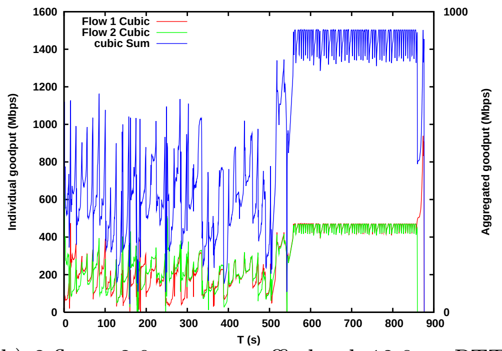
The reverse traffic here consists in similar large 30 GB file transfers, like in the forward path. The reverse transfers are started after the forward transfers with the same interval of 1 s to prevent interactions during the slow start phase just. We only consider the behavior of the transfer time for the forward path when there is reverse traffic.

General Behavior In Figure 4.5, we compare the impact of a non-congesting (0.9) and a congesting (1.1) reverse traffic level for a given congestion level (1.8) on the aggregate goodput of all the participating transfers. Here again, we can observe that most protocols present a very similar pattern. For the non-congesting case, they all present some sort of hollow during the period in which the transfers on the reverse path are active that is likely to be caused by the bandwidth taken by the ACKs from the reverse traffic (about 200 Mbps). H-TCP and Reno are the only protocols whose aggregate goodput displays some instabilities. The other are mostly stable during the period.

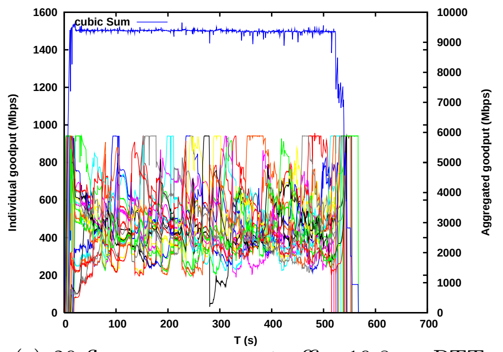
For the congesting case, we can see that the aggregate goodput is not stable at all for most protocols, and we can observe aggregate goodput drops of more than 2 Gbps that last for more than a few seconds for some protocols like CUBIC. It seems to be due to synchronization of



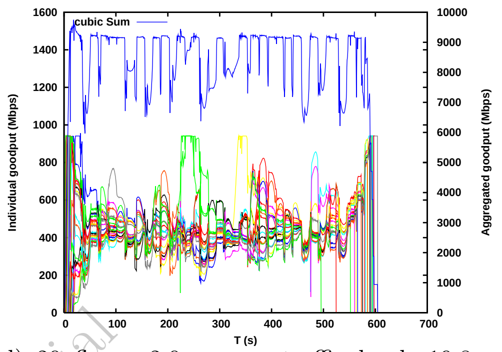
(a) 2 flows, no reverse traffic, 12.8 ms RTT



(b) 2 flows, 2.0 reverse traffic level, 12.8 ms RTT



(c) 20 flows, no reverse traffic, 19.8 ms RTT



(d) 20 flows, 2.0 reverse traffic level, 19.8 ms RTT

Figure 4.6: Impact of reverse traffic: CUBIC, 2.0 congestion level

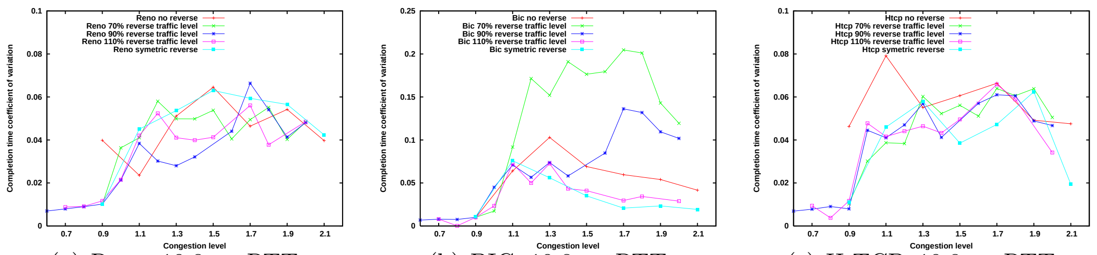
losses on the forward path. The utilized bandwidth is not at its maximum and congesting reverse traffic might lead to missed deadlines if it is not taken into account. Scalable is the protocol that seems to be the least impacted by this phenomenon, as the amplitude of the aggregate goodput spikes are less than 1 Gbps.

When looking at the effect of different levels of reverse traffic on the mean completion time for every protocol—not represented here, see [6]—, it can be observed that, for instance for CUBIC, when the reverse traffic level is less than 1.0, its effect is limited on the mean completion time (about 2.5 %). The fluctuations observed for a reverse traffic level of 0.9 are mainly due to the fact that we are close to the congestion gap and thus to a very unstable point. When the reverse traffic is congesting, we observe that the difference with the case without reverse traffic is much more important (about 10 %).

All in all, we can see that the protocols are reacting to the fact that the reverse traffic is congesting or not.

Multiplexing factor In Figures 4.6(a) and 4.6(b), we observe that the reverse traffic has a huge impact: the aggregated goodput is nearly halved in the presence of reverse traffic and the latest completion time goes from 562 s to 875 s. In this experiment, only a small number of flows (2) were used as the bottleneck size is 1 Gbps. The aggregate goodput, in blue, uses the scale on the right.

In the following experiment, as shown on Figures 4.6(c) and 4.6(d), the bottleneck size is



(a) Reno, 19.8 ms RTT (b) BIC, 19.8 ms RTT (c) H-TCP, 19.8 ms RTT

Figure 4.7: Evolution of the coefficient of variation for the completion time under different reverse traffic level

10 Gbps and we were using ten times more flows than in the previous setting. In this configuration, we observe that the multiplexing level (or number of nodes emitting simultaneously) is an important parameter. Indeed, for similar congestion and reverse traffic level, using a more important number of nodes yield better results: about 617 s (30 % faster). Even though we observe that the aggregate goodput is also deeply affected in Figure 4.6(d), its variation only amounts to up to 20 % of the available bandwidth. In the Figure 4.6(b), the variation is closer to 50 % of the available bandwidth.

Predictability Figure 4.7 presents the impact of different reverse traffic conditions on the CoV for some of the TCP variants tested in this work.

Here for most TCP variants, having reverse traffic might be a good thing as we can observe a lower CoV than in the case where there is no reverse traffic, that is to say less variability. But since the completion time is higher under reverse traffic, it is just less variable relatively to the mean completion time.

For instance, we can observe in Figure 4.7(b) that, for BIC, the cases where there is a little reverse traffic is a disaster in terms of variability, as the CoV nearly doubles. In some cases, *e.g.,* H-TCP on Figure 4.7(c), since the mean completion times for small values of reverse traffic are very close to the case without reverse traffic, the lower value for the CoV means that the standard deviation is lower too. This is a good thing if we are looking for a protocol for which we want to have predictable results. Most protocols, BIC excepted, have a CoV that remains below 6 % under most reverse traffic conditions.

So again with a reasonable margin, we could find a way for all protocols to finish within their deadlines, if we do not forget that the estimated mean completion time should be increased by about 15 % when there is congesting traffic on the reverse path. The margin depends on the exact situation that flows will face.

4.1.4 Conclusion

This work uses real experiments to examine the impact of a range of factors on transfer-delay predictability in classical bandwidth-sharing approaches proposed by high-speed TCP-like protocols. We show that when bulk data transfers start simultaneously, transfer time efficiency and predictability are strongly affected. When the congestion level is high (> 1.2) both transfer time efficiency and predictability depend on the chosen protocol. The most important factor this study reveals is the impact of reverse traffic. It strongly affects all protocols. Transfer time depends on a large number of parameters that users cannot know before starting their transfers.

4.2 L2—Unpredictability of Arbitrations

4.2.1 Motivation

Most transport protocol designers addressing wired networks do not take the behaviors of the link layer into account. They assume a complete transparency and deterministic behaviors of this layer. However, Ethernet switches are store-and-forward equipments that have limited buffering capacities to absorb congestions that can brutally occur, due to the bursty nature of TCP sources [Jiang and Dovrolis, 2005].

Ethernet switches use contention algorithms to resolve access to a shared transmission channel. These arbitration algorithms aim at limiting the amount of data that a source port may transmit to an output port per contention cycle. The amount of time dedicated to the processing of packets from one source can create starvation for another source.

The designers of these algorithms have to find a trade-off between performance and fairness in a range of traffic conditions [McKeown and Anderson, 1998]. Considering the case where huge data transfers may occur simultaneously, we explore the interactions between this L2 congestion control mechanisms and TCP's one, and try to understand how they interfere. The goal of this study is to investigate the way packets are managed in high-speed L2 equipments (switches) and the implication on transport protocols under different congestion conditions. The question is to understand how the bandwidth and losses are distributed among flows when traffic profiles correspond to huge data transfers, and ultimately how it impacts bandwidth sharing between transfers.

Section 4.2.2 details the experimental protocol adopted and the observed parameters. Section 4.2.3 presents the steady-state behaviors of packet scheduling algorithms for CBR flows and the switch's characteristic. Section 4.2.4 studies the impact on TCP flows.

4.2.2 Experiment Description

In order to observe the bandwidth sharing and loss patterns on a congested output port of a switch, a specific testbed and a restricted parametric space were used to explore 1 Gbps Ethernet port behavior.

4.2.2.1 Objectives and Test Plan

In order to characterize the switches and the impact of switches on TCP, several experiments have been performed.

The first set of experiments consists in running two contending flows at different constant rates. These two flows allow observing different behaviors of the switch under different congestion levels, to highlight the difference of behaviors of different switches using mean and variance of per-flow output rate. Fine-grained observations can also be made using sequence numbers to observe per-packet switching behaviors in the presence of two flows.

Then, as TCP tends to send packets per bursts, experiments with one CBR flow and one burst are run. The number of packets of the burst is observed as a function of the burst's length.

And lastly two different variants of TCP (BIC and Reno) are evaluated on two different switches. Performance is measured under different situations: with and without SACK—as it is designed to improve TCP performance under specific loss pattern—, and with a 0 ms and 50 ms RTT GigE link as TCP's performance are impacted by losses and high latencies.

These experiments are run the testbed described in the next section.

4.2.2.2 Testbed

The topology of the experimental testbed is described on Figure 4.8. A GtrcNET-1 is used to both generate traffic and monitor the output flows [Kodama et al., 2003]. GtrcNET-1 is an equipment made at the AIST, that allows latency emulation, bandwidth limitation, and precise per-stream bandwidth measurements in GigE networks at wire speed. GtrcNET-1 has 4 GigE interfaces (channels). Tests were performed with several switches but most of the results presented here are based on a Foundry FastIron Edge X424 and a D-Link DGS1216-T. The firmware version of the Foundry switch is 02.0.00aTe1. "Flow control" is disabled on all the used ports and the priority level is set to 0. According to the manufacturer's documentation, D-Link has 512 KB of memory. On the Foundry switch, the command line interface (`show mem`) reports 128 MB of RAM. For both switches, the way memory is shared among ports is not known.

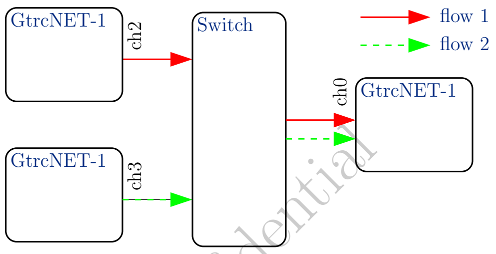


Figure 4.8: Experimental testbed

4.2.2.3 Parameter Space

We assume L2 equipments do not differentiate UDP and TCP packets. Tests that have been made corroborate this fact. Experiments were conducted with UDP flows as they can be generated easily by GtrcNET-1 and as they can be sent at a constant rate.

The following parameters were explored: input rate of flows, impact of packet length, and measurement timescale. Different levels of congestion using different flow rates were used : 800, 900, 950, and 1000 Mbps. These rates are the transmission capacity used to generate UDP packets. The transmission capacity specifies the bitrate (including Inter Frame Gaps and preamble) of an emulated Ethernet link. Experiments were strictly included in the period of packet generation. IP packet length is set to 1500 bytes as high-speed connections use full-sized packets. In order to observe the output flows' rates, the packets were counted on intervals of 400 and 1000 μs (around 33 and 83 packets at 1 Gbps).

4.2.3 L2 Switches under Heavy Congestion—Steady-state Behaviors of CBR Flows' Packets Scheduling

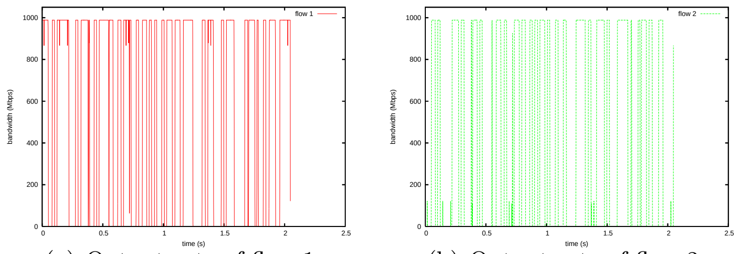
This section presents some of the flow rate patterns observed on the output port of the Foundry Fast IronEdge X424 switch when two CBR flows are sent through this output port. We only focus on 1500-byte packets as high-throughput flows use this packet size (or over with jumbo frames). Each figure of this section represents the output rate of the two flows

(Figures (a) and (b)). The sums of the output rates are always constant at 986 Mbps. The measures presented in this section are made using 1 ms intervals.

4.2.3.1 Two CBR Flows with Same Rates

The figures presented in this section use the same input rate for the two flows.

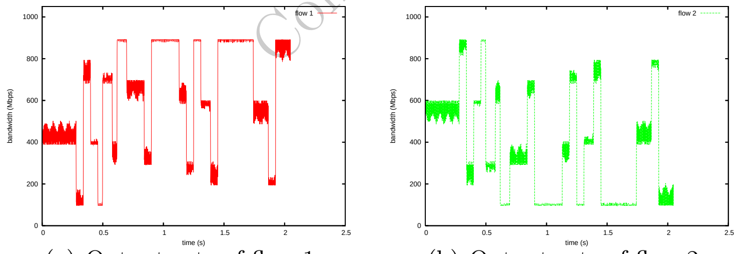
In Figure 4.9, only one flow is forwarded at a time. There are many transitions between the two flows but it seems to be completely random. It can be noticed that the aggregated rate is nearly constant and that one flow can starve for more than 100 ms (*e.g.,* flow 1 between 1.6 s and 1.7 s).



(a) Output rate of flow 1 (b) Output rate of flow 2

Figure 4.9: 1000 Mbps + 1000 Mbps (1500 bytes)

In Figure 4.10, flows do not starve but a real unfair sharing is observed for more than 300 ms. From 1.45 s to 1.75 s, one is running at more than 900 Mbps and the other at less than 50 Mbps.



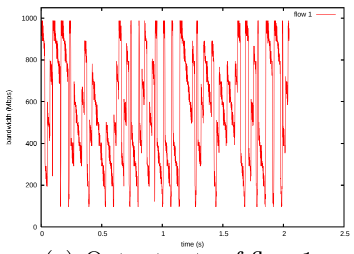
(a) Output rate of flow 1 (b) Output rate of flow 2

Figure 4.10: 900 Mbps + 900 Mbps (1500 bytes)

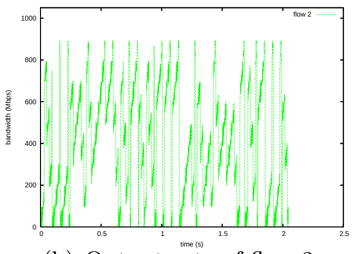
As seen in all the figures of this section, sharing among input ports can be really unfair on a “short” time scale. This can have a dramatic impact, as 100 ms may be a very long interval within TCP’s dynamic, particularly at these rates. Around 8300 1500-byte packets should have been forwarded at 1 Gbps within 100 ms.

4.2.3.2 Two CBR Flows with Different Rates

In Figure 4.11, it can be observed that when the two flows are sending at different rates (with one at wire speed), the instant flow rate on the output port varies among a set of values. When none of the flows are at 1 Gbps and they do not have the same rate, as in Figure 4.12, bandwidth sharing is closer to what we would expect to obtain, as with an observation interval of 1 ms, the throughput is nearly constant.

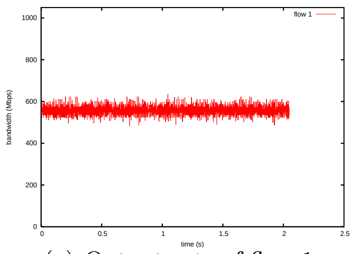


(a) Output rate of flow 1

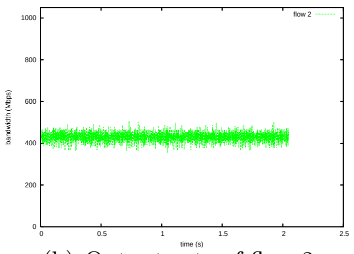


(b) Output rate of flow 2

Figure 4.11: 1000 Mbps + 900 Mbps (1500 bytes)



(a) Output rate of flow 1



(b) Output rate of flow 2

Figure 4.12: 950 Mbps + 900 Mbps (1500 bytes)

As a conclusion of this section, when the two CBR flows have the same rate or one of the flows is at the maximum rate, the behavior of this switch is unfair on a “short” time scale (100 ms) but is not when none of the two rates is at wire rate. The following sections present some quantitative measures on these situations for several switches.

4.2.3.3 Sequence Number Analysis

In this section, instead of monitoring the output rate, the sequence numbers of forwarded packets are monitored. Figures are in Appendix A. Figure A.1 shows the situation with two 1000 Mbps flows and Figure A.2 with two 400 Mbps flows on the Foundry switch. In these figures, the sequence number of transmitted packets are represented at the date it was observed on the output port.

It can be noticed on Figure A.1 that only one flow is forwarded at a time. A similar observation can be made on Figure 4.9, On Figure A.2, packets observed on the output port alternatively comes from the two flows as the zoom-in (Figure (c)) highlights.

On the D-Link switch, even when the two input rates are 1000 Mbps (Figure A.3), packets on the output port come alternatively from the two ports, but packets are alternatively dropped too. The sequence numbers of forwarded packets are growing by from 1 to 3 (Figure A.3 (c)) as the output port’s rate is limited to 1000 Mbps and some of the input’s packets have to be dropped. This is different from the Foundry switches where packets are dropped by burst. The case with 400 Mbps flows and D-Link is similar to the Foundry one. This likely means that the two tested switches have different queue-management strategies.

4.2.3.4 CBR Flows—Quantitative Measures

The following tables summarize statistical metrics for the two flows on different switches with different input rates. The first two columns indicate the input rates in Mbps for flow 1 and flow 2, the next four columns show the average, maximum, and minimum throughput (Mbps), and its variance (square Mbps) for flow 1. The next four do the same for flow 2, and the last column indicates the interval used for throughput measurements.

Measurements of the variance and the minimum of the output rates on six different switches show that for some of them, variance can be very high and the minimum throughput reaches 0 Mbps under severe congestion. This behavior has not been observed on all the switches we have tested. The first three switches tend to make one of the flows starve for periods of time longer than 100 ms when congestion is severe. These three switches also perform unfair sharing under high congestion whereas the last three always split the available bandwidth around 494 Mbps when the input rates are equals. In the case of D-Link switches, it occurs even when the input rates are different and the output port is congested.

To conclude this section, it seems switches are divided in two different classes. In the first one, under congestion, starvation of one of the flows can occur and a high variance of output rates can be experienced. In the second one, the variance is low and no starvation occurs. As TCP connections have no knowledge of which switches compose the network, it can be guessed that the behavior and performances of the connections can be highly and differently impacted, as it will be shown in Section 4.2.4.

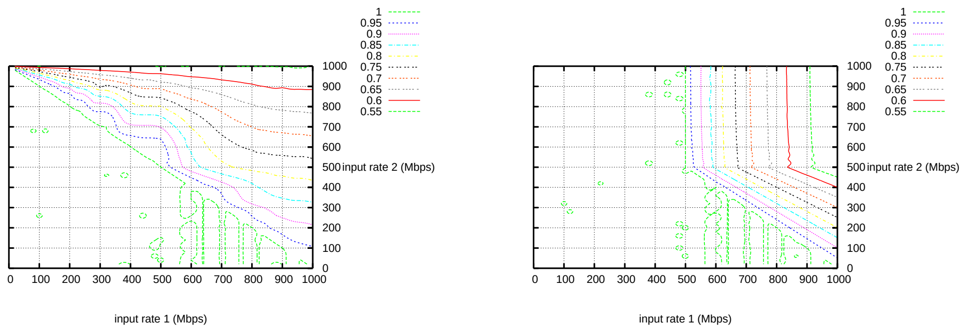
4.2.3.5 CBR Flows—Steady-state Switch's Characteristic

While the previous section showed metrics in a small number of situation for several switches, this section will present some metrics for two switches for all input rates. In order to characterize switching behaviors, the ratio of output rate divided by input rate were measured for every input rate (from 0 to 1 Gbps in steps of 20 Mbps). Variance of the flows among experiments were also measured. Each experiment lasted 12 seconds, measurements have been done on 1 ms intervals and have been repeated 3 times.

Figure 4.13(a) shows the level-set of the ratio of output rate divided by input rate of flow 1 on Foundry and D-Link switches. Behaviors for second flows are symmetric. The X axis is the input rate of the first flow and the Y axis the one of the second flow. It can be observed that the level-sets tend to join at one point—(1000,0) for flow 2. When there is no congestion (below the line joining (0, 1000) and (1000, 0)), the ratio is equal to 1. On Figure 4.13(b), the behavior on the D-Link is completely different and probably related to the packet switching algorithms used. With this switch, if the input rate of the flow 2 is less or equal to 500 Mbps, its output rate is always equal to the input rate regardless of the input rate of flow 1.

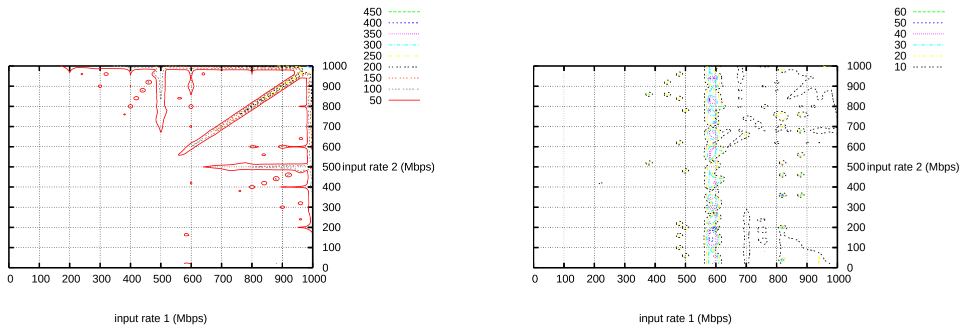
Figures 4.14(a) and 4.14(b) show the standard deviation of the output rate for flow 1 with respectively the Foundry and D-Link switches (with 1 ms measurements' interval). For the former, the standard deviation is quite low in the usual case. But when the input rates are the same or when one of them is at the maximum, more deviations can be observed. The highest standard deviation is obtained when the two flows are at 1 Gbps. That is when the alternate complete starvation of one of the flows was observed. For the latter, on Figure 4.14(b), it can be observed that even below the congestion limit, the output rate varies on this switch. However the magnitude of the standard deviation does not grow as high as with the Foundry switch.

As level-sets of output rate variances have shown, predictions on the behavior of a switch are not easy to make *a priori*. The next section will show that the differences between the



(a) Foundry (b) D-Link

Figure 4.13: Level-set of output rate over input rate for flow 1.



(a) Foundry (b) D-Link

Figure 4.14: Level-set of standard deviation of output rate of flow 1.

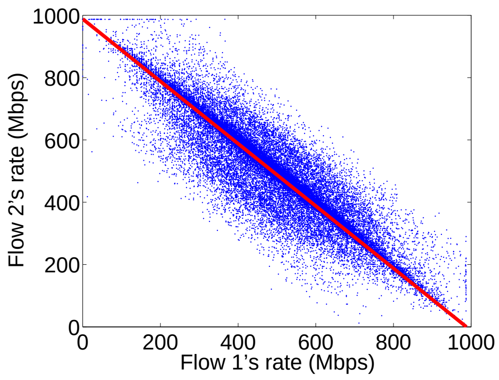


Figure 4.15: Instant rate of two flows measured before the switch using BIC with SACK (0 ms RTT)

two switches tested in this section impact TCP's performances.

4.2.4 Impact on the Transport Protocol

In the previous sections, strange behaviors of switches were observed when the congestion level was very high. As TCP uses a congestion avoidance mechanism, one can assume that this prevents the occurrence of such high congestion levels on the switches' output ports. However, in the slow start phase because the congestion window is doubling at each RTT, and during aggressive congestion-window-increase-phases as observed with BIC, flows can face severe instant congestions.

Figure 4.15 shows the instant rates measured before the switch of two BIC flows contending on the output port of a switch. This clearly shows that even when the congestion control of TCP is supposed to keep the network below congestion, switches queues can face higher instant rates than the bottleneck capacity.

4.2.4.1 Slow Start

In this section, we study the impact of L2 packet scheduling algorithms on already established flows when a new connection starts. The testbed used is similar to the one presented before but the first CBR flow generated by GtrcNET-1 was replaced by a burst of variable length generated by GNU/Linux kernel module *pktgen*. In this experiment, the rates achieved by the CBR flow and the burst were measured. We assume that the amount of bandwidth lost by the CBR flow corresponds to a number of packets lost since in a long-run situation, the switch can not buffer all the packets. Figure 4.16 represents the estimated number of losses that the first flow experienced as a function of the length of the burst with different switches. It can be seen that generally the burst get most of his packets going through the output port, which causes a large dent on the CBR flow. But again two different behaviors can be observed. Figure 4.16(b) shows very regular lines for the DELL, D-Link, and Huawei switches whereas they are very noisy for the Cisco and Foundry switches (Figure 4.16(a)) which might indicate these switches use more sophisticated algorithms.

As switches differently drop packets and it impacts TCP, the next section will compare two TCP variants (BIC and Reno) under two latencies (0 ms RTT and 50 ms RTT), with and without SACK on D-Link and Foundry switches.

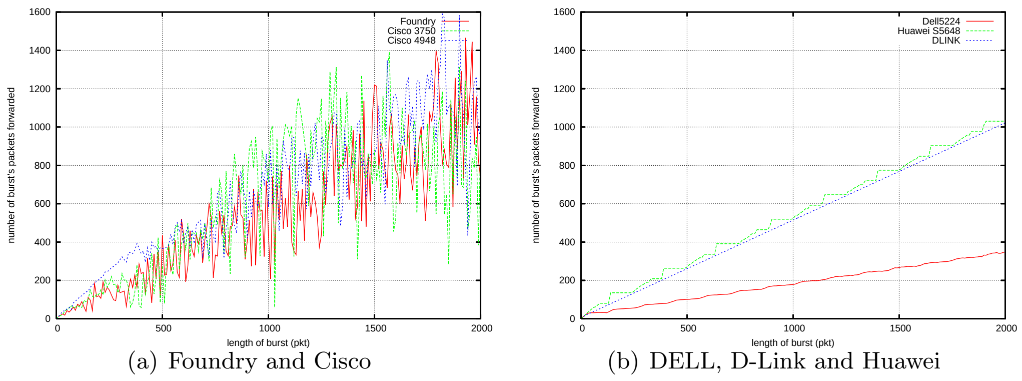


Figure 4.16: 1000 Mbps CBR flow's losses

4.2.4.2 Comparison of Transport Protocols on Different Switches

The experiments presented in this section use four hosts: two senders and two receivers, all running iperf. The two flows involved share a 1 GbE link of configurable RTT: 0 ms or 50 ms. Bottleneck takes place in the switch before this link. We observe the two flows on this link using the GtrcNET-1 box. All the experiments share the same experimental protocol: the first flow is started for 400 s, and 20 s later, the second flow is started for 380 s.

In these experiments, TCP buffers where set to 25 MB and `txqueuelen` to 5000 packets to avoid software limitation on end hosts.

First observation: the first flow manages to fill the link by itself in all situations except for Reno with 50 ms RTT on the D-Link switch where a loss occurs during the first seconds.

On the figures presented in Appendix A flows' throughputs on a 0 ms RTT GigE link are represented. Some figures present periods of time where one of the flows is nearly silent. We can observe such behaviors on Figure A.4(d) between seconds 320 and 340, on Figure A.4(k) between seconds 30 and 50, and finally between seconds 150 and 175. We did not observe such starvation on the D-Link switch nor with Reno TCP. More figures are available in [25]: BIC and RENO on Foundry and D-Link switches under 0 and 50 ms of RTT.

Comparison clearly shows the interest of using SACK as it "smooths" the bandwidth usage of the two flows: no more stop and go when packets are lost in the switch. The absence of SACK is the worst case for the Foundry switch, as in this situation we can observe in Figure A.4(d) and A.4(e) that only one flow is forwarded at a time. It can be noticed that TCP's default configuration in the current GNU/Linux kernel (2.6.18) is BIC with SACK.

Quantitative Comparison In this section, we summarize the figures of the previous section. Data presented in this section have been collected using the Web100 Linux kernel patch. Table 4.1 represents the mean goodput in Mbps of each flow in the different experiments and table 4.2 represents the number of retransmissions per second.

In Table 4.1, we can observe that the mean goodput is higher with the Foundry switch than with the D-Link one. We can also notice the highest aggregated goodput with 0 ms RTT is obtained on the Foundry switch with Reno and SACK, whereas the best results are obtained on the D-Link with BIC and SACK. With 50 ms both best results are obtained with BIC and SACK. Table 4.2 highlights the fact that there is more retransmission with the D-Link switch than with the Foundry one with 0 ms RTT, but the situation is the opposite with 50 ms RTT.

Figures of previous sections and tables of this sections have shown that the behaviors and performance of TCP variants on different switches can vary dramatically.

TCP variant	Sack?	Switch	Mean goodputs	
			0 ms RTT	50 ms RTT
BIC	Yes	Foundry	417 & 543	372 & 413
		D-Link	468 & 491	168 & 238
	No	Foundry	408 & 471	192 & 253
		D-Link	361 & 394	118 & 178
Reno	Yes	Foundry	461 & 505	345 & 434
		D-Link	434 & 461	141 & 201
	No	Foundry	433 & 475	254 & 535
		D-Link	364 & 407	154 & 198

Table 4.1: Mean goodputs of 2 flows sharing one port for 400 s (Mbps)

TCP variant	Sack?	Switch	Retransmission per seconds	
			0 ms RTT	50 ms RTT
BIC	Yes	Foundry	58.14 & 60.01	14.27 & 15.83
		D-Link	206.75 & 227.22	2.76 & 4.25
	No	Foundry	41.17 & 53.39	6.28 & 7.67
		D-Link	447.68 & 493.23	3.91 & 4.25
Reno	Yes	Foundry	47.65 & 50.02	0.34 & 3.25
		D-Link	183.84 & 192.32	0.17 & 0.59
	No	Foundry	45.10 & 46.78	0.39 & 0.46
		D-Link	100.58 & 89.32	0.10 & 0.12

Table 4.2: Number of retransmissions per second for 2 flows sharing one port for 400 s (pkt/s)

4.2.5 Conclusion

In the present work, switches are evaluated in an extreme situation which is likely not to be the one for which algorithms have been optimized, since only three ports are used—two input ports and one output port—with a high congestion level. Packet-scheduling algorithms for Ethernet equipments have been designed for heterogeneous traffic and highly multiplexed environments. Nowadays, Ethernet switches are also used in situations where these assumptions can be incorrect, such as environments where the multiplexing level is small. As sizes of flows and access rates tend to increase, this is more and more frequent.

While IP is the common layer of the Internet, Ethernet is used in almost all local networks and tends to extend to metropolitan networks and long-distance networks through layer 1 or layer 2 VPNs. For example, interconnected networks often use Ethernet over DWDM for long-distance site interconnection. In this situation, congestions between flows take place in Ethernet switches. We can face congestion on a long-latency link or local links depending on which nodes are talking together. Literature on IP router buffer sizing assumes that queues are drop-tail FIFO [Kantawala and Turner, 2005].

This work shows several conditions in which the arbitration mechanisms of switches introduce heavy unfairness (or starvation) on large intervals (300 ms) and bursts of losses which impact TCP performance. It also shows that behaviors are different from switch to switch, and not easily predictable under congestion. These observations offer some clues to better understand layer interactions.

We have also seen that the rates of two contending TCP flows that share a 1 Gbps output port, when observed at the input ports, can have their sum exceeding the output port's capacity, and hence create temporary congestions. This occurs because the flows only coordinate through the loss events in the shared queue and by the means of the feedback loop of TCP's congestion control algorithm, which can be quite long.

In the case of low multiplexing level, even if the number of flows is known, the performance achieved by a flow can vary significantly. It is then important to control the congestion at all the time scales: at a large time scale to know how many flows are contending, if there is reverse traffic, and to have an idea of the margin variability of performance; at small time scales, since the interactions between layers can add noise to the achieved performance. When contention is avoided, we observed that switches behave in a very similar way at all time scales.

Summary

From these two studies, we conclude that the current layered sharing mechanisms used in packet networks such as the Internet do not provide predictability of transfer time.

First because the sharing principle, which tries to be "fair" on the flow rate allocation, makes the obtained rate dependent on the number of participants to the congestion at a bottleneck.

Then, because performance also depends on various parameters such as the presence of reverse traffic, the variant of congestion control used, or the inter-layer interactions when there is congestion at short time scale and the multiplexing is low.

Since users have not told their expectations to any entity of the network, the allocation cannot give satisfaction to their specific needs. We propose to explore a large data transfer scheduling approach which enables controlling the starting time and congestion level in the forward and reverse paths, based on the capacity of the network and the explicitly requested network services: bulk data transfer service. Such an approach would help in determining in advance the amount of resources requested by the transfers and would, in turn, provide transfer time predictability even if the number of contending flows is not known to the users—but is to the network operator.

Such a service, combined with an adaptable and very responsive protocol which can fully exploit a dynamic and high capacity, could be a solution to provide good transfer time predictability to high-end applications.

Q1 How does the current approach behave regarding transfer completion time?

A1 *The transfer completion times are not guaranteed and predictable as soon as there is contention between flows. First fair sharing of bandwidth makes the flow rate and thus the completion time dependent on the number of flows, then when there is congestion, completion time presents an important deviation, and performances that depends on the* *exact configuration. Finally, L2 switches have different behaviors under severe congestion and the different losses pattern affect the performance of TCP flows.*

— Five —

An Alternative Bandwidth Sharing Approach Based on Services

Introduction

The previous chapter showed that the completion time of data transfer is not easy to predict even for someone with the knowledge of the number of contending flows on the path. In this section, we present the sharing approaches adopted for network resources, with their objectives and proposed enhancements to increase user satisfaction. Then we present a sharing-approach proposal based on network services with explicit requests of utility so that, for example, users can explicitly request resources to complete their transfer within a strict time window, or request a dedicated channel with a specific bandwidth available.

The first section presents the allocation mechanisms that have been proposed in the network through the lens of optimization problems and utility. In the second section, we present our proposal of allocation mechanisms in this framework and highlight its constraints, objectives, and specificities.

5.1 Allocation in Networks

In the next two sections, we survey sharing and allocation approaches proposed for the Internet and transport networks: from the points of view of, first, users, and then, network operators.

5.1.1 Benefit or Cost Sharing

The Internet and networks in general are economic entities with specific properties. As a good, depending on the purpose, the network can be viewed as either rival—the bandwidth consumed by one user can not be by another one—or nonrival—if it is about potential connections and connectivity. Similarly, depending on the point of view, it can be considered as a non-excludable good, if we consider that network neutrality really applies and anyone can connect to anyone. Alternatively, it can be considered as an excludable good: most of the time, to be connected, one has to pay his ISP, which in turn has to pay transit providers. The sharing of bandwidth in this context is not easy as the social objective of this is not well-defined.

5.1.1.1 Resource Allocation in Networks—the Network as a Good

The value of communication networks as economic entities is characterized by two externalities. The first one, Metcalfe's law—also known as the network effect—is a positive externality and increases the value to the network [Carpenter, 1996]. In short, it says that a component of the value of a network is the number of possible connections $n(n-1)/2$ in a network of n nodes—even if the actual asymptotic behavior is discussed as in [Briscoe, 2006], which proposes instead $n\log(n)$. The point is that it is more than linear and thus makes it very valuable to have a lot of participants. The second one, congestion, is a negative externality due to the finite capacity of the network.

The first externality led to drop-circuit-centric approaches such as ATM, because best-effort traffic (through ABR) was difficult to implement instead of being the default choice in the Internet and IP. This externality is also a motivation for the adoption of common standards and the interconnection of networks, since it increase the number of nodes and thus the value of each network initially not compatible or not interconnected [Majumdar et al., 2002].

Regarding the second externality, since congestion decreases the value of the network, it has been the main concern of sharing since the expansion of the Internet. Besides the constant effort to try to over-provision the network, this had two implications. One has been to prevent severe congestion/collapse through the introduction of congestion control in TCP. The second one, to try to allocate the resources in a "*fair*" manner. Some tried to have a "fair" allocation of benefits, while others tried to have a "fair" sharing of costs.

Let us start with an overview of the resource allocation problem and its evaluation.

5.1.1.2 Aggregate Utility—Welfare and Fairness

Consider a resource available in a quantity C that has to be shared between i users. Each gets x_i and the allocation is $(\ldots, x_i, \ldots)$. User i receives utility u_i of this allocation.

The aggregation of individual utilities and fair sharing is a challenging question as the way they are aggregated changes the definition of fairness [Denda et al., 2000]. Let us first consider the different feasible allocations. The *pareto-efficiency* qualifies an allocation, and more specifically defines its property of being feasible and non-dominated. Feasible means that the sum of the x_i is less than C, and non-dominated means that it is not possible to increase the utility of one i without decreasing the one of another. The sum of individual utilities can easily be improved when the allocation is not pareto-efficient.

There are usually many pareto-efficient allocations. The purpose of the *welfare function* w is to order them. Its purpose is to aggregate individual utilities in a way that justifies selecting the allocation which maximizes the welfare $w(\ldots, u_i, \ldots)$ under the capacity constraint. This measures the social utility. The welfare function is supposed to be increasing every u_i and under this assumption, allocations that maximize welfare are pareto-efficient [Varian, 1992].

Depending on the choice of w, different allocations will be preferred when taking the one maximizing it. This is closely related to the concepts of *fairness*: max-min fairness, proportional fairness, and *alpha*-fairness. Each yields to a definition of w. For example, with max-min fairness, $w(\ldots, u_i, \ldots)$ is $\min_i\{u_i\}$.

While deciding which allocation is better suited, in addition to defining what is the utility of each user, the concept of welfare is not uniquely defined and the choice of the fairness concept may lead to the selection of different allocations.

5.1.1.3 The Current Internet—Network Utility Maximization

Lets look at the “fair” sharing of benefit. The current layered design of Internet has been reverse engineered and modeled as a distributed optimization process solving the so-called *Network Utility Maximization problem* (NUM) [Chiang et al., 2007], [Srikant, 2004, Ch. 3], [Mo and Walrand, 2000]:

$$NUM: \quad \text{maximize} \quad w(\ldots, u_i, \ldots) \\ s.t. \\ Rx \leq c$$

R being the routing matrix and c the capacity vector.

Depending on the variant of the congestion control algorithm used, asymptotic behavior tends to a different welfare function. FAST, TCP Vegas, and Scalable TCP have been proved to solve NUM with a proportionally fair welfare function w as a weighted sum of u_i and utilities as $u_i(x_i) = \log(x_i)$.

This is based on the assumption that all users have the same utility function of the allocation of resource. This is probably fine for the early Internet applications, such as web or mail, but it is wrong for the coordinated utilization of different resources. In this case, the utility is not defined by always trying to get more, but by having enough for a given purpose.

