On Class-based Isolation of UDP, Short-lived and Long-lived TCP Flows*

SELMA YILMAZ IBRAHIM MATTA
Computer Science Department
Boston University
Boston, MA 02215, USA
{selma,matta}@cs.bu.edu

Abstract

The congestion control mechanisms used in TCP makes it vulnerable in an environment where flows with different congestion-sensitivity compete for scarce resources. With the increasing amount of unresponsive UDP traffic in today's Internet, new mechanisms are needed to enforce fairness in the core of the network. We propose a Diffserv-like architecture, where flows with different characteristics are classified into separate service queues at the routers. Such class-based isolation provides protection so that flows with different characteristics do not negatively impact one another. The architecture is simple and scalable, since core routers only need to keep state information for each class rather than for each flow. In this study, we examine different aspects of UDP and TCP interaction and possible gains from segregating UDP and TCP into different classes. We also investigate the utility of further segregating TCP flows into two classes, which are class of short and class of long flows. The short-lived and long-lived TCP flows have different characteristics like burst size, burstiness of arrivals, tolerance to packet loss, and sensitivity to delay. Isolation prevents negative impacts of interacting short and long TCP flows in a shared environment. Results are obtained analytically for both Tail-drop and Random Early Drop (RED) routers, and verified by simulation. Class-based isolation have the following salient features: (1) better fairness, (2) improved predictability for all kinds of flows, (3) lower transmission delay for delay-sensitive flows, (4) reduced jitter and playout buffers for UDP-based multi-media traffic, and (5) better control over Quality of Service (QoS) of a particular traffic type.

Keywords: Class-based isolation vs. sharing, packet dropping (RED, Tail-drop), fairness, queuing analysis.

1 Introduction

Internet traffic is a mixture of smooth, unresponsive (or not sufficiently responsive) real-time traffic (UDP) and bursty, congestion-sensitive traffic (TCP). However, mixing flows with different characteristics can cause performance problems. Real-time traffic not only performs poorly because of delay variations and packet drops, but also hurts congestion-sensitive traffic when they compete for scarce bandwidth [17]. TCP congestion control works well in isolation but in aggregate can be unfair [9]. In times of congestion, responsive TCP flows tend to give up bandwidth to competing unresponsive UDP flows, which is the main reason for performance degradation and unfairness for congestion-sensitive flows. In addition to bandwidth starvation imposed on responsive traffic, uncontrolled deployment of unresponsive traffic could also lead to Internet-wide congestion collapse [11]. To solve fairness issues among TCP and UDP flows, we should not focus on penalizing UDP flows. UDP flows need to have the same fair treatment as TCP flows as they usually carry multi-media content [15].

To be able to provide protection, so that flows with different characteristics do not negatively impact one another, we propose and examine a Diffserv-like architecture, where flows with different characteristics are classified into separate service queues at the routers. This can be implemented using a class-based queuing scheme [8] or by routing different groups of flows on (logically) separate communication paths [19] as described in [13]. The proposed architecture has the advantages of Diffserv architectures such as simplicity and scalability, since core routers only need to keep state information for each class

*This work was supported in part by NSF grants CAREER ANI-0096045 and MRI EIA-9871022.

rather than for each flow.

We also argue for further isolation of TCP flows as long-lived and short-lived, because they also have very different characteristics. Short-lived TCP flows arrive in a more bursty fashion, which prevents long TCP flows from operating in a predictable mode. Short TCP flows seldom need to operate beyond the slow start phase of TCP to finish their transmission, that is why their window size is generally smaller, so they generate smaller packet bursts. Since long-lived TCP connections mostly operate in congestion avoidance phase probing for bandwidth with larger windows, they produce larger packet bursts. TCP connections with large window sizes are more tolerant of packet loss than those with small windows. While long TCP flows may recover from multiple packet losses in one round-trip time (RTT) [6], short TCP flows may have to wait for a timeout period to recover from a single packet loss. It is observed that long TCP flows may completely shut off short TCP flows [13]. This causes performance problems for short TCP flows, which generally carry interactive/delay-sensitive data.

Class-based isolation, at low cost, provides a fair environment for TCP flows, which is also a better environment for real-time data carried by UDP flows. Isolation limits the delay and jitter that UDP flows experience in the queue. This will lead to smaller playout buffers for real-time data, and better predictability for all.

The rest of the paper is organized as follows: Section 2 explains our analytic model, describes our experiments, and performance measures. General observations and results of individual experiments are presented in Section 3 and Section 4, respectively. Section 5 revisits related work and finally Section 6 concludes the paper.

2 Experiments and Analytic Model

2.1 Buffer Management Schemes

In this study, we examine FIFO queuing with Tail-drop and RED with mixture of different traffic types such as UDP, long-lived TCP and short-lived TCP. We investigate the effects of isolation for each case.

FIFO queuing with Tail-drop is the most widely implemented and deployed queuing mechanism in today's routers because of its simplicity. However, it has problems such as tendency to penalize bursty connections and not providing any kind of protection against misbehaving flows that send more than their share. A single source, sending packets to the router at sufficiently high rate, can capture arbitrarily high fraction of the bandwidth of the outgoing line. Thus, in non-cooperative environments, Tail-drop would fail to provide fairness. RED on the other hand, hopes to eliminate bias against bursty traffic by limiting the queue occupancy so that there is always room left in the queue to buffer transient bursts. RED addresses fairness by dropping packets from flows in proportion to their bandwidths. In non-cooperative environments, we expect RED to perform better than Tail-drop, however, in [12, 7, 18], it is shown that RED alone may not provide fairness when a mixture of different traffic types share the same link. Also, it has been shown that RED prevents bias against bursty traffic by increasing the drop probability of smooth traffic [2]. This means RED would penalize UDP and TCP flows with small packet bursts to improve overall fairness.

2.2 Analytic Model

Bonald *et al.* [2] evaluated the performance of RED against Tail-drop. Their analytic model for RED uses the following approximation: "All packets in the same burst see the same drop probability." In this paper, we relax this approximation so as to accurately evaluate the performance of short- and long-lived TCP flows with different packet burst sizes. Here we summarize the way the parameters are set and describe the analytic model for both kinds of buffer management scheme: Tail-drop and RED.

For TCP flows, bursts of B packets arrive according to a Poisson process of rate λ_{TCP}. Since TCP is window-based, it sends a burst of packets every round-trip time (RTT). That is why we fix λ_{TCP} to 1/RTT bursts/second and vary the number of TCP flows, N, to vary TCP load. Throughout the study, we assume all TCP flows of the same class (short- or long-lived) have the exact same characteristics. We choose RTT so that $\lambda_{TCP} = 8.3$ bursts/sec. For UDP flows, packets arrive according to a Poisson process of rate λ_{UDP}. We vary UDP load through λ_{UDP}.

The total offered TCP load is $\rho_{TCP} = (B \times \lambda_{TCP} \times N)/\mu$. The total offered UDP load is $\rho_{UDP} = \lambda_{UDP}/\mu$. Thus the total offered load is $\rho_{Total} = \rho_{TCP} + \rho_{UDP}$.

The processing times of the packets in the router are assumed to be exponentially distributed with mean $1/\mu$. The queue length is set to the bandwidth-delay product, where the bandwidth is μ and delay is RTT.

The burst size, B, models the average window size of a flow during its life-time. Since long-lived TCP flows mostly operate in congestion avoidance phase and short-lived TCP flows mostly operate in slow-start, long-lived TCP flows will have bigger window sizes on average than short-lived TCP flows [3]. Unless otherwise stated, we take B=1, B=4, and $B > 4$ for UDP, short-lived TCP and long-lived TCP flows, respectively.

The number of packets buffered in the queue defines a Markov chain. Let π denote the stationary distribution of the total number of packets in the queue. Using PASTA property, the drop probability of a packet from a TCP flow and UDP flow in a Tail-drop router is given by:

$$P_{TD}(TCP) = \pi(K) + \pi(K-1)\frac{(B-1)}{B} + .. + \pi(K-B+1)\frac{1}{B} \tag{1}$$

and

$$P_{TD}(UDP) = \pi(K) \tag{2}$$

where K is the buffer size of the router in terms of packets.

With the RED buffer management scheme, incoming packets are dropped with a probability that is an increasing function $d(.)$ of the average queue size k_{avg}. The average queue size is computed using an exponential weighted moving average:

$$k_{avg} = (1-w)k_{avg} + wk \tag{3}$$

where w is fixed and k is the instantaneous queue size when a packet arrives. As in [2], we assume for simplicity that the drop rate depends on instantaneous queue size k rather than on average queue size k_{avg}. Thus, the drop function, $d(.)$, is defined as

$$d(k) = \begin{cases} \frac{(i-min_{th})}{(max_{th}-min_{th})} max_p & \text{if } \mathrm{min}_{th} \leq k < \mathrm{max}_{th}, \\ 0 & \text{if } k < \mathrm{min}_{th}, \\ 1 & \text{if } k \geq \mathrm{max}_{th}. \end{cases} \tag{4}$$

In all the experiments, for RED gateway, the parameters are $\mathrm{min}_{th}=K/2$ packets, $\mathrm{max}_{th}=K$ packets, and $\mathrm{max}_p=1$.

The model in [2] makes the following approximation: "The RED router uses the same drop probability $d(k)$ for all packets in the same burst, where k is the instantaneous queue size at the time the first packet in the burst arrives at the router." With this approximation, the model gives a lower bound on the drop rate. Using PASTA property, the drop probability of a packet from a TCP flow or a UDP flow in a RED router is approximately given by:

$$P_{RED}(TCP) = P_{RED}(UDP) = \sum_{k=1}^{K} \pi(k)d(k) \tag{5}$$

This approximation underestimates TCP drop probability, especially for large burst (window) sizes. Throughout our experiments, we need to compare performance of short- and long-lived TCP flows when they share the same queue. However, the model with the above approximation is only accurate if the burst size, B, is not large compared to the buffer size. That is, for fixed K, smaller values of B give more accurate results than larger values of B. We overcome this inaccuracy by removing the approximation. We next illustrate how the transition probabilities of the modified Markov chain are computed.

