# Promoting the Use of End-to-End Congestion Control in the Internet

Sally Floyd and Kevin Fall*

To appear in IEEE/ACM Transactions on Networking

May 3, 1999

## Abstract

This paper considers the potentially negative impacts of an increasing deployment of non-congestion-controlled best-effort traffic on the Internet.[1] These negative impacts range from extreme unfairness against competing TCP traffic to the potential for congestion collapse. To promote the inclusion of end-to-end congestion control in the design of future protocols using best-effort traffic, we argue that *router mechanisms* are needed to identify and restrict the bandwidth of selected high-bandwidth best-effort flows in times of congestion. The paper discusses several general approaches for identifying those flows suitable for bandwidth regulation. These approaches are to identify a high-bandwidth flow in times of congestion as *unresponsive*, *"not TCP-friendly"*, or simply using disproportionate bandwidth. A flow that is not "TCP-friendly" is one whose long-term arrival rate exceeds that of any conformant TCP in the same circumstances. An unresponsive flow is one failing to reduce its offered load at a router in response to an increased packet drop rate, and a disproportionate-bandwidth flow is one that uses considerably more bandwidth than other flows in a time of congestion.

## 1 Introduction

The end-to-end congestion control mechanisms of TCP have been a critical factor in the robustness of the Internet. However, the Internet is no longer a small, closely knit user community, and it is no longer practical to rely on all end-nodes to use end-to-end congestion control for best-effort traffic. Similarly, it is no longer possible to rely on all developers to incorporate end-to-end congestion control in their Internet applications. The network itself must now participate in controlling its own resource utilization.

Assuming the Internet will continue to become congested due to a scarcity of bandwidth, this proposition leads to several possible approaches for controlling best-effort traffic. One approach involves the deployment of packet scheduling disciplines in routers that isolate each flow, as much as possible, from the effects of other flows [She94]. This approach suggests the deployment of *per-flow scheduling mechanisms* that separately regulate the bandwidth used by each best-effort flow, usually in an effort to approximate max-min fairness.

A second approach, outlined in this paper, is for routers to support the continued use of *end-to-end congestion control* as the primary mechanism for best-effort traffic to share scarce bandwidth, and to deploy *incentives* for its continued use. These incentives would be in the form of router mechanisms to restrict the bandwidth of best-effort flows using a disproportionate share of the bandwidth in times of congestion. These mechanisms would give a concrete incentive to end-users, application developers, and protocol designers to use end-to-end congestion control for best-effort traffic.

A third approach would be to rely on financial incentives or *pricing mechanisms* to control sharing. Relying exclusively on financial incentives would result in a risky gamble that network providers will be able to provision additional bandwidth and deploy effective pricing structures fast enough to keep up with the growth in unresponsive best-effort traffic in the Internet.

These three approaches to sharing: per-flow scheduling, incentives for end-to-end congestion control, and pricing mechanisms, are not necessarily mutually exclusive. Given the fundamental heterogeneity of the Internet, there is no requirement that all routers or all service providers follow precisely the same approach.

However, these three approaches can lead to different conclusions about the role of end-to-end congestion control for best-effort traffic, and different consequences in terms of the increasing deployment of such traffic in the Internet. The Internet is now at a cross-roads in terms of the use of end-to-end congestion control for best-effort traffic. It is in a position to actively welcome the widespread deployment of non-congestion-controlled best-effort traffic, to actively discourage such a widespread deployment, or, by taking no action, to allow such a widespread deployment to become a simple fact

*This work was supported by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098, and by ARPA grant DABT63-96-C-0105.

[1]This is a revised version of a technical report, "Router Mechanisms to Support End-to-End Congestion Control", from February 1997. This paper expands on Sections 2, 4 and 7 of that paper; other sections of that paper will be broken out into separate documents.

of life. We argue in this paper that recognizing the essential role of end-to-end congestion control for best-effort traffic and strengthening incentives for using it are critical issues as the Internet expands to an even larger community.

As we show in Section 2, an increasing deployment of traffic lacking end-to-end congestion control could lead to *congestion collapse* in the Internet. This form of congestion collapse would result from congested links sending packets that would only be dropped later in the network. The essential factor behind this form of congestion collapse is the absence of end-to-end feedback. Per-flow scheduling algorithms supply fairness with a cost of increased state, but provide no inherent incentive structure for best-effort flows to use strong end-to-end congestion control. We argue that routers need to deploy mechanisms that provide an incentive structure for flows to use end-to-end congestion control.

The potential problem of congestion collapse discussed in this paper only applies to best-effort traffic that does not have end-to-end bandwidth guarantees, or to a differentiated-services better-than-best-effort traffic class that also does not provide end-to-end bandwidth guarantees. We expect the network will also deploy "premium services" for flows with particular quality-of-service requirements, and that these premium services will require explicit admission control and preferential scheduling in the network. For such "premium" traffic, packets would only enter the network when the network is known to have the resources required to deliver the packets to their final destination. It seems likely (to us) that premium services with end-to-end bandwidth guarantees will apply only to a small fraction of future Internet traffic, and that the Internet will continue to be dominated by classes of best-effort traffic that use end-to-end congestion control.

Section 2 discusses the problems of extreme unfairness and potential congestion collapse that would result from increasing levels of best-effort traffic not using end-to-end congestion control. Next, Section 3 discusses general approaches for determining which high-bandwidth flows should be *regulated* by having their bandwidth use restricted at the router. The most conservative approach is to identify high-bandwidth flows that are not "TCP-friendly" (i.e., that are using more bandwidth than would any conformant TCP implementation in the same circumstances). A second approach is to identify high-bandwidth flows as "unresponsive" when their arrival rate at a router is not reduced in response to increased packet drops. The third approach is to identify disproportionate-bandwidth flows, that is, high-bandwidth flows that may be both responsive and TCP-friendly, but nevertheless are using excessive bandwidth in a time of high congestion.

As mentioned above, a different approach would be the use of per-flow scheduling mechanisms such as variants of round-robin or fair queueing to isolate all best-effort flows at routers. Most of these per-flow scheduling mechanisms prevent a best-effort flow from using a disproportionate amount of bandwidth in times of congestion, and therefore might seem to require no further mechanisms to identify and restrict the bandwidth of particular best-effort flows. Section 4 compares the approach of identifying unresponsive flows with alternate approaches such as per-flow scheduling or relying on pricing structures as incentives towards end-to-end congestion control. In addition, Section 4 discusses some of the advantages of aggregating best-effort traffic in queues using simple FCFS scheduling and active queue management along with the mechanisms described in this paper. Section 5 gives conclusions and discusses some of the open questions.

The simulations in this paper use the NS simulator, available at [NS95]. The scripts to run these simulations are available separately [FF98].

# 2 The problem of unresponsive flows

Unresponsive flows are flows that do not use end-to-end congestion control and, in particular, that do not reduce their load on the network when subjected to packet drops. This unresponsive behavior can result in both unfairness and congestion collapse for the Internet. The unfairness is from bandwidth starvation that unresponsive flows can inflict on well-behaved responsive traffic. The danger of congestion collapse stems from a network busy transmitting packets that will simply be discarded before reaching their final destinations. We discuss these two dangers separately below.

## 2.1 Problems of unfairness

A first problem caused by the absence of end-to-end congestion control is illustrated by the drastic unfairness that results from TCP flows competing with unresponsive UDP flows for scarce bandwidth. The TCP flows reduce their sending rates in response to congestion, leaving the uncooperative UDP flows to use the available bandwidth.

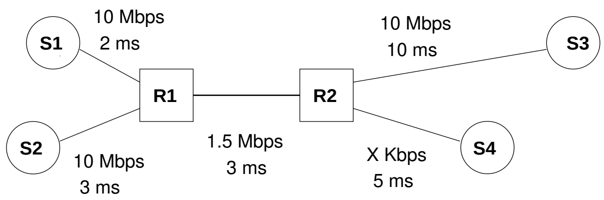


Figure 1: Simulation network.

Figure 2 graphically illustrates what happens when UDP and TCP flows compete for bandwidth, given routers with FCFS scheduling. The simulations use the scenario in Figure 1, with the bandwidth of the R2-S4 link set to 10 Mbps. The traffic consists of several TCP connections from node S1 to node S3, each with unlimited data to send, and a single constant-rate UDP flow from node S2 to S4. The routers have a single output queue for each attached link, and use FCFS

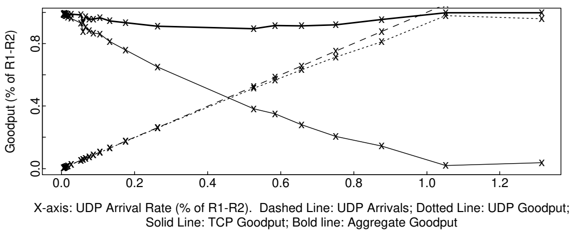


Figure 2: Simulations showing extreme unfairness with three TCP flows and one UDP flow, and FCFS scheduling.

