5-22-2002

RED-Worcester - Traffic Sensitive Active Queue Management

Vishal Phirke
Worcester Polytechnic Institute, vish@cs.wpi.edu

Mark Claypool
Worcester Polytechnic Institute, claypool@wpi.edu

Robert Kinicki
Worcester Polytechnic Institute, rek@wpi.edu

RED-Worcester - Traffic Sensitve Active Queue Management

Vishal Phirke, Mark Claypool, Robert Kinicki
Worcester Polytechnic Institute
Computer Science Department
100 Institute Road
Worcester, MA 01609
{vish|claypool|rek}@cs.wpi.edu

May 22, 2002

Abstract

Traditional Internet applications such as FTP and e-mail are increasingly sharing bandwidth with newer, more demanding applications such as Web browsing, IP telephony, video conferencing and online games. These new applications have Quality of Service (QoS) requirements, in terms of delay and throughput, that are different than QoS requirements of traditional applications. Unfortunately, current Active Queue Management (AQM) approaches offer monolithic best-effort service to all Internet applications regardless of the current QoS requirements. This paper proposes and evaluates a new AQM technique, called RED-Worcester, that employs source hints to provide service at network routers that is sensitive to the aggregate QoS requirements for all flows passing through the router. Applications indicate their delay and throughput sensitivity via a delay hint embedded in their outgoing packets. The RED-Worcester router uses the delay hint to dynamically adjust Adaptive RED [FGS01] parameters to yield better overall QoS, providing lower delay when most flows are delay-sensitive and higher throughput when most flows are throughput-sensitive. Using a new QoS metric, our simulations demonstrate that RED-Worcester yields higher overall QoS than Adaptive RED when there are delay-sensitive flows and operates equally well in other traffic scenarios. RED-Worcester fits the current best-effort Internet environment without requiring traffic monitoring.

1 Introduction

The Internet today carries traffic for applications with a wide range of delay and throughput requirements, as depicted in Figure 1. Traditional applications such as FTP and E-mail that are primarily concerned with throughput and can tolerate high delays due

to long router queues in exchange for high throughput. Emerging applications such as IP telephony, video conferencing and networked games, on the other hand, have different requirements in terms of throughput and delay than these traditional applications.

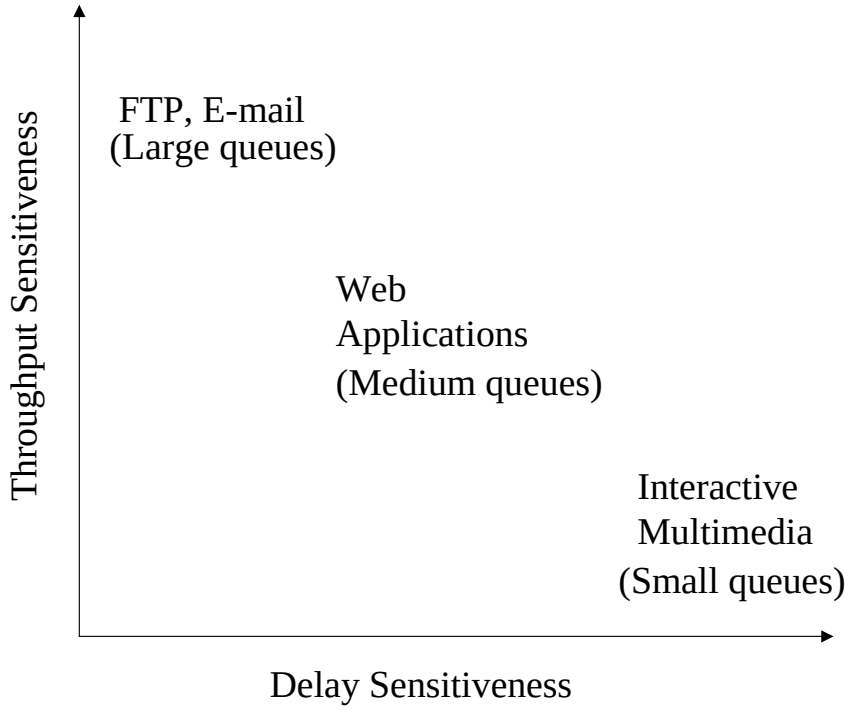

Figure 1: QoS Spectrum of Applications

In particular, interactive multimedia applications have more stringent delay constraints than loss constraints. By using repair techniques [BFPT99, LC00, PCM00, PW99] that partially or fully conceal packet loss, multimedia applications are able to operate over a wide range of packet losses. This leaves end-to-end delay and delay jitter as their major impediment to acceptable quality. Thus, interactive multimedia applications prefer smaller queues in Internet routers, even if this means higher losses and lower throughput. Since Web traffic is moderately sensitive to delay as well as throughput, it falls in the middle in terms of the delay and throughput spectrum.

Unfortunately, current Internet routers do not provide Quality of Service (QoS)[1] adapted to the current traffic mix. Current and proposed router queue mechanisms can be classified as in Figure 2 on the basis of the level of QoS support provided and the complexity of the router implementation. Most Active Queue Management (AQM) techniques are either heavy-weight by requiring significant architectural changes or focus on

[1]Throughout this work, we use the term QoS to refer explicitly to *delay* and *throughput* provided by the network.

providing higher throughput at the router without much consideration for queuing delays.