The probability of going from state k to state $(k+1)$, i.e. exactly one packet of the burst is not dropped, is given by:

$$P_1 = \sum_{j=0}^{B-1} d^j(k)(1-d(k))(d(k+1))^{(B-j-1)}. \tag{6}$$

Similarly, the probability of going from state k to state $(k+2)$, i.e. exactly two packets of the burst are not dropped, is given by:

$$P_2 = \sum_{\forall(i+j+l)=B-2} d^i(k)(1-d(k))d^j(k+1)(1-d(k+1))d^l(k+2). \tag{7}$$

Other transition probabilities can be similarly computed. To compute the drop probability of TCP under RED, instead of equation (5), we compute the expected number of drops for a typical TCP packet burst. Thus the drop probability is given by:

$$P_{RED}(TCP) = \sum_{k=1}^{K} \pi(k)(\sum_{n=0}^{B} P_n \frac{(B-n)}{B}) \tag{8}$$

where P_n is the probability that exactly n packets of the TCP burst are not dropped. The drop probability of UDP can be computed as a special case where B=1,

$$P_{RED}(UDP) = \sum_{k=1}^{K} \pi(k)d(k). \tag{9}$$

We verified our analytic model by simulation and the results (not shown) exactly matched.

2.3 Performance Parameters and Measures

The experiments in this study fall under the following categories:

- Effect of fraction of service rate allocated to different classes
- Effect of burst size on interacting flows
- Effect of having both short-lived and long-lived TCP flows
- Effect of isolation into two classes (UDP and TCP) vs. three classes (UDP, short-lived TCP, and long-lived TCP)

Each experiment is repeated for the cases where flows with different characteristics compete with each other and where they are isolated. For the rest of the paper, we refer to the first case as $sharing$ or $mixed$ case, and the second case as $isolation$. In isolation, the service rate and queue size are split in proportion to the load introduced by each class while buffer management scheme is kept the same. We use K_{class_type} and μ_{class_type} to refer the queue size and service rate allocated to a particular class, respectively. All experiments are done for both Tail-drop and RED routers, and repeated for high load (where total load=2) and low load (where total load=1). The main performance measures are: (1) drop probability as a measure of effective goodput, (2) average queue size as a measure of transmission delay, and (3) fairness across flow types. For fairness, we used Chiu and Jain's fairness index [4], which is defined as

$$f = \frac{(\sum_{i=1}^{N} x_i)^2}{N(\sum_{i=1}^{N} x_i^2)}.$$

where x_i is the drop probability of flow-type$_i$.

The average queue size is computed by $\sum_{k=1}^{K} k \cdot \pi(k)$.

3 General Observations

Before presenting the experiments, we summarize our observations:

- At high load (load $\geq$ 1), isolation improves fairness across TCP and UDP classes over shared Tail-drop by reducing drop probability of TCP at the expense of increased drop probability for UDP. Although the performance of shared RED is as good as isolated Tail-drop queues, isolated queues have an advantage that shared RED cannot provide: Better control over quality of service of each traffic type by controlling the allocation of resources to each class.
- Unlike shared RED or Tail-drop, isolation of UDP and TCP significantly reduces drop probability of short-lived TCP flows, which constitute the majority of TCP flows over the Internet. Since short-lived TCP flows are usually interactive/delay-sensitive, isolation improves the quality of service of this class. Decreased drop probability of short-lived TCP flows would also improve fairness among short TCP flows, which are more likely to timeout to detect packet drops. Timeouts are not desirable as they lead to extended idle periods and degradation of throughput.
- Further isolation of TCP ensures fairness across all 3 types of flows: UDP, short-lived TCP, and long-lived TCP. By having a separate queue for each class, it is easier to manage quality of service of a particular traffic type. If we want to increase quality of service of short-lived TCP flows, which are usually interactive/delay-sensitive and tend to timeout on packet drops, we can just allocate more resources to the class of short TCP flows to further improve quality.
- While taking advantage of isolated queues across different traffic classes, using RED within each class may further improve the intra-class quality by preventing synchronization of TCP flows as RED randomizes losses.
- At low load, we observe that isolation lowers statistical multiplexing gain compared to sharing. To avoid this, dynamic resource allocation as in CBQ [8] should be used.
- Isolation creates a better environment for real-time data by limiting the delay and jitter that UDP flows experience in the queue. This results in more predictable UDP performance and smaller playout buffers at the receiver.
- Isolation simplifies the problem of providing deterministic delay bounds by increasing the predictability of all kinds of flows.

4 Experiments

4.1 Effect of Service Rates Allocated to Classes

4.1.1 Experimental Setup

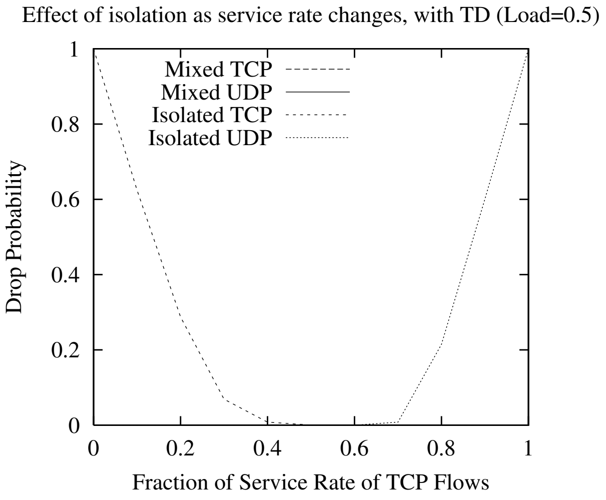
In this experiment effects of isolation into two classes, namely UDP class and TCP class, are observed as resources allocated to each class are varied.

The experiment in this section is repeated for different total load assignments of 0.5, 1, 1.5, 2 and 2.5. λ_{TCP}=8.3 bursts/second, B_{TCP}=4 packets, μ=930 packets/second, and K=56 packets. The offered load by TCP and UDP are equal. For different load values, N_{TCP} and λ_{UDP} are computed. In corresponding isolated cases, the fraction of service rate and queue size allocated to TCP is varied. The remaining of the resources not assigned to the TCP class are allocated to the UDP class. When there is excess resources (load $<$ 1), they are divided equally between the TCP and UDP classes.

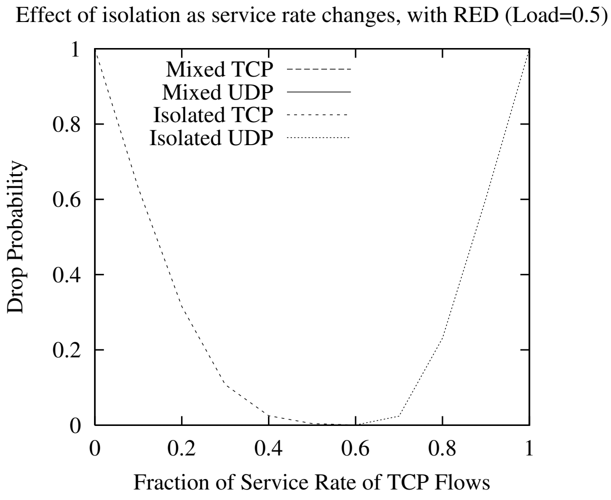
4.1.2 Results

Results are shown in Figure 1. Figures 1(a) and (b) show the cases when total arrival rate of packets is less than the total service rate. Figures 1(c) and (d) show the cases when total arrival rate of packets is equal to the total service rate. The rest of the figures are for the cases when total arrival rate of packets is greater than the total service rate. In all figures, as service rate allocated to TCP increases, service rate allocated to UDP decreases. The drop probability is inversely proportional to the allocated service rate and queue length as expected.

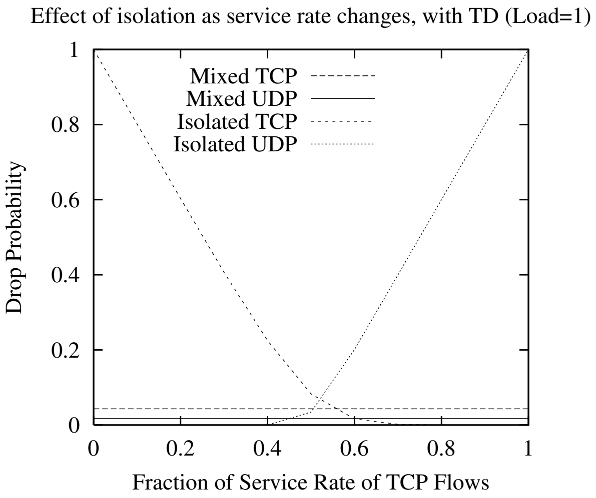
The observations for Tail-drop can be summarized as follows: When total load $\leq$1 (Figures 1(a) and (c)), we do not have a scenario such that in isolation, resources can be split between the two classes in such a way that fairness is improved without increasing the drop probability of both TCP and UDP. This is because in low-load environments, isolation reduces


Effect of isolation as service rate changes, with TD (Load=0.5)
Mixed TCP
Mixed UDP
Isolated TCP
Isolated UDP
Drop Probability
Fraction of Service Rate of TCP Flows

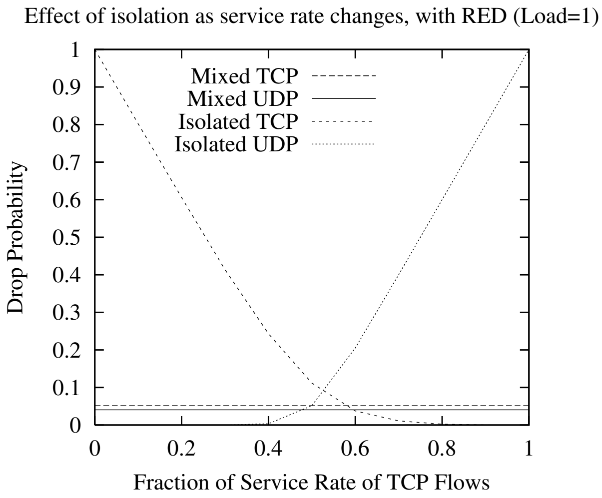
(a)


Effect of isolation as service rate changes, with RED (Load=0.5)
Mixed TCP
Mixed UDP
Isolated TCP
Isolated UDP
Drop Probability
Fraction of Service Rate of TCP Flows