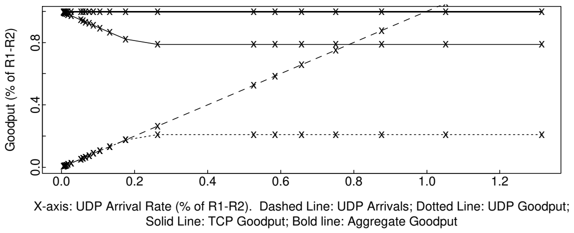


Figure 3: Simulations with three TCP flows and one UDP flow, with WRR scheduling. There is no unfairness.

scheduling. The sending rate for the UDP flow ranges up to 2 Mbps.

**Definition**: *goodput*. We define the "goodput" of a flow as the bandwidth delivered to the receiver, excluding duplicate packets.

Each simulation is represented in Figure 2 by three marks, one for the UDP arrival rate at router R1, another for UDP goodput, and a third for TCP goodput. The $x$-axis shows the UDP sending rate, as a fraction of the bandwidth on the R1-R2 link. The dashed line shows the UDP arrival rate at the router for the entire simulation set, the dotted line shows the UDP goodput, and the solid line shows the TCP goodput, all expressed as a fraction of the available bandwidth on the R1-R2 link. (Because there is no congestion on the first link, the UDP arrival rate at the first router is the same as the UDP sending rate.) The bold line (at the top of the graph) shows the aggregate goodput.

As Figure 2 shows, when the sending rate of the UDP flow is small, the TCP flows have high goodput, and use almost all of the bandwidth on the R1-R2 link. When the sending rate of the UDP flow is larger, the UDP flow receives a correspondingly large fraction of the bandwidth on the R1-R2 link, while the TCP flows back off in response to packet drops. This unfairness results from responsive and unresponsive flows competing for bandwidth under FCFS scheduling. The UDP flow effectively "shuts out" the responsive TCP traffic.

Even if all of the flows were using the exact same TCP congestion control mechanisms, with FCFS scheduling the bandwidth would not necessarily be distributed equally among those TCP flows with sufficient demand. [FJ92] discusses the relative distribution of bandwidth between two competing TCP connections with different roundtrip times. [Flo91] analyzes this difference, and goes on to discuss the relative distribution of bandwidth between two competing TCP connections on paths with different numbers of congested gateways. For example, [Flo91] shows how, as a result of TCP's congestion control algorithms, a connection's throughput varies as the inverse of the connection's roundtrip time. For paths with multiple congested gateways, [Flo91] further shows how a connection's throughput varies as the inverse of the square root of the number of congested gateways.

Figure 3 shows that per-flow scheduling mechanisms at the router can explicitly control the allocation of bandwidth among a set of competing flows. The simulations in Figure 3 use same scenario as in Figure 2, except that the FCFS scheduling has been replaced with weighted round-robin (WRR) scheduling, with each flow assigned an equal weight in units of bytes per second. As Figure 3 shows, with WRR scheduling the UDP flow is restricted to roughly 25% of the link bandwidth. The results would be similar with variants of Fair Queueing (FQ) scheduling.

## 2.2 The danger of congestion collapse

*This section discusses congestion collapse from undelivered packets, and shows how unresponsive flows could contribute to congestion collapse in the Internet.*

Informally, congestion collapse occurs when an increase in the network load results in a decrease in the useful work done by the network. Congestion collapse was first reported in the mid 1980s [Nag84], and was largely due to TCP connections unnecessarily retransmitting packets that were either in transit or had already been received at the receiver. We call the congestion collapse that results from the unnecessary retransmission of packets *classical congestion collapse*. Classical congestion collapse is a stable condition that can result in throughput that is a small fraction of normal [Nag84]. Problems with classical congestion collapse have generally been corrected by the timer improvements and congestion control mechanisms in modern implementations of TCP [Jac88].

A second form of potential congestion collapse, *congestion collapse from undelivered packets*, is the form of interest to us in this paper. Congestion collapse from undelivered packets arises when bandwidth is wasted by delivering packets through the network that are dropped before reaching their ultimate destination. We believe this is the largest unresolved danger with respect to congestion collapse in the Internet today. The danger of congestion collapse from undelivered packets is due primarily to the increasing deployment of open-loop applications not using end-to-end congestion control. Even more destructive would be best-effort applications that *increased* their sending rate in response to an increased packet drop rate (e.g., using an increased level of FEC).

We note that congestion collapse from undelivered packets and other forms of congestion collapse discussed in the following section differ from classical congestion collapse in that the degraded condition is not stable, but returns to normal once the load is reduced. This does not necessarily mean that the dangers are less severe. Different scenarios also can result in different *degrees* of congestion collapse, in terms of the fraction of the congested links' bandwidth used for productive work.

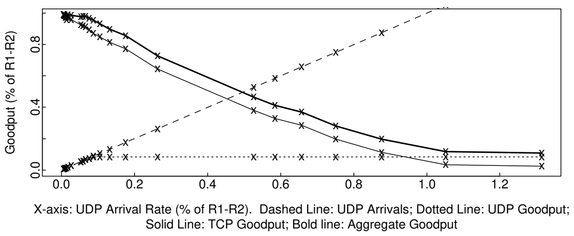


Figure 4: Simulations showing congestion collapse with three TCP flows and one UDP flow, with FCFS scheduling.

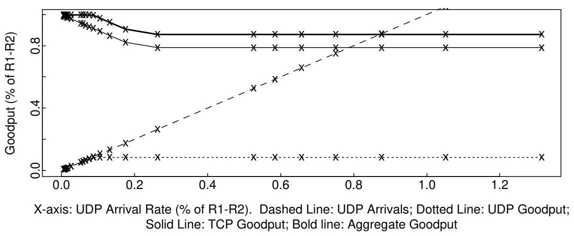


Figure 5: Simulations with three TCP flows and one UDP flow, with WRR scheduling. There is no congestion collapse.

Figure 4 illustrates congestion collapse from undelivered packets, where scarce bandwidth is wasted by packets that never reach their destination. The simulation in Figure 4 uses the scenario in Figure 1, with the bandwidth of the R2-S4 link set to 128 Kbps, 9% of the bandwidth of the R1-R2 link. Because the final link in the path for the UDP traffic (R2-S4) is of smaller bandwidth compared to the others, most of the UDP packets will be dropped at R2, at the output port to the R2-S4 link, when the UDP source rate exceeds 128 Kbps.

As illustrated in Figure 4, as the UDP source rate increases linearly, the TCP goodput *decreases* roughly linearly, and the UDP goodput is nearly constant. Thus, as the UDP flow increases its offered load, its only effect is to hurt the TCP (and aggregate) goodput. On the R1-R2 link, the UDP flow ultimately "wastes" the bandwidth that could have been used by the TCP flow, and reduces the goodput in the network as a whole down to a small fraction of the bandwidth of the R1-R2 link.

Figure 5 shows the same scenario as Figure 4, except the router uses WRR scheduling instead of FCFS scheduling. With the UDP flow restricted to 25% of the link bandwidth, there is a minimal reduction in the aggregate goodput. In this case, where a single flow is responsible for almost all of the wasted bandwidth at a link, per-flow scheduling mechanisms *are* reasonably successful at preventing congestion collapse as well as unfairness. However, per-flow scheduling mechanisms at the router can not be relied upon to eliminate this form of congestion collapse in all scenarios.

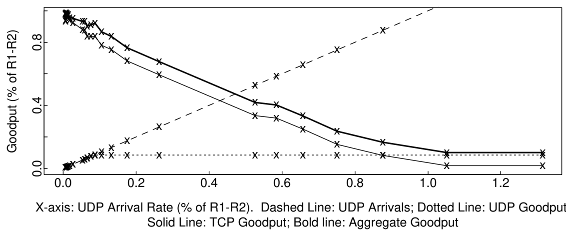


Figure 6: Simulations with one TCP flow and three UDP flows, showing congestion collapse with FIFO scheduling.

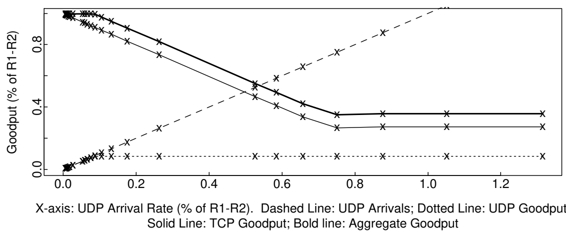


Figure 7: Simulations with one TCP flow and three UDP flows, showing congestion collapse with WRR scheduling.

In Figures 6 and 7, where a number of unresponsive flows are contributing to the congestion collapse, per-flow scheduling does not completely solve the problem. In these scenarios, a different traffic mix illustrates how some congestion collapse can occur for a network of routers using either FCFS or WRR scheduling. In these scenarios, there is one TCP connection from node S1 to node S3, and three constant-rate UDP connections from node S2 to S4. Figure 6 shows FCFS scheduling, and Figure 7 shows WRR scheduling. In Figure 6 (high load) the aggregate goodput of the R1-R2 link is only 10% of normal, and in Figure 7, the aggregate goodput of the R1-R2 link is 35% of normal.

