PLAYBACK-BUFFER EQUALIZATION FOR STREAMING MEDIA USING STATELESS TRANSPORT PRIORITIZATION

Wai-tian Tan[†]*, Weidong Cui*[‡]*, John G. Apostolopoulos*[†]

[†]Streaming Media Systems Group
Hewlett-Packard Laboratories, Palo Alto, CA

[‡]EECS Department, University of California, Berkeley, CA

ABSTRACT

Streaming media applications are plagued with playback disruptions due to client buffer underflow resulting from fluctuations in network throughput. When some clients are experiencing buffer underflow and disruptions, other clients may have excess buffered data. This suggests that disruptions can be reduced by the redistribution of resources to provide higher throughput to streaming sessions with less buffered data, that is, to sessions with greater need. This paper proposes a scheme that achieves an effective or end-to-end result of the above dynamic resource allocation without the use of either explicit per-session resource reservation or across-session coordination. Instead, each packet is stamped at the source with the buffer occupancy of the session, and prioritized scheduling is performed at the resource bottleneck to favor packets with smaller buffer occupancy. Due to the actions of end-to-end congestion control, higher throughput can then be maintained for streaming sessions with less buffer. Experimental and simulation results show that the proposed scheme can increase the mean time between playback disruptions by a factor of 4 for fixed initial buffer time, or reduce the required initial buffer by a factor of 2.5 for fixed probability of hitting a playback disruption.

1. INTRODUCTION

In streaming media applications, media data has to be continuously available at the client to prevent highly objectionable disruption or stalling in playback. Due to possible fluctuations in the available bandwidth for a streaming media session, buffering is typically employed at the client to reduce the frequency of playback disruption. Nevertheless, for a number of streaming media sessions sharing a bottleneck link, it is typical that some sessions may experience buffer depletion, and therefore suffer playback disruption. Perhaps surprisingly, while some streaming sessions may be experiencing buffer depletion, other sessions operating over the same bottleneck link may have plenty of buffered data. This could be due to either the fluctuation of available capacity on the bottleneck link or different network conditions experienced by streaming sessions. This striking unbalance suggests that a reduction in playback disruption may be achieved by explicitly or implicitly allocating more instantaneous bandwidth resources to media sessions with less buffered data.

The motivation for this work was to reduce the spread, or equalize the playback buffer for streaming media sessions sharing a bottleneck link. In such a way, the aggregate number of playback disruptions can be reduced. We refer to this as Playback Buffer Equalization (PBE). The paper proposes a scheme that achieves an effective or end-to-end result of dynamically allocating more resources to streaming media sessions with more critical need of resources without using explicit network reservation. This is accomplished by exploiting the dependence of the transmission rate of media sessions on the characteristics of the transport environments. Specifically, our approach begins by inserting in each packet a label corresponding to the buffer occupancy of the streaming session the packet belongs to. For resource bottlenecks that provide two or more types of delivery service of different grade, joint prioritization across all streaming sessions is performed at the resource bottleneck using only the labels in the packets. In such a way, streaming sessions with little playback buffer can be transmitted using a higher grade service, and thereby indirectly trigger the end-to-end congestion control mechanism to operate at higher throughput at the expense of streaming sessions that already have plenty of buffer. Since the transport prioritization at the resource bottleneck is based on labels carried in the packets, but without any other streaming or transport session information, we call our scheme Playback-Buffer Equalization using Stateless Transport Prioritization (PBE-STP). Some advantages of PBE-STP are as follows. First, no client or protocol modification - streaming clients and protocols that are already widely deployed today can be used directly without requiring global upgrade. Second, existing QoS mechanisms, such as the differentiated services

model (DiffServ) [1] of the Internet, can be employed since the scheme does not require networks to possess media-specific knowledge, or be modified in other ways. Third, the multiple streaming sessions may be delivered from multiple streaming servers which may potentially be located in different places, and no coordination among the streaming servers or even the streaming sessions from a single server is required. In addition, unlike some approaches, no frequent setup/teardown or reconfiguration of the QoS network is required since prioritization of resources is across all media sessions sharing the bottleneck resource.

