Beyond TCP-Friendliness: A New 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

November 15, 1999

Eurecom Technical Report

Abstract

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" and severely penalizing TCP streams. Also, designing these new applications to be "TCP-friendly" is often a difficult task. For these reasons, we propose a new paradigm for end-to-end congestion control (the FS paradigm) that relies on a Fair Scheduler network and assumes only selfish and non-collaborative end users. The flow isolation property of the FS paradigm is commonly agreed by the network community, however the lack of formalism of the FS paradigm hides fundamental properties.

We rigorously define the properties of an ideal congestion control protocol and show that the FS paradigm allows to devise end-to-end congestion control protocols that meet almost all the properties of an ideal congestion control protocol. The FS paradigm is fully compatible with the TCP flows. Moreover, we show that the incremental deployment of the FS paradigm is feasible per ISP and leads to immediate benefits for the TCP flows since their mean bandwidth is increased by up to 25%.

Our main contribution is the formal statement of the congestion control problem as a whole that allows to rigorously prove the validity of the FS paradigm.

Keywords: Congestion Control, Scheduling, Paradigm, Multicast, Unicast.

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[1]. The first fundamental turning point in Internet congestion control took place at the end of the eighties. Nagle proposed a strategy based on the round robin scheduling [2], whereas Jacobson proposed a strategy based on Slow Start (SS) and Congestion Avoidance (CA) [3]. Each of these solutions has its drawbacks. Nagle's solution has a high computation complexity and requires modifications to the routers. Jacobson's solution requires the collaboration of all the end users[1]. The low 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 mechanisms were put in TCP. Ten years later, the Internet still uses Jacobson's mechanisms in a somewhat improved form [4].

We define the notion of *Paradigm for Congestion Control* as a model used to devise congestion control protocols that have the same set of properties. Practically, when one devises a congestion control protocol with a paradigm, one has the guarantee that this protocol will be compatible[2] with all the other congestion control protocols devised with the same

[1]The term end user refers to all the entities that control the end host. For instance when we assume the collaboration of the end user we assume the collaboration of the protocol at the end host, and the collaboration of the persons who use the end host and can modify the protocol, etc.

[2]Compatible means that this protocol has a same set of properties than all the other congestion control protocols devised with this paradigm.

paradigm, at the expense of some constraints enforced by the paradigm. This notion of paradigm is not obvious in the Internet. A TCP-friendly paradigm was implicitly defined. However this paradigm was introduced after TCP, when new applications that can not use TCP had already appeared.

As TCP relies heavily on the collaboration of all the end users – collaboration is in the sense of the common mechanism used to achieve congestion control – the TCP-friendly paradigm was introduced (see [5], [6]) to devise congestion control protocols compatible to TCP. A TCP-friendly flow has to adapt its throughput T according to the equation[3] :

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

where, C is a constant, MTU is the size of the packets used for the connection, RTT is the round trip time, and $loss$ is the loss rate experienced by the connection. To compute this T, one needs to measure the loss rate and the RTT.

The throughput T for a TCP-friendly flow heavily decreases with the loss rate $loss$. However, this behavior does not fit to many applications' requirements. For instance, audio and video applications are loss-tolerant and the degree of loss tolerance can be managed with FEC [8]. These multimedia applications can tolerate a significant loss rate without a significant decrease in the quality perceived by the end users. The multicast flows suffer from TCP-friendliness since a source-based congestion control scheme for multicast flows has to adapt its sending rate to the worst receiver (in the sense of the loss rate), to follow the TCP-friendly paradigm. A receiver-based multicast congestion control scheme can be TCP-friendly but at the expense of a large granularity in the choice of the layer bandwidth [9] [10].

The TCP-friendly paradigm relies on the collaboration of all the users, which can not be longer assumed given the current size of the Internet [6]. This paradigm requires that *all* the applications use the same congestion control mechanism based on Eq. (1). This paradigm does not extend to the new applications being deployed across the Internet. Companies start to use non-TCP-friendly congestion control schemes[4], as they observe better performance for audio and video applications than with TCP-friendly schemes. However the benefit due to non-TCP-friendly schemes is a transitory effect and an increasing use of non-TCP-friendly schemes may lead to a congestion collapse in the Internet. Indeed, at the present time, most of the users access the Internet at 56 Kbps or less. However, with the deployment of xDSL most of the users will have, in a few years, an Internet access at more than 1Mbps. It is easy to imagine the disastrous effect of hundred of thousands unresponsive flows at 1 Mbps crossing the Internet.

It is commonly agreed that router support can help congestion control. However there are several fears about router support. The end-to-end argument [11] is one of the major theoretical foundations of the Internet, adding functionality inside the routers must not violate this principle. As TCP is the main congestion control protocol used in the Internet, router support must, at least, not penalize TCP flows (this can be related to the end-to-end argument, see[12]). 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 simple way to use network support for congestion control is to change the scheduling discipline inside the routers. PGPS-like scheduling [13] is well known for its flow isolation property. This property sounds suitable for congestion control. However, the research community does neither 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 about which properties a congestion control protocol should have and how a PGPS network (i.e. a network where each node implements a PGPS-like scheduler) can enforce these properties. The aim of this paper is to shed some light onto these questions.

A user acts selfishly if he only tries to maximize its own satisfaction without taking into account the other users (Shenker gives a good discussion about the selfishness hypothesis in [14]). The TCP-friendly paradigm is based on cooperative and selfish users. We base our new paradigm called Fair Scheduler (FS) paradigm on non-cooperative and

[3]The TCP-friendly equation is based on TCP long-term behavior, there are more accurate definitions of the TCP-friendly equation (see [7]) but they are always based on a function of the RTT and the $\sqrt{loss}$.

[4]Here congestion control may be a misleading expression, since the flows are often constant bit rate.

selfish users. We formally define the properties of an ideal congestion control protocol (see section 2.2) and show that almost all these properties are verified with the FS paradigm when we assume a network support that simply consist in having a Fair Scheduler in the routers (see section 2.3). We define a Fair Scheduler to be a Packet Generalized Processor Sharing scheduler with longest queue drop buffer management(see [13], [15], [16], and [17] for some examples). In particular, a Fair Scheduler must guarantee max-min fairness and delay bounds. Our study shows that simply changing the scheduling allows to use the FS paradigm for congestion control while outperforming the TCP-friendly paradigm. Indeed, the FS paradigm provides a basis for devising congestion control protocols tailored to the application needs. We do not introduce a new congestion control protocol, but a *model* (a paradigm) to devise efficient congestion control protocols. We do not want to replace or modify TCP. Instead, we propose an alternative to the TCP-friendly paradigm to devise *new* congestion control protocols compatible with TCP. Important to us is that the FS paradigm does not violate the end to end argument, due to the network support. The weak network support that consists in changing the scheduling is of broad utility–we show that the FS scheduling significantly improves the performances of the TCP connections– and consequently does not violate the end-to-end argument [12]. We can note that one part of our results are implicitly addressed in previous work (in particular [18] and [14]), we are making the step from an *implicit* definition of the problems and an *explicit* statement of the problem introducing a formalism that constitutes an indisputable contribution. We expect this study will stimulate new interest for this FS paradigm, fully compatible with TCP congestion control, that allows to devise end-to-end congestion control protocols that meet almost all the properties of an ideal congestion control protocol.