Figure 8 shows that the limiting case of a very large number of very small bandwidth flows without congestion control could threaten congestion collapse in a highly-congested Internet regardless of the scheduling discipline at the router. For the simulations in Figure 8, there are ten flows, with the TCP flows all from node S1 to node S3, and the constant-rate UDP flows all from node S2 to S4. The $x$-axis shows the number of UDP flows in the simulation, ranging from 1 to 9. The $y$-axis shows the aggregate goodput, as a fraction of the bandwidth on the R1-R2 link, for two simulation sets: one with FCFS

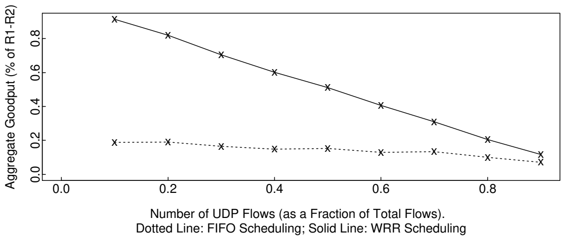


Figure 8: Congestion collapse as the number of UDP flows increases.

scheduling, and the other with WRR scheduling.

For the simulations with WRR scheduling, each flow is assigned an equal weight, and congestion collapse is created by increasing the *number* of UDP flows going to the R2-S4 link. For scheduling partitions based on source-destination pairs, congestion collapse would be created by increasing the number of UDP flows traversing the R1-R2 and R2-S4 links that had separate source-destination pairs.

*The essential factor behind this form of congestion collapse is not the scheduling algorithm at the router, or the bandwidth used by a single UDP flow, but the absence of end-to-end congestion control for the UDP traffic.* The congestion collapse would be essentially the same if the UDP traffic (somewhat stupidly) reserved and paid for more than 128 Kbps of bandwidth on the R1-R2 link in spite of the bandwidth limitations of the R2-S4 link. In a datagram network, end-to-end congestion control is needed to prevent flows from continuing to send when a large fraction of their packets are dropped in the network before reaching their destination. We note that congestion collapse from undelivered packets would not be an issue in a circuit-switched network where a sender is only allowed to send when there is an end-to-end path with the appropriate bandwidth.

## 2.3 Other forms of congestion collapse

In addition to *classical congestion collapse* and *congestion collapse from undelivered packets*, other potential forms of congestion collapse include *fragmentation-based congestion collapse*, *congestion collapse from increased control traffic*, and *congestion collapse from stale packets*. We discuss these other forms of congestion collapse briefly in this section.

*Fragmentation-based congestion collapse* [KM87, RF95] consists of the network transmitting fragments or cells of packets that will be discarded at the receiver because they cannot be reassembled into a valid packet. Fragmentation-based congestion collapse can result when some of the cells or fragments of a network-layer packet are discarded (e.g. at the link layer), while the rest are delivered to the receiver, thus wasting bandwidth on a congested path. The danger of fragmentation-based congestion collapse comes from a mismatch between link-level transmission units (e.g., cells or fragments) and higher-layer retransmission units (datagrams or packets), and can be prevented by mechanisms aimed at providing network-layer knowledge to the link-layer or vice-versa. One such mechanism is Early Packet Discard [RF95], which arranges that when an ATM switch drops cells, it will drop a complete frame's worth of cells. Another mechanism is Path MTU discovery [KMMP88], which helps to minimize packet fragmentation.

A variant of fragmentation-based congestion collapse concerns the network transmitting packets received correctly by the transport-level at the end node, but subsequently discarded by the end-node before they can be of use to the end user [Var96]. This can occur when web users abort partially-completed TCP transfers because of delays in the network and then re-request the same data. This form of fragmentation-based congestion collapse could result from a persistent high packet drop rate in the network, and could be ameliorated by mechanisms that allow end-nodes to save and re-use data from partially-completed transfers.

Another form of possible congestion collapse, *congestion collapse from increased control traffic*, has also been discussed in the research community. This would be congestion collapse where, as a result of increasing load and therefore increasing congestion, an increasingly-large fraction of the bytes transmitted on the congested links belong to control traffic (packet headers for small data packets, routing updates, multicast join and prune messages, session messages for reliable multicast sessions, DNS messages, etc.), and an increasingly-small fraction of the bytes transmitted correspond to data actually delivered to network applications.

A final form of congestion collapse, *congestion collapse from stale or unwanted packets*, could occur even in a scenario with infinite buffers and no packet drops. Congestion collapse from stale packets would occur if the congested links in the network were busy carrying packets that were no longer wanted by the user. This could happen, for example, if data transfers took sufficiently long, due to high delays waiting in large queues, that the users were no longer interested in the data when it finally arrived. Congestion collapse from unwanted packets could occur if, in a time of increasing load, an increasing fraction of the link bandwidth was being used by *push* web data that was never requested by the user.

## 2.4 Building in the right incentives

Given that the essential factor behind congestion collapse from undelivered packets is the absence of end-to-end congestion control, one question is how to build the right incentives into the network. What is needed is for the network architecture as a whole to include incentives for applications to use end-to-end congestion control.

In the current architecture, there are no concrete incentives for individual users to use end-to-end congestion control, and there are, in some cases, "rewards" for users that do *not* use it (i.e. they might receive a larger fraction of the link bandwidth than they would otherwise). Given a growing consensus among the Internet community that end-to-end congestion control is fundamental to the health of the Internet, there are some unquantifiable social incentives for protocol designers and software vendors not to release products for the Internet that do not use end-to-end congestion control. However, it is not sufficient to depend only on social incentives such as these.

Axelrod in "The Evolution of Cooperation" [Axe84] discusses some of the conditions required if cooperation is to be maintained in a system as a stable state. One way to view congestion control in the Internet is as TCP connections *cooperating* to share the scarce bandwidth in times of congestion. The benefits of this cooperation are that cooperating TCP connections can share bandwidth in a FIFO queue, using simple scheduling and accounting mechanisms, and can reap the benefits in that short bursts of packets from a connection can be transmitted in a burst. (FIFO queueing's tolerance of short bursts reduces the worst-case packet delay for packets that arrive at the router in a burst, compared to the worst-case delays from per-flow scheduling algorithms). This cooperative behavior in sharing scarce bandwidth is the foundation of TCP congestion control in the global Internet.

The inescapable price for this cooperation to remain stable is for mechanisms to be put in place so that users do not have an incentive to behave uncooperatively in the long term. Because users in the Internet do not have information about other users against whom they are competing for scarce bandwidth, the incentive mechanisms cannot come from the other users, but would have to come from the network infrastructure itself. This paper explores mechanisms that could be deployed in routers to provide a concrete incentive for users to participate in cooperative methods of congestion control. Alternative approaches such as per-flow scheduling mechanisms and reliance on pricing structures are discussed later in the paper.

Section 3 focuses on mechanisms for identifying which high-bandwidth flows are sufficiently unresponsive that their bandwidth should be regulated at routers. The main function of such mechanisms would be to reduce the incentive for flows to evade end-to-end congestion control. There are no mechanisms at a single router that are sufficient to obviate the need for end-to-end congestion control, or to prevent congestion collapse in an environment that is characterized by the evasion of end-to-end congestion control. There are only two ways to prevent congestion collapse from undelivered packets: to succeed, perhaps through incentives at routers, in maintaining an environment characterized by end-to-end congestion control; or to maintain a virtual-circuit-style environment where packets are prevented from entering the network unless the network has sufficient resources to deliver those packets to their final destination.

# 3 Identifying flows to regulate

In this section, we discuss the range of policies a router might use to identify which high-bandwidth flows to regulate. For a router with active queue management such as RED [FJ93], the arrival rates of high-bandwidth flows can be efficiently estimated from the recent packet drop history at the router [FF97]. Because the RED packet drop history constitutes a random sampling of the arriving packets, a flow with a significant fraction of the dropped packets is likely to have a correspondingly-significant fraction of the arriving packets. Thus, for higher-bandwidth flows, a flow's fraction of the dropped packets can be used to estimate that flow's fraction of the arriving packets. For the purposes of this discussion, we assume that routers already have some mechanism for efficiently estimating the arrival rate of high-bandwidth flows.

The router only needs to consider regulating those best-effort flows using significantly more than their "share" of the bandwidth in the presence of suppressed demand (as evidenced by packet drops) from other best-effort flows. A router can "regulate" a flow's bandwidth by differentially scheduling packets from that flow, or by preferentially dropping packets from that flow at the router [LM96]. When congestion is mild (as represented by a low packet drop rate), a router does not need to take any steps to identify high-bandwidth flows or further check if those flows need to be regulated.

