WPI-CS-TR-04-10 April 2004

Traffic Sensitive Active Queue Management

by

Mark Claypool
Robert Kinicki
and Abhishek Kumar

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

Traffic Sensitive Active Queue Management

Mark Claypool, Robert Kincki, and Abhishek Kumar
{claypool,rek}@cs.wpi.edu
CS Dept., Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609, USA

Abstract

Internet applications have varied Quality of Service (QoS) Requirements. Traditional applications such as FTP and email are throughput sensitive since their quality is primarily affected by the throughput they receive. There are delay sensitive applications such as streaming audio/video and IP telephony, whose quality is more affected by the delay. The current Internet however does not provide QoS support to the applications and treats the packets from all applications as primarily throughput sensitive. Delay sensitive applications can however sacrifice throughput for delay to obtain better quality. We present a Traffic Sensitive QoS controller (TSQ) which can be used in conjunction with many existing Active Queue Management (AQM) techniques at the router. The applications inform the TSQ enabled router about their delay sensitivity by embedding a delay hint in the packet header. The delay hint is a measure of an application's delay sensitivity. The TSQ router on receiving packets provides a lower queuing delay to packets from delay sensitive applications based on the delay hint. It also increases the drop probability of such applications thus decreasing their throughput and preventing any unfair advantage over throughput sensitive applications. We have also presented the quality metrics of some typical Internet applications in terms of delay and throughput. The applications are free to choose their delay hints based on the quality they receive. We evaluated TSQ in conjunction with the PI-controller AQM over the Network Simulator (NS-2). We have presented our results showing the improvement in QoS of applications due to the presence of TSQ.

1 Introduction

The Internet today carries traffic for applications with a wide range of delay and loss requirements. Traditional applications such as FTP and E-mail are primarily concerned with throughput, while Web traffic is moderately sensitive to delay as well as throughput. Emerging applications such as IP telephony, video conferencing and networked games have different requirements in terms of throughput and delay than these traditional applications. In particular interactive multimedia applications, unlike traditional applications, have more stringent delay constraints than loss constraints. Moreover, with the use of repair techniques [BFPT99, PHH98, LC00] packet losses can be partially or fully concealed, enabling multimedia applications to operate over a wide range of losses, and leaving end-to-end delays as the major impediment to acceptable quality.

Unfortunately, the current Internet does not support per application QoS. Instead all applications are treated primarily as throughput sensitive and no attempt is made to provide a lower delay to applications that desire it. Every packet arriving at a router is enqueued at the tail, thus providing the same average delay to all applications. When there is persistent congestion, the router queue builds up and eventually packets have to be dropped. A large queue build-up causes high queuing delays for all applications, regardless of their delay sensitivity.

However, if the router is capable of providing QoS support, then it could treat packets from delay-sensitive applications differently than those from throughput-sensitive applications. Since the delay-sensitive applications are loss-tolerant, the

router can try to provide them with a lower delay and approximately decrease the throughput provided to them. The loss of throughput may not decrease the overall quality of the delay-sensitive applications very significantly, but the reduction in delay can cause a significant improvement in quality. The throughput gained can be allocated to the throughput-sensitive applications, thus providing them with higher quality.

ABE [HKBT01] provides a queue management mechanism for low delay traffic. ABE allows delay-sensitive applications to sacrifice throughput for lower delays. ABE, however, rigidly classifies all applications as either delay-sensitive or throughput-sensitive. Thus applications are not able to choose relative degrees of sensitivity to throughput and delay. Approaches such as CBT [PJS99] and [NT02] provide class-based approach and with bitrate guarantees for different classes. However, these fixed and pre-determined classes are not sufficient to represent the varying QoS requirements of applications within one particular class. Similarly, DCBT with ChIPS [CC00], which extends CBT by providing dynamic thresholds and lower jitter for multimedia traffic, still limits all multimedia traffic to the same QoS.

DiffServ approaches, such as Assured Forwarding (AF) [HBWW99] and Expedited Forward (EF) [JNP99], try to give differentiated service to traffic aggregates. However the DiffServ architectures are very complicated and require the presence of traffic monitors, markers, classifiers, traffic shapers and droppers to enable the components to work together. IntServ [SBC94] provides the best possible per flow QoS guarantees. However, it requires complex signaling and reservations via RSVP by all routers along a connection on a per-flow basis, making scalability difficult for global deployment.

We present a new QoS controller called the Traffic Sensitive QoS Controller (TSQ), that provides a congested Internet router with per packet QoS support based on an application's delay sensitivity. Unlike approaches that provide fixed classes of service, each application sending traffic into the TSQ router chooses a customized delay-throughput trade-off based on its own requirements. The service is still best-effort in that it requires no additional policing mechanisms, charging mechanisms or usage control. With TSQ, applications mark each packet with a *delay hint* indicating the relative importance of delay versus throughput. The TSQ router will, on receipt of each packet, examine its delay hint and calculate an appropriate queue position where the packet is to be inserted. A packet from an application which has a low value of delay hint will be allowed to "cut-in-line" towards the front of the queue, while a packet from an application with a high value of delay hint will be inserted towards the end of the queue. To prevent delay-sensitive applications from gaining an unfair advantage over the throughput-sensitive applications, TSQ proportionately increases the drop probability of the packets inserted into the queue. The more a packet attempts to cut-in-line, the more the packet's drop probability is increased. Thus, throughput-sensitive applications mark their packets with high values of delay hints, and hence they are not cut-in-line and do they have their drop probability increased, thus providing them with good quality. TSQ requires no per-flow state information, no traffic monitoring, and no edge policing or marking.

TSQ can be used in conjunction with most AQMs that provide an aggregate drop probability, for example RED [FJ93], Blue [FKSS01], PI [HMTG01], and SFC [GH03]. We have evaluated the performance of TSQ when used in conjunction with the PI-controller (Proportional Integral controller) AQM [HMTG01] with varying mixes of delay-sensitive and throughput-sensitive flows. In order to quantify an application's QoS, we propose a QoS metric based on the minimum of an application's delay quality and throughput quality. Based on recommended application performance requirements, we provide quality metrics for Internet applications that cover a range of QoS and throughput sensitivities: interactive audio, interactive video and file transfer. Using TSQ, applications can use the knowledge of their QoS requirements to dynamically choose their delay hints so as to maximize their Quality of Service. Evaluation results suggest that TSQ with PI provides better quality for all applications than does PI by itself.

The remainder of the paper is organized as follows: Section 2 presents quality metrics we have

devised for fundamental Internet applications; Section 3 discusses the TSQ mechanism; Section 4 describes experiments and analysis of TSQ; and Sections 5 and 6 summarizes our work and discuss the possible future work.

2 Application Quality Metrics

In this section chapter we develop quality metrics for three network applications: interactive audio (Section 2.1), such as used in IP telephony, interactive video (Section 2.2), such as used in a video conference and file transfer applications (Section 2.3) such as used in peer-to-peer file systems or FTP. The quality metrics can be used to quantify application performance, allowing us to evaluate the impact of TSQ on QoS. In addition, the quality metrics could be used by end-host applications to adjust the delay hint it provides to a TSQ enabled network in order to improve overall performance.