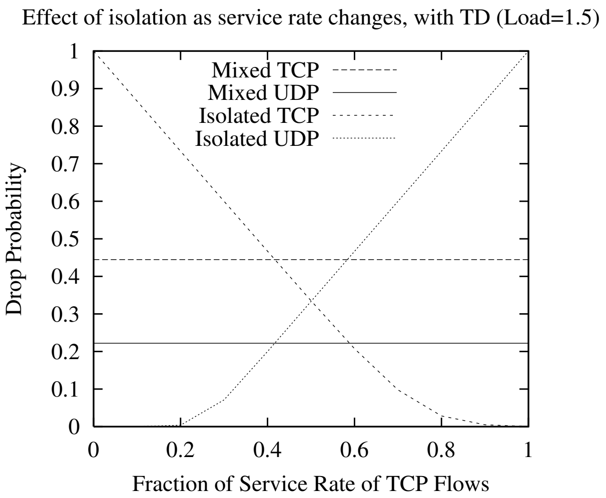
(b)


Effect of isolation as service rate changes, with TD (Load=1)
Mixed TCP
Mixed UDP
Isolated TCP
Isolated UDP
Drop Probability
Fraction of Service Rate of TCP Flows

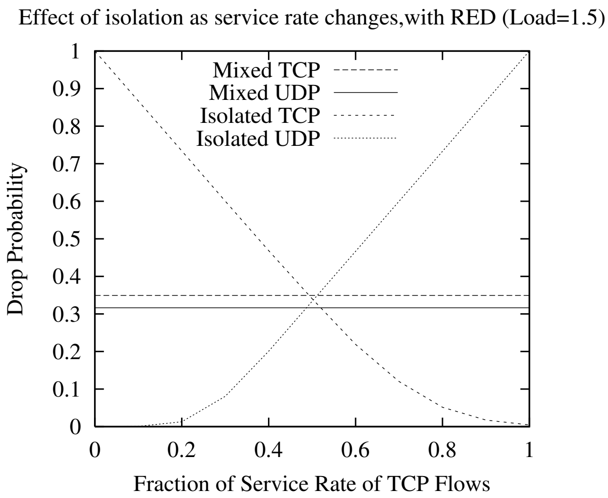
(c)


Effect of isolation as service rate changes, with RED (Load=1)
Mixed TCP
Mixed UDP
Isolated TCP
Isolated UDP
Drop Probability
Fraction of Service Rate of TCP Flows

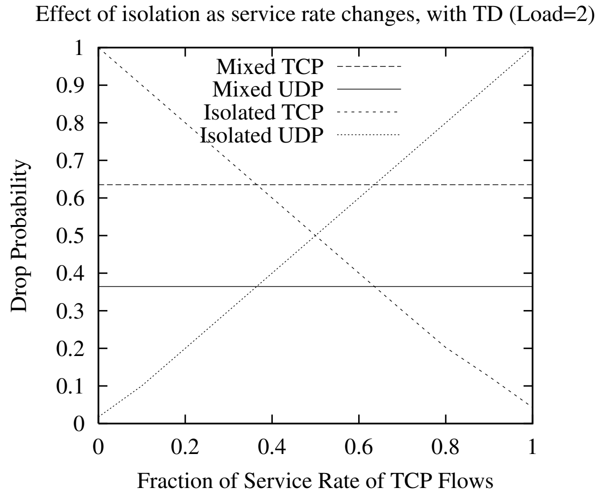
(d)


Effect of isolation as service rate changes, with TD (Load=1.5)
Mixed TCP
Mixed UDP
Isolated TCP
Isolated UDP
Drop Probability
Fraction of Service Rate of TCP Flows

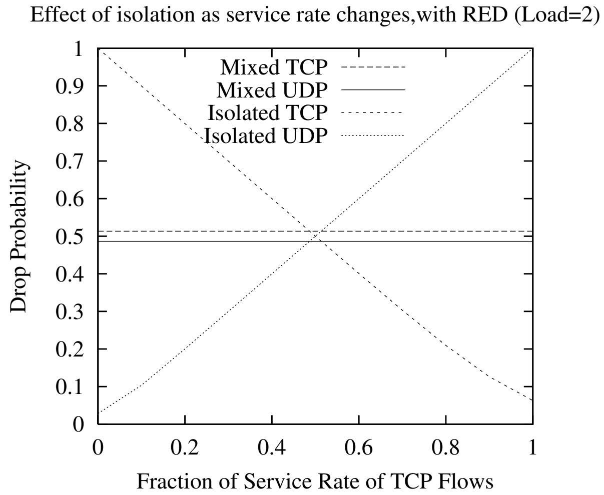
(e)


Effect of isolation as service rate changes,with RED (Load=1.5)
Mixed TCP
Mixed UDP
Isolated TCP
Isolated UDP
Drop Probability
Fraction of Service Rate of TCP Flows

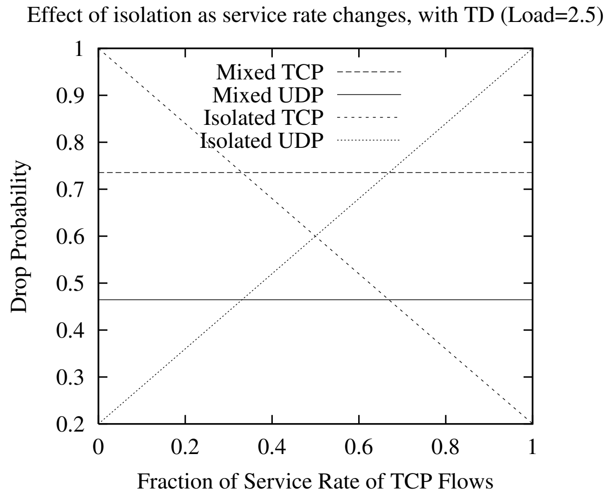
(f)



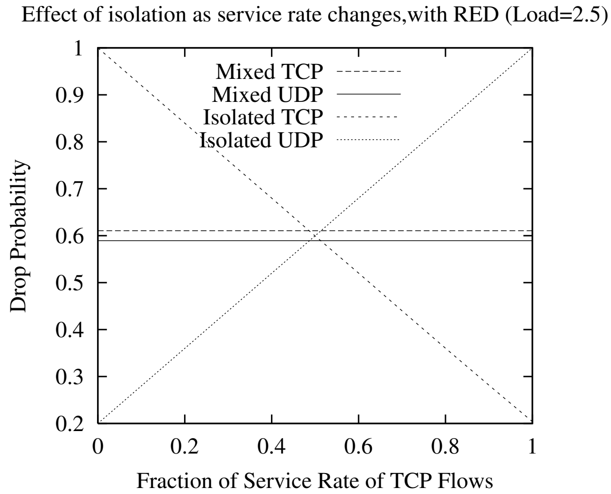
(g)



(h)



(i)



(j)

Figure 1. This figure shows effects of isolation as service rate and queue size allocated to TCP class increases whereas service rate and queue size allocated to UDP class decreases. The figures on the left are for Tail-drop router and the figures on the right are for RED router. Each instance has a different total load assignment as shown in the titles.

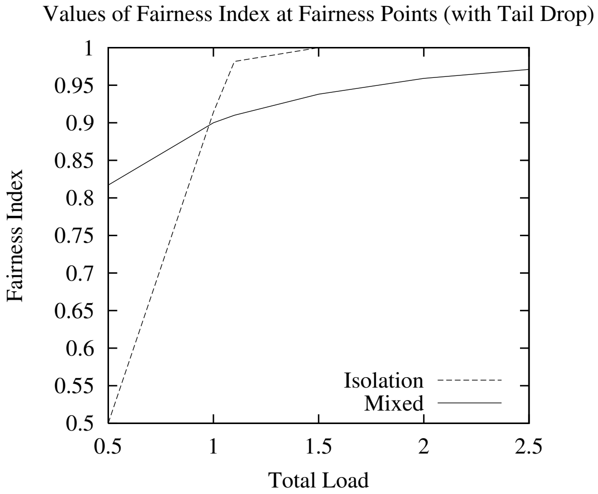


Figure 2. Fairness index for both isolation and sharing at points where resources are split equally among TCP and UDP classes (with Tail Drop).

statistical gains obtained by sharing. When total load increases (Figures 1(e), (g), and (i)), we observe that in isolation, if we split the resources equally between the two classes, we have the same drop probability for both kinds of traffic. This provides a very fair environment (see Figure 2). Table 1 shows the values of parameters of the points where fairness is improved by isolation. The common property of such points is $\mu_{TCP} = \mu_{UDP}$ and $K_{TCP} = K_{UDP}$. This is because $\rho_{TCP}/(\rho_{TCP} + \rho_{UDP}) = 50\%$. Therefore, fairness can be achieved by class-based isolation, where total service rate is divided in proportion to the load of TCP and UDP.

	UDP			TCP				
$Load$	μ	K	λ	μ	K	λ	N	B
Load=1.5	465	28	697.5	465	28	8.3	21	4
Load=2	465	28	930	465	28	8.3	28	4
Load=2.5	465	28	1162.5	465	28	8.3	35	4

Table 1. Parameters at points where fairness is observed.

We observe that at points where isolation improves fairness, the drop probability of TCP flows is reduced by the same amount as the drop probability of UDP is increased. Figure 3 plots the percentage of improvement for the TCP class. The gain from isolation is not an increasing function of load, but still significant (greater than 18%) for all practical cases.

