# ERUF: Early Regulation of Unresponsive Best-Effort Traffic

Anand Rangarajan, Anurag Acharya
Dept. of Computer Science
University of California, Santa Barbara

Technical Report TRCS99-07
Dept of Computer Science
University of California, Santa Barbara

February 1999

**Abstract**

In this paper, we propose router mechanisms to regulate unresponsive best-effort traffic. By unresponsive traffic, we mean flows that do not reduce their sending rate in response to congestion. The goal of the proposed mechanisms is to drop undeliverable packets as close to the periphery of the network as possible. The key ideas of our approach are: (1) edge routers keep track of incoming flows and their arrival rates; (2) core routers use RED for queue management and generate rate-limited source quenches on packet drops to advice sources to reduce their sending rates; and (3) edge routers snoop on source quenches passing through them and use them to control per-flow regulators. Regulators adjust their maximum sending rate using a multiplicative-decrease, additive-increase discipline. A decrease is triggered by the arrival of a source quench; an increase is triggered by *non-arrival* of source quenches for a time period. We examine the impact of these mechanisms for a variety of simulated network topologies and traffic patterns.

## 1 Introduction

The stability of the current Internet architecture depends to a large extent on the end-to-end congestion control mechanisms provided by TCP. This stability, however, is fragile as it depends on cooperation of end-hosts. Given the rapid expansion of the Internet and the commercial pressures that come with it, it is no longer possible to rely on voluntary cooperation of the end-hosts. A growing number of providers of streaming-content use UDP-based delivery with little or no congestion control. For example, RealAudio, one of the major streaming applications, uses one of several fixed data rates depending on the bandwidth available; the Real Broadcast Network uses *Robust UDP* [15] (i.e., UDP with retransmissions but no congestion control).

Since the benefits of an uncongested (or under-congested) network are available only if all (or most) end-hosts cooperate in responding to congestion, growth in unresponsive traffic will provide an incentive for even more end-hosts to eliminate congestion control. To ensure the continued stability of the Internet, it is, therefore, important for the network to regulate unresponsive traffic. Recognizing this need, the authors of RFC 2309 [4] strongly urge *"research, engineering and measurement efforts contributing to the design of mechanisms to deal with flows that are unresponsive to congestion notification or are responsive but more aggressive than TCP."* Floyd&Fall [12] believe that growth of undeliverable packets generated by open-loop applications is the largest unresolved danger with respect to congestion collapse in the Internet today.

In this paper, we propose and examine router mechanisms to regulate unresponsive best-effort traffic. By unresponsive traffic, we mean flows that do not reduce their sending rate on congestion notification (or worse still, increase their sending rate[1]). The goal of the proposed approach is to drop undeliverable packets (i.e., packets that will be dropped somewhere in the network) as as close to the periphery of the network as possible. Dropping undeliverable packets early frees network resources for other packets that can make better use of them.

There are three key aspects of our approach, which we refer to as *Early Regulation of Unresponsive Flows (ERUF)*. First, edge routers keep track of incoming flows and their arrival rates. Second, core routers use the Random Early Drop (RED) algorithm [13] for queue management and generate source quenches on packet drop to advice sources to reduce their sending rates. We assume that the generation of source quenches is suitably rate-limited.[2] Finally, edge routers snoop on source quenches passing through them and use them to control flow-specific token-bucket-based regulators [34]. These regulators adjust their maximum sending rate, referred to as the *regulator bandwidth*, using a multiplicative-decrease/additive-increase discipline. A decrease is triggered by the arrival of one or more source quenches within a single round-trip time (RTT) estimate;[3] increases are triggered by *non-arrival* of source quenches over two or more RTT estimates.[4] Since UDP-based streaming protocols usually don't acknowledge individual packets, there is no direct way to estimate round-trip time between the edge router regulating a flow and a congested router trying to quench the flow that does not involve injecting additional packets into the network. We evaluate two easy-to-compute approximations in this paper: (1) *nexthop*, which is two times the propagation delay on the next link towards the congested router; and (2) $allhops = numhops \times nexthop$, where *numhops* is the number of hops between the edge router and the congested router.

Previous research into handling unresponsive flows has taken one of two approaches. The *allocation* approach isolates individual flows by performing a fair allocation of the available bandwidth between competing flows [2, 6, 14, 23, 29, 30, 31]. Most allocation-based schemes require *all* routers to maintain state and perform all operations on a per-flow basis. Recently, Stoica et al [31] proposed CSFQ, a *Core*-State-less allocation scheme, which maintains per-flow state only in edge routers. CSFQ makes a good tradeoff between implementation complexity and perfect fairness. The design of ERUF resembles that of CSFQ in that both of them try to move per-flow operations to the periphery of the network. However, CSFQ makes its decisions locally at each router and ignores congestion that may be occurring downstream. Therefore, it doesn't try to drop undeliverable packets as early as possible.

In the *identification* approach, routers use a lightweight detection algorithm to identify unresponsive flows and then explicitly manage the bandwidth of these flows [11, 12]. Floyd&Fall [12] propose a well-designed technique which uses RED for queue management and uses the RED packet drop history to estimate arrival rates. It identifies unresponsive flows using this information and a set of tests. Their technique provides an incentive for flows to use end-to-end congestion as flows marked unresponsive can be penalized. We share their goal of dropping undeliverable packets. However, since their technique does not include a backward feedback mechanism similar to the

[1]For example, "RealVideo uses forward error correction (FEC) techniques to allow complete reconstruction of image packets that may be lost. The likelihood of frame damage occurring is reduced dramatically, making video frames impervious to all but extremely high levels of packet loss." [16]

[2]RFC 1812 [1] proposes several techniques to limit the rate at which source quenches are generated.

[3]Only the first quench causes a decrease, subsequent quenches within a single RTT estimate are ignored.

[4]No changes are made in the regulator bandwidth for a quenched flow for one RTT estimate after a source quench arrives; an increase is triggered if no source quench arrives for another RTT estimate.

source quenches used by ERUF, it is not able to drop undeliverable packets as early as possible. Furthermore, by moving the flow identification operations to the periphery of the network, ERUF can reduce the computational load at congested core routers.

A related research direction deals with development of UDP-based protocols that are "TCP-friendly" [20, 22, 27, 33, 35, 36]. These protocols use some form of end-to-end congestion control to try to limit the sending rate to a value close to that of a conformant TCP flow. We believe that this is an important research direction; since UDP-based protocols fill an obvious need, developing TCP-friendly versions of them helps reduce the danger of congestion collapse. However, some form of router-based regulation will always be necessary as we can no longer assume cooperation of all end-hosts.

We describe the ERUF scheme in Section 2. We describe the the algorithms used to install/maintain/remove the flow-specific regulators. We have evaluated the performance of ERUF for a variety of simulated network topologies and traffic patterns. Our results, presented in Section 3, show that ERUF is able to move most of the packet drops from unresponsive flows to the periphery of the network. The resources freed by dropping these packets early allows other flows to achieve significant performance improvement. Finally, we present conclusions and discuss open questions in Section 4.

## 2 Early Regulation of Unresponsive Flows (ERUF)

ERUF places the major burden of managing unresponsive flows on edge routers. Edge routers need to classify incoming packets, compute flow-specific arrival rates and rate-limit individual flows. Since edge routers are typically not on high-speed backbone links and typically do not deal with very large number of flows, the resource requirements are likely to be tractable. This approach of moving flow-specific operations to the periphery of the network has been taken by other researchers as well as by several IETF working groups [3, 5, 31].