In section 2 we define the FS paradigm for end-to-end congestion control. In section 3, we study the practical aspects of the deployment of the FS paradigm in the Internet. Section 4 compares the FS paradigm and the TCP-friendly paradigm. Section 5 addresses the related work, while section 6 summarizes our findings and concludes the paper.

2 The FS Paradigm

We formally define the FS paradigm in three steps. First, we define the notion of congestion. This definition is a slight modification of the Keshav's definition[18]. Second, we formulate six properties that an ideal congestion control protocol must meet. These properties are abstractly defined, i.e. 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 define the FS paradigm for congestion control. We show that almost all the properties of an ideal congestion control protocols are met by a congestion control protocol based on the FS paradigm.

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

2.1 Definition of Congestion

The first point to clarify when we talk about congestion control is the definition of congestion. Congestion is a notion related to both user's satisfaction and network load. If we only take into account the user's satisfaction, we can imagine a scenario, where the user's satisfaction decreases due to jealousy (for instance) and not due to any modifications in the quality of the service a user receives (for instance, user A learns that user B has a better service, and is no more satisfied with his own service). This can not be considered as congestion. If we only take into account the network load, congestion is only related to network performance, which can be a definition of congestion (for instance it is the definition in TCP), but we claim that we must take into account the user's satisfaction. We always have to keep in mind that a network exists to satisfy users. Our definition of congestion is:

Definition 1 *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 characteristics of his network connection*[5].

[5]The "characteristics of a connection" are defined by the performances characteristics observed by a user on his session.

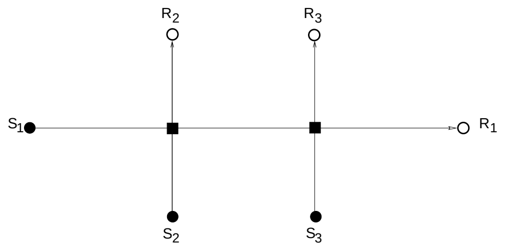


Figure 1: Example for the definition of congestion

A similar definition was first introduced by Keshav (for a discussion of this definition see [18]). Keshav's initial definition is : "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 loss of satisfaction is a signal of congestion, whereas we claim that a *modification* (increase or decrease) in network load with a decrease of satisfaction is a signal of congestion. We give an example to illustrate our view.

Let the scheduling be WFQ [13], let the link capacity be 1 for all the links, and let the receiver's satisfaction depend linearly on the bandwidth received. 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. In a first time 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 $\frac{1}{2}$. So when S_2 stops to send data, the network load decreases, but the satisfaction of R_3 decreases too. We consider this case as a congestion for R_3 in our definition, while Keshav's definition does not consider this case as congestion.

In the next section we will address the properties of an ideal congestion control protocol. We want this congestion control protocol to avoid congestion! This is not trivial, in fact we want the congestion control protocol to avoid congestion in the sense of the congestion previously defined. This link is fundamental as it contributes to the consistency of our study.

2.2 Properties of an Ideal Congestion Control Protocol

We use through this section terminology from the game theory and microeconomics; we define informally[6] the terms used. A network reaches a *Nash equilibrium* if, when every user acts selfishly, nobody can increase its own satisfaction. The bandwidth allocation A in a network is *Pareto optimal* if it does not exist another bandwidth allocation B such that all the users have a satisfaction with B higher or equal than the satisfaction with A, and at least one user has a satisfaction with B strictly higher than the satisfaction with A.

We discuss in the following a set of six abstract properties that an ideal congestion control protocol must verify. Whereas at the first sight these properties seem similar to Keshav ones, they are fundamentally different. Indeed, most of our properties are expressed in mathematical terms that allow to rigorously prove that a congestion control protocol verifies these properties. Here, the only one assumption we make is the selfish behavior of the users. So these properties remain very general. The six properties are:

Stability Given each user is acting selfishly, we want the scheme to converge to a Nash equilibrium. At Nash equilibrium, nobody can increase its own satisfaction. So this equilibrium makes sense from the point of congestion control stability. Since more than one Nash equilibrium can lead to oscillation among these equilibria, the existence and the uniqueness of a Nash equilibrium are the conditions of stability.

Efficiency Nash equilibrium does not mean efficiency. A desired property for the Nash equilibrium is to be Pareto optimal. In this case, nobody can have a higher satisfaction with another distribution of the network resources without

[6]The interested reader can refer to [14] for formal definitions.

decreasing the satisfaction of another user. The convergence time of the scheme towards the Nash equilibrium is another important parameter for efficiency. The faster convergence is, the more efficient it is. A fast convergence towards a Nash equilibrium that leads to a Pareto optimal distribution of the network resources is the condition of efficiency.

Fairness It 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 use max-min fairness (see [19])[7]. We make a fundamental remark on this fairness property. If we consider for all the users a utility function that is linearly dependent on the bandwidth received, max-min fairness is equivalent to pareto optimality. If a user does not have a utility function that depends linearly on the bandwidth received he will not be able to achieve its fair share (in the sense of max-min fairness). Therefore max-min fairness defines/imposes an upper bound on the distribution of the bandwidth: If every user wants as much bandwidth as he can have, nobody will have more than its max-min share. But if some users are willing to collaborate[8] they can achieve another kind of fairness and in particular proportional fairness[20].

Robustness against misbehaving users. We suppose that all the users act selfishly, and as there is no restriction on the utility functions, the behavior of the users can be very aggressive. Such a user must not decrease the satisfaction of the other users. Moreover, he should not significantly modify the convergence speed of the scheme towards a Nash equilibrium (see the efficiency property). Globally, the scheme must be robust against malicious, misbehaving, and greedy users.

Scalability The Internet evolves rapidly with respect to bandwidth and size. Moreover inter-LAN, trans-MAN, and trans-WAN connections coexist. A congestion scheme must scale on many axes: from an inter-LAN connection to a trans-WAN connection, from a 28.8Kbyte/s modem to a 155Mbit/s line.

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

We believe that these properties are necessary and sufficient properties of an ideal congestion control protocol. Indeed these properties cover all the 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 devise a congestion control protocol that meets all these properties. In the next section we establish the FS paradigm that allows to devise congestion control protocols that assure almost all of congestion control properties.

2.3 Definition and Validity of the FS Paradigm

A paradigm for congestion control is a model used to devise new congestion control protocols. A paradigm makes assumptions and under these assumptions we can devise compatible congestion control protocols; compatible means that the protocols have a same set of properties. Therefore, to define a new paradigm, we must clearly express the *assumption* made and the *properties* enforced by the paradigm. To be viable in the Internet the paradigm must be compliant with the end-to-end argument [11]. Mainly the congestion control protocols devised with the paradigm have to be end-to-end and should not have to rely on specific network support. These issues are addressed in this section.

[7]Fairness is a tricky property. By definition, fairness means fairness among many flows. We do not make any assumption about the congestion control scheme of the other flows, however we want that a flow F_r regulated by a congestion control scheme that follows the properties of this section achieves max-min fairness with the other flows. In particular, F_r crossing TCP flows has to achieve max-min fairness with the TCP flows.