In the Tail-drop model, as load increases, $\pi(k)'s$ change in a manner that $\pi(0)$ decreases, and $\pi(K)$ increases. Intuitively, the system will be in the state where the queue is empty with smaller probability and will be in the state where the queue is full with higher probability. As can be seen from equations (1) and (2), $P_{TD}(UDP)$ is affected only by increases in $\pi(K)$, whereas $P_{TD}(TCP)$ is also affected by increases in $\pi(K-1), \cdots, \pi(K-B+1)$. The question is how $\pi(K)$ vs. $\pi(K-1), \cdots, \pi(K-B+1)$ change with load? This directly affects drop probabilities of TCP and UDP and fairness among them. For $\rho_{Total} \leq 2.5$, which covers all practical cases, multiplexing UDP and TCP always hurts TCP to some degree. To see this, for mixed case, Figure 4 plots the ratio $\frac{\pi(K)}{\pi(K)+\pi(K-1)\frac{3}{4}+\pi(K-2)\frac{2}{4}+\pi(K-3)\frac{1}{4}}$ vs. $\log_2(\rho_{Total})$. The relationship is almost linear with slope given by:

$$\frac{P_{TD}(UDP)}{P_{TD}(TCP) \cdot log_2(\rho_{Total})} \approx 0.17. \tag{10}$$

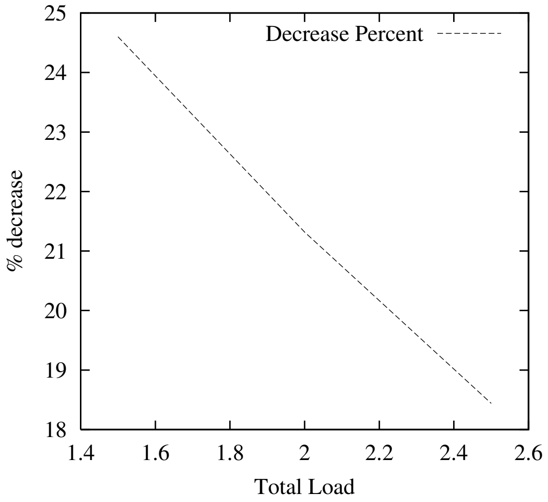


Figure 3. Percentage of decrease in drop probability of TCP flow when flows are isolated (at points where fairness is observed with Tail Drop).

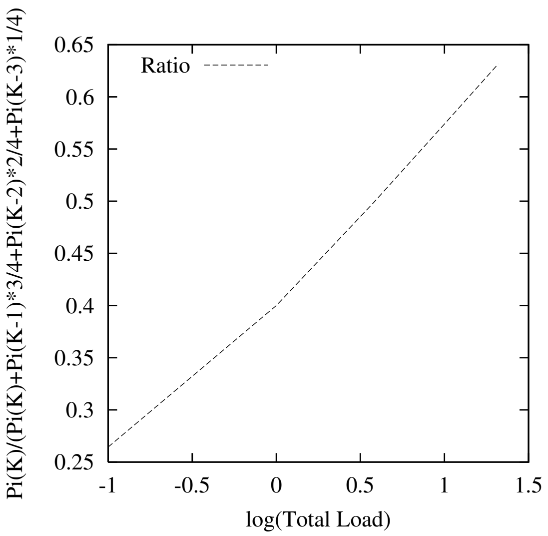


Figure 4. This figure shows the ratio of $P_{TD}(UDP)/P_{TD}(TCP)$ for different loads in the mixed (sharing) case. Burst size of TCP is 4 packets.

Using equation (10), we can find the value of total load where TCP would have equal or lower drop probability than that of UDP, i.e. $0.17 \times \log_2(\rho_{Total}) \geq 1$, which yields $\rho_{Total} \geq 2^6$. This shows that, for all practical cases, mixing UDP and TCP traffic would hurt TCP traffic.

Figures 1(f), (h), and (j), show that multiplexing TCP and UDP in a RED router does not hurt TCP flows seriously. Isolation provides little improvement of TCP drop rate and fairness.

In conclusion, if Tail-drop buffer management scheme is used, isolation is necessary to improve overall fairness, and performance of TCP flows. We observe that a shared RED queue performs as well as isolated Tail-drop queues. However, isolated queues have the advantage of providing better control over quality of service for a particular class.

4.2 Effect of Burst Size

4.2.1 Experimental Setup

The goal of this experiment is to examine the way UDP and TCP flows interact as the burst size of TCP flows increases. Effects of isolating traffic into UDP and TCP classes are observed in terms of drop probability of each flow, fairness, and average queue size. We consider two scenarios:

(1) *High Load:* ρ_{Total}=2, μ_{Total}= 1660 packets/second, K=100 packets, ρ_{UDP}= ρ_{TCP}=1, λ_{UDP}=1660 packets/second, λ_{TCP}= 8.3 bursts/second. N and B_{TCP} are varied.

(2) *Low Load:* Parameters are the same as high load, except ρ_{Total}=1, ρ_{UDP}= ρ_{TCP}=0.5, λ_{UDP}=830 packets/second.

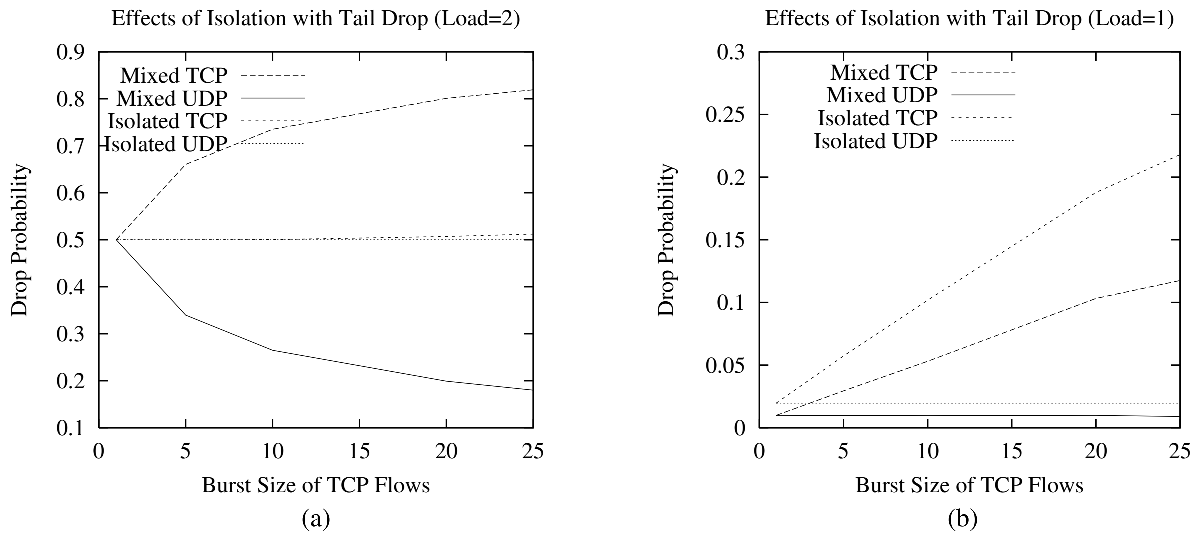


Figure 5. Effects of isolation as burst size of TCP flows increases (with Tail Drop): (a) High Load (b) Low Load

4.2.2 Results

The results are shown in Figure 5 and Figure 6 for Tail-drop and RED, respectively.

At high load, both Tail-drop and RED show similar behavior: Mixing TCP and UDP hurts TCP. As burst size of TCP increases, performance degradation to TCP increases. Of course, even in this case, RED is much more fair than Tail-drop. The average drop probability of TCP and UDP in the shared case is very close to individual drop probabilities in isolation, which means that what is lost by TCP class is gained by UDP class. The reason behind shared Tail-drop's behavior is its known bias against bursty traffic [12, 2, 10]. As burst size increases, it becomes more difficult to find a space in the queue to

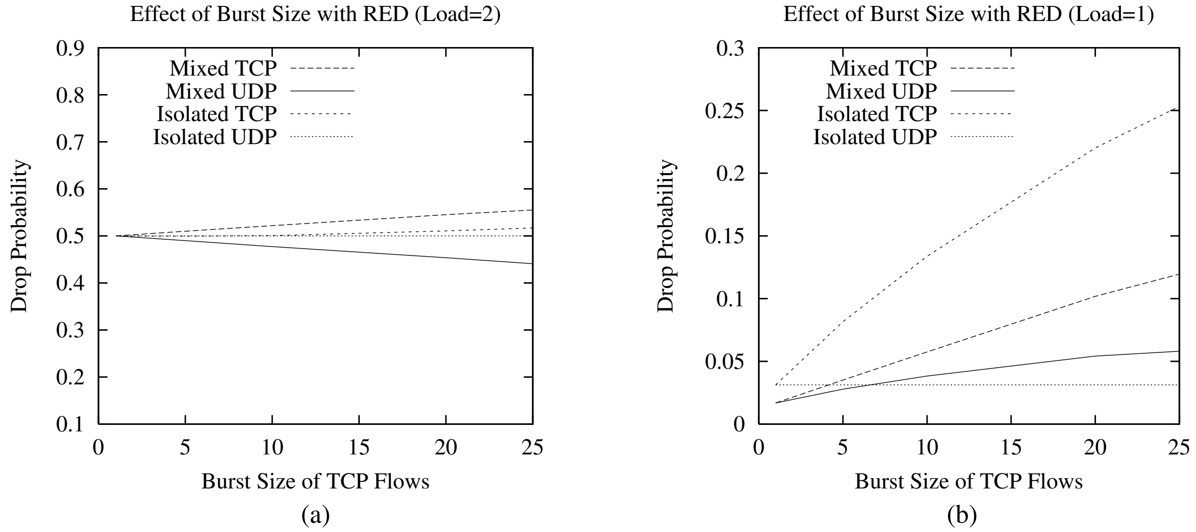


Figure 6. Effects of isolation as burst size of TCP flows increases (with RED): (a) High Load (b) Low Load

squeeze into. For shared RED, since we are using instantaneous queue size rather than average queue size, the big transient bursts might not be handled the way they could have. However, even in this case, unfairness to TCP does not seem serious (see Figure 7).