5.1.1.4 Other Approaches—Differentiation of Individual Utilities

The different applications that co-exist in the Internet have different needs regarding the network. Some applications such as video-streaming need a constant rate, while conferencing also needs low delay and low jitter. A model for data transfer has been presented as *elastic traffic* [Shenker, 1995, Roberts, 2004]. This model considers: (i) that the application can adapt to the available bandwidth, and (ii) the utility function is an increasing, strictly concave, and continuously differentiable function of the allocated rate.

Integrated Services (IntServ) has been proposed to allow users to signal their needs through reservation messages describing what the traffic is supposed to look like and what is requested at each router of the path. This was also designed to provide fine-grained QoS at the packet level. It proposes two types of service: Guaranteed Quality of Service and Controlled-Load. Regarding the Controlled-Load service, according to its IETF’s RFC [Wroclawski, 1997]:

> “ The intention of this service specification is that network elements deliver a level of service closely approximating best-effort service under unloaded conditions. ”

Finally, best-effort traffic is also available in this approach. But the large number of states to be processed and stored by network equipments led this proposal not to be deployed at the epoch it was proposed (1997).

To address the different kinds of utility that users/flows can have, and in order to provide packet-level quality differentiation, DiffServ has been proposed. In this case, the requested resources are not expressed in terms of bandwidth but in terms of the service policy applied at routers: *Per-Hop Behaviors* (PHB); however, bandwidth resources are still limited, as is the traffic that can be accepted for a given PHB. Packets experience a loss probability and a queueing delay that depend on the class of utility they belong to and the total amount of traffic in that class. Several PHB have been defined: *Expedited Forwarding* (EF), *Assured Forwarding* (AF), and best-effort. EF claims to be non-queueing, and thus with low latency and jitter. However, this requires the amount of traffic using this PHB to be small compared to the other: admission control is required. This PHB is suitable for video streaming among other things. AF proposes different service-rate/drop-precedence pairs. Finally, based on these PHB, services can be defined and proposed to users: *premium*, *assured*, *best-effort*.

While IntServ proposal was a fine-grained approach defining packet-level behaviors for packet processing at routers, DiffServ manages aggregates. *Flow-aware networking* lies in-between [Roberts and Oueslati-Boulahia, 2000]. It is based on the observation that the QoS is good as long as the network is over-provisioned. The admission control and routing decisions must consider each flow and grant it if it is known not to create congestion.

5.1.1.5 Cost Sharing

While, in the previous sections, the objective was to share the benefit through the utility functions, some authors, such as B. Briscoe or F. P. Kelly, advocate to have mechanisms that share the cost of congestion [Briscoe, 2007, Kelly et al., 1998].

B. Briscoe claims that *flow-rate fairness* is not justified but *cost fairness* is, since: (i) "user benefit per bit rate might be different by ten orders of magnitude different for different types of flows (*e.g.,* SMS and video)"; (ii) since the congestion that one can create is paid by all, it has to be limited/charged: "if the network becomes congested, each user restricts every other user, which can be interpreted as a cost to all—an externality in economic terms".

Following [MacKie-Mason and Varian, 1995] and the Pigouvian taxes used to reduce the impact of selfishness in routing games [Nisan et al., 2007, Chap. 18], the cost of a flow has to take into account the congestion it creates through the "marginal cost of the capacity needed to alleviate the congestion". The marginal cost is the slope of the cost function against the capacity.

The next section looks at the allocation problem from the point of view of the network operator.

5.1.2 Traffic Engineering

On the network operator's side, decisions regarding resource allocation are made in the management plane: this is known as *Traffic Engineering*.

Traffic engineering refers to the activities conducted by network operators to manage their transport networks in terms of allocation of resources to specific traffic. This encompass routing of new traffic/demands, QoS routing, capacity planning, survivability and recovery of the network [Resende and Pardalos, 2006, Vasseur et al., 2004, Ash, 2006, Masip-Bruin et al., 2006, Bertsekas, 1998].

Flows refer to identified demand—source, destination, volume, constraints—of the traffic. Flow allocation has a long history, starting from the works of Ford & Fulkerson in 1956. The basic idea of the formulation of the minimum cost flow problem is to assign variables that give the *rate* of the flow for every *demand*, on every particular *link*. Depending on the kind of problem, links can have a capacity, and the demands can be specified or constrained by total volumes and a set of paths. Using these variables and parameters and flow conservation at the nodes, depending on the set of constraints and the objective function, the optimization problem can be written as a rational or integer linear problem, a convex or a

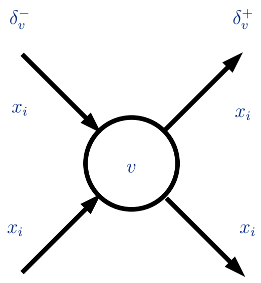


Figure 5.1: Link-Node notations.

combinatorial optimization problem. This model and many variations are described in details in [Pióro and Medhi, 2004].

The resulting optimization problem has the following form:

$$\begin{aligned} TE: \quad & \text{minimize} && C(\ldots, x_i^d, \ldots) \\ & s.t. && \\ \text{conservation laws:} \quad & && \forall d, \forall v \sum_{i\in\delta^-(v)} x_i^d = \sum_{i\in\delta^+(v)} x_i^d \\ \text{demand constraints:} \quad & && \forall d, u^d(\ldots, x_i^d, \ldots) \in U^d \\ \text{network constraints:} \quad & && (\ldots, x_i^d, \ldots) \in X \end{aligned}$$

where d represents the demands, v the nodes, i the links and $\delta^-(v)$ and $\delta^+(v)$ respectively the incoming and outgoing links connected to node v, as depicted on Figure 5.1. $u^d(.)$ maps the allocations to the constraint space U^d as defined by the users in their request for d. X gives the feasible allocation from the network's point of view. For example, if only a discrete rate can be used as in the lambda switching paradigm, X can be a subspace of the Cartesian product of the set of rates available on each links. Note that X can also contain the capacity constraints or wavelength-continuity constraints [Kuri et al., 2003, Ozdaglar and Bertsekas, 2003, Banerjee and Mukherjee, 2000, Gençata and Mukherjee, 2003].

5.1.3 Conclusion

The value of the network depends on two aspects: (i) the number of potential connections that can be seen as traffic without much more constraints than delivery of data—best-effort traffic ; (ii) the ability to serve traffic with more specific QoS constraints, such as allocated rate, delay, loss rate... or higher-order QoS attributes such as transfer completion time.

As seen in the previous section, the social objective of network resource allocation is not clear for two reasons: first, individual utility is difficult to model and depends on the user's purpose, and second, the aggregation of these utilities into a social utility is also problematic. As observed by Condorcet's paradox, aggregation of non-cyclic individual preferences (which can be ordered using the utility of the allocations and the natural order) can lead to cyclic ordering. This is generalized by Arrow's impossibility theorem [Arrow, 1950], which states that it is not possible to decide a social welfare function—the social welfare function gives the social preference as a function of individual preferences—that satisfies the properties listed thereafter, if the system has at least two users and three options. The properties mentioned are: non-dictatorship (not the choice of only one), universality (all preferences of all voters are taken into account to define a unique social choice), independence of irrelevant alternatives (must be stable on subsets), pareto efficiency (if an allocation is preferred by all, it must be so in the social welfare function). Aggregating preferences is thus a real problem.

While the currently used sharing mechanisms consider that all users have the same utility functions, several propositions to differentiate user utilities, and improve the QoS they experience, have been proposed [Firoiu et al., 2002]. These approaches do not take time into account by proposing in-advance reservation, and the requested services are strict. This makes the approaches not interesting from a network operator's point of view, since he does not get any additional flexibility but instead has more constraints concerning resource allocation.

In the next section, we present a service-based[1] approach that allows users to give their utility—and thus constraints—regarding an allocation. Given this, the network operator can perform the allocation fulfilling the requirements, but exploiting the offered flexibility to minimize its cost. This enables the different agents of the network's eco-system to benefit from these QoS approaches. Using strict priority, this approach also permits using the remaining resources for best-effort IP traffic maintaining the network effect.

In the absence of such a service, users and developers, such as Skype or peer-to-peer networks, have built their own QoS solutions, based on overlay networks where sharing, routing decisions, and data forwarding are all done by the users—actually by the software—at the application level [Andersen et al., 2001, Subramanian et al., 2004, Lee et al., 2008]. Since network operators have the infrastructure and control the equipments, QoS building blocks could be a valuable service for the emergence of new high-quality applications.

5.2 In-advance Explicit Requests, Delegation of Utility and Allocation

From the previous section, we know that the environment is made of users and network operators. Users can value QoS, but they also value connectivity and best-effort traffic. However, without advance notification or flexibility in the requested service, network operators do not have many incentives to support resource reservation.

5.2.1 Problem Formulation & Proposal

Users get different utility from their resources allocation, depending on the purpose these resources have. It can justify explicit reservation if they require something different from the default best-effort traffic.

On the other side, the network provider has different constraints depending on the technologies he uses, previously allocated resources or on the goal he wants to achieve, which is supposed to be its users satisfaction.

The core of our proposition relies on explicit requests made by users. Requests specify the sets of allocations that provide them a non-zero utility. A step-function allocations is decided by the network operator on a first-come,first-served basis to incite users to submit in-advance. This would give the network operators some slackness regarding the allocation, and allow them to possibly change the provisioning of their networks, if the time advance is large enough.

[1]In the economics acception of *service*: "a set of singular and perishable benefits"

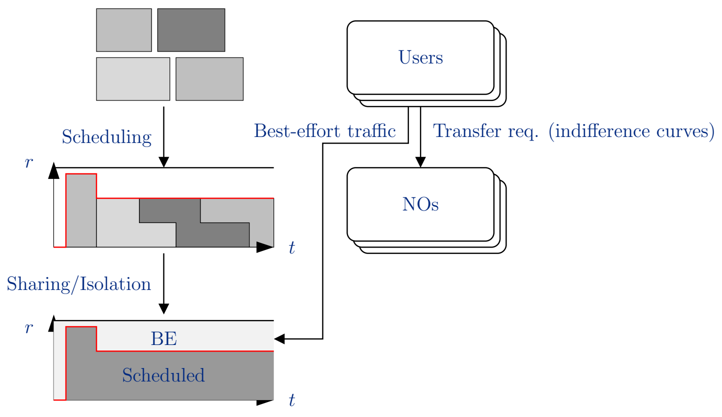


Figure 5.2: Scheme of the proposition: Users can either submit requests or use best-effort, the Network Operator schedules the requests and know the large demands in-advance.

From the perspective of the bandwidth-delivery classification proposed in Section 3.4, this solution can be categorized as fitting in the overlay model since users describe and submit their needs while the operators take the decisions. By exploiting the fluidity of packets switching, this solution has the advantage of satisfying the users while giving flexibility to the network. Indeed, it will know large demands in advance, and can allocate them in a way that reduces its cost and gives the remaining bandwidth to best-effort traffic for which, by definition, no requests are sent. Figure 5.2 illustrates this.

The next section will present this process in an optimization framework, before both the user's and the network operator's parts are presented in details. Then, related works are presented.

5.2.2 Framework

Starting from the optimization problem presented in Section 5.1.2, and considering that demands are constrained by user requests which consider time, by exploiting fluidity of packet flows, the optimization problem becomes NO_OPT. In this problem, the x_i^d (or $x_{i,j}^d$) represent rates. We will come back on this while considering the control of the sending rate to conform to the allocation in Chapter 7.

As long as the cost is positive, non-decreasing, and convex, this problem structure remains even when time is introduced. This includes linear cost. Using a scheme similar to the one used in the proof of Prop. 9.1.12 and 9.1.13, we can show that optimal allocations are step functions with steps defined on the intervals obtained by dividing the time axis with dates provided by the requests (start times, deadlines). Then the x_i^d become $x_{i,j}^d$ where j is an index on time intervals, conservation laws apply per-intervals and the optimization problem is:

$$
\begin{aligned}
NO\_OPT: \quad \text{minimize} \quad & C(\dots, x^d_{i,j}, \dots) \\
\text{s.t.} \quad & \\
\text{conservation laws:} \quad & \forall d, \forall v, \forall j, \sum_{i \in \delta^-(v)} x^d_{i,j} = \sum_{i \in \delta^+(v)} x^d_{i,j} \\
\text{user constraints:} \quad & \forall d, u^d(\dots, x^d_{i,j}, \dots) \in U^d \\
\text{network constraints:} \quad & (\dots, x^d_{i,j}, \dots) \in X
\end{aligned}
$$

Another way to present this is the time-expanded network structure proposed by Ford & Fulkerson [Schrijver, 2003, Chap. 12] or [Hall et al., 2007] and used in Section 9.2.1. Finally, for a request d, the rate profiles on each links are defined by the matrix:

$$\left[x^d_{i,j}\right]_{(i,j)\in\text{Links}\times\text{Intervals}}.$$

Description of the demands and the resulting constraints define the user part of the problem—what they have requested and expect to get—while conservation law, objective and possibly extra technology-related constraints depends on the network operator as he will decide the allocation.

Our proposition is based on the submission by users of a description of which allocations can fulfill their needs. That is, attributes given in the requests are hard constraints and have to be completely fulfilled by the allocation. This will be detailed in next section.

5.2.3 User Part—Requests

Different kind of requests have been considered throughout the literature to express users constraints. We can distinguish several parts in their specifications: endpoints, time constraints in a broad sense, amount of resource requested, constraints on the resources that can be used to serve the request. From an economic point of view, requests describe the utility that users will have from their allocation.

An allocation out of the support of the utility function is useless from an utility-maximization point of view since the utility is null. In the general case, utility functions can vary smoothly with the allocation's parameters, or in a stepwise manner when the allocation is hard constrained.

It is quite difficult for a user and an operator to agree on a common scale to compare utility values and evaluate how a given allocation is useful for the user. To overcome this, we consider binary utility functions with utility equal to either 0 or 1. In this case the meaning of the utility function is straightforward and just describes if the allocation can be used or not to satisfy the user. The boundaries specified by this model are hard constraints. In this case, the non-null indifference curve of the utility function—where it is equal to 1—is used as the request. It is the set of point where the utility function is 1.

In Figure 5.3, the red bold line represents the set of allocations of interest to the user for a transfer of volume v on two links. x_1 and x_2 give the volumes sent on each links. c_1 and c_2 are the maximum volume the client is able to send during the time interval considered—because of disk i/o constraint for example.

If the allocation is not on this line, it is of no use for the client and the utility of the allocation is null. Actually, if the allocation is above this line, the user can do his transfer but

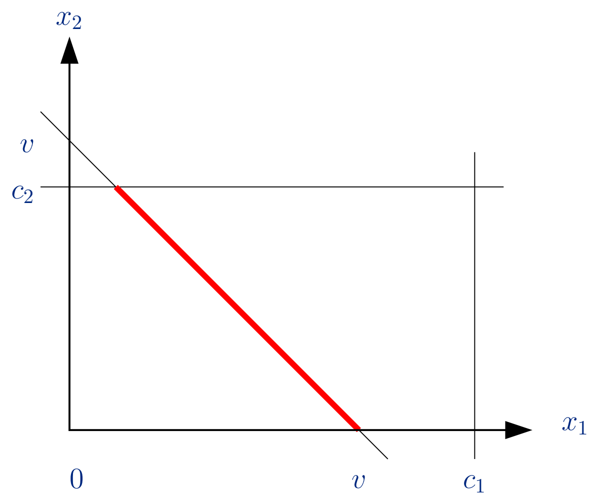


Figure 5.3: Example of binary utility.

the allocation is too large and thus not justified. Since allocating extra bandwidth is costly for the network and useless for the user, or at least not useful enough for him to say it is required, this kind of over-allocation is not considered as being of interest; it is a waste of resources. In NO_OPT, the previously mentioned region can be expressed as $u^d(x_1, x_2) = (x_1, x_2)$ and $U^d = \{(x, y)|x + y = v \wedge x \leq c_1 \wedge y \leq c_2\}$. For other metrics such as end-to-end delay, the region where the utility is not zero can be larger and only be limited by an upper-bound on the delay.

These constraints derived from the utility are expressed using state variables describing the allocation.

Depending on the purpose, users can express their need of resources through different means.

They can specify the bandwidth (throughput/goodput) they want, or a minimum and maximum value, both minimum and maximum profiles of throughput if the need varies with time. It can also be expressed as a property of the allocated profile on each link, such as using the volume the allocated resources permit to transfer between the source and destination: $\int_t \min_{l \in \text{path}} p_l(t)$ where $p_l(t)$ gives the rate reserved on link l at time t.

The next sections will survey what can be taken as a request parameter and describe different kinds of request.

5.2.3.1 Endpoints

First, a request can be directional or not. In the first case, there is a source and a destination which can both refer to a single element, or to a set. In the second case, there still are two endpoints—or sets—but no privileged direction.

If it is a single endpoint, it can be fully specified, for example by an IP address, or loosely specified, as being a member of a group—this is the case of the destination in an anycast request. In multicast, the destinations are all the members of a set.

Once constraints on the endpoints have been specified, time constraints need to be.

5.2.3.2 Time Constraints

Time constraints can qualify the reservation process, or the execution phase. For example, in the first case, it can specify that the decision of the admission control must be made available to the client before a given date, or the decision regarding the exact resources that will be allocated to serve the client can have to be definitively fixed by a another given date.

If the time constraint applies to the allocation itself, it can specify a time window for the allocation which can possibly start now. A request for such an allocation is usually called *immediate*, by opposition to *in-advance* requests which have to be served at a later time. A *deadline* can specify the time by which the requests have to be served.

Now we go to the specification of the desired allocation.

5.2.3.3 Constraints on Path Selection

First, the request can contain more specifications than source/destination constraining the resources that can be used. It can also be technology constraints, *e.g.,* only use Ethernet links. It can also be classical link/path metrics such as delay, loss rate, or number of hops. These can be classified in three classes depending on how they are computed for a path, based on the metric of its constitutive links: additive, multiplicative, and convex [Griffin and Sobrinho, 2005, Paul and Raghavan, 2002]. Other pruning constraints can specify which links must not be used to serve the request for privacy/political concerns or for path diversity with resiliency as goal [Manayya KB, 2009, Medhi and Ramasamy, 2007, Kuipers and Van Mieghem, 2005, Hershberger et al., 2007]. As an example, the non-zero indifference curve of a binary utility specifying the usefulness of the allocations can require to upper-bound the delay by giving the maximum acceptable value.

5.2.3.4 Conclusion

The transfer of utility through requests defining a non-zero indifference curve of users' binary utility functions has the advantage of giving network operators some flexibility in the choice of allocation while not decreasing the utilities, and thus allows meeting hard constraints such as deadlines.

Depending on the request and the associated constraints, the allocation problem can have different forms. For example, the user's utility function might give a single point in the U^d. Then, depending on the constraints associated with the network, there can still be flexibility (*e.g.,* choice of the path) or not. In this case, since there is no flexibility given by the user, from his point of view, the problem is an admission control. But from the operator's point of view, he might still want to minimize the cost of this allocation. This cost will be detailed in the next section.

It can be observed that the optimization problem can consider several requests at the same time. If the instance has a solution, there is no problem. If there is no solution, determining which subset of requests can be accepted to maximize the operator's profit/utility is in general NP-hard, like a variety of bin packing problem [Marchal et al., 2005].

We next look at the network side of the problem.

5.2.4 Network Operator Part—Model

As already mentioned, the part of the optimization problem on the network operator's side includes both the objective of the allocation problem and the network model.

The network operator can consider different objective functions. It can also take into account the value of the network. In this case, the utility of users reserving resources is supposed to be satisfied as long as the request is accepted. Thus, when accepting requests, the objective can be to make this allocation as best-effort-friendly as possible, so as to also take into account this part of the network's value.

In case the network is not static and can be dynamically provisioned, this objective function can also take the provisioning cost into account. These objective functions will be discussed in Chapters 6 and 8.

Regarding the constraints, the exact formulation of the problem depends on whether the resulting allocation is permitted to be bifurcated or not—if many parallel paths can be used at the same time or not—, to use discrete or rational rates, to use a single step or several when time is involved. This will also be discussed in the aforementioned chapters.

Even when the optimization problem has a solution, it does not preclude the network operator from rejecting the request if it is not "profitable". This is proposed in [Awerbuch et al., 1993] for on-line allocations—when requests come one after the other, future arrivals are not known yet, and previously allocated requests can not be rerouted. The optimal strategy in terms of profit is known as ROUTE_OR_BLOCK. It accepts a request if the cost which depends on the load of the links is not too high and the request is profitable. But since it breaks the first-come-first-served incentive, this scheme is not used. If it were, users would not have a strong incentive to send their requests as much in advance as they can, which would in turn mitigate the network operator's benefit by reducing their advance knowledge of what is going to be used.

Studying the exact long-term behaviors of users under these different admission policies is part of the future works.

Quantifying the inefficiency induced by the selfishness of users can be studied using the concept of price of anarchy in Wardrop-like games. While this might not be an answer to the question of long-term user behavior, it gives an insight about the potential benefit for users.

In Chapter 9 using a model of routing games extended to time, we show that under some assumptions, explicit requests scheduled by a centralized coordinator can help reducing the social cost of all the allocations. This makes it interesting for users to coordinate through submission, and makes the existence of the coordinator sustainable.

5.2.5 Conclusion

Besides the well-known problem of network flow [Cormen et al., 2001, Ch. 26], Ford and Fulkerson also considered dynamic flows that take time into account. These problems of dynamic flows have also been studied recently in [Hall et al., 2007].

A dynamic bandwidth-scheduling scheme which exploits the quasi-flexible nature of connectivity reservations as we do has been proposed in [Naiksatam and Figueira, 2007]. This problem was also studied by Burchard et al. in [Burchard, 2005], where the concepts of malleable reservations to address bandwidth fragmentation was proposed. Our approach is similar to both proposals. However unlike these related works, our work formulates the problem with another objective function (congestion factor) which offers a tractable solution.

From the user's point of view, advance reservation—similarly to the classification of switching paradigms criteria of Chapter 3—trades packet-blocking risk for circuit-blocking risk at the cost of a signalization. But this signalization can be of interest for the network operator as it carries some information about the expected outcomes of users. Indifference curves of binary utilities and fluid models of flows offer a trade-off between user satisfaction and network operator interest.

This proposal has a cost: "fairness", as considered in the NUM problem, has been put aside. The mechanism itself is fair in the sense that anyone can try to submit a request earlier than others. But this might need some admission control to prevent someone from issuing disproportional requests. In the end, the access link are still physical constraints to what can be requested as it is in the QoS model based on fairness and over-provisioning of the core, where the allocations are mainly governed by the capacity of the access links. This mitigates the ideal of fairness [Bonald and Massoulié, 2001].

Summary

Our approach combines the benefits of circuits and the benefits of packet networks, namely the fluidity of flow allocations. This approach maintains the value of the network.

After a presentation of the utilitarian goal achieved by actual sharing mechanisms of the Internet and a survey of the different attempts to bring QoS in the picture, we proposed a QoS solution based on the allocations providing non-zero utility to users and explicit advance reservation. This solution is claimed to both provide satisfaction to users and flexibility to the network operator. In addition, the proposition takes into account the identified value of a network and its externalities: network effect and congestion. Users have an incentive to request, in advance and explicitly, bandwidth for their needs, which gives network operators the knowledge of future traffic. The first-come first-served serving basis for allocation encourages users to submit as early as possible.

In the proposed approach, the congestion issue is managed by the network operator, which performs the allocations. Regarding charging or limitation of the contribution to the congestion, the network operator can devise tariffs varying with time to give users an incentive to use off-peak resources. Alternatively, this can be enforced through accounting and limitation of what can be requested. But in the end, network operators have to resort to charging rules that would not deter users from using reserved resources. Even though experimentations made on charging for quality made in the INDEX project in late '90s have shown that users are willing to pay for better service for some usages, it is admitted that they usually prefer simple charging rules. In this study, 5 different service rates were offered for different prices, in addition to a low-rate free service. It has shown that users uses in average 3.5 out of the 5 charged services each week [Edell and Varaiya, 1999].

In the next chapters, we will present our proposal of a service for a static network from algorithmic, architectural, and flow-control points of view. Then we will consider a provisioned network and propose a tiered architecture with distinct service providers and network operators, together with a network-provisioning scheme. Finally we will present our proposition of extension of routing games to consider time as a tool to study advance reservations and provisioning, and quantify the inefficiency potentially introduced by the selfishness of users.

Q2 How can bandwidth scheduling services help to provide guarantees on massive transfer completion time?

A2 *By allocating dedicated resources to transfers that have been explicitly specified in-advance and keeping track of on-going and up-coming transfers to be able to schedule the allocations.*

Q3 How do services fit in the economical environment?

A3 *This service both helps users with their concerns of guarantees and performance, and operators that will be able to plan future utilization of the network, schedule the requests when they prefer—as long as it meets users' constraints—and have a direct relation between allocated resources and user requests. Flow rate fairness is no more a criteria and congestion is managed by the network operator who can possibly charge the users or alternatively limit their utilization of the guaranteed service.*

Q4 How do they help regarding the value of the network?

A4 *The congestion, which reduce the quality of service experienced, is avoided for scheduled services. Best-effort traffic is still feasible and as simple as before in order to keep to benefit of network effect.*

— SIX —
DATA TRANSFER SCHEDULING

INTRODUCTION

In the previous chapter, we have proposed a framework for the in-advance, explicit requesting of resources. This has been motivated by the three following requirements: (i) satisfying users with connectivity needs, (ii) satisfying users with specific needs, and (iii) giving the network operator the advance knowledge and flexibility regarding the allocations of future demands.

In this chapter, we present a transfer scheduling service that fits this model. We develop its structure and algorithmic. The first section describes the model and the objective of the operator, the second section details the flows-allocation algorithms.

6.1 MODELS AND OBJECTIVES

6.1.1 SHARING MODEL

In this work, we consider two isolated classes of traffic: one is the best-effort traffic class and the other the scheduled traffic class. As scheduled aggregate can vary in time with the variation of the demand, the boundary between the two classes also depends on time. This aggregate comprises the different allocations scheduled to satisfy explicit requests sent by the users.

Lets consider a single link (strictly speaking, a half-link). Each link has its own sharing profile. The model is the same for all the links. We first assume that each scheduled request does not consume more than its allocated resources. Note that in the remainder of this chapter, *profile* refers to a non-negative function of time defining a rate as defined in Def. 6.1.1.

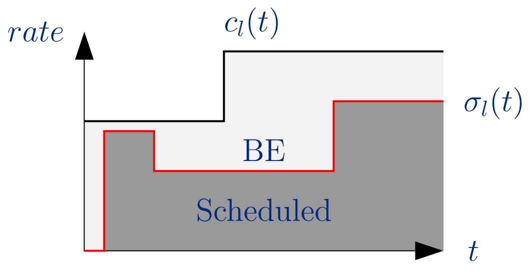


Figure 6.1: Sharing of link capacity between best-effort and scheduled traffic.

Definition 6.1.1 (Profile). A profile is a non-negative, piecewise-continuous function of time.

Since requests come in advance and are subject to admission control and scheduling, the scheduler can ensure that there will remain service rate for best-effort traffic and scheduled traffic will not consume all the ressources. This has to be specified by scheduling policies.

Definition 6.1.2 (Link capacity profile). For a link $l \in E$ of the network represented as a digraph $G = (V, E)$, $t \mapsto c_l(t)$ is a profile and gives the capacity of this link as a function of time.

Assumption 6.1.3. $c_l(t)$ is a step-function of time with rational breaks.

In this chapter, c_l is supposed to be given for each link. The infrastructure is assumed to be static—it cannot be resized on-demand. The links can still have planned outages or capacity upgrades and thus have their capacity profiles updated.

Definition 6.1.4 (Profile of scheduled aggregate). For a link l of the network G, $t \mapsto \sigma_l(t)$ gives the amount of bandwidth used by the scheduled traffic.

Obviously $\sigma_l(t)$ has to be less than $c_l(t)$ for any t. For a link l, at time t, the difference $c_l(t) - \sigma_l(t)$ gives the amount of bandwidth that best-effort will be able to use. Figure 6.1 presents these two profiles and the two classes of traffic.

6.1.2 Network Model

Clients and network operators do not have access to the same information describing the network. Therefore, we define the *user network model* and the *operator network model*.

6.1.2.1 User's Point of View: User Network Model

From the user's point of view, the network is seen as a cloud. This means he knows the *edgepoints* but not the links that connect them together. He has to assume that all pairs of edgepoints are interconnected or, in a slight variation of the model, the network operator can expose the list of pairs of edgepoints that can be used. This vision of the network as clouds with their schedulers is the one adopted by the OGF *Network Service Interface* working group [NSI09]

But as a network comprises several domains, potentially managed by different operators, users see clouds interconnected by edgepoints. We suppose that each domain has its own scheduling service but there can possibly be more than one. In addition, some sites might be providing other kinds of services such as: computing power, storage, or content. We focus on network services.

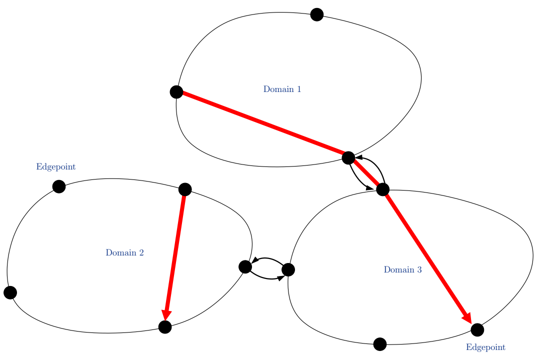


Figure 6.2: User model of the network.

Note that users are not necessarily anchored to one domain. Unlike what RSVP proposes, in the service model, the requests do not have to be issued by the source and can be submitted by a third party. A user can request to transfer data from the domain where his data is stored to a site where it will be processed on rented computing resources.

Figure 6.2 shows an example of network topology as seen by a user. He can see three different domains, how they are interconnected and some of the edgepoints of each domains he has access to. Red arrows represent source and destination pairs that a user can use for a request he can issue.

In order to request the network service, represented by the arrow on the left, the scheduler of domain 2 only needs to be contacted. The service represented by the arrow on the right requires coordination between the schedulers of domain 1 and domain 3. Both the chain model and the tree model—presented in Chapter 3—can be used. The reservation can be made by asking first domain. The scheduler of this domain will issue the request to the next domain. Alternatively users can send requests to each domains, but they have to ensure they will obtain something they can use for their transfers. Finally users can also send a request to another scheduler and delegate the negotiation.

A *network service* is a defined as follow:

Definition 6.1.5 (Network service). A network service is defined by the contract defining: How will the client use the allocation? What is he supposed to conform to to get what he expects? What will the provider ensure regarding the allocation?

This is a *Service Level Agreement* (SLA).

As a follow-up to the framework presented in previous chapter, users will express their needs—the regions where their utilities are not null—in the request for a network service. The allocation associated to the network service must then be enough to support it. For a file transfer, the total volume that can be transferred during the time window has to be equal to the requested volume. But the user also needs to be able to send at the rate specified in the upper bound profile $\lambda_r^{max}(t)$.

In order to be able to use the resources, the description of the network service must contain the information needed by users to identify their flows. Once the packets are tagged with this label, they can be routed to the right resources. This can be done using an Ethernet VLAN identifier, for example.

Finally, the user part of the contract can include payment but also constraints for traffic

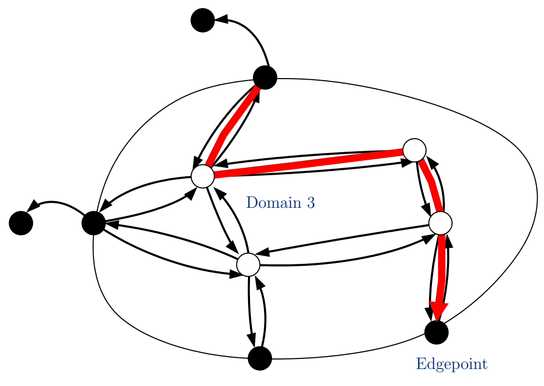


Figure 6.3: Operator model of a domain.

admission. Similar scheme has been proposed for IntServ requests (RSVP): the T-Spec and R-Spec parts of reservation messages. T-Spec specifies what the traffic will look like and R-Spec defines what is expected from the network side—what has been reserved.

The precise form of the network service for data transfer will be detailed in the next section.

6.1.2.2 Operator's Point of View: Operator Network Model

Let's focus on one single domain or "cloud". As briefly stated before, we consider a domain as a digraph where links of the network are represented by two half-links weighted by a capacity profile. The propagation delays of the links are not considered for scheduling but can be used for routing. Note that considering it in the scheduling problem is possible as the time expansion used can still be done [Hall et al., 2007].

The nodes of the graph are the routers of the domain. Some of them are special: they are referred to as edgepoints when they interconnect other domains or sites, and they are published to users. They are the nodes that users will use in the requests.

Regarding the interconnection with other domains, it can be done on edgepoints or on a set of links, each domain having its edgepoints and managing its half uplink to the other domain. It does not change much for the description of the model. However, it has an impact on the information that is shown to users to describe the interconnections of clouds.

Figure 6.3 shows the operator's view of the domain 3. In addition to the edgepoints, it also knows the half links and internal routers, and the uplinks. Since he has received the request made by the users and decided of its allocation, he knows exactly where this transfer will be going through.

In the remainder of this chapter, unless otherwise stated, we focus on one domain under the supervision of one scheduler.

6.1.3 Request Model

In this chapter, we consider an ideal fluid transport protocol which can fully use the allocated capacity. We exploit the fluidity of flows of packets that are able to adapt to the available bandwidth, and formulate the utility of users in terms of volume to be transferred. The next chapter will details what the practical challenges are and how they can be overcome.

$(s_r, d_r, v_r, \lambda_r^{max}, \lambda_r^{min})$	transfer request r
(s_r, d_r)	source-destination pair
v_r	volume
λ_r^{max}	upper profile for rate
λ_r^{min}	lower profile for rate
$[t_r^s, t_r^d]$	reservation window
$t \mapsto p_{r,l}(t)$	bandwidth allocation profile of r on link l
$\delta_{r,l}$	routing matrix for request r
$t \mapsto c_l(t)$	link capacity profile
$t \mapsto \sigma_l(t)$	profile of scheduled aggregate

Table 6.1: Notations

6.1.3.1 Malleable Requests

The kind of requests we consider here are defined by a volume and two profiles. One gives the minimum value the allocated rate must have, the other gives the maximum rate. Finally, the volume gives the sum over time of the allocated profile (or possibly profiles, if multiple paths are used). This definition captures various types of requests.

Definition 6.1.6 (Transfer Request). Each request r is a 5-tuple $(s_r, d_r, v_r, \lambda_r^{max}, \lambda_r^{min})$ where s_r is the source, d_r is the destination, v_r is the volume to transfer, λ_r^{max} is the profile describing the maximum rate the sender can send data at, and λ_r^{min} is the profile describing the minimum rate he wants to obtain over time.

Functions $t \mapsto \lambda_r^{min}(t)$ and $t \mapsto \lambda_r^{max}(t)$ are assumed to be profiles as defined in Def. 6.1.1. There are also assumed to have a bounded supports. We note: $t_r^s \triangleq \inf(\text{supp}(\lambda_r^{max}))$ the *start time* and $t_r^d \triangleq \sup(\text{supp}(\lambda_r^{max}))$ the *end time*. The start and end times are supposed to be rational numbers.

Notations are summarized in Table 6.1

We assume that, for file transfer, path delay is not really important as it is always less than a few hundreds of milliseconds and many orders of magnitude less than the transfer time of files: about 80 s for a 1 GB transfer at 100 Mbps. Path selection is left to the network operator and will be discussed later.

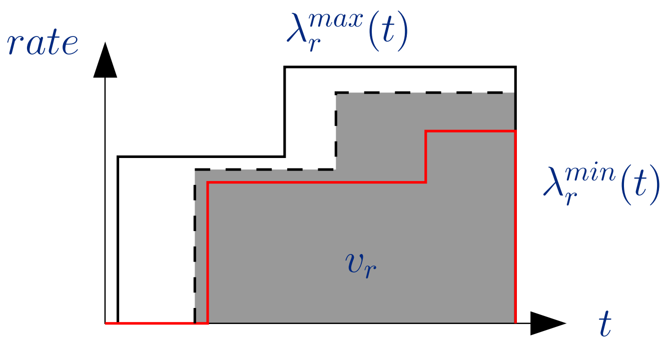


Figure 6.4: A valid request.

Within the requests that can be specified with this model, some are not valid. And so we define a *valid request*:

Definition 6.1.7 (Valid Request). A request is said to be valid if the following conditions hold:

- $\forall t, \lambda_r^{min}(t) \leq \lambda_r^{max}(t)$
- $\displaystyle\int_t \lambda_r^{min}(t)\mathrm{d}t \leq v_r \leq \int_t \lambda_r^{max}(t)\mathrm{d}t < \infty$

Figure 6.4 illustrates a valid request. The dashed line represents an example of an allocation that can be used to fulfill the requirements of this request.

Note that if $\lambda_r^{min}(t) = \lambda_r^{max}(t)$ and $v_r = \int_t \lambda_r^{max}(t)\mathrm{d}t$, the request specifies a profile. This profile can also be a constant rate on a time interval—a single rate profile. Similarly, if $\lambda_r^{min}(t) = 0$, the request describes an elastic transfer between two dates.

6.1.3.2 Feasibility

The submission of a valid request to the network operator leads to a reservation, the object manipulated by the scheduler.

Definition 6.1.8 (Bandwidth Allocation Profile). Once scheduled, a request r has an allocation represented by its *bandwidth allocation profile* on each link: $t \mapsto p_{r,l}(t)$.

Using this definition, the aggregate scheduled traffic $\sigma_l(.)$ is obtained by summing the profiles of the scheduled requests: for a set of such requests, for every link l,

$$\sigma_l(.) = \sum_{r\in R} p_{r,l}(.) \tag{6.1}$$

.

We now define the notion of *feasibility* for a set of requests.

Definition 6.1.9 (Feasible Set of Request). For a network $G = (V, E)$ with links l in E labeled by capacity $c_l(.)$, a set R of valid requests is said to be feasible if for each request r and link l there exist profiles $p_{r,l}(.)$ such that:

- *Rate constraints at source:* $\forall r \in R, \forall t, \lambda_r^{min}(t) \leq p_r(t) \leq \lambda_r^{max}(t)$ and;
- *Flows conservation:* $\forall r \in R, \forall n \in V\backslash\{s_r, d_r\}, \forall t, \displaystyle\sum_{l\in\delta^-(n)} p_{r,l}(t) = \sum_{l\in\delta^+(n)} p_{r,l}(t)$ and;
- *Volume constraints:* $\displaystyle\int_t p_r(t)\mathrm{d}t = v_r$ and;
- *Capacity constraints:* $\forall t, \forall l \in E, \displaystyle\sum_{r\in R} p_{r,l}(t) \leq c_l(t)$.

where $p_r(t) \triangleq \displaystyle\sum_{l\in\delta^+(s_r)} p_{r,l}(t)$ and δ^- and δ^+ are respectively sets of incoming and outgoing links of a node.

A request can be accepted if the set of requests comprising all the previously accepted requests and the request itself is feasible.

Still, the profiles $[p_{r,l}(.)]_{r\in R, l\in E}$ leading to a feasible allocation are not necessarily unique and some flexibility can remains for the scheduler. The next section discuss which vectors of profile to select among the feasible ones.

6.1.4 Allocation Objective

As already mentioned, in order to maintain the value of the network, best-effort traffic has to be considered with care. To prevent it from being starved, the objective we use for the allocation problem is to minimize the *congestion factor* γ and can be defined here as the maximum of the rate of aggregate scheduled traffic by the rate of capacity profile over the time and over the links [Chen and Vicat-Blanc Primet, 2007].

Definition 6.1.10 (Congestion Factor). Consider a network $G = (V, E)$ with links l in E labeled by capacity $c_l(.)$ and a set of request R with their bandwidth allocation profiles. Bandwidth allocation profiles are supposed to satisfy the first three constraints of Def. 6.1.9. They also define for each link an aggregate of scheduled traffic $\sigma_l(.)$.

The congestion factor of a network for this setting is defined as:

$$\gamma = \max_{l \in E} \left(\max_{t} \frac{\sigma_l(t)}{c_l(t)} \right)$$

(with the assumption that $0/0 = 0$ and $a/0 = +\infty$ if $a > 0$).