ERUF requires that congested routers generate (rate-limited) source quenches in response to packet drops. These source quenches are used by edge routers to detect congestion (or incipient congestion) further downstream. There are pros and cons to using source quenches for this purpose. Using source quenches for propagating congestion information to edge routers has three advantages over setting an *explicit congestion notification (ECN)* bit in the packet header (the other main explicit congestion notification technique proposed in the literature [10, 26, 25]). First, source quenches are protocol-independent and can be used for protocols which do not have acknowledgment (*ack*) packets flowing back to the source; ECN-bit-based approaches work only for protocols that include *ack* packets. Second, even for protocols with *ack* packets, using source quenches requires *no* cooperation from the end-hosts (recall that the regulation is done by the edge routers and not the end-hosts). On the other hand, an ECN-bit-based approach relies on cooperation from the receivers to copy the ECN bit into *ack* packets. Finally, in a source-quench-based approach, edge routers need to inspect the relatively rare ICMP packets to extract congestion notifications whereas extracting congestion information from ECN bits would require edge routers to inspect every *ack* packet flowing through them.

The primary disadvantage of using source quenches is that it adds traffic in the reverse direction of the original flow. This can increase the congestion along paths that have multiple congested routers in both directions. Citing studies done in late 80s [7, 21], the authors of RFC 1812 [1] recommend that routers should *not* generate source quenches. The cited studies, however, were done using routers with a Drop-Tail packet-dropping discipline which can generate in a large number of source quenches in a short time. Along with others [10, 19], we believe that the introduction of active queue management techniques such as RED will limit the overhead of source quench messages. In Section 3, we present some simulation results for the impact of source quenches on congested reverse paths.

In the rest of this section, we describe the details of the ERUF architecture. We present algorithms for installing/maintaining/removing the flow-specific regulators. We would like to emphasize, however, that the main point of this paper is the overall ERUF technique; while we believe the specific algorithms described work reasonably well for the wide variety of topologies and traffic patterns we have considered in this paper, we expect them to evolve as the Internet community gains more experience with the dynamics of unresponsive flows.

## 2.1 Managing flow regulators

Edge routers classify packets and detect flows. They also estimate the arrival rate for each flow.[5] Edge routers use *Longest Queue Drop (LQD)* [32] for per-flow buffer management. Initially, no regulator is associated with a flow. As long as a flow receives no source quenches, it is allowed to continue unregulated. When a source quench arrives at an edge router, it identifies the flow the quench is associated with, records the arrival rate of the flow at that point (which we refer to as *regulator_max*) and imposes a token-bucket-based regulator which halves its sending rate. These regulators adjust their bandwidth using a multiplicative-decrease/additive-increase discipline. A decrease is triggered by the arrival of one or more source quenches within a single round-trip time (RTT) estimate and reduces the regulator bandwidth by half. Note that only the first quench causes a decrease in the regulator bandwidth; subsequent quenches within a single RTT estimate are ignored. This protects the regulator from being rapidly driven down by a sequence of source quenches before the source has had a chance to respond to the congestion.[6] No changes are made in the regulator bandwidth for a quenched flow for one RTT estimate after a source quench arrives. After this, the bandwidth of a regulator associated with a flow is increased by one *average* packet size for every RTT estimate that passes without an arrival of a source quench for the flow. When the regulator bandwidth is greater than or equal to *regulator_max* (i.e., the arrival rate of the flow when the regulator was imposed), the regulator is removed and the flow is allowed to continue unimpeded.

In the absence of *ack* packets, an edge router has no reliable way of determining the round trip time between itself and the congested router that does not involve injecting additional packets into the network. We consider two easy-to-compute approximations in this paper: (1) *nexthop*, which is two times the propagation delay on the next link towards the congested router;[7] and (2) $allhops = numhops \times nexthop$, where *numhops* is the number of hops between the edge router and the congested router. The first approximation (*nexthop*) is a strict lower bound on the actual RTT and is likely to lead to the regulator bandwidth increasing faster than a conformant TCP would increase its congestion window. The second approximation (*allhops*) is neither a lower bound nor an upper bound on the actual RTT.[8] However, for network topologies with long paths and terrestrial

[5]We currently use a simple window-based algorithm to estimate the arrival rate of flows. The size of window is dynamically adjusted depending on the burstiness of the flow. The window size is initially set to one and is increased by one for every packet till the estimate of the arrival rate converges. This algorithm worked well (i.e., converged rapidly with windows with 4-7 packets) for all the network topologies and traffic patterns we used in our experiments. In the long run, we plan to use a less memory-intensive scheme - for example, the algorithm proposed by Stoica et al [31] which uses exponential averaging with variable weights.

[6]A rapid sequence of source quenches can be generated either by a congested router with a Drop-Tail queue management discipline or (more rarely) by a RED router in the presence of extremely bursty traffic.

[7]This estimate has been previously used by Floyd&Fall [11].

[8]If the first link towards the congested router is a satellite link and all others are terrestrial, *allhops* can be an

links, it might provide a better approximation to the actual RTT. We empirically investigate the behavior of these approximations in Section 3 by using them for a variety of network topologies and traffic patterns.

ERUF does not require fine-grain flow identification and can work with flows of any granularity. However, ERUF works best if a small number of source quenches can significantly reduce the amount of data being sent towards the congested router. ERUF does not work well for traffic patterns with a large number of small unaggregated flows since each source quench results in only a small reduction in the amount of data being sent towards the congested router (each quench can at best reduce the sending rate of a single small flow).[9] Given that routers limit the rate at which source quenches are generated (as they should), it may not be possible to simultaneously quench enough of the flows. Note that this problem is not specific to ERUF. Traffic consisting of a large number of small unresponsive flows poses a problem for any congestion-management algorithm. Floyd&Fall [12] show that this traffic pattern causes congestion collapse even for scheduling techniques such as *weighted round-robin* which try to isolate individual flows. To handle this situation, we propose that edge routers use some degree of flow aggregation and identify flows either by source-destination pairs or by destination alone. Each aggregate is regulated as a whole and a source quench generated for *any* flow in an aggregate results in a decrease in the regulator bandwidth for the entire aggregate. We evaluate the impact of using aggregation in Section 3.

The ERUF algorithm is similar, in spirit, to the *SQuID* algorithm proposed by Prue&Postel in RFC 1016 for specifying *end-host* response to source quenches [24]. There are, however, several differences. First, the SQuID algorithm adjusts the per-packet regulator delay instead of the regulator bandwidth. For flows with variable packet size, this can lead to undesirable variations in the sending rate. Second, the SQuID algorithm uses a additive-increase/additive-decrease discipline for responding to source quenches. Subsequent research and experience has shown that multiplicative-decrease/additive-increase provides better stability [17, 18].

# 3 Experiments

We evaluated the performance of ERUF for a variety of simulated network topologies and traffic patterns. We used the *ns-2* simulator[10] for our experiments. For simulating scenarios with ERUF, we modified *ns-2* to: (1) generate a source quench every time RED drops a packet, (2) estimate arrival rates for flows, (3) snoop source quenches, and (4) install/maintain/remove regulators.

For the core set of experiments, we simulated five network topologies corresponding to frequently occurring scenarios. We ran each experiment for 500 seconds of simulated time. In the core set of experiments, we used the *nexthop* approximation for round-trip time between the edge router and the congested router (we present results for *allhops* in Section 3.2). For each topology, we compared the performance achieved with and without ERUF. For unresponsive flows, we used constant-bit-rate UDP flows as might be generated by streaming audio/video applications. For responsive flows, we used *ftp*-type long-term TCP flows.[11] We used 100 byte packets for the CBR flows and 1500 byte packets for the TCP flows.