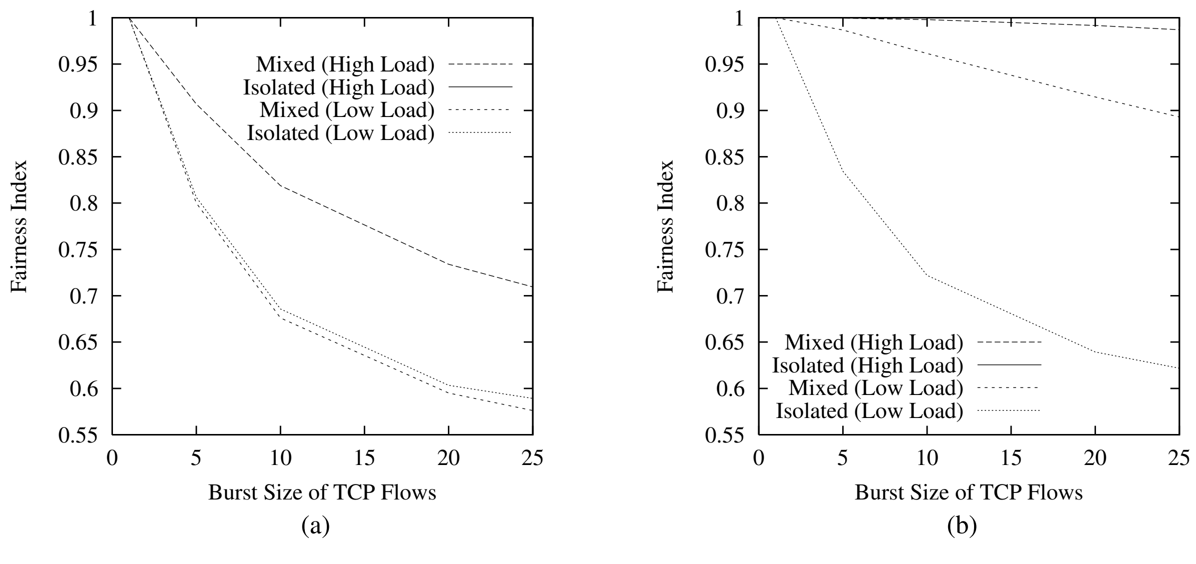


Figure 7. Fairness Index for effect of burst size experiments: (a) Tail Drop (b) RED

At low load, for both Tail-drop and RED, isolation increases drop probability of TCP and does not seem to be a necessary alternative. The reason is that at low load, isolation reduces statistical gains compared to sharing.

In conclusion, when the load is high, which means network resources are scarce, isolating traffic into TCP and UDP classes provides fairness and performance improvements for TCP flows. Isolation also improves predictability for both TCP and UDP flows. The gain is more obvious for longer-lived TCP flows (with large bursts) as shown by the fairness index in Figure 7. If Tail-drop buffer management is deployed, isolation is a necessity, whereas for RED, the gain observed by

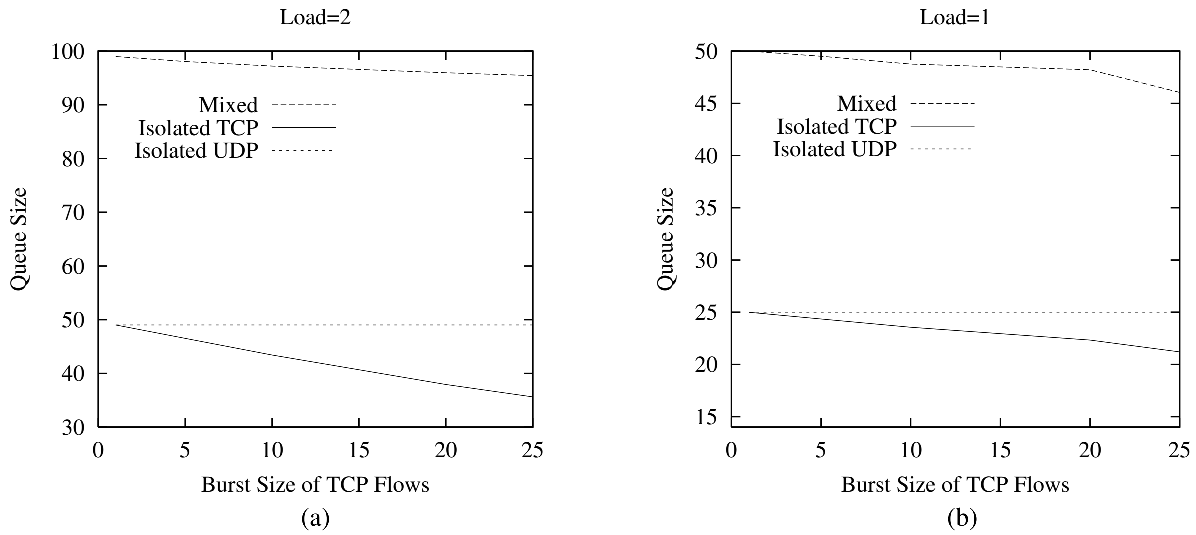


Figure 8. Queue Size for each class for effect of burst size experiment (with Tail Drop): (a) High Load (b) Low Load

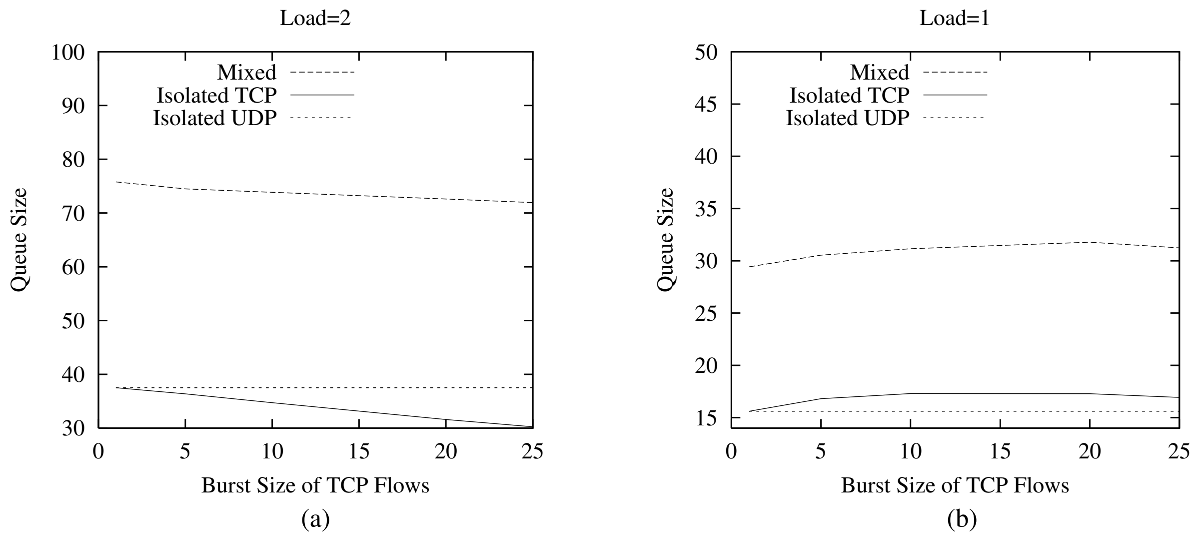


Figure 9. Queue Size for each class for effect of burst size experiment (with RED): (a) High Load (b) Low Load

isolation is not significant. However, isolation helps to reduce transmission delay, which can be measured by average queue length, for both Tail-drop and RED queues. The average queue size is shown in Figures 8 and 9. For all cases, isolation provides smaller queue size. Note also that for all cases, RED has smaller queue size than Tail-drop. This is because RED begins dropping packets as soon as instantaneous queue size reaches $K/2$ rather than waiting for the queue to fill completely. Both isolated RED and Tail-drop provide UDP (usually carrying multi-media traffic) a consistent low delay service in spite of the presence of very bursty (TCP) traffic. This would lower jitter and increase predictability. Overall, isolation creates a better environment for real-time traffic. The results about the queue size (equivalently, transmission delay) observed here are also observed for the rest of the experiments, so they are not shown in the remainder of the paper.

4.3 Effects of Isolation into 2 Classes

4.3.1 Experimental Setup

The goal of this experiment is to see the effects of isolation of flows into two classes, namely TCP and UDP. In this experiment, the TCP class has two kinds of flows: long-lived TCP, and short-lived TCP. We consider two scenarios:

(1) *High Load:* ρ_{Total}=2, μ_{Total}= 1660 packets/second, K=100 packets, $\rho_{UDP}/(\rho_{TCP-long}+\rho_{TCP-short}+\rho_{UDP})$=50%, $\rho_{TCP-long}/(\rho_{TCP-long}+\rho_{TCP-short})$=80%, λ_{UDP}=1660 packets/second, $\lambda_{TCP-long}$= $\lambda_{TCP-short}$= 8.3 bursts/second, $N_{TCP-short}$=10 and $B_{TCP-short}$=4 packets. $N_{TCP-long}$ and $B_{TCP-long}$ are varied. As we increase the ratio $N_{TCP-short}/N_{TCP-long}$, we are actually reducing the number of long-lived TCP flows while increasing their burst size to keep $\rho_{TCP-long}$ constant.

(2) *Low Load:* The same parameters as high load, except ρ_{Total}=1, μ_{Total}= 3320 packets/second, K=200 packets.

The load offered by long-lived TCP is chosen to be 80% of total TCP load. This percentage reflects the fact that only a small percentage of flows are long-lived, however they carry the majority of total traffic bytes in the Internet [16].

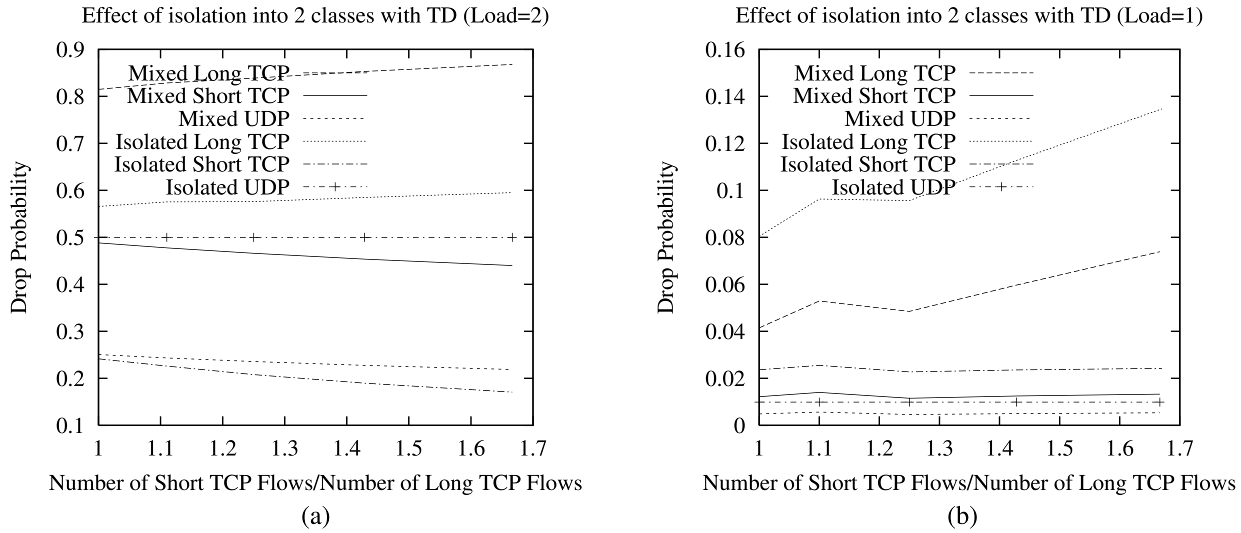


Figure 10. Effects of isolating TCP and UDP flows into 2 classes: UDP and TCP (with Tail Drop): (a) High Load (b) Low Load.