Proposition 6.1.11. *A set of requests is feasible iff γ is between 0 and 1.*

Proof. The three first constraints of Def. 6.1.9 are assumed to be satisfied.

γ is always positive.

$\Rightarrow$ Due to capacity constraint of Def. 6.1.9, $\forall t, \forall l \in E, \sum_{r \in R} p_{r,l}(t) \leq c_l(t)$, then from (6.1), $\forall t, \forall l \in E, \sigma_l(t) \leq c_l(t)$, finally $\gamma \leq 1$.

$\Leftarrow$ Since the maximum of the ratio of $\frac{\sigma_l(t)}{c_l(t)}$ is less than 1, $\sigma_l(t)$ is always less than $c_l(t)$. Note that with the above definition of γ, if $c_l(t)$ is null for some t, γ is less than 1 (actually finite) only if $\sigma_l(t)$ is null too.

□

For a set of requests with bandwidth allocation profiles that satisfy them—that enable issuers to perform their transfers within their time constraints—, the allocation is feasible, in the sense that it will not lead to congestion, iff γ is less than 1.

Since γ quantifies the worst congestion of the network (over time), when deciding the allocation of requests, we will try to minimize it. Minimizing the congestion factor is the objective of the optimization problem solved for the allocation as presented in previous chapter.

As demonstrated by Theorem 1 in [Chen and Vicat-Blanc Primet, 2007], minimization of the congestion factor can be achieved by considering only bandwidth allocation profiles as step functions of time. This does not say that allocation profiles have to be step functions, but that there are allocation profiles as step functions that provide a congestion factor as good as the optimal one. Furthermore, for the steps of these functions, it is enough to consider steps that come from the start and end times of requests, and breaks in capacity profile.

Assumption 6.1.12. In the remainder, we assume allocation profiles are step functions.

6.1.5 Conclusion

This section has presented the model we use for flow allocation. In this service model, users want the resources needed to, *e.g.,* transfer a specific file. The network operator is supposed to provide them if he is sure that the user can actually perform this transfer and is not limited by, *e.g.,* disk I/O capacity.

In practice, this network service will probably require some qualification test from a user before he is able to apply for it—as it is done with video on-demand for codec availability.

It is made of the constraints coming from the requests that must be satisfied by the allocation in order to serve as a network service. We have also presented the different states a reservation can have and their semantic. Finally, the objective for the optimization problem as well as the source of flexibility coming from profiles of rate over time and space has also been presented. Additional constraints will be added to this in the next section, where different cases of network topology will be considered. The next section presents the online version, where requests come one after the other.

6.2 Flow Allocation Algorithms

In the previous section, we have shown the model and optimization objective adopted for the allocations. We now present the algorithms to perform flow allocation.

6.2.1 Problem

Since requests are received online and some transfers have to start at random times, the allocation of new requests has to be done online, based on the current state of the system. We first define the states the reservations can have individually.

6.2.1.1 State Diagram and Submission

We consider different states for reservations. They can have an allocation—planned resources—or not and the resources can be configured or not.

Assumption 6.2.1 (State Diagram). For a reservation, we distinguish four logical states: *New*, *Scheduled*, *Granted* and *Rejected*.

Additional operational states such as "Running" or "Terminated" can also be considered when dealing with how to implement it but they will not be discussed here. The "Running" state can be used to specify that the network has been configured and can be used for the reservation.

To each state are associated different guarantees. A reservation enters the"New" state if its request has been submitted, but no resources are allocated and no decision regarding acceptation or rejection of the request has been taken.

The "Scheduled" state means that the reservation has been accepted but the exact allocation is not known—it can still be changed by the operator. "Rejected" simply means that the request has not been accepted, so no resources will ever be allocated to this reservation.

Finally, in the "Granted" state, the resource allocation is final and will not be changed anymore. In this state, the user can query the operator to get the description of the allocation such as the profile and the identifier he is supposed to use for his traffic.

Looking at transitions, the transition between New and Scheduled states is based on the operator's decision to accept or reject the request. The same applies for the transition from New to Rejected. Finally, the transition from Scheduled to Granted is based on a time and occurs $\delta_{S \to G}$ before t_r^s.

Regarding the transition from New to Scheduled, in addition to decisions based on policies, this transition has to keep the system in a correct state.

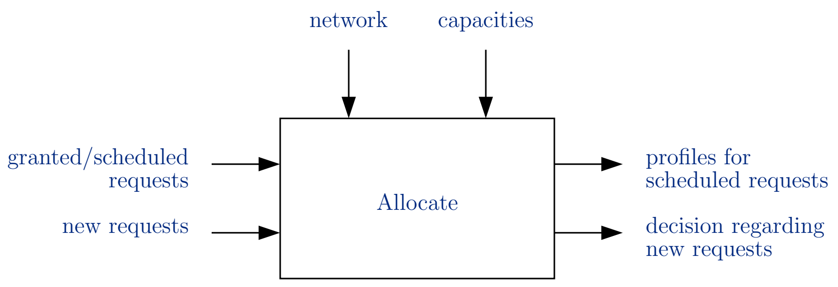

network
capacities
granted/scheduled requests
new requests
Allocate
profiles for scheduled requests
decision regarding new requests

Figure 6.5: Basic structure of the allocation procedure.

A request can be accepted and reaches the Scheduled state if the set of requests comprising all the previously accepted requests (Scheduled and Granted) and the request itself is feasible. This remark is used to perform the online scheduling of requests that comes one after the other.

6.2.1.2 Online Scheduling

For the sake of correctness with the model presented such as the absence of congestion, the allocation algorithms must take the current global state of the network from one valid state to another with possibly new requests accepted. By valid we mean that the set of scheduled and granted requests are feasible in the network.

Already granted requests can no longer have their profiles changed, and they need not be considered individually: only the aggregate is useful. The state of the network is then described by:

- the network and its capacity profiles;
- the requests in the Scheduled state;
- the aggregate of profiles of Granted requests;
- the new requests.

The result of the allocation algorithm is made of the profiles for all the Scheduled requests and the updated states of the requests that were New before running the algorithm: Scheduled—possibly Granted or Running—or Rejected.

This algorithm will be presented in the next section after a presentation of two different cases of network model.

6.2.2 Type of Network and Path Selection

In the set of constraints presented so far, nothing prevents the allocation from using multiple paths, and possibly all of them, between the source and destination of a request.

If the network is a tree, by definition, there is at most one path between any two nodes of the network. This case is typically the one encountered in Ethernet LAN since it does not support loops. In this case, the alternative paths are either removed by design—because there are no physical loops in the network topology—or logically by the Ethernet spanning tree algorithm.

Here, the size of the problem can be reduced. Since the path between the source and destination is known before and there is only one, for one request, because of the flow conservation constraints, the bandwidth allocation profile is the same for all the links used by a request. In this case, the feasibility definition can be reduced as in the following proposition.

Proposition 6.2.2. *For a network $G = (V, E)$ with links l in E labeled by capacity $c_l(.)$ such that G is a tree, a set R of valid request is said to be feasible iff for each there exists a profile $p_r(.)$ such that:*

- Rate constraints at source: $\forall r \in R, \forall t, \lambda_r^{min}(t) \leq p_r(t) \leq \lambda_r^{max}(t)$ *and;*
- Volume constraints: $\displaystyle\int_t p_r(t)\mathrm{d}t = v_r$ *and;*
- Capacity constraints: $\displaystyle\forall t, \forall l \in E, \sum_{r \in R} \delta_{r,l}\ p_r(t) \leq c_l(t)$.

where $p_r(t)$ is the bandwidth allocation profile along the only path between s_r and d_r and $\delta_{r,l}$ is the routing matrix. More formally,

$$\delta_{r,l} = \begin{cases} 1 & \textit{if } r \textit{ uses link } l; \\ 0 & \textit{else.} \end{cases}$$

Proof. The constraints of the proposition obviously imply the ones of the definition, since the unique profile for a request, once mapped to all the links of the path given by the routing matrix, gives a bandwidth allocation that satisfies the flow conservation for each request.

For the other direction, since for a request there is only one path, all the traffic of this path goes on this path and conservation constraints make it so that $p_{r,l}(.)$ are equal along the path and null out of it: it can thus be called $p_r(.)$. Finally, the capacity constraints are expressed using the sum of the profile for each path using the considered link. □

Here, instead of asking the solver to find the path, it can be computed before, and then given to the solver as an input. The optimization procedure will only have to consider the temporal part of the problem through the profiles, and not the space part of choosing the path.

In case the network is not a tree but a graph, it can be of interest to consider single-path or multipath allocations.

The multipath problem has been studied in the case of overlay networks where the routing is made in the overlay at application level and can thus be decided by the application even if the actual path that packets will take also depends on the network's decisions [3]. In this work, there is no restriction regarding the paths, and requests can be served on multiple parallel paths.

For practical reasons, multipath allocations are not always desirable. For example, they introduce the need to resort to reordering of packets on arrival since different paths can have different delays. In order to have requests served on a single path, in the remainder of this chapter, we assume the path or routing matrix is given with the request, before the allocation profiles are optimized.

In the case of a tree, it does not change anything. In the case of a normal graph, however, this sacrifices the optimality that an integer linear formulation would give at the benefit of a polynomial solution, as will be demonstrated in the next section. Path selection can be done using a shortest path algorithm in terms of number of hops.

Assumption 6.2.3. For every request, the selected path is the shortest path in terms of number of hops.

Assumption 6.2.3 makes sense: since shortest path is used, less links are used than in a longer path, and less resources are used to serve a given request. This obviously leads to rejecting some requests if, for example, the shortest path is overloaded. The link metric used to determine the shortest path could include utilization of the link as in [Awerbuch et al., 1993] to move some requests out of an overloaded links.

In the next section, the scheduling part of the allocation process will be described.

6.2.3 Scheduling

In addition to assumption 6.2.3 we add the following assumption.

Assumption 6.2.4. Once accepted, requests can not be preempted. In addition, they are processed and scheduled on a first-come-first-served basis.

Besides simplifying the problem, this assumption find its justification by creating an incentive for users to submit their requests as soon as possible.

The allocation of a request—or a set of requests—has to go through the steps listed thereafter:

- Path computation;
- Collection of requests that can be rescheduled;
- Scheduling and decision regarding acceptation.

We will first describe the algorithm to get the requests that will be rescheduled. To do so, we first define what we call *overlapping requests*.

Definition 6.2.5 (Overlapping Requests). Two requests are said to overlap if their paths share at least one link and their time windows overlap.

By extension, a request is said to overlap with a set if it overlaps with at least one request of the set.

In order to collect all the requests that can be rescheduled to make room for the new requests, starting from a set of requests as the new requests, we iteratively collect the overlapping requests in the set of scheduled requests. This is done in a iterative way as overlapping is not a transitive relation. Since transferring a volume from one time interval to another can help to accept a new request, the collection algorithm will stop when no new requests can be added to the set R of requests to (re-)schedule. The algorithm 1 takes as input the set of new requests R^{new}, the set of scheduled requests $R^{sched.}$ and the routing matrix $\delta_{r,l}$.

This algorithm terminates as the number of elements of $\tilde{R}$—which is initialized with $R^{sched.}$—strictly decreases when the flag is set to true.

By construction, when the algorithm terminates, if there are still requests in $\tilde{R}$, they do not overlap with R.

The while-loop is executed at most $|R^{sched.}|$. For each iteration, all the requests of $\tilde{R}$—whose size is bounded by $|R^{sched.}|$—are considered for overlapping with requests of R. It follows that this algorithm has a polynomial running time (less than $O(|R^{sched.}|^3)$).

Because of Assumption 6.1.12, we need to determine the partition of the time axis in time intervals for the step functions. This partition of time comes from the time window of requests in the set of requests R that can be rescheduled, and also from the links' capacity profiles and aggregate profiles of requests that will not be rescheduled.

For a set of requests to be rescheduled R, a collection of aggregates profiles $p_l^a(.)$ of requests that will not be rescheduled R', capacity profiles $c_l(.)$, and a routing matrix $\delta_{r,l}$, Algorithm 2 returns the partition of time $\mathcal{T}$ represented by the list of bounds of the time intervals that comprise the partition. Both $p_l^a(.)$ and $c_l(.)$ are assumed to be step functions: $p_l^a(.)$ because it is a sum of profiles assumed to be step functions, and $c_l(.)$ by assumption on the link capacity

Algorithm 1 Collect requests to reschedule

```
Input: $R^{new}, R^{sched.}, \delta_{r,l}$
Output: $R$
    // Initialization.
 1: $R \leftarrow R^{new}$
 2: $R \leftarrow \emptyset$
 3: $\tilde{R} \leftarrow R^{sched.}$
 4: $updated\_flag \leftarrow false$
    // Main loop: stop if there is no more scheduled request to consider or the fixed point as
    been reached.
 5: while $(\tilde{R} \neq \emptyset) \wedge updated\_flag$ do
 6:   $\tilde{R}' \leftarrow \emptyset$
 7:   $R' \leftarrow \emptyset$
 8:   $updated\_flag \leftarrow false$
      // Classify requests as overlapping/non-overlapping.
 9:   for all $r \in \tilde{R}$ do
10:     if $r$ overlaps $R$ according to the routing matrix $\delta_{r,l}$ then
11:       $R' \leftarrow R' \bigcup \{r\}$
12:       $updated\_flag \leftarrow true$
13:     else
14:       $\tilde{R}' \leftarrow \tilde{R}' \bigcup \{r\}$
15:     end if
16:   end for
      // Update the set for the next iteration.
17:   $\tilde{R} \leftarrow \tilde{R}'$
18:   $R \leftarrow R'$
19: end while
20: return $R$
```

profiles. We note $breaks(.)$ the function which returns the time instant where a step function changes of value. Figure 6.6 illustrates the partitioning of time.

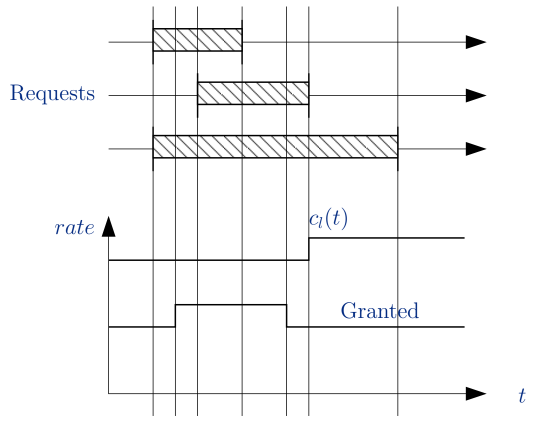


Figure 6.6: Partitioning of time

This algorithm also offers a polynomial running time since the "new" breaks only come from the set of request R and there are only two per request of R.

Then, from this partition, as allocation profiles are constant on each interval, using a linear program that encodes the constraints of Def. 6.1.9, with the objective function of minimizing the congestion factor, from Prop. 6.1.11, we get that an allocation is feasible iff the achieved congestion factor is less than 1.

We use i for the intervals defined by the breaks of $\mathcal{T}$ and with a slight abuse of notation write $i \in \mathcal{T}$ to denote them and $|i|$ for the length of the interval. We also use $p_{r,l,i}$ to denote the constant rate that $p_{r,l}(.)$ has on the interval i. Similarly, $p^a_{l,i}$ is the rate of $p^a_l(.)$ and $c_{l,i}$, $\lambda^{max}_{r,i}$ and $\lambda^{min}_{r,i}$ are the values of the corresponding profile on interval i. Finally we use

$$\eta_{r,i} = \begin{cases} 1 \text{ if } r \text{ is active during interval } i; \\ 0 \text{ else.} \end{cases}$$

Then the linear program is the following:

$$\begin{aligned} BDTS: \quad & \text{minimize} \quad \gamma \\ & \text{s.t.} \\ & \forall r \in R, \sum_{i \in \mathcal{T}} \sum_{l \in E} \delta_{r,l}\ \eta_{r,i}\ p_{r,l,i}\ |i| = v_r \\ & \forall l \in E, \forall i \in \mathcal{T}, \sum_{r \in R} \delta_{r,l}\ \eta_{r,i}\ p_{r,l,i} + p^a_{l,i} \leq \gamma\ c_{l,i} \\ & \lambda^{min}_{r,i} \leq p_{r,l,i} \leq \lambda^{max}_{r,i} \end{aligned}$$

In the case of a single request to schedule, since the previous reservations and corresponding allocations are assumed to be feasible, the question of admitting the request is based on the existence of a feasible solution for the previous requests plus the new one. If the path computation is supposed to be fixed, which we assumed, the decision relies on the existence

Algorithm 2 Build partition of time

```
Input: $R, p_l^a(.), \delta_{r,l}, c_l(.)$
Output: $\mathcal{T}$
    // Initialization
 1: $\mathcal{T} \leftarrow \emptyset$
 2: $t_{min} = \min\{t_r^s | r \in R\}$
 3: $t_{max} = \max\{t_r^d | r \in R\}$
    // For all (re-)scheduled requests:
 4: for all $r \in R$ do
 5:   // Add start and end time,
 6:   $\mathcal{T} \leftarrow \mathcal{T} \bigcup \{t_r^s, t_r^d\}$
 7:   // and breaks of capacity profiles of utilized links.
 8:   for all $l$ do
 9:     if $\delta_{r,l} = 1$ then
10:       $\mathcal{T} \leftarrow \mathcal{T} \bigcup breaks(c_l)$
11:     end if
12:   end for
13: end for
    // Add time breaks of already scheduled requests.
14: $\mathcal{T} \leftarrow \mathcal{T} \bigcup breaks(p_l^a)$
    // Remove all the time breaks out of the time interval $[t_{min}, t_{max}]$
15: $\mathcal{T}' \leftarrow \emptyset$
16: for all $t \in \mathcal{T}$ do
17:   if $t_{min} \leq t \leq t_{max}$ then
18:     $\mathcal{T}' \leftarrow \mathcal{T}' \bigcup \{t\}$
19:   end if
20: end for
21: $\mathcal{T} \leftarrow \mathcal{T}'$
22: return $\mathcal{T}$
```

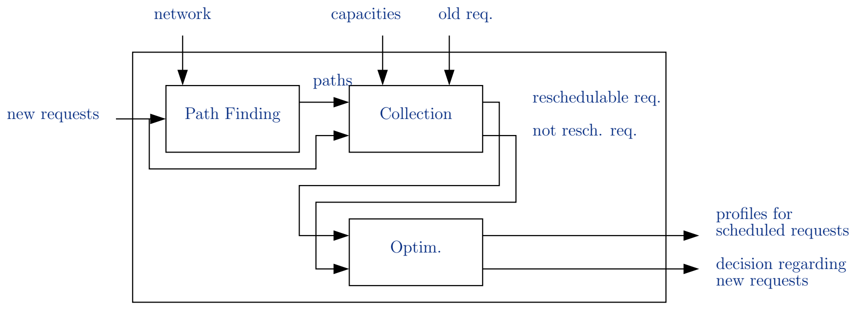


Figure 6.7: Detailed structure of the allocation procedure.

of a solution to the linear program which tries to reschedule all the requests that can be. If there is a solution, the new request can be accepted and its state switched to Scheduled. If there is none, it has to be rejected as it does not fit. In this case, note that it might have been possible to accept it with a different path.

But if we put constraints on the path, since the BDTS optimization problem has been proved to be solvable in a polynomial time in [Chen and Vicat-Blanc Primet, 2007], the whole procedure to accept a single request has a polynomial running time. Note that the reschedulable request collection phase depends on the number of overlapping requests and can lead to a large partition of the time axis. The number of time intervals can be reduced by preventing the breaks from being at random time instants and by forcing them to be mostly the same.

Figure 6.7 shows the proposed allocation procedure, in case there is only one request submitted, if the procedure fails to find a feasible solution, the request is rejected. The network topology, the capacity of links, the old requests as well as the new requests are the input parameters of the allocation procedure. The output of the procedure is made of a admission decision for the new requests and bandwidth allocation profiles for all the scheduled requests (old and newly accepted). When a new request comes, it is first assigned a path that depends on the request and the network topology. This assigned path as well as the paths assigned to old requests are then used to determine the set of requests that have to be scheduled or rescheduled. Finally, the optimisation is performed taking into account both not-reschedulable and schedulable requests.

In case multiple new requests are considered together by the allocation procedure, if the optimization succeeds, all the requests can be accepted. If the optimization does not find a solution, it essentially means that the requests does not fit. Some will have to be rejected: under the assumption 6.2.4, this can be done by applying a dichotomy procedure on the sorted list until a feasible sub-list is found. This adds a log factor of the number of new requests to the complexity.

To conclude this section, note that, under the assumption 6.2.4 of considering the requests in their arrival order and not allowing preemption, in case of a tree or if the path for each request is uniquely determined by some other constraint, the proposed approach leads to an optimal congestion factor.

6.2.4 Conclusion

In this section, the allocation problem of a request and a polynomial solution have been presented. The model allows specifying malleable requests and take into account both best-effort and scheduled traffic.

The solution is based on *a priori* path computation to reduce the utilization of resources, and a linear program to optimize the amount of resources available for best-effort traffic, while satisfying the constraints of the network service.

Summary

The service presented in this chapter proposes two kinds of network service: malleable transfers and best-effort traffic.

It is made to accommodate both best-effort traffic and scheduled traffic, which make the value of the network. This exploits the two aspects of the value of the network: the network effect due to potential connectivity and the guaranteed service isolated from congestion.

By allowing large users seeking guarantees to explicitly request resources, the bulk data transfer scheduler transfers flexibility to the network operator that schedules them with the knowledge of the other, previously received requests, and internal needs.

In Chapter 7, we show how the service can be implemented and articulated with the different function provided in the functional planes. We show how a time-varying profile can be followed and transfers timely done.

In Chapter 8, the present service is extended and shifted to a different agent: the service provider that runs the service, schedules the requests, and provisions its virtual infrastructure by issuing requests to a network operator owning the resources.

Using a cost function modeling time-slot preference and congestion aversion, we will show in Chapter 9 that this service is legitimate and sustainable.

Q5 Which kind of request is suitable for malleable network service?

A5 *Malleable requests specified using a profile giving the maximum requested rate as a step function of time, together with one giving the minimum rate and the volume that the actual allocation must sum to provide an expressive form for requests. This can be used for deadline-constrained file transfers as well as CBR or planned time-varying usages such as following the daily variation of needs.*

Q6 How can them be allocated?

A6 *Using the fluidity provided by packet networks, the optimization problem for the allocation is formulated as a network flow problem on the time-expanded network with the objective of minimizing the congestion factor. If the congestion factor is less than 1, requests can be accepted and the already-scheduled requests are not getting late'.*

— Seven —
Articulation of Service, Control and Data Planes

Introduction

In the previous chapter, the optimization and algorithmic parts of the service proposed to schedule large transfers have been presented.

In this chapter, we study the interactions between the three planes involved in our proposal: the service plane, the control plane, and the data plane. The service plane manages the interactions with the users and the scheduling of the requests with the in-advance capacity planning. The management plane has been left over and the service plane directly interacts with the control plane that configures the network resources in order to serve the reservation. But for networks having a management plane, the configuration sent by the service to the network elements can go through a *Network Management System* (NMS) which, for example, will be in charge of ensuring the protection of the allocated resources.

Once the allocation has been performed by the service, the resources have to be configured by the control plane so that users can effectively use them and do their transfers through the properly configured data plane.

This configuration is typically the role of the control plane. In addition to the regular functions of a control plane as they have been identified in Chapter 3, we also include the

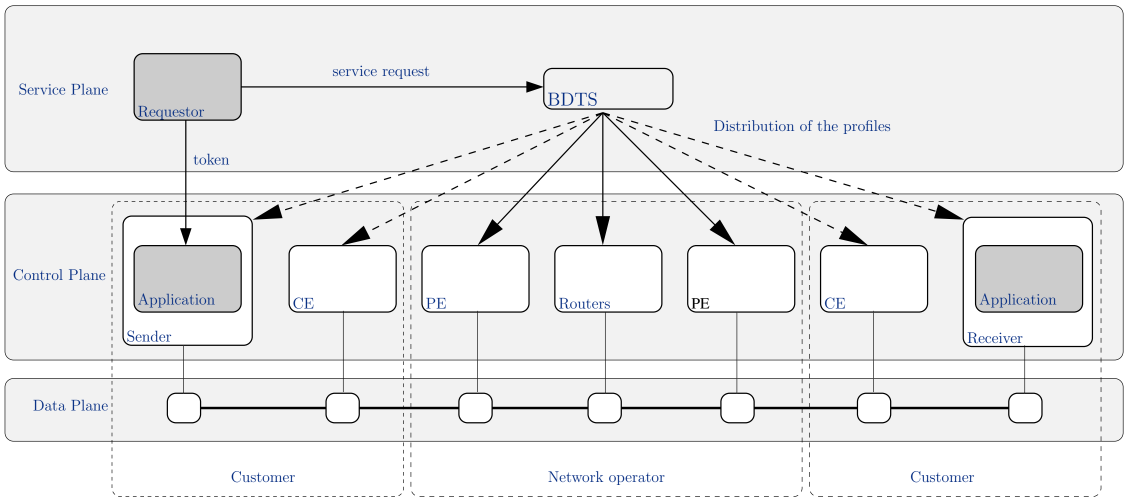


Figure 7.1: Distribution of profile information once service has been requested.

configuration to supply the allocation of requests in this plane—both in term of path and rate. This configuration is then used in the data plane to send the data in accordance with the profile. The profiles must be distributed to the network equipments as well as to the endpoints, which can be the user's application or the machine where it is running. Figure 7.1 presents the overall picture of the situation: the application or an agent acting on its behalf—the requestor—has got an identifier of the allocation resulting of its request made to *Bulk Data Transfer Service* (BDTS). When the reservation is about to start, BDTS has to distribute the configuration to different elements on the path in order to get the resources configured and allow the transfer from the sender to the receiver. Since different parts of the path belong to different agents, the distribution of profiles is likely to be done in different messages. The part between the two *provider edges* (PE) is owned by the network operator and contains the intermediate routers. From the *customer edge* (CE) to the sender or receiver, it is under the authority of the customer—it could be another agent that has delegated part of its authority to the user in order to let him use the resources. In this chapter, we discuss how the profile can be enforced sender-side. Note that conforming to the profile is in the sender's interest. If he does not and the traffic is policed at the PE, some packets will be dropped. Since the sender does not know which packets have been lost, the loss will only be detected after the receiver has sent duplicate ACKs, or negative ACK if the transport protocol supports it.

As in previous chapter, here, scheduled traffic is assumed to be isolated from best-effort traffic and processed with a higher priority. This implies that in case of congestion—due to an excess of best-effort traffic and not to the scheduled one which is subject to admission control—best-effort traffic can be dropped but not the scheduled one. Two classes with strict priorities are sufficient to implement such a policing. We will focus on the control of flows of the scheduled class since best-effort traffic uses the remaining bandwidth.

In the first section, the architecture of the service is presented. The flow controller is presented in the second section. Then, in that section, a discussion is held regarding the objective that this control must pursue, before the proposed solution is evaluated in the last section.

7.1 Service Architecture

The previously presented optimization process and algorithms have been implemented in a bandwidth broker: BDTS. This section describes the main aspects of the broker's architecture.

7.1.1 Overlay Model

Since the adopted model is an overlay model, users do not have access to the information managed by the service: they can only submit requests that are allocated by the service, and then are informed of the allocated bandwidth profile.

This basically means that the interface between the user and the service is very simple. Two functions are needed: one to submit the requests with the parameters previously mentioned, and one to get a description of the allocation once it has been made. Other functions such as the ability to cancel or modify the reservation could have been added.

The `submit()` operation realizes the transition from the New to Scheduled or Rejected states. `getProfile()` returns a profile corresponding to the allocated resources once it has been done. This last function is not available before the state of the request is changed to Granted as the allocation is not definitive and still subject to changes. Note that the actual path that will be used by the allocation is not shown to the user since it is imposed by the network.

7.1.2 Architecture: Scheduling Algorithm and BDTS Functions

In this section, we describe in detail, the design of the software components that form the BDTS service we have developed. Keeping the components as independent and flexible as possible, we discuss the interfaces that interweave these components.

The BDTS service maintains a calendar of the resources that have been allocated to requests. When a request is received, it is pushed to the scheduler that determines the allocation profile using the algorithms presented in the previous section. The profile generated by the scheduler contains the information of the traffic rate over time. The scheduler passes it on to a flow control service, which regulates the bandwidth of the data stream according to the profile, using the underlying network technology.

The customer of the BDTS service submits *transfer jobs* or *t-jobs*. A t-job specifies the constraints of the requested transfer. The BDTS scheduler then computes a bandwidth allocation profile, a network bandwidth function over time according to the t-job requirements, the other previously allocated requests, and the available network bandwidth.

For every link, a *calendar* is maintained in a network information system component. This component contains the reservations, their allocation profiles and the link they use and the capacity profiles for every links.

7.1.3 Admission Control and Scheduling

The process of establishing a reservation has two steps: admission control and effective scheduling. First, the caller creates a t-job, and *submits* it to the BDTS service, which evaluates the possibility of accepting it by computing the availability of all the resources necessary for such a transfer, within the requested time window. The reply for a t-job submission is either *accept* or *deny*, depending on its availability. At submission time, no details about time and space allocations are given. This enables the system to adapt to future load and incoming requests.

The second step, called *granting*, happens later in time, closer to the time when the application is ready to start transmitting data. At that moment the network allocation is calculated by the optimizer and made available to the requester. Postponing the confirmation of t-job schedules allows for multiple recalculations of them, in order to accommodate further requests, provided that a new request does not violate the constraints of any previously accepted t-jobs.

In response to the submission of a t-job, the caller receives a *token* which is a simple globally unique identifier. This token represents a reservation and should be used later by the application when binding an open socket to this reservation. Before the t-job time window starts, the application has to bind the reservation (the token) to an open socket. When a socket is bound to a reservation, its bandwidth profile is *granted* by the scheduler and will no longer be modified. Under request, the application can also have access to the profile associated with the reservation, for example when it wants to adapt its behavior.

7.1.4 Application Interface and Transfer Jobs

The application interface was designed to enable t-job submission to BDTS. This call is made by the application, or by an agent on behalf of it, prior to any scheduled communication. The interactions between the application and the flow control part will be detailed in Section 7.2.

The application submits a t-job to the *t-job Management*, or TJM, providing source and destination, rate constraints as two profiles and data volume to transfer. If there is enough bandwidth to schedule the new t-job within the specified time limits, the application receives a token as a positive reply (the negative is a null token).

TJM is the core of BDTS, and contains components to schedule t-jobs, update link calendars, store submissions, and optimize bandwidth usage. Figure 7.2 depicts it (in gray), with three components: a scheduler, a database, and an optimizer. Together they appropriately schedule t-jobs submitted by application agents. Besides optimizer and database, the scheduler also contacts some external components, *Link Calendar Database* and *Network Topology*, which are information systems associated to the network resources.

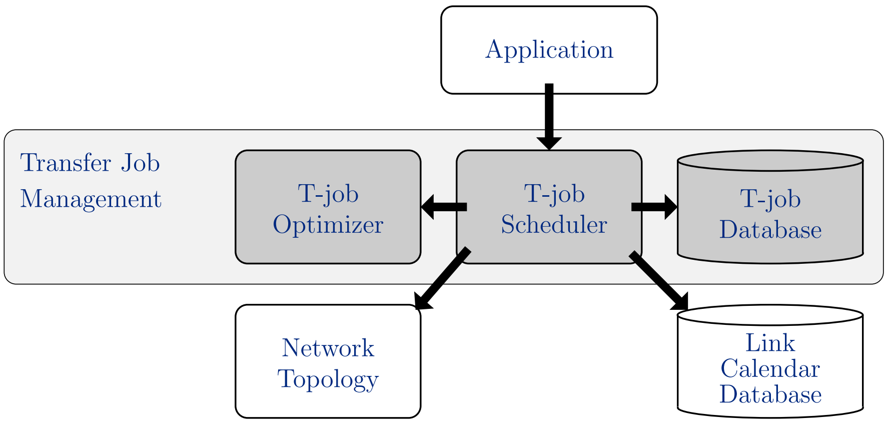


Figure 7.2: Transfer Job Management and interactions

7.1.5 Conclusion

In accordance with the overlay model, the software architecture for the proposed data transfer scheduler does not expose any detail of the topology of the underlying network. Users can only submit their request which is accepted or not. Whenever it is, they can obtain the profile of the allocation in order to be able to fully utilize it.

In the internal architecture, one part is responsible of scheduling the requests and manages their lifetime, and another one provides information regarding the topology and the planned utilization of resources. The latter could actually be provided by other entities such as an information service of the network.

Finally, an important part that has not yet been described, is all the control functions and activation processes that must be done to configure the network so that users can access their allocations. The part of this which takes place on the sender's side is the focus of the next section.

7.2 Interactions of Applications with Control and Data Planes

Since we consider fine-grained reservation as per-transfer or per-flow reservations, the control plane has to be extended to the end-host and the applications of the users. This extension is driven by two objectives: (i) to enable users to specify which flow is supposed to benefit from the reservation and (ii) to help the applications to use the reservations.

Because of the layered design of the Internet protocol stack, reliable transport protocols are able to adapt to changes of capacity or losses that can occur. In our proposal, since the bandwidth allocation profile is known by the sender in advance, the transport protocol can benefit from it and adapt more quickly to changes of rate than when relying on its classical mechanisms.

7.2.1 From User Needs to the Utilization of Allocated Rate

As described in the previous chapter, users issue requests to the scheduler, which in return gives an identifier of the reservation : the token. When the reservation is about to start, it becomes granted and the allocated profile is made final. In order to be used, this reservation has to be associated with a communication channel. A well-established concept to identify communication channel, from the application's point of view, is the socket. This basically identifies the endpoint of a connection. The action of associating a socket and a t-job is called binding.

Thus every node, for which the transfers are to be scheduled, runs a local *Flow Control* service, called *FLOC* for short, in order to guarantee that pre-reserved departing data flows respect the scheduled bandwidth profile. The interactions between the application, its agent, and BDTS appear in Figure 7.3. The figure contains two major blocks, TJM which contains the scheduler and other components, and FLOC, which controls the local operating system to manipulate the transmission rates.

The application binds a reservation by calling FLOC, passing the socket number, the process identification and the token as arguments. FLOC fetches the appropriate bandwidth reservation from BDTS (for the given token) and programs the local operating system to enforce the corresponding communication rates. Communications not attached to reservations fall back to a shared, best-effort, low priority traffic class, as explained previously.

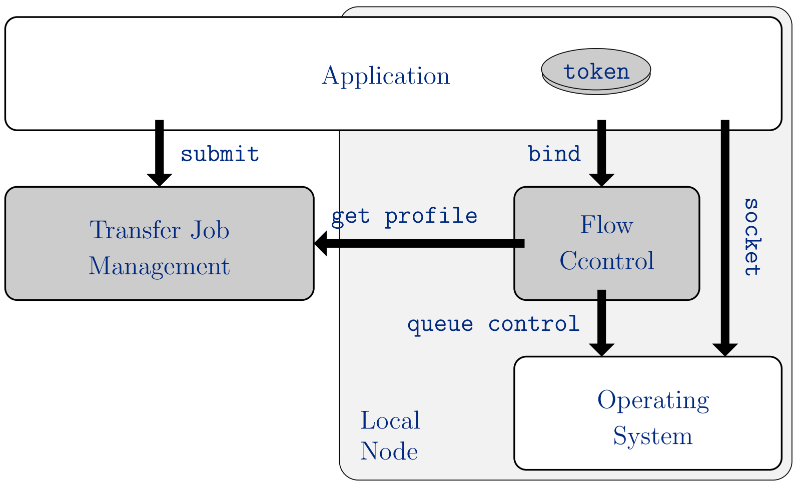


Figure 7.3: Application interface.

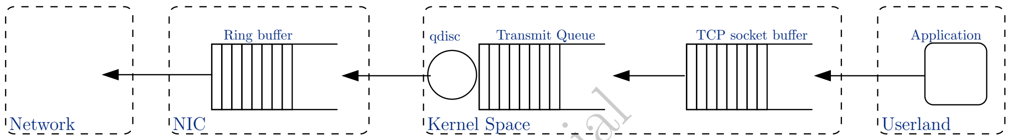


Figure 7.4: Illustration of the journey of packets before they reach the network.

The next two sections will detail the two interfaces: between the application and the OS, and between the application and FLOC, which controls the OS.

7.2.2 Data Plane: Applications/Operating System Interactions

In the remainder of the chapter, we consider TCP sockets in GNU/Linux. Whenever a `write()` is issued on a socket, the data is copied in a kernel buffer. In order to be able to retransmit it, this buffer stores the data as long as it has not been acknowledged by the recipient. A copy of the data is sent to a facility called *qdisc* which is responsible for queue management. This process is supervised by TCP's congestion control which limits the number of in-flight packets according to the value of the congestion window. This value evolves according to the congestion events and acknowledgments received.

Qdisc is basically made of three elements: filters, classes, and qdiscs. Filters classify packets and decide where they will be queued. Classes are the queues with possibly other attributes. And qdiscs manage how packets are de-queued from classes.

When the network interface card has finished sending the packets it has in its internal memory, it raises an interruption that makes the driver de-queue packets from the qdisc and put them into the card—actually, it asks the DMA engine to do so. Depending on how much packets are delivered by the de-queue function, and where they are coming from, different packet scheduling disciplines can be implemented and the output rate of a class of packets be moderated. Figure 7.4 represents the different queues that are encountered by data/packets before reaching the network.

7.2.3 Control Plane: Applications/Controller Interactions

The interactions between FLOC and the applications are based on the binding function previously mentioned that associates a token to a socket and a process identifier.

This enables FLOC to configure the qdisc according to the profile obtained from BDTS. This configuration is three-fold. First the queue discipline is set up to allow the creation of the per-reservation (token) class and associated filters. Then the classes have to be configured according to the rate of the profile. Third, since it is a profile and varies with time, the configuration has to be updated according to it.

7.2.3.1 Configuration of qdisc

As already mentioned, the choice of the qdiscs defines how packets from different classes will be served.

As we want to be able to give a fixed amount of bandwidth to different reservations, the qdisc must be able to assign a given rate to a class of traffic. The objective guiding the choice is to ensure the isolation of reservations as the sharing has already been decided by BDTS's scheduler. The details of the possible alternatives will be discussed in Section 7.3.2 where the solutions for rate limitation will be discussed.

7.2.3.2 Class and Filter

Whenever a new reservation is declared to FLOC, a new class is added to the qdisc so that flow using this reservation can get the expected amount of bandwidth.

In order to add the concept of flow to the classifier—which is basically a packet classifier—and avoid a filtering on the 5-tuple made of source and destination, ports and addresses, and protocol, a mark inherited by the packets from the socket has been added. This mark is given to the socket by FLOC. The corresponding filter is then created to put the packet coming from the socket in the class defined by the reservation.

7.2.3.3 Profiles and Time

Since the reservations are allocated on time-varying profiles, the amount of bandwidth that a flow will get through its class also has to vary with time.

In order to implement this, a thread of the FLOC daemon is responsible to change the configuration of the class according to the profile of bandwidth received from BDTS. Listing 7.1 shows such a profile. Rate limitation for reservation with token 0x42 is created at EPOCH date 1244561065 with a rate of 150 Mbps, then its rate is modified two times before being deleted 40 seconds later. This script is processed by FLOC's parser and each of these commands is translated by FLOC into qdisc management commands which are queued and issued to the kernel of the sender's machine at the proper time.

```
[1244561065] L0 := limit 150. token 0x42
[1244561085] modify L0 123.
[1244561095] modify L0 246.
[1244561105] del L0
```

Listing 7.1: FLOC profile

7.2.4 Conclusion

This combination of user actions and FLOC actions allows flows from user applications to be associated to reservations and rate control mechanisms at the packet level. This configuration evolves with time as reservations come and terminate, and as bandwidth profiles change. These changes are done by a thread of FLOC that is waiting and will take an action at the next event. In the next section, we will see what are the expected properties of these mechanisms and how they can be met.

7.3 Mechanisms of the Control and Data Planes

The control of the sending rate has two objectives, one is to satisfy the user regarding the request he made and the allocation the operator provided. The other is to keep the model utilized by the operator correct, to predict what is available. In other words, flows must not use more resources than the bandwidth allocation profile to satisfy the operator, and must not use less to satisfy the requests.