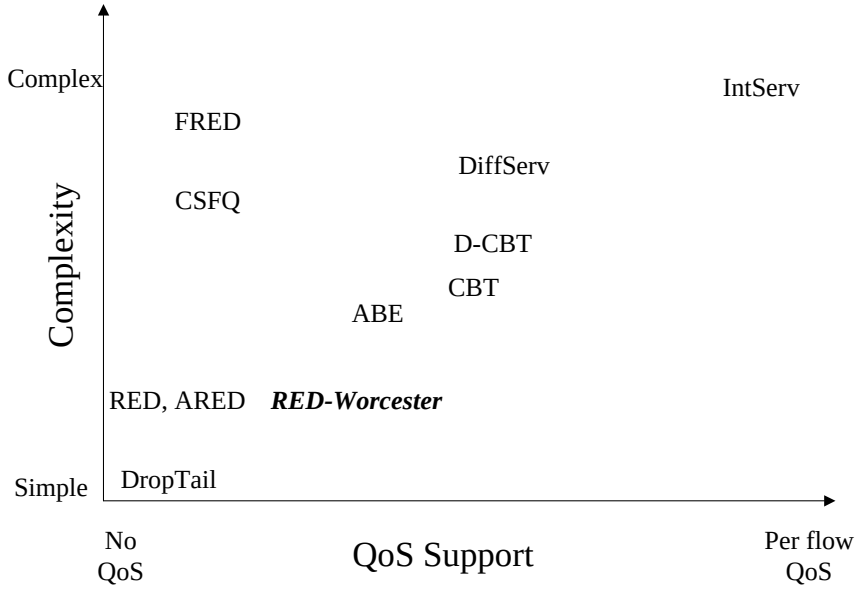

Figure 2: Approaches for QoS in Internet Routers

Due to the simplicity of the FIFO queuing mechanism, drop-tail queues are the most widely used queuing mechanism in Internet routers today. When drop-tail buffers overflow, newly arriving packets are dropped regardless of the application constraints for the packet. To accommodate bursty traffic, drop-tail routers on the Internet backbone are over-provisioned with large FIFO buffers [IMD01]. When faced with persistent congestion, drop-tail routers yield high delays for all flows passing through the bottlenecked router. This best-effort service provides no consideration for interactive multimedia flows or even Web flows that can be severely affected by high delays. Clearly, drop-tail provides very limited, if any, QoS support.

RED [FJ93], probably the best-known AQM mechanism, attempts to keep the average queue size at the router low while keeping throughput high. By detecting the onset of congestion earlier than drop-tail, RED avoids the global synchronization of TCP flows that hampers aggregate throughput. Adaptive RED (ARED) [FGS01] adjusts the RED operating parameters to accommodate a wider range of traffic loads. However, both RED and ARED provide equal treatment to incoming traffic and tend to be tuned for high throughput without any consideration for the traversing applications' aggregate delay requirements.

Streaming multimedia applications avoid using TCP as TCP can suffer from bursty data rates and TCP retransmits data lost due to packet drops at the router, which in turn results in poorer quality multimedia due to increased delay and jitter. Instead, streaming

media applications often choose UDP as their transport protocol [WCZ01]. However, UDP is unresponsive to packet drops that indicate congestion at the router. FRED [LM97] and CSFQ [SSZ98] routers are specifically designed to limit the bandwidth of unresponsive flows to achieve throughput fairness among flows. While these strategies provide good QoS for flows highly concerned about throughput, FRED and CSFQ ignore the willingness of interactive multimedia applications to accept reduced throughput if accompanied by reduced delay.

Since unresponsive UDP flows can lead to network congestion collapse and because UDP does not adjust its data rates when faced with network congestion, there is a strong movement to make streaming multimedia traffic *TCP-friendly* [FF99]. Thus, in the near future many multimedia applications are likely to use TCP-friendly protocols that adjust their tranmission rates smoothly, such as in [FHPW00, RHE99, TZ99], without actually using TCP. Hence, this research assumes all flows, including multimedia flows, either use TCP or TCP-friendly protocols that respond to packet drops as indicators of congestion.

ABE [HKBT01] provides a queue management mechanism for low delay traffic. In ABE, delay-sensitive applications can sacrifice throughput for lower delays. However, ABE traffic classification is rigid in that applications are either delay-sensitive or throughput-sensitive and ABE requires a complicated deadline based scheduler to serve packets.

CBT [PJS99] provides class-based treatment with guarantees on bandwidth limits for different classes, and DCBT with ChIPS [CC00] extends CBT by providing dynamic thresholds and lower jitter for multimedia traffic. However, both CBT and DCBT provide per class differentiated throughput service. Application delay constraints are not considered in these AQM algorithms.

DiffServ approaches, such as Assured Forwarding (AF) [HBWW99] and Expedited Forwarding (EF) [JNP99], provide differentiated service to traffic aggregates. However, they require complicated mechanisms to negotiate service level agreements. Additionally, DiffServ requires traffic monitors, markers, shapers, classifiers and droppers and a framework to enable these components to work together.

By requiring per flow signaling and reservations via RSVP [Wro97] by all routers on the connection path, IntServ provides the best possible per flow QoS guarantees. However, the complexity of the RSVP signaling and the per flow soft state at each router means that IntServ does not scale well with number of flows.

This paper presents a new AQM technique called *RED-Worcester*[2] to provide QoS support at a best effort router. In RED-Worcester applications mark each packet with a suggested delay, referred to as a *delay hint*, indicating the relative importance of delay

[2]Similar to various versions of TCP, which are named after cities in Nevada, we have named our RED active queue management technique after a city in Massachusetts.

versus throughput to the success of the application. As a simple extension to ARED, the RED-Worcester objective is to improve overall QoS support at the router by satisfying the average performance requirements of incoming packets in terms of throughput and delay. While not a differential service, RED-Worcester provides a service that better matches the average requirements of all applications without adding much complexity and it requires no additional policing mechanisms, charging mechanisms or usage control. Under the proposed service, RED-Worcester routers operate equally well under scenarios with only traditional traffic and operate better under scenarios with mixed or multimedia only traffic.

This study evaluates RED-Worcester via simulation under a variety of traffic mixes and compares its performance with ARED. The results show that as traffic mix changes from mostly throughput-sensitive to mostly delay-sensitive, RED-Worcester improves the average QoS support at the router.