PBE-STP may be beneficial in a number of practical scenarios involving multiple users behind a single resource bottleneck. One possible example is multiple cable modem users sharing the same connection to an Internet service provider. Another example is campus networks where many workstations are potentially sharing only a few connections to the Internet.

The remainder of the paper is organized as follows. We first survey in Section 2 related approaches to mitigate the effects of playback disruption in streaming media applications. We then describe our proposed scheme in Section 3. Evaluation of some possible implementations of the general design is given in Section 4, followed by additional simulation results in Section 5. Finally, a conclusion is given in Section 6.

2. RELATED WORK

One of the major challenges of streaming media is guaranteeing that media data is continuously available at the client. To this end, many advances have been made, including those assuming only a best-effort network, and those assuming infrastructure support in the form of network QoS [2] or intelligent resource management [3]. In this section, we first summarize prior work for best-effort networks, which are complementary with our proposed scheme and can be used in conjunction. We then describe prior work that exploits infrastructure support and discuss their advantages and disadvantages. Finally, we survey related works that target different problems but with some relevance to our work.

2.1. Related Work for Best-effort Networks

Client buffering: Current streaming systems, including those from Microsoft and Real Networks, implement congestion control whereby a streaming server 1) estimates the available bandwidth of a media streaming session based on prevalent network conditions, and 2) streams the media according to the estimated available bandwidth. Since the available bandwidth fluctuates with time, if no buffering is used then playback disruption would occur if the instantaneous available bandwidth is lower than the media rate. To reduce the number of playback disruptions, existing streaming clients typically employ a buffer of 5 to 15 seconds. With such buffering, a client can pre-fetch data that is not immediately needed when the available bandwidth is above the media rate, and then drains the buffer when the available bandwidth is below the media rate. In such a way, playback disruption will not occur when the available bandwidth is temporarily below the media rate, unless the buffer is also empty.

There have been a variety of strategies proposed to improve the effectiveness of client buffering, including slowing down the media playout rate at the client to reduce its consumption rate and help prevent buffer underflow [4].

Multi-rate switching: Another complementary technique to buffering is multi-rate switching whereby multiple copies of the same content at different bit-rates are made available. Early implementations of streaming media systems coded the same content at a few strategic media rates targeted for common connection speeds (e.g. one for dialup modem and one for DSL/cable) and allowed the client to choose the appropriate media rate at the beginning of the session. However, these early systems only allowed the media rate to be chosen once at the beginning of each session. In contrast, multi-rate switching enables dynamic switching between different media rates during a single streaming session. This mid-session switching between different media rates enables better adaptation to longer-term fluctuations in available bandwidth than can be achieved by the use of the client buffer alone. Examples include Intelligent Streaming from Microsoft and SureStream from Real Networks [5].

Layered or scalable compression: A more elegant approach to adapt to bandwidth fluctuations is to use layered or scalable compression. This is similar in spirit to multi-rate switching, but instead of producing multiple copies of the same content at different bit-rate, layered compression produces a set of (ordered) bitstreams (sometimes referred to as layers) and different subsets of these bitstreams can be selected to represent the media at different target bit-rates [6]. Many commonly used compression standards, such as MPEG-2, MPEG-4 and H.263 have extensions for layered coding. Nevertheless, layered or scalable approaches are typically not used because they incur a significant compression penalty as compared to non-layered/non-scalable approaches.

2.2. Related Work using Infrastructure Support

A typical scheme for streaming media that exploits network QoS support by explicit reservation has the following two characteristics. First, different streaming media sessions negotiate (and potentially re-negotiate over time) resources independently of each other. Second, resource-allocation decisions are often made in an uncoordinated fashion, and are typically based on instantaneous resource availability. Examples of such schemes are [2, 7]. Consequently, there is

no guarantee that a session in critical need of resources can obtain them even when other sessions sharing the same resource bottleneck may have excess resources. Furthermore, the complexity of such schemes is high due to frequent establishment and teardown of reservations.