Based on information from previous work [Gan02, IKK93, DCJ93, Zeb93], we have devised quality functions for these three applications in terms of their network delay and the network throughput called the *delay quality* (Q_d) and *throughput quality* (Q_t), respectively. We define the overall quality of the application as the minimum of the two quality metrics:

$$Q(d,T) = min(Q_d(d), Q_t(T)) \quad (1)$$

The value of $Q(d,T)$ lies between 0 and 1, where a quality of 1 represents the maximum quality that the application can receive, and a quality of 0 represents performance that is of no use to the application at all.

2.1 Audio Conference Quality

In this section we discuss the quality functions that we have derived for audio conference applications. The quality functions are of two types, the delay quality function and the throughput quality function. We have graphed the quality functions for the application versus one-way delay and throughput respectively.

2.1.1 Effect of Delay on Audio Conference Quality

Audio conference applications are relatively sensitive to increased delays but less sensitive to reduced throughput. [Gan02] suggests that audio conference quality in terms of delay is essentially divided into 3 parts. A one-way delay of 150 ms or less means excellent quality, a one-way delay of 150-400 ms means good quality, and a one-way delay in excess of 400 ms is poor quality. Also, [IKK93] has observed the variation of audio quality with delay in terms of Mean Opinion Scores (MOS scores). Figure 1 from [IKK93] shows the variation of MOS scores for free conversation with round-trip delay.

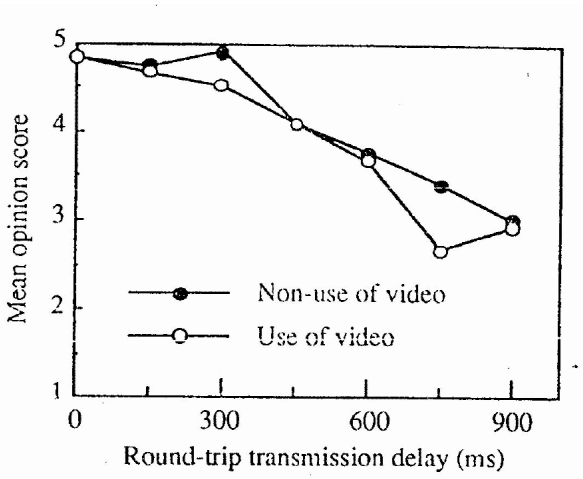

Figure 1: Mean Opinion Scores versus Round-Trip Delay

Based on this previous work, we have produced the graph in Figure 2 depicting the delay quality of an audio conference application. The best quality possible is 1 (equivalent to a MOS of 5) when there is a zero delay. The audio application has an excellent quality if the one way delay is within 150 ms. As delay increases, the initial decrease in quality is not significant, and a delay of 150 ms provides the application with a quality of 0.98. However, as the delay increases above 150 ms, the drop in quality becomes significant, with a delay of 300 ms reducing quality to 0.7 (equivalent to a MOS score of 3.5) and to 0.5 (equivalent to a MOS score of 3) when delay is 400 ms. As the delay increases higher than 400 ms, we propose that the degradation is about twice the degradation in quality from

150 to 400 ms delay. Thus, from the graph we can see the three broad sections of quality described in [Gan02] and also get quantitative values of the quality for intermediate one-way delays. The set of equations governing the delay quality of an audio conference application are as follows:

$$
\begin{aligned}
Q_d(d) = &\quad -0.00133 \times d + 1 && d \leq 150 \\
Q_d(d) = &\quad -0.00192 \times d + 1.268 && 150 \leq d \leq 400 \\
Q_d(d) = &\quad -0.004 \times d + 2.1 && 400 \leq d \leq 525 \\
Q_d(d) = &\quad 0 && 525 \leq d
\end{aligned}
$$

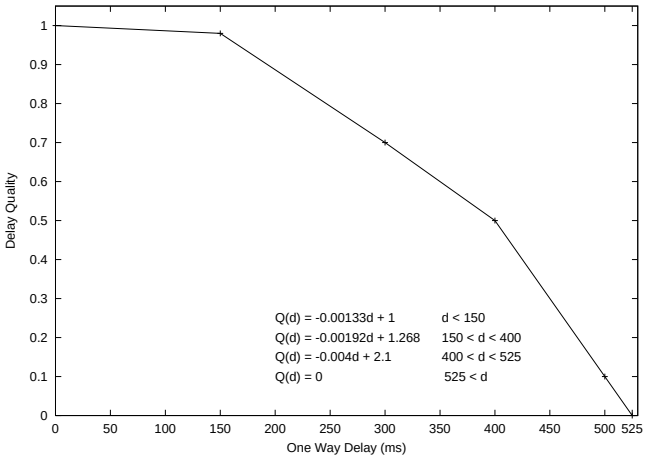

Figure 2: Delay Quality for Audio Conference versus One-Way Delay

2.1.2 Effect of Throughput on Audio Conference Quality

Figure 3 depicts the quality for an audio conference application versus the throughput that the application receives (the *throughput quality*). The application has a throughput quality of 1 when the throughput is 128 Kbps, since at this bit-rate the quality of audio is of CD quality, which we assign as the best possible. The throughput quality decreases linearly as the throughput is halved since every time one fewer bit is used to encode the audio, the throughput of the audio codec is reduced by half. We assume that the quality of the audio application reduces linearly with the reduction in the number of encoding bits. Hence the variation of audio quality with throughput is a logarithmic curve, where a reduction in throughput above 64

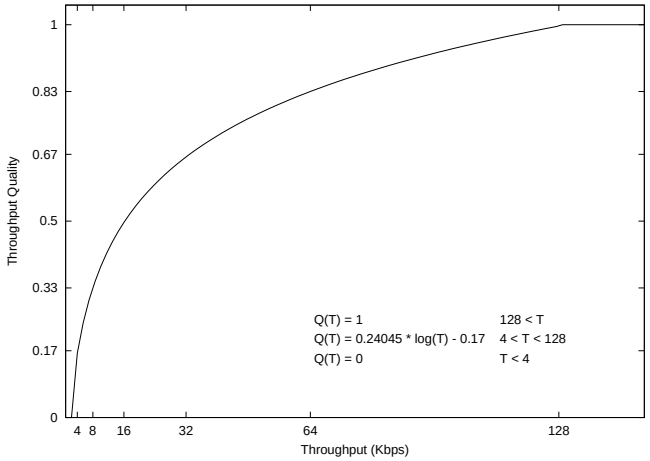

Figure 3: Throughput Quality for Audio Conference versus Throughput

Kbps does not greatly reduce the quality of the application, while a reduction in throughput below 64 Kbps does. The throughput quality is 1 for 128 Kbps throughput, decreases to 0.83 for 64 Kbps and falls further to 0, when the throughput is 2 Kbps, appropriate since 4 Kbps is the lowest codec rate available for audio application [Cor98]. The set of equations for the throughput quality are as follows:

$$
\begin{aligned}
Q_t(T) = &\quad 1 && 128 \leq T \\
Q_t(T) = &\quad 0.24045 \times log(T) - 0.17 && 4 \leq T \leq 128 \\
Q_t(T) = &\quad 0 && T \leq 4
\end{aligned}
$$

2.2 Video Conference Quality

As another representative delay sensitive application but with alternate throughput sensitivities, we derived quality metrics for an interactive video application, specifically a typical H.323 video conference.

