Revisiting the Fair Queueing Paradigm for End-to-End Congestion Control

A. Legout, and E. W. Biersack

Institut EURECOM
B.P. 193, 06904 Sophia Antipolis, FRANCE
{legout,erbi}@eurecom.fr
(Published in IEEE Network Magazine)

February 25, 2002

Abstract

Today, the dominant paradigm for congestion control in the Internet is based on the notion of TCP-friendliness. To be TCP-friendly, a source must behave in such a way as to achieve a bandwidth that is similar to the bandwidth obtained by a TCP flow that would observe the same Round Trip Time (RTT) and the same loss rate. However, with the success of the Internet comes the deployment of an increasing number of applications that do not use TCP as a transport protocol. These applications can often improve their own performance by not being TCP-friendly, which severely penalizes TCP flows. To design new applications to be TCP-friendly is often a difficult task.

The idea of the Fair Queueing (FQ) paradigm as a means to improve congestion control was first introduced by Keshav. While Keshav made a fundamental step toward a new paradigm for the design of congestion control protocols, he did not formalize his results so that his findings could be extended for the design of new congestion control protocols. We make this step and formally define the FQ paradigm as a paradigm for the design of new end-to-end congestion control protocols. This paradigm relies on FQ scheduling with per-flow scheduling and longest queue drop buffer management in each router. We assume only selfish and non-collaborative end users.

Our main contribution is the formal statement of the congestion control problem as a whole, which enables us to demonstrate the validity of the FQ paradigm. We also demonstrate that the FQ paradigm does not adversely impact the throughput of TCP flows and we explain how to apply the FQ paradigm for the design of new congestion control protocols. As a pragmatic validation of the FQ paradigm, we discuss a new multicast congestion control protocol called packet Pair receiver-driven Layered Multicast (PLM).

Keywords: Congestion Control, TCP-friendly, Fair Queueing, Paradigm.

1 Introduction

Congestion Control has been a central research topic since the early days of computer networks. Nagle first identified the problems of congestion in the Internet [18]. The fundamental turning point in Internet congestion control took place in the nineteen eighties. Nagle proposed a strategy based on *round robin* scheduling [19], whereas Jacobson proposed a strategy based on *Slow Start* (SS) and *Congestion Avoidance* (CA) [10]. Each of these solutions has its drawbacks: Nagle's solution has a high computational complexity and requires modifications to the routers. Jacobson's solution requires the collaboration of all end users. The modest performance of the routers and the small size of the Internet community at that time led to the

adoption of Jacobson's proposal. SS and CA were incorporated into TCP and more than ten years later the Internet still uses these mechanisms in a somewhat improved form [31].

We define the notion of a *Paradigm for Congestion Control* as a model to be used in designing congestion control protocols that have the same set of properties. Practically, when one designs a congestion control protocol with a paradigm, one has the guarantee that this protocol will have the same set of properties as all the other congestion control protocols designed with this paradigm. The benefits of the paradigm come from the set of properties it guarantees. However, the tradeoff is that the paradigm imposes some constraints that must be respected. This notion of a paradigm is not obvious in the Internet. A TCP-friendly paradigm was implicitly defined and introduced only a few years ago, when new applications that cannot use TCP had already been developed.

As TCP relies heavily on the collaboration of all the end users – collaboration in the sense of the common mechanism used to achieve congestion control – the TCP-friendly paradigm was introduced (see [21], [7]) to design congestion control protocols compatible with TCP.

A TCP-friendly flow must adapt its throughput T according to the equation:

$$T = \frac{C * MTU}{RTT * \sqrt{loss}} \tag{1}$$

where C is a constant, MTU is the packet size used, RTT and $loss$ are the round trip time and loss rates experienced by that flow. To compute the throughput T, one needs to measure the loss rates and the RTT. The TCP-friendly equation models the TCP long-term behavior for low loss rate. Padhye et al. [22] introduced an improved TCP-friendly equation that is a good approximation of TCP long-term behavior, even for high loss rate.

The throughput T for a TCP-friendly flow must decrease when its loss rate $loss$ increases. However, this behavior does not suit the needs of many applications. For example, audio and video applications are loss-tolerant and the degree of loss tolerance can be controlled with forward error correction [3]. While these applications can tolerate a certain loss rate without a significant decrease in the quality perceived by the end user, they cannot tolerate frequent variations of throughput. The TCP-friendly paradigm is also not appropriate for multicast flows as a source-based congestion control scheme for multicast flows must adapt its sending rate to the slowest receiver (in the sense of the throughput computed according to equation (1)). A receiver-based multicast congestion control scheme can be TCP-friendly only at the expense of a fine granularity in the choice of the layer bandwidths [17] [33].

The TCP-friendly paradigm requires that *all* the applications collaborate by adopting the same congestion control behavior based on Eq. (1). Given the current size of the Internet [7], such a collaboration can no longer be assumed. Applications start to use non-TCP-friendly congestion control schemes as they observe better performance for audio and video applications than with TCP-friendly schemes. However, the benefit reaped by non-TCP-friendly schemes is transitory and an increasing use of non-TCP-friendly schemes may lead to a congestion collapse in the Internet [18]. Congestion collapse occurs when an increase in the network load will result in a decrease in the useful work done by the network [7].

It is commonly agreed that router support can help congestion control. However, there are several concerns about router support. The end-to-end argument [28] is one of the major theoretical foundations of the Internet. The end-to-end argument states that a service should only be implemented in the network if the network can provide the full service, or if this service is useful for all clients. Adding functionality inside the routers must not violate this principle. As TCP is the main transport protocol used in the Internet, router support must not penalize TCP flows [24]. Moreover it is not clear which kind of router support is desirable: router support can range from simple buffer management to active networking. One of the major reasons the research community distrusts network support is the lack of a clear statement about the use of network support for congestion control.