The remainder of this paper is organized as follows: Section 2 describes the RED-Worcester mechanism; Section 3 discusses the simulation topology and our QoS metric; Section 4 presents our results with detailed analysis and Section 5 summarizes our work and discusses possible future work.

2 RED-Worcester

This section explains the RED-Worcester mechanism designed to improve overall QoS support at the bottleneck router. RED-Worcester uses delay hints as explained in Section 2.1 to determine the requirements of incoming traffic in terms of delay and throughput at the router. RED-Worcester directly extends Adaptive RED, described in Section 2.2, by using the delay hints to calculate a target average queue size as explained in Section 2.3. Associated implementation overhead is considered in Section 2.4.

2.1 Delay Hints

In general, *source hints* are packet labels sent by end hosts or edge routers at the ingress to a network that carry information about a flow such as round trip time, bandwidth used, protocol type or other attributes. In RED-Worcester, application end-hosts indicate their sensitivity to queuing delays as source hints that we refer to as *delay hints*. The delay hint is not an absolute bound on queuing delay, but rather indicates to Internet routers the relative importance of delay versus throughput for this flow. If delay hints are put in packet headers by edge routers, then existing applications can be easily supported without modification, whereas if delay hints are inserted by the end hosts, then applications will

have more flexibility in choosing their own hints. For flows which do not provide delay hints, RED-Worcester uses a default delay hint which corresponds to ARED's target queue size.

RED-Worcester does not guarantee queuing delays based on the delay hint, but uses the delay hint to calculate the average queuing requirements of incoming packets and adjust the target average queue accordingly.

2.2 Adaptive RED

RED-Worcester directly extends Adaptive RED [FGS01] (ARED). ARED builds a target queue range on top of the RED [FJ93] mechanisms. RED monitors the outgoing queue for impending congestion, accommodating bursty traffic by keeping an exponential weighted moving average of the queue (q_{ave}). When congestion is detected, indicated by the queue average rising above a fixed minimum threshold (min_{th}), packets are randomly dropped with a fixed drop probability for each packet. The probability of packet drop increases linearly from zero at the minimum threshold to a maximum drop probability (max_p) at the maximum threshold (max_{th}).

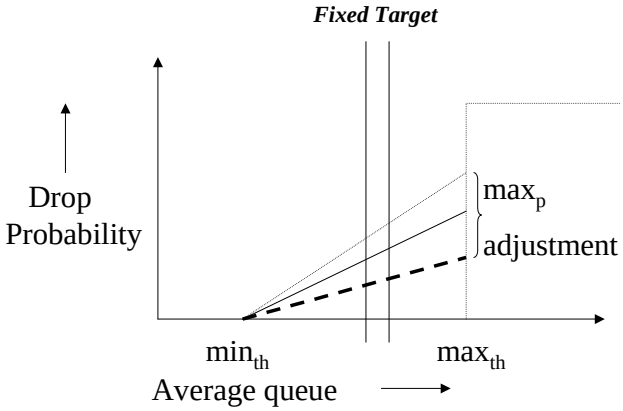

Figure 3: Adaptive RED

ARED keeps RED's basic structure but adjusts max_p to keep the q_{ave} within a target range between min_{th} and max_{th} as shown in Figure 3. The default target is centered halfway between min_{th} and max_{th} with a range of 0.1 on either side. The ARED algorithm adjusts its target as depicted in Figure 4.

Every *interval* **seconds:** *// 500 ms, by default*
  if (q_{avg} > target and $max_p < 0.5$) then
    increase max_p:
      max_p += α *// about 0.1, by default*
  elseif (q_{avg} < target and $max_p > 0.01$) then
    decrease max_p:
      max_p ×= β *// 0.9, by default*

Figure 4: Adaptive RED Algorithm for Adjusting Average Queue to the Target Queue

2.3 Moving Target

In ARED, the targeted average queue size is independent of traffic QoS requirements. We argue that the targeted average queue size should be based on the incoming traffic's requirements. RED-Worcester extends ARED by providing a moving target queue size based on an exponentially weighted moving average of the delay hints of the incoming packets.

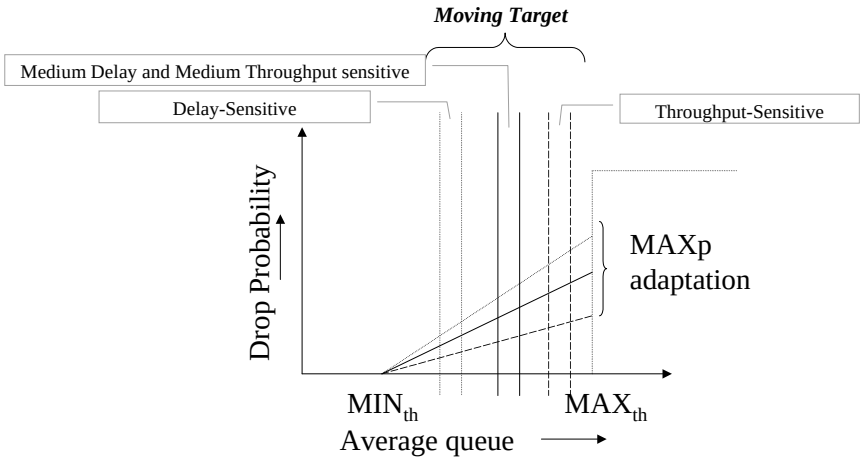

Figure 5: RED-Worcester

Unlike ARED's fixed target queue size, RED-Worcester updates its target queue size based on incoming traffic requirements as depicted in Figure 5. Thus, when incoming traffic is mostly throughput-sensitive, RED-Worcester maintains a higher average queue to improve the overall throughput. On the other hand, when incoming traffic is mostly delay-sensitive, RED-Worcester lowers the average queue size to reduce the average queuing delays. In case of medium throughput and medium delay sensitive traffic, RED-Worcester maintains an average queue size which gives medium throughput and delay performance.