The first two approaches in this section assume that a "flow" is defined on the granularity of source and destination IP addresses and port numbers, so each TCP connection is a single flow. The approach discussed in Section 3.3, of identifying flows that use a disproportionate share of the bandwidth in times of congestion, could also be used on aggregates of flows. This use of aggregation is most likely to be attractive for routers in the interior of the network with a high degree of statistical multiplexing, where each flow uses only a small fraction of the available bandwidth. For such a high-bandwidth backbone router, flow identification and packet classification on a fine-grained basis is not necessarily a viable approach.

The approaches discussed in this section are designed to detect a small number of misbehaving flows in an environment characterized by conformant end-to-end congestion control. They would not be effective as a substitute for end-to-end congestion control, and are only useful as an incentive to limit the benefits of evading end-to-end congestion control. The only effective substitute for end-to-end congestion control would be a virtual-circuit-style mechanism that prevented packets from being sent on the first link of a packet unless sufficient resources were guaranteed to be available for that packet along all hops of the end-to-end path.

Additional issues not addressed further in this paper are that practices such as encryption and packet fragmentation could make it more difficult for routers to classify packets into fine-grained flows. The practice of packet fragmentation should decrease with the use of MTU discovery [MD90]. The use of encryption in the IP Security Protocol (IPsec) [KA98] could prevent routers from using source IP addresses and port numbers for identifying some flows; for this traffic, routers could use the triple in the packet header that defines the Security Association to identify individual flows or aggregates of flows.

The policies outlined in this section for regulating high-bandwidth flows range in the degree of caution. One policy would be only to regulate high-bandwidth flows in times of congestion when they are known to be violating the expectations of end-to-end congestion control, by being either unresponsive to congestion (as described in Section 3.2) or exceeding the bandwidth used by any conformant TCP flow under the same circumstances (as described in Section 3.1). In this case, an unresponsive flow could either be restricted to the same bandwidth as a responsive flow (the more cautious approach), or could be given *less* bandwidth than a responsive flow (the less cautious but more powerful approach.) The second response would provide a concrete incentive for the use of end-to-end congestion control, but would also include the danger of incorrectly throttling flows that are in fact using conformant end-to-end congestion control.

Another policy would be to regulate any flows determined to be using a disproportionate share of the bandwidth in a time of congestion (as described in Section 3.3). Such flows might be unresponsive to congestion, or might simply be using conformant congestion control coupled with a significantly smaller roundtrip time or larger packet size than other competing flows. The most appropriate response to a flow identified as using a disproportionate share of the bandwidth is to use the more cautious approach of simply restricting that flow to the same bandwidth seen by other responsive flows. This response essentially constitutes a modified and limited form of per-flow scheduling that is only invoked for high-bandwidth flows in times of congestion.

The following sections discuss issues in detecting flows that are unresponsive, not TCP-friendly, or simply using disproportionate bandwidth in a time of congestion.

## 3.1 Identifying flows that are not TCP-friendly

**Definition**: *TCP-friendly flows*. We say a flow is *TCP-friendly* if its arrival rate does not exceed the arrival of a conformant TCP connection in the same circumstances. The test of whether or not a flow is TCP-friendly assumes TCP can be characterized by a congestion response of reducing its congestion window at least by half upon indications of congestion (i.e., windows containing packet drops), and of increasing its congestion window by a constant rate of at most one packet per roundtrip time otherwise. This response to congestion leads to a maximum overall sending rate for a TCP connection with a given packet loss rate, packet size, and roundtrip time. Given a packet drop rate of $p$, the maximum sending rate for a TCP connection is $T$ Bps, for

$$T \leq \frac{1.5\sqrt{2/3} * B}{R * \sqrt{p}}, \tag{1}$$

for a TCP connection sending packets of B bytes, with a fairly constant roundtrip time, including queueing delays, of R seconds. This equation is discussed in more detail in Appendix B. To apply this test, for each output link, a router should know the maximum packet size $B$ in bytes for packets on that link, and a minimum roundtrip time $R$ for any flows using that link.

The router can use its measurement of the aggregate packet drop rate for each link output queue over a recent time interval to estimate $p$, the packet drop rate experienced by a particular flow. Given the packet drop rate $p$, the minimum roundtrip time $R$, and the maximum packet size $B$, a router can use equation (1), or the improved form of the equation given in [PFTK98], to easily calculate the maximum arrival rate from a conformant TCP connection in similar circumstances. Actual TCP connections will generally use less than this maximum bandwidth, because they have limited demand, a longer roundtrip time, a window size limitation, a smaller packet size, a less-aggressive TCP implementation, a receiver that sends delayed ACKs, or additional packet drops from elsewhere in the network.

Given $R$ and $B$, equation (1) reduces to a simple table at the router: if the steady-state packet drop rate is "*x*", then the arrival rate of an individual flow should be at most "*y*". If a flow's drop rate (the ratio of a flow's dropped packets to its arriving packets) is lower than the aggregate drop rate for the queue, the router will overestimate the flow's actual drop rate, but at the same time will underestimate the flow's arrival rate in Bps. These effects tend to cancel, implying the estimates should not lead to problems with incorrect identification of unresponsive or unfriendly flows. This is confirmed by our simulations to date.

The test of TCP-friendliness does not attempt to verify that a flow responds to each and every packet drop exactly as would a conformant TCP flow. It does however assume a flow should not use more bandwidth than would the most aggressive conformant TCP implementation in the same circumstances. The TCP protocol itself is subject to change, and the congestion control mechanisms used to derive equation (1) could at some point be changed by the IETF (Internet Engineering Task Force), the responsible standards body. Nevertheless, the two limitations on TCP's window increase and decrease algorithms have been followed by all conformant TCP implementations since 1988 [Jac88], and have an installed base in the end-systems of the Internet that will persist for some time, even if at some point in the future changes might be proposed to the TCP standards to allow more aggressive responses to congestion. As long as best-effort traffic is dominated by such an installed base of TCP traffic, it would be reasonable for routers to restrict the bandwidth of any best-effort flow with an arrival rate higher than that of any conformant TCP implementation in the same circumstances.

The TCP-friendly test does not attempt to detect all flows which are not TCP-friendly. For example, the router might know a lower bound on any flow's roundtrip time, but the router does not know any flow's actual round-trip time. For routers with attached links with large propagation delays, the TCP-friendly test of equation (1) gives a useful tool for identifying flows which are not TCP-friendly. For routers with attached links of smaller propagation delay, the TCP-friendly test of equation (1) is less likely to identify any unfriendly flows. Such routers cannot exclude the possibility that a conformant TCP flow could receive a disproportionate share of the link bandwidth simply because it has a significantly smaller roundtrip time than competing TCP flows.

**Limitations of this Test**: The TCP-friendly test can only be applied to a flow at the level of granularity of a single TCP connection.

It can be difficult to determine the maximum packet size $B$ in bytes or a minimum roundtrip time $R$ for a flow. An individual flow whose arrival rate significantly exceeds the maximum TCP-friendly arrival rate is either not using TCP-friendly congestion control, or has larger packets or a smaller round-trip time than assumed by the router. Close to 100% of the packets in the Internet are 1500 bytes or smaller [TMW97]; routers could detect those high-bandwidth flows that use larger packets simply by observing the sizes of packets in the recent history of dropped packets. However, there is no simple test for a router to determine the end-to-end round-trip time of an active connection.

The minimum roundtrip time $R$ could be set to twice the one-way propagation delay of the attached link; this would limit the appropriateness of this test to those routers where the propagation delay of the attached link is likely to be a significant fraction of the end-to-end delay of a connection's path.

Care should be taken to only apply the TCP-friendly test to measurements taken over a sufficiently large time interval. The time period should not correspond to only one or two flow round-trip times. If a very long round-trip time flow is incorrectly identified as not TCP-friendly because of a short measurement interval relative to its roundtrip time, then the router will notice the flow's delayed response to congestion a short time later, and can respond accordingly (e.g. by removing bandwidth restrictions it may have applied, see below).

Another consideration in applying equation (1) is the prevalence of packet drops from buffer overflow. Equation (1) only applies for non-bursty packet drop behavior, where a flow receives at most one packet drop per window of data, and therefore each packet drop corresponds to a separate indication of congestion to the end nodes. In particular, when congestion is high, and there is significant buffer overflow, multiple packets dropped from a window of data are likely to be fairly common.

**Response by the Router**: Our proposal is that routers should freely restrict the bandwidth of best-effort flows determined not to be TCP-friendly in times of congestion. Such flows are "stealing" bandwidth from TCP-friendly traffic and, more seriously, are contributing to the danger of congestion collapse. Any such flow should only have its bandwidth restriction removed when there is no longer any significant link congestion, or when it has been shown to reduce its arrival rate appropriately in response to congestion.