One way to use network support for congestion control is to change the scheduling discipline inside the routers. Fair Queueing scheduling [12] is well known for its flow isolation property. However, the research community does not agree on the utility of this scheduling discipline for congestion control, even if its flow isolation property is appreciated, nor on the way to use this scheduling discipline. We strongly believe that the lack of consensus is due to a fuzzy understanding concerning the properties a congestion control protocol should have, and how a network, where each node implements FQ, can enforce these properties. The aim of this paper is to shed some light on these questions. There are several definitions of FQ that can lead to significantly different behaviors (see [1] or [32] for example). By *FQ scheduling*, we mean a per-packet approximation of a fluid Generalized Processor Sharing (GPS) scheduling; An *FQ policy* is a per-flow FQ scheduling with Longest Queue Drop (LQD) buffer management. A *FQ network* is a network where each router implements a FQ policy.

A user acts selfishly if he tries to maximize his own satisfaction without taking into account other users (Shenker gives a good discussion of the selfishness hypothesis [30]). The FQ paradigm is based on non-cooperative and selfish users.

In section 2 we formalize the FQ paradigm for end-to-end congestion control. In section 3, we study the practical aspects of the deployment of the FQ paradigm in the Internet. Section 4 compares the FQ paradigm and the TCP-friendly paradigm. Section 5 shows how we place our study compared to prior work, while section 6 summarizes our findings and concludes the paper.

2 The FQ Paradigm

We formalize the FQ paradigm in three steps.

- First, we define the notion of congestion. Our definition is a slight modification of Keshav's definition [12].
- Second, we formulate six properties that an ideal congestion control protocol must meet. These properties are abstractly defined, i.e., this means they are independent of any mechanism (for instance we talk about fairness but not about scheduling and buffer management, which are two mechanisms that influence fairness).
- Third, we give a formal definition of the FQ paradigm for congestion control. We show that almost all the properties of an ideal congestion control protocol are met by a congestion control protocol based on the FQ paradigm.

We note that all aspects of congestion control, from the definition of congestion to the definition of a paradigm to design new congestion control protocols, are addressed with the same formalism. This formalism allows us to carry out a consistent study of the congestion control problem.

2.1 Congestion

Before we can talk about congestion control we must define congestion. Congestion is a notion related both to user's satisfaction and to network load. If we only take into account user satisfaction, we can imagine a scenario where the user's satisfaction decreases, for instance, due to jealousy and not due to any modifications in the quality of the service a user receives. For example, user A learns that user B has a better service and is no longer satisfied with his own service. This situation can not be considered as congestion. If we only take into account network load, congestion is only related to network performance. However we insist that one must take into account user satisfaction since a network exists to satisfy users.

Our definition of **congestion** is: A network is said to be congested from the perspective of user *i*, if the satisfaction of *i* decreases due to a modification of the performance (bandwidth, delay, jitter, etc.) of his network connection.

A similar definition was first introduced by Keshav [12] who stated: "A network is said to be congested from the perspective of user i if the satisfaction of i decreases due to an *increase* in network load". Our only one point of disagreement with Keshav is about the influence of network load. He says that only an *increase* in network load that results in a decrease of satisfaction is a signal of congestion, whereas we say that a *modification* (increase or decrease) in network load with a decrease of satisfaction is a signal of congestion. The following example illustrates our point of view.

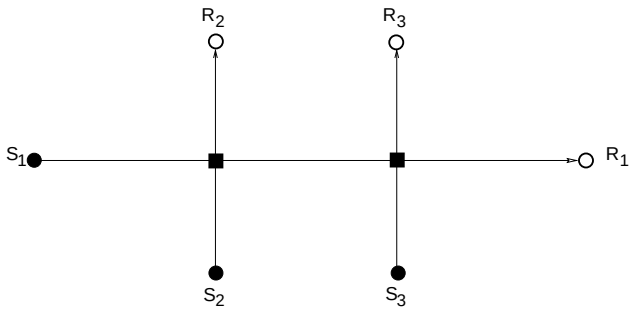


Figure 1: Example for the definition of congestion.

Let the scheduling be Weighted Fair Queueing (WFQ) [23], let the link capacity be 1 for all the links, and let the receiver satisfaction depend linearly on the bandwidth seen (see Fig. 1). The flow F_1 (sender S_1 and receiver R_1) has a weight of 1, the flow F_2 (sender S_2 and receiver R_2) has a weight of 2, the flow F_3 (sender S_3 and receiver R_3) has a weight of 1. Initially, the three sources have data to send, the satisfaction of R_1 is $\frac{1}{3}$, the satisfaction of R_2 is $\frac{2}{3}$, and satisfaction of R_3 is $\frac{2}{3}$. Then S_2 stops sending data, the satisfaction of R_1 becomes $\frac{1}{2}$ and the satisfaction of R_3 becomes $\frac{1}{2}$. This means that the network load decreases, but the satisfaction of R_3 decreases too. We consider this case as a congestion for R_3 according our definition, while Keshav's definition does not consider this case as congestion.

In the next section, we will address the properties of what we claim to be an ideal congestion control protocol. We want such a congestion control protocol to avoid congestion! This is not a trivial statement, since we want the congestion control protocol to avoid congestion as we have previously defined congestion.

2.2 Properties of an Ideal Congestion Control Protocol

In this section, terms from game theory and microeconomics are used; we define informally the terms used. Formal definitions can be found in [30]. A network reaches a *Nash equilibrium* if no user can increase his own satisfaction in an environment where every user acts selfishly. The bandwidth allocation A in a network is *Pareto optimal* if there does not exist another bandwidth allocation B such that under allocation B all users have at least as high a satisfaction as under allocation A and there is at least one user who has under allocation B a satisfaction that is strictly higher than under allocation A. A *utility function* formalizes the degree of satisfaction of a user for a given level of service. For each user, one can define a utility function that allows the comparison of his satisfaction for different levels of service. However, the scale of satisfaction is always relative to a given user. Therefore, it is not possible to compare utility functions for different users.

We discuss now a set of six abstract properties that an ideal congestion control protocol must meet. Whereas at first sight these properties seem similar to the ones given by Keshav [12], they are in fact fundamentally different. Most of our properties are expressed in well defined mathematical terms (for

example, Pareto optimality, max-min fairness) which enables us to demonstrate that a congestion control protocol meets these properties. In this paper, the only assumption we make is that users behave selfishly. The six properties that define an ideal congestion control protocol are:

Stability Given that each user is acting selfishly, we want the scheme to converge to a Nash equilibrium, where no user can increase their own satisfaction. This equilibrium makes sense from the point of view of congestion control stability. Since the existence of more than one Nash equilibrium can lead to oscillations among these equilibria, the existence and uniqueness of a Nash equilibrium are the conditions of stability.