A related dynamic resource allocation scheme that explicitly provides more bandwidth to streaming sessions with less buffered data has been proposed in [3]. Instead of relying on explicit resource reservation, the scheme simply dynamically allocates the bottleneck resource (wireless bandwidth) according to the need of each streaming session. However this approach requires matching custom-designed streaming clients and custom-designed infrastructure to achieve the desired bandwidth allocation across sessions. Specifically, each of the custom-designed clients transmits a proprietary feedback to the custom-designed infrastructure for every streaming session. The proprietary feedback contains enough information to estimate at the infrastructure the current session state (buffer occupancy). The custom-designed infrastructure then keeps and maintains the states (buffer occupancy) for each session sharing the same wireless bandwidth, and explicitly allocates the available bandwidth according to the "Join the Shortest Queue" (JSQ) policy, where the session with the least buffer (in time) will receive the next available bandwidth resource in the form of code or frequency/time slot depending on the actual wireless system used. In such a way, the system allocates more resources to streaming sessions with more critical needs for bandwidth.

The above scheme provides adaptive resource allocation among multiple sessions, however it has a number of drawbacks: The requirement of a proprietary client feedback means that 1) clients need to support an additional feedback mechanism, and 2) existing clients cannot be used. Unlike server upgrades, client upgrades are typically difficult due to the large number of parties involved. A custom-designed infrastructure is required to implement the specific resource allocation scheme, and to collect feedback from clients. As a result, existing infrastructure cannot be substituted. The custom-designed infrastructure has to maintain up-to-date states for each streaming session, thereby incurring additional complexity in the overall system. This can lead to scalability problems when supporting a large number of sessions.

In [8], a scheme to share bottleneck bandwidth across several TCP flows according to application need is proposed. Specifically, the scheme does not require modifications to the delivery infrastructure, but instead relies on additional out-of-band collaboration of TCP clients to achieve the desired allocation.

2.3. Other Related Work

In a DiffServ framework, the authors of [9] showed an example where traffic prioritization is more effective when based on "flow state" than on application types. Specifically, their goal was to improve the performance of interactive TCP applications such as Telnet and Web. They propose a scheme to give priority at the sender to low-volume interactive traffic by (1) assuming that TCP flows with smaller congestion window size (i.e. TCP flow state) correspond to the interactive TCP applications that they would like to prioritize, and (2) giving those identified flows higher priority at the sender. This work is related to ours in the sense that it is designed to perform traffic prioritization to improve the performance of some session over others. However, the proper priority can be maintained for TCP flows from a single server only, and it is performed at that server. In contrast, with PBE-STP the "flow state" (client buffer in seconds) is embedded in each packet and therefore global prioritization across all flows from multiple servers can be performed at an intermediate resource bottleneck within the network.

Information exchange between data senders and routers can improve end-to-end data transmission. In Explicit Congestion Notification (ECN) [10], routers set the Congestion Experienced (CE) bit in the IP packet header to inform data senders of network congestion. In PBE-STP, the information flows in the opposite direction from data senders to routers.

3. PBE-STP: OVERVIEW AND DESIGN

The general framework of the proposed system is depicted in Fig. 1, and consists of the following elements:

Media Streaming Server that implements any commonly employed streaming media standard and congestion control algorithm. For each streaming sessions, the streaming servers first derive the client buffer occupancy of each session using existing knowledge of the start-time and the amount of data transmitted for that session. A label representing the time-varying Time-To-Depletion for each session is then recorded in each IP packet belonging to that session.

QoS Network that offers at least two types of services that differ in end-to-end packet loss rate, delay, or both. Example includes Asynchronous Transfer Mode (ATM) networks, Integrated Services (IntServ) and Differentiated Services (DiffServ) Internet.

Packet Classifier performs assignment of each packet to the different available types of service based only on the label contained in a packet.

Streaming Clients that are compatible with the associated Media Streaming Server. They are not aware of the presence and actions of the packet classifier above.