**Example Test**: a *TCP-friendly test*. One possibility for a TCP-friendly test that we explored in simulations would be to identify a high-bandwidth best-effort flow as not TCP-friendly if its estimated arrival rate is greater than $1.45B/(R\sqrt{p})$, for B the maximum packet size in bytes, $R$ twice the propagation delay of the attached link, and $p$ the aggregate packet drop rate for that queue. A flow's restriction would be removed if its arrival rate returns to less than $1.22B/(R\sqrt{p})$, for the new packet drop rate $p$.

## 3.2 Identifying unresponsive flows

The TCP-friendly test is based on the specific congestion control responses of TCP, and many routers may not want to use such a "TCP-centric" measure. The TCP-friendly test is also of limited usefulness for routers unable to assume strong bounds on TCP packet sizes and round-trip times. A more general test would be simply to verify that a high-bandwidth flow was *responsive* (i.e. its arrival rate decreases appropriately in response to an increased packet drop rate).

Equation (1) shows that for a TCP flow with persistent demand, if the long-term packet drop rate of the connection increases by a factor of $x$, then the arrival rate from the source should decrease by a factor of roughly $\sqrt{x}$. For example, if the long term packet drop rate increases by a factor of four, than the arrival rate should decrease by a factor of two. This suggests a test for identifying unresponsive flows if the drop rate is changing. If the steady state drop rate increases by a factor $x$, and the presented load for a high-bandwidth flow does not decrease by a factor reasonably close to $\sqrt{x}$ or more, then the flow can be deemed not to be using congestion control (unresponsive). Similarly, if the steady state drop rate increases by a factor $x$, and the presented load for aggregated traffic does not decrease by a factor reasonably close to $\sqrt{x}$ or more, then either the mix of the aggregated traffic has changed, or the traffic as an aggregate is not using congestion control, and can be categorized as unresponsive.

Applying this test to a flow requires estimates of a flow's arrival rate and packet drop rate over several long time intervals. The flow's arrival rate could be estimated from the history of packet drops maintained by active queue management, and the flow's packet drop rate could be estimated using the aggregate packet drop rate at the queue.

This test does not attempt to detect all flows that are not responding to congestion, but is only applied to the high bandwidth flows. When the packet drop rate remains relatively constant, no flows will be identified as unresponsive. In addition, the router has limited information about the flow's responses to congestion. The primary congestion indications experienced by a flow might be coming from elsewhere in the network. In addition, the arrival rate seen by a router is a result not only of the sending rate, but also of the drop rate experienced by a flow at a congested link earlier on its path.

An additional refinement of this "responsiveness" test would be to distinguish three separate subcases: flows with an increasing or relatively constant average arrival rate (as indicated by the drop metric) in the face of an increasing packet drop rate at the router; a flow whose average arrival rate generally tracks longer-term changes in the packet drop rate at the router; and a flow whose average arrival rate seems to change independently of changes in the router's packet drop rate.

**Limitations of this Test**: As discussed in the previous section, care should be taken when applying this test. In particular, a test for unresponsiveness is less straightforward for a flow with a variable demand. In addition to possible end-to-end congestion mechanisms such as senders adjusting their coding rates or receivers subscribing and unsubscribing from layered multicast groups, the original data source itself could be ON/OFF or otherwise have strong rate variations over time. If a high-bandwidth flow is restricted because it has been identified as unresponsive, and it is later determined to be responding to congestion by reducing its arrival rate, then the restriction is removed.

If the only tests deployed along a path were tests for responsiveness, this could give flows an incentive to start with an overly-high initial bandwidth. Such a flow could then reduce its sending rate in response to congestion, and still receive a larger share of the bandwidth than competing flows.

**Response by the Router**: The router should freely restrict the bandwidth of best-effort flows determined to be unresponsive in times of congestion. Such flows are "stealing" bandwidth from responsive TCP-friendly traffic, and, more importantly, increasing the danger of congestion collapse.

Instead of applying the test passively by observing how the flow's arrival rate changes in response to changes in the packet drop rate, another possibility would be to apply the test actively. This could be done by purposefully increasing the packet drop rate of a high bandwidth flow in times of congestion, and observing whether the arrival rate of the flow on that link decreases appropriately.

**Example Test**: a *test for unresponsiveness*. One possibility for an unresponsiveness test is to identify a high-bandwidth best-effort flow as unresponsive if the packet drop rate increases by more than a factor of four, but the flow's arrival rate has not decreased to below 90% of its previous value. Restrictions would be removed from an unresponsive flow only if, after an increased packet drop rate, its arrival rate returns to at most half of its arrival rate when it was restricted.

## 3.3 Identifying flows using disproportionate bandwidth

A third test would be simply to identify flows that use a *disproportionate share of the bandwidth* in times of high congestion, where a disproportionate share is defined as a significantly larger share than other flows in the presence of suppressed demand from some of the other flows. A router might restrict the bandwidth of such flows even if the flows are known to be using conformant TCP congestion control. A conformant TCP flow could use a "disproportionate share" of bandwidth under several circumstances: if it was the only TCP with sustained persistent demand, or the only TCP using large windows, or the only TCP with a significantly smaller roundtrip time or larger packet sizes than other active TCPs.

Let $n$ be the number of flows with packet drops in the recent reporting interval. The most obvious test to check if a flow was using a disproportionate share of the bandwidth in times of congestion would be to test if the flow's fraction of the aggregate arrival rate was greater than some small constant times $1/n$, when the aggregate packet drop rate was greater than some preconfigured threshold deemed as an unacceptable level of congestion. Our test is a modification of this approach that, instead of using a preconfigured threshold for the acceptable packet drop rate, simply allows for greater skewedness in the distribution of best-effort bandwidth when packet drop rates are lower. The goal is only to prevent flows from using a highly disproportionate share of the bandwidth when there is likely to be "sufficient" demand from other best-effort flows.

The first component of the disproportionate-bandwidth test is to check if a flow is using a disproportionate share of the bandwidth. We define a flow as using a *disproportionate share* of the best-effort bandwidth if its fraction of the aggregate arrival rate is more than $\log(3n)/n$, for $\log$ the natural logarithm. We chose this fraction because it is close to one (i.e., 0.9) for $n$ equal to two, and grows slowly as a multiple of $1/n$.

The second component of our test takes into account the level of congestion itself, as reflected in the aggregate packet drop rate $p$. We define a flow as having a high arrival rate *relative to the level of congestion* if its arrival rate is greater than $c/\sqrt{p}$ Bps for some constant $c$. This definition is motivated by our characterization in the appendix of the relationship between the arrival rate and the packet drop rate for conformant TCP. For our simulations we set $c$ to 12,000, which is close to $1.5\sqrt{2/3}B/R$ for $B = 512$ bytes and $R = 0.05$ seconds.

**Limitations of this Test**: Gauging the level of unsatisfied demand is problematic. For a large round-trip time TCP flow with persistent demand, a single packet drop can represent a significant suppressed demand. For a short bursty web transfer, a single packet drop might not mean much in terms of unsatisfied demand.

**Response by the Router**: A conservative approach would be to limit the restriction of a high-bandwidth responsive flow so that over the long run, each such flow receives as much bandwidth as the highest-bandwidth unrestricted flow. In restricting the bandwidth of a high-bandwidth flow that has not been identified as either unresponsive or not TCP-friendly, care should be taken not to "punish" it by restricting its bandwidth too severely.

**Example test**: a *disproportionate-bandwidth test*. Let $p$ be the aggregate packet drop rate for the unrestricted best-effort traffic, and let $n$ be the number of flows with packet drops in the most recent interval. One possibility for a disproportionate-bandwidth test would be to identify a best-effort flow as using disproportionate-bandwidth if the estimated arrival rate is greater than $12,000/\sqrt{p}$ and the arrival rate is also greater than a fraction $\log(3n)/n$ of the best-effort bandwidth. The restriction would be removed when one of these conditions is no longer true.

# 4 Alternate approaches

An alternative to the use of the router mechanisms proposed in this paper would be the ubiquitous deployment, at all congested routers in the Internet, of per-flow scheduling mechanisms such as round-robin or fair queueing scheduling. In general, per-flow scheduling algorithms separately schedule packets from each flow, dividing the available bandwidth among the various flows and providing isolation between them. Per-flow scheduling mechanisms at routers would indeed take care of many of the fairness issues concerning competing best-effort flows. With per-flow scheduling, it might also seem that there is no need for further mechanisms to identify and restrict the bandwidth of best-effort flows that do not use appropriate end-to-end congestion control. In this section we argue that (1) even routers with per-flow scheduling mechanisms still need additional mechanisms as an incentive for best-effort flows to use end-to-end congestion control; and (2) FCFS scheduling has some advantages for best-effort traffic that are apart from issues of implementation efficiency or incentives regarding end-to-end congestion control.

As we have seen in Section 2, per-flow scheduling cannot, by itself, prevent congestion collapse from undelivered packets. To what extent would the use of per-flow scheduling mechanisms encourage end-to-end congestion control for best-effort traffic? Recommendations for the ubiquitous deployment of per-flow scheduling for best-effort traffic are based on an assumption that in a heterogeneous world, best-effort flows cannot be relied upon to be responsive to congestion, and therefore they should be isolated from each other. In some sense, per-flow scheduling has incentives in the wrong direction, encouraging flows to make sure that "their" queue in the congested router never goes empty (so that they never lose "their" turn at scheduling).