Efficiency When the resource allocation is Pareto optimal, no user can have a higher satisfaction with any other allocation of the resources without decreasing the satisfaction of another user. The notion of Pareto optimality guarantees the efficiency of a congestion control protocol. The convergence time of the scheme is another important parameter for efficiency. The faster the convergence, the more efficient the congestion control protocol. A fast convergence towards a Pareto optimal distribution of the network resources is the condition of efficiency.

Fairness This is perhaps the most delicate part of congestion control. Many criteria for fairness exist, but there is no criterion agreed on by the whole networking community. We choose to use max-min fairness as we consider that this is a reasonable notion of fairness. If we consider for all users a utility function that is linearly dependent on the bandwidth seen, the max-min fair allocation is Pareto optimal. If a user has a non-linear utility function, with respect to the bandwidth seen, he will not be able to achieve his fair share, according to the definition of max-min fairness. Therefore max-min fairness gives an upper bound on the distribution of the bandwidth: If every user wants as much bandwidth as he can have, no user will have more than their max-min share. But, if some users are willing to cooperate they can achieve another kinds of fairness such as proportional fairness [11].

Robustness against misbehaving users. We presume that all users act selfishly. A selfish user must not decrease the satisfaction of another user. Moreover, he should not significantly modify the convergence speed of the scheme. Globally, the scheme must be robust against malicious, misbehaving, and greedy users.

Scalability The Internet evolves rapidly with respect to bandwidth, size, and the number of users. Inter-LAN, trans-MAN, and trans-WAN connections coexist. A congestion control protocol must scale on many axes: from an inter-LAN connections to a trans-WAN connections, or from a 28.8 Kbit/s modem to a 2.4 Gbit/s line. Moreover, a congestion control protocol must be capable of scaling with the number of receivers.

Feasibility comprises all the technical requirements. We restrict ourselves to the Internet architecture. The Internet connects a wide range of hardware and software systems and therefore a congestion control protocol must also cope with this heterogeneity. On the other hand, a congestion control protocol must be simple enough to be efficiently implemented. To be accepted as an international standard, a protocol needs to be extensively studied. Simplicity of a protocol will favor this process.

We claim that these six properties are necessary and sufficient properties to be able to define an ideal congestion control protocol. These properties cover all aspects of a congestion control protocol, from the theoretical notion of efficiency to the practical aspect of feasibility. However, it is not clear how we can design a congestion control protocol that meets all these properties. In the next section, we give a formal definition of the FQ paradigm for the design of congestion control protocols. We will show that the FQ paradigm meets five of the six properties of an ideal congestion control protocol as defined in this paper.

2.3 A Formal Definition of the FQ Paradigm and its Validation

A paradigm for congestion control is a model used to design new congestion control protocols. A paradigm makes assumptions and under these assumptions we can design compatible congestion control protocols; compatible means that the protocols have the same set of properties. Therefore, to formally define a paradigm, we must clearly express the *assumptions* made, and the *properties* guaranteed by the paradigm. To be viable in the Internet, the paradigm must be compliant with the end-to-end argument [28]. The congestion control protocols designed with the paradigm must be end-to-end and should not rely on specific network support. These issues are addressed in this section.

We first give a definition of a **Fair Queueing policy**. A Fair Queueing policy is a per-packet approximation of a fluid Generalized Processor Sharing (GPS) scheduling [23] with per-flow scheduling and Longest Queue Drop (LQD) buffer management.

In the remainder of the paper, the term *FQ policy* always refers to this definition (for brevity we use the notation *FQ* instead of *FQ-LQD with per-flow scheduling*). There are many approximations of the GPS scheduling policy (see [23], [5], and [2] for some examples). The better the approximation, the better the properties guaranteed by the FQ paradigm. The WF^2Q scheduling policy [1] is a good approximation of the GPS fluid model. We introduced the notion of flow in the definition of the Fair Queueing policy. A user typically has several utility functions, depending on the application considered. We define a flow as a set of packets that impact only one utility function for a given user. According to this definition, we could approximate a flow as an application level flow, however, a discussion about the notion of flow is pretty tricky and orthogonal to our study[1].

When defining the FQ paradigm, we make a distinction between the assumption that involves network support, which we call the Network Part of the paradigm (NP), and the assumptions that involve the end systems, which we call the End System Part of the paradigm (ESP). The assumptions required for the FQ paradigm are:

- For the NP of the paradigm we assume a *Fair Queueing* network that means a network where every router implements a Fair Queueing policy
- For the ESP, the end users are assumed to be selfish and non-cooperative. This is a sufficient but not a necessary condition. Cooperation among the users is possible if it increases their satisfaction.

We call this paradigm the Fair Queueing (FQ) paradigm, because like the TCP-friendly paradigm, we use the name of the mechanism central to the paradigm, namely the Fair Queueing policy, to qualify the paradigm. With the TCP-friendly paradigm, equation (1) guarantees efficiency, stability, and fairness, although not in the sense we defined these three properties in section 2.2. Since TCP guarantees efficiency, stability, and fairness by only one congestion control mechanism at the end system, compromises among the three properties are unavoidable. The idea of the FQ paradigm is to rely on the support of the network to guarantee the properties required for an ideal congestion control protocol as defined in this paper, and to let the protocol at the end system only address the application's needs. The FQ paradigm, unlike the TCP-friendly paradigm, does not make any assumptions about the congestion control mechanism used at the end systems. The FQ paradigm allows full freedom when designing the ESP part of an end-to-end congestion control protocol. This characteristic of the paradigm is very appealing and allows a great diversity in the congestion control mechanisms used.

We now discuss which of the six properties of an ideal congestion control protocol, as previously defined in this paper, are enforced by the FQ paradigm.

[1] We would like to point out that neither Fair queueing nor the TCP-friendly paradigm can prevent misbehaving users from using multiple separate flows to increase their share of resources.

Stability Under the NP and ESP hypothesis, the existence and uniqueness of a Nash equilibrium is assured (see [30])and therefore the congestion control protocols designed with the FQ paradigm meet the condition of stability.