To evaluate the ability of ERUF to regulate unresponsive flows, we used three metrics: *goodput*,

overestimate of the actual RTT (depending on queuing delays). If the satellite link is now moved to the second link towards the congested router, *allhops* is likely to be an underestimate of the actual RTT.

[9] By *small* flows, we mean flows whose sending rate is small compared to the bandwidth of the congested link.

[10] Available from *http://www-mash.cs.berkeley.edu/ns/*. We used version 2.1b5.

[11] The results presented in this paper were obtained using TCP SACK. We repeated the entire core set of experiments with TCP Tahoe and TCP Vegas but saw no significant difference in the results.

aggregate packet drop rate at edge routers and aggregate packet drop rate at non-edge routers. Floyd&Fall [12] define *goodput* of a flow as the bandwidth delivered to the receiver, excluding duplicate packets. The goodput metric shows the performance improvements achieved by the network by dropping undeliverable packets as early as possible. It also shows how well ERUF is able to shield responsive flows from the impact of unresponsive ones. The other two metrics, aggregate packet drop rate at edge routers and aggregate packet drop rate at non-edge routers, jointly provide an indication of the resources conserved by dropping undeliverable packets early.

To examine the impact of source quenches on congested reverse paths, we conducted a set of experiments with a network topology that is congested in both directions. Our goal was to determine how effective source quenches are in such a situation and what impact they have on performance of competing traffic. To evaluate the efficacy of ERUF for traffic patterns with a large number of small flows and to examine the impact of flow aggregation, we simulated two congestion scenarios: one that has a varying number of flows with the same total sending rate and the other that aggregates all flows to the same destination.

In addition, we evaluated the impact of ERUF on competing TCP flows for several topologies and up to 10 competing flows. We repeated these experiments with TCP Reno/Tahoe/Vegas/SACK. ERUF did not cause more than one additional packet drop in any experiment.[12]

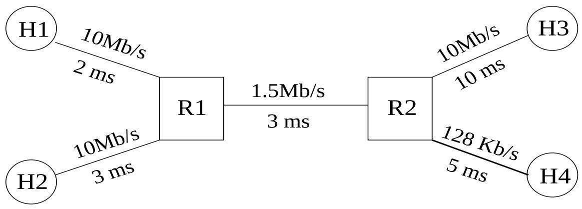
### 3.1 Core experiments

**Dumbbell scenario:** this scenario, shown in Figure 1, corresponds to a long-haul link connecting two local-area networks and an ISDN line. The congested long-haul link is shared by one TCP flow (H1 to H3) and three CBR flows (from H2 to H4). This scenario was originally used by Floyd&Fall [12] to demonstrate congestion collapse due to lack of end-to-end congestion control. Figure 1 shows that for this scenario, ERUF drops almost all undelivered packets at the periphery of the network. Note that the total number of packets dropped with and without ERUF is about the same. The goodput graphs in Figure 1 show that the resources freed by dropping packets early allows the TCP flow from H1 to H3 to increase its goodput by a factor of up to 3.5.

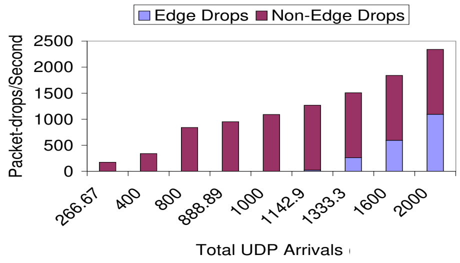
**Multi-dumbbell scenario:** in the *dumbbell* scenario, all three CBR flows being regulated originated from the same domain (or host). We used the *multi-dumbbell* scenario, shown in Figure 2, to evaluate the effectiveness of ERUF's regulation when there are multiple unresponsive flows going to the same destination from different domains/hosts. This scenario is a variant of the *dumbbell* with two CBR flows from H2 to H4 and one CBR flow from H5 to H4. Like *dumbbell*, it has one TCP flow from H1 to H3. As in the *dumbbell* scenario, ERUF is successful in dropping most of the undeliverable packets early. The impact of ERUF on the goodput of the TCP flow is much higher in this case (in the extreme case, it is about 35 times higher). In the absence of ERUF, the goodput of the TCP flow continues to drop below 400 Kb/s (where it levels off for *dumbbell*) as its packets can be dropped at multiple routers (R1 and R2) [9].

**Cross-Traffic scenario:** in the first two scenarios, the competing TCP and CBR flows were either sourced from the same domain or sink'ed in the same domain. This scenario, shown in Figure 3, represents cross-traffic situations where competing flows share neither source nor sink. The traffic consists of two CBR flows from H1 to H3, two CBRs from H2 to H4 and one TCP flow from H5 to H6. As shown in Figure 3, ERUF is able to move nearly all packet drops to the periphery of the network and allows the TCP flow achieve up to a 20-fold improvement in goodput – without ERUF, the TCP goodput falls to about 50 Kb/s whereas with ERUF, it stabilizes at about 1 Mb/s.

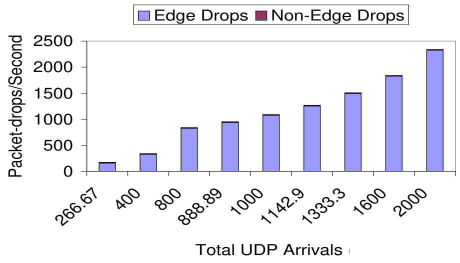
[12]Detailed results illustrating this could not be included due to space limitations.



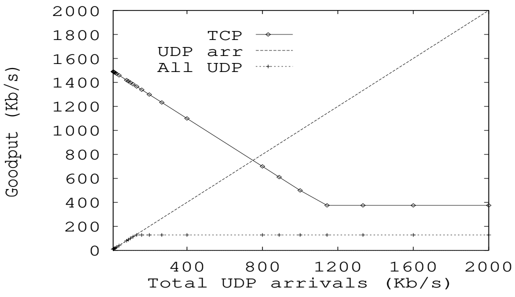

The traffic consists of one TCP flow from H1 to H3 and three equal CBR flows from H2 to H4. The total sending rate of the CBR flows is varied between 8.89 Kb/s to 1999.99 Kb/s.



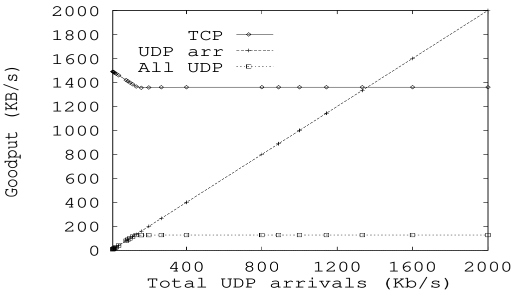

Packet drop rates without ERUF




Packet drop rates with ERUF




Goodputs without ERUF



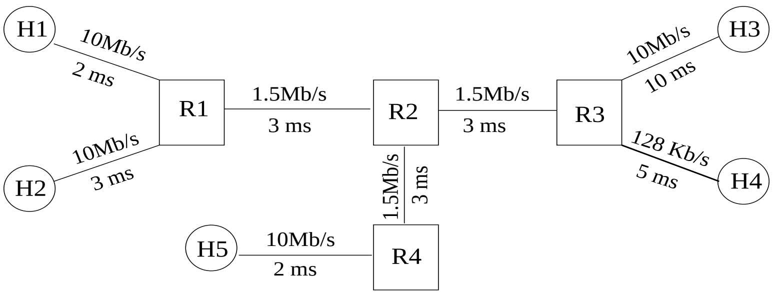

Goodputs with ERUF

Figure 1: Description and results for the *dumbbell* scenario.