An advantage of simple FCFS scheduling over per-flow scheduling is that FCFS scheduling is more efficient to implement. Implementation efficiency can be a concern as link speeds and the number of active flows per link both increase. Apart from considerations of implementation efficiency, however, FCFS scheduling is in many ways the optimal scheduling algorithm for a class of traffic where the long-term aggregate arrival rate is restricted by either admission controls or, in the case of best-effort traffic, by compatible end-to-end congestion control procedures. In comparison to Fair Queueing [DKS90] or Round Robin scheduling, FCFS scheduling reduces the tail of the delay distribution [CSZ92]. In particular, FCFS scheduling allows packets arriving in a small burst to be transmitted in a burst, rather than having the packets "spread out" and be delayed by the scheduler.

In some sense, FCFS scheduling and per-flow Fair Queueing or Round Robin scheduling are two ends of a spectrum. The middle ranges of the spectrum would include not only FCFS scheduling, enhanced by mechanisms for the differential treatment of unresponsive flows, but could also include relaxed variants of per-flow scheduling that allow for small bursts to be transmitted by each flow and include additional incentives for end-to-end congestion control. This middle range would also include FCFS scheduling with differential dropping for flows using a disproportionate share of the bandwidth [LM96], or scheduling mechanisms such as Class-Based Queueing (CBQ) [FJ95] or Stochastic Fair Queueing (SFQ) [McK90] that can operate on levels of granularity between the two extremes of either a single flow or the entire aggregate of best-effort traffic.

The differential treatment of unresponsive flows can consist of preferentially dropping packets from unresponsive flows while keeping those packets in the same queue, or of reclassifying packets from unresponsive flows to a separate queue or queues. Another choice concerns the granularity at which regulation should be applied. The approaches outlined in Sections 3.1 and 3.2 of identifying unfriendly or unresponsive flows can best be applied to the level of granularity of a single flow; the responsiveness of an aggregate of flows is quite different from the responsiveness of a single flow. In contrast, the approach outlined in Section 3.3 of identifying flows using disproportionate bandwidth could also be applied to aggregates of flows. As with any scheduling or packet dropping mechanism applied to an aggregate, there is a fundamental question of the relative allocation of scarce network resources to the various aggregates. This issue remains problematic even at the level of granularity of single flows: an application can open $N$ separate flows to the same destination instead of one, for example,[2] or frequently change port numbers for active flows.

A more speculative issue is whether min-max fairness is the ideal fairness metric to use for best-effort traffic at a specific router. Min-max fairness has the advantage of being simple to define at a router; indeed, it is the basis for our approach in this paper for defining flows using a disproportionate share of the

[2] This particular form of evasion of end-to-end congestion control would be reduced by the development of mechanisms for shared congestion control among flows with the same source and destination [Flo99].

link bandwidth. However, instead of considering the network as a whole, the *min-max* definition of fairness restricts attention separately to each isolated component. A more appropriate fairness metric for recognizing each flow's equal access to the scarce resources of the Internet would take into account such global factors as the number of congested links on each flow's path.

Another alternative to the router mechanisms described in this paper might be the deployment of *pricing structures* sensitive to the behavior of each flow in the global Internet that would elicit the desired behavior. Although pricing structures could be envisioned that provide an incentive for applications to use end-to-end congestion control, the state required by such a pricing scheme would be nontrivial.

In contrast, router mechanisms that detect and restrict the bandwidth of uncooperative flows could be deployed incrementally, without requiring global knowledge or global consistency in the network infrastructure, to provide a concrete incentive to flows to use appropriate congestion control mechanisms. Such mechanisms could be deployed at a congested router, using information from packet drops (or other congestion indications) generated at the router itself.

In a network engineered so that the typical case is one of sufficient bandwidth for the demand, distinctions between the various scheduling algorithms and incentive mechanisms would become less important. Similarly, in such a network the possibility of congestion collapse due to congested links carrying packets that would later be dropped in the network would become more remote. It is hard to predict, however, when or if the scenario of sufficient bandwidth for the demand is likely to be achieved.

# 5 Conclusions and future work

We have argued in this paper on the need for end-to-end congestion control, and further, on the need for mechanisms in the network to detect and restrict unresponsive or high-bandwidth best-effort flows in times of congestion. Such mechanisms would provide a incentive in support of end-to-end congestion control for best-effort traffic.

Clearly there is more work still to be done in developing and investigating the approaches outlined in this paper. We have not yet outlined a specific proposal for mechanisms for identifying and controlling unresponsive flows. We believe the most important issue is not the precise functioning of the mechanisms to restrict the bandwidth of unresponsive best-effort flows, but simply that such mechanisms be deployed. Mechanisms such as these would go a long way to making concrete the essential role played by congestion control for best-effort traffic in the Internet.

# 6 Acknowledgments

This paper results in part from a long collaboration with Van Jacobson. It also results from a long history of discussions and disagreements in the IETF Transport Area Directorate, the Internet End-to-End Research Group, and elsewhere. We are particularly indebted to Hari Balakrishnan, Greg Minshall, Lixia Zhang, and the anonymous reviewers from SIGCOMM 97 and from IEEE/ACM Transactions on Networking for feedback on this paper, to Kinh Tieu who worked with us on the related issue of using RED packet drops to detect high-bandwidth flows, and to Jean Bolot, Bob Braden, Jamshid Mahdavi, Matt Mathis, and Scott Shenker for discussions of some of these matters.

# References

[Axe84] R. Axelrod. *The Evolution of Cooperation*. Harper-Collins, 1984.

[BPS+98] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, S. Stemm, and R.H. Katz. "TCP Behavior of a Busy Internet Server: Analysis and Improvements". *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Mar. 1998. URL http://www.cs.berkeley.edu/ hari/papers/infocom98.ps.gz.

[CSZ92] D.D. Clark, S. Shenker, and L. Zhang. "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism". *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 14–26, 1992.

[DKS90] A. Demers, S. Keshav, and S. Shenker. "Analysis and Simulation of a Fair Queueing Algorithm". *Internetworking: Research and Experience*, 1:3–26, 1990.

[FF96] K. Fall and S. Floyd. "Simulation-based Comparisons of Tahoe, Reno, and Sack TCP". *ACM Computer Communication Review*, Jul. 1996. URL http://www-nrg.ee.lbl.gov/nrg-papers.html.

[FF97] S. Floyd and K. Fall. "Router Mechanisms to Support End-to-End Congestion Control". Unpublished manuscript, URL http://www-nrg.ee.lbl.gov/floyd/papers.html, Feb. 1997.

[FF98] S. Floyd and K. Fall. "Promoting the Use of End-to-End Congestion Control in the Internet". Submitted to IEEE/ACM Transactions on Networking, URL http://www-nrg.ee.lbl.gov/floyd/end2end-paper.html, Feb. 1998.

[FJ92] S. Floyd and V. Jacobson. "On Traffic Phase Effects in Packet-Switched Gateways". *Internetworking: Research and Experience*, 3(3):115–156, Sep. 1992.

[FJ93] S. Floyd and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance". *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug. 1993. URL http://www-nrg.ee.lbl.gov/nrg-papers.html.

[FJ95] S. Floyd and V. Jacobson. "Link-sharing and Resource Management Models for Packet Networks". *IEEE/ACM Transactions on Networking*, 3(4), 1995. URL http://www-nrg.ee.lbl.gov/nrg-papers.html/.

[Flo91] S. Floyd. "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic". *ACM Computer Communication Review*, 21(5):30–47, Oct. 1991. URL http://www-nrg.ee.lbl.gov/nrg-papers.html/.

[Flo94] S. Floyd. "TCP and Explicit Congestion Notification". *ACM Computer Communication Review*, 24(5):10–23, Oct. 1994.

[Flo99] S. Floyd. "Multiplexing, TCP, and UDP: Pointers to the Discussion", 1999. Web Page, URL http://www.aciri.org/floyd/tcp_mux.html.

[Jac88] V. Jacobson. "Congestion Avoidance and Control". *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 314–329, 1988. An updated version is available via ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[KA98] S. Kent and R. Atkinson. "Security Architecture for the Internet Protocol". Request for Comments (Proposed Standard) RFC 2401, Internet Engineering Task Force, November 1998.

[KM87] C. Kent and J. Mogul. "Fragmentation Considered Harmful". *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 390–401, Aug. 1987.

[KMMP88] C. Kent, K. McCloghrie, J. Mogul, and C. Partridge. "IP MTU Discovery options". Request for Comments RFC 1063, Internet Engineering Task Force, July 1988.

[LM96] D. Lin and R. Morris. "Dynamics of Random Early Detection". *SIGCOMM Symposium on Communications Architectures and Protocols*, 1996.