[8]In the FS paradigm we assume selfish and non-collaborative end users, however we do not exclude collaborative protocols. We just do not need this collaborative assumption to achieve good properties.

For simplicity things we make a distinction between the assumption that involves the network support – we call that the Network Part of the paradigm (NP) – and the assumptions that involve the end systems – we call that the End System Part of the paradigm (ESP).

The assumptions required for our new paradigm are:

- For the NP of the paradigm we assume a *Fair Scheduler* network, i.e. a network where every router implements a Fair Scheduler;
- For the ESP, the end users are assumed to be selfish and non-collaborative.

We call this paradigm the Fair Scheduler (FS) paradigm[9]. We can note that the FS paradigm, unlike the TCP-friendly paradigm, does not make any assumptions on the mechanism used at the end systems. The FS paradigm guarantees full freedom when devising a congestion control protocol. This property of the paradigm is appealing but may lead to a high heterogeneity of the congestion control mechanisms used. Therefore, one can have a legitimate fear about the set of properties enforced by the FS paradigm. If the FS paradigm enforces less properties than the TCP-friendly paradigm, the FS paradigm does not make sense. In fact we show, in the following, that our simple FS paradigm enforces almost all the properties of an ideal congestion control protocol and consequently outperforms the TCP-friendly paradigm.

Stability Under the NP and ESP hypothesis, the existence and uniqueness of a Nash equilibrium is assured (see [14]). The congestion control protocols devised with the FS paradigm therefore meet the condition of stability.

Efficiency Under the NP and ESP hypothesis, even a simple optimization algorithm (like a hill climbing algorithm) converges fast to the Nash equilibrium. However, the Nash equilibrium is not Pareto optimal in the general case. If all the users have the same utility function, the Nash equilibrium is Pareto optimal. One can point out that ideal efficiency can be achieved with full collaboration of the users (see [14]). However, it is contrary to the ESP assumptions. The congestion control scheme devised with our new paradigm does not have necessarily ideal efficiency.

Fairness Every fair scheduler achieves max-min fairness. Moreover, as a Fair Scheduler is implemented in every network node, every flow achieves its max-min fairness rate on the long term average (see [21]). Our NP assumption enforces fairness.

Robustness Using a Fair Scheduler enforces that the network is protected against malicious, misbehaving, and greedy users (see [16]). We can note that one user opening multiple connections can increase its share of the bottleneck, however in practice, the number of connections that a single user can open is limited. Therefore, we do not expect this multiple connections effect to be a significant weakness of the robustness property.

Scalability According to the ESP assumption, the only one constraint of the end-to-end protocol designer must consider are selfish and non-collaborative end users. Unlike the TCP-friendly paradigm the designer has a great flexibility to devise scalable end-to-end congestion control protocols with the FS paradigm.

Feasibility A fair scheduler (HPFQ [17]) can be implemented today in Gigabit routers (see [22]). So the practical application of the NP assumption is no longer an issue (see section 3.2 for a discussion on the practical deployment of Fair Schedulers in the Internet).

We see that the FS paradigm does not allow to devise an ideal efficient congestion control protocol, because the Nash equilibrium can not be guaranteed to be Pareto optimal. The simple case that consists in considering the user satisfaction of everyone as the same linear function of the bandwidth received leads to ideal efficiency (as every user has the same utility function). However, in the general case ideal efficiency is not achieved. According to the NP assumption,

[9]Like the TCP-friendly paradigm, we compose the name of our new paradigm using the name of the fundamental mechanism involved in the paradigm, namely the Fair Scheduler.

every network node implements a Fair Scheduler, so we can manage the tradeoff among the three main performance parameters: bandwidth, delay, and loss (see [13]). This tradeoff can not be made with the TCP-friendly paradigm, therefore our paradigm leads to a significantly higher efficiency (in the sense of the satisfaction of end users) than the TCP-friendly paradigm.

We have given the assumptions made and the properties enforced by the FS paradigm. The NP contains only the Fair Scheduler assumption. As this mechanism is of broad utility – we will show in section 3.1 that a Fair Scheduler has a great impact on TCP flows – it does not violate the end-to-end argument [12]. The issue related to the practical introduction of the paradigm are studied in section 3.

The FS paradigm, like the TCP-friendly paradigm, applies for both unicast and multicast since the paradigm does not make any assumption on the transmission mode. Moreover, the FS paradigm enforces properties of great benefits for multicast flows. For instance, the efficiency property leads, with the FS paradigm, to a tradeoff between the performance parameters bandwidth, delay, and loss. Whereas the FS paradigm guarantees that this tradeoff can be made end-to-end, it is not the purpose of this paper to address the end-to-end protocol design to achieve this tradeoff.

In conclusion, we have defined a simple paradigm for end-to-end congestion control, called FS paradigm, that relies on a Fair Scheduler network and only makes the assumption that the end users are selfish and non-collaborative. We can note that the FS paradigm is less restrictive than the TCP-friendly paradigm, as it does not make any assumptions on the mechanism used at the end users. Whereas the benefits of the FS paradigm with respect to flow isolation are commonly agreed on by the research community, its benefits for congestion control has been less clear since the congestion control properties are often not clearly defined. We showed that the FS paradigm allows to devise end-to-end congestion control protocols that meets almost all the properties of an ideal congestion control protocol. The remarkable point is that simply using Fair Schedulers allows to devise end-to-end congestion control protocols that are tailored to the application needs (due to the great flexibility when devising the congestion control protocol and due to the tradeoff among the performance parameters) while being a nearly ideal congestion control protocol.

We applied with success the FS paradigm to devise a new multicast congestion control protocol (see [23]). This protocol is based on cumulative layers and outperforms all the other protocols based on cumulative layers. In summary our protocol converges to the optimal link utilization in the order of one RTT and follows this optimal rate with no loss induced. Moreover, and as theoretically guaranteed by the FS paradigm, our protocol is fair with the TCP flows.

3 Practical Aspects of the FS Paradigm

In the previous sections we defined the FS paradigm. Now we investigate the practical issues that come with the introduction of such a paradigm in the Internet.

3.1 Behavior of TCP with the FS Paradigm

In this section, we evaluate the impact of the NP assumption of the FS paradigm on the today's Internet. A central question if we want to deploy the FS paradigm in the today's Internet is: As the NP assumption requires modifications in the network nodes, how will the use of a Fair Scheduler affect the TCP behavior and performance? Suter shows the benefits of a fair scheduler on TCP flows [24]. While his results are very promising, they are based on simulations for a very simple topology. We decided to explore the influence of the NP hypothesis on TCP with simulations on a large topology.

The generation of realistic network topologies is a subject of active research [25, 26, 27, 28]. It is commonly agreed that hierarchical topologies better represent a real Internetwork than do flat topologies. We use `tiers` ([26]) to create hierarchical topologies consisting of three levels: WAN, MAN, and LAN that aim to model the structure of the Internet topology [26] and call this Random Topology *RT*. For details about the network generation with `tiers` and the parameters used the reader is referred to Appendix A.