2.2.1 Effect of Delay on Videoconference Quality

Since the nature of the interactivity of a video conference is the same as that in an audio conference, the delay requirements of a video conference are similar to those of an audio conference application described in Section 2.1.1. Hence the plot in Figure 2 and the formulas for audio conference delay

quality suggested in Section 2.1.1 also apply to delay quality of a video conference.

2.2.2 Effect of Throughput on Video Conference Quality

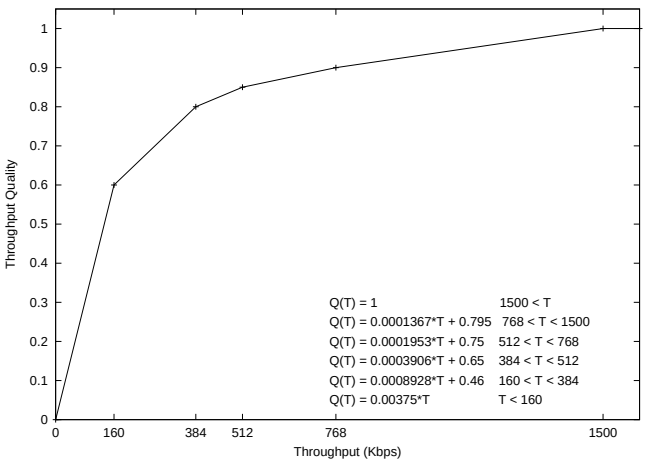

Figure 4: Delay Quality for Video Conference versus Throughput

Typically, an H.323 video conference requires a bitrate of 384 Kbps for good quality [Cor00]. If the application receives this throughput we assign it a quality of 0.8 (derived from the MOS scale, where a score of 4 on a scale of 1-5 is considered good). As the throughput provided to the application increases, the quality of the application increases, but in a smaller proportion. Thus, the quality increases to 0.85 when throughput is 512 Kbps, and to 0.9 when the throughput is 768 Kbps. A videoconference gets its best quality of 1 when the throughput is 1.5 Mbps based on the specification that a H.323 video conference operating at 1.5 Mbps is of excellent quality [Cor00]. Any subsequent increase in the throughput does not improve the quality. An H.323 video conference has average quality if it has a throughput of 160 Kbps. Thus, we assign this throughput a quality value of 0.6 corresponding to a MOS score of 3 which is considered as "fair" quality. Any further reduction in throughput will cause the quality to fall off sharply. We thus come up with the following set of equations which determine the throughput quality for the video application (and depicted in Figure 4):

$$
\begin{aligned}
Q_t(T) = &\quad 1 && 1500 < T \\
Q_t(T) = &\quad 0.0001367 \times T + 0.795 && 768 \leq T \leq 1500 \\
Q_t(T) = &\quad 0.0001953 \times T + 0.75 && 512 \leq T \leq 768 \\
Q_t(T) = &\quad 0.0003906 \times T + 0.65 && 384 \leq T \leq 512 \\
Q_t(T) = &\quad 0.0008928 \times T + 0.46 && 160 \leq T \leq 384 \\
Q_t(T) = &\quad 0.00375 \times T && T \leq 160
\end{aligned}
$$

2.3 File Transfer Quality

In this section we discuss the quality metrics we used to measure the quality of file transfer applications. File transfer applications, unlike the audio conference and video conference applications, are not delay sensitive (relative to router queuing delays). Instead, the quality of these applications is almost entirely dependent on their throughput.

2.3.1 Effect of Delay on File Transfer Quality

A file transfer application's quality will degrade only if the delay increases on the order of tens of seconds, which is well beyond the scope of router queuing delays. Since, in our experiments, the delay is generally on order of few 100 ms, we ignore the effect of delay on FTP quality beyond 1000 ms. The delay quality of a file transfer application is as follows:

$$Q_d(d) = \quad 1 \quad d \leq 1000$$

2.3.2 Effect of Throughput on File Transfer Quality

The quality of a file transfer application depends almost entirely on the throughput that it can get from the network. In our quality metrics, the quality requirements of a file transfer is dependent upon the size of the file that it is transferring. A small file will require a lower throughput to attain good quality as compared to a very large file. We propose that a file transfer application has maximum quality if it can finish transferring a file in 1 second. Thus for 10 Mb file, a quality of 1 is attained from a throughput of 10 Mbps. If the throughput obtained is greater, the quality does not improve, while a decrease in quality is directly proportional to a decrease in throughput. Similarly for a smaller file of 10 Kb, the required throughput for best quality is

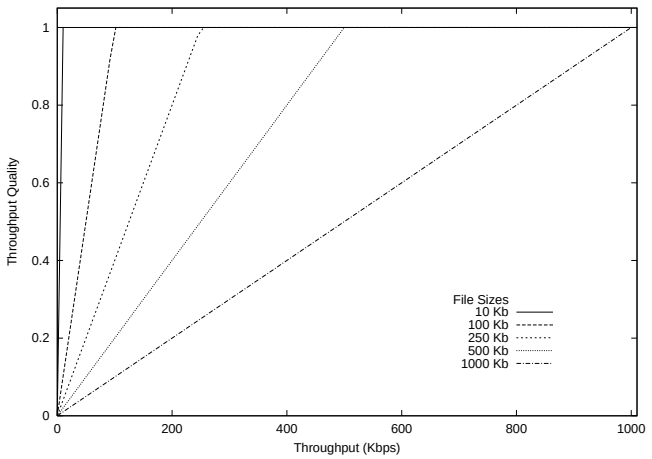

Figure 5: Throughput Quality of File Transfer Application versus Throughput

only 10 Kbps. We derive the following equation for throughput quality of file transfer applications:

$$Q_t(T, S) = \quad T/S$$

where S is the size of the file. Figure 5 depicts quality graphs for file transfer applications with various file sizes.

3 Traffic Sensitive QoS Controller

The Traffic Sensitive QoS controller (TSQ) provides Quality of Service when used in conjunction with most existing Active Queue Management (AQM) mechanisms. TSQ accommodates delay sensitive applications, such as interactive multimedia, by providing a low queuing delay, while at the same time not penalizing the throughput of the traditional applications, such as file transfers. TSQ achieves this per-application QoS by providing a trade-off between queuing delays and drop probabilities. The applications inform TSQ about their delay sensitivity by providing a *delay hint*. A TSQ-enabled router provides flows with a low delay hint with a lower delay by using a "cut-in-line" mechanism. In order to avoid penalizing throughput-sensitive applications, TSQ adjusts the drop probability of a delay-sensitive packets based on the reduction in delay it provides to the packet.

In Section 3.1, we described how applications notify the TSQ router about their delay sensitivity by using a *delay hint*. In Section 3.2, we describe the "cut-in-line" mechanism which is used to provide delay sensitive applications with lower queuing delays. Section 3.3 discusses the adjustment in drop probability that is made for the delay-sensitive flows so that they do not get unfair advantage over throughput-sensitive flows. Section 3.4 concludes with a diagram and algorithm detailing TSQ.