For each incoming packet at the RED-Worcester router, the target queue size that a

packet can tolerate is calculated based on the delay hint specified in the packet:

$$Target = C \times D/P \tag{1}$$

where C is the link capacity in bps, D is the delay hint in seconds and P is the packet size in bits. The moving target is calculated as the exponentially weighted average of the tolerable target queue size for each incoming packet:

$$Moving\ Target = (1 - w_t) \times Moving\ Target + w_t \times Target \tag{2}$$

where w_t is the weight used for calculating the exponentially weighted moving average, set to $w_t = 0.02$ in this investigation. Using this moving target for ARED ensures that the average queue size at the RED-Worcester router reflects the average QoS requirements of the incoming packets.

2.4 Overhead

To use the QoS services provided by RED-Worcester, packets have to be labeled with delay hints. Based on discussion in [SZ99], there are from 4 to 17 bits in the IP header that may be available to carry source hints. Using milliseconds as the units for delay hints, 10 bits covers queuing delays from 0 to 1023 ms. Since 10 ms granularity is more than sufficient for most applications, the number of bits needed can be reduced to 7. The labeling of delay hints itself can be done either by end hosts, most likely the applications, or by edge routers at the ingress of a network. The RED-Worcester router has to read the labels from the incoming packets, an overhead similar to that in other approaches such as ABE [HKBT01], AF [HBWW99] and CBT [PJS99]. RED-Worcester calculates the moving queue average target, a calculation similar to the average queue calculation in ARED.

Thus, RED-Worcester adds a little complexity to ARED and, in return, provides a mechanisms for aggregate QoS support at the router.

3 Experiments

This section describes the experimental setup and NS-2 simulation details associated with our comparison study of the performance of RED-Worcester versus ARED. The NS-2 simulator [oCB] provides the ability to simulate drop-tail, RED and ARED routers. Additionally, NS-2 includes code to simulate both TCP protocols and TCP-friendly rate controlled protocols such as TFRC [FHPW00]. RED-Worcester was simulated by extend-

ing the ARED implementation and TCP NewReno and TFRC were modified to send delay hints.

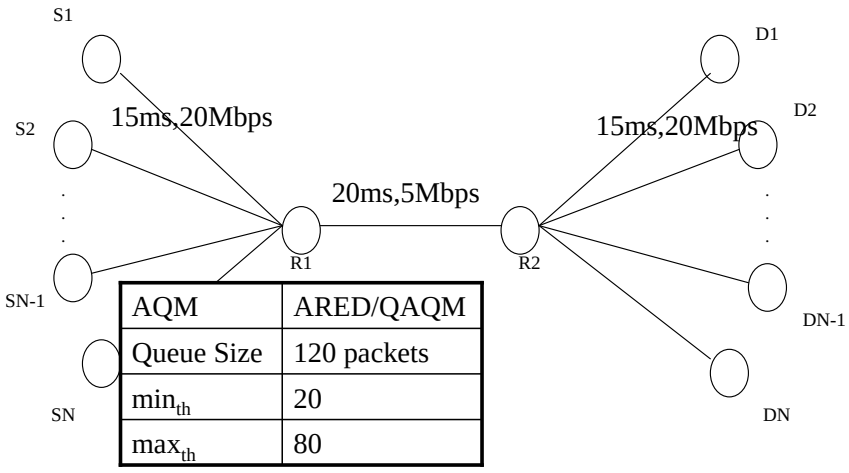

Figure 6: Network Topology

The generic network topology used for all experiments is shown in Figure 6. S1-SN are traffic sources and D1-DN are the corresponding destinations. The S-D pairs were varied to provide traffic mixes that included different combinations of TCP NewReno and TFRC flows, where the TFRC flows are meant to represent TCP-friendly multimedia flows. All sources send fixed-length 1000-byte packets. All links connecting sources to router R1 and all links connecting destinations to router R2 have a 20 Mbps capacity and a 15 ms delay. With the bandwidth and delay of the bottleneck link going from R1 to R2 set at 5 Mbps and 20 ms respectively, packet drops only occur at R1, where all performance measurements are made. For both RED-Worcester and ARED, the queue size at the congested router is 120 packets and min_{th} and max_{th} are set to 20 packets and 80 packets, respectively. All simulated flows start at time 0 seconds and end at time 100 seconds. Performance measurements are taken during the stable interval between 20 seconds and 80 seconds to avoid transient conditions from startup and stopping.

We first evaluate how RED-Worcester's target adjusts to changing traffic requirements. We plot average queue size under different traffic mixes to see that it does reflect the average requirements of the incoming flows and we also compare overall drop rates for RED-Worcester and ARED. We measure queuing delays in milliseconds at the congested router.

However, analyzing throughput and delay alone is not sufficient as application performance realistically involves tradeoffs between throughput and delay. We propose a new metric, QoS, that provides a quantitative performance metric designed to capture to some

degree the nature of the throughput-delay tradeoff. The correct choice of QoS function depends upon the the application's sensitivity to delay and throughput. An application may dynamically adjust its delay hints to current network conditions in attempt to improve its QoS. To include a wide spectrum of individual application types that can select their own tailored QoS objective, we provide a QoS measure that is generic while permitting customized combinations of delay and throughput measures for individual applications:

$$QoS = T^{\alpha}/D^{\beta} \tag{3}$$

where T is throughput in bps, D is queuing delay in seconds at the router and $\alpha + \beta = 1$. While the choices of α and β depend upon the throughput and delay tolerances of applications themselves, the expectation is that the relativity of the delay hints from RED-Worcester enabled sources will vary accordingly. Note our new metric, QoS, is in fact a traffic-sensitive variant of the power performance metric.