The Network Simulator ns [29] is commonly agreed to be the best simulator for the study of Internet protocols. We use ns with the topology generated by tiers. All the parameters of the topology are defined in Appendix A. The queue length is 50 packets for both FIFO and FQ scheduling (the shared buffer is 50 packets large). The buffer management used with FIFO scheduling is drop tail, and the buffer management used with FQ is longest queue drop with tail drop. The TCP flows are simulated using the ns implementation of TCP Reno, with packets of 1000 bytes size and a maximum window of 5000 packets (large enough not to bias the simulations). The TCP sources have always a packet to send. The unresponsive flows are simulated with UDP connections and CBR sources with a 10Mbit/s throughput.

We study two different scenarios:

TCP flows only. We add from $k = 50$ to $k = 1600$ TCP flows randomly distributed on the topology RT (i.e. the source and the receiver of a flow are randomly distributed among the LANs of RT). We do for each configuration of unicast flows an experiment with FIFO scheduling and an experiment with FQ scheduling. These experiments show the impact of the NP assumption on unicast flows. All the simulations are repeated five times and the average is taken over the five repetitions. All the plots are with 95% confidence intervals.

TCP and unresponsive flows. For this simulation we consider a unicast environment consisting of $k = 1000$ TCP flows randomly distributed on the topology RT. We add from $k_c = 1$ to $k_c = 150$ CBR flows randomly distributed on the topology RT. This simulation shows the impact of *fully unresponsive flows* (the CBR flows send at 10Mbit/s, the bandwidth of the LANs) with FIFO scheduling (as used in today's Internet), and with FQ scheduling (as suggested by the FS paradigm). All the simulations are repeated five times and the average is taken over the five repetitions. All the plots are with 95% confidence intervals.

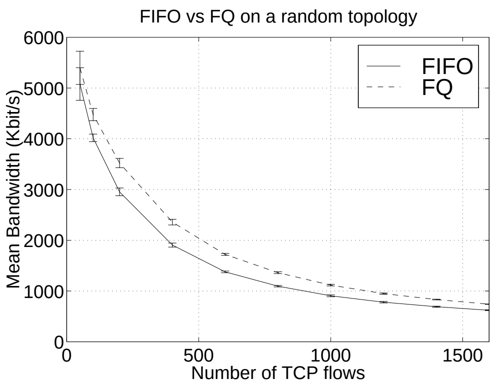
We choose a simulated time of 50 seconds, large enough to obtain significant results. All the TCP flows start randomly within the first simulated second. All the unresponsive flows start randomly between the forth and the fifth simulated second. We compute the mean bandwidth $\bar{F}_i$ for all TCP and all unresponsive flows i, $i = 1, ..., k + k_c$. In section 3.1.1, we do additional simulations with a simulated time of 200 seconds to study the behavior of a TCP flow throughout the time (see figure 3).

We consider three measures to evaluate the results: i) the mean bandwidth $\bar{B} = \frac{1}{k+k_c} \sum_{i=1}^{i=k+k_c} \bar{F}_i$. $\bar{B}$ shows the efficiency of the scheduling discipline in the sense of the satisfaction of the users if we consider a utility function that is linearly dependent of the bandwidth received for each receiver. Indeed for each experiment only the scheduling discipline changes. ii) the minimum bandwidth $\min_{i=1,...,k+k_c} \bar{F}_i$ shows the worst case performance for 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, etc (see [21]). So the minimum bandwidth shows which scheduling discipline leads to the bandwidth allocation closest to the max-min fair allocation. iii) the standard deviation $\sigma = \sqrt{\frac{1}{k+k_c-1} \sum_{i=1}^{i=k+k_c} (\bar{F}_i - \bar{B})^2}$ gives an indication about the uniformity of the bandwidth distribution among the users.

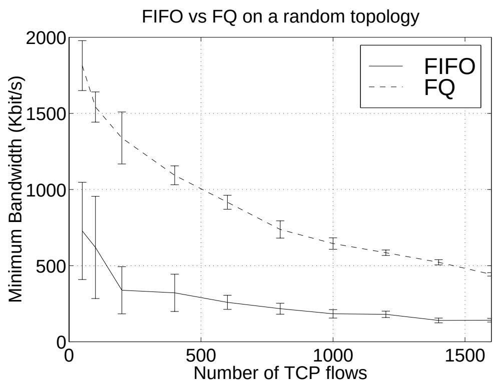
3.1.1 TCP Flows Only

Figure 2(a) shows the mean bandwidth for all the receivers as the number of TCP flows increases. The performance of the scenario with FQ scheduling surpasses the scenario with FIFO scheduling. For instance, the bandwidth $\bar{B}$ obtained with FQ is around 25% higher than the bandwidth achieved with FIFO for $k = 1000$. This is a very significant result: For the same topology and the same scenario, just changing the scheduling discipline results in a gain of 25% of bandwidth for TCP connection.

The reason of this result is due to the bursty nature of TCP traffic. FIFO scheduling can not smooth TCP traffic and preserves the bursty nature of TCP flows resulting in bursts of data packets and in Ack compression [30]. When a queue overflows due to a burst of data packets, every flow that shares this queue can experience losses. In our case a flow can be a flow of data packets but also a flow of Acks (the TCP back channel). Loss slows down each flow that experiences losses. In figure 3(a) we see a trans-WAN TCP flow in a scenario with FIFO scheduling. Our measurements indicate 270

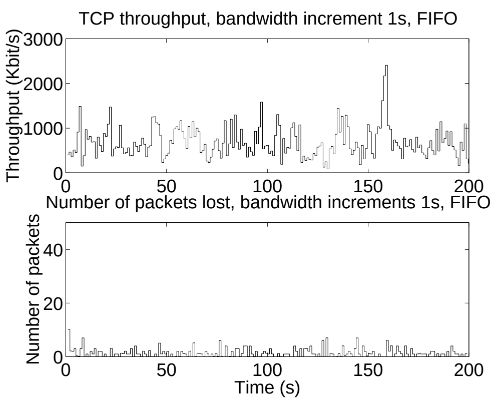


(a) mean bandwidth for TCP flows.

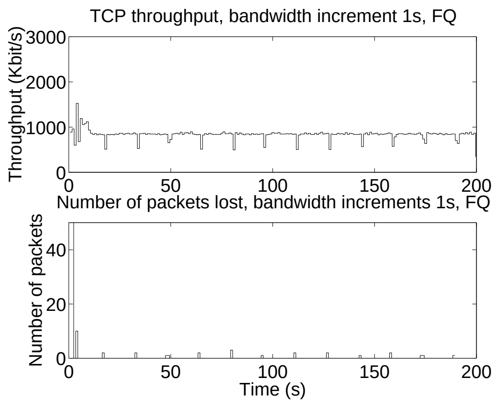


(b) minimum bandwidth for TCP flows.

Figure 2: FIFO versus FQ, increasing the number of unicast flows $k = 50, ..., 1600$.



(a) FIFO scheduling, bandwidth increment 1s.



(b) FQ scheduling, bandwidth increment 1s.