[LM97] T. V. Lakshman and U. Madhow. "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss". *IEEE/ACM Transactions on Networking*, pages 336–350, Jun. 1997. URL http://www.ccrc.wustl.edu/ ton/jun97.html#Lakshman.

[McK90] P. McKenney. "Stochastic Fairness Queueing". *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, 1990.

[MD90] J. Mogul and S. Deering. "Path MTU Discovery". Request for Comments (Draft Standard) RFC 1191, Internet Engineering Task Force, November 1990.

[MF97] J. Mahdavi and S. Floyd. "TCP-Friendly Unicast Rate-Based Flow Control". Technical note sent to the end2end-interest mailing list, URL http://www.psc.edu/networking/papers/tcp_friendly.html, Jan. 1997.

[MSMO97] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm". *ACM Computer Communication Review*, 27(3):67–82, Jul. 1997.

[Nag84] J. Nagle. "Congestion control in IP/TCP internetworks". Request for Comments RFC 896, Internet Engineering Task Force, January 1984.

[NS95] "NS (Network Simulator)", 1995. URL http://www-mash.cs.berkeley.edu/ns/.

[OKM96] T. Ott, J. Kemperman, and M. Mathis. "The Stationary Distribution of Ideal TCP Congestion Avoidance", Aug. 1996. URL http://networks.ecse.rpi.edu/ natun/papers/tcp-equn.ps.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. "Modeling TCP Throughput: A Simple Model and its Empirical Validation". *SIGCOMM Symposium on Communications Architectures and Protocols*, Aug. 1998. URL http://www.acm.org/sigcomm/sigcomm98/tp/abs_25.html.

[RF95] A. Romanow and S. Floyd. "Dynamics of TCP Traffic over ATM Networks". *IEEE Journal on Selected Areas in Communications*, 13(4), 1995. URL http://www-nrg.ee.lbl.gov/nrg-papers.html.

[RF99] K. K. Ramakrishnan and Sally Floyd. "A Proposal to add Explicit Congestion Notification (ECN) to IP". Jan. 1999. RFC 2481.

[She94] S. Shenker. "Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines". *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 47–57, Aug. 1994.

[TMW97] K. Thompson, G. Miller, and R. Wilder. "Wide-Area Internet Traffic Patterns and Characteristics". *IEEE Network*, 11(6):10–23, Nov. 1997.

[Var96] G. Varghese. "On Avoiding Congestion Collapse". viewgraphs, Washington University Workshop on the Integration of IP and ATM, Nov. 19 1996.

# A One TCP connection or many?

This section discusses the negative impact on the network of breaking a single TCP connection into multiple connections at the application level to increase throughput. In particular, we show that while the use of concurrent connections increases throughput for those applications that break a TCP connection into multiple connections (relative to those applications that do not do this), it also increases the packet drop rate shared by all of the best-effort traffic (see also [BPS+98]). Breaking a single TCP connection into multiple connections is one example of a possible spiral of increasingly-aggressive TCP congestion control behaviors that leads to increasing packet drop rates in the Internet.

For a TCP connection that has been separated into $N$ different TCP subconnections, a single packet drop results in one of the $N$ subconnections, receiving $1/N$-th of the aggregate bandwidth, having its throughput cut in half. Thus, a single packet drop causes the aggregate arrival rate to be dropped to a fraction $(2N-1)/(2N)$ of its previous value. Then, because each TCP subconnection continues to increase its congestion window by one packet per RTT for those TCP subconnections that have not yet reached the receiver's advertised window, the aggregate TCP connections together increase their arrival rate by up to $N$ packets per RTT. This is much more aggressive congestion control that would lead to a correspondingly-larger steady-state packet drop rate in the Internet. A router could detect a TCP connection that had been separated into $N$ different TCP subconnections by defining the granularity of a "flow" by source and destination IP addresses only.

# B Characterizing TCP-friendly flows

Since congestion control was introduced to TCP in 1988 [Jac88], TCP flows in the Internet have used packet drops as an indication of congestion, and have responded by reducing their offered load by half for each window of data experiencing a packet drop. For a responsive flow with persistent demand, increasing the packet drop rate for a flow at a router should, after a short delay, result in a decreased arrival rate from that flow at that router. In this section we give an upper bound on the arrival rate from any single conformant TCP connection at a router, given a steady-state packet drop rate at the router, an upper bound on the TCP packet size, and a lower bound on the TCP connection's roundtrip time. Using this characterization, routers can characterize selected flows as using more bandwidth than would any TCP flow in the same circumstances.

In this section we explore the relationship between throughput and the packet drop rate for a *conformant* TCP connection [Flo91, OKM96, MF97, MSMO97]. By a *conformant* TCP connection, we mean a TCP connection where the TCP sender follows the following two essential components of today's TCP congestion control. First, the TCP data sender interprets any packet drop in a window of data as an indication of congestion, and responds by reducing the congestion window at least in half. Second, during the congestion avoidance phase in the absence of congestion, the TCP sender increases the congestion window by at most one packet per roundtrip time (or more precisely, by at most one packet per window of data). These two components lead to a simple relationship between the "steady-state" packet drop rate received by a TCP connection, and the "steady-state" average throughput achieved by that connection.

There are many reasons why conformant TCP implementations might respond to congestion less aggressively than allowed by the limits of congestion control described above. TCP implementations have potentially-long delays due to retransmit timeouts; at times, TCP senders invoke slow-start in responding to congestion; TCP connections may be limited by maximum bounds on the window size, imposed by buffering or lack of window scaling at either at the sender or receiver; for TCP connections where the receiver only sends an ACK packet for every two data packets, the TCP sender increases the congestion window by less than one packet per roundtrip time.

We assume a *steady-state* model of TCP as introduced in Section 5 of [Flo91]. For the purposes of heuristic analysis, we assume a single packet is dropped from a TCP connection each time the congestion window is increased to $W$ packets (and never when the congestion window is below $W$ packets). The *steady-state* model assumes a non-zero but non-bursty average packet drop rate of $p$, where an individual TCP connection has at most one packet drop in a window of data. The TCP sender responds to a packet drop by cutting the congestion window at least in half. After a packet is dropped, the TCP sender increases its congestion window by at most one packet each roundtrip time, until the congestion window again reaches its old value of $W$ packets (and, in steady state, the TCP connection receives another packet drop). The assumption in this model of a deterministic and repeatable pattern, although admittedly unrealistic, leads to results verified by simulations in this section and by independently derived and more rigorous analysis in [OKM96, LM97, PFTK98]. The equation that results from this steady-state model has also been proposed as a basis for new congestion-control mechanisms [MF97].

We consider a TCP connection sending packets (or more precisely, segments) of $B$ bytes, with a fairly constant roundtrip time, including queueing delays, of $R$ seconds. Each time a packet is dropped, the TCP sender has a congestion window of $W$ packets.

By decreasing its window by at least half for each packet drop and increasing its window by at most one per round-trip time afterwards, the TCP sender transmits at least

$$\frac{W}{2} + \left(\frac{W}{2}+1\right) + ... + W \approx \frac{3}{8}W^2. \tag{2}$$

packets for each packet dropped. The fraction $p$ of the sender's packets that are dropped is then bounded by the reciprocal of that value:

$$p \leq \frac{8}{3W^2}. \tag{3}$$

From equation (3),

$$W \leq \sqrt{\frac{8}{3p}}. \tag{4}$$

For our steady-state model assuming a link with steady-state packet drop rate $p$, equation (4) gives the maximum congestion window $W$ of a TCP connection when a packet is dropped. With a steady-state packet drop rate of $p$ in the steady-state model, the TCP connection sends $\frac{3}{8}W^2$ packets between packet drops. Because the congestion window is decreased by at least half, and increased by at most one packet per roundtrip time, there are at least $W/2$ roundtrip times between packet drops in the steady-state model. The maximum sending rate for a TCP connection over a single cycle of the steady-state model is thus $T$ Bps, for

$$T \leq \frac{0.75 * W * B}{R}.$$

Substituting for $W$ from equation (4), we get

$$T \leq \frac{1.5\sqrt{2/3} * B}{R * \sqrt{p}}. \tag{5}$$

This upper bound on TCP's average sending rate applies for any conformant TCP that decreases its congestion window by at least half, and, after the congestion window has been decreased by half, increases the congestion window by at most one packet per roundtrip time.[3] Thus, this upper bound also applies to a TCP restricted by the receiver's advertised window, or by TCP variants such as Vegas TCP which sometimes refrain from increasing the congestion window during the congestion avoidance phase. Assuming a steady-state packet drop rate of $p$, and thus, in the steady-state model, that the TCP connection gets to send $1/p$ packets between packet drops, clearly the TCP connection maximizes its average throughput by increasing its congestion window by the maximum allowed amount each roundtrip time.