In the experiments discussed in the next section, $\alpha = 1$, is used for throughput-sensitive flows (such as Email or file transfer). Since these flows can tolerate delay, QoS for them depends only upon throughput. $\alpha = 0.5$ and $\beta = 0.5$ is used for delay-sensitive flows (such as an interactive videoconference), since these flows require low delay, but also moderate amounts of throughput. For medium throughput and medium delay-sensitive flows (such as HTTP for Web browsing), we use $0.5 < \alpha < 1$ and $0 < \beta < 0.5$.

4 Evaluation

This section presents simulation results and analysis comparing RED-Worcester and ARED. The simulations evaluate performance under traffic mixes that vary from mostly throughput-sensitive flows to mostly delay-sensitive flows. Figure 7 provides details on the traffic mixes used for our simulations.

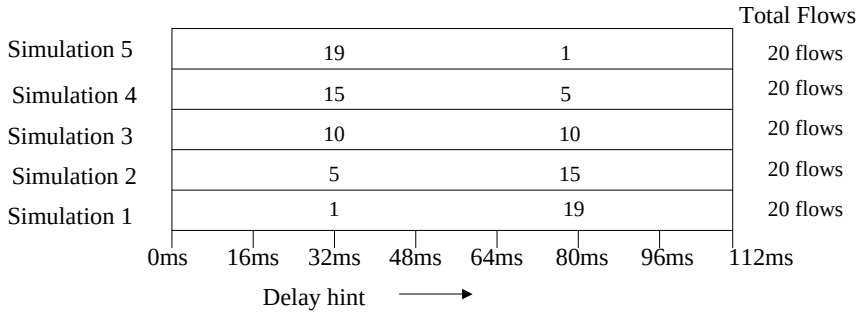

Figure 7: Traffic Mixes

Each simulation ran 20 flows using the network topology shown in Figure 6. Throughput-sensitive flows were represented by TCP flows carrying a delay hint of 80 ms, which corresponds to a queue size of 50 packets at the bottlenecked router R1. Delay-sensitive flows were represented by TFRC [FHPW00] flows carrying a delay hint of 32 ms, which corresponds to a queue size of 20 packets at R1. The X-axis for all the graphs in this section indicates changing traffic mixes corresponding to the five different simulations described in Figure 7.

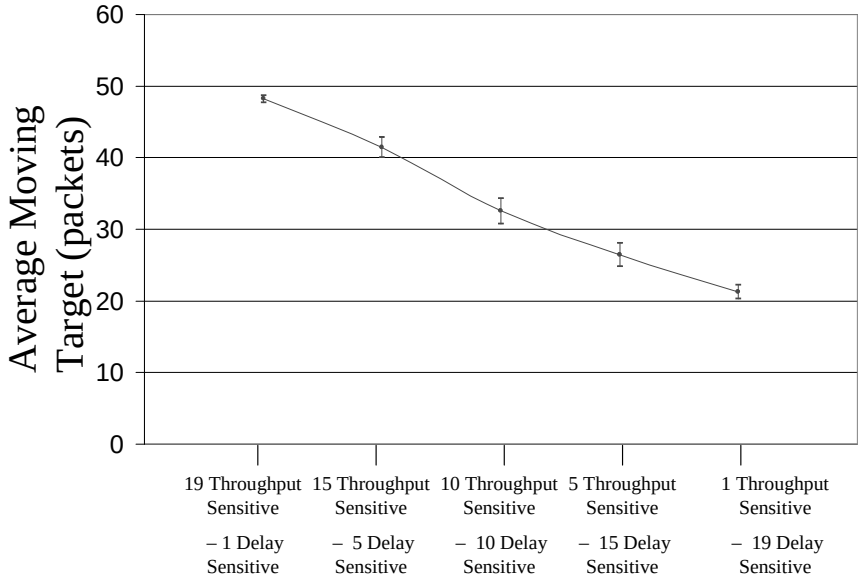

Figure 8: RED-Worcester's Moving Target

Figure 8 shows RED-Worcester's *moving target* for different incoming traffic mixes, with the bars showing the standard deviations in the moving target. When most of the incoming flows are throughput-sensitive carrying a delay hint of 80 ms, i.e., queue size of 50 packets, RED-Worcester's moving target settles at around 48 packets. As the number of delay-sensitive flows carrying a delay hint of 32 ms, i.e., queue size of 20 packets, increases, the moving target moves down linearly and, when most of the flows are delay sensitive, settles at around 22 packets.

Figure 9 shows the max_p adaptation over time in RED-Worcester as the traffic mix changes from completely throughput-sensitive (low drop probability) to a mix that is more delay-sensitive (higher drop probability). When most of the incoming flows are throughput-sensitive, max_p settles to a comparatively lower value, allowing the average queue size to grow. This improves the overall throughput performance of the RED-Worcester router by better serving the QoS needs of the throughput-sensitive flows. Contrastingly, when most of the incoming flows are delay-sensitive, max_p settles at a com-

paratively higher value that forces RED-Worcester to maintain a smaller average queue size. This reduction in the queuing delays at the router is exactly the service adjustment that the delay-sensitive flows expect. When the percentages of throughput-sensitive and delay-sensitive flows in the incoming traffic are equal, max_p settles to a value such that the average queue size at the router will be of medium size. Thus, the RED-Worcester router compromises when there are equal number of throughput-sensitive and delay-sensitive flows and gives a medium delay and medium throughput service.