Figure 3: A trans-WAN TCP flow, throughput at the receiver and number of packets loss with FIFO scheduling and with FQ scheduling in the random topology RT.

lost data packets and 214 lost Acks for this flow. There is a total of 182222 packets lost for all the flows and 149984 Acks lost for all the flows during 200 simulated seconds.

FQ scheduling smoothes the TCP traffic, reduces the Ack compression, and reduces the bursts of data packets [24]. As the buffer management is longest queue drop, when a queue overflows, only the flows with the longest queue experience losses. This will result in flow isolation and in a very low probability to *loose Acks*. In figure 3(b) we see a trans-WAN TCP flow in a scenario with FQ scheduling. Our measurements show 92 lost data packets and 0 lost Acks for this flow. There is a total of 126298 packets lost for all the flows and 0 Acks lost for all the flows during 200 simulated seconds.

The comparison in figure 3 of the *same* connection in the *same* network but with a *different* scheduling discipline shows that FQ reduces the number of data packets lost and even prevents Ack loss. With FIFO scheduling the TCP flows experience frequent losses, which leads to a high instability in the throughput of the TCP flows. With the FQ scheduling, the TCP flow experiences sparse losses, the TCP throughput is clocked by regular slow start phases. The flow is isolated, the losses are only due to its own queue overflow.

One can note a corollary result in figure 3. There are many packets lost during the first simulated second with FQ

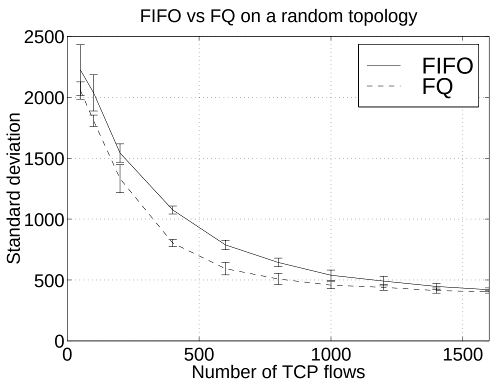


Figure 4: FIFO versus FQ, standard deviation of bandwidth for an increasing the number of unicast flows $k = 50, ..., 1600$.

but not with FIFO. This result seems at first glance favorable to FIFO scheduling whereas it is very unfavorable. Indeed during the first TCP cycle, the congestion windows only increase with the Slow Start phase until the flow experiences losses. With FQ scheduling and due to the flow isolation, every flow experiences a loss due to its own slow start phase. Since during the slow start phase, the window size increases exponentially, the number of packets lost can be, in the worst case, up to half of the window size. As every TCP flow starts during the first simulated second there is a high loss peak, however each flow converges fast to its fair share. With FIFO scheduling there is no flow isolation, so when the first flow causes a buffer overflow, all the flows that share the same bottleneck potentially experience losses. As these flows do not experience losses due to their own slow start phase, they experience a lower number of losses, which explains the low loss with FIFO scheduling. Whereas the flows experience more losses with FQ than with FIFO scheduling in the first simulated seconds, the flows converge faster and to a higher equilibrium bandwidth than with FIFO scheduling. We see from figure 3 that the absolute throughput with FQ compared to FIFO is already significantly higher since the first seconds of the experiment. Thus FQ benefits short TCP connections.

The bandwidth benefit and the reduction of the losses due to FQ are very interesting results. However we need to verify that these benefits are not at the expense of a decrease in fairness. The FQ scheduling increases significantly the minimum bandwidth compared to the FIFO scheduling (see figure 2(b)). For 1000 TCP flows, the minimum bandwidth with FIFO scheduling is around 185Kbit/s, while the minimum bandwidth with FQ scheduling is around 650Kbit/s. The minimum bandwidth with FQ scheduling is more than 250% higher than the minimum bandwidth with FIFO scheduling, for 1000 TCP flows. Thus the FQ scheduling leads to a bandwidth allocation closer to the max-min fair allocation than the FIFO scheduling.

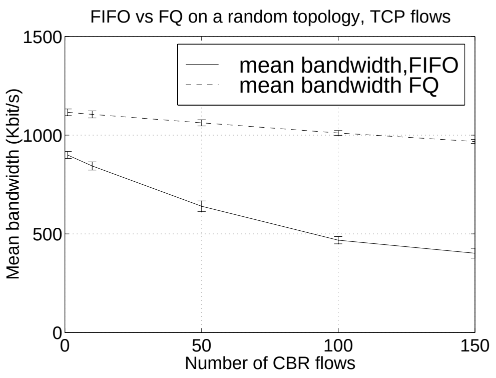
The FQ scheduling leads to a lower standard deviation of bandwidth σ than the FIFO scheduling (see figure 4) and therefore to a higher uniformity in the bandwidth allocation than FIFO scheduling.

In conclusion, whereas the NP assumption requires changes in the network, which is a hard task, our simulations show that already the increase in TCP performances justifies the NP assumption. We now study the question, how introducing the NP assumption in the Internet will affect TCP performance in the presence of flows that are *not* TCP-friendly.

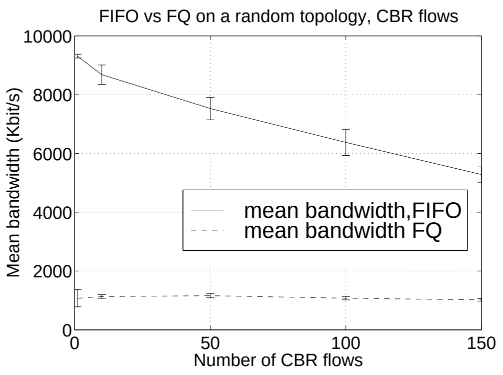
3.1.2 TCP and Unresponsive Flows

To have meaningful measurements we must choose enough TCP flows to create our TCP environment. Figure 4 shows little change in the standard deviation for more than $k = 1000$ flows . The bias due to the random locations of the users is negligible for $k \geq 1000$.

We therefore choose a TCP environment of $k = 1000$ flows and add unresponsive flows. This simulation aims to model the behavior of the Internet and to evaluate the robustness of our paradigm in the presence of worst case unresponsive flows, which send at the maximum LAN throughput of 10Mbit/s.

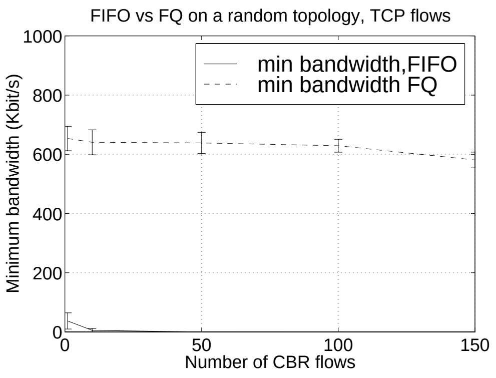


(a) mean bandwidth for TCP flows.