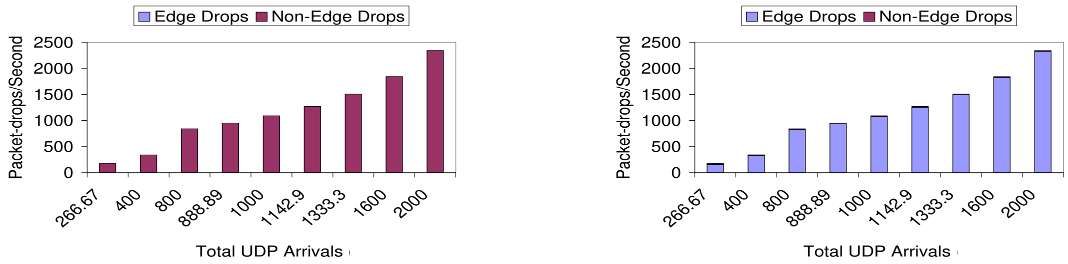
Note that the goodput of TCP in this case is lower than in the *dumbbell* scenario. This is to be expected as the TCP flow traverses more congested routers [9].

**Waist scenario:** this scenario, shown in Figure 4, represents situations in which a single link shared by the responsive and unresponsive flows is the only congested link in their paths (e.g., flows between two campus-size networks). The traffic pattern consists of three CBR flows (H2 →H6, H3 → H7, H4 → H8) and one TCP flow (H1 → H5). Without ERUF, there are no appreciable packet drops till the offered load is more than the long haul link's bandwidth. The packet drop rate climbs rapidly thereafter. Note that the unresponsive flows hog as much of the long haul link's bandwidth as possible resulting in poor performance for TCP. With ERUF, the regulators are able to drop a significant number of packets from unresponsive flows before they reach the waist which significantly improves the goodput for the TCP flow.

**Video-server scenario:** the previous scenarios considered traffic patterns with competition between responsive and unresponsive flows. This scenario, shown in Figure 5, represents situations in all the competing flows are unresponsive. Such a situation would occur, for example, with a streaming-content provider with multiple server machines. The traffic pattern consists of one CBR

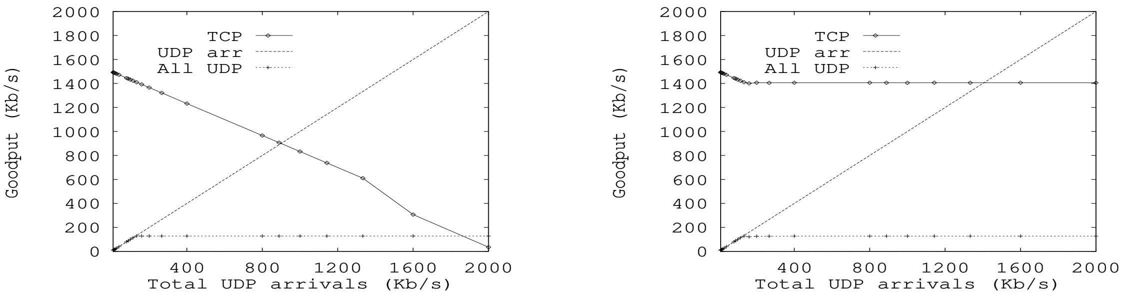



The traffic consists of one TCP flow from H1 to H3, two CBR flows from H2 to H4 and one CBR flow from H5 to H4. All CBR flows have equal sending rates; the total sending rate of the CBR flows is varied between 8.89 Kb/s to 1999.99 Kb/s.




Packet drop rates without ERUF

Packet drop rates with ERUF




Goodputs without ERUF

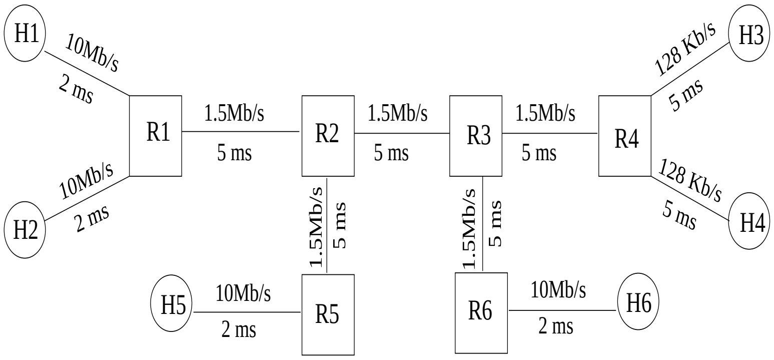
Goodputs with ERUF

Figure 2: Description and results for the *multi-dumbbell* topology.

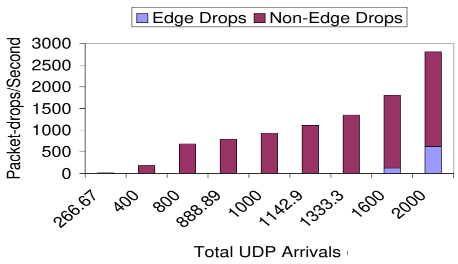
flow from H1 to H4, one CBR flow from H2 to H5 and three CBR flows from H3 to H6. Note that this scenario is similar to the *dumbbell* scenario, the main difference being that all competing flows are unresponsive. As for other scenarios, ERUF is able to move almost all packet drops to the periphery of the network. A large fraction of the drops occur for the three flows from H3 to H6 which share a 128 Kb/s link. The resources freed by dropping these packets early allows the other two flows (H1 $\rightarrow$ H4 and H2 $\rightarrow$ H5) to improve their goodput. Without ERUF, the R1 $\rightarrow$ R2 link saturates when the total UDP arrival rate (from all flows) reaches 1.5 Mb/s; without ERUF, the three flows from H3 to H6 use at most 130 Kb/s of this link (averaged over 500 seconds).

## 3.2 Evaluating RTT approximations

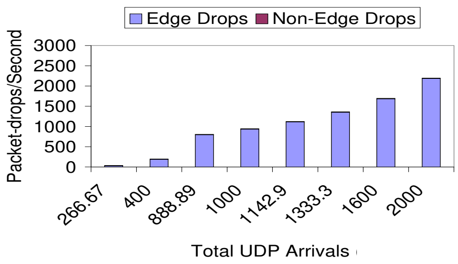
To compare the two approximations proposed for the round-trip time between the edge router and the congested router, we repeated the entire core set of experiments with the *allhops* approximation. For all five topologies and traffic patterns, we found no significant difference (in either the packet



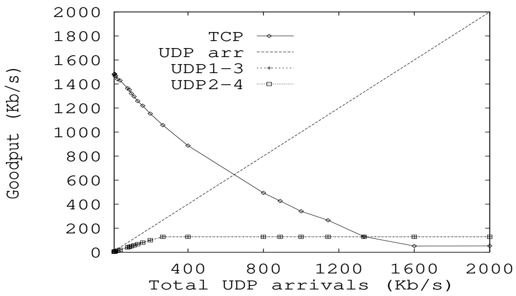

The traffic consists of one TCP flow from H5 to H6, two CBR flows from H1 to H3 and two CBR flows from H2 to H4. All CBR flows have equal sending rate; the total sending rate of the CBR flows is varied between 8.89 Kb/s to 1999.99 Kb/s.