7.3.1 Rate Control

7.3.1.1 Objective—Ideal Transport Protocol

In the previous chapter, the fluid model adopted to model the traffic has lead to the formulation of the BDTS optimization problem. This formulation relies on the fluidity of the traffic in the sense that volume constraints and capacity constraints (through γ) uses the fact that a rational amount of rate/volume can be transferred from one time interval to another as long as the sum of small volumes is equal to the requested volume. Furthermore, the same "rate" was used for both volume constraint and capacity constraints. Since we want a linear formulation of the problem, this implies the existence of a linear relation between the two metrics: throughput and goodput.

In order to be as close as possible to this model, we define an *ideal transport protocol* having the properties listed thereafter and try to make the real system close to it. The wanted properties are:

P1: linearity (affine relation is enough) between throughput and goodput;
P2: linearity of integration of goodput over time and paths;
P3: no performance degradation due to the presence of other flows (isolation of flows).

Using the aforementioned properties, we define the ideal transport protocol as having the following relation between throughput ($r_{thr}(t)$) and goodput ($r_{goodput}(t)$):

$$r_{goodput}(t) = \frac{941.49}{1000} r_{thr}(t)$$

where both are expressed in Mbps. This relation is adapted for regular setting and TCP payload on 1 Gbps Ethernet. The precise calculations that have lead to this can be found in [1]. In order to satisfy **P2**, the ideal protocol has to satisfy the following relation (path bifurcation is not considered here):

$$\int_{t\in T} r_{goodput}(t) = v_T$$

where v_T is the volume this transport protocol is able to transfer during time interval T.

7.3.1.2 Rate Limitation: Average Rate and Peak Rate

In Chapter 2, it has been said that fair queueing mechanisms can emulate a bit-by-bit round robin or weighted round-robin to emulate flow isolation at a queue. But when the senders are different machines, in the absence of direct feedback or back-pressure from the shared queue to the senders, the TCP flows will reduce their sending rates because of congestion events, that are losses which require the receiver to send dupacks. This takes time. In order to prevent losses from occurring at this shared queue, whether there is a fair queueing discipline or not, we decided to have rate limitation on the sender's side.

Before going further, we have to look at the definition of rate. As transmission of information is based on packet, the fluid model reaches its limits when we look at a time scale close to the transmission time of a packet. When a packet is transmitted, the sending rate is actually the bit rate of the line. The only solution to reduce the average rate (at a larger time scale) is to introduce silence: this can be done in different ways.

Depending on how often silences are introduced, for a same average rate, bursts of packets will have different sizes. This has an impact on the (transient) congestion. Since buffers at switches or routers are limited, even if the sum of the average rates is less than the service rate of the queue, overflow can happen: if several bursts reach the queue at the same time for example.

To determine these silence intervals, different sources of time can be used in order to enforce a rate limitation.

In an end-host system, four different sources can be found:

(i) Userland timers;
(ii) TCP self-clocking, namely the RTT of the transfer's path;
(iii) the OS's kernel timers;
(iv) Packet-level clocking.

Even though CPU have very high frequency clock which would allow to make precise busy wait, this solution is not practical since it wastes CPU resources.

Mechanisms based on userland timer: Application-level timers can provide timers with a frequency up to 8192 Hz through RTC[1] under GNU/Linux. RTC timers above 64 Hz are normally not accessible to users. Another solution, the use of POSIX functions like `nanosleep()` has some limitations as the man-page suggests:

> *The current implementation of `nanosleep()` is based on the normal kernel timer mechanism, which has a resolution of 1/HZ s [. . .] Therefore, `nanosleep()` pauses always for at least the specified time, however it can take up to 10 ms longer than specified until the process becomes runnable again.*

This scheduling issue also applies for processes that use RTC-based timers as UDP-based transport protocol (UDT). These mechanisms provide a coarse-grained limitation.

Mechanisms based on TCP self-clocking: TCP self clocking depends on the RTT of the path and is subject to variations, but ranges from about a hundredth millisecond within a LAN to several hundreds of milliseconds for worldwide connections.

[1]Real Time Clock

Since TCP is self-clocked by its acknowledgment mechanisms, the limitation mechanism based on the TCP congestion windows is thus clocked on the RTT. TCP can not send more than its congestion window over a period of time of one RTT. We can thus limit the congestion window value to control the rate at the RTT timescale. This solution requires a precise knowledge of the RTT and only provides a controlled rate at the RTT timescale, which can slightly vary.

Mechanisms based on the OS's kernel timer: OS's kernel timers depend on the kernel implementation and can be set up to 1000 Hz under recent GNU/Linux kernels.

The GNU/Linux kernel uses an internal time source that raises an interrupt every `HZ`. This value can be set between 100 Hz and 1000 Hz. GNU/Linux provides several qdisc to control the rate limitation. These mechanisms use HZ clocking. Note that HTB (Hierarchical Token Bucket) provides a classful limitation mechanisms.

Packet clocking based mechanisms Packet-level clocking depends on the underlying link. Gigabit Ethernet carries a single 1500-byte packet in 12 μs.

OS-based timers can use precise hardware timers through interrupts, but because interrupt handling takes time, they can not be set to a duration corresponding to the transmission time of a packet. The most precise solution to enforce bandwidth limitation is to increase the packets' departure intervals. This solution is known as pacing. Although hardware solutions were proposed in [Kobayashi, 2006], generic Gigabit Ethernet NICs do not provide pacing.

Software implementations, like PSPacer [Takano et al., 2005], also exist for packet pacing. PSPacer introduces Ethernet PAUSE packets between packets carrying data. These PAUSE packets are discarded at the first switch encountered. They just introduce spacing between real packets, thus limiting the effective rate. PSPacer is implemented as a GNU/Linux kernel qdisc but does not use any timers at the cost of a higher PCI bus bandwidth utilization. PSPacer actually uses a byte-clock by counting bytes sent and defines the departure date of a given packets in terms of "byte" time.

These time sources allow the creation of different sending patterns as shown on Figure 7.5. The upper figure gives a schematic view of a RTT-based rate limitation, the lower ones packet-level rate limitation, and in-between timer-based rate limitation. These different rate-limitation patterns generate different burst types, which impact the global performance. Consequently, even if the rates of two flows are limited to ensure the sum does not exceed the available bandwidth, bursts of packets can lead to losses. The instantaneous aggregated rate, being larger than the output rate, may overflow the intermediate buffers. These losses generate retransmissions and TCP's congestion window moderation.

These different time sources can support different rate-limitation mechanisms. In the absence of any rate-limitation mechanism, packets are sent in bursts due to TCP self-clocking.

Summary: To conclude this section, we summarize in Table 7.1 the time resolution of the different rate limitation mechanisms. Since for a given average rate, a bunch of packet has to be sent at every time event, this gives the size of the burst sent at the wire rate.

We note that pacing is a good candidate to satisfy **P3**. In next section, we compare some instances of these rate limitation mechanisms.

7.3.2 Single Step

Lets start on a single-step profile.

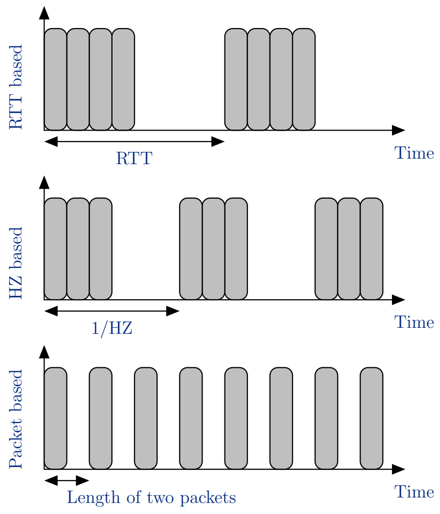
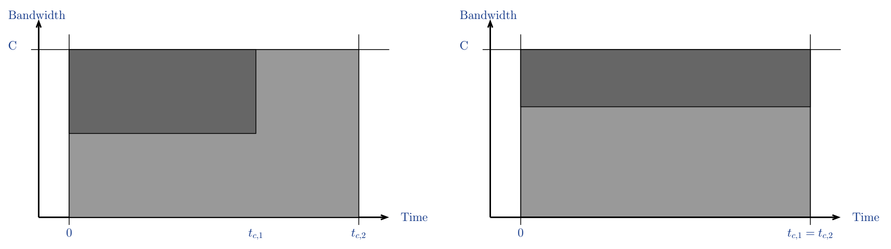


Figure 7.5: Different sending patterns.

Source of time	Userland timers	RTT	Kernel timers	Packet clocking
Resolution	< 8192 Hz	Vary with path latency	1000 Hz	$1/12\,\mu s$ = 83 kHz (@ 1 Gbps)
Drawbacks	Can be missed. High frequency of interruption	Can reach tens of ms	Burst of about 100 packets @ 1 Gbps	Buses bandwidth if done in software

Table 7.1: Comparison of rate limitation mechanisms.



(a) TCP sharing. (b) Rate limited.

Figure 7.6: Different solutions to realize the transfers.

7.3.2.1 Rate Control

As seen in the previous section, different solutions to implement rate control exist. We have studied some of them in [1]. In this work, we have compared solutions to implement a single step. The setting was the following: two flows sharing a bottleneck, one with a file to transfer two times bigger than the other and different settings of latency for the shared link. The different solutions studied were:

- TCP congestion control: in this setting, TCP congestion control was used to realize the sharing (basically a fair sharing until the smallest flow finishes);
- Rate limitation of the flows at sources according to the sharing ratio on the interval. Two different queue schedulers have been compared: Hierarchical Token Bucket (HTB) and PSPacer (PSP). The first qdisc implements a Deficit Round-Robin, while the second one paces every packet by inserting spaces between each packet of the data transfer.

The sharings utilized by these solutions are represented on Figures 7.6(a) and 7.6(b). In these figures, $t_{c,1}$ and $t_{c,2}$ are the completion times of the first and second transfers.

Essentially, the first setting using TCP is used as a reference. The objective of this study was to determine which solutions are suitable or not to implement the rate limitation in FLOC.

The conclusion of this study under various latencies—from 0 ms to 100 ms of RTT—shows that only the rate limitation at the source with PSPacer gives predictable completion time of the two transfers. In this study, the predictability has been studied by looking at the 99-percentiles of completion time over 100 repetitions of each experiments. Figure 7.8 shows the distribution of completion times observed over the repetitions for the latencies.

Figure 7.7 shows the 75- and 99-percentile of these distributions. It can be observed that for PSPacer, the difference between these two percentiles is not observable on the figure while it is very clear for the other solutions.

Using PSPacer, the relative deficiency of 99-percentile compared to ideal completion time—using the ideal transport protocol defined earlier—varies from 3.2 % under no latency and 10 ms of RTT to 3.9 % when the RTT is 100 ms. As a reference, relative deficiency of the 99-percentile with TCP sharing grows as 2.8 %, 16.5 % and 77.0 % of the ideal completion time with the three considered latencies. Using rate control at the source with HTB, it also reaches 66.2 % for 100 ms.

The difference of performance of these approaches is explained by the moderation of the maximum burst size and peak rate which avoid congestion and losses and then permit to realize isolation of flows.

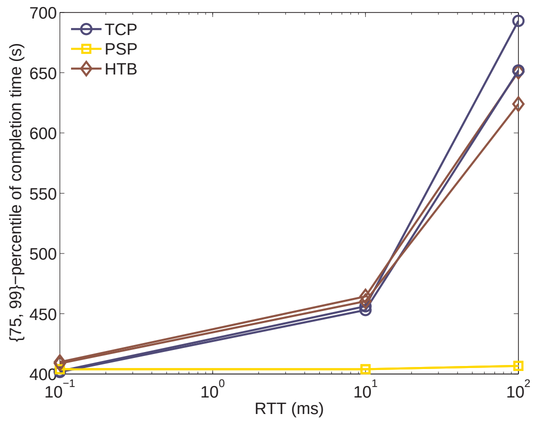
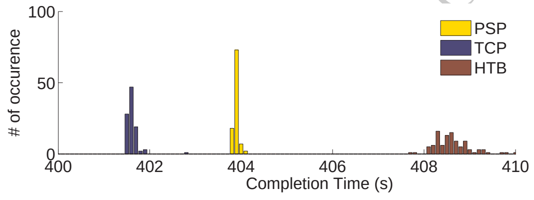
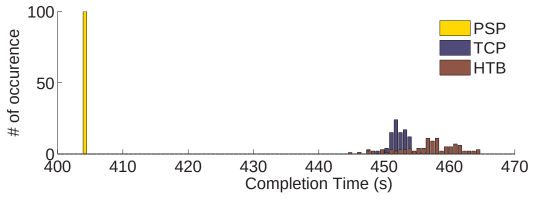


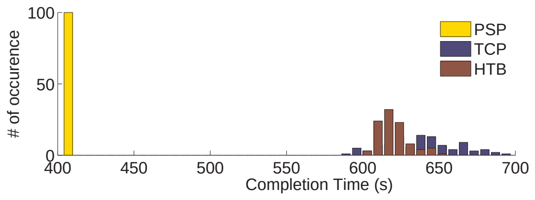
Figure 7.7: 75- and 99-percentiles of completion times.



(a) 0.1 ms RTT.



(b) 10 ms RTT.



(c) 100 ms RTT.

Figure 7.8: Distribution of completion time with and without rate limitation using BIC TCP under different RTT (100 repetitions).

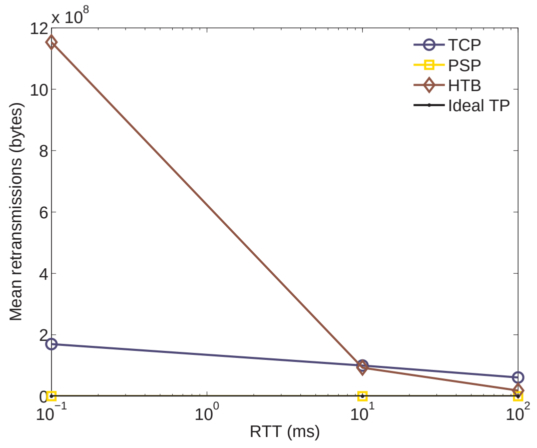


Figure 7.9: Mean retransmitted bytes as a function of the RTT (10 repetitions).

As shown on Figure 7.9, the number of retransmitted bytes is high under TCP or HTB. Since with a legacy TCP, retransmissions occur due to losses and losses induce reduction of the congestion window, it also shows that the sending rate has been reduced during the transfer. TCP with PSP does not show this.

As a conclusion to this section, packet pacing of TCP flows is a solution to implement rate control. Since it prevents bursts, it leads to predictable performances which, in addition, are not sensitive to latency. Before going to rate control on multi-step profiles, we will discuss the synchronization of transfer and control of time.

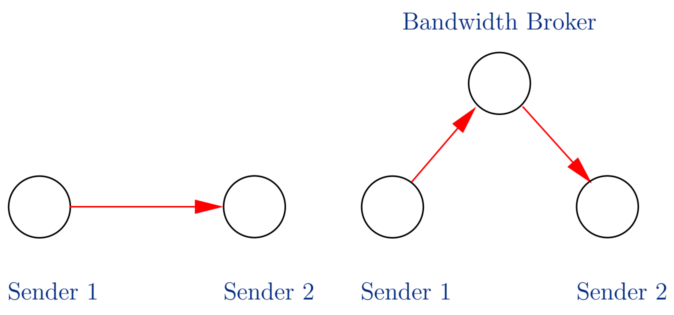
7.3.2.2 Time Control

Synchronization mechanisms are also required to enforce transfer start times. Two main solutions are available to control the time: time synchronization (real time) or explicit synchronization through signaling (virtual clocks). They both introduce a synchronization error. This error has to be much smaller (*e.g.,* less than x %) than the difference between the minimum and maximum completion time of a transfer. For example, in a 1 Gbps real dedicated network, the difference between minimum and maximum completion time of the 15 GB file transfer with a single flow TCP-based protocol varies from 340 ms at 0.1 ms RTT to 9.3 s at 100 ms RTT.

The first solution relies on the time accuracy several machines can obtain from a time source. As attaching a GPS device to each machine is not practical, NTP-like synchronization of the clock will have to be considered. In [Mills, 1994] published in 1994, NTP synchronization over a LAN is said to have an accuracy ranging from 500 μs to 2 ms representing 0.14 % (resp. 0.59 %) of the difference between the maximum and minimum completion times of a 15 GB file transfer time with 0.1 ms RTT on a 1 Gbps link.

The second solution to provide synchronization is to use signalization. Both direct and indirect signalization can be used. With the direct signalization method, the first sender signals an event to the next sender (for example when the transfer is done as shown in Figure 7.10(a)). Using indirect signalization, each sender exchange signals through a bandwidth broker as shown on Figure 7.10(b).

As a robust signalization mechanism is needed, a three-way handshake should be used.



(a) Direct Signalization. (b) Indirect Signalization.

Figure 7.10: Signalization scenarios.

For example, the mechanisms used for the establishment of a TCP connection ensures a signalization resilient to packet loss. This basically means that the cost of the direct signalization will be in the order of $1.5 * RTT$ (due to SYN-SYN/ACK-ACK) where RTT is the path's round-trip time between two senders plus the time to cross the local network stacks and the processing time.

In order to evaluate this cost, we measure the time required to perform two signalizations (one in each directions) between two machines and then divide this time by two. Figure 7.11 shows the distribution of this duration on a 0.1 ms RTT, 10 ms RTT, and finally 100 ms RTT network. As expected, measures show that the cost grows linearly with a 1.5 rate from 0.374 ms at 0.1 ms RTT to 150.4 ms at 100 ms RTT. These durations represent respectively 0.11 % and 1.6 % of the difference of completion for the previously considered 15 GB transfer at 0.1 ms and 100 ms RTT. Thus, the cost of direct and indirect signalization is still a small part (less than 2 %) of the variations of transfer completion times if the processing time is assumed to be small.

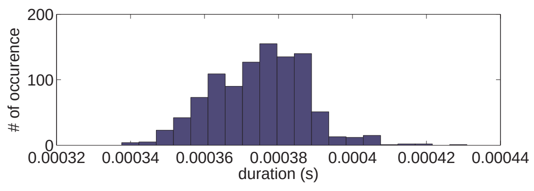
We conclude from this section that synchronization by NTP and direct or indirect signalization do not introduce much overhead and unpredictability to the completion time for the file size considered in this paper. But as the synchronization cost is independent from the size of transferred files, when this size decreases, the relative cost of time control will increase and will not be negligible with respects to the transfer completion time variability and will have to be considered.

In the next section we present how profiles can be enforced using a modified version of TCP and pacing.

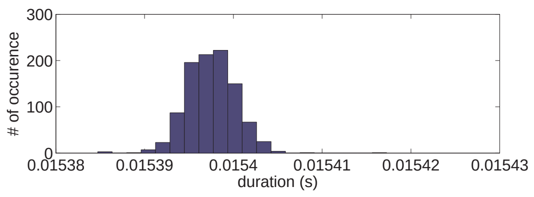
7.3.3 Profile

In the previous section we have seen that for a single step, TCP with pacing is a correct solution. In addition, we have also seen that time synchronization is not an issue. A profile can be seen as a succession of steps. In this section we focus on the transition between steps of different rates. While, for a single step, the TCP Slow-Start mechanism makes the sending rate quickly fill the bandwidth enforced by the qdisc, when there is change of rate, the congestion control algorithm of TCP is used.

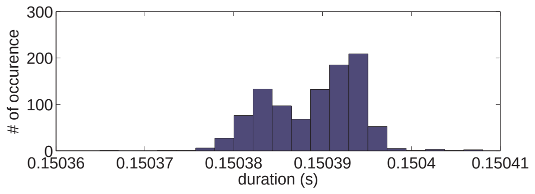
To be more precise, since there is no loss in the network (there can be due to transmission error but they are not due to congestion), the congestion window reaches the total amount of queueing between the sender and the receiver. Then it is never reduced because of source



(a) 0.1 ms RTT.



(b) 10 ms RTT.



(c) 100 ms RTT.

Figure 7.11: Distribution of direct-signalization time (1000 repetitions).

rate limitation which leads to filling the transmit buffer before the qdisc. This generates *send stall* events that truncate the congestion window instead of reducing it.

When the rate of the profile is decreased, there is no problem to adapt the sending rate since this is achieved solely by the qdisc. When the rate of the profile increases, the packets stored in the transmit buffer will first be sent a the new rate. Once the buffer is empty, the qdisc has to wait for new packets to be generated and queued by the TCP layer. This process is limited by the waiting of reception of ACKs that will trigger the packet generation process and by the congestion window which can limit the number of packets that can be sent. The updating of the congestion window takes time and so does the convergence to the new sending rate.

Two approaches have been studied, one based on TCP and a modified congestion control algorithm and one based on UDT Blast, a rate-based transport protocol. Details are given in the next two sections.

7.3.3.1 TCP without Congestion Control

It consists in the normal TCP implementation of TCP in the GNU/Linux kernel with two modifications. First, every function of the `struct tcp_congestion_ops` keeps the congestion window constant by continuously setting the congestion window to 30000 packets. As a side effect it also removes the Slow-Start phase.

This setting provides a congestion windows large enough to adapt rate increases from 0 to 1000 Mbps up to 360 ms of RTT since TCP will maintain 30000 packets ready to be sent without needing to receiving ACKs (provided the transmit queue is large enough to store them). The second modification is that, in the function `tcp_select_initial_window` of the receiver's kernel, `rcv_wnd` is not reset to the initial receiver window value (3.mss) when it is too large during the slow-start phase. This enables the sender to send at the maximum rate from the start of the connexion (except before receiving the first ACK as the scaling of the advertised receiver window is not possible before). Basically, it is designed as a hack of the TCP stack that disables the congestion control and sets the congestion control window to a large value. It is referred throughout this chapter as TCP.

To limit the bandwidth used by this TCP without congestion control, the PSPacer kernel module is used. It allows a precise pacing of the packets on the emission's site. To allow fast transition from a low rate to a high one, the `txqueuelen` variable is set to a very large value (100 000 packets) so that packets can be sent upon rate changes without needing to packetize new data.

7.3.3.2 UDT

UDT [Gu and Grossman, 2007] is provided as a library built on top of the UDP transport protocol and adds a congestion control scheme and a retransmission mechanism. It was designed to provide an alternative for TCP in networks where the bandwidth-delay product is large.

UDT Blast is one of the rate-control mechanisms proposed by UDT. It controls the sending rate of the transport protocol. We have modified the UDT used by the *gridftp* program so that it follows a rate profile: a set of dates according to a relative timer and the corresponding rates that it is allowed to send at. This has been achieved by using the CUDPBlast class of UDT4 implementation and an extra thread which reads the profile file and calls the `setRate(int mbps)` method on time. The performance of this version in terms of maximum achievable throughput is strictly identical to the original UDT *xio* module. It was done so to allow a fair comparison between the TCP and the UDT tests.

Since both the aforementioned adapted transport protocols have been implemented in the same file-transfer tool, it will be used as a comparison tool in next section.

7.3.4 Performances of Multi-Step Rate Control

We now go to multi-step profiles.

7.3.4.1 Methodology

Model Before going to the scenarios, we present the simple model we consider to express the rates (goodput/throughput) obtained from the two different transport protocols for a given input. This model will also be used to control the throughput to value determined by the BDTS scheduler (with an open loop control).

Figure 7.12 shows the two blocks of the considered model. TP is transport protocol (along with rate limitation and file serialization mechanisms). It takes as an "input" a rate and gives as an output a "throughput" and a "goodput". The first one being the bandwidth used on the wire and the second one the transfer rate of the file.

Since UDT and TCP do not use the same mechanism to control their sending rates and as they have a different packet format, the input signal does not have the same meaning for both of them.

The second block of this model aims at determining the input rate that must be specified to profile enforcement mechanisms to attain the required throughput. This target throughput is specified by the means of the "reference". We used throughput-like (instead of goodput-like) references since shared resources, namely the links or bottlenecks, constrain the throughput that the flows can share.

We have used linear models parametrized by the latency to model the different transfer functions.

Model Calibration The model has been fitted using the following scenarios on a dumbbell topology with a latency emulator and throughput monitor: GtrcNet-1. These scenarios are based on a systematic exploration of input rate space under different RTT for a single rate

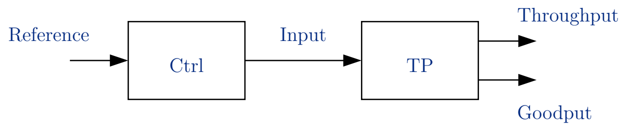


Figure 7.12: Model

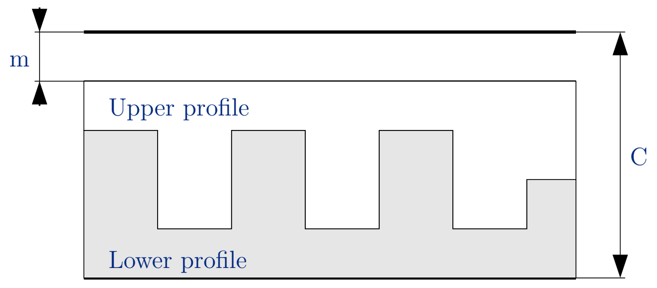


Figure 7.13: Ideal profiles with link capacity C and margin m.

profile. Input rate ranges from 1 to 1000 Mbps by steps of 1 Mbps. RTT is set to 0 ms, 10 ms, 100 ms and 300 ms.

Model Evaluation The results from the previous section will provide us with relations between "input" and "goodput"/"throughput". They will be used to predict the completion time of a transfer under a specified profile. The transferred file has a size of 3200 MB.

But before this, this model is used to predict the completion time of a single rate transfer under different RTT and at different rates. The metric used in this scenario is lateness as the model is supposed to predict the completion time.

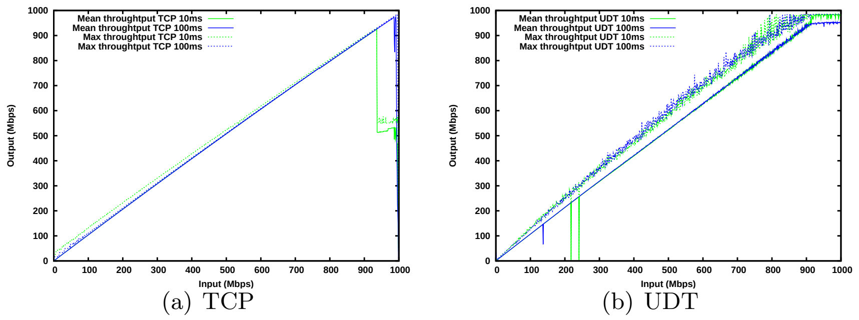
Two complementary profiles are used in this section. These profiles are presented in Figure 7.13. It can be noted that a margin m is introduced as a fraction of the bottleneck capacity that none of the flows are supposed to use. We will vary this to evaluate the contention when the profiles are run together. Both profiles end with a step at the same rate. Both transfers finish on this step and we can compare the completion time as the average allocated bandwidth remains the same for both since they entered the final step.

In order to evaluate the suitability of our model to the profiles, two transfers with complementary profiles take place at the same time. Lateness is used to compare the performances of our prediction under different margins m ranging between 1 and 10 % for both TCP and UDT. This experiment takes place on an inter-site link of Grid'5000 with 12 ms RTT. This latency is not one used to fit the model but is in the range covered by the fitting.

7.3.4.2 Results

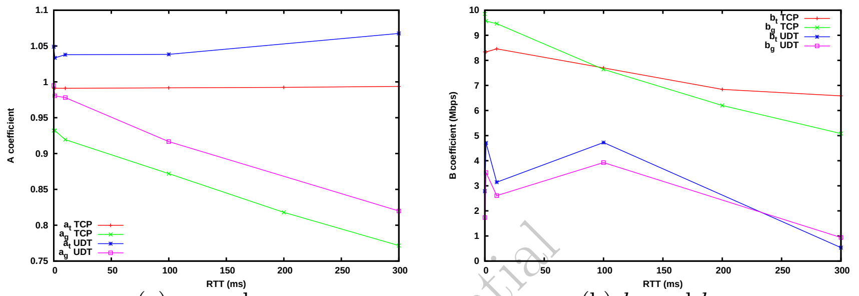
The main results are presented in this section and details can be found in [18].

Calibration Figures 7.14(a) and 7.14(b) present the mean and max throughput achieved by TCP, resp. UDT. The mean throughput for TCP and UDT are lines, but for TCP the maximum instant throughput is also a straight line, while for UDT it is wildly fluctuating, reaching values larger than 10 % of the mean throughput. This behavior could be problematic



(a) TCP (b) UDT

Figure 7.14: Maximum and mean throughput as a function of input rate under different latencies.



(a) a_g and a_t (b) b_g and b_t

Figure 7.15: Coefficients for TCP and UDT as a function of RTT.

when several flows are contending for a bottleneck as, at some instants, UDT can use more bandwidth than in average. It is more bursty than TCP with PSPacer.

We conclude from this that as expected, the solution using TCP and pacing is more suited to **P3**.

From the whole set of experiments, we derived the coefficient a_t,a_g, b_t and b_g of the linear regression of the model:

$$throughput(X, RTT) = \frac{a_t(RTT)}{a_g(RTT)}(goodput(X, RTT) - b_g(RTT)) + b_t(RTT)$$

and

$$goodput(X, RTT) = \frac{a_g(RTT)}{a_t(RTT)}(throughput(X, RTT) - b_t(RTT)) + b_g(RTT)$$

.

This formula is used to generate the appropriate values for the profile.

The resulting coefficients are shown in Figure 7.15(a) and 7.15(b). As these coefficients still depend on the RTT, we also need to model them as a function of RTT. As in Figure 7.15(a), the relationship between the coefficients and the RTT seems to be linear, we perform another linear regression.

The final model is good for TCP as the correlation coefficients are above 0.99 for the goodput and above 0.98 for the throughput. The modeling obtained for UDT is less accurate,

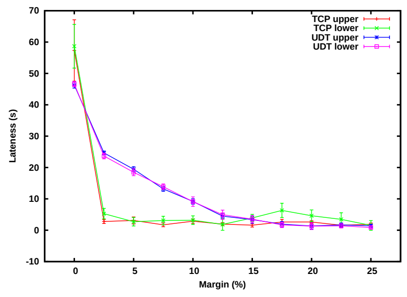


Figure 7.16: Lateness (95 % confidence interval) as a function of margin m for a 60 s transfers using TCP and UDT for 12 ms RTT.

especially for the b coefficients, but as they are rather small. This validates property **P1** for TCP with pacing.

Evaluation Using the previously established formula and fitting, we study the impact of sharing a bottleneck using step profiles as defined in Figure 7.13 and comparing predicted completion times using the model. This is done under 12 ms of latency. Figure 7.16 shows the lateness of the two transfers using the profile defined above for TCP and UDT with a 12ms RTT. It can be noted that TCP performs very well as soon as the margin is high enough to prevent losses. This threshold lies at 2.5 % of margin. Concerning UDT, we can observe that there is no such threshold and that the lateness decreases smoothly until 25 %, where it reaches 1 s. This has to be related with Figure 7.14(a) and 7.14(b) where the maximum instant throughput is shown. UDT's max throughput was always well above the mean throughput.

We also evaluated the lateness introduced by the increase of rate on a profile. It does not depend on RTT and is of 15 ms for both TCP and UDT. Since rate changes introduce lateness, **P2** is not strictly verified. But as this lateness is independent of the height of the jump and the RTT, since the maximum number of potential rate change is know at the time the BDTS optimization problem is formulated, the total deficiency can be bounded and integrated in the volume to schedule.

7.3.5 Conclusion

The affine model proposed to predict the completion time is a function of RTT and the throughput of the profile. The model is accurate for TCP as soon as the margin is more than 2.5 % of the capacity. However, the model is less accurate for UDT as its throughput is more bursty. This causes the instant throughput to be significantly higher than the predicted mean throughput. Still, the lateness is less than 1 s on 60 s transfers when the margin is higher than 15 %. We also showed that the cost of transition is independent of RTT and the steps' height. This cost, expressed in terms of lateness, has been evaluated to be between 10 ms and 20 ms each.

Summary

We conclude this chapter by highlighting that the requested properties for an ideal transport protocol in Section 7.3.1.1 are either satisfied by TCP without congestion control and pacing, or can be circumvented. UDT also satisfies this if the capacity margin is large enough to prevent congestion due to bursts and instant throughput.

Thanks to the properties P1 to P3, we can then use an affine function to go from the throughput to the goodput in the formulation of BDTS optimization problem and obtain allocations that can be used to satisfy the intend specified in the requests: transferring a file of a given volume with some constraints on time and rates.

This approach doesn't require any support from the network, only adaptation of the transport protocols at the sender and receivers. In the eventuality where the network provides more support as discussed in [RE09] or [MEF09], a similar approach can also be applied regarding the availability of ready-to-be-sent packets. In this case, the isolation between flows would be provided by the network itself, since it would provide the appropriate pacing between packets throughout the network.

Q7 How can the allocations be enforced?

A7 *The allocated profile is distributed through the control plane to the network elements of the path that can then configure the resources for the reservation. Using a modified version of TCP that quickly reacts to variations of the available bandwidth, the sender can exploit the allocation.*

— Eight —

Extending BDTS Approach to Dynamic Bandwidth Provisioning

Introduction

Large-scale distributed scientific applications, but also industrial data-intensive ones, require the use of high-performance network infrastructures. For example, collaborative engineers need to interact with massive amounts of data to analyze the simulation results over high-resolution visualization screens fed by storage servers during time-limited working sessions. To address this anticipated sporadic terabyte demand, dynamically reconfigurable optical networks have been explored in labs and testbeds. One of the directions recently investigated in large-scale distributed computing and data-processing systems is the possibility of dynamically establishing dedicated lambda paths to support the execution of these applications as presented in Chapter 3.

[Jukan and Karmous-Edwards, 2007] suggests making network reconfigurations available to application users by making all the resources visible, and allowing them to send signaling messages in the carriers' networks. This approach raises some confidentiality concerns but can be applied to specific distributed applications, like e-science, deployed over National Research and Educational Networks (NREN).

To develop and expand these facilities to an industrial context [Audouin et al., 2007], new management and control functions are required to adapt the existing Telecom network infrastructures to deliver commercial IT services for customers or companies. In the currently proposed frameworks and interfaces [NSI09], users will express their bandwidth requirements to the network provider in terms of rate and deadline. A commercial interface will include the maximum cost users are willing to sustain. On the other side, the network providers will publish their services according to their policies and negotiations to maximize utilization rates.

However, the granularity that a realistic and efficient optical bandwidth reservation service can offer, typically of the order of gigabits per second, is not flexible enough to meet the heterogeneity of the end-users' requirements. This may lead to poor resource utilization, over-provisioning, and unattractive service pricing.

To mitigate this, we propose to adapt the offered service to the real needs of the users, by providing them with an interface allowing them to express their constraints on the allocation, possibly in a malleable way.

The key idea is to mix rigid requests and malleable ones. We consider rigid requests—*e.g.,* for real-time video and audio conferencing applications—that specify bandwidth provisioning while malleable demands are for time-constrained large data transfers. An intermediate service, similar to BDTS, is in charge of carrying out and grouping giant transfer tasks (with a volume greater than several GB) in specified time intervals. It aggregates the malleable requests and provisions the bandwidth accordingly.

Such a service considers that most users will care about transferring a large amount of data in a limited and predictable time frame rather than requiring an exact fixed rate for a given time window. This approach relieves the end user from the burdens of bandwidth reservation and allocation, while ensuring flow-completion in a timely fashion.

Users express their malleability through bandwidth-provisioning requests with minimal rate, maximal deadline, but also volume and maximal achievable rate. The goal of this work is to demonstrate that the flexibility proposed increases the efficiency in lambda-path usage. In particular, we study how the network provisioning service, with the help of the data mover service, can serve different traffic distributions in an efficient way.

The remainder of this chapter is organized as follows. Section 8.1 defines the model and formulates the problem. In Section 8.2, simulation results are presented to demonstrate the impact of the homogeneity of the request's parameters such as volume, rate, and patience, as well as the influence of the proportion of malleable requests in the mix. We conclude in Section 8.2.6 on rate granularity and the limitations of the proposed approach.

8.1 Model and Problem Formulation

Let us consider a network model defined as a cloud, as shown on Figure 8.1, owned by a *network operator* (NO), exposing a *Dynamic Bandwidth Provisioning service* that provides on-demand lambda paths or provisioned links between a set of end points and peering points with other network clouds. The network is defined by its set of points of presence $s \in \{s_1, ...s_S\}$ where S is the number of exposed points which are supposed to be a subset of the edgepoints of what the network operator will propose to the service provider.

We assume here that any two points exposed by one cloud can be the source and destination of a reservation. On top of this bandwidth provisioning service, the *service provider* (SP) offers *users* (U) a service of scheduled and guaranteed data transfers from one end point to another and a service of bandwidth provisioning.

The SP that schedules user requests tries to maximize resource usage. As a different economic agents from the NO, the SP does not have access to the routing plane and hence can only manage movement capacity or bandwidth capacity between the network points of

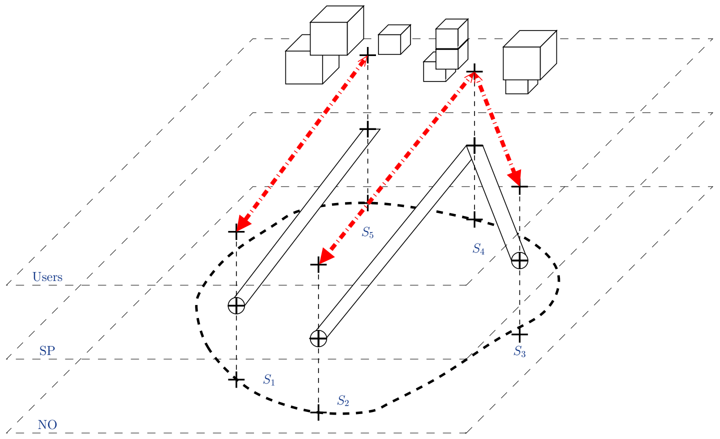


Figure 8.1: Network model: the NO discloses only the available sites, the SP can provision paths between two sites to serve user requests.

presence.

The three main actors in this model are then: the users that aim at transferring files from one site to another with strict completion deadlines or renting fixed bandwidth for a time window, the service provider that tries to maximize resource usage by scheduling user requests and provisioning network paths. The network operator provides the lambda path service.

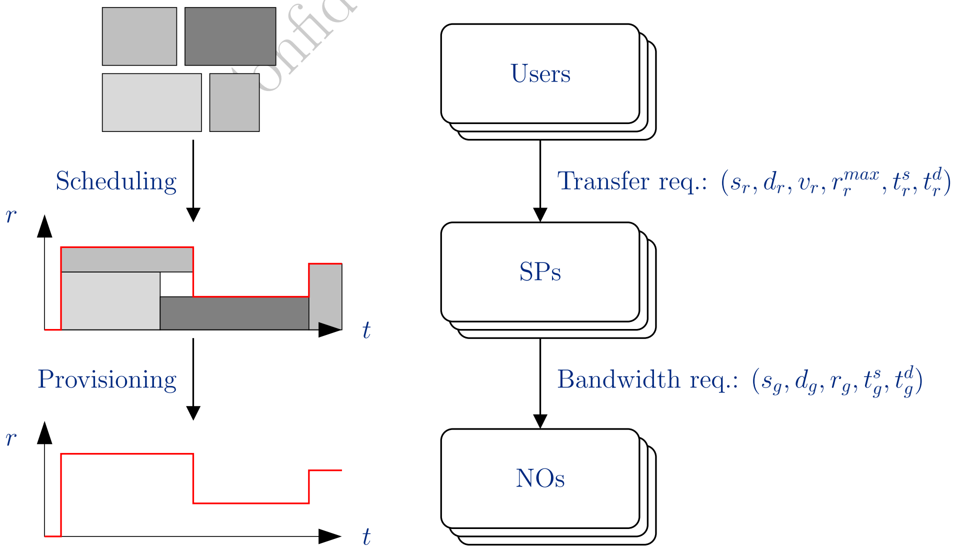

Scheduling
Provisioning
r
t
r
t
Users
SPs
NOs
Transfer req.: $(s_r, d_r, v_r, r_r^{max}, t_r^s, t_r^d)$
Bandwidth req.: $(s_g, d_g, r_g, t_g^s, t_g^d)$

Figure 8.2: Transfer and bandwidth requests exchanged between actors. The SP groups user requests in bandwidth requests issued to the NO.