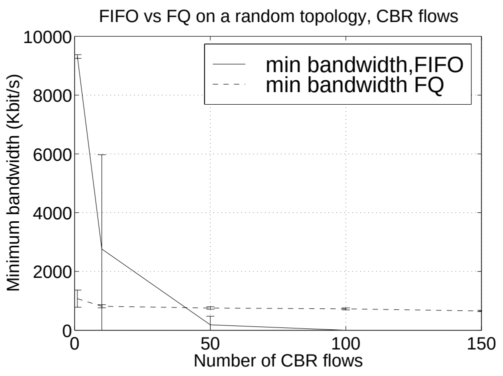


(b) mean bandwidth for CBR flows.

Figure 5: Increasing the number of CBR flows $k_c = 1, ..., 150$, $k = 1000$ unicast flows.



(a) minimum bandwidth for TCP flows.



(b) minimum bandwidth for CBR flows.

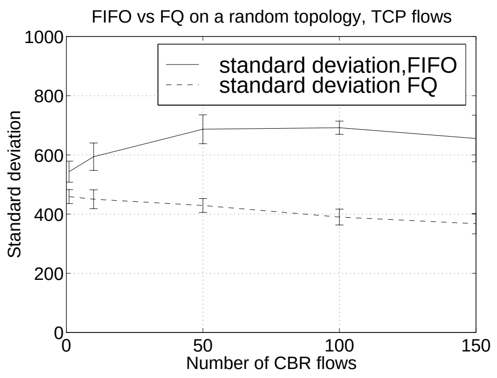
Figure 6: Increasing the number of CBR flows $k_c = 1, ..., 150$, $k = 1000$ unicast flows.

To evaluate the impact of the unresponsive flows we compare the curves for $k_c = 1, ..., 150$ with the curves of section 3.1.1 for $k = 1000, ..., 1150$.

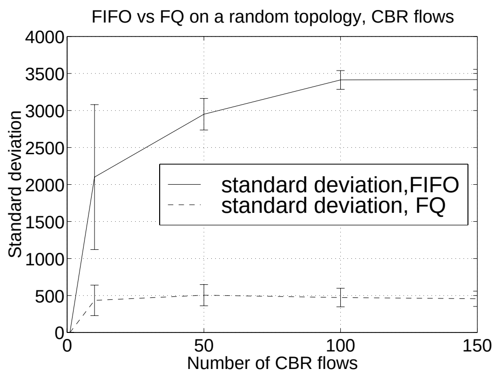
With FQ scheduling, the TCP flows do not suffer at all from the unresponsive CBR flows . The mean bandwidth decreases (with k_c) with the same slope (see figure 5(a)) than the mean bandwidth of the experiment with only TCP flows (see figure 2(a) for $k = 1000, ..., 1150$). We see the same behavior for the minimum bandwidth (compare figure 2(b) and figure 6(a)) and for the standard deviation (compare figure 7(a) and figure 4). The results for the unresponsive CBR flows with FQ scheduling are roughly the same than for the TCP flows. This is due to the equal share enforced by the fair scheduler. No flow can receive more than its equal share. However, as the CBR flows are unresponsive, the equal share is achieved at the expense of a high loss rate for the unresponsive CBR flows.

With FIFO scheduling, the TCP flows experience a significant drop in performance. The mean bandwidth heavily decreases when the number of unresponsive flows increases (compare figure 5(a) and figure 2(a) for $k = 1000, ..., 1150$). Figure 6(a) shows the disastrous effect of the unresponsive flows on the worst case receiver. The standard deviation increases with the number of unresponsive flows (compare figure 7(a) and figure 4).

The mean bandwidth for the unresponsive flows is higher with FIFO scheduling than with FQ scheduling (figure 5(b)) but the mean bandwidth of the unresponsive flows for FIFO scheduling decreases heavily with k_c. We would expect that



(a) standard deviation for TCP flows.



(b) standard deviation for CBR flows.

Figure 7: Increasing the number of CBR flows $k_c = 1, ..., 150$, $k = 1000$ unicast flows.

the unresponsive flows benefit from FIFO scheduling at the expense of TCP flows. While this is true for the mean bandwidth, we surprisingly observe the opposite for the minimum bandwidth and the standard deviation for the unresponsive flows (see figure 6(b) and figure 7(b)). Only one unresponsive CBR flow can get all the bandwidth, but if two or more unresponsive CBR flows cross, their highly aggressive behavior leads to a high loss rate among all the crossing unresponsive flows. Moreover, we can observe synchronization among the unresponsive CBR flows that leads to high unfairness, a flow can lose all the packets whereas another flow on the same path can experience no loss.

In conclusion, the FIFO scheduling leads to low bandwidth performance and unfairness for both TCP flows and unresponsive CBR flows. The NP assumption fully protects the TCP flows against unresponsive ones, moreover the high unfairness among unresponsive flows with FIFO scheduling disappears with the NP assumption. This result shows that the FS paradigm gives a great flexibility in devising the end-to-end protocols, since even highly unresponsive flows can coexist with TCP flows without penalty for the TCP flows.

In the next section we discuss the aspects of the incremental deployment of the FS paradigm.

3.2 Remarks on the Deployment of the New Paradigm

One practical question concerning the FS 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 Scheduler* capability in the routers. The deployment of the end-to-end protocols is not an issue due to the NP assumption, since the paradigm enforces no constraint on the end-to-end protocols, provided that the end users are selfish and non-collaborative. For a new application, one can easily develop an end-to-end protocol for this new application and distribute this protocol with this application. On the other hand, we can develop an end-to-end protocol for old applications and incrementally distribute these protocols with no fear, indeed the ones who use the new protocol will see a significant enhancement in the performance whereas the others, who do not upgrade yet, do not see a significant modification in their performance. So the FS paradigm allows an easy deployment of the end-to-end protocol. We note that this is not the case with the TCP-friendly paradigm, since it heavily relies on the collaboration of **all** the end users. If one wants in the case of a collaborative paradigm to add a new CC scheme, this CC scheme has to implement the same mechanism than the previous CC schemes. If one wants to change this mechanism, one has to change it in every end user, which is practically infeasible.

Second, the deployment of the NP requires that every router implements a Fair Scheduler. If we deploy an end-to-end protocol without the NP assumption, we can cause congestion collapse. Deploying the NP in the Internet seems unrealistic. However we have to take into account the administrative reality of the Internet. The Internet is an interconnection of ISPs. Each ISP has the full control of its network and offers specific services on its network, independent of the rest of the Internet. For instance, some ISPs start providing the multicast functionality inside their network whereas Internet, as a whole, is still not multicast capable[10]. ISPs are operating in a competitive environment that forces them to innovate and improve their service offered to keep the customers. In the past, ISPs have continuously upgraded the capacity of their links and installed, for instance, caches to improve their service. If an ISP has installed caches, his client will find with a probability P (as P ranges between 0.5 and 0.7 according to [31]) the web documents they access in the ISP's cache. Upgrading all the routers within an ISP with a *Fair Scheduler* will give a number of immediate benefits. Customers surfing on the web will have a higher TCP performance (around 25% higher see section 3.1.1) and therefore shorter download times (with a probability P) whenever a document is in the cache or on a server directly connected to the same ISP. 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.

In conclusion, the deployment of the new paradigm can be incremental. For an ISP, upgrading all its routers with Fair Schedulers is a substantial investment, but we believe that this investment will improve the quality of the service, which can be a significant commercial argument. So the ISPs have a financial interest in the deployment of this paradigm.