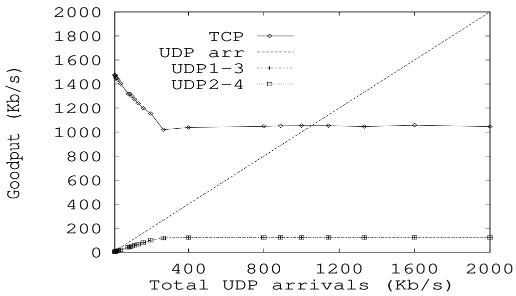

Packet drop rates without ERUF




Packet drop rates with ERUF




Goodputs without ERUF

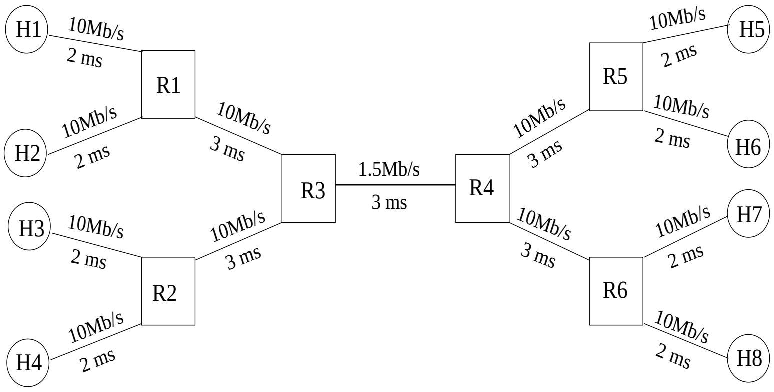



Goodputs with ERUF

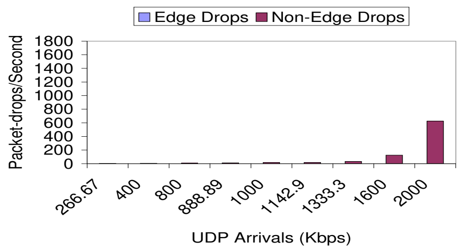
Figure 3: Description and results for the *cross-traffic* topology.

drop rates or the goodput of individual flows) between the two approximations. To convince ourselves, we constructed two additional topologies, shown in Figure 6, which accentuated the differences between the approximation techniques. These topologies are variants of *dumbbell* with an additional satellite link inserted: the first topology, *NearSat*, has the satellite next to the edge router and the other, *FarSat*, has the satellite away from the edge router. For *NearSat*, *nexthop* is a reasonable approximation for RTT and *allhops* is a loose over-estimate. For *FarSat*, both *nexthop* and *allhops* are loose under-estimates.

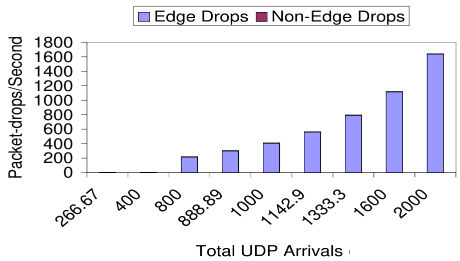
As shown in Figure 7, for both topologies, ERUF is able to move most packet drops to the network periphery. The choice of the approximation for RTT appears to make little difference. Similarly, the goodput for the TCP flow improves significantly for both topologies and the selection



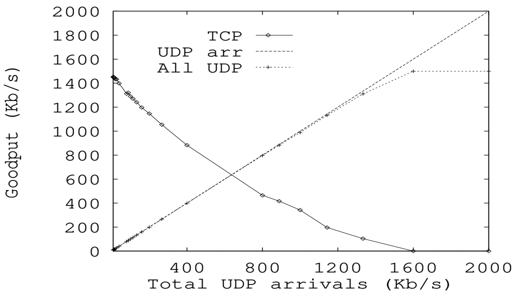

The traffic consists of one TCP flow from H1 to H5, one CBR flow from H2 to H6,one CBR flow from H3 to H7 and one CBR flow from H4 to H8. All CBR flows have equal sending rates; the total sending rate of the CBR flows is varied between 8.89 Kb/s to 1999.99 Kb/s.



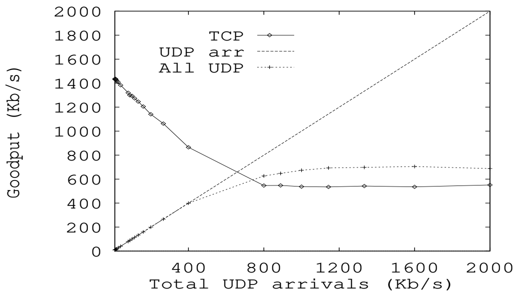

Packet drop rates without ERUF




Packet drop rates with ERUF




Goodputs without ERUF

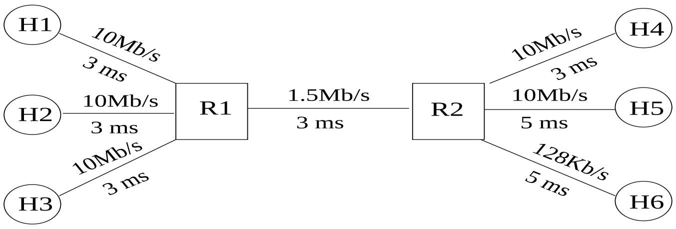



Goodputs with ERUF

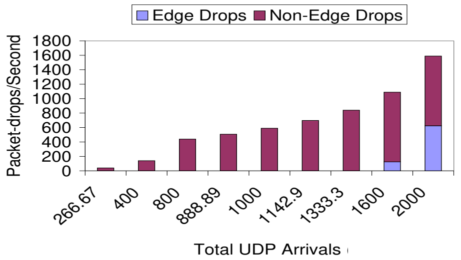
Figure 4: Description and results for the *waist* topology.

of the RTT approximation makes little difference. Note, however, that in the *FarSat* case, the total goodput of the CBR flows with ERUF is less than their goodput without ERUF. This is because both approximations under-estimate RTT. As a result, the regulator bandwidth grows much faster than it should resulting in more quenches being generated. Furthermore, since the RTT estimate after a source quench is also used to ignore subsequent quenches, a loose underestimate results in fewer quenches being ignored. As a result, the unresponsive flows get quenched more often than they should on *FarSat*.[13]

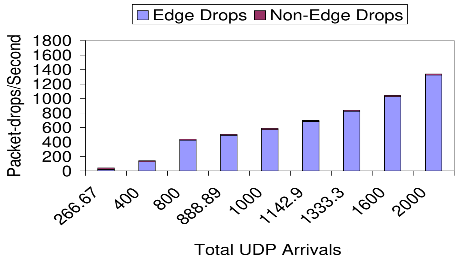
[13]Detailed results illustrating this could not be included due to space limitations.



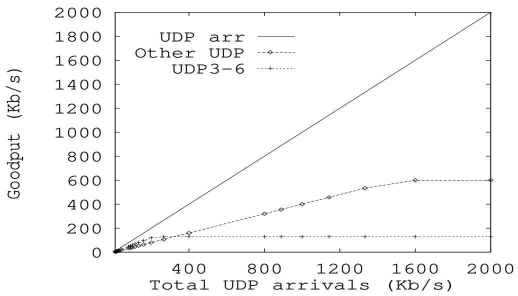

The traffic consists of one CBR flow from H1 to H4, one CBR flow from H2 to H5 and three CBR flows from H3 to H6. All CBR flows have equal sending rates; the total sending rate of the CBR flows is varied between 8.89 Kb/s to 1999.99 Kb/s.