Efficiency Under the NP and ESP hypothesis, even a simple optimization algorithm (like a hill climbing algorithm) quickly converges to the Nash equilibrium. However, the Nash equilibrium is not Pareto optimal in the general case. Therefore, the congestion control scheme designed with the FQ paradigm does not necessarily assure ideal efficiency. If all users have the same utility function, the Nash equilibrium is Pareto optimal. Ideal efficiency can be achieved with full cooperation of all users (see [30]), but this is contrary to the ESP assumptions.

Fairness The Fair Queueing policy achieves max-min fairness. Moreover, as a Fair Queueing policy is implemented in every network node, every flow achieves a max-min fair rate on the long term average (see [9]). Our NP assumption enforces fairness.

Robustness Using a Fair Queueing policy ensures that the network is protected against malicious, misbehaving, and greedy users (see [5]). In theory, a user could open multiple connections to increase its share of the bottleneck. However, in practice the number of connections that a single user can open is limited.

Scalability With the ESP assumption, end users are selfish and non-collaborative. This is clearly a sufficient condition for scalability. On the other hand, the TCP-friendly paradigm relies on *all* users collaborating by using the the same mechanism. As the user population increases, the assumption of collaborating users is obviously much more difficult to achieve than simply letting every user behave selfishly (as the FQ paradigm does).

Feasibility Today a Fair Queueing policy (HPFQ [2]) can be implemented in Gigabit routers (see [13]). It is often argued that per-flow scheduling cannot be implemented due to the high number of active flows crossing a backbone router. However, Kumar et al. [13] argue that there is no need to keep state for all active flows, but only for those active flows that are currently backlogged. Section 3.2 provides a discussion on the practical deployment of a Fair Queueing policy in the Internet.

We have presented the assumptions made and the properties enforced by the FQ paradigm. We saw that the only property the FQ paradigm does not necessarily assure is ideal efficiency.

According to the NP assumption, every network node implements a Fair Queueing policy. This mechanism is of broad utility: The Fair Queueing policy allows tradeoff among the three main performance parameters: bandwidth, delay, and loss (see [23]). This tradeoff cannot be made with the TCP-friendly paradigm.

The NP contains only the Fair Queueing assumption. In section 3.1 we will show that a Fair Queueing policy has a positive impact on TCP flows and that it does not violate the end-to-end argument [24]. Issues related to the practical introduction of the paradigm are presented and discussed in section 3.

The FQ paradigm, like the TCP-friendly paradigm, applies to both unicast and multicast, since the paradigm does not make any assumption concerning the transmission mode. Moreover, the FQ paradigm enforces properties that are of great benefit for multicast flows (see section 3.3).

To summarize, we have formally defined a simple paradigm for end-to-end congestion control, called the FQ paradigm, which relies on a Fair Queueing network and only makes the assumption that the end users are selfish and non-collaborative. The FQ paradigm, as opposed to the TCP-friendly paradigm, does not make any assumptions concerning the mechanism used by end users. Whereas the benefits of the FQ paradigm with respect to flow isolation are commonly agreed on by the research community, its benefits

for congestion control have been less clear: The ESP of the FQ paradigm gives a lot of flexibility in the design of end-to-end congestion control protocols tailored to the needs of the application. The NP of the FQ paradigm allows tradeoff between bandwidth, delay, and loss.

Any congestion control protocol designed according the FQ paradigm will automatically meet at least five of the six properties of an ideal congestion control protocol as defined in this paper. In section 3.3 we discuss how to design a new congestion control protocol according to the FQ paradigm.

3 Practical Aspects of the FQ Paradigm

In the previous sections, we gave a formal definition of the FQ paradigm. We now investigate the practical issues involved with the introduction of such a paradigm in the Internet.

3.1 Behavior of TCP with the FQ Paradigm

In this section we evaluate the impact of the NP assumption of the FQ paradigm on today's Internet. If we want to deploy the FQ paradigm in today's Internet, there is a central question: How will the use of a Fair Queueing policy that is deployed in all network nodes affect the behavior and performance of TCP flows? Suter showed the benefits of a FQ policy for TCP flows [32]. While his results are very promising, they are based on simulations for a very simple topology. We therefore must explore this issue further with simulations on a large topology.

The generation of realistic network topologies is a subject of active research [6]. It is commonly agreed that hierarchical topologies better represent a real Internetwork than do flat topologies. We use `tiers v1.1` ([6]) to create hierarchical topologies consisting of three levels: WAN, MAN, and LAN that aim to model the structure of the Internet topology [6] and we call this Random Topology *RT*. We give a brief description of the topology used for all the simulations. A WAN consists of 5 nodes and 6 links and connects 20 MANs, each consisting of 2 nodes and 2 links. To each MAN, 9 LANs are connected. Therefore, the core topology consists of $5 + 40 + 20 \cdot 9 = 225$ nodes. The capacity of WAN links is 155Mbit/s, the capacity of MAN links is 55Mbit/s, and the capacity of LAN links is 10Mbit/s. The WAN link delay is uniformly chosen in [100,150] ms, the MAN link delay is uniformly chosen in [20,40] ms, and the LAN link delay is 10 ms. Each LAN is represented as a single leaf node in the tiers topology. All the hosts connected to the same LAN are connected to the same leaf node and send their data on the same 10 Mbit/s link.

The Network Simulator NS [20] is considered to be the best simulator for the study of Internet protocols. We use NS with the topology generated by tiers. For each simulation, we choose either a small queue length (50 packets) or a large queue length (500 packets) for both the FIFO and FQ queues, this means the FQ shared buffer is 50 or 500 packets large. The buffer management used with FIFO scheduling is tail drop and the buffer management used with FQ scheduling is longest queue drop (LQD) with tail drop. The TCP flows are simulated using the NS implementation of TCP Reno, with a packet size of 1000 bytes and a receiver window of 5000 packets, large enough not to bias the simulations. The TCP sources always have a packet to send.