4 The FS Paradigm versus the TCP-friendly Paradigm

TCP has been for many years the main Internet congestion control protocol with an indisputable achievement. However, every new congestion control protocol deployed in the Internet must be TCP compatible i.e. this new protocol does not have to significantly decrease the performance of the TCP flows.

Both the TCP-friendly and the FS paradigm allow to devise end-to-end congestion control protocols compatible with TCP. A paradigm is only a formal way to define how to devise congestion control protocols. To compare two paradigms we have to look at the properties of the protocols devised with these paradigms. We compare the congestion control protocols according to the properties of an ideal congestion control protocol. The results are summarized in table 1 where a $+$ shows which paradigm outperforms the other one for a given property.

Properties	FS paradigm	TCP-friendly paradigm
Stability	$+$	$-$
Efficiency	$+$	$-$
Fairness	$+$	$-$
Robustness	$+$	$-$
Scalability	$+$	$+$
Feasibility	$-$	$+$

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

The TCP-friendly paradigm does not lead to ideal stability neither efficiency, due to the lack of assumption on the scheduling discipline (with selfish users only a Fair Scheduling can leads to ideal stability and in some case to ideal efficiency [14]). The FS paradigm does not lead to ideal efficiency in the general case either, however the FS 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, the fairness of this paradigm is biased by the RTT. The weakest point of the TCP-friendly paradigm is its lack of robustness: As this paradigm relies on the collaboration of the end users, it is easy to grab the bandwidth from the TCP-friendly flows. Both the TCP-friendly paradigm and the FS paradigm are scalable.

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

The weakest property of the FS paradigm is the feasibility. The TCP-friendly paradigm is the most feasible paradigm because it does not require any modification in the current Internet. The FS paradigm requires modification of the scheduling inside routers. We showed in section 3.2 that this deployment is feasible per ISP because ISPs have a financial interest in this deployment. The question is: there is a paradigm the TCP-friendly paradigm that is immediately feasible, but has many weakness (mainly the robustness and the efficiency for new applications, for instance multicast), on the other hand we propose a new paradigm, the FS paradigm, that outperforms the TCP-friendly paradigm but requires some effort to be deployed. We believe that the FS paradigm is an appealing solution and we hope this study will stimulate interest in this new paradigm.

5 Related Work

There is surprisingly little literature on congestion control paradigms. Most of the studies are about how to devise TCP-friendly end-to-end congestion control schemes. See [32] and [6] for unicast congestion control examples, and see [10], [33], [34], [35], and [36] for multicast congestion control examples.

Keshav [18] presents a comprehensive study of congestion control. While we agree with him in many points, our approach to the problem is fundamentally different. Keshav gives a new definition of congestion control, defines a set of properties for a congestion control scheme, and introduces two mechanisms, FQ and packet-pair flow control protocol that verify together his properties for the congestion control scheme. We agree with Keshav's definition of congestion and reuse it with a slight modification (see section 2.1). His congestion control properties influence the classification of our congestion control properties and his concept of congestion control with one part in the network (with FQ) and one part at the end host (with packet-pair) strongly influences our paradigm. However, Keshav's aim was to study the problems of congestion control and to present as a solution a new unicast congestion control scheme. Our aim is to define a model (*a new paradigm*) to devise end-to-end congestion control schemes. To achieve this goal, we define a set of properties for congestion control schemes. The definitions are abstract (they do not take into account any mechanism) and use a mathematical foundation. This formalism allows to prove the feasibility of the FS paradigm (see section 2.3) and to define a general background for the study of end-to-end congestion control.

Shenker applies game theory to study congestion control [14]. While his approach is rather different from ours, most of his results provide the mathematical foundation for the FS paradigm. He shows that one can achieve, with the selfish and non-collaborative behavior of the users, a congestion control that has a set of desired good properties. The only requirement is to have switching with a *fair share allocation function*. Shenker shows the benefits of the fair share policy for congestion control. However he does not clearly identify the properties of an ideal congestion control protocol and does not define the paradigm for devising congestion control protocols. Shenker's work is complementary to ours. We formally define the problem of congestion control, and propose a paradigm for congestion control. Shenker presents mathematical results that validate our work.

Lefelhocz *et al.* discuss a new paradigm for best effort congestion control [37]. They provide a good 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 adjustment. These mechanisms meet the FS paradigm: the scheduling and the buffer management are part of our NP; the feedback and the end adjustment are part of the end-to-end protocol. Our study shows how to use these mechanisms and why these mechanisms are sufficient. Moreover we 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 *prove* why our paradigm is an appealing alternative to the TCP-friendly paradigm. Our results can be seen as a generalization of their study.

Suter *et al.* show the benefits of FQ with appropriate buffer management for TCP flows. However, their study is based on very simple scenarios. We extend their study to a large topology and point out the most important issue with FQ and TCP flows: the suppression of the Ack loss and the significant reduction in data packets losses.

Another way to devise a new paradigm is the Diffserv or Intserv paradigm. There is active research on these topics, but to the best of our knowledge, there is no similar study to ours with these paradigms. Moreover the Diffserv and Intserv paradigms lead to much more complex mechanisms than the FS paradigm, for instance these paradigms are not viable without pricing (see [38]).

6 Conclusion

We defined a new paradigm, called FS paradigm, for end-to-end congestion control. This paradigm relies on a Fair Scheduler network and makes the assumption that the end users are selfish and non-collaborative. Whereas the FS paradigm is commonly agreed to have interesting properties, the research community has no clear understanding of what these properties precisely are. This lack of formalism leads to a mistrust toward this paradigm, which explains why end-to-end congestion control protocols have not been studied with the FS paradigm.

We start the paper with a definition of the notion of congestion that is a slight modification of Keshav's definition. We formally define a set of six properties for an ideal congestion control protocol. These properties are based on notions of game theory and microeconomics, thus allowing the use of the rigorous results previously established using these theories. The rigorous definition of the properties is important since this definition is highly reusable (we only make the assumption of selfishness for the definitions) and this definition allows to rigorously prove the validity of the FS paradigm. Then, we define the FS paradigm. We show that this new paradigm allows to devise congestion control protocols that have almost all the properties of an ideal congestion control protocol. What is remarkable is that the only assumption the paradigm makes on the end users is their selfish and non-collaborative behavior. Under this assumption, to devise nearly ideal congestion control protocols, we just need a network support that consists in a Fair Scheduler. To the best of our knowledge we are the first that define the properties of an ideal congestion control protocol and a paradigm for the design of end-to-end congestion control protocols with such a formalism, and prove the validity (in the sense of the properties of an ideal congestion control protocol) of this paradigm.

The second part of our study is about the practical aspects that come with the introduction of the FS paradigm in the Internet. Our simulations on a large topology show the great benefits due to the FS paradigm for TCP flows. The mean bandwidth of the TCP flows is increased by 25% and the minimum bandwidth is increased by 250% with the FS paradigm compared to a simulation of the current Internet. As indicated, the incremental deployment by a simple ISP will yield immediate benefits to this ISP's clients. In conclusion the FS paradigm applied in the today's Internet leads immediately to great benefits for the TCP flows and opens a new way in devising new very efficient unicast and multicast end-to-end congestion control protocols. This new paradigm offers an appealing alternative to the TCP-friendly paradigm.