4.3.2 Results

The results are shown in Figures 10 and 11 for Tail-drop and RED, respectively.

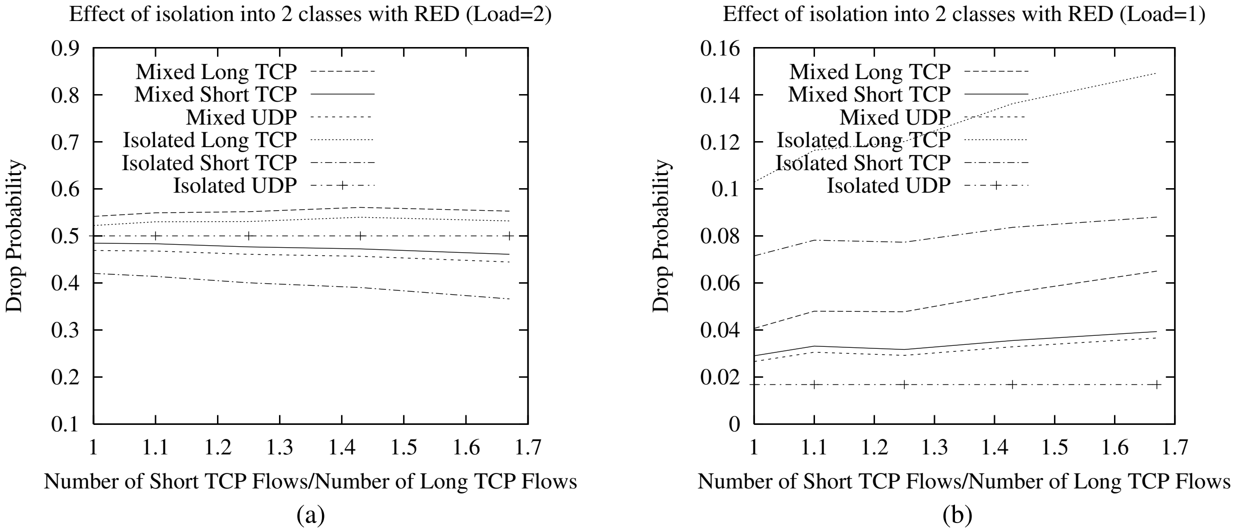


Figure 11. Effects of isolating TCP and UDP flows into 2 classes: UDP and TCP (with RED): (a) High Load (b) Low Load.

In the case of high load, when different kinds of TCP flows (short- and long-lived) and UDP share a Tail-drop router, isolating them into two classes helps both kinds of TCP flows. The improvement is very significant for short-lived TCP flows, which typically carry interactive/delay-sensitive data. However, drop probability of UDP increases when it is isolated. This means that UDP is taking advantage of both short- and long-lived TCP flows when they are mixed. Another interesting observation is when long and short TCP flows are in the same class, drop probability of short TCP flows decreases at the expense of increased drop probability for long TCP flows, and this unfairness increases as burst size of long TCP flows increases. This can be seen also from the fairness index plotted for the isolated TCP class in Figure 12. From this perspective, it seems that long TCP flows are the only victim of aggregation, and there is no way for short TCP flows to be shut off by long TCP flows. This is only because the model does not reflect the fact that when a packet is dropped, short TCP flows would back off and maybe wait for a timeout period to send again. With RED, results are similar to the results obtained by Tail-drop, however as usual, unfairness is small (see Figure 12). Overall, isolation improves fairness for TCP flows and their performance.

In the case of low load, isolation increases drop probability of TCP flows and does not improve fairness, since we are losing statistical multiplexing gains obtained by sharing.

4.4 Effects of Isolation into 3 Classes

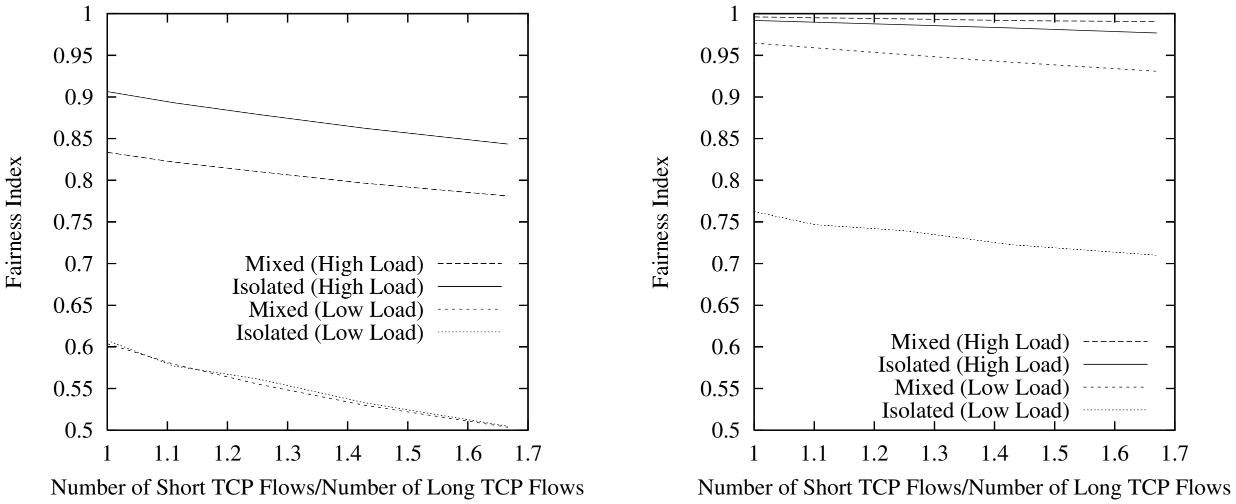
4.4.1 Experimental Setup

All the parameters are the same as in Section 4.3. However, in this experiment we examine the effects of isolating flows into 3 classes instead of 2, which are UDP, long TCP, and short TCP. By doing this, we expect to have better fairness values for isolated TCP flows.

4.4.2 Results

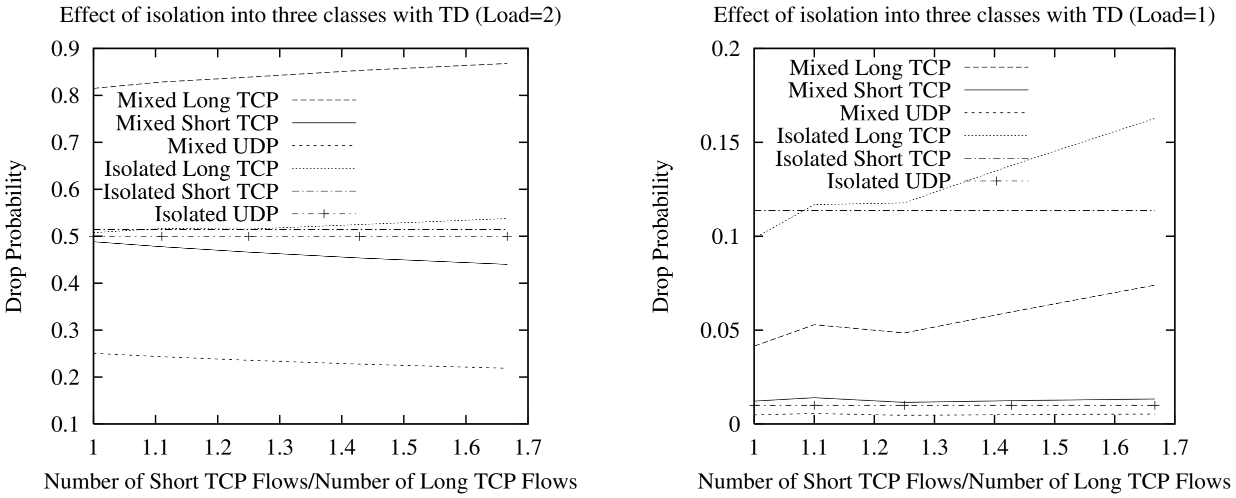
The results are shown in Figures 13 and 14 for Tail-drop and RED, respectively.

In the case of high load, isolation provides better and predictable drop probability for all classes. UDP no longer hurts TCP flows, and short-lived TCP flows no longer hurt long-lived TCP flows. The fairness index for these scenarios is shown in Figure 15. In isolation, we observe perfect fairness (=1) across flow types, which means short- and long-lived TCP flows should be further isolated to prevent negative effects on each other. Note that shared RED also provides fairness index very



(a) (b)

Figure 12. Fairness Index for isolation into 2 classes: UDP and TCP: (a) Tail Drop (b) RED



(a) (b)

Figure 13. Effects of isolation into 3 classes: Long-lived TCP, short-lived TCP and UDP (with Tail Drop): (a) High Load (b) Low Load.