Our simulation scenarios are as follows: We add from $k = 50$ to $k = 1600$ TCP flows randomly distributed on the topology RT. This means that the source and the receiver of a flow are randomly distributed among the LANs. For each configuration of the TCP flows, we carry out an experiment with FIFO scheduling and an experiment with FQ scheduling; each time with a queue size of 50 and 500 packets. These experiments show the impact of the NP assumption on the throughput of unicast flows. All the simulations are repeated five times and the average is computed over the throughput values obtained during five repetitions. All the plots are with 95% confidence intervals. All the TCP flows start randomly within the first

simulated second and there are no flows arriving or departing during the simulation. Our aim, with this kind of scenario, is to avoid noise due to dynamic scenarios. At the beginning of a simulation, all the TCP sources must discover the available bandwidth and there is a high probability that the bottleneck queues will overflow during a slow start phase. Later, the additive increase, multiplicative decrease mechanism of TCP leads to an equilibrium where the bottleneck queue overflows during a congestion avoidance phase. Therefore, the TCP source sees only one loss per TCP cycle.

We compute the mean throughput $\bar{F}_i$ seen at the application (this is also sometimes referred to as goodput) over the entire simulation for each TCP flow i, $i = 1, ..., k$. We consider three metrics:

- The mean throughput $\bar{B} = \frac{1}{k} \sum_{i=1}^{i=k} \bar{F}_i$. $\bar{B}$ shows the efficiency of the scheduling discipline from the point of view of the satisfaction of users if we consider a utility function that is linearly dependent on the bandwidth seen for each receiver.

- The minimum throughput $\min_{i=1,...,k} F_i$ shows the worst case performance experienced by any receiver. We say that an allocation is max-min fair if the smallest assigned bandwidth seen by a user is as large as possible and, subject to that constraint, the second-smallest assigned bandwidth is as large as possible. (see [9]). Therefore, the minimum throughput shows which scheduling discipline leads to the bandwidth allocation closest to the max-min fair allocation.

- The standard deviation $\sigma = \sqrt{\frac{1}{k-1} \sum_{i=1}^{i=k} (\bar{F}_i - \bar{B})^2}$ gives an indication of the uniformity of the bandwidth distribution among users.

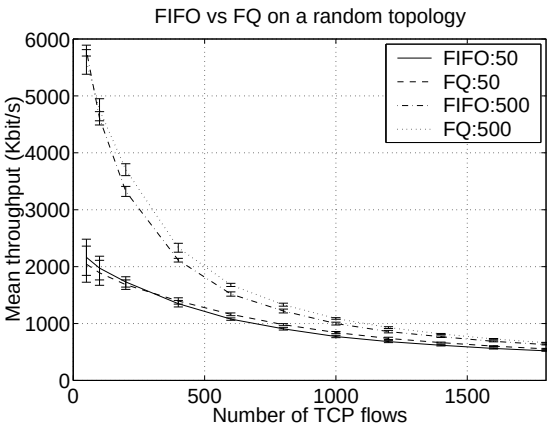
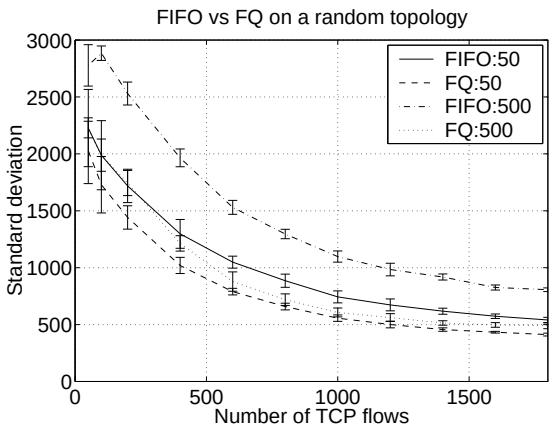


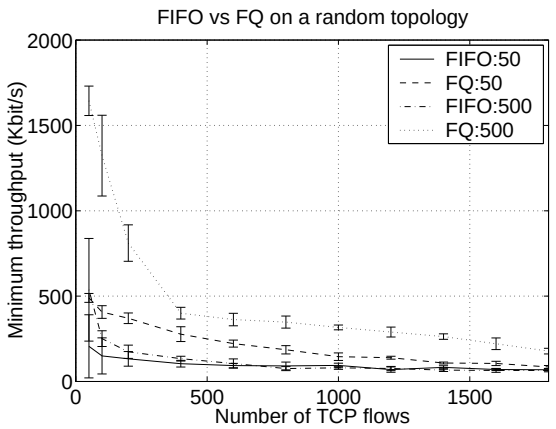
Figure 2: FIFO versus FQ, mean throughput $\bar{B}$ for an increasing the number of unicast flows $k = 50, ..., 1600$ and for two different queue lengths.

Fig. 2 shows the mean throughput for all receivers as a function of the number of TCP flows. Fig. 2 shows that a larger queue size leads to higher mean throughput. As the buffer size increases, the amount of time the bottleneck link is fully utilized also increases and therefore the mean throughput will increase.

Fig. 2 shows that FQ scheduling leads to a higher mean throughput than FIFO scheduling. For instance, for 1000 TCP flows ($k = 1000$) the mean throughput $\bar{B}$ obtained with FQ scheduling is 9% higher than with FIFO scheduling for both small and large queue sizes. Fig. 2 also shows that for a small number of TCP flows, the mean throughput obtained with FIFO scheduling is higher than with FQ scheduling. However, as the confidence intervals largely overlap (the mean value of one measure is contained in the confidence interval of the other), this result is not statistically significant.



(a) standard deviation σ of the mean throughput F_i.



(b) minimum throughput.

Figure 3: FIFO versus FQ, increasing the number of unicast flows $k = 50, ..., 1600$ and for two different queue lengths.

Figure 3 shows that FQ scheduling significantly improves fairness among the TCP flows. Fig. 3(a) shows that FQ scheduling always leads to a lower standard deviation than FIFO scheduling and the minimum throughput shown in Fig. 3(b) is higher with FQ scheduling than with FIFO scheduling. Therefore FQ scheduling provides a fairness closer to max-min fairness than FIFO scheduling.

In conclusion, our simulations show that FQ scheduling increases the stability of the system, improves the speed of convergence toward the equilibrium and the mean throughput of the TCP flows.

3.2 Remarks on the Deployment of the FQ Paradigm

One practical question concerning the FQ paradigm is its deployment in the Internet.