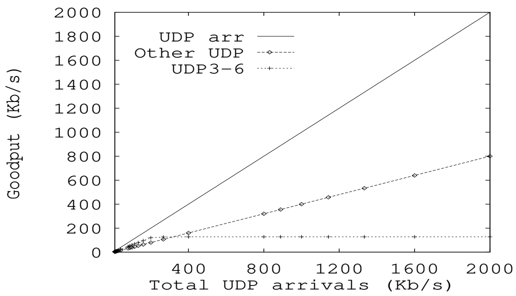

Packet drop rates without ERUF




Packet drop rates without ERUF

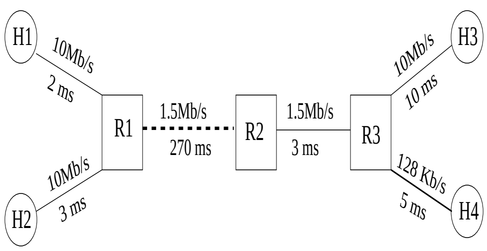



Goodputs without ERUF

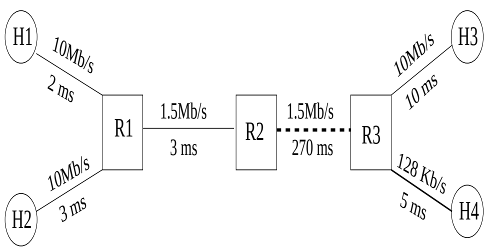



Goodputs with ERUF

Figure 5: Description and results for the *video-server* topology.

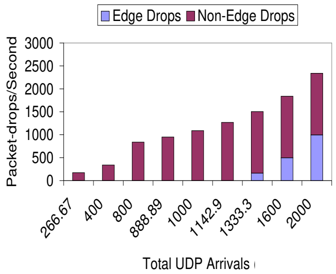



*NearSat*

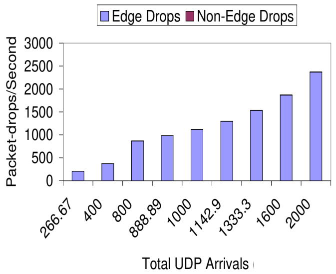



*FarSat*

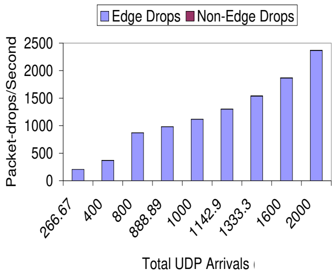
Figure 6: The *NearSat* and *FarSat* topologies. The satellite links are shown by dashed lines. The traffic consists of one TCP flow from H1 to H3, and three CBR flows from H2 to H4. All CBR flows have equal sending rates; the total sending rate of the CBR flows is varied between 8.89 Kb/s to 1999.99 Kb/s.



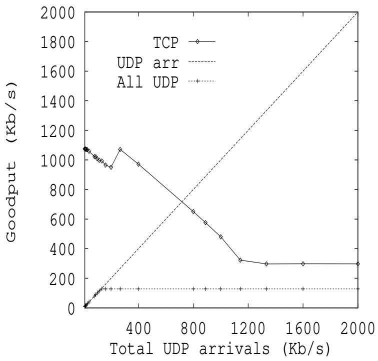

*NearSat:* without ERUF



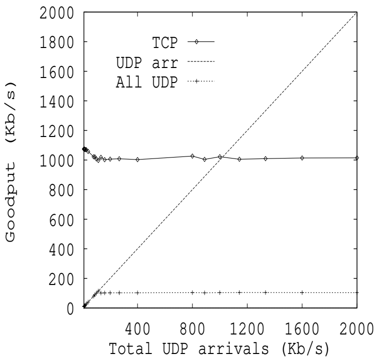

*NearSat:* with ERUF & *nexthop*



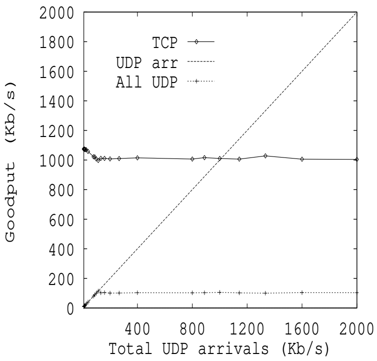

*NearSat:* with ERUF & *allhops*



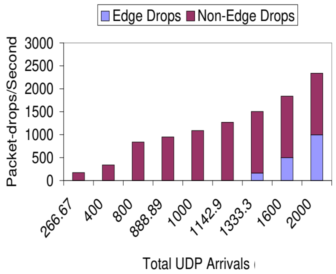

*NearSat:* without ERUF



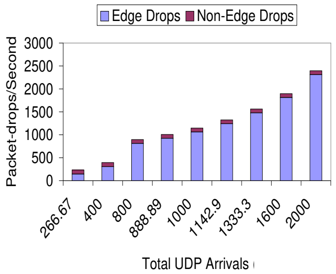

*NearSat:* with ERUF & *nexthop*



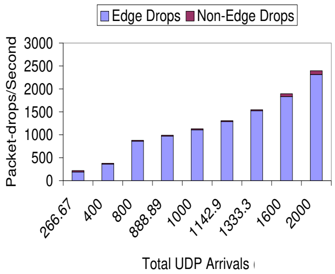

*NearSat:* with ERUF & *allhops*



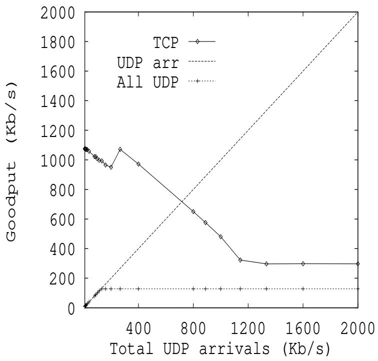

*FarSat:* without ERUF



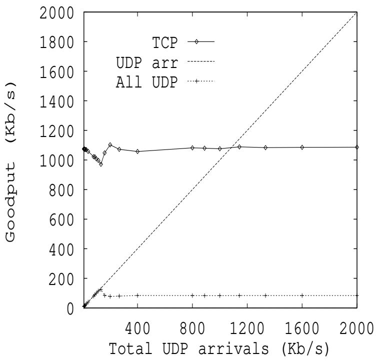

*FarSat:* with ERUF & *nexthop*



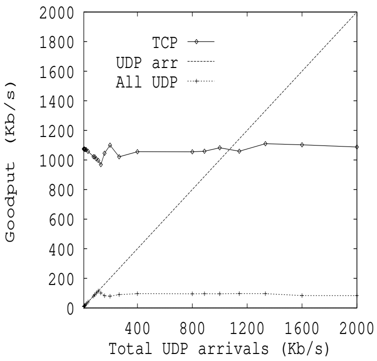

*FarSat:* with ERUF & *allhops*




*FarSat:* without ERUF




*FarSat:* with ERUF & *nexthop*




*FarSat:* with ERUF & *allhops*

Figure 7: Results for the *NearSat* and the *FarSat* topologies.