3.1 Delay Hints

Applications wanting to use the benefits of TSQ need to provide the router with a measure of their sensitivity to delay. This is done by providing a *delay hint* (d) in the header of each IP packet, where a low delay hint means that the application requires a low network delay for good quality and a high delay hint means that the application is throughput-sensitive and does not require a low delay for good quality. Applications such as interactive multimedia and network games will typically provide low delay hints. On the other hand, applications such as file transfer will typically provide the highest delay hints.

Based on the discussion in [SZ99] there are 4 to 17 bits available in the IP header that can be used to carry hint information. In our current implementation of TSQ, the range of delay hints is from 1 to 16 requiring 4 bits in the packet header. Thus, an application which chooses the minimum delay hint of 1 will be extremely delay-sensitive, in contrast to an application which can tolerate some delay and hence will have the maximum delay hint of 16. If the number of bits used for the delay hints is increased, the applications will have more levels of delay-sensitivity to choose from, hence more accurately representing their QoS requirements, but at the cost of increased overhead in each packet header. Similarly if the number of bits used to represent delay hints is reduced, the applications will have a smaller range of delay-sensitivity to choose from, but less overhead per packet. The optimal number of bits for delay hints is left as future work.

3.2 Cut-in-Line

Typically routers use a FIFO queue to hold packets. Since all packets are enqueued at the end of the queue, all packets and therefore all applications receive the same queuing delay. The queuing delay obtained by each packet depends upon the current queue length (q) and the outgoing link capacity. TSQ provides delay-sensitive packets with a lower queuing delay by "cutting" packets in line according to their delay hints. A packet from a delay sensitive application with a low delay hint will generally be queued towards the front of the queue leading to a lower queuing delay for that packet. A packet from a throughput-sensitive application having a high delay hint will generally be enqueued towards the end of the queue. However queue insertion based solely on delay-hints may cause starvation of packets with high delay hints. For example, a packet with a high delay-hint at the end of the queue can be starved in the face of a large number of low delay-hint packets cutting in line at (or above) the link capacity in front of this packet. To avoid this, we introduce an aging mechanism to prevent starvation.

Th TSQ cut-in-line mechanism is implemented by using a weighted insertion into the queue. At the arrival time (t_a) of a packet, we calculate the queuing delay that the packet would experience if it was inserted at the end of the queue; we call this queuing delay the *drain time* (t_d) of the queue. TSQ calculates the packet weight (w) according to its delay hint and time of arrival at the queue.

$$w = \frac{d \times t_d}{2^n} + t_a \tag{2}$$

where n is the number of bits used to represent the delay hint (4 in our current implementation). The packets in the queue are inserted in order sorted by their weights, with the lower weight packets inserted towards the front of the queue and the higher weight packets inserted towards the end of the queue. The new position of the packet in the queue is referred to as q'. Thus, a high delay-hint will cause a packet to have a higher weight and hence a higher value of q', while a delay hint of 1 will cause a packet to have a $q' = q$. Newly arriving packets will have their weights slightly increased due to the effect of the time of arrival on their weight, thus preventing starvation of older packets.

This cut-in-line requires a weighted insertion that can be implemented using a probabilistic data structure such as skip lists [Pug90], giving complexity $O(log(q))$, where q is the number of packets in the queue.

3.3 Drop Probability

During congestion, many AQM techniques produce a drop probability (p)) which is applied to packets arriving at the router. All arriving packets are subject to the same drop probability, with packets that are randomly dropped not being inserted in the queue. However, in the case of the TSQ, a uniform drop probability for all packets will potentially result in a higher throughput for the delay-sensitive applications, since TSQ is providing a lower delay to its packets. Hence, TSQ increases the drop probability for packets with delay hints lower than the maximum (2^n, or 16 in our implementation). The increase in drop probability is related to the reduction in queuing delay that the packet would otherwise experience if it were inserted in the queue in the position calculated by the cut-in-line mechanism. Thus, for a packet from a throughput sensitive application which would otherwise be inserted at the end of the queue, the drop probability from the AQM technique is not increased, hence the application benefits from any throughput advantage provided by the underlying AQM.

To determine the appropriate drop probability of packets that have cut-in-line, TSQ starts with the steady state throughput T of a TCP flow in which throughput is inversely proportional to the queuing delay and the square root of the loss rate [PFTK98]:

$$T = \frac{K}{r \times \sqrt{p}} \tag{3}$$

where r is the round-trip time, p is the loss rate and K is a constant for all flows based on the network conditions. The round trip delay r is the sum of the queuing delay and the round-trip propagation delay. Since some packets can have a decreased queuing delay by cutting in line, we compensate by increasing the drop probability for those packets. Let the new queuing delay after TSQ be q',

the new drop probability be p', and the round-trip propagation delay be l. The throughput obtained by the flow will now be T':

$$T' = \frac{K}{(l+q') \times \sqrt{p'}} \quad (4)$$

We want to prevent the new throughput T' from being greater than the throughput obtained without TSQ, ($T' \leq T$). Hence, we calculate the new drop probability p' as:

$$p' = \frac{(l+q)^2 \times p}{(l+q')^2} \quad (5)$$

The value of p' depends on the new queue position value q' and the queue position q if TSQ were not present (in other words, the instantaneous queue length when the packet arrived). p' also depends on the one way propagation delay l of the network. Since it is difficult for the router to determine the one way propagation delay of every flow, we keep the value of l as a constant, but is typically between 40-100 ms for many Internet links [CPS02]. Setting l to lower values in this range will result in a more aggressive increase in drop probability, while setting l to higher values in this range will result in less aggressive increase in drop probability. For our experiments, we fixed the one way propagation delay constant for the router at 40 ms.[1]

3.4 Summary

Figure 6 summarizes the TSQ algorithm.

```
/* constants:
C - capacity of the link
l - network latency
n - number of bits used for delay hints
/
/* variables:
q - current length of queue
q' - position to inserted packet
w - packet weight
d - delay hint
t_d - drain time
t_a - packet arrival time
p - AQM drop probability
p' - drop probability after TSQ
/
on receiving packet pkt:

// Calculate its drain time
t_d = q/C

// Calculate packet weight
w = (d × t_d)/2^n + t_a

// Determine new position of packet in the queue
q' = weightedInsert(w,pkt)

// Calculate new drop probability
p' = (l+q)^2 × p / (l+q')^2

// Generate random number between 0 and 1
r = uniform[0,1]

if (r ≤ p') then
   drop(pkt)
else
   insertPacket(q', pkt)
```

Figure 6: TSQ Algorithm

4 Experiments

This chapter describes experiments to evaluate the Traffic Sensitive Quality of Service Mechanism (TSQ) over an existing Active Queue Management (AQM) technique, the PI-controller [HMTG01]. The PI-controller attempts to provide a steady queuing delay by keeping the queue size stable around a target queue length, adjusting the drop probability in response to the rate of incoming packets. Like many AQMs, PI provides an explicit drop probability required for TSQ.