First, one can note that the issues concerning the deployment of the paradigm are only related to the deployment of the *Fair Queueing* policy in the routers. The deployment of end-to-end protocols is not an issue due to the NP assumption, since the paradigm does not constrain the end system. One can easily develop an end-to-end congestion control protocol for a new application and distribute this protocol with the application. One can develop end-to-end congestion control protocols for existing applications, thereby incrementally upgrading these applications without negative impact on the other applications. In fact, those which use the new protocol will see a significant enhancement in performance, whereas others, which have not been upgraded will not see a significant modification in their performance. Therefore the FQ paradigm allows an incremental deployment of new end-to-end protocols. This is not the case with the TCP-friendly paradigm, since it relies heavily on the collaboration of *all* end users. In the case of a collaborative paradigm, if one wants to add a new congestion control protocol, one must implement the same mechanism as in previous congestion control protocols. If one wants to change this mechanism, it must be changed in every end user, which is not feasible.

Second, the deployment of the NP requires that every router implements a Fair Queueing policy. If we deploy an end-to-end protocol without the NP assumption, we will almost certainly penalize applications that follow the TCP-friendly paradigm. Deploying the NP on the entire Internet seems unrealistic. However, the Internet is an interconnection of Internet Service Providers (ISPs) and we can exploit this fact: Each ISP has the full control of its network and can offer specific services on that network, independently of

the rest of the Internet. For example, some ISPs have started providing multicast functionality inside their network, whereas the Internet as a whole, is still not multicast capable[2]. ISPs are operating in a competitive environment that forces them to innovate and improve the service offered to the customers. In the past, ISPs have continuously upgraded the capacity of their links and, for example, installed caches to improve their service. If an ISP has installed caches, its clients will find the Web documents they access with a probability P (where P ranges between 0.5 and 0.7 according to [27]) in the ISP's cache. Upgrading all the routers within an ISP with a *Fair Queueing* policy will give a number of immediate benefits. Whenever a document is in the cache or on a server directly connected to the same ISP, customers surfing on the Web will experience a higher TCP throughput (around 10% higher, see section 3.1) and therefore shorter down-load times. If the ISP is also multicast capable, its clients can also use new end-to-end protocols that significantly improve the performance of the multicast connection, such as PLM [16]. The deployment of the FQ paradigm will be far easier for a new ISP that has no existing legacy infrastructure.

Given that the applications that use the FQ paradigm are deployed only *inside* an ISP, the deployment of the FQ paradigm can be incremental. For an ISP, upgrading all its routers with Fair Queueing is a substantial investment. However, the fact that this investment will improve the quality of the service can be a significant commercial argument. Therefore, ISPs have a financial interest in the deployment of this paradigm.

3.3 PLM: A Pragmatic Validation of the FQ Paradigm

In this section we explain how to apply the FQ paradigm for the design of a new congestion control protocol through an example: PLM. PLM stands for *packet Pair receiver-driven Layered Multicast*. We only provide an overview of the PLM protocol, which is described in detail in [16]. PLM is based on a cumulative layered scheme and on the use of the packet pair mechanism to infer the bandwidth available at the bottleneck in order to decide the appropriate layers of which to subscribe. PLM assumes that the routers are multicast capable, but does not make any assumption on the multicast routing protocol used. PLM is receiver-driven, so the burden of the congestion control mechanism is on the receiver's side. The only assumption we make about the sender is that he has the ability to send data via cumulative layers and to emit packets in pairs for each layer. The Packet Pair (PP) bandwidth inference mechanism introduced by Keshav [12] allows an explicit estimate of the available bandwidth to be obtained. Keshav showed that when one sends a PP, that is, two packets sent as fast as possible (back-to-back), onto a network where every router implements the Fair Queueing policy, the packets of the PP will be spaced out at the receiver by the available bandwidth on the path between the source and the receiver. The PP bandwidth inference mechanism is simple and does not have the drawbacks of the bandwidth inference mechanisms based on congestion signals such as loss. For PLM, at the receiver, we simply collect the PP estimates of the available bandwidth and add or drop layers according to these estimates. We do not add any mechanism to improve stability or fairness and PLM does not use any complex filtering mechanisms.

Our evaluation of the PLM protocol [16] showed that PLM meets most of the requirements of an ideal congestion control protocol as defined in this paper. PLM is stable, the receivers quickly converge to the available bandwidth and do not suffer from pathological oscillations. PLM is efficient and tracks the available bandwidth with no loss induced; even in an environment with self similar and multi-fractal traffic. PLM is fair with the other PLM sessions and with TCP. It is robust against misbehaving sources and is scalable because of the cumulative layered architecture. Finally, PLM is practically feasible and is a very simple protocol.

We show in [15] that the two most popular cumulative layered multicast congestion control protocols RLM [17] and RLC [33] suffer from pathological behaviors and that PLM outperforms both of them. In fact,

[2]We note similarities in the deployment of the multicast functionality per ISP and the deployment of the FQ paradigm per ISP as both require that all the routers support the respective capability.

most of the problems in RLM and RLC come from the bandwidth inference mechanism that must guarantee properties such as efficiency, stability, and fairness. The bandwidth inference mechanism of RLM and RLC is based on a congestion signal, such as loss. However, loss as a congestion signal has many weaknesses: the bottleneck queue must overflow and loss does not give information about the available bandwidth.

4 The FQ Paradigm versus the TCP-friendly Paradigm

TCP, which has been the main congestion control protocol for many years, has indisputably contributed to the stability and the efficiency of the Internet. However, every new congestion control protocol deployed in the Internet must be TCP-friendly.

Both the TCP-friendly and the FQ paradigms allow to design end-to-end congestion control protocols that are compatible with TCP. A paradigm is only a formal way of defining how to design congestion control protocols. To compare two paradigms we must look at the properties of the protocols designed with these paradigms. We compare the congestion control protocols according to the properties of an ideal congestion control protocol as defined in this paper. The results are summarized in Table 1 where a "+" sign indicates which paradigm outperforms the other one for a given property.