### 3.3 Impact of source quenches on congested reverse paths

To examine the impact of source quenches on congested reverse paths, we used a variant of the *dumbbell* scenario with additional traffic in the reverse direction (see Figure 8). The additional traffic is a single CBR flow from H5 to H6 whose sending rate is varied from 8.89 Kb/s to about 2000 Kb/s. The other traffic consists of three CBR flows with an aggregate sending rate of 1 Mb/s from H2 to H4 and one TCP flow from H1 to H3. We conducted two experiments - one with ERUF enabled in all routers and the other with ERUF enabled in all routers *except R4*. The goal of doing the second experiment was to evaluate the maximum impact of source quenches (using ERUF in R4 will reduce the sending rate for the flow from H5 to H6 which competes with the source quenches). Without ERUF being enabled in R4, we see no significant impact of source quenches on the goodput of any flow. With ERUF enabled in all routers, the source quenches generated by drops at R4 result in regulation of the backward flow – this effect occurs when the sending rate for the reverse flow reaches the bandwidth of the reverse link. There is no significant difference in the goodput of the other flows. From these results, we believe that with ERUF and RED, the impact of source quenches is likely to be positive even in the presence of a congested reverse path.

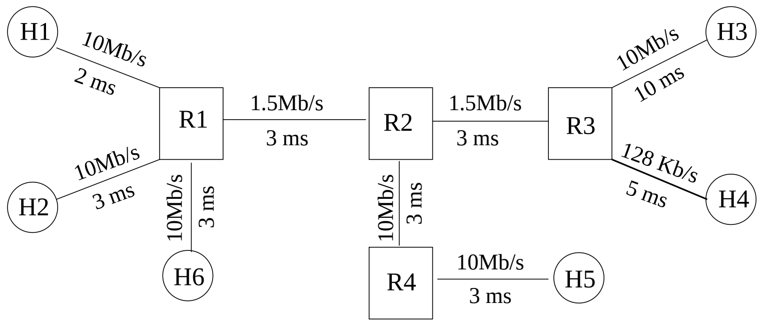
### 3.4 Handling large number of small flows

To evaluate the efficacy of ERUF for this situation and to examine the impact of flow aggregation, we simulated a scenario with a varying number of flows with the same total sending rate (see Figure 9). We conducted two experiments: one with fine-grain flow identification and the other with flow aggregation that aggregates all flows to the same destination. The traffic pattern consisted of one TCP flow from H1 to H4 and 1-50 CBR flows from H2 to H3 (with an aggregate sending rate of 1.5 Mb/s). Without ERUF, we find that the unresponsive flows use up almost all of the bandwidth in the R2-R3 link. With ERUF, the performance depends on the number of competing CBR flows. For a small number of flows, ERUF drops a significant fraction of the packets from the unresponsive flows which allows the TCP flow to achieve significantly better goodput. As the number of CBR flows increases (and the sending rate of individual flows drops), each source quench results in fewer drops. As a result, the total number of drops reduces; as does the goodput of the TCP flow. If, however, we aggregate all the 50 CBR flows between H2 and H3, ERUF achieves the same performance for this aggregate as it does for a single large flow. This result argues for flow-aggregation at edge routers. Several IETF working groups recommend a similar aggregate-at-edge-routers approach [3, 5].

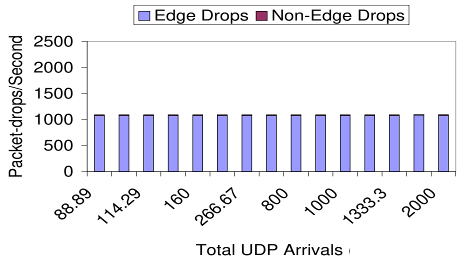
## 4 Conclusions and open questions

In this paper, we have presented router mechanisms to regulate unresponsive best-effort traffic. The goal of the proposed mechanisms is to drop undeliverable packets (i.e., packets that will be dropped somewhere in the network before reaching their destination) as close to the periphery of the network as possible.

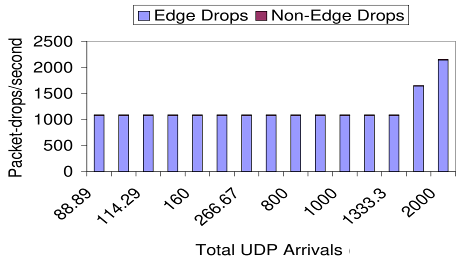
Simulation results for a variety of network topologies and traffic patterns show that the proposed mechanisms are able to move most packet drops from unresponsive flows to the periphery of the network. The resources freed by dropping these packets early allows other flows to achieve significant performance improvement. ERUF works best if a small number of source quenches can significantly reduce the amount of data being sent towards the congested router. To handle a large number of small bandwidth flows, we propose that edge routers aggregate flows.



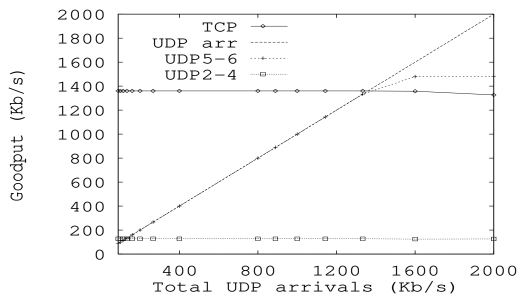

The traffic consists of one TCP flow from H1 to H3, three CBR flows with a total sending rate of 1 Mb/s and one CBR flow with different sending rates from H5 to H6.



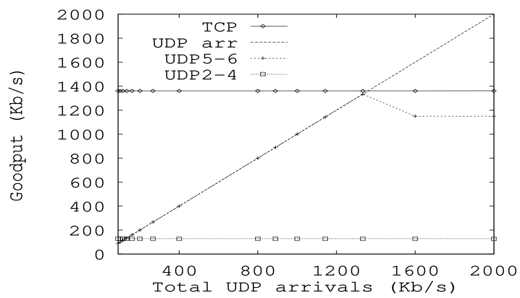

Packet drop rates without ERUF in R4




Packet drop rates with ERUF in all routers



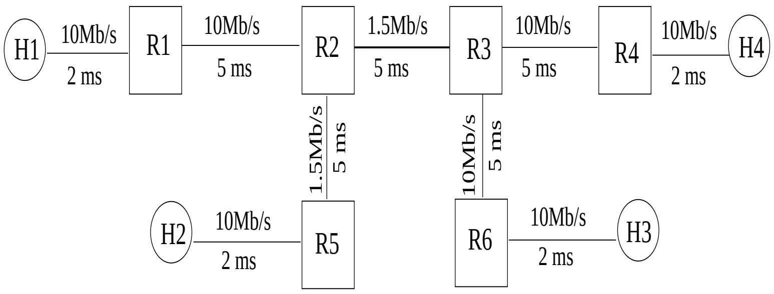

Goodputs without ERUF in R4



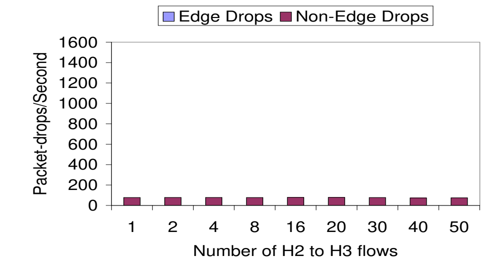

Goodputs with ERUF in all routers

Figure 8: The impact of source quenches for networks that are congested in both directions.