In Fig. 1, a number of streaming servers, A to K, are streaming to clients, 1 to N, through the same resource bottleneck in a QoS Network. Packets are marked with

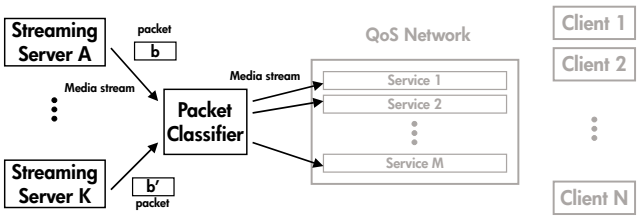

Fig. 1. *Architecture of the proposed scheme. The network and streaming clients, which do not require modifications, are shown shaded.*

labels corresponding to the time-to-depletion of their respective session at the servers. The packet classifier then classifies each packet into one of the M available service classes based only on the packet's label. Specifically, packets with labels corresponding to a smaller time-to-depletion are preferentially given better service. Transmitting packets of a session using a better service has a number of effects. First, the direct effect is that the packets themselves will be delivered faster, or dropped with a lower probability. Second, an important but indirect effect is that sessions whose packets are transmitted using a better service will effectively observe a "better" channel, and by virtue of their congestion control, boost their transmission rates, as illustrated by Fig. 2. Despite the fact that congestion control mechanisms in the streaming servers and clients are not aware of the existence of the proposed techniques, the preferential assignment of a better service to sessions with less client buffer will trigger higher transmission rates to sessions with greater need. Note that since the assignment of labels to the M network services is performed locally at the resource bottleneck, global optimization for all streams across different servers can be achieved.

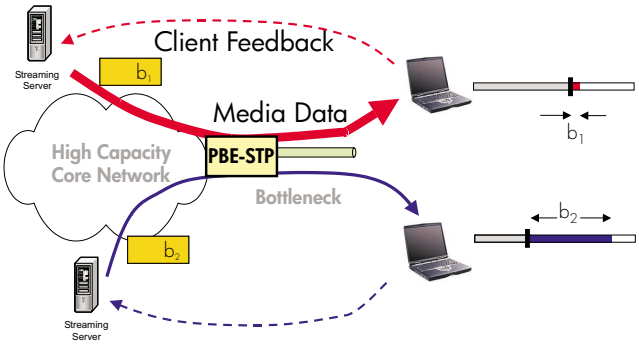

Fig. 2. *PBE-STP exploits behaviour of congestion control mechanisms to achieve preferential allocation of resources.*

Typical congestion control mechanisms rely on end-to-end measurable quantities such as round-trip propagation delay and packet loss statistic to infer the level of congestion in the transmission path. When the congestion level is high, the typical action is to lower the transmission rate so that through the collective actions of all sessions, the level of congestion can be controlled. Generally, higher end-to-end propagation delay and higher packet loss rates are associated with higher levels of congestion. For instance, it is well known that for TCP flows, the goodput (G), or the rate at which data can be reliably transmitted, is related to the round-trip propagation delay RTT, the packet loss rate p, and the maximum transmittable unit size MTU, roughly by [11]:

$$G = \frac{1.22 \cdot MTU}{RTT \cdot \sqrt{p}} \quad (1)$$

Despite the popularity of TCP-friendly congestion control for streaming media, which explicitly attempts to transmit at the same average rate as TCP, the actual congestion control mechanisms employed by streaming media systems may be different. Nevertheless, the general action is still to transmit at a lower rate with increasing round-trip time or packet loss rate, and vice versa. As a result, the proposed method can be applied to general congestion control mechanisms as well.

For the purpose of this paper, we focus on video sessions that are conducted using TCP only. This allows us to focus on the key issues, without becoming entangled in all the variations of TCP-friendly congestion control. Also note that TCP is the protocol commonly employed for HTTP streaming.

4. PBE-STP: EVALUATING THREE IMPLEMENTATIONS

In this section, we describe evaluation results of three specific implementations of PBE-STP using a physical network test-bed depicted in Fig. 3. Specifically, we consider the scenario where 9 flows of constant bit-rate 1.2 $Mbps$ and duration 300 seconds are streamed from the servers in Fig. 3 to clients in the receiver machine through a 10 $Mbps$ bottleneck at node R. Six of the flows start at time 0 and the remaining three flows start at times 60, 120, and 180 seconds, respectively. A 3-bit label representing the time-varying Time-To-Depletion for each session is then recorded in the Type-Of-Service (TOS) field of each IP packet belonging to that session. Time-To-Depletion for a streaming session is the time for which media playback can be sustained using only data that has already been buffered (playback buffer occupancy). The meaning of the label is given in Fig. 4. The bottleneck node R is a Linux machine that implements various traffic differentiation schemes to be described next.