Figure 8.2 shows the requests format and the relations between actors. Users issue transfer requests to the SP, which groups them and issues bandwidth requests to the NO based on the bandwidth and time window requirements of each group.

$(s_r, d_r, v_r, r_r^{max}, t_r^s, t_r^d)$	user to SP transfer request r
(s_r, d_r)	source-destination pair
v_r	volume
r_r^{max}	maximum rate
$[t_r^s, t_r^d]$	reservation window
$t \mapsto p_r(t)$	bandwidth allocation profile of r
r_r^{min}	minimum constant rate to serve r
P_r	patience of r
$g^{m,n} = (s_g, d_g, r_g, t_g^s, t_g^d)$	bandwidth request issued by SP to NO on period m for period n
(s_g, d_g)	source-destination pair
r_g	requested bandwidth
$[t_g^s, t_g^d] = [n.\delta, (n+1).\delta]$	reservation window

Table 8.1: Request-related notations

8.1.1 Requests and Constraints

The *transfer requests* sent by users to the SP are defined as follows:

Definition 8.1.1 (Transfer Request). Each request r is a 6-tuple $(s_r, d_r, v_r, r_r^{max}, t_r^s, t_r^d)$ where s_r is the source, d_r is the destination, v_r is the volume to transfer, r_r^{max} is the maximum rate the sender can send data at.

The notations are summarized in Table 8.1 and 8.2.

A transfer can only start after t_r^s and must be finished before t_r^d (reservation window). t_r^a is the arrival date of request r and t_r^r is the date of the request's acceptance decision. The minimum constant rate r_r^{min} is defined as:

$$r_r^{min} = \frac{v_r}{t_r^d - t_r^s}.$$

We also define the *patience* P_r as:

$$P_r = \frac{r_r^{max}}{r_r^{min}}.$$

Note that users can also ask for fixed bandwidth by tailoring the request to the SP with no patience ($P_r = 1$). A *bandwidth-specified user request* of rate r_r between t_r^s and t_r^d is thus specified as a 5-tuple $(s_r, d_r, r_r, t_r^s, t_r^d)$ rewritten by the SP as the 6-tuple: $(s_r, d_r, (t_r^d - t_r^s).r_r, r_r, t_r^s, t_r^d)$. Therefore, the SP exposes two different services with two request formats: one for malleable transfers, one for bandwidth-on-demand.

Now consider the interactions between a SP and a NO. The *bandwidth requests* sent to the NO are not malleable and are defined as follows:

Definition 8.1.2 (Bandwidth Request). A bandwidth request g is a 5-tuple $(s_g, d_g, r_g, t_g^s, t_g^d)$ where s_g is the source, d_g is the destination, r_g is the requested rate, t_g^s is the start time, and t_g^e the end of the reservation. Similarly to transfer requests, t_g^a is the arrival date of request g. We assume that bandwidth is provisioned α in advance, by slots of duration δ.

Assumption 8.1.3. The bandwidth request issued for slot n is constrained to have: $t_g^a \leq t_g^s - \alpha$, $t_g^s = n.\delta$ and $t_g^e = (n+1).\delta$.

$[t_r^s, t_r^d]$	reservation window of request r
t_r^a	submission time of request r
δ	duration of a period
α	advance in time required to reserve/provision bandwidth
$[t_g^s, t_g^d] = [n.\delta, (n+1).\delta]$	reservation window of bandwidth request $g^{m,n}$
$t_g^a = m.\delta - \alpha$	submission time of bandwidth request $g^{m,n}$

Table 8.2: Time-related notations

In the remainder of this chapter, we assume a negotiation process that allows the SP to choose *a priori* the NO they want to use. Thus, the interactions between one given SP and one given NO are studied.

If this request is issued by the SP to the NO at $t_g^a = m.\delta - \alpha$ with $m \leq n$, it will be noted $g^{m,n}$. The final bandwidth request for slot n is noted: $g^{n,n}$. In order to avoid over-estimation of in-advance bandwidth reservations, we assume (Assumption 8.1.4) that a SP can only re-provision for a given slot, by increasing the requested rate.

Assumption 8.1.4. At time $m^{'}.\delta - \alpha$, when updating the advance bandwidth requests previously made at $m.\delta - \alpha$ for slot n, the SP can only increase the requested bandwidth. More formally: $\forall m < m^{'} \leq n, r_{g^{m,n}} \leq r_{g^{m',n}}$

This is illustrated on Figure 8.3 where new bandwidth requests for slots n and $n+1$ issued at time $n.\delta - \alpha$ are shown in plain lines while old bandwidth requests are dashed.

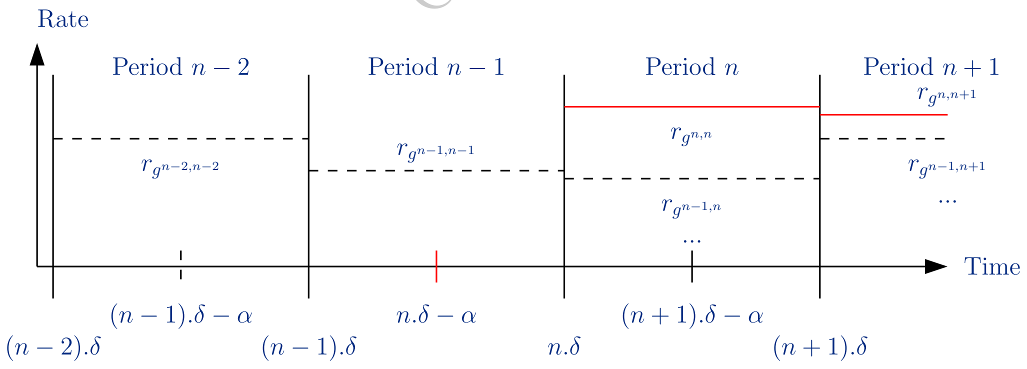


Figure 8.3: Bandwidth provisioning slots $n-2$ to n seen at time $n.\delta - \alpha$

Assumption 8.1.5. The SP can only rent end-to-end bandwidth resource to the NO. The SP does not have routing facilities.

As previously stated, Assumption 8.1.5 is based on realistic situations in which network operators do not provide routers, nor direct access to routing, to their customers. This implies that there is no routing opportunity from the SP's point of view. Routing issues are addressed by the NO. This also avoids the need to expose detailed topological information of the core network.

We define the *bandwidth allocation profile of* r as a step-function $t \mapsto p_r(t)$ defining the rate allocated to this transfer over time. A valid *bandwidth allocation profile* p_r must verify constraints 8.1, 8.2, 8.3.

$$\forall t \in [t_r^s, t_r^d], 0 \leq p_r(t) \leq r_r^{max} \tag{8.1}$$

$$\forall t \notin [t_r^s, t_r^d], p_r(t) = 0 \tag{8.2}$$

$$\int_{t_r^s}^{t_r^d} p_r(t)\ \mathrm{d}t = v_r \tag{8.3}$$

a_r^s is the actual start time of transfer r and a_r^f its actual finish time. More formally, $a_r^s = \min\{t | p_r(t) \neq 0\}$ and $a_r^f = \max\{t | p_r(t) \neq 0\}$.

In this model, the SP issues bandwidth reservations with the same source and destination sites as transfer requests. Bandwidth reservation have to satisfy the following constrains to support transfer requests. Let's consider a set of N transfer requests $R = \{r_1, \ldots, r_N\}$ and a set of M non-overlapping[1] bandwidth requests $G = \{g_1, \ldots, g_M\}$; validity constraints are (in addition to 8.1, 8.2 and 8.3 for each requests in R) the following:

$$\forall g \in G, \forall t \in [t_g^s, t_g^e], \sum_{r \in R} p_r(t) \leq r_g \tag{8.4}$$

$$\forall r \in R, \forall t \notin \bigcup_{g \in G} [t_g^s, t_g^e], p_r(t) = 0 \tag{8.5}$$

$$\forall r \in R, s_r = s_g \tag{8.6}$$

$$\forall r \in R, d_r = d_g \tag{8.7}$$

Figure 8.4 shows a group of transfer requests and the corresponding bandwidth request to serve them. Due to Assumption 8.1.5, and without loss of generality, the remainder of this work focuses on transfer requests with the same source/destination pair, as transfer requests with different source or destination do not interact from the SP's viewpoint.

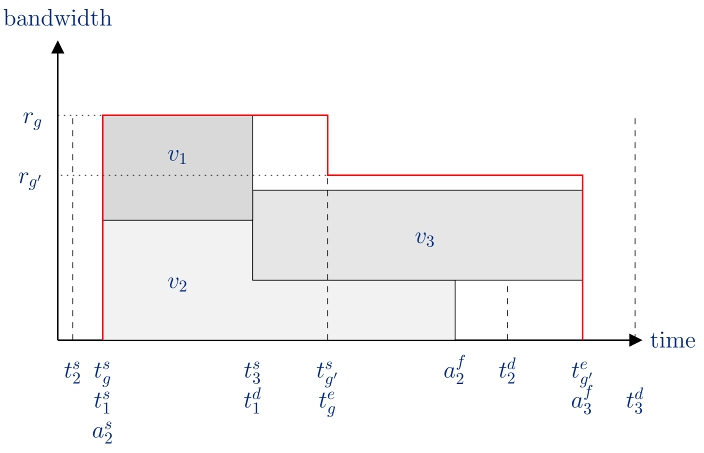


Figure 8.4: Transfer requests ($r \in \{1, 2, 3\}$) grouped in bandwidth requests g and g'

Before defining the states associated with transfer requests, we precise what we call *in-advance* requests—by opposition with *immediate* requests.

[1]$\forall g, g' \in G, (t_g^s, t_g^e) \cap (t_{g'}^s, t_{g'}^e) = \emptyset$.

Definition 8.1.6 (In-advance requests). As long as the date is less than $n.\delta - \alpha$, request r is said to be *in-advance* if its start time t_r^s is after $n.\delta$

Definition 8.1.7 (Immediate requests). As long as date is less than $n.\delta - \alpha$, requests r such that t_r^s is before $n.\delta$ are called *immediate* requests.

The basic distinction criteria is whether they are received before the decision regarding provisioning of the link during their time window has to be made or not.

8.1.2 Requests State Diagram

User transfer or bandwidth-specified requests are received and processed by the SP. The state diagram of requests comprises four states: (i) *New*, (ii) *Scheduled*, (iii) *Granted*, (iv) *Rejected*.

A request r is: (i) *New* when the request has just been received and is valid but has been neither accepted nor rejected, (ii) *Scheduled* when it has been accepted but allocated profile $t \mapsto p_r(t)$ can still be changed, (iii) *Granted* when it can not be changed anymore, and finally (iv) *Rejected* when the request is not accepted by the SP.

Theses states and allowed transitions are depicted in Figure 8.5. The state of request r is changed from *Scheduled* to *Granted* t_{grant} before t_r^s in order to give some time to the sender before transfer's start time and transitions *New* to *Scheduled* or *Rejected* depend on the decision taken by the SP when first considering this request.

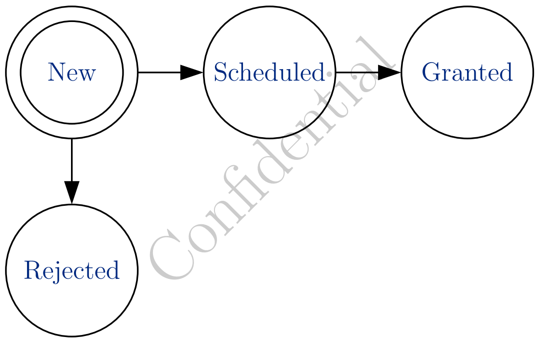


Figure 8.5: State diagram of the transfer requests: once *Scheduled*, a transfer can not be rejected anymore, but the profile can still change until the transfer reaches the *Granted* state.

8.1.3 Negotiation Processes

This section introduces the general negotiation processes of this proposal. This general presentation is then narrowed to match the specific focus of this work.

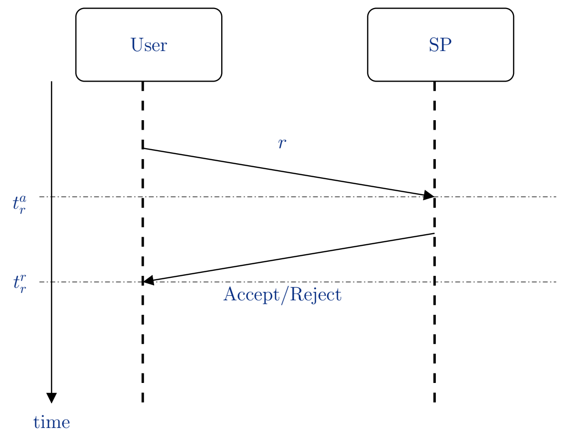
The negotiation process between Users and SPs is defined as follow:

(i) A User issues transfer requests to one SP

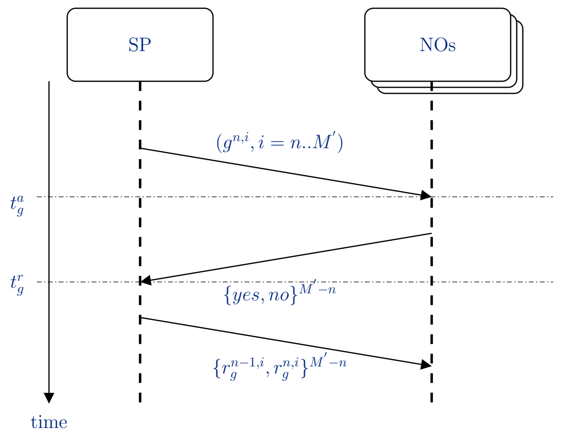
(ii) The SP decides to accept or reject the request and informs the user.

Figure 8.6(a) summarizes this process. Obviously, the SP's decision to accept or reject r has to happen before the transfer's start time t_r^s: $t_r^r \leq t_r^s$.

SP/NO negotiation is a similar process for bandwidth requests, as Figure 8.6(b) shows. But as transfer requests might span many slots, SPs have to issues several bandwidth requests to NOs, one for each slot. Each bandwidth request can theoretically be accepted or rejected by the NO which would require the SP to choose which requests he wants to cancel in this



(a) Users/SPs interactions: Users submit transfer requests and SPs accept or reject them.



(b) SP/NO interactions: SPs submit their bandwidth requests for future slots to NOs, which answer "yes" if they can accept each request. If not, SPs have two options: they can keep the previous reservation for that slot or choose the new one.

case. Similarly to U/SP negotiation, the negotiation process has to be finished before the bandwidth reservation's start time.

Assumption 8.1.8. In this work, we assume that they are never rejected.

Assumption 8.1.8 implies that in-advance transfer requests can all be accepted. The SP has infinite resources, as demonstrated in the next section. This also implies the third phase of the negotiation in never used. This assumption is justified by considering that the SP only represents a small part of the bandwidth requests sent to the NO. In case this does not apply and the NO rejects one of the increases of provisioning, the SP has to consider which set of feasible increase better suits its needs, then inform the NO about its choices, and reject the requests that can not be served.

The next section will study a strategy for SPs to provision their virtual infrastructure to serve in-advance transfer requests.

8.1.4 Problem Formulation

For every slot n at $n.\delta - \alpha$, the SP has to decide for each source/destination pair which bandwidth request to issue for the next slot to accommodate the already known transfer requests R_n ($\forall r \in R_n, t_r^a \leq n.\delta - \alpha$) and how to schedule these transfers. We first formulate the problem under the assumption no rejection occurs. In this context, the objective is to minimize the aggregated provisioned capacity. If we take rejection into account, the problem becomes a maximization of a global reward. We discuss this extension in Section 8.2.6.

According to the previously defined state diagram, R_n is partitioned into three subsets $R_n = R_n^{new} \bigcup R_n^{sched.} \bigcup R_n^{granted}$ where $R_n^{granted}$ is the set of requests already granted during slots before $n.\delta - \alpha$, R_n^{new} contains new valid requests which have not yet been scheduled (or rejected) and $R_n^{sched.}$ contains transfer requests that can still be re-scheduled. It can be noted that requests in $R_n^{sched.} \bigcup R_n^{granted}$ were already in R_{n-1}. Figure 8.6 summarizes this. Notations are summarized in Table 8.3.

Let $G_n^{final} = \{g^{i,i} | 1 \leq i \leq n-1\}$ be the set of bandwidth reservation made for slots up to $n-1$ (they can not be modified anymore), $G_n^{prev.} = \{g^{n-1,i} | n \leq i \leq M\}$ and $G_n^{new} = \{g^{n,i} | n \leq$

R_n	set of all user requests know at $n.\delta - \alpha$
R_n^{new}	requests of R_n that have not yet been scheduled (received between $(n-1).\delta - \alpha$ and $n.\delta - \alpha$)
$R_n^{sched.}$	already scheduled requests
$R_n^{granted}$	requests with profile that can not be changed anymore
G_n^{final}	bandwidth requests that can not be changed anymore
$G_n^{prev.}$	bandwidth requests for upcoming slots as requests at $(n-1).\delta - \alpha$
G_n^{new}	bandwidth requests for upcoming slots as sent to NO at $n.\delta - \alpha$

Table 8.3: State-related notations

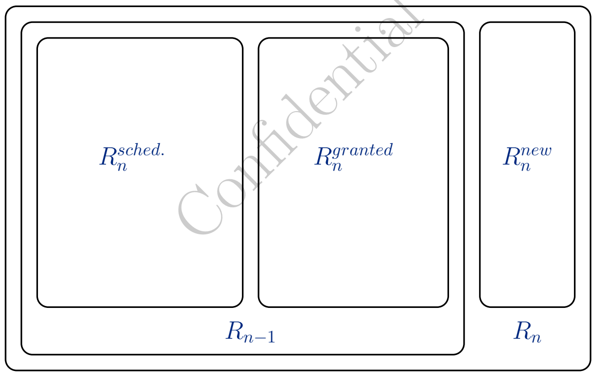


Figure 8.6: Partitioning of R_n

$i \leq M\}$ where M is the greatest slot utilized by a transfer request.

Using previously defined validity constraints for bandwidth requests and transfer requests and constraints on increasing bandwidth requests, we formulate the problem as:

$$
\begin{aligned}
BP(n): \quad & \text{minimize} \sum_{g \in G_n^{final} \bigcup G_n^{new}} r_g \\
& \text{s.t.} \\
& \forall r \in R_n, \forall t \in [t_r^s, t_r^d], 0 \le p_r(t) \le r_r^{max} \\
& \forall r \in R_n, \forall t \notin [t_r^s, t_r^d], p_r(t) = 0 \\
& \forall r \in R_n, \int_{t_r^s}^{t_r^d} p_r(t)\ \mathrm{dt} = v_r \\
& \forall g \in G_n^{final} \bigcup G_n^{new}, \forall t \in [t_g^s, t_g^e], \sum_{r \in R_n} p_r(t) \le r_g \\
& \forall t \notin \Big(\bigcup_{g \in G_n^{final} \bigcup G_n^{new}} [t_g^s, t_g^e] \Big), \forall r \in R_n, p_r(t) = 0 \\
& \forall i, n \le i \le M, r_{g^{n-1,i}} \le r_{g^{n,i}}
\end{aligned}
$$

The first constraint of this linear program states that the service rate must lie within the maximum sending rate constraint given by the user. The second constraint states the service rate must be null out of the reservation window while the third ensures the sum over time of the service rates equals the requested volume. The fourth constraint says that the bandwidth provisioned must be higher than the sum of the profiles of the served transfer requests, and the fifth constraint ensures transfer requests are not served out of a bandwidth request. The last constraint says that, from one negotiation to another, the bandwidth requested by the SP to the NO can not decrease.

Let $\mathcal{I}$ be the set of time intervals defined by dividing the time axis on all t_g^s, t_g^e, t_r^s, and t_r^d. In this case, $\gamma_{r,i}$ will be 1 if request r can be served on interval i and 0 else, $\gamma_{g,i}$) equals 1 if the bandwidth request g covers interval i and 0 else (all i are supposed to be covered by a bandwidth request g possibly with $r_g = 0$), l_i is the length of interval i and $p_{r,i}$ the constant rate of $p_r(t)$ on interval i. Then the problem BP(n) can be rewritten as a linear program:

$$
\begin{aligned}
BPLP(n): \quad & \text{minimize} \sum_{g \in G_n^{final} \bigcup G_n^{new}} r_g \\
& \text{s.t.} \\
& \forall r \in R_n, \forall i \in \mathcal{I}, 0 \le p_{r,i} \le r_r^{max} \\
& \forall r \in R_n, \forall i \in \mathcal{I}, (1 - \gamma_{r,i}).p_{r,i} = 0 \\
& \forall r \in R_n, \sum_{i \in \mathcal{I}} \gamma_{r,i}.p_{r,i}.l_i = v_r \\
& \forall i \in \mathcal{I}, \forall g \in G_n^{final} \bigcup G_n^{new}, \gamma_{g,i}. \sum_{r \in R_n} p_{r,i} \le r_g \\
& \forall j, n \le j \le M, r_{g^{n-1,j}} \le r_{g^{n,j}}
\end{aligned}
$$

where variables are: $\{p_{r,i} | r \in R_n^{new} \bigcup R_n^{sched.}, i \in \mathcal{I}\}$ and $\{r_g | g \in G_n^{new}\}$. $\{p_{r,i} | r \in R_n^{granted}, i \in \mathcal{I}\}$ is not part of the variable as the profiles of *Granted* requests can not be changed anymore.

It can be proved that, provided that requests in R_n^{new} are *in-advance* requests, BPLP(n) has a solution. This is the object of next proposition.

Proposition 8.1.9. *If requests of R_n^{new} are* in-advance *requests, BPLP(n) has a solution.*

Proof. We prove by induction that the solution space is not empty.

Let's assume BPLP$(n-1)$ has a solution. (i) All *New* requests are valid requests and thus have $P_r \geq 1$ meaning that a pure rectangle of r_r^{max} on $[t_r^s, t_r^d]$ can be used as their bandwidth allocation profiles. (ii) To serve these *New* requests in addition to the old ones, starting from the solution of BPLP$(n-1)$, it only requires increasing the bandwidth of the slots greater than n—this is allowed by Assumptions 8.1.4 and 8.1.8—, and reusing the profiles of the old solution for requests in $R_n^{sched.} \bigcup R_n^{granted}$.

This demonstrates that BPLP(n) has at least one solution without changing the profile of *Granted* requests. The first iteration BPLP(1) can obviously be solved. □

We can observe in the formulation of BPLP(n) that requests in $R_n^{granted}$ can have their profiles $p_r(t)$ be summed and processed as a single profile since they will not be changed anymore and request-centric constraints (8.1–8.3) have been verified for these requests in BPLP$(n-1)$. This would allow forgetting the past history of the allocations and prevent problems from growing at every iteration.

Once this problem has been solved, *New* and *Scheduled* requests are marked as *Scheduled* or *Granted* depending on their start time. $R_{n+1}^{granted}$ and $R_{n+1}^{sched.}$ can thus be prepared for the next slot while R_{n+1}^{new} is filled when requests arrive. Once new bandwidth-reservation requests have been sent to the NO, sets G_{n+1}^{final} and $G_{n+1}^{prev.}$ can be determined. The whole procedure for provisioning and scheduling transfers is depicted in Algorithm 3.

8.1.5 Complexity Analysis

This problem can be seen as a *min-cost multicommodity flows over time with uniform path-lengths problem* on a simple dumbbell graph. The cost function would sum the differences between the provisioned bandwidth and the bandwidth already allocated to transfers of $R_n^{granted}$. The dumbbell graph considered is made of input edges between the vertices representing the requests and the input vertex of the central edge. The transit times of these input edges are equal to the start time of the respective requests. Similarly, the transit times of output edges are equal to the difference between the largest deadline minus the one of the request considered. The central edge of the dumbbell has no transit time. The result of the problem we are interested in is the schedule of flows on the central edge. This problem has been proved to be solvable in a polynomial time in Theorem 1 of [Hall et al., 2007].

The problem enjoying a LP formulation runs in a polynomial time in the number of optimization variables. Given R the total number of requests considered per schedule period, and G the number of periods considered, there is at most $G + R(2(R+G)-1)$ optimization variables $(p_{r,i})$ since the number of intervals defined by start time and deadline and period is less than $2(R+G)-1$. The linear program has $6R(R+G)+3G-1$ constraints. Using the complexity of Karmarkar's algorithm for a fixed maximum size of variable, we obtain a complexity in $O((G+R(2(R+G)-1))^{3.5})$.

Another solution to solve this is to use a static min-cost multicommodity flow computation on an time-expanded networks of $O((2R+2)^2)$ nodes and $O((2R+2)(2R-1))$ links as proposed by Theorem 2 of [Hall et al., 2007].

If the bandwidth-allocation profile had been restricted to constant-value single-interval form, the problem would have been NP-Complete as demonstrated in [Chen and Vicat-Blanc Primet, 2007]. Therefore, allowing rational step functions and malleable requests is necessary to find a solution in polynomial time.

Algorithm 3 Schedule and provision at $t = n.\delta - \alpha$

```
Input: $R_n^{new}, R_n^{sched.}, R_n^{granted}, G_n^{final}, G_n^{prev.}$
Output: $R_{n+1}^{sched.}, R_{n+1}^{granted}, G_{n+1}^{final}, G_{n+1}^{prev.}, \{t \mapsto p_r(t) | r \in R_{n+1}^{sched.} \bigcup R_{n+1}^{granted}\}$
        // Initialize set of request for next slot
 1:     $R_{n+1}^{sched.} \leftarrow \emptyset$
 2:     $R_{n+1}^{granted} \leftarrow R_n^{granted}$
        // Determine profiles for transfer requests and bandwidth requests
 3:     Solve BPLP($n$)
        // Update transfer requests' states
 4:     for all $r \in R_n^{new} \bigcup R_n^{sched.}$ do
 5:         if $t_r^s - t_{granted} \leq (n+1).\delta - \alpha$ then
 6:             $R_{n+1}^{granted} \leftarrow R_{n+1}^{granted} \bigcup \{r\}$
 7:         else
 8:             $R_{n+1}^{sched.} \leftarrow R_{n+1}^{sched.} \bigcup \{r\}$
 9:         end if
10:     end for
        // Send bandwidth requests to NO
11:     Issue bandwidth requests $g \in \{G_n^{new}\}$ to NO.
        // Update bandwidth req. sets for next slot.
12:     $G_{n+1}^{final} \leftarrow G_n^{final} \bigcup \{g^{n,n}\}$
13:     $G_{n+1}^{prev.} \leftarrow G_n^{new} \backslash \{g^{n,n}\}$
14:     return $R_{n+1}^{sched.}, R_{n+1}^{granted}, G_{n+1}^{final}, G_{n+1}^{prev.}, \{t \mapsto p_r(t) | r \in R_{n+1}^{sched.} \bigcup R_{n+1}^{granted}\}$
```

If SPs have access to the routing and can compose bandwidth requests to serve transfer requests, as it has been done in Chapter 6, the optimization problem can be formulated using the routing matrix and solved in a manner similar to Algorithm 3.

8.2 Performance Evaluation

In the previous section we have unified malleable transfer requests and fixed bandwidth requests in a common format. We have formulated the Dynamic Bandwidth Provisioning problem BPLP. Then we have proposed and proved a linear program-based solution for any given set of malleable and fixed transfer requests. The goal of the performance evaluation is now to explore how time and rate granularity, as well as request malleability, impact resource provisioning and utilization. In order to evaluate the impact of requests' characteristics, we performed simulations with different workloads. The proposed provisioning algorithm has been implemented in jBDTS [jBD09] which is used for the simulations.

The rates proposed by NOs are generally discrete. Therefore, in the following simulations, Algorithm 3 is compared to a modified version of it which uses BPLP as a linear relaxation of the mixed integer linear problem with discrete value for r_g. In this modified version, rounding

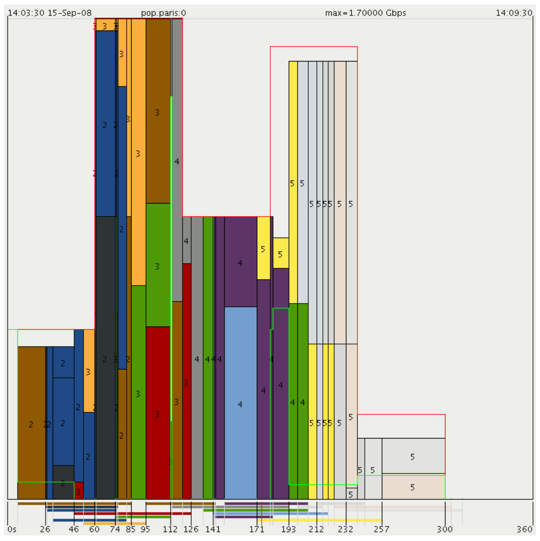


Figure 8.7: Example of scheduling of random transfers scheduled by `jBDTS`. Transparent transfers are *Scheduled* while others are *Granted*. The red line represents the provisioned capacity after rounding to the upper 100 Mbps, and the green one the unused capacity. Tight colored lines at the bottom of the picture represent $[t_r^s, t_r^d]$ intervals and the red tick represents *now*. Figures in the middle of each block are the period numbers when transfers have last been scheduled.

of r_g to the upper discrete value is performed just before sending connectivity requests at line 11 of Algorithm 3. We used discrete steps of 100 Mbps.

The first two experiments both use a common set of default parameters parametrized by the slot duration δ:

- Slot duration: δ;
- Provisioning advance: $\alpha = \delta/2$;
- Granting advance: $t_{grant} = \delta/20$;
- Transfer request inter-arrival: $\delta/12$;
- Number of requests: 2000;
- Attributes of requests:
  - $t_r^d - t_r^s = d = \delta/2$;
  - $t_r^s - t_r^a = 2.\delta$;
  - $r_r^{min} = R = 53\,\text{Mbps}$ $(v_r = R.\delta/2)$;
  - $P_r = 2$.

Figure 8.7 shows an example of schedule and capacity planning for a random set of transfer requests with 100 Mbps discrete steps of available bandwidth. We can observe that, as the rounded r_g is reused as a lower bound from one period to the next, BPLP can fill this provisioned bandwidth with *Scheduled* requests.

8.2.1 Impact of r_r^{min}

In this experiment, we vary the value of r_r^{min}. The total submitted volume is not constant. We can observe on Figure 8.8, that, in the presence of rounding, the extra bandwidth decreases

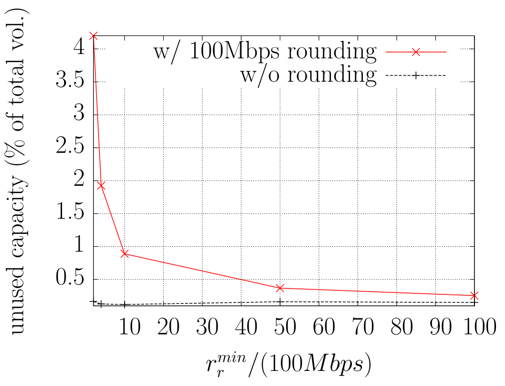


Figure 8.8: Provisioned but unused bandwidth as a function of r_r^{min} without rounding to discrete bandwidth value and with a rounding to the upper 100 Mbps; x-axis is in multiples of 100 Mbps.

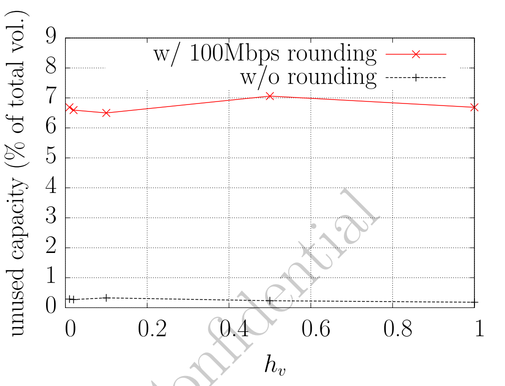


Figure 8.9: Provisioned but unused bandwidth as a function of transfer volume homogeneity parameter h_v without rounding to discrete bandwidth value and with rounding to the upper 100 Mbps.

relatively as the total volume increases, with the maximum error being when r_r^{min} is close to the bandwidth step value. This looks reasonable as steps represent a larger amount of the provisioned bandwidth in this case. The no-rounding case is not affected by this variation as provisioning is just scaled with r_r^{min}.

8.2.2 Impact of Volume Homogeneity

In this experiment we vary homogeneity h_v of the requested volumes (duration d remaining constant) and submit alternatively one request with volume $v_r = h_v.R.d$ and one with $v_r = (2 - h_v).R.d$, leaving the total volume constant for any h_v.

It can be observed on Figure 8.9 that the homogeneity of volume has no impact on the provisioning as it remains constant for the different values of h_v. This is because requests overlap during an important time interval, and large requests encompass small ones. We can also notice that without rounding, the extra bandwidth reserved is much less than 1 % of the total scheduled volume while it is about 6 % with rounding to the upper 100 Mbps multiple.

These preliminary experiments have shown that increasing volume heterogeneity has no impact. This can be explained by an important overlapping of reservation windows as large

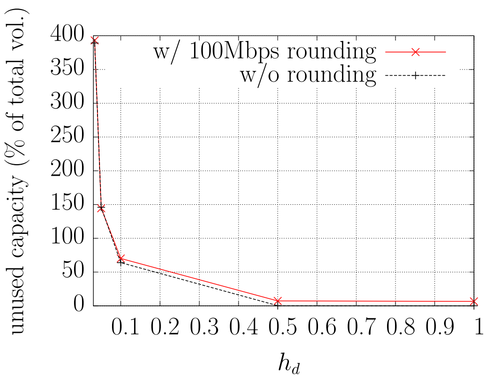


Figure 8.10: Provisioned but unused bandwidth as a function of transfer duration homogeneity parameter h_d without and with rounding to the upper 100 Mbps

requests encompass small ones, and by our assumption of unbounded provisionable capacity.

8.2.3 Impact of Reservation Window Duration Heterogeneity

In this experiment, we mix requests with the same volume but heterogeneous reservation windows. We alternatively submit one request with duration $h_d.\delta/2$ and one with $(2-h_d).\delta/2$.

Figure 8.10 shows that duration heterogeneity has a very important influence on the unused capacity. This is because decreasing the duration of some requests without varying their volumes, increases their r_r^{min}. With as small a h_d as 1/30, r_r^{min} is 30 times higher than in default case. Furthermore, due to no-rejection policy, reserved bandwidth has to be greater than r_r^{min} even when constraining requests have small volumes compared to the volume they force the SP to provision. We can observe that performance of discrete and continuous bandwidth cases are similar with the 5 % difference previously observed.

8.2.4 Impact of Patience

Given that most of the time the duration and the volume of request will be fixed by the application semantic, we now observe the impact of the patience factor, which represents the request's malleability. In our model, patience is higher than, or equal to, 1 otherwise requests would not be accepted. If patience equals 1, requests can not be reshaped and have to be scheduled as one single rectangle at r_r^{max} during $[t_r^s, t_r^d]$. The impact of this factor can be observed on Figure 8.11 where extra reserved bandwidth decreases quickly as P_r increases. When P_r is greater than 1.5, performance is constant. This means that the patience factor has a strong impact.

8.2.5 Impact of Proportion of Malleable Requests

In this set of experiments, we used two classes of requests: one with patience equal to 1 (rigid requests) and one with varying patience (malleable requests).

Requests used here have the following parameters: There are 30 clients each acting as an ON-OFF source with exponential OFF times having a mean duration of 2 hours. ON times are given by the reservation windows $[t_r^s, t_r^e]$. Each client submits 8 requests. The volumes of each request follow a Pareto distribution with a minimum volume equal to 100 GB and a shape parameter equals to 1.5, leading to an average volume of 300 GB per request. Each source has a maximum sending rate of 100 Mbps or 1 Gbps randomly chosen with equal probability.

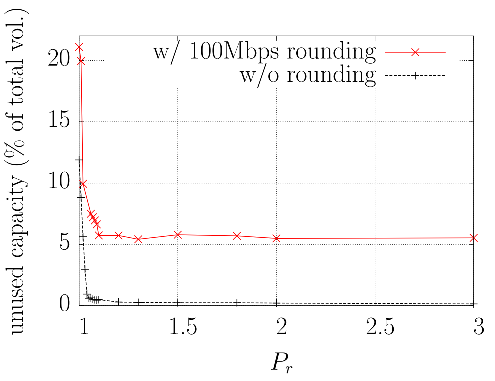


Figure 8.11: Provisioned but unused bandwidth as a function of patience without and with rounding to the upper 100 Mbps

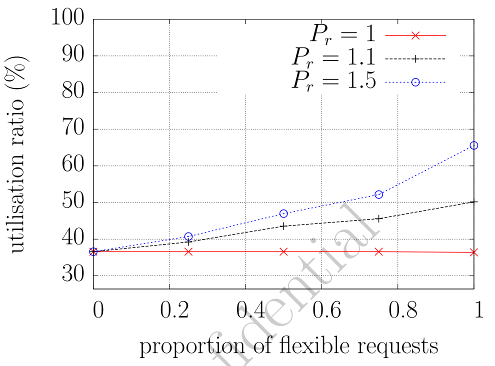


Figure 8.12: Provisioned but unused bandwidth as a function of p for different patiences ($P_r \in \{1, 1.1, 1.5\}$) with rounding to the upper 100 Mbps

Similarly, each transfer can be flexible and have a varying patience ($P_r \in \{1, 1.1, 1.5\}$) or can have no flexibility ($P_r = 1$). The probability of being the former is p.

As an example, with a sending rate of 100 Mbps, a 300 GB transfer has a minimum completion time (with a patience equal to 1) of 6.67 hours. This is much smaller than the chosen slot duration, which is equal to 24 hours. In this experiment, as the maximum rate is fixed, increasing patience decreases the minimum rate r_r^{min} and thus increases the transfer's potential duration.

We can observe on Figure 8.12 that provisioning of this considered set of requests lead to an important waste of bandwidth when there is no flexibility ($P_r = 1$ or $p = 0$). But the gain is linear with proportion of malleable requests as r_r^{min} of this kind of requests is P_r times smaller when the requests are malleable and the number of such requests is in proportion p of the total number of requests.

The second observation is that increasing the patience is an efficient way to decrease the over-provisioning. Increasing the patience of 10 % from $P_r = 1$ to $P_r = 1.1$, increases the utilization ratio from 36 % to 50 % when all requests are flexible. However this gain is not linear with patience as going from $P_r = 1.1$ to $P_r = 1.5$ only improves the utilization ratio by 17 %.

8.2.6 Conclusion

While the users' requests can be seen as flows of packets—*e.g.,* FTP transfers over a TCP connection—and thus have a great flexibility in terms of reachable rates, provisioned bandwidth has to be considered as circuits with guaranteed bandwidth. These circuits can be realized through MPLS circuits and thus offer fine granularity in terms of rate. In this case, a rational solution of the linear program can be approximated well by the provisioned rate. But they can also be implemented using coarse-grained technologies such as MEF ev-lines [MEF09] which offer Ethernet rates: 10, 100, 1000, 10000 Mbps with the possibility of trunking them.

In this case, the problem would require to solve a mixed integer linear program which is known to be much more complex. Instead, in the proposed solution, a linear program is solved and the rate is rounded towards the next greater available rate. But as the optimization problem is solved at every periods and requests potentially overlap several periods, the achieved solution is better than a simple rounding: at the next schedule, some requests have the opportunity to be shaped to fully utilize already reserved bandwidth and thus prevent the increase of provisioned bandwidth in other time slots.

When the malleable transfer requests come continuously and sufficiently in advance (more than one period) the solution is efficient.

Users can be incited to do so by economical or qualitative incentives: different pricing—a solution used by airliners—or different probabilities of being rejected—which FCFS admission rules would do.

It is also interesting to consider other objective functions, such as the total provision cost. The scheduling-optimization problems with different objective functions can be addressed using the same techniques as in this work, if the objective function can be formulated as a linear optimization problem.