We conducted a variety of experiments to test the effect of TSQ on the quality of audio conference, interactive video and file transfer flows, comparing performance with PI and TSQ to perfor-

[1] Note that this value is fixed for the TSQ router for all experiments although the experiments will be simulated on networks with different propagation delays.

mance with only PI. We also measured the variation in queuing delay and throughput for the audio and video flows to illustrate the basic effects of TSQ. Finally, we ran experiments to measure the effect of unresponsive flows when using TSQ in order to verify that non-responsive flows do not benefit from TSQ.

4.]

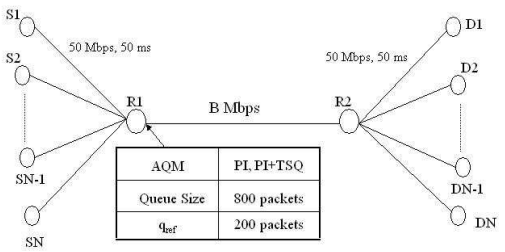

Figure 7: Network Topology

All implementation and experiments were done in the Network Simulator (NS-2).[2] Figure 7 shows the generic network topology for all the experiments in the simulation. There are N sources S1..SN and N destinations D1..DN. These N flows are connected to a single common link giving rise to a bottleneck at router R1. Each of the connections between the sources and the bottleneck node have a link capacity of 50 Mbps and propagation delay of 50 ms. Similar connections exist between the egress router (R2) and the destinations. The bottleneck link capacity is B Mbps. The one way propagation delay of the network is D ms. This bottleneck router runs PI [HMTG01] plus our implementation of the TSQ algorithm in Figure 6. PI is configured with the values recommended in [HMTG01]: $\alpha = 0.00001822$, $\beta = 0.00001816$, $\omega = 170$, $q_{ref} =$ 200 packets and $q_{max} = 800$ packets. The average packet size is 1000 bytes.

4.2 Audio Quality Evaluation

In this experiment we evaluate the performance of a single interactive audio flow sharing the network with other TCP based bulk file transfer flows.

[2]http://www.isi.edu/nsnam/ns/

4.2.1 Setup

The network topology is as described in Section 4.1 with the bottleneck link capacity B=15 Mbps and the one-way propagation delay D=50 ms providing one-way propagation delays between each of the sources and their respective destinations at 150 ms. The number of flows N=100, with 99 TCP based FTP bulk transfer flows that are not delay sensitive and so provide the maximum delay hint of 16, and 1 audio conference flow simulated as a TCP-friendly source sending data at a rate of 128 Kbps. We run the experiment for 100 seconds of simulation time, whereupon we e change the delay hint of the audio flow for the next run in order to evaluate the performance of the audio flow over a range of delay hints.

4.2.2 Analysis

We analyze the effect of different delay hints on the queuing delay and throughput of the audio flow.

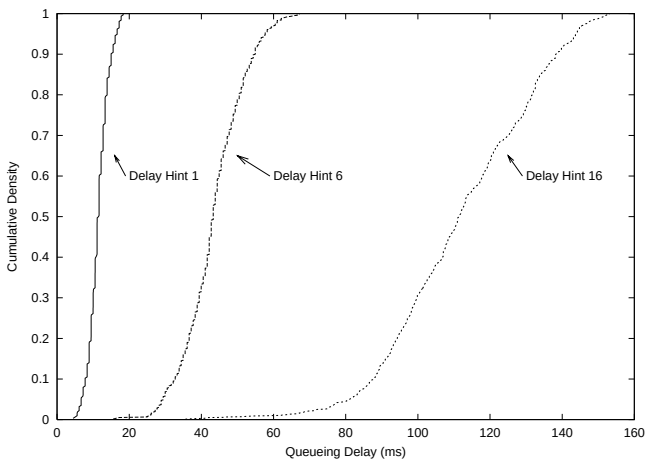

Figure 8: CDF of Queuing Delay for Audio Conference Flow with Delay Hints of 1, 6 and 16.

Figure 8 depicts a CDF of the queuing delay experienced by the audio flow for 3 different delay hints. The CDF is plotted for a delay hint 1, which gives the minimum delay, a delay hint 6, which gave the audio flow its optimal quality, and a delay hint 16, which gives the maximum delay. The median queuing delay is lower for the lower delay hints, and the CDF curves for hints 1 and 6 are steeper than for hint 16, which implies that there is less variation in the per-packet queuing delay

with lower hints. Hence, for delay sensitive applications an AQM with TSQ can provide a lower average queuing delay with less variation than can an AQM alone.

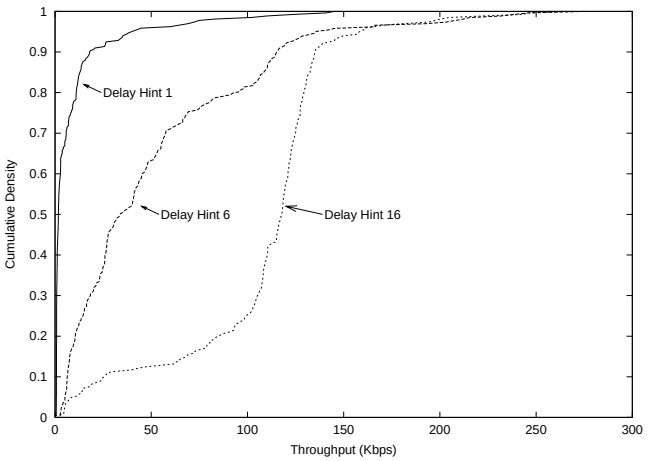

Figure 9: CDF of Throughput for Audio Conference Flow with Delay Hints of 1, 6 and 16.

Figure 9 shows a CDF plot for the throughput obtained by the audio flow for the delay hints of 1, 6 and 16. The throughput is calculated every round-trip time (300 ms in these experiments). The throughput distributions of the file transfer flows are similar to the distributions obtained with delay hints of 16. If TSQ were not used, then the throughput distribution would be similar to that of a flow with delay hint 16. As is evident from the figure, the median throughput decreases as the delay hint decreases.

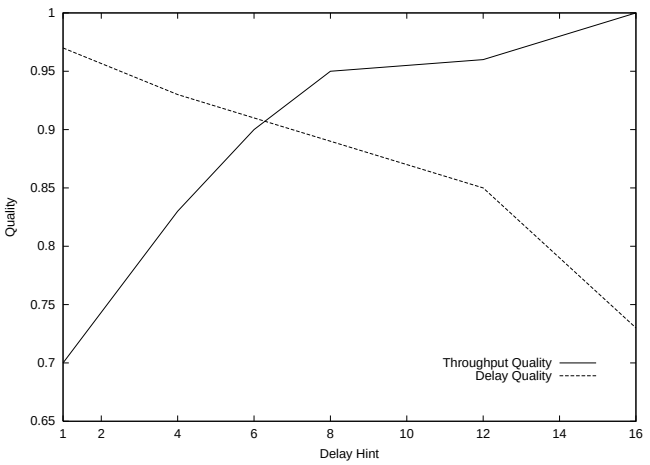

Figure 10: Throughput and Delay Quality for Audio Conference Flow versus Delay Hint

Using the quality model described in Section 2 and the throughput and total delay (queuing delay plus propagation delay), we compute the quality of the audio flow for different delay hints. Figure 10 shows the variation of the delay quality and throughput quality of the audio flow with different delay hints. The delay quality of the audio application improves with a decrease in delay hint, while its throughput quality decreases. In other words, as the application indicates its preference for lower delay, it is "cutting" in line more, hence getting a lower average queuing delay which improves its delay quality. However, correspondingly the audio flow gets dropped with a higher probability, hence achieving a lower throughput and causing a drop in the throughput quality. The overall quality of an application is the minimum of the delay quality and the throughput quality. Thus the application gets its best overall quality at a delay hint of 6. When TSQ is not used, the delay obtained by all applications is similar to that obtained by an application with delay hint 16.