4.1. Baseline: First-In-First-Out

As a baseline for comparison, we describe the results for the scenario described above when the Linux box R implements

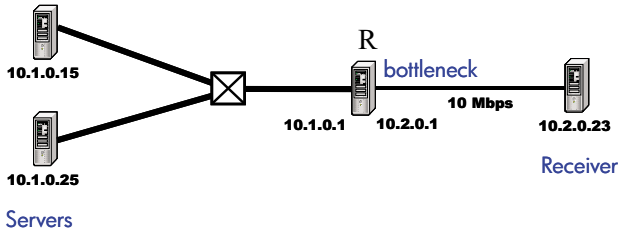

Fig. 3. *Physical test-bed for evaluating PBE-STP implementations. All links besides the bottleneck link are 100 $Mbps$.*

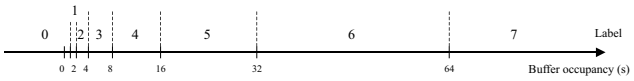

Fig. 4. *Mapping between client playback buffer occupancy and the 3-bit label carried in the TOS field of IP header.*

simple first-in-first-out (FIFO) forwarding. The top graph of Fig. 5 shows the amount of data received by the different streaming sessions in a typical run of the scenario. The amount of data is reported in seconds, and is the ratio of the amount of received data in bits to the bit-rate of the media. Media playback starts five-seconds after streaming starts (i.e. 5 sec pre-roll buffer), and the dotted lines show for each stream the amount of data that needs to be consumed as a function of time to maintain the streaming session with no playback disruption. In other words, playback disruption occurs when the solid line crosses the corresponding dotted line, as indicated by the red triangular markers in Fig. 5. After a playback disruption, a client would pause and rebuffer for 5 seconds before resuming playback. The difference between the solid and the dotted lines of a streaming session is the client buffer occupancy, which is shown in the lower plot of Fig. 5. We see that there is generally a very large spread in the amount of client buffer across sessions that start at different times. Between time 0 to 60 seconds, there are only 6 sessions sharing the bottleneck link, and all sessions are building up their client buffer. At about time 150 seconds, there are 8 flows sharing the bottleneck link, and none is gaining in buffer occupancy. Finally, when the last session starts at time 180 seconds, the throughput that each of the flows receives is lower than the media rate, and they are all draining their client buffers. The last stream, with the least buffer build-up from uncongested times, is the first to experience playback disruption, followed by the second to last stream. Furthermore, we note that the last two sessions are experiencing playback disruptions at a time when the other sessions have ample amounts of buffered data at the clients. The goal of PBE-STP is to exploit such spread to provide higher throughput to sessions with less buffer at the expense of sessions with more buffer.

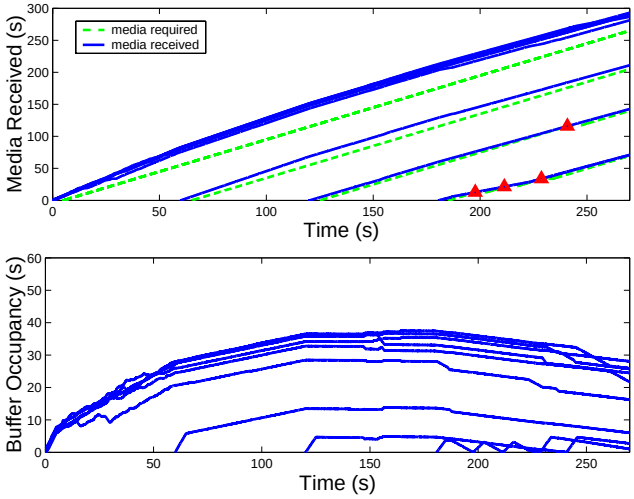

Fig. 5. *Example illustrating the situation under conventional FIFO. Media sessions that start later (when bandwidth is constrainted) have much less client buffer built up as compared to media sessions that start earlier (when bandwidth is available), and therefore are afflicted by numerous playback disruptions.*