Properties	FQ paradigm	TCP-friendly paradigm
Stability	+	−
Efficiency	+	−
Fairness	+	−
Robustness	+	−
Scalability	−	−
Feasibility	−	+

Table 1: The FQ paradigm versus the TCP-friendly paradigm.

Since the TCP-friendly paradigm does not make any assumption about the scheduling discipline, it achieves neither ideal stability nor ideal efficiency as defined in this paper. With selfish users, only a Fair Queueing policy can lead to ideal stability and in some cases to ideal efficiency [30]. In general, the FQ paradigm does not lead to ideal efficiency. However, the FQ paradigm allows a tradeoff among the performance parameters: bandwidth, delay and loss, which is impossible with the TCP-friendly paradigm. The TCP-friendly paradigm does not lead to ideal fairness, as the fairness of this paradigm is biased by the RTT. The weak point of the TCP-friendly paradigm is its lack of robustness: It relies on the collaboration of end users. Both the TCP-friendly paradigm and the FQ paradigm are scalable.

The TCP-friendly paradigm is the most feasible paradigm because it does not require any modification to the existing Internet. The weak point of the FQ paradigm is feasibility since it requires a modification of the scheduling inside routers. We argued in section 3.2 that the deployment of the FQ paradigm is feasible per ISP and that ISPs have a financial interest in this deployment.

The FQ paradigm is an appealing model since, with reasonable support from the network, it considerably simplifies the design of new congestion control protocols. On the other hand, the design of new congestion control protocols with the TCP-friendly paradigm is one of the most complex problems in networking.

5 Prior Work

There is surprisingly little literature on congestion control *paradigms*. Most of the studies are about how to design TCP-friendly end-to-end congestion control schemes. See [8], [7], and [25] for unicast congestion control, and [17] and [26] for multicast congestion control.

Keshav [12] was the first to make a fundamental step towards a new paradigm for the design of congestion control protocols based on the assumption of an FQ network. While we agree with him on many points, our approach to the problem is fundamentally different. Keshav's aim was to study the problems of congestion control and to present a new unicast congestion control scheme as a solution. He introduced the idea of using a FQ policy inside the network in order to improve its own unicast congestion control protocol. However, he did not generalize his findings to the design of arbitrary congestion control protocols. We have introduced the formalism required to generalize Keshav's FQ paradigm to the design of new end-to-end congestion control protocols.

Shenker applies game theory to study congestion control [30]. He shows that one can achieve, with selfish and non-collaborative users, a congestion control that has a set of desired "good" properties. The only requirement is to have a scheduler with a *fair share allocation function*, such as an FQ scheduler. Shenker shows the benefits of the fair share policy for congestion control, but he does not clearly identify the properties of an ideal congestion control protocol, as defined in this paper and does not define a paradigm for designing congestion control protocols. Our formal definition of the notion of congestion and of the properties of an ideal congestion control protocol, as well as that of the FQ paradigm allows us to apply Shenker's result to demonstrate the validity of the FQ paradigm.

Lefelhocz et al. discuss a new paradigm for best effort congestion control [14] and provide an interesting discussion of the question: "Why do we need a new paradigm?" The solution proposed is a set of four mechanisms required for congestion control: scheduling, buffer management, feedback and end system adjustment. These mechanisms meet the FQ paradigm requirements: the scheduling and buffer management are part of what we call the Network Part; and the feedback and the end system adjustment are part of what we call the End System Part. Our study shows why these mechanisms are sufficient. We also show that selfish and non-collaborative end users can achieve nearly ideal congestion control. In their study, Lefelhocz et al. explain why they *believe* the four mechanisms are necessary and sufficient, we develop the formalism needed to *show* why they are necessary and sufficient. Our results can be seen as an extension of their study.

Suter et al. shows the benefits of FQ scheduling with appropriate buffer management for TCP flows [32]. Their discussion about the influence of the buffer management on the efficiency of an FQ per-flow scheduling policy very much inspires our definition of the policy that must be used with the FQ paradigm. However, their evaluation is based on very simple scenarios. We extend their study to a large topology and point out the most important issue with a FQ policy and TCP flows: that an FQ policy improves the stability and the speed of convergence toward an equilibrium of a system composed of TCP flows.

Servetto and Vetterli [29] study the problem of video multicast over an FQ network. They propose that the source sends, independent of the current network conditions, at a rate higher than its fair share. The routers then use FQ to drop the "excess" traffic and protect the competing flows due to the isolation property of FQ. However, the authors fail to address the problem that sources that determine their rate independent of network conditions may cause a congestion collapse of the network due to packets dropped inside the network before they reach their final destination [7].

Another way to design a paradigm for congestion control is the Diffserv or Intserv paradigm. There is active research on these topics, but to the best of our knowledge, there is no study similar to ours with these paradigms. Moreover the Diffserv and Intserv paradigms lead to much more complex mechanisms than the FQ paradigm. For example these paradigms are not viable without pricing (see [4]). We believe that even in a network with quality of service, a best effort class will always be popular and useful. The FQ paradigm is

a paradigm for best effort networks and, in particular, it applies to a best effort class.

6 Conclusion

We formalized the FQ paradigm for end-to-end congestion control protocols. This paradigm relies on a Fair Queueing network and makes the assumption that end users are selfish and non-collaborative. Whereas the FQ paradigm is commonly agreed to possess interesting properties, the research community has no clear understanding of what precisely these properties are.

Our study shows that the FQ paradigm provides a basis for designing congestion control protocols tailored to application needs. We do not propose replacing or modifying TCP; instead, we propose an alternative to the TCP-friendly paradigm which makes it possible to design *new*, efficient congestion control protocols compatible with TCP. It is important that the FQ paradigm does not violate the end-to-end argument due to the network support required. The network support that consists in changing the scheduling and buffer management in the routers is of broad utility. We demonstrate that the Fair Queueing policy significantly improves the performance of TCP connections and consequently, does not violate the end-to-end argument [24]. While one part of our results is implicitly addressed in previous work (in particular [12] and [30]), we go beyond an *implicit* definition of the problem to an *explicit* and formal statement of the problem. Moreover, we show how to apply the FQ paradigm for the design of new congestion control protocols and we discuss the PLM protocol that is a pragmatic validation of the FQ paradigm.