4.3 Video Quality Evaluation

The experiments conducted in the previous section indicate TSQ can be used to improve the quality of applications that are primarily delay sensitive. We next present experiments evaluating TSQ for interactive video applications that are sensitive to both delay and throughput.

4.3.1 Setup

The network topology is as described in Section 4.1 with the bottleneck link capacity B=4 Mbps and the one-way propagation delay D=50 ms providing one-way propagation delays between each of the sources and their respective destinations at 150 ms. The number of flows N=20, of which 19 are bulk file transfers and 1 is a TCP-friendly CBR source sending data at a rate of 500 Kbps, typical of a H.323 video-conference [Cor00]. We run each experiment for 100 seconds, and then change the delay hint for the video flow for the next run.

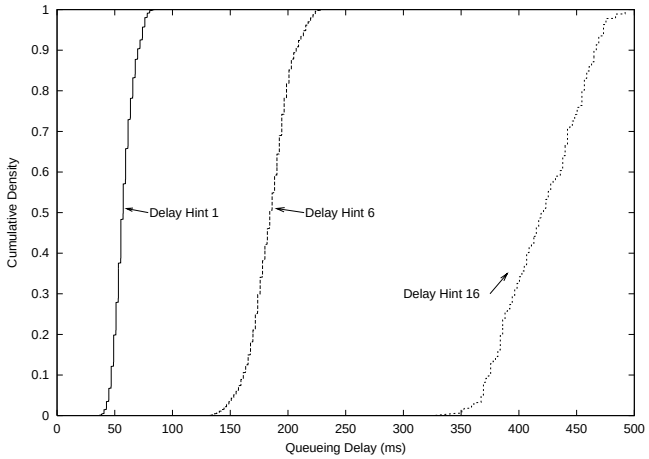

Figure 11: CDF of Queuing Delay of Video Conference Flow for Delay Hints of 1, 6 and 16

4.3.2 Analysis

Figure 11 shows the CDF of the queuing delay for the video flow for delay hints of 1, 6 and 16. As seen in Section 4.2.2 for the audio conference, the median queuing delay for the video conference is lower for the lower delay hints. Also, the CDF curves for delay hints of 1 and 6 are much steeper than for delay hints of 16, which implies low variance in the queuing delay. Thus, similar to for the audio conference, TSQ can provide a lower queuing delay with less variation to video conference applications.

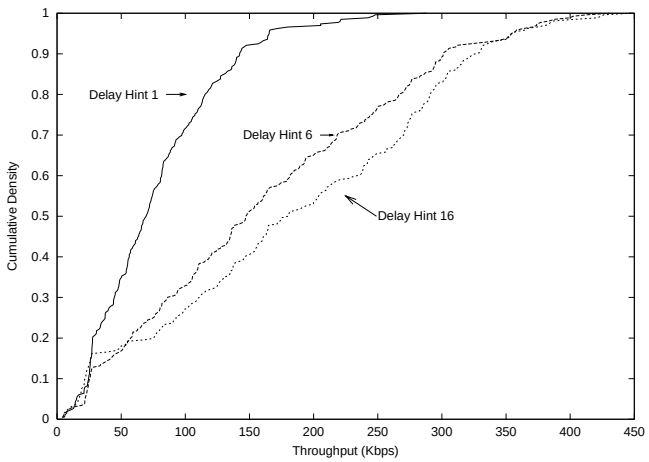

Figure 12: CDF of Throughput of Video Conference Flow for Delay Hints of 1, 6 and 16.

Figure 12 shows the CDF of the throughput obtained by the video conference flows for the same 3 delay hints. The throughputs are calculated over 1 round-trip time(300 ms in our experiments). The three CDF curves are more nearly the same for the video conference as compared to the CDF curves for the audio conference (Figure 12), indicating that the decrease in throughput is not significant when the delay hint is reduced.

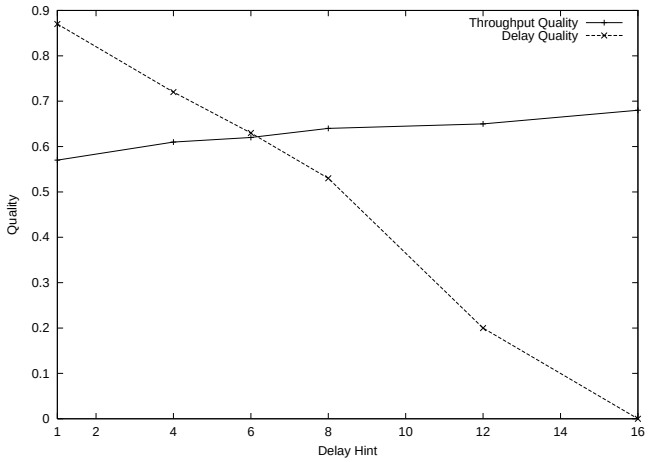

Figure 13: Throughput and Delay Quality for Video Conference Flow versus Delay Hint

The graph in Figure 13 shows how the quality of the video flow is affected by different delay hints. For lower delay hints, the average queuing delay and hence the average delay for the video flow decreases, resulting in a significant gain in delay quality, while the drop in throughput quality is less significant. The overall quality of the video conference for different delay hints is the minimum of the two curves, and is maximize when the delay hint is 6.

4.4 Mixed Traffic Evaluation

The experiments conducted so far had one single delay sensitive flow (an audio conference in the first set of experiments and a video conference in the second set of experiments). We now evaluate the performance of TSQ when there is a varying mix of delay sensitive and throughput sensitive flows.

4.5 Setup

The experimental setup for this experiment is similar to the first set of experiments (B=15, D=50, N=100). Within the 100 flows, we changed the relative number of delay sensitive (audio) flows with respect to the number of throughput sensitive (file

transfer) flows. The traffic mixes we ran include: 1 audio flow, 99 file transfer flows; 25 audio, 75 file transfer; 50 audio, 50 file transfer; and 75 audio, 25 file transfer.[3] The audio flows were a TCP-friendly CBR sources sending data at a rate of 128 Kbps and using a delay hint of 6 (the optimum delay hint from Section 4.2), while the file transfer flows used the maximum delay hint of 16.