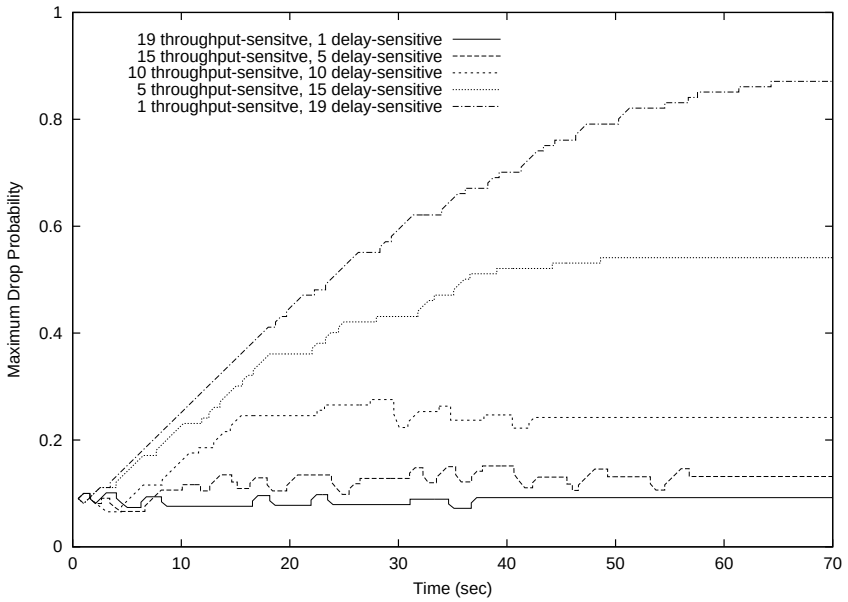

Figure 9: max_p Increase Analysis

Figure 10 shows the max_p adaptation over time in RED-Worcester as the traffic mix changes from completely delay-sensitive (high drop probability, max_p of 1 in this case) to a mix that is more throughput-sensitive (lower drop probability). Due to the additive increase and the multiplicative decrease of max_p as built into ARED, reduction in max_p to the stable state is much quicker than the increase in max_p, as seen in Figure 9.

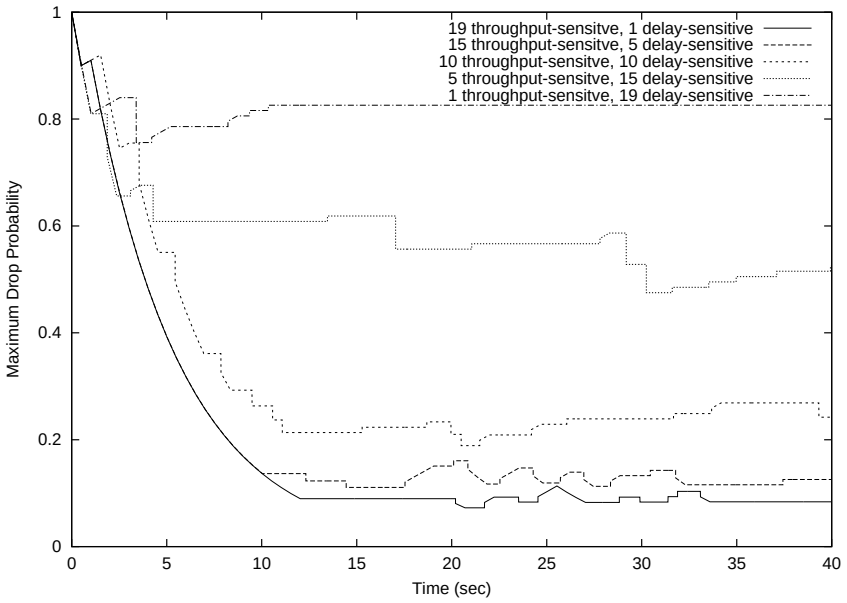

Figure 10: max_p Decrease Analysis

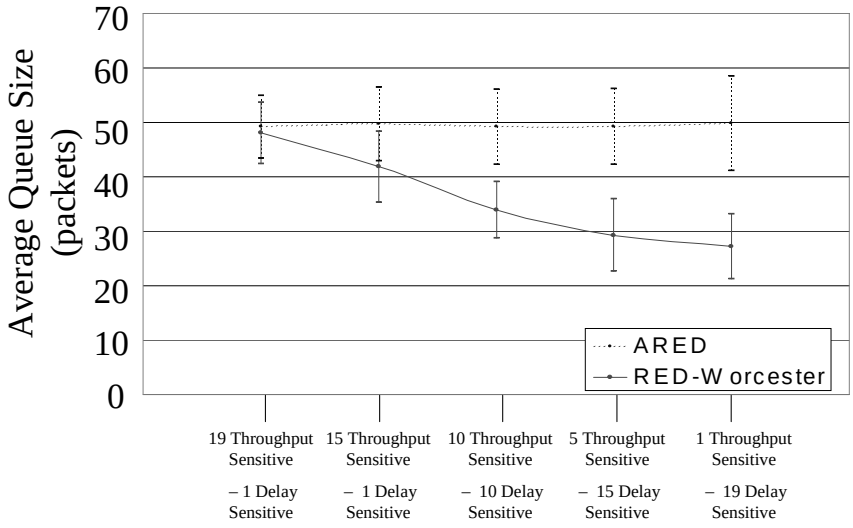

Figure 11: Average queue size in RED-Worcester

Figure 11 graphs average queue size measured in packets for RED-Worcester and ARED. ARED maintains a fixed average queue size half-way between min_{th} and max_{th} irrespective of the incoming traffic mix. With min_{th} set to 20 packets and max_{th} set to 80 packets, ARED's target queue size remains fixed at 50 packets. However, by using a moving target, RED-Worcester adapts its average queue size based on the incoming traffic QoS requirements. Thus, when the incoming traffic has a higher fraction of throughput-

sensitive flows, RED-Worcester maintains a higher average queue size, and when the incoming traffic has a higher fraction of delay-sensitive flows, RED-Worcester maintains a lower average queue size.