The main strength of the FQ paradigm is *the separation of the requirements imposed by the network and of the requirements of the end system*. There is no restriction on the end system when designing a new congestion control protocol and the FQ paradigm guarantees almost all the properties of an ideal congestion control protocol as defined in this paper. To the best of our knowledge, we are the first to define the properties of an ideal congestion control protocol and to define a FQ paradigm for the design of end-to-end congestion control protocols with such a formalism. Further, we demonstrate the validity of this paradigm, in reference to the definition of the properties of an ideal congestion control protocol.

The second part of our study concerns practical aspects relating to the introduction of the FQ paradigm in the Internet. Our simulations on a large topology demonstrate the benefits of the Fair Queueing policy for TCP flows. The Fair Queueing policy improves the stability of a system of TCP flows and increases the mean throughput of the TCP flows compared to the FIFO scheduling policy. As indicated, the incremental deployment by a single ISP will yield immediate benefits to the ISP's clients.

Acknowledgments

The authors would like to thank Peter O'Reilly and the anonymous reviewers for their helpful comments. Eurecom's research is partially supported by its industrial partners: Ascom, Cegetel, France Telecom, Hitachi, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

References

[1] J. Bennett and H. Zhang, "WF2Q: Worst-case Fair Weighted Fair Queueing", In *Proc. of IEEE INFOCOM'96*, pp. 120–128, San Francisco, CA, USA, March 1996.

[2] J. C. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms", *IEEE/ACM Transactions on Networking*, 5(5):675–689, October 1997.

[3] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-Based error control for Internet Telephony", In *Proc. of IEEE INFOCOM'99*, pp. 1453–1460, New York, March 1999.

[4] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, "Pricing in Computer Networks: Motivation, Formulation, and Example", *IEEE/ACM Transactions on Networking*, 1(6):614–627, December 1993.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", In *Proc. of ACM SIGCOMM'89*, pp. 1–12, Austin, Texas, September 1989.

[6] M. B. Doar, "A Better Model for Generating Test Networks", In *Proceedings of IEEE Global Internet*, pp. 86–93, London, UK, November 1996, IEEE.

[7] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[8] J. S. Golestani and S. Bhattacharyya, "End-to-End Congestion Control for the Internet: A Global Optimization Framework", In *Proc 6th Int. Conf. on Network Protocols*, pp. 137–150, October 1998.

[9] E. L. Hahne, "Round-Robin Scheduling for Max-Min Fairness in Data Networks", *IEEE Journal on Selected Areas in Communications*, 9(7):1024–1039, September 1991.

[10] V. Jacobson, "Congestion Avoidance and Control", In *Proc. of ACM SIGCOMM'88*, pp. 314–329, Stanford, CA, August 1988.

[11] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability", *Journal of the Operational Research Society*, 49:237–252, March 1998.

[12] S. Keshav, *Congestion Control in Computer Networks*, Ph.D. Thesis, EECS, University of Berkeley, CA 94720, USA, September 1991.

[13] V. P. Kumar, T. V. Lakshman, and D. Stiliadis, "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet", *IEEE Communications Magazine*, 36(5):152–164, May 1998.

[14] C. Lefelhocz, B. Lyles, S. Shenker, and L. Zhang, "Congestion Control for Best-Effort Service : Why We Need a New Paradigm", *IEEE Network*, pp. 10–19, January/February 1996.

[15] A. Legout and E. W. Biersack, "Pathological Behaviors for RLM and RLC", In *Proc. of NOSSDAV'00*, pp. 164–172, Chapel Hill, North Carolina, USA, June 2000.

[16] A. Legout and E. W. Biersack, "PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes", In *Proc. of ACM SIGMETRICS'2000*, pp. 13–22, Santa Clara, CA, USA, June 2000.

[17] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast", In *SIGCOMM 96*, pp. 117–130, August 1996.

[18] J. Nagle, "Congestion control in TCP/IP internetworks", *Computer Communication Review*, 14(4):11–17, October 1984.

[19] J. Nagle, "On packet switches with infinite storage", *IEEE Transactions on Communications*, COM-35(4):435–438, April 1987.

[20] NS, UCB/LBNL/VINT Network Simulator - ns (version 2), http://www.isi.edu/nsnam/ns.

[21] T. Ott, J. Kemperman, and M. Mathis, "The stationay distribution of ideal TCP Congestion Avoidance", Technical report, Bellcore, August 1996.

[22] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", In *Proc. of ACM SIGCOMM'98*, pp. 303–314, Vancouver, Canada, August 1998.

[23] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks", In *Proc. IEEE INFOCOM'93*, pp. 521–530, 1993.

[24] D. P. Reed, J. H. Saltzer, and D. D. Clark, "Commentaries on Active Networking and End to End Arguments", *IEEE Network*, 12(3):66–71, May/June 1998.

[25] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-End Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", In *Proc. of IEEE INFOCOM'99*, New York, NY, USA, March 1999.

[26] I. Rhee, N. Ballaguru, and G. N. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast", Technical report TR-98-01, North Carolina State University, North Carolina, January 1998.

[27] P. Rodriguez, K. W. Ross, and E. W. Biersack, "Distributing Frequently-Changing Documents in the Web: Multicasting or Hierarchical Caching", *Computer Networks and ISDN Systems. Selected Papers of the 3rd International Caching Workshop*, pp. 2223–2245, 1998.

[28] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to end arguments in system design", *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[29] S. D. Servetto and M. Vetterli, "Video Multicast over Fair Queueing Networks", In *Proc. of ICIP'00*, Vancouver, BC, September 2000.

[30] S. Shenker, "Making Greed Work in Networks: A Game-Theoric Analysis of Switch Service Disciplines", In *Proc. of ACM SIGCOMM'94*, pp. 47–57, London, UK, October 1994.

[31] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", Request for Comments RFC 2001, Internet Engineering Task Force, January 1997.

[32] B. Suter, T. V. Lakshman, D. Stiliadis, and A. Choudhury, "Design Considerations for Supporting TCP with Per-flow Queueing", In *Proc. of IEEE INFOCOM'98*, pp. 299–306, April 1998.

[33] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer", In *Proc. of IEEE INFOCOM'98*, pp. 996–1003, San Francisco, CA, USA, March 1998.