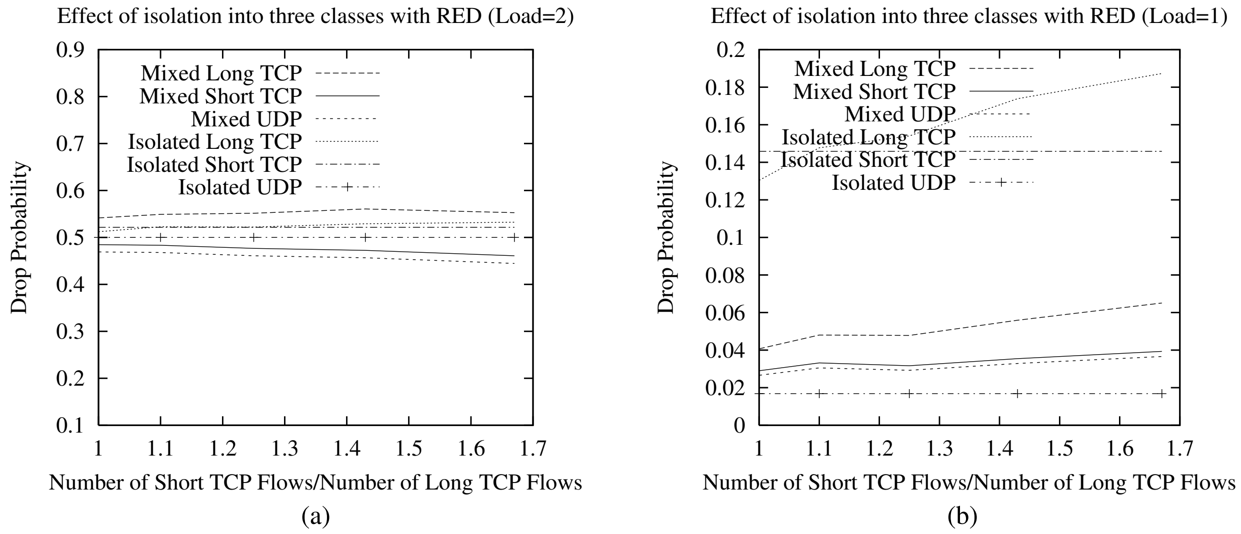


Figure 14. Effects of isolation into 3 classes: Long-lived TCP, short-lived TCP and UDP (with RED): (a) High Load (b) Low Load.

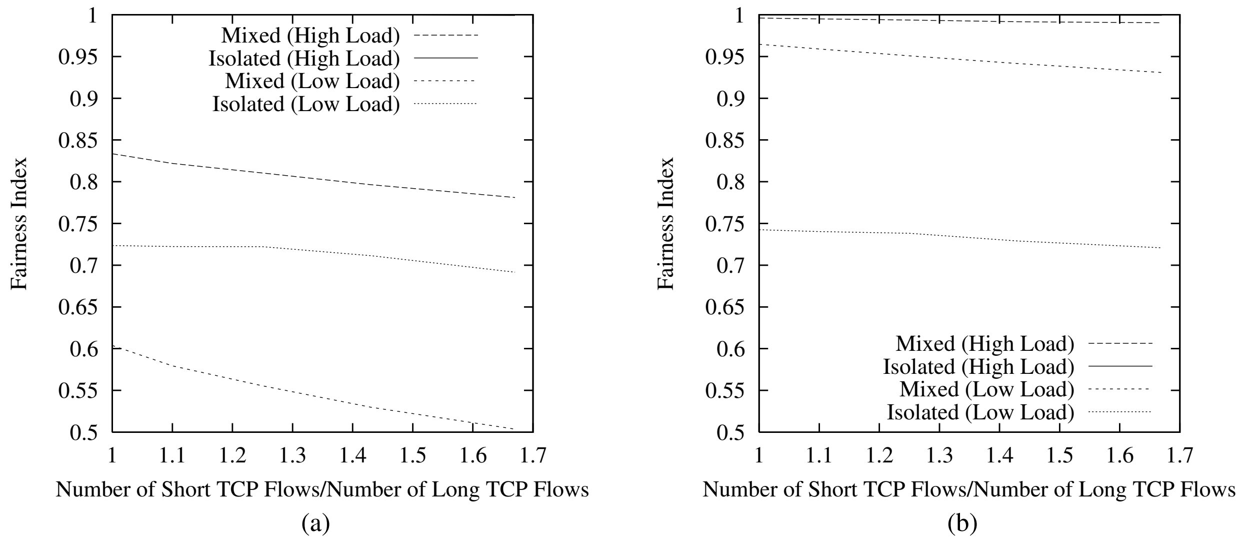


Figure 15. Fairness Index for isolation into 3 classes: (a) Tail Drop (b) RED

close to 1. However, isolated queues would provide better predictability for UDP (which mostly carries multi-media traffic) and short-lived TCP even in the presence of large bursts from long-lived TCP flows. Also, by having a separate queue for each flow type, it is easier to manage quality of service. To further improve quality of service of short-lived TCP flows, which are usually interactive/delay-sensitive and tend to timeout on packet drops, we can just allocate more resources to the class of short TCP flows.

In the case of low load, isolation increases drop probability of TCP flows for both Tail-drop and RED. The fairness index for Tail-drop increases with isolation, however for RED it decreases. As we observed in the previous experiments, we are losing statistical multiplexing gains compared to sharing since we model static partitioning of resources among classes. In reality, a class-based queuing discipline would allow one class to borrow resources from another under-utilized class [8].

4.5 Effects of Isolation When There is Only Short-lived and Long-lived TCP Flows

4.5.1 Experimental Setup

The goal of this experiment is to see the effects of isolation into 2 classes, namely long TCP and short TCP when traffic is aggregate of only long and short TCP flows. We consider two scenarios:

(1) *High Load:* ρ_{Total}=2, μ_{Total}=830 packets/second, K=50 packets, $\rho_{TCP-long}$ / ($\rho_{TCP-long} + \rho_{TCP-short}$) = 80%, $\lambda_{TCP-long}$ = $\lambda_{TCP-short}$ = 8.3 bursts/second, $N_{TCP-short}$=10 and $B_{TCP-short}$=4 packets. $N_{TCP-long}$ and $B_{TCP-long}$ are varied.

(2) *Low Load:* The same parameters as in the high load case, except ρ_{Total}=1, μ_{Total}= 1660 packets/second, K=100 packets.

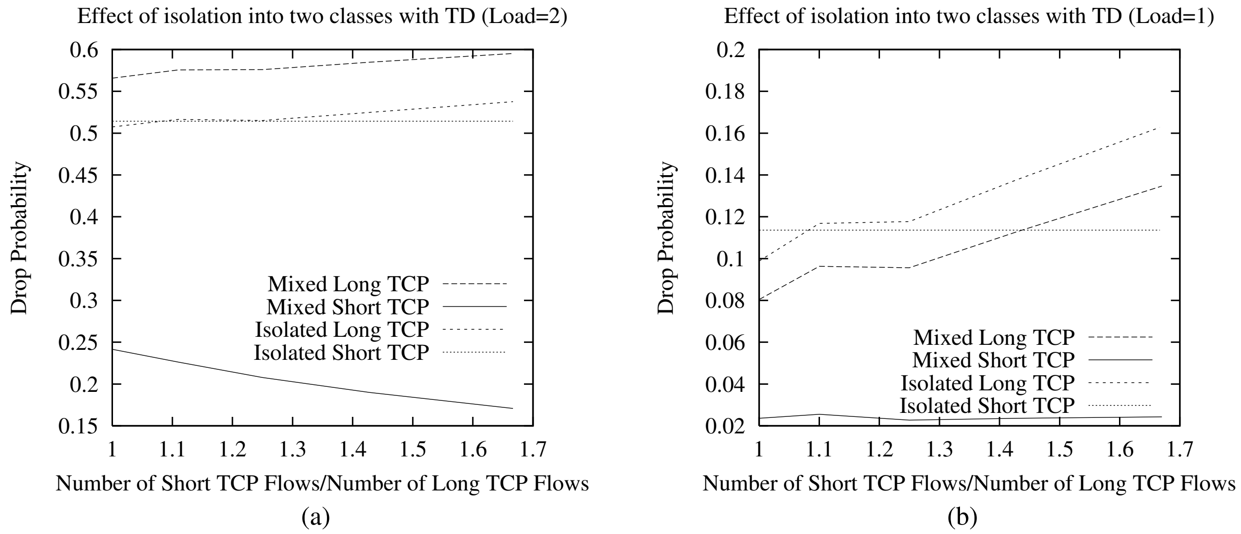


Figure 16. Effects of isolating TCP flows into 2 classes: Long-lived TCP, short-lived TCP (with Tail Drop): (a) High Load (b) Low Load.

4.5.2 Results

The results are shown in Figures 16 and 17 for Tail-drop and RED, respectively.

At high load, with isolation, we observe improvement in fairness and predictability for both short- and long-lived flows. Also performance of long-lived TCP flows is improved. The result is true for both Tail-drop and RED buffer management

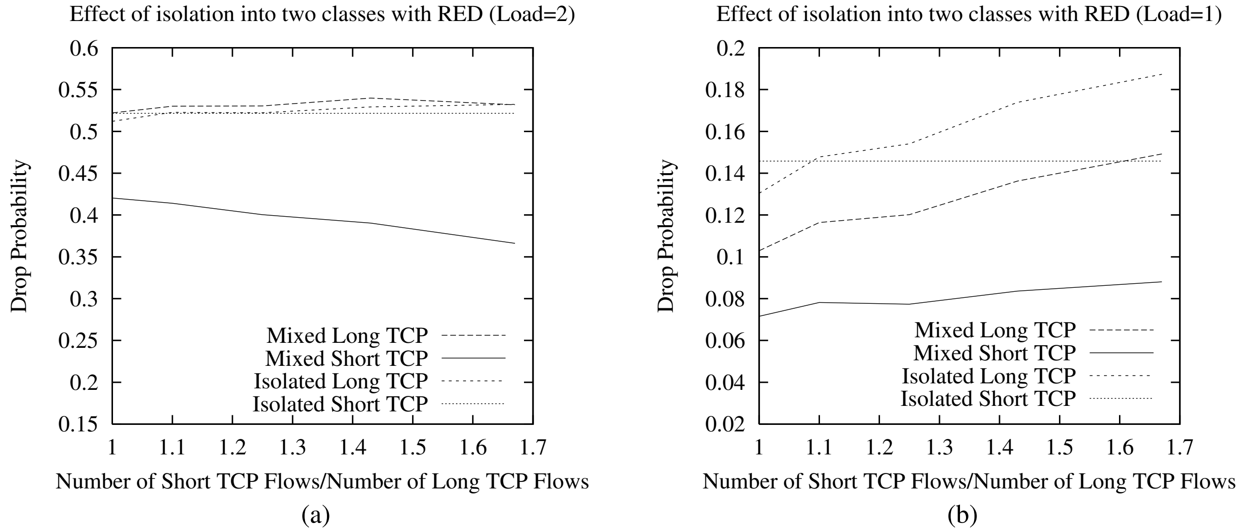


Figure 17. Effects of isolating TCP flows into 2 classes: Long-lived TCP, short-lived TCP (with RED): (a) High Load (b) Low Load.

schemes at high load. When short- and long-lived TCP flows share the same queue, short-lived TCP connections take advantage of long-lived TCP connections. Unfairness increases with increasing burst size of long-lived TCP connections. Isolation improves fairness as can be seen from Figure 18. The reason is that short-lived flows inject constant number of packets at the same rate. However, as $N_{TCP-long}$ gets smaller, long-lived flows send less frequent but bigger bursts. When long-lived flows do send to a shared queue, they are likely to see full buffer which is filled up with packets of short-lived flows. By isolation, we also believe that unseen negative effect of long flows on short flows (as explained in Section 4.3) would also be prevented. As a result, isolation would provide a better environment for all kinds of flows.