This might at first seem counter-intuitive. However, the purpose of the steady-state model in this section is to explore the relationship between the steady-state packet drop rate and the steady-state arrival rate from the TCP connection. Certainly in a specific scenario with all else being equal, a TCP that refrains from increasing its congestion window from time to time might increase its own throughput by decreasing the aggregate packet drop rate. This does not change the fact that the inequality in equation (1) still describes the relationship between the packet drop rate and the arrival rate for that connection.

[3] The same result was derived by [OKM96], using a more rigorous model, with a constant of 1.3 instead of 1.22 ($\approx 1.5\sqrt{2/3}$).

For TCP connections where the data receiver sends at most one ACK for every two packets, we could show a stronger upper bound on the sending rate. For a TCP connection with a delayed-ACK receiver, the sender receives one acknowledgment for every two packets, and increases its window more slowly that a TCP connection that receives an ACK for every packet. With a delayed-ACK receiver, the fraction of that connection's arriving packets that are dropped is

$$p = \frac{1}{\sum_{i=0}^{W}(W/2 + i/2)} \approx \frac{1}{(3/4)W^2}. \tag{6}$$

This gives an upper bound on the arrival rate of

$$T \leq \frac{1.5\sqrt{1/3} * B}{R * \sqrt{p}}. \tag{7}$$

Although the language in this paper refers only to packet drops, proposals have been made to add explicit congestion notification to TCP/IP [Flo94, RF99]. If explicit congestion notification were deployed, then instead of dropping a packet to provide feedback about congestion, a router could simply "mark" packets by setting the the Explicit Congestion Notification bit in packet headers.

**Limitations of the Model:** Equations (5) and (7) do not take into account TCP delays due to waiting for retransmit timers to time out. Thus, equation (5) drastically overestimates the bandwidth for steady-state scenarios when the congestion window $W$ is less than four packets when a packet is dropped. From equation (4), this occurs when the packet drop rate is $16\%$ or higher. (If the congestion window is four or higher, the TCP connection can recover from a single packet drop using Fast Retransmit, after receiving several duplicate acknowledgments. If the congestion window is smaller, then the TCP connection generally has to wait for a retransmit timeout. [FF96]) In the extreme case, for a packet drop rate of $100\%$, our steady-state model would assume that the TCP connection stubbornly sends one packet every roundtrip time, and equation (5) (because it used an approximation in equation (2)) gives a TCP sending rate of slightly over one packet per roundtrip time. Incorporating the notion of retransmit timer backoff in the model, as in [PFTK98], gives a much more realistic result.

## B.1 Simulations verifying the "TCP-friendly" characterization

In this section we use simulations to loosely verify the "TCP-friendly" characterization in equation (5). This equation has also been verified with simulations and experiments in [MSMO97].

Figure 9 illustrates the simulation topology used to evaluate the "TCP-friendly" characterization. The solid line in the top graph of Figure 10 shows the TCP-friendly bandwidth from equation (5) as a function of the packet drop rate. This is also shown in the straight line in the bottom graph. The curved

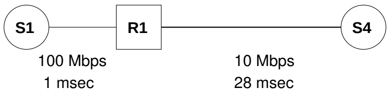


Figure 9: Simulation network.

solid line in the bottom graph shows the revised equation from [PFTK98]. Figure 10 assumes a TCP connection with minimum roundtrip time of $R = 0.06$ seconds and a maximum packet size of $B = 1460$ bytes. The $x$-axis shows $p$, the fraction of arriving packets that are dropped, and the $y$-axis shows $T$, the upper bound on TCP arrival rate in KBps. The bottom graph repeats the top graph on a log-log scale.

Each dashed line in Figure 10 shows the results from a single simulation set. Each simulation consists of two competing connections, one TCP and the other UDP, from node S1 to node S4. For each simulation set the sending rate of the UDP flow ranges from zero up to the available bandwidth of the congested link. The router uses FCFS scheduling and RED queue management. The RED packet drop mechanisms are generally able to prevent both the FIFO buffer from overflowing and RED's average queue size from exceeding its maximum threshold. The TCP connection sees a roundtrip time, including queueing delay, of roughly 60 ms.

Each simulation is represented by a number in Figure 10. The simulations in a simulation set differ from each other only in the sending rate of the UDP flow. Numbers "1" through "3" show simulations where the TCP connection uses 1460-byte packets. Numbers "4" through "6" show simulations with 512-byte packets. Simulation sets "2" and "5" use Tahoe TCP, and the others use SACK TCP. Simulation sets "3" and "6" use data receivers with delayed ACKs (sending one ACK to acknowledge two data packets), and the others use single ACKS (sending an ACK for every data packet). For all of the simulations, the TCP clock granularity is 100 ms. The $x$-axis in Figure 10 shows the fraction of the TCP connection's arriving packets that are dropped, and the $y$-axis shows the TCP connection's sending rate.

For the SACK and Tahoe simulations with 1460-byte packets and single-ACK receivers (simulation sets "1" and "2"), the simulation results are a reasonable match to the computed TCP-friendly bandwidth. For drop rates lower than 2%, the SACK and Tahoe TCPs receive more than the computed TCP-friendly bandwidth. Examining the output traces shows that in these simulations, it is not uncommon for two packets to be dropped from a single window of data in a congestion epoch. When this happens, the two packet drops constitute a single indication of congestion to the end nodes.

For packet drop rates greater than 5%, Figure 10 shows that the TCP-friendly bandwidth greatly overestimates the arrival rate of a TCP connection. As mentioned earlier, this is because

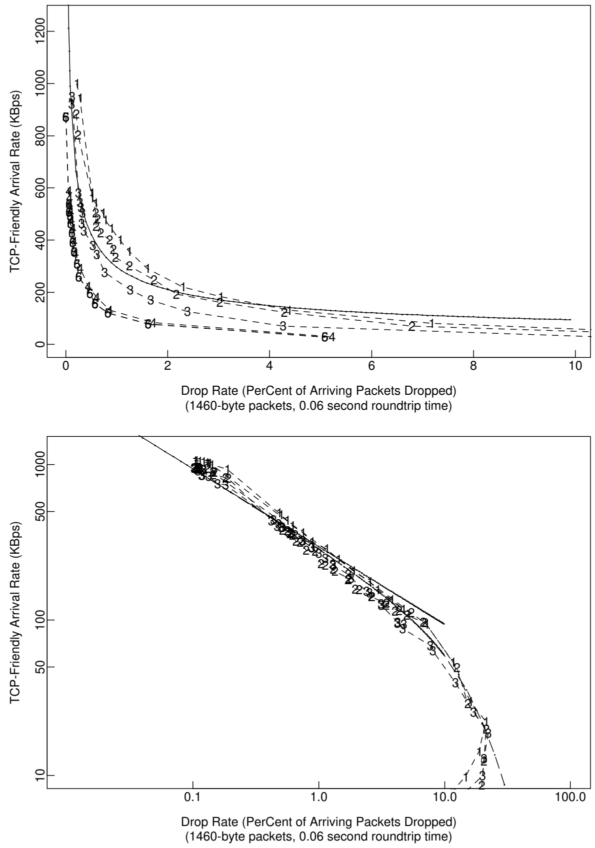


Figure 10: TCP-friendly bandwidth for a 60-ms roundtrip time and 1460-byte packets.

the version of the steady-state model used in this paper does not take into account delays due to retransmit timers.

Simulations with 512-byte packets closely match equation (5) using 512-byte packets. As seen in Figure 10, the more aggressive the TCP congestion control (i.e. a TCP with 1460-byte packets is more aggressive than TCP with 512-byte packets), the higher the steady-state packet drop rate needed to sustain the same per-connection bandwidth. A spiral of increasingly-aggressive congestion control would lead to a matching spiral of an increasingly-high steady-state packet drop rate, in the context of a fixed available bandwidth.

Figure 11 shows the results for SACK TCP with a delayed-ACK receiver with the simulated topology of figure 9. For a fixed throughput, a TCP connection with a delayed-ACK receiver should receive half the packet drop rate of a TCP connection that receives an ACK for every packet. The top solid line shows the analytical results for an immediate-ACK receiver, and the bottom solid line shows the analytical results for an delayed-ACK receiver. For a given packet drop rate, a TCP connection with a delayed-ACK receiver will receive less throughput than a TCP connection with an immediate-ACK receiver.

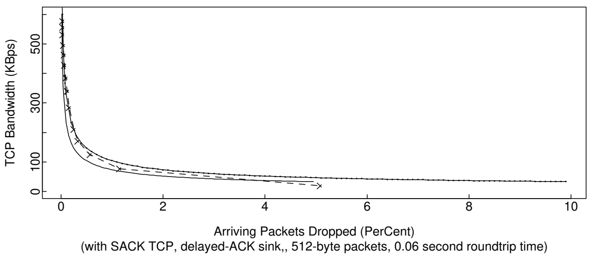


Figure 11: TCP bandwidth vs. steady-state drop rate, for SACK TCP with a delayed-ACK receiver, a 60-ms roundtrip time and 512-byte packets.