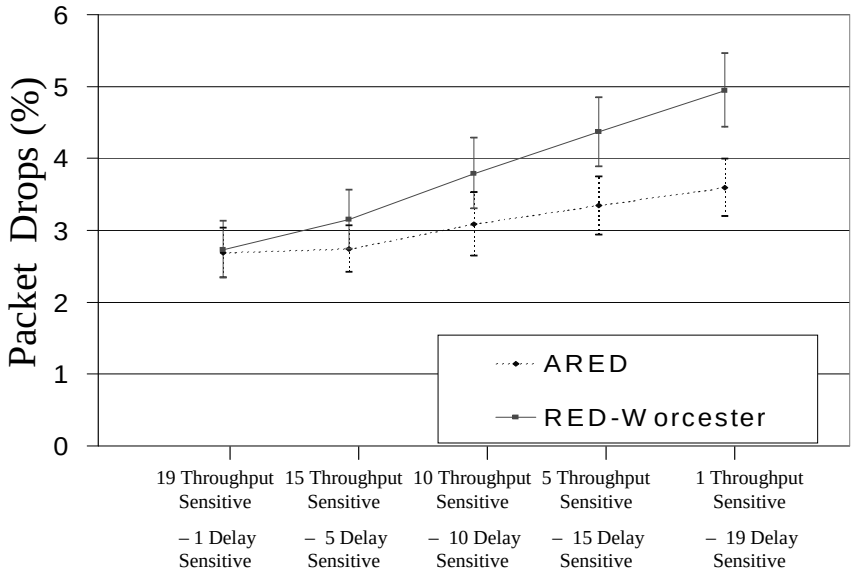

Figure 12: Percent packet drops

Figure 12 presents the average per flow packet drop percentages for RED-Worcester and ARED. The RED-Worcester packet drop percentage increases linearly as the traffic mix changes from mostly throughput-sensitive traffic to mostly delay-sensitive traffic. This is because as the fraction of delay-sensitive flows increases in the incoming traffic mix, RED-Worcester's moving target shifts to smaller average queue sizes requiring a higher drop percentage to maintain. Surprisingly, even in ARED, as the fraction of delay-sensitive flows increases, the drop rate goes up. When the total number of flows is constant and TCP flows are replaced by TFRC flows, ARED has to increase its dropping rate to maintain the same average queue size. This suggests that TFRC flows are not precisely TCP equivalent and may use more bandwidth than TCP flows under some situations.

Figure 13 shows the QoS performance of throughput-sensitive flows in RED-Worcester normalized to the QoS of throughput-sensitive flows in ARED. As noted in Section 3, the QoS for throughput-sensitive flows is calculated as $QoS = T^1/D^0$, since throughput-sensitive flows can tolerate queuing delays. The performance of throughput-sensitive flows suffers slightly in RED-Worcester as compared to ARED. The throughput-sensitive flows in our simulations used a delay hint corresponding to a queue size of 50 packets, which is also the ARED's fixed target queue size. Since RED-Worcester adapts its target based on incoming traffic requirements, RED-Worcester's target queue size is always below ARED's

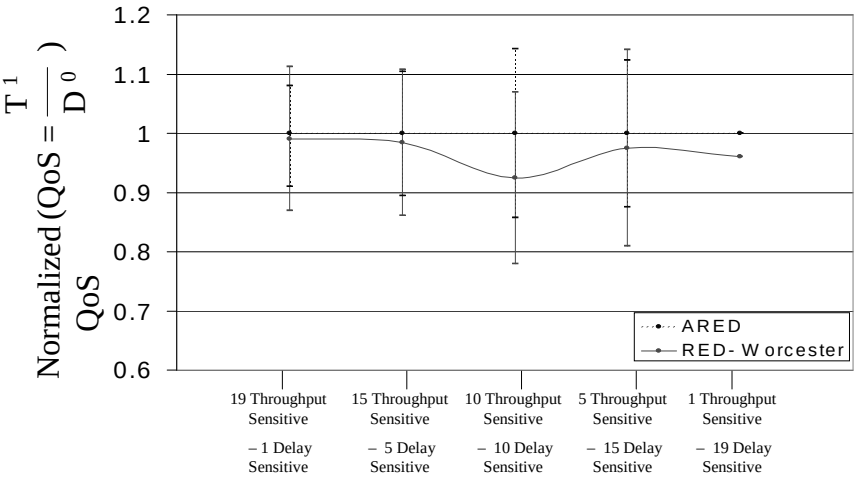

Figure 13: Normalized QoS for Throughput-Sensitive Flows

fixed target in the presence of delay-sensitive flows, thus giving lower throughput performance. Note that if the throughput-sensitive flows had used higher delay hints than ARED's target queue size, then when throughput-sensitive flows were in majority in the incoming traffic mix, their QoS performance would have been better in RED-Worcester than in ARED. RED-Worcester does not favor any particular type of flows, but instead tries to meet the average requirements of the incoming traffic. ARED, on the other hand, due to its fixed target, gives comparatively slightly better performance to throughput sensitive flows than delay sensitive flows in cases with a mixture of traffic.

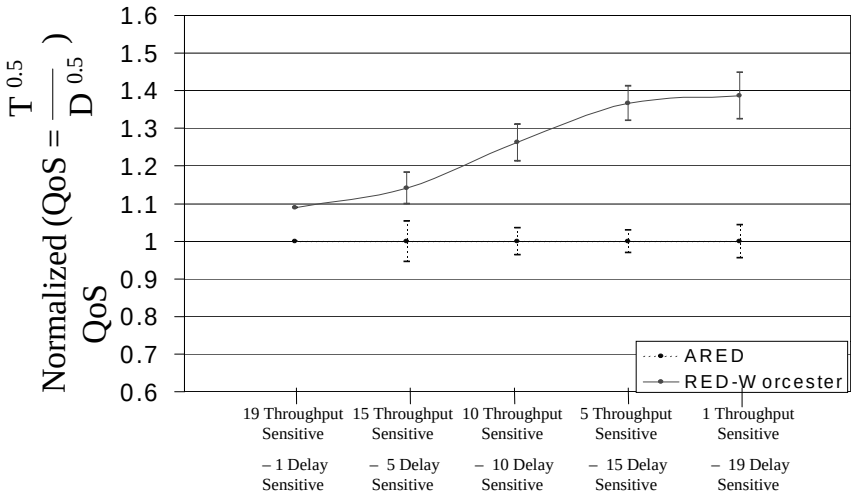

Figure 14: Normalized QoS for Delay-Sensitive Flows

Figure 14 shows the QoS performance of delay-sensitive flows in RED-Worcester normalized to the QoS of delay-sensitive flows in ARED. As noted in Section 3, the QoS