4.6 Analysis

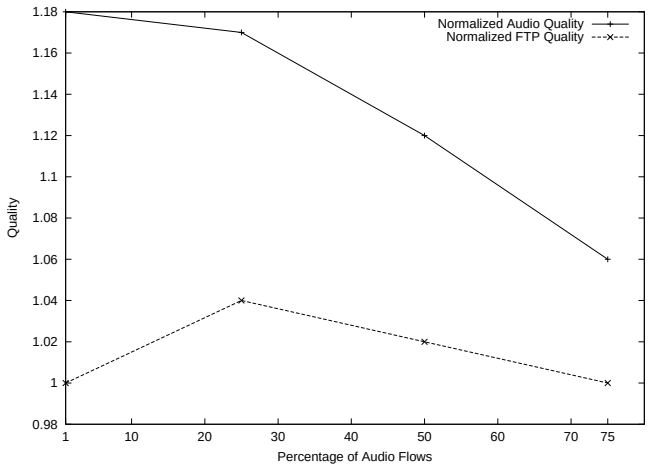

Figure 14: Normalized Quality of Audio Flows and File Transfer Flows for Varying Traffic Mixes

We calculated the average quality obtained by the file transfer flows and the audio flows for the various traffic configurations. This quality was then normalized against the quality that the application obtained when TSQ was not enabled (the bottleneck router only ran PI). In other words, the normalized quality of an application when TSQ is switched off is 1. If an application receives better QoS when TSQ is enabled, then its normalized quality is greater than 1. Conversely, if the quality of the application is worse when TSQ is not enabled, then normalized quality is less than 1.

Figure 14 shows that as the percentage of audio flows in the network increases, the average gain in quality of the audio application decreases. This is because as the number of delay sensitive flows increases in the network, the delay sensitive flows will cut in line less than they would when there are more throughput sensitive flows, reducing the quality gains. However, notice at all times the normalized quality is greater than 1, hence, the quality of service obtained using TSQ is always higher than that obtained without TSQ even with increasing numbers of audio flows.

For the file transfer flows, the normalized quality increases initially with an increase in number of flows. However, as the number of audio flows increases beyond 25 percent, the normalized file transfer quality starts decreasing. Again, for all traffic mixes, the normalized file transfer quality is greater or equal to 1. Thus, TSQ provides better or equal quality for both audio conference and file transfer applications than does the underlying AQM (PI in our experiments) without TSQ.

4.7 Unresponsive Flows

In the previous experiments we have made all interactive audio and video flows TCP friendly, while in practice there may be interactive audio and video flows that are unresponsive to network congestion. In this section we evaluate the behavior of unresponsive flows when TSQ is used. During congestion, an unresponsive application will not reduce its data rate in response to packet loss. Hence, we investigate whether unresponsive UDP flows can gain an unfair advantage by taking advantage of TSQ. In the first set of experiments, we introduced a single unresponsive UDP flow in a network with only file transfer TCP flows. We observed the effect of the UDP flow on the average throughput of the TCP flows. We repeat the experiment with different values for the delay hints for the UDP flow.

In the second set of UDP experiments we evaluate the effect on quality of UDP and TCP applications with varying mixes of UDP and TCP flows. The quality of these applications were normalized against the quality achieved under similar network conditions if TSQ was not used.

4.7.1 Set 1

In this set of experiments, the network topology is similar to those in previous experiments (B=15, D=50, N=100), with 99 bulk file transfers using TCP, and 1 audio flow over UDP. The audio flow

[3]The extreme case of 99 audio flows and 1 file transfer flow was not evaluated, as this configuration did not cause sufficient congestion for any queuing delay build-up and hence was not useful for comparative evaluation.

is unresponsive CBR sending data at a rate of 600 Kbps, which is more than the flow's fair share of bandwidth of 150 Kbps. The file transfer use the maximum delay hint of 16 while the unresponsive UDP flow uses a different delay hint in each 100 second run.

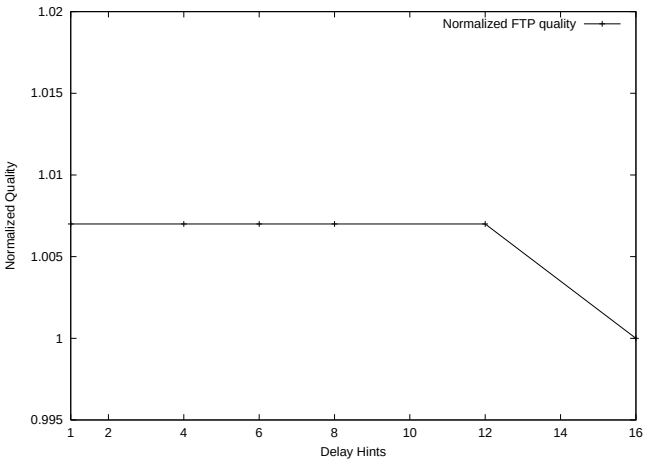

Figure 15: Normalized File Transfer Quality versus Delay Hint in the Presence of a High-Bitrate, Unresponsive Flow

We measured the average throughput for the 99 file transfer flows in each run. Figure 15 shows the average file transfer throughput when running with UDP flows with different delay hints. The throughput is normalized against the average file transfer throughput when the same experiment is run on PI without TSQ enabled. As we can see from the graph, the average file transfer throughput remains almost constant in each of the runs, with the file transfer throughput being a little higher when the UDP flow tries to "cheat" by using a lower delay hint. This makes AQM routers that use TSQ no more vulnerable to unresponsive flows than if they did not use TSQ.

4.7.2 Set 2

In this set of experiments, the network topology is similar to those in previous experiments (B=15, D=50, N=100), where the 100 flows are a mix of unresponsive audio flows running over UDP and file transfers running over TCP. The audio flows send at an unyielding rate of 128 Kbps and use a delay hint of 6, while the TCP flows are elastic and use a delay hint of 16. We vary the mix of UDP flows from 1 to 75 (1, 25, 50 and 75).

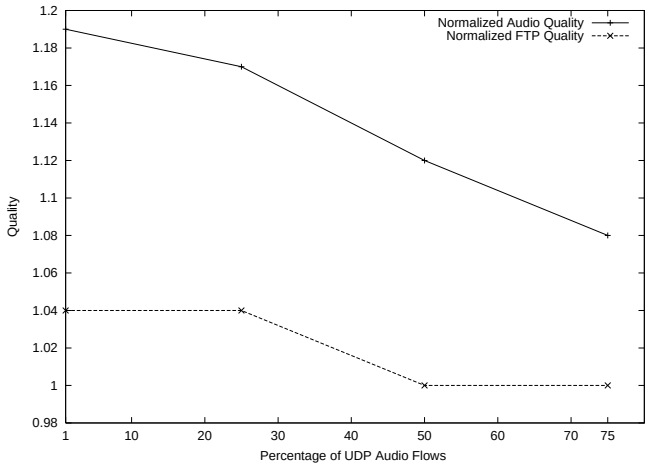

Figure 16: Normalized Quality versus Traffic Mix with Unresponsive Audio Conference Flows

Figure 16 plots the average audio and file transfer quality normalized against the average quality obtained without TSQ. As the number of UDP audio flows in the network increases, the normalized quality decreases for both the UDP and file transfer applications. However, at all times the normalized quality is above 1 for both the UDP audio and TCP file transfers. Hence, there is an improvement in the average quality of both the UDP audio and TCP file transfer applications due to the TSQ for a varying mixes of flows, suggesting TSQ will not more negatively impact network performance in the presence of unresponsive flows.