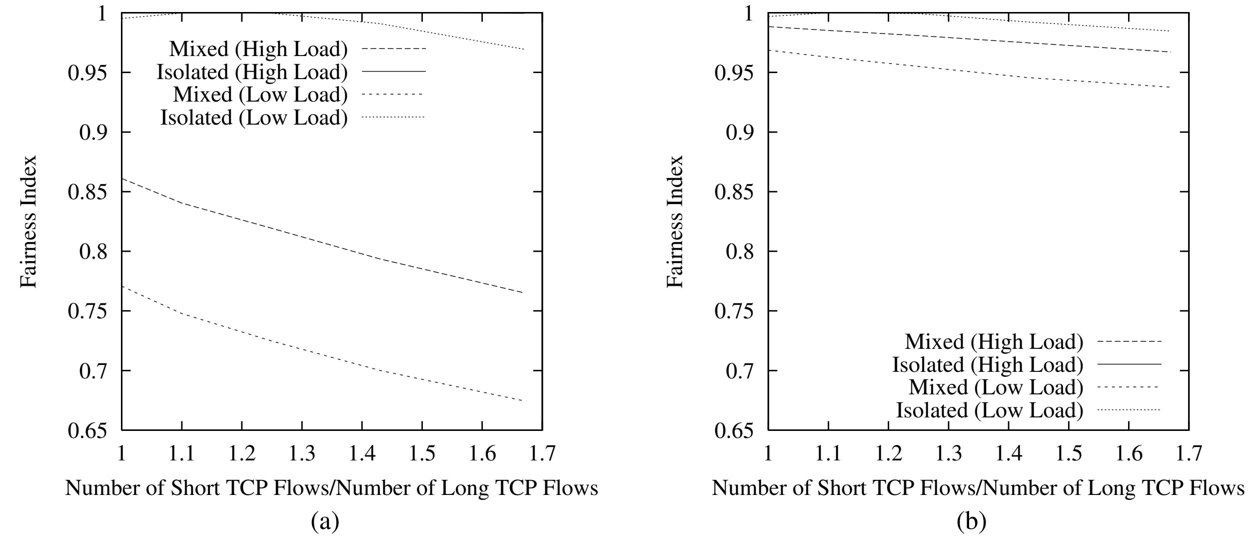


Figure 18. Fairness Index for isolation of TCP flows into 2 classes: (a) Tail Drop (b) RED

At low load, isolation improves fairness for both Tail-drop and RED at the expense of increasing drop probability for both kinds of flows.

5 Related Work

In order to address the problem of non-responsive flows, a number of studies [11, 12, 7] propose detecting non-responsive flows and limiting their rates so that they do not impact the performance of responsive flows. Others [5] suggest isolation to minimize interaction between connections in the context of Intserv per-flow architectures. Our concern here is on a simple, scalable, and less costly solution in the context of Diffserv architectures, where per-flow information is only maintained at access routers to classify packets. Core routers implement a simple class-based isolation scheme.

In the same context, [18] addresses fairness issues to solve the problem of unfriendly flows. However, their method is computationally more expensive than the one we are proposing here. In [15], Pieda *et al.* study TCP and UDP interaction using different drop preference mapping when they share the same Assured Forwarding (AF) PHB class and conclude that 3-drop preference mapping is not enough to solve fairness issues. In [14], Nandy *et al.* suggest using intelligent traffic conditioners to map UDP and TCP into different AF class queues. They observe that isolation solves fairness issues among TCP and UDP. Our work here not only examines the interaction of TCP and UDP flows, but also concentrates on short- and long-lived TCP flows. Our focus has been on isolating traffic with different characteristics (UDP, short- and long-lived TCP flows) to improve predictability of the service, fairness among TCP flows of different size, and better control over QoS of a particular type of traffic.

In [1], Bhaniramka *et al.* study isolation of TCP and UDP traffic over MPLS. Again this study does not consider how different (short- and long-lived) TCP flows are affected by non-responsive flows and effects of isolation in such environments. Our study extends our work in [13] to include non-responsive flows and analytically evaluate the benefits of isolation.

6 Conclusion and Future Work

Class based-isolation provides better fairness among different types of traffic in high-load environments. Isolation of UDP and TCP improves fairness for TCP flows and reduces their drop probability. This improvement is independent of the buffer management scheme and is pronounced for short-lived TCP flows, which usually carry interactive/delay-sensitive data and more likely to timeout on packet drops. Further isolation of TCP into short and long classes improves fairness and predictability among different types of TCP flows.

Isolation is a necessary and simple alternative if the deployed buffer management scheme is shared Tail-drop. With shared RED, we see similar results at lower scale. However, RED (even when average queue size is not used) helps to reduce bias against bursty traffic, and performs well enough to provide fairness. We expect that the results for RED would be better if average queue length was used. Even if this would be the case, we still believe isolation has other merits that RED cannot provide: Better control over quality of service of each flow type (class). To provide better quality for a particular type of traffic, we can just allocate more resources to that class. This is particularly important for interactive/delay-sensitive traffic such as short-lived TCP flows. At low-load environments, isolation must implement dynamic resource allocation (as in CBQ [8]) to avoid statistical loss.

As isolation provides better fairness and performance improvement for TCP flows, UDP flows generally experience increased drop probability. However, isolation limits the delay and jitter that UDP flows experience. Thus isolation improves service predictability for UDP flows usually carrying real-time data.

Finally our results suggest isolated queues for better fairness, performance and QoS management across different flow types, while using RED within each class to improve intra-class fairness by preventing synchronization of TCP flows. For future work, we intend to look at fairness within each class, where flows do not have the exact same characteristics. The goal is to identify the minimum number of classes for a mix of TCP flows with various lifetimes/window sizes and round-trip times.

Acknowledgement

Thanks to T. Bonald for his clarifications on the model in [2].

References

[1] P. Bhaniramka, W. Sun, R. Jain, “Quality of Service using Traffic Engineering over MPLS: An Analysis”, GlobeCom 99, March 1999.

[2] T. Bonald, M. May, J. Bolot, “Analytic evaluation of RED Performance”, INFOCOM 2000.

[3] N. Cardwell, S. Savage, T. Anderson, “Modeling TCP Latency”, *IEEE/INFOCOM’2000*, Tel-Aviv, Israel, March 2000.

[4] D.M. Chiu and R. Jain, “Analysis of Increase Decrease Algorithmsfor Congestion Avoidance in Computer Networks”, *Computer Networks and ISDN Systems*, pp. 1-14, 1989.

[5] A. Demers, S. Keshav, and S. Shenker, “Analysis and Simulation of a Fair Queuing Algorithm”, *Internetworking: Research and Experience*, 1:3-26, 1990.

[6] K. Fall, S. Floyd, “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP”, Computer Communication Review, V. 26 N. 3, July 1996, pp. 5-21.

[7] W. Feng, D. Kandlur, D. Saha, K. Shin, ”Blue: A New Class of Active Queue Management Algorithms”, U. Michigan CSE-TR-387-99, April 1999.

[8] S. Floyd, V. Jacobson, “Link-sharing and Resource Management Models for Packet Networks”, IEEE/ACM Transactions on Networking, pp. 365-386, August 1995.

[9] S. Floyd, V. Jacobson, “On traffic Phase Effects in Packet-Switched Gateways”, *Internetworking: Research and Experience*, 3(3):115-156, September 1992.

[10] S. Floyd, V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, *IEEE/ACM Trans. on Networking*, vol. 1, pp. 397-413, 1993.

[11] S. Floyd, K. Fall, ”Promoting the Use of End-to-End Congestion Control in the Internet”, IEEE/ACM Transactions on Networking, August 1999.

[12] D. Lin, R. Morris, “Dynamics of Random Early Detection”, *ACM/SIGCOMM’97*, 1997.

[13] I. Matta, L. Guo, “Differentiated Predictive Fair Service for TCP Flows”, In Proceedings of ICNP’2000: The 8th IEEE International Conference on Network Protocols, Osaka, Japan, October 2000.

[14] B. Nandy, N. Seddigh, P. Pieda, N. Ethridge, “Intelligent Traffic Conditioners for Assured Forwarding Based Differentiated Services Networks”, IFIP High Performance Networking (HPN 2000), Paris, June 2000.

[15] P. Pieda, N. Seddigh, B. Nandy, “Dynamics of TCP and UDP Interaction in QoS Differentiated Services Networks”, 3rd Canadian Conference on Broadband Research, November 1999.

[16] A. Shaikh, J. Rexford, and K.G. Shin, “Load-Sensitive Routing of Long-Lived IP Flows”, in *SIGCOMM’99*, Boston, MA, September 1999.

[17] S. Shenker, “Fundamental Design Issues for the Future Internet”, IEEE Journal on Selected Areas in Communications, Vol. 13, No.7, pp. 1176-1188, September 1995.

[18] I. Stoica, S. Shenker, H. Zhang, ”Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks”, Proceedings of SIGCOMM’98, Vancouver, Canada, pp. 118-130.

[19] I. Stoica, H. Zhang, “LIRA: An Approach for Service Differentiation in the Internet”, NOSSDAV’98, London, UK, July 1998.