Another point worth noting is the work done in [Rouskas and Jackson, 2003] on the design of optimal service levels—bandwidth granularity—that a network operator can propose to a client so that he can minimize the average unused capacity. This result requires knowledge of the distribution of provisioned rate to distribute the service levels.

Summary

The scheduling and provisioning service proposed in this chapter reuses the interface to specify malleable transfer requests and fixed bandwidth requests in a common format based of maximum rate and volume, as in Chapter 5. In addition, it extends the model of Chapter 6 to provision the infrastructure supporting the allocations.

The bandwidth-provisioning problem has been formulated as an optimization problem aiming to minimize the sum of allocated bandwidth. It provides a solution for *in-advance* malleable and fixed requests scheduling and the provisioning of the underlying network infrastructure in case this infrastructure has infinite resources (or supposed so). The objective of the proposed optimization problem is to minimize the total volume of bandwidth reserved which makes it a reasonable objective for service providers that will have to pay for this resource. Performance evaluation demonstrates how time and rate granularity as well as request malleability affect resource provisioning and utilization.

For the artificial workload we considered, we showed that, due to the accept-all policy, some transfer request patterns lead to highly inefficient provisioning and a large amount of wasted bandwidth. This exhibits a limit of the proposed strategy which should require users to specify requests within a given range of parameters. This would enable SP to constrain the patience to be high enough—with $P_r \geq 1.5$ for example—and thus avoid too strict requests. Alternatively these requests with disproportionate r_r^{min} could also be charged differently based on their flexibility.

In the present chapter, immediate requests are not considered. These transfer requests can be served in the extra bandwidth allocated by BPLP or by an extra amount of bandwidth estimated by a traffic predictor focusing on immediate transfer requests. In the context of self-sizing networks adaptively dimensioned as traffic changes, several dynamic bandwidth-allocation strategies have been proposed [Sahinoglu and Tekinay, 2001, Liang and Han, 2007, Krithikaivasan et al., 2007]. These approaches are all based on predictors. Another perspective of this work is to use different kinds of workload for the evaluation: unfortunately, no real traces are available for now.

Q8 How can dynamic provisioning of network infrastructure be used to provide a bandwidth-scheduling service?

A8 *In a tiered architecture—with users, service providers, and network operators—, users can request malleable services to service providers, which will schedule them and issue rigid requests to network operators to provision their virtual infrastructures, used to deliver the services. Obviously, users can also directly request bandwidth from network operators, but since the duration of a slot and the proposed rate might be too large for a single users, it might lead to a poor utilization ratio.*

— Nine —
Routing Games over Time to Study In-Advance Transfer Planning

Introduction

Routing games is a game-theoretical subject that sheds light on an important practical problem in the Internet (or any transportation network for that matter): the benefits and implications of routing traffic without any central authority. ‘Routing’ here refers to source routing, where an individual user is able to decide his route, or path, from the source to the destination.

The literature on routing games is wealthy. Since the introduction of the concept of equilibrium in transportation networks by J. G. Wardrop in 1952, this kind of Nash equilibrium has been widely studied and extended. Among these, we note the introduction of the price of anarchy and the results on classes of cost functions by Roughgarden *et al.* [Roughgarden, 2002]. The dynamics of convergence to equilibrium have been investigated in [Fischer and Vöcking, 2004]. The concepts of non-atomic and atomic routing games are detailed in [Nisan et al., 2007, Ch. 18]. Since flow-over-time or dynamic flows are used in several works on bandwidth and flow allocation, including ours, it is interesting and important

v_r	demand (volume)
(s_r, d_r)	source/destination pair
t_r^s	start time (assumed to be a rational)
t_r^d	deadline (assumed to be a rational)
P_r	set of unique labels for $s_r - d_r$ paths of $G = (V, E)$

Table 9.1: Table of notations for Routing Games over Time

to extend non-atomic routing games to routing games that take time into consideration. This chapter presents this extension.

While previous works on routing games consider path selection as the only variable decision for a user, we explore, along with path selection, another important variable—time preference. We call this game, *routing game over time* (RGoT). In this game, users face a cost on the selected path that depends on the time as well as on the total amount of traffic sent on this path.

The motivation for including the time criterion for decision making comes from the need to meet time constraints during data transfers. In this chapter, we propose a model which enables quantifying the inefficiency—price of anarchy—of realizing such a system as a result of selfish decisions, with no centralized coordination. The model for RGoT is presented in Section 9.1. Therein, we define the concept of equilibrium and optimal allocation. We show that under convex costs functions, allocation can be described using step functions. In Section 9.2 we prove that RGoT can be solved by solving an instance of a classical routing game, and map some of the important results known for this kind of game. Finally, in Section 9.3, using a simple example, we illustrate and motivate the need for coordination.

9.1 Routing Games over Time

In this section, we define the model for RGoT, and then extend the concepts of *equilibrium flows* and *optimal allocations* to this model.

9.1.1 Model

This section describes the RGoT model used in this work. It is inspired from the *non-atomic routing game* model and from the model of flow over time. This model of routing game assumes infinite number of *users*. Each user is small, and as such does not contribute significantly to the *cost*. *Request* as presented in the following definition refers to a kind of request; meaning that the users that form a particular request have same constraints in terms of source, destination, and paths; and total aggregate volume is known. We now proceed to give some important definitions. The notations used are listed in Table 9.1 for a given request r.

Definition 9.1.1 (Request). In the directed graph G, $r \triangleq (s_r, d_r, v_r, t_r^s, t_r^d, P_r)$, is a request between vertices s_r and d_r, dates t_r^s, t_r^d, allowed to transfer over one or more paths from P_r and with a total volume of v_r. Every path in P_r is unique, in the sense that if two requests use the same path (as set of edges) their labels will be different. In addition, we let $\mathcal{P} \triangleq \bigcup_i P_r$.

These requests induce a traffic on the network which is characterized by the rate traffic going on each path. This is called *allocation vector*.

Definition 9.1.2 (Allocation vector). If f_p is the rate for request r over the path $p \in P_r$, the allocation vector f is defined as being the vector $[f_p]_{p\in\mathcal{P}}$ with $f_p : \underset{t\mapsto f_p(t)}{\mathbb{R}_+ \to \mathbb{R}_+}$. In addition, we define *link allocation* as: $f_e : \underset{t\mapsto f_e(t)}{\mathbb{R}_+ \to \mathbb{R}_+}$ where $f_e(t) \triangleq \sum_{p\in\mathcal{P}:e\in p} f_p(t)$

This allocation vector gives the rates over time with which each request will send on its paths to achieve transfer of the specified volume during the time window. If it does so, it is said to be a feasible allocation. Link allocation gives the total rate seen on a link at a given date.

Definition 9.1.3 (Feasible allocation).

$$\forall r \in R, \int_{t_r^s}^{t_r^d} \sum_{p\in P_r} f_p(t)\mathrm{d}t = v_r$$

Next, we define per-edge cost. It is assumed to be a piecewise constant function with regard to the time.

Definition 9.1.4 (Per-edge cost function). For an edge e, $c(e, b, t)$ is the cost function for transferring at rate b at time t over link e. For the cost function, we furthermore assume: given I_c, a finite partition of time ($\mathbb{R}_+$) with rational breaks, c has the following structure: $c(e, b, t) \triangleq \sum_{J\in I_c} \mathbb{1}_{t\in J} c_{J,e}(b)$ and $c_{J,e}(.)$ is assumed to be non-negative, continuous and non-decreasing.

Using the definitions of allocation vector and cost, allocation f incurs cost $c(e, f_e(t), t)$ at time t on edge e. From the assumption on the form of cost, cost on this interval can be split on intervals I_c as done below. But before doing this, we define RGoT.

Definition 9.1.5 (RGoT). (G, R, c), where G is a network, R a set of requests and c a cost function, defines an RGoT.

In the remainder, we focus on the interval:

$$\mathbb{T} \triangleq \left[\min_{r\in R}\{t_r^s\}, \max_{r\in R}\{t_r^d\}\right]$$

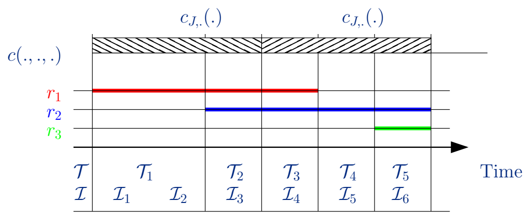
All time breaks—t_r^s, t_r^d, and changes of cost function—are rationals. Hence, we can divide time axis in intervals of same lengths, such that all these breaks come at the boundaries by taking this length as the least common multiplier of denominators of interval lengths. This partitioning of time is used in Def. 9.1.7 to define the set of time intervals. It is illustrated in Figure 9.1(a). This is an artifact, used later, to reduce this game to normal non-atomic routing games.

Definition 9.1.6 (Time intervals). For a game (G, R, c) with I_c being the set of time interval used to define cost function c, we define $\mathcal{T}$ as the set of time intervals defined by all the start time t_r^s and deadline t_r^d of request $r \in R$ and start and end of intervals I_c included in $\mathbb{T}$. $\mathcal{T}$ is a partition of $\mathbb{T}$.

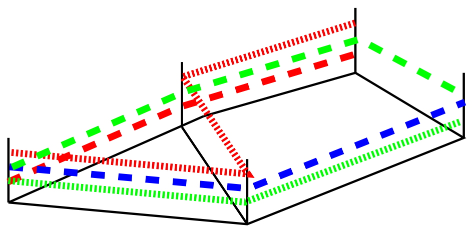
Definition 9.1.7 (Same length time intervals). We define $\mathcal{I}$ as the refined set of intervals of $\mathcal{T}$ such that all intervals of $\mathcal{I}$ have the same length (called $|J|$ in the remaining). This set is obtained by subdividing $\mathcal{I}$.

Definition 9.1.8 (Intervals of a request).

$$\mathcal{I}_r \triangleq \{J \in \mathcal{I} | J \subset [t_r^s, t_r^d]\}$$


$c_{J,.}(.)$
$c_{J,.}(.)$
$c(.,.,.)$
r_1
r_2
r_3
$\mathcal{T}$ $\mathcal{T}_1$ $\mathcal{T}_2$ $\mathcal{T}_3$ $\mathcal{T}_4$ $\mathcal{T}_5$
Time
$\mathcal{I}$ $\mathcal{I}_1$ $\mathcal{I}_2$ $\mathcal{I}_3$ $\mathcal{I}_4$ $\mathcal{I}_5$ $\mathcal{I}_6$

(a) Timeline and its discrete versions.



(b) Example of requests with their alternate paths.

Figure 9.1: Example of requests with their intervals

In the remainder of this work, $c_{J,e}$ has been extended to J in $\mathcal{I}$ by using the function $c_{I,e}$ of the interval $I \in I_c$ that contains J.

Definition 9.1.9 (Per-path cost on an interval). For any time interval J,

$$c_{J,p}(f) \triangleq \int_{t\in J} \sum_{e\in p} c(e, f_e(t), t)\mathrm{d}t$$

Definition 9.1.10 (Per-path cost).

$$\begin{aligned}
c_p(f) &\triangleq c_{\mathbb{T},p}(f) \\
&= \int_{t\in\mathbb{T}} \sum_{e\in p} c(e, f_e(t), t)\mathrm{d}t \text{ (Def. 9.1.9)} \\
&= \int_{t\in\mathbb{T}} \sum_{e\in p} \sum_{J\in I_c} \mathbf{1}_{t\in J} c_{J,e}(f_e(t))\mathrm{d}t \text{ (def. of c())} \\
&= \sum_{J\in\mathcal{T}} \int_{t\in J} \sum_{e\in p} c_{J,e}(f_e(t))\mathrm{d}t \\
&= \sum_{J\in\mathcal{T}} \sum_{e\in p} \int_{t\in J} c_{J,e}(f_e(t))\mathrm{d}t \\
c_p(f) &= \sum_{J\in\mathcal{I}} \sum_{e\in p} \int_{t\in J} c_{J,e}(f_e(t))\mathrm{d}t
\end{aligned} \tag{9.1}$$

Having defined the model, next sections define some specific feasible flows which arise from selfish user behaviors, equilibrium flows, and from social interest, optimal allocations.

9.1.2 Equilibrium Flows

As said before, users behave selfishly on their infinitely small fraction of volume belonging to one request. This leads to allocations which form a subset of feasible allocations, called equilibrium flows. We show that they can be defined using step functions with steps on $\mathcal{I}$ when cost is convex. The definition of equilibrium that follows, is similar to (characterization of) Wardrop equilibrium.

Definition 9.1.11 (Equilibrium). f is an *equilibrium flow* in (G, R, c) if:

(i) f is a feasible allocation vector, and;

(ii) for every $r \in R$, for every interval J and $\tilde{J}$ included in $[t_r^s, t_r^d]$ such that $|J| = |\tilde{J}|$ and any p, $\tilde{p} \in P_r$ where $\int_{t \in J} f_p \mathrm{d}t > 0$, $c_{J,p}(f) \leq c_{\tilde{J},\tilde{p}}(f)$.

If f is an equilibrium flow, for every $r \in R$ and any $p \in P_r$, $c_p(f) \leq c_p(\tilde{f})$ with $\tilde{f}$ such that, f_p has been replaced by $\tilde{f}_p$, and $\int_{t_r^s}^{t_r^d} f_p(t)\mathrm{d}t = \int_{t_r^s}^{t_r^d} \tilde{f}_p(t)\mathrm{d}t$, as it is a feasible flow and following second point of Def. 9.1.11 with $I = J = \mathcal{T}$ (f_p is supposed to be null out of $[t_r^s, t_r^d]$ as it does not contribute to feasibility).

We proceed to show that step functions are suitable for equilibrium flows, when the cost is convex.

Proposition 9.1.12. *If cost function $c_{J,e}(b)$ is convex, for any link allocation f_e, there is one step function constant on each $J \in \mathcal{I}$ that transfers the same volume on each time interval for a cost at least as good.*

Proof. We proceed in two steps.

(i) $f_e(t)$ is not better than constant function on J:
Let J be an interval without arrivals/departures/cost-breaks, we suppose $f_e(t)$ is integrable, $\int_J f_e(t)\mathrm{d}t = f_{J,e}|J|$ and $c_{J,e}(b)$ be a convex function. Let $\mu(.)$ be the Lebesgue measure, $\mu(J) = |J|$ and $\eta(.) \triangleq \mu(.)/\mu(J)$. Obviously $\eta(J) = 1$ and:

$$\int_J c_{J,e}(f_e(t))\mu(\mathrm{d}t) = \mu(J)\int_J c_{J,e}(f_e(t\mu(J)))\eta(\mathrm{d}t)$$

As $\eta(J) = 1$, using Jensen's inequality:

$$c_{J,e}(\int_J f_e(t\mu(J))\eta(\mathrm{d}t)) \leq \int_J c_{J,e}(f_e(t\mu(J)))\eta(\mathrm{d}t)$$

(from definition of $\eta(.)$ and above equation)

$$\Rightarrow c_{J,e}(\int_J \frac{f_e(t)}{\mu(J)}\mu(\mathrm{d}t)) \leq \frac{1}{\mu(J)}\int_J c_{J,e}(f_e(t))\mu(\mathrm{d}t)$$

(from definition of $f_{J,e}$)

$$\Rightarrow \mu(J)c_{J,e}(\frac{f_{J,e}|J|}{\mu(J)}) \leq \int_J c_{J,e}(f_e(t))\mu(\mathrm{d}t)$$

$$\Rightarrow |J|c_{J,e}(f_{J,e}) \leq \int_J c_{J,e}(f_e(t))\mu(\mathrm{d}t) \tag{9.2}$$

(9.2) says that the cost of a function constant on J and equal to $f_{J,e}$ has a cost as good as any other function which transfer the same volume on J.

(ii) For any link allocation $f_e(.)$, there is one step function as "good":
First part of this proof can be applied on each intervals and $f_e(.)$ be replaced by a step function with a cost at least as good.

□

Proposition 9.1.13. *If cost function $c_{J,e}(b)$ is convex, for any feasible allocation f there is $\tilde{f}$ made of step functions which: (i) for all $p \in \mathcal{P}$ has a cost $c_p(\tilde{f})$ not worse than $c_p(f)$; (ii) has steps on $\mathcal{I}$.*

Proof. We start with one f. For each $p \in \mathcal{P}$, we define $\tilde{f}_p$ as the step function which transfers the same volume as f_p on each intervals of $\mathcal{I}$ but using a constant rate. This allocation is also feasible as it transfers the same volumes during the same intervals and thus volume constraints are satisfied. Furthermore, its steps are on $\mathcal{I}$. $\tilde{f}_e$ are step functions, as sum of step functions $\tilde{f}_p$. Using Eq. (9.1) and Prop. 9.1.12, $c_p(\tilde{f}) \leq c_p(f)$. □

Next we show that for a convex cost function, for any equilibrium flow, there is a corresponding equilibrium flow made of step function which is as good in terms of cost. If we come back to infinitely small users making requests, this basically means, they have no incentive of using anything other than a step function constant on intervals of $\mathcal{T}$ or $\mathcal{I}$.

Proposition 9.1.14. *Equilibrium flows can be taken as step functions.*

Proof. Equilibrium flows are feasible allocation and can thus be taken as step function for a cost not higher, as proved by Prop. 9.1.13. □

It can be seen that with cost functions that are not non-negative, convex or non-decreasing in f_e, the link allocation, this proposition does not apply. This is because, we can not exploit the structure of time intervals to define the step functions since it might be cheaper to reduce the duration of transfer while increasing the utilized rate, possibly without limits. As an example, this kind of allocation (equilibrium flow and later optimal allocation) as step functions of time do not apply for sub-linear cost as it is always better to group the utilization of the resources.

9.1.3 Social Cost and Optimal Allocation

Having defined equilibrium allocation that will result from selfish decisions of users realizing requests of R, we now define optimal allocation that minimizes the total cost charged to serve the requests. The total cost is referred as social cost. By dividing it by $\int_t \sum_p f_p$, average cost can be obtained.

Definition 9.1.15 (Social cost).

$$\begin{aligned} C(f) &\triangleq \int_t \sum_e f_e(t) c(e, f_e(t), t) \mathrm{d}t \\ &= \int_t \sum_e \sum_{J \in I_c} \mathbf{1}_{t \in J} f_e(t) c_{J,e}(f_e(t)) \mathrm{d}t \text{ (def. of c())} \\ &= \sum_{J \in I_c} \int_{t \in J} \sum_e f_e(t) c_{J,e}(f_e(t)) \mathrm{d}t \\ C(f) &= \sum_{J \in \mathcal{I}} \sum_e \int_{t \in J} f_e(t) c_{J,e}(f_e(t)) \mathrm{d}t \end{aligned} \tag{9.3}$$

Optimal allocation—*in the social sense*—is:

Definition 9.1.16 (Optimal allocation). A feasible flow of (G, R, c) is optimal if it minimizes $C(f)$ over other feasible flows.

As in equilibrium flow, if the social cost is convex, we can take optimal allocation as step function as stated in Prop. 9.1.20. To begin with, we observe that convexity of cost function implies convexity of social cost.

Remark 9.1.17. $x \mapsto x.c_{J,e}(x)$ is convex on $\mathbb{R}_+$. Since this is a product of two convex, non-decreasing and positive functions on $\mathbb{R}_+$.

Proposition 9.1.18. *If cost function $c_{J,e}(b)$ is convex, positive and non-decreasing, for any link allocation f_e, there is one link allocation as step function, constant on each $J \in \mathcal{I}$, that transfers the same volume on each time interval for a social cost at least as good.*

Proof. As stated in Remark 9.1.17, $x \mapsto x.c_{J,e}(x)$ is convex on $\mathbb{R}_+$. Allocation are positive functions. Then, using Jensen's inequality (or Hermite-Hadamard inequality) we show that there is a step function with steps in $\mathcal{I}$ which is as good as any feasible allocation and transfer the same volume on each time interval. □

Proposition 9.1.19. *If cost function $c_{J,e}(b)$ is convex, positive and non-decreasing, for any feasible allocation f, there is one $\tilde{f}$ made of step functions which: (i) has a costs $C(\tilde{f})$ not worse than $C(f)$; and (ii) has steps on $\mathcal{I}$.*

Proof. We start with one f. For each $p \in \mathcal{P}$, we define $\tilde{f}_p$ as the step function which transfers the same volume as f_p on each interval of $\mathcal{I}$, but using a constant rate. This allocation is also feasible as it transfers the same volumes during the same intervals and thus volume constraints are satisfied, steps are on $\mathcal{I}$. $\tilde{f}_e$ are step functions as sum of step functions $\tilde{f}_p$. Using Eq. (9.3) and Prop. 9.1.18, $C(\tilde{f}) \leq C(f)$. □

We conclude this section showing that there is an optimal allocation among the set of feasible allocations made of step functions.

Proposition 9.1.20. *Optimal allocation can be taken as step function with steps on $\mathcal{I}$.*

Proof. Optimal allocations are feasible allocation and can thus be taken as step function with step on $\mathcal{I}$ for a cost not higher as proved by Prop. 9.1.19. □

In the next section, we exploit the structure of step function and revisit previously introduced definitions to establish equivalence between step allocations and generic allocations under convex costs.

9.1.4 Discrete Allocations—Reduction to Step Function

As users and regulator try to minimize their costs (equilibrium flow) or social cost (optimal allocation), all the feasible allocations of interest are step functions and their steps are on $\mathcal{I}$ (Prop. 9.1.13 and 9.1.19).

Hence, interesting feasible allocations are step functions with steps on $\mathcal{I}$, they can be described by discrete value giving their value on the steps.

***Definition 9.1.21** ($f_{J,p}$).* For a feasible allocation f, we define the $|\mathcal{I}|\times|\mathcal{P}|$ matrix $[f_{J,p}]_{(J,p)\in\mathcal{I}\times\mathcal{P}}$, where $f_{J,p} \triangleq \displaystyle\int_{t\in J} \frac{f_p(t)}{|J|}dt$ is the constant rate of a step function which transfers as much as f on interval J.

Remark 9.1.22. It can be noted that in Def. 9.1.21, $f_{J,p}$ for J not used by a request r using path p is supposed to be null. More formally:

$$\forall r \in R, \forall p \in P_r, \forall J \in \mathcal{I}\backslash\mathcal{I}_r, f_{J,p} = 0$$

This can be justified, as it would not contribute to the total volume, but will add to the cost; hence rendering useless from a social cost and individual cost point of view.

Remark 9.1.23. As f is supposed to be a step function on $\mathcal{I}$, it can be reconstructed from the matrix $[f_{J,p}]$ by:

$$\forall p \in \mathcal{P}, f_p(t) = \sum_{J \in \mathcal{I}} \mathbb{1}_J(t) f_{J,p} \tag{9.4}$$

Similarly we define discrete link allocation and show that continuous-time link allocation can be reconstructed from the discrete version.

***Definition 9.1.24** ($f_{J,e}$).*

$$f_{J,e} \triangleq \sum_{p \in \mathcal{P}: e \in p} f_{J,p} \tag{9.5}$$

Proposition 9.1.25.

$$f_e(t) = \sum_{J \in \mathcal{I}} \mathbb{1}_J(t) f_{J,e} \tag{9.6}$$

Proof.

$$\begin{aligned} f_e(t) &= \sum_{p \in \mathcal{P}: e \in p} f_p(t) \text{ (from Def. 9.1.2)} \\ &= \sum_{p \in \mathcal{P}: e \in p} \sum_{J \in \mathcal{I}} \mathbb{1}_J(t) f_{J,p} \text{ (from Eq. (9.4))} \\ f_e(t) &= \sum_{J \in \mathcal{I}} \mathbb{1}_J(t) f_{J,e} \text{ (from Def. 9.1.24)} \end{aligned}$$

□

From above definition and definition of feasible flow, we get:

Proposition 9.1.26. *f is feasible iff:*

$$\forall r \in R, \sum_{J \in \mathcal{I}_r} \sum_{p \in P_r} |J| f_{J,p} = v_r \tag{9.7}$$

Proof. From Def. 9.1.3, f is feasible iff:

$$\begin{aligned} & \forall r, \int_{t_r^s}^{t_r^d} \sum_{p \in P_r} f_p(t) \mathrm{d}t = v_r \\ \iff & \forall r, \int_{t_r^s}^{t_r^d} \sum_{p \in P_r} \sum_{J \in \mathcal{I}} \mathbb{1}_J(t) f_{J,p} \mathrm{d}t = v_r \text{ (from Remark 9.1.23)} \\ \iff & \forall r, \sum_{p \in P_r} \sum_{J \in \mathcal{I}} \int_{t_r^s}^{t_r^d} \mathbb{1}_J(t) f_{J,p} \mathrm{d}t = v_r \\ \iff & \forall r, \sum_{p \in P_r} \sum_{J \in \mathcal{I}_r} |J| f_{J,p} = v_r \text{ (from Remark 9.1.22)} \end{aligned}$$

□

RGoT	RGoT (step functions)	Time-expanded RG
$G/P_r/p/R$ (Def. 9.1.1, 9.1.5)		$\bar{G}/\bar{P}_r/\bar{p}/\bar{R}$ (Def. 9.2.1, 9.2.2, 9.2.4)
f/f_p (Def. 9.1.2)	$[f_{J,p}]$ (Def. 9.1.21)	$\bar{f}/[f_{\bar{p}}]$ (Def. 9.2.5)
f_e (Def. 9.1.2)	$f_{J,e}$ (Def. 9.1.24/Prop. 9.1.25)	$f_{\bar{e}}$ (Def. 9.2.6/Prop. 9.2.8)
c_e (Def. 9.1.4)	$c_{J,e}$ (Def. 9.1.4)	$c_{\bar{e}}$ (Def. 9.2.12)
c_p (Def. 9.1.10)	$c_{J,p}$ (Def. 9.1.9/Prop. 9.1.27, 9.1.28)	$c_{\bar{p}}$ (Def. 9.2.13/Prop. 9.2.14)
C (Def. 9.1.15)	Prop. 9.1.29	$\bar{C}$ (Def. 9.2.16/Prop. 9.2.17)
feasible (Def. 9.1.3)	Prop. 9.1.26	Def. 9.2.10/Prop. 9.2.11
equilibrium (Def. 9.1.11)	Prop. 9.1.30	Def. 9.2.18/Prop. 9.2.19
optimal (Def. 9.1.16)	Prop. 9.1.31	Def. 9.2.20/Prop. 9.2.21

Table 9.2: Map of the results

9.1.5 Reduction of Cost Function, Equilibrium and Optimal

In this section, we prove that, under convex cost function, both continuous and discrete formulations are equivalent.

Table 9.2 presents the big picture of the different definitions, propositions and their counter parts in the different forms of the game. The first column corresponds to the definitions of RGoT we introduced before. Second column is the discrete version presented in previous sections. In this section, we prove that, under convex cost function, both formulations are equivalent. Finally, last column is the reduction of the RGoT to non-atomic routing game with time-expanded network.

Proposition 9.1.27. *If cost function $c_{J,e}(b)$ is convex, for any f (supposed to be a vector of step functions constant on each J), for all $p \in \mathcal{P}$, and for any J in $\mathcal{I}$,*

$$c_{J,p}(f) = \sum_{e \in p} |J| c_{J,e}(f_{J,e}) \tag{9.8}$$

Proof. This comes from Def. (9.1.9) applied to step function f and interval J of $\mathcal{I}$ and $[f_{J,e}]$ defined using equations (9.6) and (9.5). □

Proposition 9.1.28. *If cost function $c_{J,e}(b)$ is convex, for any f (supposed to be a vector of step functions constant on each J), for all $p \in \mathcal{P}$,*

$$c_p(f) = \sum_{J \in \mathcal{I}} \sum_{e \in p} |J| c_{J,e}(f_{J,e}) \tag{9.9}$$

Proof. This is obtained by applying Eq. (9.1) to the step function f and $[f_{J,e}]$ defined using $[f_{J,p}]$ as given by Eq. (9.6) and (9.5). □

Proposition 9.1.29. *If cost function $c_{J,e}(b)$ is convex, for any f (supposed to be a vector of step functions constant on each J),*

$$C(f) = \sum_{J \in \mathcal{I}} \sum_{e} |J| f_{J,e} c_{J,e}(f_{J,e}) \tag{9.10}$$

Proof. Same as previous proof but using Eq. (9.3). □

Proposition 9.1.30. *f is an equilibrium flow in (G, R, c) iff:*

(i) f is a feasible allocation vector, and;

(ii) for every $r \in R$, for every $J, \tilde{J} \in \mathcal{I}_r$, $c_{J,p}(f) \leq c_{\tilde{J},\tilde{p}}(f)$ with p, $\tilde{p}$ in P_r where $\sum_J f_{J,p} > 0$.

Proof. We prove the equivalence in two parts:

- $\Rightarrow$ f is supposed to be made of step functions with steps on $\mathcal{I}$. First point is straightforward. By replacing (9.7) in second point of Def. 9.1.11 and applying it on J and $\tilde{J}$ in $\mathcal{I}_r$ we obtain above mentioned conditions.
- $\Leftarrow$ First point is again straightforward. Regarding second point, condition 2) in Def. 9.1.11 is different is the sense that it is for any pair of sub-interval of $[t_r^s, t_r^d]$. For any such sub-interval and by linearity of the integration over time, cost can be expressed as a sum of cost $c_{J,p}$ with J in $\mathcal{I}_r$ or part of a such interval. From this, we get two partitions of J and $\tilde{J}$. They can be different but we can obtain a one to one mapping of sub-intervals of same length from J to $\tilde{J}$. On each of these pairs of sub-intervals, costs are independent of time and f_P is constant. Applying condition 2) of Prop. 9.1.30 on the underlying intervals in $\mathcal{I}_r$ and scaling the results with respect to the length of the sub-interval, once summed, proves condition 2) of Def. 9.1.11.

□

We summarize this section by the following proposition:

Proposition 9.1.31. *f is an optimal allocation iff $[f_{J,p}]$ is feasible and $C(f)$ as given in (9.10) is minimized.*

Proof. Trivial. □

9.1.6 Conclusion

With Proposition 9.1.31, we have shown the equivalence of the continuous and discrete versions of the routing game under convex cost functions. Next, we move on to the time-expanded network.

9.2 Reduction to Non-atomic Routing Game

Since we have defined the RGoT and exhibited the step function structure under convex costs, we can now reduce the game to non-atomic routing game, and then exploit the results already known on this class of games.

9.2.1 Time-expanded Routing Game

To use the results from the classical non-atomic routing game, which we refer as *Routing Game in Time-Expanded Network*, we now present that the two games can be mapped, and we show how to do so.

Definition 9.2.1 (Time-expanded (TE) network). For a RGoT (G, R, c), $\bar{G}$ is the time-expanded graph obtained from G by duplicating it one for each interval of $\mathcal{I}$ and adding one pair of virtual source and virtual sink for each request r in R. These extra sources/sinks are connected to the actual source/sink of each duplicate of G in time intervals of $\mathcal{I}_r$. We note $\bar{e} = (J, e)$.

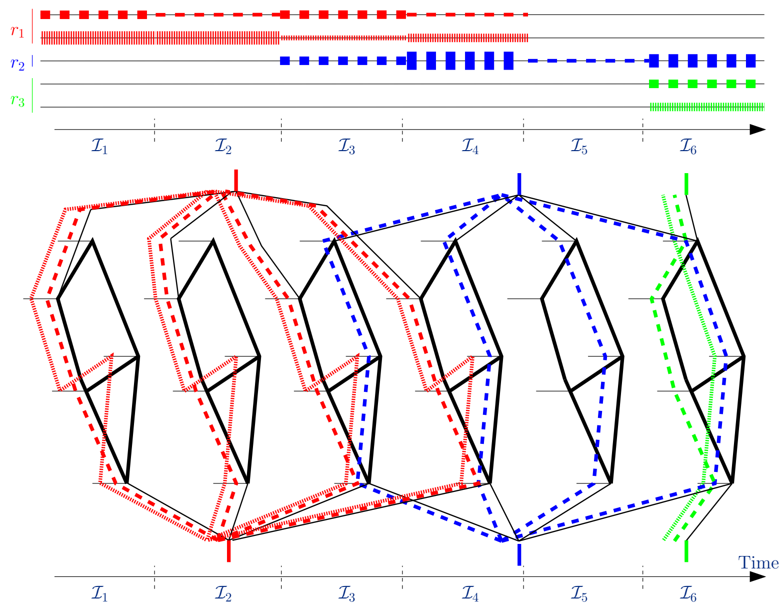


Figure 9.2: Time-expanded network

All edges of $\bar{G}$, except those connecting the actual and virtual sources/sinks, are of the form $\bar{e} = (J, e)$. $\bar{G}$ has $|V||\mathcal{I}| + 2|R|$ vertices and $|E||\mathcal{I}| + 2\sum_{r\in R} |\mathcal{I}_r|$ edges.

Definition 9.2.2 (Path in TE network). For every request r in R, we define $\bar{P}_r$ as the set of paths in $\bar{G}$: for every path p in P_r and every interval J in $\mathcal{I}_r$, $\bar{P}_r$ contains a path $\bar{p} = (J, p)$ which contains the same edges in $\bar{G}$ as p in G in addition to the two extra edges that connect the virtual source and virtual sink of r to the nodes in $\bar{G}$, which represents, for each interval their actual source/destination in G. We let, $\bar{\mathcal{P}} = \bigcup_{r\in R} \bar{P}_r$

Observe that $\bar{P}_r$ has $|P_r||\mathcal{I}_r|$ paths.

To illustrate, the time-expanded graph of requests presented in Figure 9.1 is as in Figure 9.2. The network of Figure 9.1(b) can be seen replicated in Figure 9.2 for the intervals of a request. If two requests (*e.g.*, r_1 and r_2) have overlapping interval ($\mathcal{I}_3$), the graph is shared by paths of the same requests ($\bar{P}_1$ and $\bar{P}_2$). Virtual source/sink nodes connect paths of one request.

Proposition 9.2.3. $\bar{e} = (J_e, e) \in \bar{p} = (J_p, p)$ *iff*

$$\begin{cases} e \in p \\ J_e = J_p \end{cases}$$

Proof. By construction of $\bar{G}$ in Def. 9.2.1. □

The edges of $\bar{G}$ that do not have their counterparts in G are only used by one request and have a null cost in the time-expanded routing games. Hence, they will not contribute to the social cost.

We now define the requests that are used in the new graph: $\bar{R}$, the allocations and feasible allocations in these games.

Definition 9.2.4 (Requests in the TE network). For every request r in R, we define: $\bar{r} \triangleq (\bar{s}_r, \bar{d}_r, \bar{P}_r, \bar{v}_r \triangleq \frac{v_r}{|J|})$ and the set $\bar{R}$ as the set of requests $\bar{r}$.

Definition 9.2.5 (Allocation in the TE network).

$$\bar{f} \triangleq [f_{\bar{p}}] \triangleq [f_{J,p}]$$

Definition 9.2.6 (Link allocation).

$$f_{\bar{e}} \triangleq \sum_{\bar{p}\in\bar{\mathcal{P}}:\bar{e}\in\bar{p}} f_{\bar{p}}$$

Remark 9.2.7. For each edge of $\bar{G}$ which does not have a $\bar{e} = (J, e)$ form, there is no need to define $f_{\bar{e}}$. In the remainder, we use the notation $\bar{e} = (J, e)$ to refer to edges that contribute to the costs and define the counterpart in G and time intervals.

Proposition 9.2.8. *For any $\bar{e} = (J_e, e)$, $f_{\bar{e}} = f_{J_e,e}$.*

Proof. By Def. 9.2.6, $f_{\bar{e}} = \sum_{\bar{p}\in\bar{\mathcal{P}}:\bar{e}\in\bar{p}} f_{\bar{p}}$ and by Def. 9.2.5, $f_{\bar{p}} = f_{J,p}$. Using Prop. 9.2.3 and definition of $\bar{p}$ and $\bar{e}$, we get: for any $\bar{e} = (J_e, e)$,

$$\begin{aligned} f_{\bar{e}} &= \sum_{(J,p)\in\mathcal{I}\times\mathcal{P}:e\in p\wedge J_e=J} f_{J,p} \\ &= \sum_{p\in\mathcal{P}:e\in p} f_{J_e,p} \\ f_{\bar{e}} &= f_{J_e,e} \end{aligned}$$

where $f_{J_e,e}$ is the one defined in Def. 9.1.24. □

Remark 9.2.9. Using Prop. 9.2.8 and Prop. 9.1.25, we get:

$$f_e(t) = \sum_{\bar{e}=(J,e)\in\mathcal{I}\times\{e\}} \mathbb{1}_J(t) f_{\bar{e}}$$

Using Remark 9.1.23 and Def. 9.2.5, we also have:

$$f_p(t) = \sum_{\bar{p}=(J,p)\in\mathcal{I}\times\{p\}} \mathbb{1}_J(t) f_{\bar{p}}$$

Flow allocation f can thus be reconstructed.

Definition 9.2.10 (Feasible allocation). $\bar{f}$ is a feasible allocation for $\bar{R}$ iff:

$$\forall r \in \bar{R}, \sum_{\bar{p}\in\bar{P}_r} f_{\bar{p}} = \bar{v}_r$$

This is the definition of feasibility in normal routing game.

Proposition 9.2.11. *f is feasible iff $\bar{f}$ is feasible.*

Proof.

$$\begin{aligned}
& f \text{ is feasible} \\
\iff & \forall r \in R, \sum_{J \in \mathcal{I}_r} \sum_{p \in P_r} |J| f_{J,p} = v_r \text{ (from Prop. 9.1.26)} \\
\iff & \forall r \in R, \sum_{(J,p) \in \mathcal{I}_r \times P_r} f_{J,p} = \frac{v_r}{|J|} \text{ (as } |J| \text{ is uniform)} \\
\iff & \forall r \in \bar{R}, \sum_{\bar{p} \in \bar{P}_r} f_{\bar{p}} = \bar{v}_r \text{ (Def. of } \bar{G}, \bar{R} \text{ and } f_{\bar{p}}) \\
\iff & \bar{f} \text{ is feasible}
\end{aligned}$$

□

In the following two sections, we map the cost functions, equilibrium and optimal allocation from RGoT to routing games on the TE graph defined in this section.

9.2.2 Cost Function in TE Network

Here we show the mapping of cost between RGoT and time-expanded routing games.

Definition 9.2.12 ($c_{\bar{e}}$). For all $\bar{e} = (J, e)$, $c_{\bar{e}}(.) \triangleq |J| c_{J,e}(.)$. We call $\bar{c}$ the function that associates $c_{\bar{e}}(.)$ to $\bar{e}$.

Definition 9.2.13 ($c_{\bar{p}}$). For any $\bar{f}$ and for all $\bar{p}$, $c_{\bar{p}}(\bar{f}) \triangleq \sum_{\bar{e} \in \bar{p}} c_{\bar{e}}(f_{\bar{e}})$.

Proposition 9.2.14. *For all p in $\mathcal{P}$,*

$$c_p(f) = \sum_{\bar{p} \in \mathcal{I} \times \{p\}} c_{\bar{p}}(\bar{f})$$

$f_p(t)$ is still supposed to be null out of I_r for each $p \in P_r$.