5 Conclusions

The current Internet supports applications with primary Quality of Service (QoS) requirements of delay and throughput. Unfortunately, the current Internet however does not differentiate between application QoS requirements and instead provides uniform service to all applications. We assert that the Internet can instead provide QoS mechanisms while remaining best effort, raising the overall QoS for most applications, while preserving the robustness and scalability of the network, all without requiring complicated policing, pricing or per-flow accounting mechanisms.

In this paper, we have presented a Traffic Sensitive QoS controller (TSQ). TSQ is sensitive to the

varying QoS requirements of diverse Internet traffic, and thus provides different delay and throughput treatments to packets from different types of applications. TSQ can be used in conjunction with many current AQM techniques allowing the full performance benefits to quality that the underlying AQM has to offer. Applications inform TSQ about their delay sensitivity by embedding within each packet a delay hint, an indicator of an application's delay sensitivity. Based on the delay hint of each packet, TSQ makes a decision as to where the packet must be inserted in the queue (thus potentially decreasing its queuing delay) and how much the drop probability of the packet must be increased (thus potentially decreasing its throughput). This mechanism helps delay-sensitive applications attain better QoS, while at the same time avoids hurting, and sometimes helps, the QoS of throughput sensitive applications.

In order to quantify an application's QoS, we propose a QoS metric based on the minimum of an application's delay quality and throughput quality. Based on earlier work in perceived quality, we have contributed quality metrics for some typical Internet applications: interactive audio, interactive video and file transfer. Quality function such as these, along with a TSQ-enabled Internet, can dynamically choose their delay hints so as to maximize their Quality of Service.

Our evaluation of TSQ with varying traffic mixes shows TSQ can increase the average quality of all applications (8% to 18% for delay sensitive applications and up to 4% for throughput sensitive applications) over the quality obtained by using the AQM without TSQ, all while not allow unresponsive traffic to gain further advantage over responsive traffic than does the underlying AQM.

6 Future Work

Our current implementation of TSQ uses 4 bits in the IP header to embed the delay hint, allowing applications to choose from 16 levels of delay sensitivity. A larger range of delay hints will be available if more bits are used to embed the delay hint, but at the cost of more bits of overhead. Hence, further research is required to determine the appropriate number of bits needed to support a range of delay sensitivities without inducing unnecessary overhead.

Another area of potential future research is in developing quality metrics. We have devised quality metrics representative of three applications (interactive audio, interactive video and file transfer), however, other applications may have different QoS requirements. In addition, there may be other ways to quantify QoS, such as taking the average (or the sum) of the throughput and delay qualities, suggesting further investigation into the quality metrics and requirements of other applications on the Internet is appropriate.

Another possible extension would be to build applications that can take advantage of TSQ by dynamically changing their delay hints. These applications could then evaluate the quality that they obtained by using their current delay hint and adapt their delay hint if they are not satisfied with the QoS received. How rapidly an application would adapt to changing network QoS would need to be explored.

References

[BFPT99] J-C. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-Based Error Control for Internet Telephony. In *Proceedings of IEEE INFOCOM*, March 1999.

[CC00] Jae Chung and Mark Claypool. Dynamic-CBT and ChIPS - Router Support for Improved Multimedia Performance on the Internet. In *Proceedings of the ACM Multimedia Conference*, November 2000.

[Cor98] Real Networks Corporation. Real Networks Guide for Audio Production, 1998.

[Cor00] RadVision Corporation. Multipoint Conferencing Specifications, 2000.

[CPS02] Andrew Corlett, D. I. Pullin, and Stephen Sargood. Statistics of One-Way Internet Packet Delays. In *Proceedings of 53rd Internet Engineering Task Force*, March 2002.

[DCJ93] Spiros Dimolitsas, Franklin L. Corcoran, and John G. Phipps Jr. Impact of Transmission Delay on ISDN Videotelephony. In *Proceedings of Globecom '93 – IEEE Telecommunications Conference*, pages 376 – 379, Houston, TX, November 1993.

[FJ93] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, August 1993.

[FKSS01] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: An Alternative Approach To Active Queue Management. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2001.

[Gan02] Implementation Architecture Specification for the Premium IP Service, 2002. http://archive.dante.net/geant/public-deliverables/GEA-02-079v2.pdf.

[GH03] Yuan Gao and Jennifer Hou. A State Feedback Control Approach to Stabilizing Queues for ECN-Enabled TCP Connections. In *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, April 2003.

[HBWW99] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. *IETF Request for Comments (RFC) 2597*, June 1999.

[HKBT01] P. Hurley, M. Kara, J. Le Boudec, and P. Thiran. ABE: Providing a Low Delay within Best Effort. *IEEE Network Magazine*, May/June 2001.

[HMTG01] C. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proceedings of IEEE Infocom*, April 2001.

[IKK93] Satoru Iai, Takaaki Kurita, and Nobuhiko Kitawaki. Quality Requirements for Multimedia Communcation Services and Terminals – Interaction of Speech and Video Delays. In *Proceedings of Globecom – IEEE Telecommunications Conference*, pages 394 – 398, Houston, TX, November 1993.

[JNP99] V. Jacobson, K. Nichols, and K. Poduri. Expedited Forwarding PHB Group. *IETF Request for Comments (RFC) 2598*, June 1999.

[LC00] Yanlin Liu and Mark Claypool. Using Redundancy to Repair Video Damaged by Network Data Loss. In *Proceedings of IS&T/SPIE/ACM Multimedia Computing and Networking (MMCN)*, January 2000.

[NT02] Wael Noureddine and Fouad Tobagi. Improving the Performance of Interactive TCP Applications using Service Differentiation. In *Proceedings of IEEE Infocom*, June 2002.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and Its Empirical Validation. In *Proceedings of ACM SIGCOMM*, 1998.

[PHH98] Carlos Perkins, Orlin Hodson, and Vicky Hardman. A Survey of Packet-Loss Recovery Techniques for Streaming Audio. *IEEE Network Magazine*, Sep/Oct 1998.

[PJS99] Mark Parris, Kevin Jeffay, and F. Smith. Lightweight Active Router-Queue Management for Multimedia Networking. In *Proceedings of Multimedia Computing and Networking (MMCN), SPIE Proceedings Series*, January 1999.

[Pug90] William Pugh. Skip Lists: A Probabilistic Alternative to Balalnced

Trees. *Communications of the ACM*, 33(6):668–676, June 1990.

[SBC94] S. Shenker, R. Braden, and D. Clark. Integrated Services in the Internet Architecture: An Overview. *IETF Request for Comments (RFC) 1633*, June 1994.

[SZ99] Ion Stoica and Hui Zhang. Providing Guaranteed Service Without Per Flow Management. In *ACM SIGCOMM Computer Communication Review , Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, August 1999.

[Zeb93] J.A. Zebarth. Let Me Be Me. In *Proceedings of Globecom – IEEE Telecommunications Conference*, pages 389 – 393, Houston, TX, November 1993.