Our study raises new questions. The FS paradigm allows to devise nearly ideal end-to-end congestion control. Which mechanisms should we use in the end-to-end protocol to reap full benefits of the paradigm? We address this question in another study for the context of multicast congestion control for audio and video applications (see [23]). We devise a new cumulative layered congestion control protocol that converges to the optimal rate in the order of one RTT with no loss induced in the discovery of the available bandwidth. We consider this new congestion control protocol as a practical proof of the great benefits of the FS paradigm.

Acknowledgment

The authors want to thank Jon Crowcroft, Dah Ming Chiu, and Brian Whetten for their useful comments. Eurecom's research is partially supported by its industrial partners: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

References

[1] J. Nagle, “Congestion control in tcp/ip internetworks,” *Computer Communication Review*, vol. 14, no. 4, pp. 11–17, October 1984.

[2] J. Nagle, “On packet switches with infinite storage,” *IEEE Transactions on Communications*, vol. COM-35, no. 4, pp. 435–438, April 1987, Also in [PART 88] pages 136-139.

[3] V. Jacobson, “Congestion avoidance and control,” in *Proc. ACM SIGCOMM 88*, pp. 314–329. Stanford, CA, August 1988, Also in [PART 88] pages 273-288.

[4] W. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery algorithms,” Request for Comments RFC 2001, Internet Engineering Task Force, January 1997.

[5] T. Ott, J. Kemperman, and M. Mathis, “The stationay distribution of ideal tcp congestion avoidance,” Tech. Rep., Bellcore, 1997.

[6] S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the internet,” Tech. Rep., LBNL, May 1999, To appear in IEEE/ACM Transactions on Networking, August 1999.

[7] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling tcp throughput: A simple model and its empirical validation,” in *Proccedings of ACM SIGCOMM’98*, Vancouver, Canada, 1998.

[8] J-C. Bolot, S. Fosse-Parisis, and D. Towsley, “Adaptive fec-based error control for internet telephony,” in *Proc. Infocom’99*, New York, March 1999.

[9] S. McCanne, V. Jacobson, and M. Vetterli, “Receiver-driven layered multicast,” in *SIGCOMM 96*, Aug. 1996, pp. 117–130.

[10] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft, “Tcp-like congestion control for layered multicast data transfer,” in *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, March 1998.

[11] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to end arguments in system design,” *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp. 277–288, November 1984.

[12] David P. Reed, Jerome H. Saltzer, and David D. Clark, “Commentaries on active networking and end to end arguments,” *IEEE Network*, vol. 12, no. 3, pp. 66–71, May/June 1998.

[13] A. K. Parekh and R. G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks,” in *Proc. IEEE INFOCOM’93*, 1993, pp. 521–530.

[14] Scott Shenker, “Making greed work in networks: A game-theoric analysis of switch service disciplines,” in *Proceeding of ACM SIGCOMM’94*, University College London, London, UK, October 1994, vol. 24, pp. 47–57, The SIGCOMM Quarterly Publication.

[15] D. Stiliadis and A. Varma, “A general methodology for designing efficient traffic scheduling and shaping algorithms,” in *Proceedings of IEEE INFOCOM ’97*, April 1997.

[16] A. Demers, S. Keshav, and S. Shenkar, “Analysis and simulation of a fair queueing algorithm,” in *Proc. SIGCOMM’89*, Austin, Texas, September 1989, pp. 1–12.

[17] Jon C.R. Bennett and H. Zhang, “Hierarchical packet fair queueing algorithms,” *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 675–689, October 1997.

[18] Srinivasan Keshav, *Congestion Control in Computer Networks*, Ph.D. thesis, EECS, University of Berkeley, CA 94720, USA, September 1991.

[19] Dimitri Bertsekas and Robert Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1992.

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

[21] Ellen L. Hahne, "Round-robin scheduling for max-min fairness in data networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 1024–1039, September 1991.

[22] Vijay P. Kumar, T. V. Lakshman, and D. Stiliadis, "Beyond best effort: Router architectures for the differentiated services of tomorrow's internet," *IEEE Communications Magazine*, vol. 36, no. 5, pp. 152–164, May 1998.

[23] Arnaud Legout and Ernst W. Biersack, "Fast convergence for cumulative layered schemes," Tech. Rep., Institut Eurecom, June 1999, Submitted to INFOCOM'2000.

[24] Bernhard Suter, T. V. Lakshman, Dimitrios Stiliadis, and Abhijit Choudhury, "Design considerations for supporting tcp with per-flow queueing," in *INFOCOM'98*, April 1998.

[25] Ken Calvert, Matt Doar, and Ellen W. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, June 1997.

[26] Matthew B. Doar, "A better model for generating test networks," in *Proceedings of IEEE Global Internet*, London, UK, November 1996, IEEE.

[27] Ellen W. Zegura, Ken Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Infocom '96*, March 1996.

[28] Ellen W. Zegura, Kenneth Calvert, and M. Jeff Donahoo, "A quantitative comparison of graph-based models for internet topology," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 770–783, December 1997.

[29] UCB/LBNL/VINT Network Simulator - ns (version 2), http://www-mash.cs.berkeley.edu/ns/.

[30] S. Floyd, "Connections with multiple congested gateways in packet-switched networks part 2:two-way traffic," Unpublished draft, December 1991.

[31] Pablo Rodriguez, Keith W. Ross, and Ernst 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.

[32] 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*, Oct. 1998, pp. 137–150.

[33] Thierry Turletti, Sacha Fosse-Parisis, and Jean Bolot, "Experiments with a layered transmission scheme over the internet," Research report, INRIA, B.P.93, Sophia-Antipolis Cedex, France, November 1997.

[34] Injong Rhee, Nallathambi Ballaguru, and George N. Rouskas, "Mtcp: Scalable tcp-like congestion control for reliable multicast," Tech. Rep. TR-98-01, North Carolina State University, North Carolina, January 1998.

[35] Huayan Amy Wang and Mischa Schwartz, "Achieving bounded fairness for multicast and tcp traffic in the internet," in *Proc. ACM SIGCOMM 98*, Vancouver, British Columbia, CANADA, October 1998, vol. 28, ACM SIGCOMM.

[36] Dante DeLucia and Katia Obraczka, "A multicast congestion control mechanism using representatives," Tech. Rep. 97-651, Computer Science Department - University of Southern California, May 1997.

[37] 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.

[38] Ron Cocchi, Scott Shenker, Deborah Estrin, and Lixia Zhang, "Pricing in computer networks: Motivation, formulation, and example," *IEEE/ACM Transactions on Networking*, vol. 1, no. 6, December 1993.

A Tiers Setup

We give a brief description of the topology used for all the simulations. The random topology *RT* is generated with `tiers` v1.1 using the command line parameters `tiers 1 20 9 5 2 1 3 1 1 1 1`. 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.

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 10Mbit/s link.