Proof.

$$\begin{aligned}
c_p(f) &= \sum_{J \in \mathcal{I}} \sum_{e \in p} |J| c_{J,e}(f_{J,e}) \text{ (from Prop. 9.1.28)} \\
&= \sum_{J \in \mathcal{I}} \sum_{\bar{e} \in \bar{p}} c_{\bar{e}}(f_{\bar{e}}) \text{ (from Def. 9.2.6, 9.2.12 and Prop. 9.2.3)} \\
c_p(f) &= \sum_{\bar{p} \in \mathcal{I} \times \{p\}} c_{\bar{p}}(\bar{f}) \text{ (from Def. 9.2.13)}
\end{aligned}$$

□

Proposition 9.2.15. *For all p in $\mathcal{P}$ and J in $\mathcal{I}$, $c_{\bar{p}}(\bar{f}) = c_{J,p}(f)$ with $\bar{p} = (J, p)$.*

Proof.

$$\begin{aligned}
c_{\bar{p}}(\bar{f}) &= \sum_{\bar{e} \in \bar{p}} c_{\bar{e}}(f_{\bar{e}}) \text{ (from Def. 9.2.13)} \\
&= \sum_{e \in p} |J| c_{J,e}(f_{J,e}) \text{ (from Def. 9.2.6, 9.2.12 and Prop. 9.2.3)} \\
c_{\bar{p}}(\bar{f}) &= c_{J,p}(f) \text{ (from Prop. 9.1.27)}
\end{aligned}$$

□

Definition 9.2.16 (Social cost). $\bar{C}(\bar{f}) = \sum_{\bar{e}} f_{\bar{e}} c_{\bar{e}}(f_{\bar{e}})$

Proposition 9.2.17. $\bar{C}(\bar{f}) = C(f)$

Proof.

$$\begin{aligned}
C(f) &= \sum_{J \in \mathcal{I}} \sum_{e} |J| f_{J,e} c_{J,e}(f_{J,e}) \text{ (by Prop. 9.1.29)} \\
&= \sum_{\bar{e}} f_{\bar{e}} c_{\bar{e}}(f_{\bar{e}}) \text{ (by Prop. 9.2.8 and Def. 9.2.12)} \\
C(f) &= \bar{C}(\bar{f}) \text{ (by Def. 9.2.16)}
\end{aligned}$$

Transition between first and second line is obtained by recalling that per-edge cost over edges that are not of the form $\bar{e} = (J, e)$ is null. □

9.2.3 Equilibrium and Optimal Allocations

The two definitions given here are from non-atomic routing games. The two propositions basically state that our RGoT can be *solved* by recasting it to this well-known game and using established results, as will be shown in next section.

Definition 9.2.18. $\bar{f}$ is an *equilibrium flow* in $(\bar{G}, \bar{R}, \bar{c})$ if:

(i) $\bar{f}$ is a feasible allocation vector, and;
(ii) for every $r \in \bar{R}$, $c_{\bar{p}}(\bar{f}) \leq c_{\tilde{\bar{p}}}(\bar{f})$ with $\bar{p}$, $\tilde{\bar{p}}$ in $\bar{P}_r$ where $f_{\bar{p}} > 0$.

Proposition 9.2.19. *$\bar{f}$ is an equilibrium flow in $(\bar{G}, \bar{R}, \bar{c})$ iff f is an equilibrium flow in (G, R, c).*

Proof. First point comes from Prop. 9.2.11. Second point is obtained by replacing $c_{J,p}$ in Prop. 9.1.30 as established in Prop. 9.2.15. □

Definition 9.2.20. A feasible flow of $(\bar{G}, \bar{R}, \bar{c})$ is optimal if it minimizes $\bar{C}(\bar{f})$ over other feasible flows.

Proposition 9.2.21. *A feasible flow $\bar{f}$ of $(\bar{G}, \bar{R}, \bar{c})$ is optimal iff f is optimal in (G, R, c).*

Proof. The proof comes from the equivalence established in propositions 9.2.11 and 9.2.17. □

9.2.4 Results—Existence and PoA

Next theorem on the existence of equilibrium flows for non-atomic routing games comes from the literature [Nisan et al., 2007].

Theorem 9.2.22 ([Nisan et al., 2007, Theorem 18.8])**.** *Under the assumption of existence of feasible allocation(s), for a non-atomic instance $(\bar{G}, \bar{R}, \bar{c})$ there exists at least one equilibrium, and for any two equilibria $\bar{f}$ and $\tilde{\bar{f}}$, $\bar{c}_{\bar{e}}(\bar{f}_e) = \bar{c}_{\bar{e}}(\tilde{\bar{f}}_e)$.*

From Theorem 9.2.22, it follows that, $\bar{C}(\bar{f}) = \bar{C}(\tilde{\bar{f}})$; *i.e.,* , all equilibrium have the same social cost. We also derive from the theorem that our routing game over time have equilibria since the time-expansion of the problem always exists and this always has at least one equilibrium.

Corollary 9.2.23. *From Theorem 9.2.22 and reduction of the game made in previous sections (game, costs, equilibrium and optimal), it follows that (G, R, c) has at least one equilibrium, and they all have the same social cost.*

Existence of optimal allocation is trivial, as there is always some (possibly one) feasible allocation.

Definition 9.2.24 (PoA). Price of Anarchy (PoA) is the ratio of the cost of worst equilibrium to the cost of optimal allocation.

Since all equilibria in routing game have the same cost, PoA is the ratio of the cost of any equilibrium to the optimal social cost. Furthermore, since cost and social cost are same for RGoT and its TE counterpart, PoA of $(\bar{G}, \bar{R}, \bar{c})$ is equal to the PoA of the corresponding (G, R, c).

9.2.5 Conclusion

We proved that routing games over time, with positive, convex and non-decreasing costs, can be "solved" using time-expansion of the original games and regular normal games.

9.3 Application to Scheduling

In this section, we illustrate the mapping of PoA between the two forms of game, and its implications, for example, on the benefit of doing centralized scheduling to transfer data. We consider a simple example of a single link network with time-slot preference and congestion. The cost model used in this example accounts for the utility of users which is to transfer a given volume during a time window. As long as the transfer is supposed to finish in time, the user gets a full utility or face a null cost. When the transfer is running late, we used a linear cost to extend the cost function to this case.

First, we describe the cost functions associated with this model, and then discuss equilibrium flows, optimal allocations and PoA.

9.3.1 Lateness and Time-slot Preference as Cost

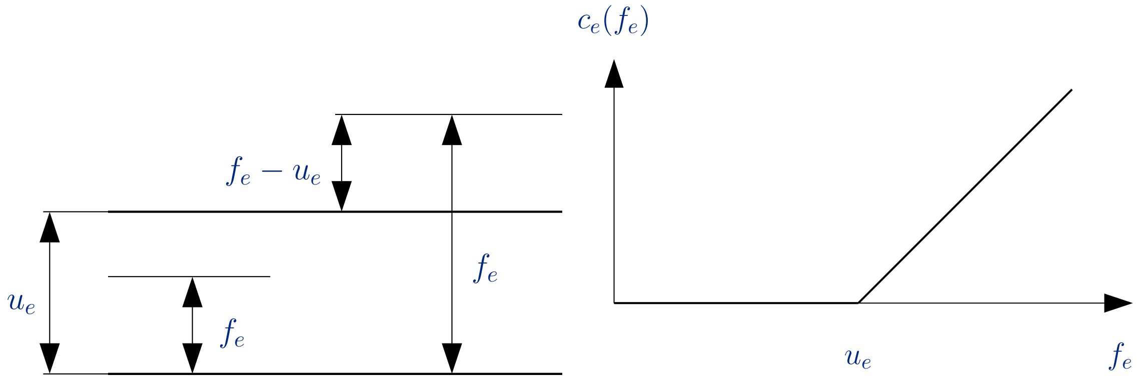
In this model, f_p (and thus f_e) is the expected rate that players plan to have. As long as the link is not oversubscribed, they will not be late with respect to their deadlines. If it is congested, lack of bandwidth increases lateness—cost—as a linear function of this oversubscription. Finally, cost due to congestion on edge e is given by:

$$c_e(f_e) = \begin{cases} 0 & \text{if } f_e \in [0, u_e] \\ \frac{f_e - u_e}{u_e . \alpha(e,R)} & \text{else} \end{cases}$$

Another component of this cost represents users' preference of one time-slot over the other. This is achieved using the following piecewise cost function:

$$c(e, f_e, t) = \begin{cases} d_1 & \text{if } t < T/2 \\ d_2 & \text{else} \end{cases}$$

We illustrate this cost on the following example described in the following assumption statement:



(a) Cost model for congestion on a link of capacity u_e.

(b) Cost of congestion $c_e(f_e)$ for edge e.

Figure 9.3: Cost model for congestion

Assumption 9.3.1. This RGoT uses one single link and one request of total volume v, but the time interval $[0, T]$, where $0 < T$, is divided in two equal parts and is assigned user preferences: d_1 and d_2 with $0 \leq d_1 < d_2$. The cost function adopted for this link of capacity u is then:

$$c(e, f_e, t) = \begin{cases} d_1 + c_e(f_e) & \text{if } t < T/2 \\ d_2 + c_e(f_e) & \text{else} \end{cases}$$

with:

$$c_e(f_e) = \begin{cases} 0 & \text{if } f_e \in [0, u] \\ \frac{f_e - u}{u_e . \alpha(e,R)} & \text{else} \end{cases}$$

Let $s = \frac{1}{u_e . \alpha(e,R)}$, $s > 0$.

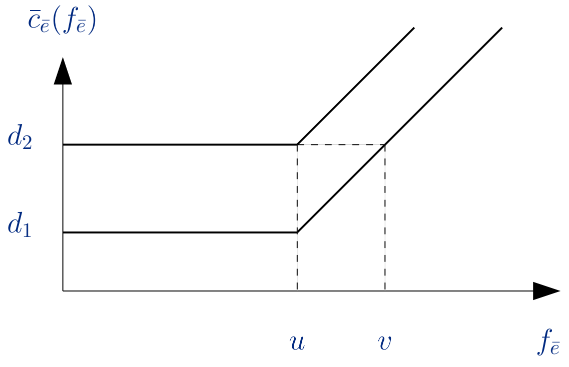
These costs are represented by Figure 9.3(a). For each time interval, the cost is convex, positive and non-decreasing in link allocation.

Using time-expansion as defined earlier, we get the routing game with two links (1 and 2) sharing same source and destination; first of these links has cost $\bar{c}_1$, and second $\bar{c}_2$ as defined presently:

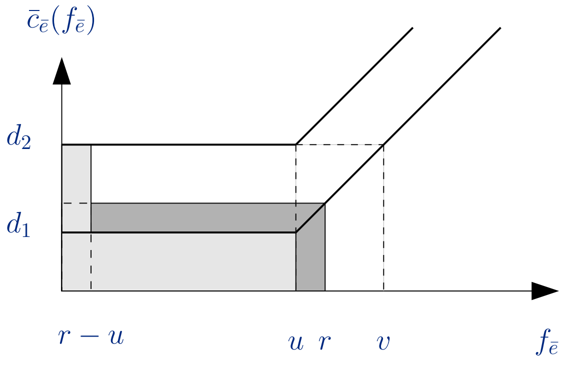
$$\bar{c}_{\bar{e}}(f_{\bar{e}}) = \begin{cases} d_{\bar{e}} & \text{if } f_{\bar{e}} \in [0, u] \\ d_{\bar{e}} + s(f_{\bar{e}} - u) & \text{else} \end{cases}$$

with $f_e(t) = \mathbb{1}_{t \in [0,T/2)} f_{\bar{1}} + \mathbb{1}_{t \in [T/2,T]} f_{\bar{2}}$ in RGoT. Figure 9.4(a) show the cost functions on the time-expanded network. It includes time preference and congestion aversion for the two links of $\bar{G}$. Note that, the slope s equals $\frac{d_2 - d_1}{v - u}$.

Let $r = v/T$. If the cheapest link is not congested, *i.e.*, , $r \leq u$, as its cost is less than that of the other link, both equilibrium flow and optimal allocation exclusively use this path/link. This results in a PoA of 1 as social cost of both are the same. But when the cheaper link starts to get congested, it might remain less expensive than the second link up to a point. Beyond this point, it is less expensive from a social cost point of view to route the flow on the two links. But the equilibrium flow still uses only one link as its cost $c_p(.)$ remains lesser than the other. Figure 9.4(b) illustrates this situation with two alternative feasible flow and their social cost as shaded. The sum of the areas of the two light gray rectangle are less than the one of the dark rectangle (which is the social cost of equilibrium). More formally:



(a) Cost functions.



(b) Two feasible flows.

Figure 9.4: Time preference and congestion aversion

Proposition 9.3.2. *For* $\max\{u, v-u\} < r$ *and* $r < \min\{v, 2u\}$, *equilibrium flow uses exclusively link 1 and PoA* > 1.

Proof. Provided r is between $\max\{u, v-u\}$ and $\min\{v, 2u\}$, equilibrium flows only use link 1 and the social cost of such an allocation is:

$$C_{eq} = (d_1 + (r-u)s)r$$

If we consider a different feasible allocation which allocates only u on first link and the remaining on second link, its social cost is:

$$C = u\ d_1 + (r-u)d_2$$

We now demonstrate that this allocation has a social cost strictly less than the equilibrium flow:

$$\begin{aligned} & C < C_{eq} \\ \iff & u\ d_1 + (r-u)\ d_2 < (d_1 + (r-u)s)r \\ \iff & (r-u)\ d_2 < (r-u)\ d_1 + (r-u)s\ r \\ \iff & \frac{d_2 - d_1}{s} < r \\ \iff & v - u < r \end{aligned}$$

which holds by assumption.

As the cost of this feasible allocation is strictly less than the one of equilibrium flow, PoA is strictly more than 1. □

9.3.2 Equilibrium and Optimal Allocation

Define $x_1(r)$ and $x_2(r)$ as the part of the traffic r sent on links 1 and 2 for equilibrium flow. Similarly, $x_1^*(r)$ and $x_2^*(r)$ are the fraction for optimal allocation. We have $x_1(r) + x_2(r) = r$ and $x_1^*(r) + x_2^*(r) = r$. $C_{eq}(r)$ denotes the social cost of equilibrium under demand r and $C^*(r)$ the social cost of optimal allocation.

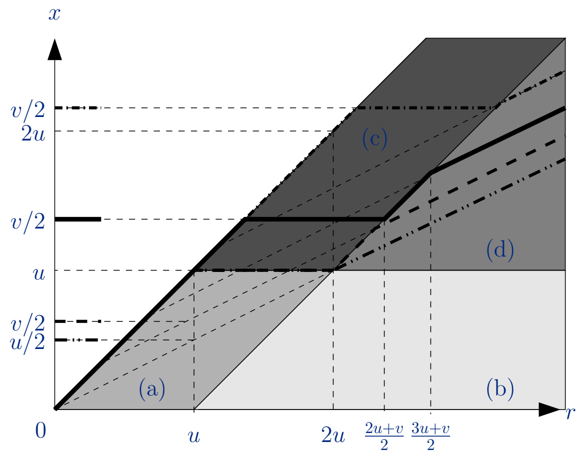


Figure 9.5: Domain of feasible allocations $x(r)$ and optimal allocations for different values of $v/2$

9.3.2.1 Optimal allocations

Since costs are defined in pieces, social cost are also piecewise. In the following, x denotes x_1 and x_2 is $r - x$.

It follows that the social cost of a feasible allocation $C(x, r) = x\ c_1(x) + (r - x)c_2(r - x)$ has the following expression in the different regions (a),(b),(c) and (d) of Figure 9.5:

(a) $\underline{x < u \text{ and } r - x < u}$: $C(x, r) = x\ d_1 + (r - x)d_2$ and $\frac{\partial C}{\partial x} = d_1 - d_2 < 0$

(b) $\underline{x < u \text{ and } r - x > u}$: $C(x, r) = x\ d_1 + (r - x)(d_2 + s(r - x - u))$ and $\frac{\partial C}{\partial x} = d_1 - d_2 - (2r + u - x)s < 0$

(c) $\underline{x > u \text{ and } r - x < u}$: $C(x, r) = x(d_1 + s(x - u)) + (r - x)d_2$ and $\frac{\partial C}{\partial x} = d_1 - d_2 + (2x - u)s$, which is strictly positive when $x > v/2$, and strictly negative when $x < v/2$.

(d) $\underline{x > u \text{ and } r - x > u}$: $C(x, r) = x(d_1 + s(x - u)) + (r - x)(d_2 + s(r - x - u))$ and $\frac{\partial C}{\partial x} = d_1 - d_2 + 2(2x - r)s$, which is strictly negative below $x = (r + (v - u)/2))/2$ and strictly positive above.

As cost functions are continuous, so are optimal and equilibrium allocations as functions of r.

We conclude from previous list that optimal allocations are:

- $\underline{r \le u}$: $x_1^*(r) = r$ and $x_2^*(r) = 0$
- $\underline{u \le r \le 2u}$: $x_1^*(r) = \max\{\min\{r, v/2\}, u\}$ and $x_2^*(r) = r - \max\{\min\{r, v/2\}, u\}$
- $\underline{2u \le r}$: $x_1^*(r) = \max\{\min\{r, (r + (v - u)/2))/2\}, u\}$ and $x_2^*(r) = r - \max\{\min\{r, (r + (v - u)/2))/2\}\}$

The optimal allocations as a function of r are shown on Figure 9.5 by the emphasized lines (plain and dashed) for different values of $v/2$.

9.3.2.2 Equilibrium flows

Regarding equilibrium flows:

- $\underline{r < v}$: Since $c_1(r) < c_2(r)$, $x_1(r) = r$ and $x_2(r) = 0$.

- $\underline{v \le r < u+v}$: $x_1(r) = v$ and $x_2(r) = r - v$ is the equilibrium flow.
- $\underline{u+v \le r}$: $x_1(r) = (v-u)/2 + r/2$ and $x_2(r) = (u-v)/2 + r/2$ is the equilibrium flow.

9.3.2.3 PoA

For the specific case where $u < v/2 < 2u$, we have:

- $\underline{r \le v/2}$: since $x_1^*(r) = r$, $PoA = 1$.
- $\underline{v/2 \le r \le u + v/2}$: in this region, $x_1^*(r) = v/2$ and $C^* = (d_1 - d_2 + (v/2-u)s)\ v/2 + r\ d_2$ while $x_1(r) = r$ and $C_{eq} = r\ (d_1 + (r-u)s)$.

$$\begin{aligned} PoA = \frac{C_{eq}}{C^*} &= \frac{r\ (d_1 + (r-u)s)}{(d_1 - d_2 + (v/2-u)s)v/2 + r\ d_2} \\ &= \frac{r\ (d_1 + (r-u)s)}{r\ d_2 - (v/2)^2 s} \end{aligned}$$

PoA is increasing as long as $s < (2r+v)d/v^2$ since r is less than v if s is less than $2(u+v)d/v^2$ maximum PoA is reached for $r = u + v/2$ and is:

$$PoA = \frac{(2u+v)(-2d + s\ v - 2s\ u)}{-4d\ u - 2d\ v + s\ v^2}$$

In the other case, PoA will first increase and then decrease before leaving the interval. Maximum PoA is reached for $r = \frac{s\ v^2/d - v}{2}$ and its value is:

$$PoA = \frac{(s\ v - d)^2}{d^2}$$

- $\underline{u + v/2 < r}$: Since $v/2 > u$, $r > 2u$ and then the system is clearly overloaded (both links congested) which is not really interesting in this scenario.

9.3.2.4 Conclusion

In this example based on assumption 9.3.1, we see that as long as none of the links are congested, the PoA remains 1. This basically means that selfish behavior leads to optimal allocation. Therefore, under the presented model, no centralized control is needed to achieve optimal allocation. This model, following the Wardrop equilibrium concept, assumes that users have complete and up-to-date information to decide which path to use based on the cost. The cost depends on the link allocation reached so far. Thus, it assumes that users know the current link allocation. This information is not needed for the time-slot preference part of the cost, but for the congestion-aversion part. Hence, implementing this scheduling mechanism without coordinator mandates the congestion information be available to users, that too, in advance. In both cases, coordination is essential: either to take the decision or to publish the information.

In the case where the system is near congestion, as studied above, the PoA is greater than 1. Here, coordination can improve the social cost by sacrificing some of the flows. This can not be achieved by selfish users unwilling to spontaneously take a more costly path.

From [Roughgarden, 2003], we know that, using affine cost functions, it is possible to construct an instance of the non-atomic routing game with a PoA up to 4/3. This value is reached in a two-link settings. When the cost functions belong to the class of quadratic functions, this bound becomes $3\sqrt{3}/(3\sqrt{3}-2)$. Actually, the more non-linear the cost functions are, the higher the worst PoA is. Using polynomial cost functions of degree p, the PoA tends to infinity as p does. In the next section, we will see how this inefficiency can be mitigated through coordination or modification of the cost function.

9.3.3 Reducing Social Cost

Since PoA can be large, it is worth using optimal allocation to minimize social cost. Minimization of social cost can be achieved by a centralized system using optimal allocation or the game can be modified so that equilibrium of the new game is an optimal allocation for the original game. This corresponds to a reduction of the PoA in the new instance.

9.3.3.1 Cost Sharing and Resource Coordination

Consider a resources manager (RM) who proposes some commodities for a cost which depends on the total demand. Users know only about their own individual decisions. The cost function is likely to be convex, positive and non-decreasing as it ultimately results from the cost of congestion. A resource coordinator (RC) aggregating the requests and demanding the optimal allocation for the aggregate can do better than selfish user behaviors. It follows that RC can improve the social cost, *i.e.,* total cost. RC can then share the benefit among all entities: RC and clients. This makes this configuration profitable for both clients and RC.

9.3.3.2 Pigouvian Tax

Another alternative is to modify the game so that cost perceived by users deter them from ultimately reaching an allocation other than the optimal of original game. For this purpose the related theorem is:

Theorem 9.3.3 (Theorem 18.27 of [Nisan et al., 2007]). *Let $\bar{f}^*$ be an optimal flow for $(\bar{G}, \bar{R}, \bar{c})$ and let $\tau_{\bar{e}} = \bar{f}^*_{\bar{e}}\ \bar{c}'_{\bar{e}}(\bar{f}^*_{\bar{e}})$ denote the marginal cost tax for edge $\bar{e}$ with respect to $\bar{f}^*$. Then $\bar{f}^*$ is an equilibrium flow for $(\bar{G}, \bar{R}, \bar{c} + \tau)$*

It basically says that, equilibrium of the modified game is optimal allocation of the original game. From the realization point of view, this solution still has the problem of sharing the information, and also in addition making the users value the modified cost.

9.3.4 Conclusion

Routing games over time provide an interesting framework to study the in-advance reservation of malleable requests. Thanks to the powerful results of routing games, and the mapping of games over time to regular normal games, the results of the former can be used to better understand and improve the behavior of the modeled system.

Summary

This chapter presented a new model for RGoT. It shows that under convex cost this can be solved using the results from non-atomic routing games.

Based on an example and under the assumptions of this model—users know the current planning of occupancy when they enter the game, and know the cost functions—we have seen that selfish user behavior increases the social cost. Just because they prefer one time interval to another, users selfishly raise the cost of the cheapest links—by creating congestion on this link—instead of sacrificing their time-slot preference to keep the social cost low. This motivates the need to coordinate. Service providers orchestrate the utilization of resources and help reducing the part of the social cost, the lateness, that is due to congestion created by users.

A possible extension of this work is to use atomic or splittable flow models instead of the non-atomic model in order to consider the perception users have of their own impact on the cost. Another is to study how to use Pigouvian taxes to reduce the PoA, as it is a classical solution in economics to mitigate the effect of externality.

Q9 What is the benefit of the service approach? Is it sustainable?

A9 *Services provide coordinated decision-making entities, that can use their global view of costs and utilization to optimize the social cost of allocations. Because there is a gap between the optimal social cost and the one reached by the selfish decisions of actors—the price of anarchy is strictly greater than 1—, the service is sustainable.*

Q10 What are the alternative approaches to reduce the inefficiency introduced by the selfishness of agents?

A10 *It is possible to change the cost that users will face to deter them from being selfish as proposed in Theorem 9.3.3, and potentially raise the cost for all the participants, e.g., by creating congestion that everyone will have to pay. This is known has Pigouvian taxes. Another solution not presented here is to increase the capacity to reduce the congestion.*

— Ten —
Conclusions

Internet Model and Packet Networks The Internet has been designed 40 years ago. At that time, network resources were rare and the robustness of communications of utmost importance. A paradigm, completely different from that of existing telephone networks, has been used: packet networks. The Internet is based on this paradigm, and its protocol suite, TCP/IP, has been strongly influenced by the layering and end-to-end principles. Different attempts to control the Quality of Service of the Internet have been conducted. They failed to be widely deployed.

Hence, the performance achieved by flows on the Internet depends on the implemented transport protocols and their mechanisms. The physical network is not reliable by itself and does not decide or enforce an allocation. Reliability is obtained, following the end-to-end principle, by mechanisms lying at endpoint systems. So are the sharing mechanisms: TCP and its congestion control algorithm decide, in a distributed manner, the allocation that each flow gets. The decision is based on losses taken as congestion events.

In this layered network, losses are due to congestion occurring at lower layers—L2 or L3 of the OSI model. At these layers, the network is made of queues and servers deciding where the packets have to be sent next. In case of contention, when there are too many packets for a destination, the decision made in L2 switches and L3 routers leads to dropping packets. This has an impact on the performance that TCP can achieve.

From Optical Circuit Paradigm to Bandwidth On-Demand While the historical support for telecommunications is wire, optical technologies have enabled the deployment of high-speed communication channels. Different paradigms exist to transport data in optical networks: optical circuit switching, optical burst switching, and optical packet switching. The fundamental difference between the utilization models they support is the moment when the blocking risk faced. In circuit-switched approaches, it is at the setup of the connection, while for the burst and packet approaches it can occur at any intermediate router and for any piece of the information flow. But the setup of a circuit requires information to be exchanged between the intermediate routers and this has to be done in advance.

The processes and information required to operate a network designed to carry information have been organized in so-called planes: the Data plane, the Control plane, and the Management Plane. Control planes providing routing and signalization mechanisms have now been automatized and unified to control equipments from different technologies and vendors.

The challenge is now to deliver flexibility, for now dedicated to the network operator itself, to its customers. This bandwidth-delivery problem has been explored using different models. Depending on the economical environment that this system will meet—research & education networks, private networks, the Internet—, one is better suited than the other. The key is then to find a mechanism that gives satisfaction to all the agents—users, with their demanding applications, and operators—of this environment in order to have the system deployed and efficiently used.

Exploration of Current Bandwidth Sharing Model To have a better insight into the limits of current bandwidth-sharing approaches in the context of the future usage of the Internet, we have explored how current high-speed transport protocols behave, regarding transfer time predictability of gigabits of data among endpoints, in a range of conditions. In a fully controlled long-distance 10 Gbps network testbed, the behaviors of several TCP variants have been compared in the presence of diverse congestion level and reverse traffic situations. Identified factors have a very strong impact on transfer time predictability for several transport protocols. One factor being low aggregation level, we explored in more depth the interaction of L2 and L4 in such conditions.

This has been done is two phases. First, the behavior of a range of Ethernet switches when two long lived connections compete for the same output port was investigated. Then, we explored the impact of these behaviors on TCP in long fat networks (LFNs). Several conditions in which arbitration mechanisms introduce heavily unfair bandwidth sharing and loss burst which impact TCP performance have been shown. We have shown that different switches lead to different behaviors of the transport protocol and that the sharing pattern highly varies depending on the setting and time.

Chapter 4 concludes that with the current sharing and control models, the transfer time of large files is not predictable enough for applications.

An Alternative Bandwidth-Sharing Approach Based on Services To overcome this limitation, we propose a different sharing approach based on explicit requests from users for their specific needs. The value of a network is concerned by two externalities. One is positive: the network effect which makes its value increases more than linearly with the number of users. The second is negative: congestion, since the bandwidth used by one user cannot be by another and the capacity is finite. Sharing mechanisms have to maintain or improve this value. The sharing approaches as done in the Network Utility Maximization problem try to maximize the sum of the utilities of users under the capacity constraints where the utility functions considered are logarithm functions of the allocated bandwidth. Different approaches have been proposed to changes this. At the same time, traffic engineering approaches, used by operators in order to optimize their networks, try to minimize the cost of serving some given demands but are not made directly available to users.

In Chapter 5, we have presented our approach that combines the benefits of circuits—offering throughput, loss, and order guarantees, as well as moving the risk of blocking at establishment instead of facing it for each packet—to the benefits of the fluidity of flow allocations provided by packets networks. This approach maintains the value of the network as best-effort traffic is still possible and congestion is guaranteed not to occur for traffic for which bandwidth has been reserved. In addition, this mechanism also tries to satisfy both users and network operators, by satisfying the users' needs and giving in-advance knowledge of the future traffic together with flexibility regarding the exact allocation to the network operators.

Data Transfer Scheduling While file transfer is a bandwidth consuming application of the Internet, and transfer completion time is difficult to predict for users as shown in Chapter 4, we have proposed a bandwidth-reservation service that is able to schedule large file transfers. The transfers are scheduled in advance and are specified by a malleable request. The requests specify the upper and lower bounds of the allocated bandwidth over the time, together with the volume that the user wants to transfer with this request.

If the request is accepted, a profile of bandwidth varying with time is allocated for this transfer only. Time and volume constraints are given in the request and scheduling gives a profile that allows completing the transfer within the time constraints of the user.

This service exploits the fluidity offered by packet networks. Since the request model also includes upper and lower bounds on the allocated rate, at any instant, non-malleable requests of bandwidth, possibly specified as a time-varying profile, can also be scheduled.

We have presented the global sharing model, which accepts both scheduled traffic and best-effort traffic for the benefit of the network's value. Then, the network model used for the formulation of the problem and the feasibility of allocations have been studied. Finally, flow allocation algorithms have been presented and we have shown how the scheduling can be done in a polynomial time.

Articulation of Service, Control and Data Planes Once a bandwidth allocation profile has been decided, it has to be enforced on the network elements so that the user can actually use the resources. This implies to propagate the configuration information throughout the planes of the network, from the service plane to the data plane.

On behalf of the user or his application, an agent on the sender's machine must interact with the scheduling service in order to get the information required to use the allocated resources—the bandwidth allocation profile that describes the amount of bandwidth allocated as a function of time. This profile can then be used to configure the network stack on the sender's side to generate a profile-conforming traffic.

Chapter 7 presents the time and rate control mechanisms that are involved, the performance achieved by the transport protocol under this scheme. We have shown that a modified TCP can be used to implement the service presented in the previous chapter that assumes an ideal transport protocol able to adapt to the changes of rate specified by the bandwidth-allocation profile.

Extending BDTS Approach to Dynamic Bandwidth Provisioning As mentioned in the previous chapters, the anticipated sporadic and high-volume demands generated by time-constrained applications has led to envision services using dynamically reconfigurable optical networks to support this. However, the time and rate granularity of a bandwidth-reservation service and those of the transfer tasks using the reserved capacity, are not necessarily of the same order of magnitude, *e.g.,* one day and 1 Gbps for the bandwidth-reservation service, 100 Mbps during 2 hours for a single 90 GB transfer. This may lead to poor resource utilization and over-provisioning. Chapter 8 explored how request aggregation is able to tackle this problem.

Considering a tiered architecture, we have explored the interactions between a bandwidth-reservation service, a data-mover service providing guaranteed time-constrained data transfer and its clients. We have formulated the underlying optimization problem and proposed a strategy for bandwidth provisioning for in-advance malleable transfer requests. Simulations have shown that the temporal parameters of requests (deadline and patience) are the dominant criteria, and that a small malleability can improve performance: adding 10 % of patience to rigid requests improves the utilization ratio by 39 %.

Routing Games over Time to Study In-Advance Transfer Planning In Chapter 9, using game theory, we have studied how a service coordinating transfers can improve the social cost. This is done by quantifying the price of anarchy in the underlying game. To do so we have proposed an extension of routing games to time.

The routing game presents an interesting framework to analyze the practical problem of source routing in the Internet. It is particularly useful in quantifying the inefficiency of selfish user behavior that results in networks without any central authority. This game assumes that the only user criteria for decision making is path cost.

In this work, we have taken a step further, and model a routing game where user decision is based not only on path, but also on time. We have shown that, under convex cost functions, this new routing game over time can be mapped to the classical routing game, thereby presenting a model that can exploit well-established results in the subject.

Using a simple cost function, we have demonstrated the usefulness of the model, and motivated the need for resource coordination to minimize inefficiency or cost.

Summary Throughout these chapters, we have tried to provide answers to the concerns raised at the beginning of this dissertation. One of the elements of this thesis that I tried to emphasize is that the Internet must be considered as an economic system. Agents that are in this environment have different objectives, different information or knowledge, but they interact with each others. This system is growing because agents find their own selfish interest for participating in this economy. Some produce bandwidth, content, or services, while others consume them, or use it to do real business, such as selling real products.

New kinds of services and goods emerge and new business models are experienced. The evolution of music distribution is an example of that: from the classical CD to the DRM-free mp3 sold by Radiohead on their website at a price decided by the buyer, in a few years, completely different business models have been tried. Predicting which one will prevail in a specific area or for a specific kind of services is a difficult task. But computer science and networking researchers are looking towards the “Future Internet” and the understanding of this large and complex environment grows regularly. One thing is almost sure: telecommunication networks have become the support and the driving force of this evolving world and will remains for years.

This work had a look at different levels of the problem of bandwidth delivery to support new types of usage. Both practical and theoretical parts have been addressed. For the former: the study of how to enforce the allocation. For the latter: how to optimize the allocations, or the study of the benefit of doing coordinated allocations through game theory. An extension of an important class of games has been proposed to take time into account in the study of the in-advance malleable reservations, and assess its benefit in terms of social cost.

Future Works Some parts of the problem have been left over. As an example, we have shown that under a cost model which includes timeslot preference and congestion aversion, the scheduling service improves the social cost of the allocation. It could be interesting to study how this benefit can be shared among the agents and study the appropriate incentives to ensure that users will use the service and not bypass it. Implementation theory, mechanisms design, and theory of incentives have interesting concepts to study and quantify the characteristics of the different mechanisms that can be used to realize a given objective.

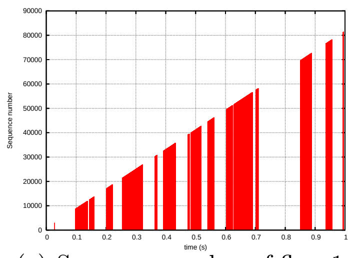
Performance-evaluation models and dimensioning tools for BDTS and provisioning problems could also be future works.

Another aspect that has been left over, is the allocation of immediate requests. Since they will come too late to trigger the reconfiguration and provisioning of the infrastructure, this demand has to be predicted.

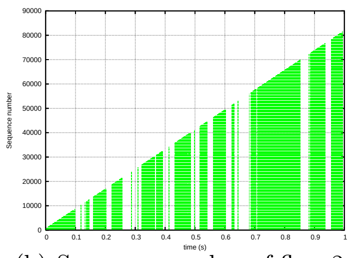
For now, the next challenge will be to make all the knowledge and experience accumulated while studying this, a groundwork for our new company, *Tiss-LinK*, that aims at providing new solutions for resources orchestration, for both users and providers.

We do believe that mechanisms that will emerge and survive the evolution of the Internet are beneficial for all the agents.

A
Figures of Chapter 4

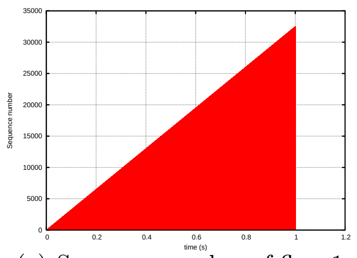


(a) Sequence number of flow 1

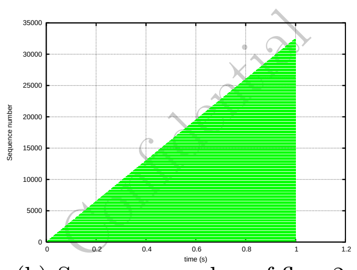


(b) Sequence number of flow 2

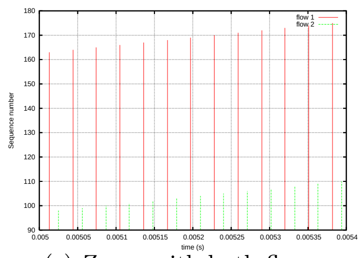
Figure A.1: Evolution of sequence number of packets on the output port (1000 Mbps + 1000 Mbps (1500 bytes) on Foundry switch)



(a) Sequence number of flow 1

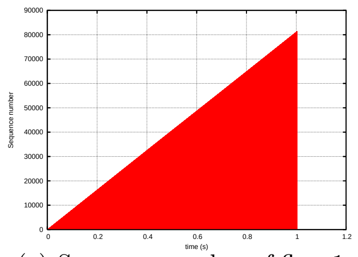


(b) Sequence number of flow 2

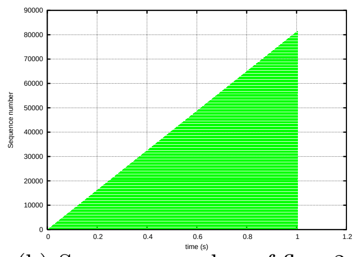


(c) Zoom with both flows

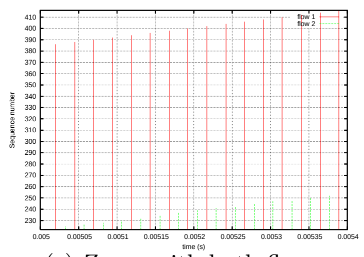
Figure A.2: Evolution of sequence number of packets on the output port (400 Mbps + 400 Mbps (1500 bytes) on Foundry switch)



(a) Sequence number of flow 1

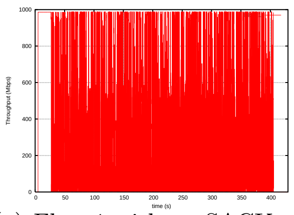


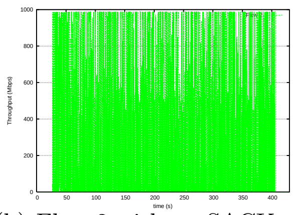
(b) Sequence number of flow 2

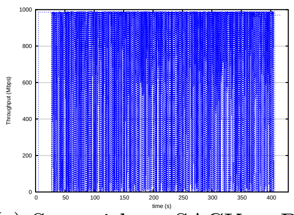


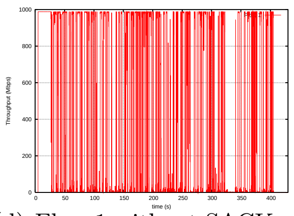
(c) Zoom with both flows

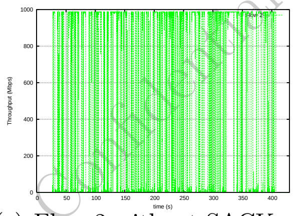
Figure A.3: Evolution of sequence number of packets on the output port (1000 Mbps + 1000 Mbps (1500 bytes) on D-Link switch)

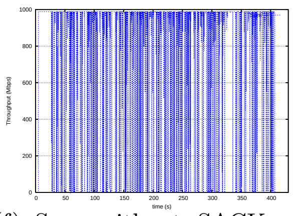

(a) Flow 1 without SACK on D-Link switch

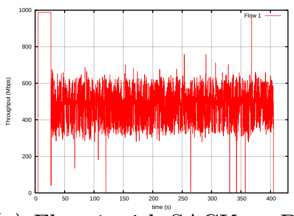

(b) Flow 2 without SACK on D-Link switch


(c) Sum without SACK on D-Link switch

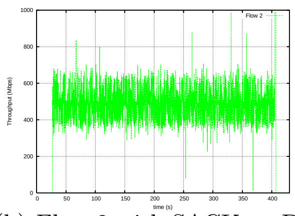

(d) Flow 1 without SACK on Foundry switch


(e) Flow 2 without SACK on Foundry switch

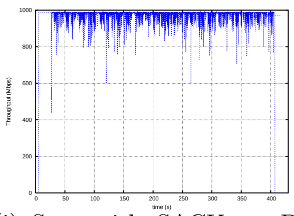

(f) Sum without SACK on Foundry switch

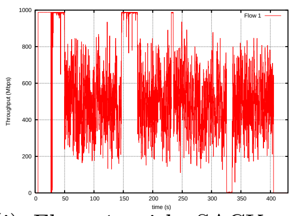


(g) Flow 1 with SACK on D-Link switch

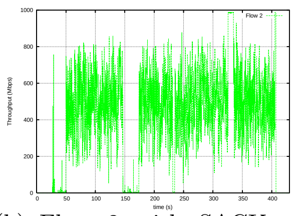


(h) Flow 2 with SACK on D-Link switch

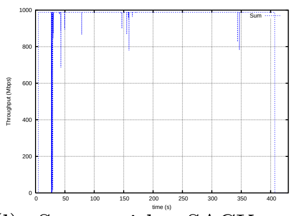

(i) Sum with SACK on D-Link switch



(j) Flow 1 with SACK on Foundry switch



(k) Flow 2 with SACK on Foundry switch



(l) Sum with SACK on Foundry switch

Figure A.4: Two BIC flows (0 ms RTT)

— B —
Publications

International Journals

[1] Sebastien Soudan, Bin Bin Chen, and Pascale Vicat-Blanc Primet. Flow scheduling and endpoint rate control in gridnetworks. *Future Generation Computer Systems*, 25(8):904–911, 2009.