for delay-sensitive flows is calculated as $QoS = T^{0.5}/D^{0.5}$, since delay-sensitive flows can tolerate some reduction in throughput while valuing delay more. QoS for delay-sensitive flows improves in RED-Worcester as the fraction of delay-sensitive flows increases in the incoming traffic mix. RED-Worcester's moving target brings the average queue size down as the fraction of delay-sensitive flows increases, reducing the overall average queuing delays. On the contrary, in ARED, delay-sensitive flows suffer as ARED maintains the average queue size at a fixed target of 50 packets irrespective of the QoS constraints of the incoming traffic.

Our evaluation shows that although RED-Worcester cannot provide precise delay-throughput tradeoffs to individual flows as per their QoS requirements, RED-Worcester router dynamically adjusts the RED operating parameters to better meet the aggregate throughput and delay requirements of the incoming flows than ARED.

5 Summary

This paper presents RED-Worcester, an active queue management technique designed to provide better overall Quality of Service (QoS) support under the current best-effort Internet environment. With RED-Worcester, applications include a delay hint in their packets as an indication of their preference for either lower queuing delay or higher throughput at the router. RED-Worcester maintains an average queue size to match the average delay requirements of all incoming flows. Thus, when most of the incoming flows are throughput-sensitive, RED-Worcester maintains a higher average queue size to provide better throughput, whereas when most of the incoming flows are delay-sensitive, RED-Worcester maintains a lower average queue size to provide low delay service. When there is no majority of any particular type of traffic, RED-Worcester maintains an average queue size to meet the average QoS requirements of the flows.

RED-Worcester does not provide the delay and throughput guarantees provided by IntServ approaches, but nor does RED-Worcester have the inherent scalability difficulties required by such per-flow reservations. RED-Worcester does not pre-define classes of service as do DiffServ approaches, but instead provides a continuum of service classes. The service afforded by RED-Worcester does not require additional monitoring or charging of either flows that take advantage of the low delay service or flows that take advantage of the high throughput service, allowing it to integrate with the current best-effort Internet environment.

We have evaluated RED-Boston with varying percentages of delay-sensitive and throughput-sensitive flows and also with flows having varying delay and throughput requirements. Using a new QoS metric, we find that RED-Worcester is able to adapt the overall router

queue parameters to best suit the overall QoS of the current traffic mix.

Currently, RED-Worcester uses FIFO scheduling on the outgoing queue which provides the same delay for all packets, regardless of the individual QoS requirements. An extension to RED-Worcester could sort the outgoing queue based on their delay constraints and then insert incoming packets into the queue to match their relative delay needs. In order to ensure fairness, a per-packet drop probability would need to be computed to provide higher drop rates to packets that were inserted at the head of the queue and lower packet drop rates to packets that were inserted at the tail of the queue.

Another extension to RED-Worcester could be interaction between cascaded RED-Worcester routers. When dequeuing a packet, RED-Worcester could be modified to update the cumulative amount of time the packet has waited in the router queue, thus providing additional information for trading of delay and throughput for RED-Worcester routers downstream.

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.

[FF99] Sally Floyd and Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, February 1999.

[FGS01] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management. Under submission, http://www.icir.org/floyd/papers/adaptiveRed.pdf, 2001.

[FHPW00] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings of ACM SIGCOMM Conference*, pages 45 – 58, 2000.

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

[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.

[IMD01] G. Iannaccone, M. May, and C. Diot. Aggregate Traffic Performance with Active Queue Management and Drop from Tail. *ACM Computer Communication Review*, July 2001.

[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)*, January25-27 2000.

[LM97] D. Lin and R. Morris. Dynamics of Random Early Detection. In *Proceedings of ACM SIGCOMM Conference*, September 1997.

[oCB] Universiy of California Berkeley. The Network Simulator - ns-2. Interent site `http://www.isi.edu/nsnam/ns/`.

[PCM00] C. Padhye, K. Christensen, and W. Moreno. A New Adaptive FEC Loss Control Algorithm for Voice Over IP Applications. In *Proceedings of IEEE International Performance, Computing and Communication Conference*, February 2000.

[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.

[PW99] K. Park and W. Wang. QoS-Sensitive Transport of Real-Time MPEG Video Using Adaptive Forward Error Correction. In *Proceedings of IEEE Multimedia Systems*, pages 426 – 432, June 1999.

[RHE99] Rezza Rejaie, Mark Handley, and D. Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proceedings of IEEE Infocom*, 1999.

[SSZ98] Ion Stoica, Scott Shenker, and Hui Zhang. *Core*-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *Proceedings of ACM SIGCOMM Conference*, September 1998.

[SZ99] Ion Stoica and Hui Zhang. Providing Guaranteed Services Without Per Flow Management. In *Proceedings of ACM SIGCOMM Conference*, September 1999.

[TZ99] D. Tan and A. Zakhor. Real-time Internet Video Using Error Resilient Scalable Compression and TCP-friendly Transport Protocol. *IEEE Transactions on Multimedia*, May 1999.

[WCZ01] Yubing Wang, Mark Claypool, and Zheng Zuo. An Empirical Study of RealVideo Performance Across the Internet. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[Wro97] J. Wroclawski. The Use of RSVP with IETF Integrated Services. *IETF Request for Comments (RFC) 2210*, September 1997.