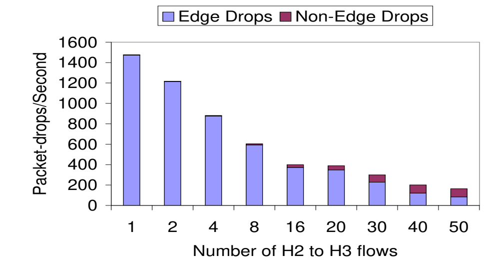
In this paper, we have focused on regulation of unresponsive flows. Many TCP implementors are developing "faster" TCP implementations that eliminate one or more of the congestion control mechanisms (e.g., [8, 28]). As these implementations proliferate, they too can cause congestion collapse. Since ERUF tends to underestimate the round-trip time between the edge router and the congested router, it increases the regulator bandwidth faster than a conformant TCP would increase its congestion window. This leaves a window for non-conformant TCP implementations to exploit. We plan to extend ERUF to reduce this window. One of the alternatives we are considering is a hybrid of ERUF and the technique proposed by Floyd&Fall [12]. The idea is to estimate the maximum bandwidth for a TCP-conformant flow at the congested router (as proposed in [12]) and to piggyback this information on the source quench. This could allow the edge router to impose tighter regulators on errant flows.



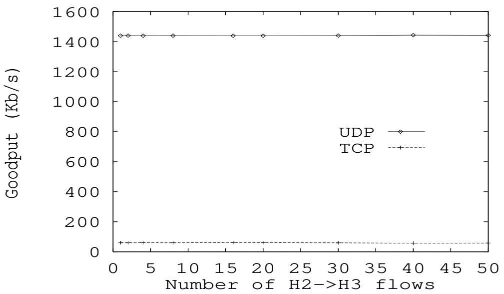

The traffic consists of one TCP flow from H1 to H4, and 1-50 CBR flows from H2 to H3. The total sending rate of the CBR flows is 1.5 Mb/s for all configurations.



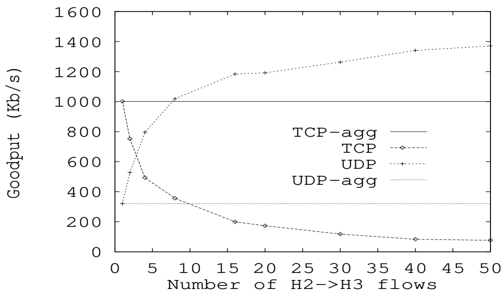

Packet drop rates without ERUF




Packet drop rates with ERUF




Goodputs without ERUF




Goodputs with ERUF

Figure 9: The impact of source quenches for large number of small flows and for flow-aggregation. The `TCP-agg` and `UDP-agg` lines in goodput graphs correspond to the experiment with one TCP flow and 50 CBR flows grouped into a single aggregate.

# References


[1] F. Baker. Requirements for IP version 4 routers. RFC 1812, Network Working Group, Jun 1995.

[2] J. Bennett and H. Zhang. $WF^2Q$: Worst-case fair weighted fair queuing. In *Proc. of INFO-COM'96*, pages 120–8, 1996.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, Network Working Group, Dec 1998.

[4] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the internet. RFC 2309, Network Working Group, April 1998.

[5] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A framework for multiprotocol label switching. Network Working Group Internet Draft[14], Nov 1997.

[6] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. In *Proc. of SIGCOMM'89*, pages 3–12, 1989.

[7] G. Finn. A connectionless congestion control algorithm. *ACM Computer Communicatons Review*, 19(5), Oct 1989.

[8] Flash Networks' SatBooster Enhances Internet Communications up to Fivefold Over Satellite Links. Available at *http://www.flash-networks.com/press.htm*, 1997. Flash Networks Press Release.

[9] S. Floyd. Connections with multiple congested gateways in packet switched networks part i: One-way traffic. *ACM Computer Communication Review*, 21(5):30–47, Oct 1991.

[10] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5):10–23, 1994.

[11] S. Floyd and K. Fall. Router mechanisms to support end-to-end congestion control. Available at *http://www-nrg.ee.lbl.gov/floyd/papers.html*, Feb 1997.

[12] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. Submitted for publication, 1998.

[13] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug 1993.

[14] S. Golestani. A self-clocked fair queuing scheme for broad-band applications. In *Proc. of INFOCOM'94*, pages 636–46, 1994.

[15] RealNetworks Inc. RBN$^{TM}$ (Real Broadcast Network) White Paper. Available at *http://www.real.com/solutions/rbn/whitepaper.html*, Jan 1999.

[16] RealNetworks Inc. RealVideo Technical White Paper. Available at *http://www.real.com/-devzone/library/whitepapers/overview.html*, Jan 1999.

[17] V. Jacobson. Congestion avoidance and control. In *Proc. of SIGCOMM'88*, pages 314–29, 1988.

[18] R. Jain. A delay based approach for congestion avoidance in interconnected heterogenous networks. *ACM Computer Communication Review*, 19(5):56–71, Oct 1989.

[19] P. Karn. Email message to end2end mailing list *Re: Buffer size = BW-delay product*[15], Apr 7 1998.

[14] Available at *http://www.ietf.org/internet-drafts/draft-ietf-mpls-framework-02.txt*

[15] Available at *ftp://ftp.isi.edu/end2end/*.

[20] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control. Technical note sent to the end2end-interest mailing list[16], Jan 1997.

[21] A. Mankin, G. Hollingsworth, G. Reichlen, K. Thompson, R. Wilder, and R. Zahavi. Evaluation of internet performance - FY89. Technical Report MTR-89W00216, MITRE Corporation, Feb 1990.

[22] J. Padhye, Jim Kurose, D. Towsley, and R. Koodli. A TCP-Friendly Rate Adjustment Protocol for Continuous Media Flows over Best Effort Networks. Technical Report TR 98-04, Dept of Computer Science, University of Massachusetts, Amherst, Oct 1998. A brief version will appear in SIGMETRICS'99.

[23] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control – the single node case. In *Proc. of INFOCOM'92*, 1992.

[24] W. Prue and J. Postel. Something a host could do with a source quench. RFC 1016, Jul 1987.

[25] K. Ramakrishnan and S. Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481, Network Working Group, Jan 1999.

[26] K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, 8(2):158–81, 1990.

[27] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *Proc. of IEEE INFOCOM'99*, 1999. To appear.

[28] E. Roberts. IP Acceleration Software: Torquing up TCP/IP. Available at *http://www.data.com/hot_products/speedseeker.html*, Jan 1998. Data Communications Online.

[29] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. In *Proc. of SIGCOMM'94*, pages 47–57, 1994.

[30] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round robin. In *Proc. of SIGCOMM'95*, pages 47–57, 1995.

[31] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks. In *Proc. of SIGCOMM'98*, 1998. 118-30.

[32] B. Suter, T. Lakshmanan, D. Stiliadis, and A. Choudhary. Design considerations for supporting TCP with per-flow queuing. In *Proc. of INFOCOM'98*, 1998.

[33] W. Tan and A. Zakhor. Internet video using error resilient scalable compression and cooperative transport protocol. In *Proc. of ICIP Volume 3*, pages 458–62, 1998.

[34] A. Tannenbaum. *Computer Networks*. Prentice-Hall, third edition, 1996.

[35] T. Turletti, S. Parisis, and J. Bolot. Experiments with a layered transmission scheme over the internet. Technical Report 3296, INRIA Sophia-Antipolis, Nov 1997.

[36] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proc. of INFOCOM'98*, 1998.

[16] Available at *http://www.psc.edu/networking/papers/tcp_friendly.html*