[2] Pascale Vicat-Blanc Primet, Sebastien Soudan, and Dominique Verchere. Virtualizing and scheduling optical network infrastructure for emerging it services [invited]. *J. Opt. Commun. Netw.*, 1(2):A121–A132, 2009.

International Conferences with Proceedings

[3] Andrei Agapi, Sebastien Soudan, Marcelo Pasin, Pascale Vicat-Blanc Primet, and Thilo Kielmann. Optimizing deadline-driven bulk data transfers in overlay networks. In *ICCCN 2009 Track on Pervasive Computing and Grid Networking (PCGN)*, San Francisco, USA, Aug 2009.

[4] Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, Fumihiro Okazaki, Ryousei Takano, Sebastien Soudan, and Pascale Vicat-Blanc Primet. A study of large flow interactions in high-speed shared networks with Grid'5000 and GtrcNET-10 instruments. In *PFLDnet 2007*, Feb. 2007.

[5] Romaric Guillier, Ludovic Hablot, Sebastien Soudan, and Pascale Vicat-Blanc Primet. Evaluating transport protocols in very high speed grid networks: How to benefit from huge available bandwidth? poster, SuperComputing 2006, October 2006.

[6] Romaric Guillier, Sebastien Soudan, and Pascale Vicat-Blanc Primet. TCP variants and transfer time predictability in very high speed networks. In *Infocom 2007 High Speed Networks Workshop*, May 2007.

[7] Sebastien Soudan, Dinil Mon Divakaran, Eitan Altman, and Pascale Vicat-Blanc Primet. Equilibrium in size-based scheduling systems. In *16th International Conference on Analytical and Stochastic Modelling Techniques and Applications*, pages 234–248, Madrid, Spain, Jun 2009.

[8] Sebastien Soudan, Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, Fumihiro Okazaki, Ryousei Takano, and Pascale Vicat-Blanc Primet. Investigation of ethernet switches behavior in presence of contending flows at very high-speed. In *PFLDnet 2007*, Feb. 2007.

[9] Sebastien Soudan, Romaric Guillier, and Pascale Primet. End-host based mechanisms for implementing flow scheduling in gridnetworks. In *GridNets '07: Proceedings of the first international conference on Networks for grid applications*, pages 1–8, ICST, Brussels, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[10] Sebastien Soudan and Pascale Vicat-Blanc Primet. Passerelle à base de processeurs réseaux pour le contrôle des flux de grille. poster, CFIP 2006, October 2006.

[11] Sebastien Soudan and Pascale Vicat-Blanc Primet. Mixing malleable and rigid bandwidth requests for optimizing network provisioning. In *21st International Teletraffic Congress*, Paris, France, Sep. 2009.

[12] Dominique Verchere, Olivier Audouin, Bela Berde, Agostino Chiosi, Richard Douville, Helia Pouyllau, Pascale Vicat-Blanc Primet, Marcelo Pasin, Sebastien Soudan, Thierry Marcot, Veronique Piperaud, Remi Theillaud, Dohy Hong, Dominique Barth, Christian Cadéré, V. Reinhart, and Joanna Tomasik. Automatic network services aligned with grid application requirements in carriocas project. In *GridNets '08*, ICST, Brussels, Belgium, Oct. 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Research Reports

[13] Andrei Agapi, Thilo Kielmann, Marcelo Pasin, Sebastien Soudan, and Pascale Vicat-Blanc Primet. Globally optimal bulk data transfers in overlay routing networks. Technical Report TR-0157, Institute on Grid Systems, Tools and Environments, CoreGRID - Network of Excellence, August 2008.

[14] Olivier Audouin, Bela Berde, Agostino Chiosi, Richard Douville, Hélia Pouyllau, Dominique Verchère, Pierre Bozonet, Marcelo Pasin, Pascale Vicat-Blanc Primet, Sebastien Soudan, Lynda Zitoune, Véronique Vèque, Elisangela Vieira, Jean-Pierre Gallois, Iksoon Hwang, Ana Cavalli, Lionel Tuhal, Mohamad Chaitou, Jean-Franç̧ois Peltier, Jean-Louis Leroux, Thierry Marcot, Véronique Piperaud, Rémi Theillaud, Dominique Barth, Christian Cadéré, Vincent Reinhart, and Joanna Tomasik. Spécification des protocoles et algorithmes embarqués. Technical report, Alcatel-Lucent, July 2007. ”Livrable #D2.4 : CARRIOCAS - Pôle IdF System@tic”.

[15] Dinil Mon Divakaran, Sebastien Soudan, Pascale Vicat-Blanc Primet, and Eitan Altman. A survey on core switch designs and algorithms. Research Report 6942, INRIA, 2009.

[16] Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, Fumihiro Okazaki, Pascale Primet, Sebastien Soudan, and Ryousei Takano. A study of large flow interactions in high-speed shared networks with Grid5000 and GtrcNET-10 instruments. Research Report RR-6034, INRIA, 2006.

[17] Romaric Guillier, Ludovic Hablot, Pascale Vicat-Blanc Primet, and Sebastien Soudan. Evaluation des liens 10 GbE de Grid5000. Research Report 6047, INRIA, 12 2006.

[18] Romaric Guillier, Sebastien Soudan, and Pascale Vicat-Blanc Primet. UDT and TCP without congestion control for profile pursuit. Research Report 6874, INRIA, 03 2009. Also available as LIP Research Report RR2009-10.

[19] Ludovic Hablot, Olivier Glück, Jean-Christophe Mignot, Romaric Guillier, Sebastien Soudan, and Pascale Vicat-Blanc Primet. Interaction between mpi and tcp in grids. Research Report 6945, INRIA, 06 2009.

[20] S. Soudan and D.M. Divakaran. Equilibrium in size-based scheduling systems. In R. Holzer, editor, *Modeling and Control of Complex and Self-Organizing systems*, number MIP-0916, pages 22–27. University of Passau, 2009.

[21] Sebastien Soudan. Partage de bande passante et plan de contrôle optique dans les grilles. Master’s thesis, ED MathIF, ENS-Lyon, June 2006. DEA2006-01.

[22] Sebastien Soudan, Christian Cadéré, Dominique Barth, and Pascale Primet Vicat Blanc. Dynamic Bandwidth Provisioning and Malleable Bulk Data Transfer Scheduling. Research Report, INRIA, 2008.

[23] Sebastien Soudan, Dinil Mon Divakaran, Eitan Altman, and Pascale Vicat-Blanc Primet. Equilibrium in size-based scheduling systems. Research Report 6888, INRIA, 04 2009. Also available as LIP Research Report RR2009-11.

[24] Sebastien Soudan, Dinil Mon Divakaran, Eitan Altman, and Pascale Vicat-Blanc Primet. Extending routing games to flows over time. Research Report 6931, INRIA, 05 2009. Also available as LIP Research Report RR2009-17.

[25] Sebastien Soudan, Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, Fumihiro Okazaki, Pascale Primet, and Ryousei Takano. Investigation of Ethernet switches behavior in presence of contending flows at very high-speed. Research Report RR-6031, INRIA, 2006.

[26] Sebastien Soudan, Romaric Guillier, and Pascale Primet. End-host based mechanisms for implementing Flow Scheduling in GridNetworks. Research Report RR-6205, INRIA, 2007.

[27] Sebastien Soudan and Pascale Vicat-Blanc Primet. Grid flow control: Prototype of a grid gateway based on Network Processors. Research Report RR-5909, INRIA, 2006. Also available as LIP Research Report RR2006-18.

[28] Pascale Vicat-Blanc Primet, Marcelo Pasin, Olivier Audouin, Agostino Chiosi, Jean-Michel Houssin, Bela Berde, Dominique Verchère, Sebastien Soudan, Dominique Barth, Christian Cadéré, Loubna Echabbi, Joanna Tomasik, Vincent Reinard, Véronique Vèque, and Lynda Zitoune. Scénarios et besoins réseau pour les applications distribuées. Technical report, INRIA, July 2007. ”Livrable #D2.1 : CARRIOCAS - Pôle IdF System@tic”.

References

Online References

[FLO09] FLOC: FLow Controller website, May 2009. http://www.ens-lyon.fr/LIP/RESO/Software.html.

[G-L09] G-lambda project website, May 2009. http://www.g-lambda.net.

[jBD09] jBDTS: Bulk Data Transfer Scheduling service website, May 2009. http://www.ens-lyon.fr/LIP/RESO/Software.html.

[MEF09] MEF: Metro Ethernet Forum website, May 2009. http://metroethernetforum.org.

[NSI09] NSI-WG: OGF Network Service Interface WG (NSI-WG) website, May 2009. http://www.ggf.org/gf/group_info/view.php?group=nsi-wg.

[NTT09] 14 tbps over a single optical fiber: Successful demonstration of world's largest capacity, press release, July 2009. http://www.ntt.co.jp/news/news06e/0609/060929a.html.

[RE09] IEEE 802.3 Residential Ethernet Study Group, May 2009. http://www.ieee802.org/3/re_study/index.html.

References

[TMR, 2007] (2007). Metrics for the evaluation of congestion control mechanisms. IETF Internet Draft: http://www.ietf.org/internet-drafts/draft-irtf-tmrg-metrics-07.txt.

[Acquaah et al., 2008] Acquaah, P., Liu, J.-M., and Chan, H. A. (2008). Emission and discard priority scheme for optical burst switched networks. *J. Opt. Netw.*, 7(12):977–988.

[Andersen et al., 2001] Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R. (2001). Resilient overlay networks. *SIGOPS Oper. Syst. Rev.*, 35(5):131–145.

[Anderson et al., 1992] Anderson, T. E., Owicki, S. S., Saxe, J. B., and Thacker, C. P. (1992). High speed switch scheduling for local area networks. In *Proceedings of the fifth international conference on Architectural support for programming languages and operating systems, ASPLOS-V*, pages 98–110.

[Anderson et al., 1993] Anderson, T. E., Owicki, S. S., Saxe, J. B., and Thacker, C. P. (1993). High-speed switch scheduling for local-area networks. *ACM Trans. Comput. Syst.*, 11(4):319–352.

[Arrow, 1950] Arrow, K. J. (1950). A difficulty in the concept of social welfare. *Journal of Political Economy*, 58:328.

[Ash, 2006] Ash, G. R. (2006). *Traffic Engineering and QoS Optimization of Integrated Voice & Data Networks (Morgan Kaufmann Series in Networking (Hardcover))*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Audouin et al., 2007] Audouin, O., D.Erasme, Jouvin, M., Leclerc, O., Mouton, C., Primet, P., Rodrigues, D., and Thual, L. (2007). CARRIOCAS project: An experimental high bit rate optical network for for computing intensive distributed applications. In *BroadBand Europe'07*.

[Avrachenkov et al., 2004] Avrachenkov, K., Ayesta, U., Brown, P., and Nyberg, E. (2004). Differentiation Between Short and Long TCP Flows: Predictability of the Response Time. In *Proceedings of the IEEE INFOCOM*.

[Awduche et al., 1998] Awduche, D. O., Agogbua, J., and McManus, J. (1998). An approach to optimal peering between autonomous systems in the internet. In *IC3N '98: Proceedings of the International Conference on Computer Communications and Networks*, page 346, Washington, DC, USA. IEEE Computer Society.

[Awerbuch et al., 1993] Awerbuch, B., Azar, Y., and Plotkin, S. (1993). Throughput-competitive on-line routing. In *SFCS '93: Proceedings of the 1993 IEEE 34th Annual Foundations of Computer Science*, pages 32–40, Washington, DC, USA. IEEE Computer Society.

[Baake and Wichmann, 1999] Baake, P. and Wichmann, T. (1999). On the economics of internet peering. *Netnomics*, 1(1):89–105.

[Banerjee and Mukherjee, 2000] Banerjee, D. and Mukherjee, B. (2000). Wavelength-routed optical networks: linear formulation, resource budgeting tradeoffs, and a reconfiguration study. *Networking, IEEE/ACM Transactions on*, 8(5):598–607.

[Baran, 1962] Baran, P. (1962). On distributed communications networks. Technical Report P-2626, RAND Corp.

[Baroncelli et al., 2005] Baroncelli, F., Martini, B., Valcarenghi, L., and Castoldi, P. (2005). A service oriented network architecture suitable for global grid computin g. *Optical Network Design and Modeling, 2005. Conference on*, pages 283–293.

[Bertsekas, 1998] Bertsekas, D. (1998). *Network optimization : continuous and discrete models.* Athena Scientific.

[Bolze et al., 2006] Bolze, R., Cappello, F., Caron, E., Daydé , M., Desprez, F., Jeannot, E., Jégou, Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P., Quetier, B., Richard, O., Talbi, E.-G., and Irena, T. (2006). Grid'5000: a large scale and highly reconfigurable experimental grid testbed. *International Journal of High Performance Computing Applications*, 20(4):481–494.

[Bonald and Massoulié, 2001] Bonald, T. and Massoulié, L. (2001). Impact of fairness on internet performance. *SIGMETRICS Perform. Eval. Rev.*, 29(1):82–91.

[Briscoe, 2007] Briscoe, B. (2007). Flow rate fairness: dismantling a religion. *SIGCOMM Comput. Commun. Rev.*, 37(2):63–74.

[Briscoe, 2006] Briscoe, B. T. B. (2006). Metcalfe's law is wrong. *IEEE Spectrum*, pages 26–31.

[Burchard, 2005] Burchard, L.-O. (2005). Networks with advance reservations: Applications, architecture, and performance. *J. Netw. Syst. Manage.*, 13(4):429–449.

[Campanella et al., 2006] Campanella, M., Krzywania, R., Reijs, V., and Sevasti, A. (2006). The bandwidth on demand service for the european research and education networks. In *Photonics in Switching, 2006. PS '06. International Conference on*, pages 1–4.

[Chang and Jamin, 2006] Chang, H. and Jamin, S. (2006). To peer or not to peer: Modeling the evolution of the internets as-level topology.

[Cheliotis, 2001] Cheliotis, G. (2001). *Structure and Dynamics of Bandwidth Markets.* PhD thesis, National Technical University of Athens.

[Chen and Vicat-Blanc Primet, 2007] Chen, B. and Vicat-Blanc Primet, P. (2007). Scheduling deadline-constrained bulk data transfers to minimize network congestion. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 410–417. IEEE Computer Society.

[Chiang et al., 2007] Chiang, M., Low, S., Calderbank, A., and Doyle, J. (2007). Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312.

[Chiu and Jain, 1989] Chiu, D.-M. and Jain, R. (1989). Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Comput. Netw. ISDN Syst.*, 17(1):1–14.

[Cho et al., 2006] Cho, K., Fukuda, K., Esaki, H., and Kato, A. (2006). The impact and implications of the growth in residential user-to-user traffic. *SIGCOMM Comput. Commun. Rev.*, 36(4):207–218.

[Ciulli et al., 2008] Ciulli, N., Carrozzo, G., Giorgi, G., Zervas, G., Escalona, E., Qin, Y., Nejabati, R., Simeonidou, D., Callegati, F., Campi, A., Cerroni, W., Belter, B., Binczewski, A., Stroinski, M., Tzanakaki, A., and Markidis, G. (2008). Architectural approaches for the integration of the service plane and control plane in optical networks. *Optical Switching and Networking*, 5(2-3):94–106. Advances in IP-Optical Networking for IP Quad-play Traffic and Services.

[Clark et al., 2005] Clark, D., Wroclawski, J., Sollins, K., and Braden, R. (2005). Tussle in cyberspace: defining tomorrow's internet. *Networking, IEEE/ACM Transactions on*, 13(3):462–475.

[Clark, 1995] Clark, D. D. (1995). The design philosophy of the darpa internet protocols. *SIGCOMM Comput. Commun. Rev.*, 25(1):102–111.

[Cormen et al., 2001] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms, Second Edition.* The MIT Press.

[Crowcroft, 2007] Crowcroft, J. (2007). Net neutrality: the technical side of the debate: a white paper. *SIGCOMM Comput. Commun. Rev.*, 37(1):49–56.

[Crowcroft et al., 2003] Crowcroft, J., Hand, S., Mortier, R., Roscoe, T., and Warfield, A. (2003). Qos's downfall: at the bottom, or not at all! In *RIPQoS '03: Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS*, pages 109–114, New York, NY, USA. ACM.

[De Leenheer et al., 2006] De Leenheer, M., Thysebaert, P., Volckaert, B., De Turck, F., Dhoedt, B., Demeester, P., Simeonidou, D., Nejabati, R., Zervas, G., Klonidis, D., and O'Mahony, M. (2006). A view on enabling-consumer oriented grids through optical burst switching. *Communications Magazine, IEEE*, 44(3):124–131.

[Denda et al., 2000] Denda, R., Banchs, A., and Effelsberg, W. (2000). The fairness challenge in computer networks. In *QofIS '00: Proceedings of the First COST 263 International Workshop on Quality of Future Internet Services*, pages 208–220, London, UK. Springer-Verlag.

[Dhamdhere and Dovrolis, 2008a] Dhamdhere, A. and Dovrolis, C. (2008a). Can isps be profitable without violating "network neutrality"? In *NetEcon '08: Proceedings of the 3rd international workshop on Economics of networked systems*, pages 13–18, New York, NY, USA. ACM.

[Dhamdhere and Dovrolis, 2008b] Dhamdhere, A. and Dovrolis, C. (2008b). Ten years in the evolution of the internet ecosystem. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 183–196, New York, NY, USA. ACM.

[Dukkipati and McKeown, 2006] Dukkipati, N. and McKeown, N. (2006). Why flow–completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev.*, 36(1):59–62.

[Duser and Bayvel, 2002] Duser, M. and Bayvel, P. (2002). Analysis of a dynamically wavelength-routed optical burst switched network architecture. *J. Lightwave Technol.*, 20(4):574.

[Edell and Varaiya, 1999] Edell, R. and Varaiya, P. (1999). Providing internet access: What we learn from index. *IEEE Network*, 13:18–25.

[Ellanti et al., 2005] Ellanti, M. N., Gorshe, S. S., Raman, L. G., and Grover, W. D. (2005). *Next Generation Transport Networks: Data, Management, and Control Planes*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Farrel and Bryskin, 2005] Farrel, A. and Bryskin, I. (2005). *GMPLS: Architecture and Applications (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Figuerola et al., 2007] Figuerola, S., Ciulli, N., de Leenheer, M., Demchenko, Y., Ziegler, W., and Binczewski, A. (2007). Phosphorus: single-step on-demand services across multi-domain networks for e-science. In Wang, J., Chang, G.-K., Itaya, Y., and Zech, H., editors, *Proceedings of SPIE, the International Society for Optical Engineering*, volume 6784. SPIE.

[Firoiu et al., 2002] Firoiu, V., Le Boudec, J.-Y., Towsley, D., Zhang, Z.-L., and Le Boudec, J.-Y. (2002). Theories and Models for Internet Quality of Service. *Proceedings of the IEEE*, 90(9):1565–1591.

[Fischer and Vöcking, 2004] Fischer, S. and Vöcking, B. (2004). On the evolution of selfish routing. In Albers, S. and Radzik, T., editors, *Proceedings of the 12th Ann. European Symp. on Algorithms (ESA)*, LNCS 3221, pages 323–334. Springer-Verlag.

[Floyd and Fall, 1999] Floyd, S. and Fall, K. (1999). Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Trans. Netw.*, 7(4):458–472.

[Floyd and Jacobson, 1993] Floyd, S. and Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413.

[Fusaro, 2002] Fusaro, P. C. (2002). *Energy convergence: the beginning of the multi-commodity market*. John Wiley and Sons.

[Gençata and Mukherjee, 2003] Gençata, A. and Mukherjee, B. (2003). Virtual-topology adaptation for wdm mesh networks under dynamic traffic. *IEEE/ACM Trans. Netw.*, 11(2):236–247.

[Griffin and Sobrinho, 2005] Griffin, T. G. and Sobrinho, J. L. (2005). Metarouting. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 1–12, New York, NY, USA. ACM.

[Grosso et al., 2009] Grosso, P., Marchal, D., Maassen, J., Bernier, E., Xu, L., and de Laat, C. (2009). Dynamic photonic lightpaths in the starplane network. *Future Generation Computer Systems*, 25(2):132–136.

[Gu and Grossman, 2007] Gu, Y. and Grossman, R. L. (2007). UDT: UDP-based data transfer for high-speed wide area networks. *Comput. Networks*, 51(7):1777–1799.

[Hall et al., 2007] Hall, A., Hippler, S., and Skutella, M. (2007). Multicommodity flows over time: Efficient algorithms and complexity. *Theor. Comput. Sci.*, 379(3):387–404.

[Hardin, 1968] Hardin, G. (1968). The tragedy of the commons. *Science*, 162:1243 – 1248.

[Hershberger et al., 2007] Hershberger, J., Maxel, M., and Suri, S. (2007). Finding the k shortest simple paths: A new algorithm and its implementation. *ACM Trans. Algorithms*, 3(4):45.

[Hurwicz and Reiter, 2006] Hurwicz, L. and Reiter, S. (2006). *Designing Economic Mechanisms.* Cambridge University Press.

[Iyer and McKeown, 2001] Iyer, S. and McKeown, N. (2001). Techniques for fast shared memory switches. Technical Report HPNG Technical Report - TR01-HPNG-081501, Stanford University.

[Jacobson, 1988] Jacobson, V. (1988). Congestion avoidance and control. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 314–329, New York, NY, USA. ACM.

[Jain and Varaiya, 2005] Jain, R. and Varaiya, P. (2005). Efficient market mechanisms for network resource allocation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 1056–1061.

[Jiang and Dovrolis, 2005] Jiang, H. and Dovrolis, C. (2005). Why is the internet traffic bursty in short time scales? *SIGMETRICS Perform. Eval. Rev.*, 33(1):241–252.

[Jukan and Karmous-Edwards, 2007] Jukan, A. and Karmous-Edwards, G. (2007). Optical control plane for the grid community. *Communications Surveys & Tutorials, IEEE*, 9(3):30–44.

[Kameda and Altman, 2008] Kameda, H. and Altman, E. (2008). Inefficient noncooperation in networking games of common-pool resources. *Selected Areas in Communications, IEEE Journal on*, 26(7):1260–1268.

[Kantawala and Turner, 2005] Kantawala, A. and Turner, J. (2005). Link buffer sizing: A new look at the old problem. In *ISCC '05: Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC'05)*, pages 507–514, Washington, DC, USA. IEEE Computer Society.

[Karol et al., 1987] Karol, M., Hluchyj, M., and Morgan, S. (1987). Input versus output queueing on a space division switch. *IEEE Trans. Commun.*, 35:1347–1356.

[Kelly et al., 1998] Kelly, F. P., Maulloo, A., and Tan, D. (1998). Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of Operation Research*, 49(3):237–252.

[Kleinrock, 1969] Kleinrock, L. (1969). Models for computer networks. In *Proceedings of the Int. Conf. on Communication*, Boulder, Colo.

[Kleinrock, 1976] Kleinrock, L. (1976). *Queueing Systems, Volume II: Computer Applications.* Wiley Interscience.

[Kleinrock, 2002] Kleinrock, L. (2002). Creating a mathematical theory of computer networks. *Oper. Res.*, 50(1):125–131.

[Kobayashi, 2006] Kobayashi, K. (2006). Transmission timer approach for rate based pacing TCP with hardware support. In *PFLDnet 2006*, Nara, Japan.

[Kodama et al., 2003] Kodama, Y., Kudoh, T., Takano, T., Sato, H., Tatebe, O., and Sekiguchi, S. (2003). GNET-1: Gigabit ethernet network testbed. In *Proceedings of the IEEE International Conference Cluster 2004*, San Diego, California, USA.

[Krithikaivasan et al., 2007] Krithikaivasan, B., Zeng, Y., Deka, K., and Medhi, D. (2007). ARCH-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic. *Networking, IEEE/ACM Transactions on*, 15(3):683–696.

[Kuipers and Van Mieghem, 2005] Kuipers, F. A. and Van Mieghem, P. F. A. (2005). Conditions that impact the complexity of qos routing. *IEEE/ACM Trans. Netw.*, 13(4):717–730.

[Kuri et al., 2003] Kuri, J., Member, S., Puech, N., Gagnaire, M., Dotaro, E., and Douville, R. (2003). Routing and wavelength assignment of scheduled lightpath demands. In *Procs. of ICOCN 2002, (Singapore*, pages 1231–1240.

[Le Boudec and Thiran, 2001] Le Boudec, J.-Y. and Thiran, P. (2001). *Network Calculus.* Lecture Notes in Computer Science (LNCS). Springer Verlag.

[Lee et al., 2008] Lee, S.-J., Banerjee, S., Sharma, P., Yalagandula, P., and Basu, S. (2008). Bandwidth-aware routing in overlay networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1732–1740.

[Liang and Han, 2007] Liang, Y. and Han, M. (2007). Dynamic bandwidth allocation based on online traffic prediction for real-time mpeg-4 video streams. *EURASIP J. Appl. Signal Process.*, 2007(1):51–51.

[Low et al., 2003] Low, S. H., Paganini, F., Wang, J., and Doyle, J. C. (2003). Linear stability of tcp/red and a scalable control. *Comput. Netw.*, 43(5):633–647.

[MacKie-Mason and Varian, 1995] MacKie-Mason, J. and Varian, H. (1995). Pricing congestible network resources. *Selected Areas in Communications, IEEE Journal on*, 13(7):1141–1149.

[Magoni and Pansiot, 2001] Magoni, D. and Pansiot, J. J. (2001). Analysis of the autonomous system network topology. *SIGCOMM Comput. Commun. Rev.*, 31(3):26–37.

[Majumdar et al., 2002] Majumdar, S., Vogelsang, I., and Cave, M. (2002). *Handbook of Telecommunications Economics*. North Holland.

[Manayya KB, 2009] Manayya KB (2009). Constrained shortest path first. IETF Internet Draft: `http://www.ietf.org/internet-drafts/draft-manayya-constrained-shortest-path-first-01.txt`.

[Marchal et al., 2005] Marchal, L., Vicat-Blanc Primet, P., Robert, Y., and Zeng, J. (2005). Optimizing network resource sharing in grids. volume 2, pages 6–840.

[Masip-Bruin et al., 2006] Masip-Bruin, X., Yannuzzi, M., Domingo-Pascual, J., Fonte, A., Curado, M., Monteiro, E., Kuipers, F., Mieghem, P. V., Avallone, S., Ventre, G., Aranda-Gutirrez, P., Hollick, M., Steinmetz, R., Iannone, L., and Salamatian, K. (2006). Research challenges in qos routing. *Computer Communications*, 29(5):563–581. Networks of Excellence.

[Massoulie, 2002] Massoulie, L. (2002). Stability of distributed congestion control with heterogeneous feedback delays. *Automatic Control, IEEE Transactions on*, 47(6):895–902.

[McKeown, 1999] McKeown, N. (1999). The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Trans. Netw.*, 7(2):188–201.

[McKeown and Anderson, 1998] McKeown, N. and Anderson, T. E. (1998). A quantitative comparison of iterative scheduling algorithms for input-queued switches. *Computer Networks and ISDN Systems*, 30(24):2309–2326.

[Medhi and Ramasamy, 2007] Medhi, D. and Ramasamy, K. (2007). *Network Routing*. Elsevier.

[Mills, 1994] Mills, D. L. (1994). Precision synchronization of computer network clocks. *SIGCOMM Comput. Commun. Rev.*, 24(2):28–43.

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

[Naiksatam and Figueira, 2007] Naiksatam, S. and Figueira, S. (2007). Elastic reservations for efficient bandwidth utilization in lambdagrids. *Future Gener. Comput. Syst.*, 23(1):1–22.

[Nisan et al., 2007] Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA.

[Nord et al., 2003] Nord, M., Bjornstad, S., and Gauger, C. (2003). Ops or obs in the core network. In *Proceedings of the 7th IFIP working conference on optical network design & modeling*. Presented at: ONDM 2003 : Budapest, Hungary, 2003.

[Nuyens and Wierman, 2008] Nuyens, M. and Wierman, A. (2008). The foreground-background queue: A survey. *Perform. Eval.*, 65(3-4):286–307.

[Obara, 1991] Obara, H. (1991). Optimum architecture for input queuing ATM switches. *Electronics Letters*, 27(7):555–557.

[Obara and Hamazumi, 1992] Obara, H. and Hamazumi, Y. (1992). Parallel contention resolution control for input queuing ATM switches. *Electron. Lett.*, 28(9):838–839.

[Ozdaglar and Bertsekas, 2003] Ozdaglar, A. E. and Bertsekas, D. P. (2003). Routing and wavelength assignment in optical networks. *IEEE/ACM Trans. Netw.*, 11(2):259–272.

[Park, 2005] Park, K. I. (2005). *Qos In Packet Networks (The Kluwer International Series in Engineering and Computer Science)*. Springer-Verlag TELOS, Santa Clara, CA, USA.

[Paul and Raghavan, 2002] Paul, P. and Raghavan, S. V. (2002). Survey of qos routing. In *ICCC '02: Proceedings of the 15th international conference on Computer communication*, pages 50–75, Washington, DC, USA. International Council for Computer Communication.

[Pióro and Medhi, 2004] Pióro, M. and Medhi, D. (2004). *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Qiao and Yoo, 1999] Qiao, C. and Yoo, M. (1999). Optical burst switching–a new paradigm for an optical internet. *Journal on High-Speed Networks*, 8(1):69–84.

[Resende and Pardalos, 2006] Resende, M. G. C. and Pardalos, P. M. (2006). *Handbook of Optimization in Telecommunications.* Springer, 1 edition.

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

[Roberts and Oueslati-Boulahia, 2000] Roberts, J. W. and Oueslati-Boulahia, S. (2000). Quality of Service by flow–aware networking. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1773):2197–2207.

[Roughgarden, 2003] Roughgarden, T. (2003). The price of anarchy is independent of the network topology. *J. Comput. Syst. Sci.*, 67(2):341–364.

[Roughgarden, 2002] Roughgarden, T. A. (2002). *Selfish routing.* PhD thesis, Cornell University, Ithaca, NY, USA. Adviser-Tardos,, Eva.

[Rouskas and Jackson, 2003] Rouskas, G. N. and Jackson, L. E. (2003). Optimal granularity of mpls tunnels. In *In Proceedings of the 18th International Teletraffic Congress (ITC-18*, pages 1–10. Elsevier Science.

[Sahinoglu and Tekinay, 2001] Sahinoglu, Z. and Tekinay, S. (2001). A novel adaptive bandwidth allocation: wavelet-decomposed signal energy approach. *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, 4:2253–2257.

[Saltzer et al., 1984] Saltzer, J. H., Reed, D. P., and Clark, D. D. (1984). End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288.

[Schrijver, 2003] Schrijver, A. (2003). *Combinatorial Optimization - Polyhedra and Efficiency.* Springer.

[Shenker, 1995] Shenker, S. (1995). Fundamental design issues for the future internet. *Selected Areas in Communications, IEEE Journal on*, 13(7):1176–1188.

[Shreedhar and Varghese, 1995] Shreedhar, M. and Varghese, G. (1995). Efficient fair queueing using deficit round robin. *SIGCOMM Comput. Commun. Rev.*, 25(4):231–242.

[Souza et al., 1994] Souza, R. J., Krishnakumar, P. G., Özveren, C. M., Simcoe, R. J., Spinney, B. A., Thomas, R. E., and Walsh, R. J. (1994). GIGAswitch System: A High-performance Packet-switching Platform. *Digital Technical Journal*, 6(1):9–22.

[Srikant, 2004] Srikant, R. (2004). *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications).* SpringerVerlag.

[Subramanian et al., 2004] Subramanian, L., Stoica, I., Balakrishnan, H., and Katz, R. H. (2004). Overqos: an overlay based architecture for enhancing internet qos. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pages 6–6, Berkeley, CA, USA. USENIX Association.

[Sun et al., 2007] Sun, C., Shi, L., Hu, C., and Liu, B. (2007). DRR-SFF: A Practical Scheduling Algorithm to Improve the Performance of Short Flows. In *ICNS '07: Proceedings of the Third International Conference on Networking and Services*, page 13.

[Takano et al., 2005] Takano, R., Kudoh, T., Kodama, Y., Matsuda, M., Tezuka, H., and Ishikawa, Y. (2005). Design and evaluation of precise software pacing mechanisms for fast long-distance networks. In *PFLDnet 2005*, Lyon, France.

[Tamir and Frazier, 1988] Tamir, Y. and Frazier, G. L. (1988). High-performance multi-queue buffers for VLSI communications switches. *SIGARCH Comput. Archit. News*, 16(2):343–354.

[Varghese, 2004] Varghese, G. (2004). *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices (The Morgan Kaufmann Series in Networking).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Varian, 1992] Varian, H. R. (1992). *Microeconomic Analysis.* W. W. Norton & Company.

[Varvarigos and Sharma, 1997] Varvarigos, E. A. and Sharma, V. (1997). The ready-to-go virtual circuit protocol: a loss-free protocol for multigigabit networks using fifo buffers. *IEEE/ACM Trans. Netw.*, 5(5):705–718.

[Vasseur et al., 2004] Vasseur, J.-P., Pickavet, M., and Demeester, P. (2004). *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Verdi et al., 2007] Verdi, F., Magalhães, M., Cardozo, E., Madeira, E., and Welin, A. (2007). A service oriented architecture-based approach for interdomain optical network services. *Journal of Network and Systems Management*, 15(2):141–170.

[Weltz et al., 2005] Weltz, M., He, E., Vicat-Blanc Primet, P., and al. (2005). Survey of protocols other than tcp. Technical report, Open Grid Forum. GFD 37.

[Widjaja, 1995] Widjaja, I. (1995). Performance analysis of burst admission-control protocols. *Communications, IEE Proceedings-*, 142(1):7–14.

[Wu et al., 2002] Wu, J., Savoie, J., and Arnaud, B. (2002). Functional requirements of peer-to-peer optical networking. volume 2, pages 1–2.

[Xiao, 2008] Xiao, X. (2008). *Technical, Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Xiong et al., 2000] Xiong, Y., Vandenhoute, M., and Cankaya, H. (2000). Control architecture in optical burst-switched WDM networks. *IEEE Journal on Selected Areas in Communications*, 18(10):1838–1851.

[Yang and Reddy, 1995] Yang, C.-Q. and Reddy, A. (1995). A taxonomy for congestion control algorithms in packet switching networks. *Network, IEEE*, 9(4):34–45.

[Yang et al., 2006] Yang, X., Lehman, T., Tracy, C., Sobieski, J., Gong, S., Torab, P., and Jabbari, B. (2006). Policy-based resource management and service provisioning in GMPLS networks. In *Adaptive Policy-based Management in Network Management and Control Workshop at INFOCOMM 2006*, Barcelona, Spain.

[Yao et al., 2000] Yao, S., Mukherjee, B., and Dixit, S. (2000). Advances in photonic packet switching: an overview. *IEEE Communications Magazine*, 38(2):84–94.

[Zalesky, 2009] Zalesky, A. (2009). To burst or circuit switch? *IEEE/ACM Trans. Netw.*, 17(1):305–318.

[Zhang, 1990] Zhang, L. (1990). Virtual clock: a new traffic control algorithm for packet switching networks. *SIGCOMM Comput. Commun. Rev.*, 20(4):19–29.

Standards, Recommandations & RFCs

[G.8, 2000] (2000). ITU-T recommendation g.805: Generic functional architecture of transport networks. Technical report, International Telecommunication Union.

[M.3, 2000a] (2000a). ITU-T recommendation m.3010: Principles for a telecommunications management network. Technical report, International Telecommunication Union.

[M.3, 2000b] (2000b). ITU-T recommendation m.3400: Tmn management functions. Technical report, International Telecommunication Union.

[G.8, 2001a] (2001a). ITU-T recommendation g.8080: Architecture for the automatically switched optical network (ason). Technical report, International Telecommunication Union.

[G.8, 2001b] (2001b). ITU-T recommendation g.872: Architecture of optical transport networks. Technical report, International Telecommunication Union.

[G.6, 2002] (2002). ITU-T recommendation g.694.1: Spectral grids for wdm applications: Dwdm frequency grid. Technical report, International Telecommunication Union.

[Allman et al., 1999] Allman, M., Paxson, V., and Stevens, W. (1999). TCP Congestion Control. RFC 2581 (Proposed Standard). Updated by RFC 3390.

[Almquist, 1992] Almquist, P. (1992). Type of Service in the Internet Protocol Suite. RFC 1349 (Proposed Standard). Obsoleted by RFC 2474.

[Blake et al., 1998] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. (1998). An Architecture for Differentiated Service. RFC 2475 (Informational). Updated by RFC 3260.

[Braden et al., 1994] Braden, R., Clark, D., and Shenker, S. (1994). Integrated Services in the Internet Architecture: an Overview. RFC 1633 (Informational).

[Braden et al., 1997] Braden, R., Zhang, L., Berson, S., Herzog, S., and Jamin, S. (1997). Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205 (Proposed Standard). Updated by RFCs 2750, 3936, 4495.

[Carpenter, 1996] Carpenter, B. (1996). Architectural Principles of the Internet. RFC 1958 (Informational). Updated by RFC 3439.

[Cerf et al., 1974] Cerf, V., Dalal, Y., and Sunshine, C. (1974). Specification of Internet Transmission Control Program. RFC 675.

[Davie et al., 2002] Davie, B., Charny, A., Bennet, J., Benson, K., Boudec, J. L., Courtney, W., Davari, S., Firoiu, V., and Stiliadis, D. (2002). An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246 (Proposed Standard).

[Farrel et al., 2006] Farrel, A., Vasseur, J.-P., and Ash, J. (2006). A Path Computation Element (PCE)-Based Architecture. RFC 4655 (Informational).

[Heinanen et al., 1999] Heinanen, J., Baker, F., Weiss, W., and Wroclawski, J. (1999). Assured Forwarding PHB Group. RFC 2597 (Proposed Standard). Updated by RFC 3260.

[Kompella and Lang, 2004] Kompella, K. and Lang, J. (2004). Procedures for Modifying the Resource reSerVation Protocol (RSVP). RFC 3936 (Best Current Practice).

[Kompella and Rekhter, 2005] Kompella, K. and Rekhter, Y. (2005). OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS). RFC 4203 (Proposed Standard).

[Lasserre and Kompella, 2007] Lasserre, M. and Kompella, V. (2007). Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling. RFC 4762 (Proposed Standard).

[Malkin, 1998] Malkin, G. (1998). RIP Version 2. RFC 2453 (Standard). Updated by RFC 4822.

[Meyer, 2001] Meyer, D. (2001). Extended Assignments in 233/8. RFC 3138 (Informational).

[Moy, 1998] Moy, J. (1998). OSPF Version 2. RFC 2328 (Standard).

[Nagle, 1984] Nagle, J. (1984). Congestion control in IP/TCP internetworks. RFC 896.

[Rekhter et al., 2006] Rekhter, Y., Li, T., and Hares, S. (2006). A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard).

[Rosen et al., 2001] Rosen, E., Viswanathan, A., and Callon, R. (2001). Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard).

[Shenker et al., 1997] Shenker, S., Partridge, C., and Guerin, R. (1997). Specification of Guaranteed Quality of Service. RFC 2212 (Proposed Standard).

[Swallow et al., 2005] Swallow, G., Drake, J., Ishimatsu, H., and Rekhter, Y. (2005). Generalized Multiprotocol Label Switching (GMPLS) User-Network Interface (UNI): Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Support for the Overlay Model. RFC 4208 (Proposed Standard).

[Vasseur and Roux, 2009] Vasseur, J. and Roux, J. L. (2009). Path Computation Element (PCE) Communication Protocol (PCEP). RFC 5440 (Proposed Standard).

[Wroclawski, 1997] Wroclawski, J. (1997). Specification of the Controlled-Load Network Element Service. RFC 2211 (Proposed Standard).

[Zakon, 1997] Zakon, R. (1997). Hobbes' Internet Timeline. RFC 2235 (Informational).