4.2. Strict Priority Queues

One possible scheme for prioritizing traffic flows with different labels is to use simple priority queues, where the number of queue equals the number of label. In such a way, it is guaranteed that packets with labels of higher priority are always delivered first. We employed a Linux routing daemon implementing an 8-class priority queue at node R of Fig. 3. Specifically, the routing daemon examines the 3-bit label in the TOS field of the IP header without considering per-flow information, and simply assigns the packet to one of the eight queues corresponding to the eight possible labels. The results of a typical run using priority queues are given in Fig. 6. Comparing the bottom plots of Figs. 5 and 6, we notice that using priority queues, PBE-STP is effective in bringing the client buffer occupancy of new sessions to a similar level as the other existing sessions in a short time. Specifically, under priority queue, when new sessions start at times 60, 120 and 180 seconds, they receive large instantaneous throughput to boost their client buffer occupancy, as shown by the steep slopes in the bottom plot of Fig. 6. In contrast, under FIFO later sessions never attain the same client buffer occupancy as earlier sessions.

We observe that playback disruptions are still present under PBE-STP with strict priority queues. But unlike the FIFO case, disruptions do not concentrate in later sessions, but can appear in any session regardless of starting time. Such disruptions are typically preceded by long periods of zero throughput as indicated by horizontally flat lines in the

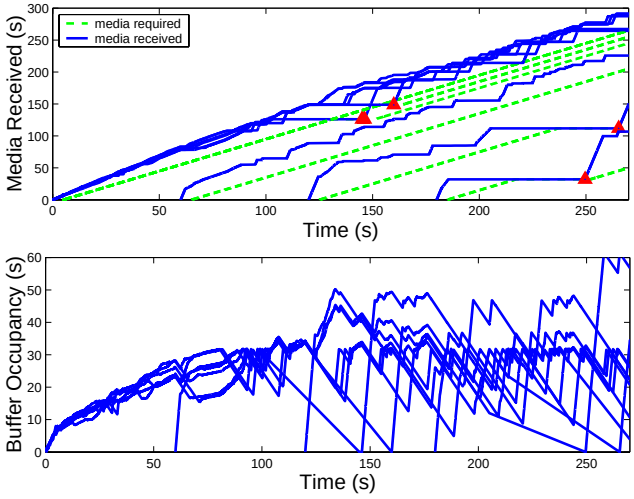

Fig. 6. *Using priority queues, PBE-STP allows sessions starting at different times to achieve roughly the same client buffer occupancy. Nevertheless, long periods of zero throughput are observed.*

upper plot of Fig. 6. Such complete loss of throughput for extended periods is due to a combination of TCP behavior and starvation caused by priority queuing. Specifically, a session with low priority may temporarily receive zero throughput due to preemption by higher priority traffic. This in effect may trigger the exponential back-off algorithm of TCP [12] which causes a TCP source to suspend transmission *up to 64 seconds* even when resources may become available before then. For instance, in the top plot of Fig. 6, the last playback disruption at around time 250 seconds is preceded by a 64-second period of zero throughput.

4.3. Premium + Best Effort Services

To avoid triggering TCP exponential back-off and starvation in general, one general approach is to limit the amount of high-priority traffic. In this section, we consider the practical setting of having two types of traffic, premium or high priority traffic that has restricted throughput, and best-effort or low priority traffic. In the context of Fig. 3, we employed a routing daemon implementing a two class priority queue at R where the throughput of the high priority queue is restricted to λ^*. Without distinguishing packets from different sessions, the routing daemon simply looks up the label, b, in the TOS field of each packet, and compares it to a classification threshold, T. If $b \leq T$, the packet will be delivered using the Premium service. Otherwise, the packet will be delivered using the Best-Effort service. For each of the eight labels b, we maintain the running average of traffic load λ_b for that label using the rate estimation technique in [13]. The classification threshold T is then adjusted so that the aggregate high priority traffic will not exceed λ^*:

$$T = \max \left\{ L \;\middle|\; \sum_{b=0}^{L} \lambda_b \leq \lambda^* \right\} \tag{2}$$

Note that the Packet Classifier does not need to keep states for each media session, but only aggregate statistics across all sessions, such as the average traffic rate for each label, and the classification threshold.

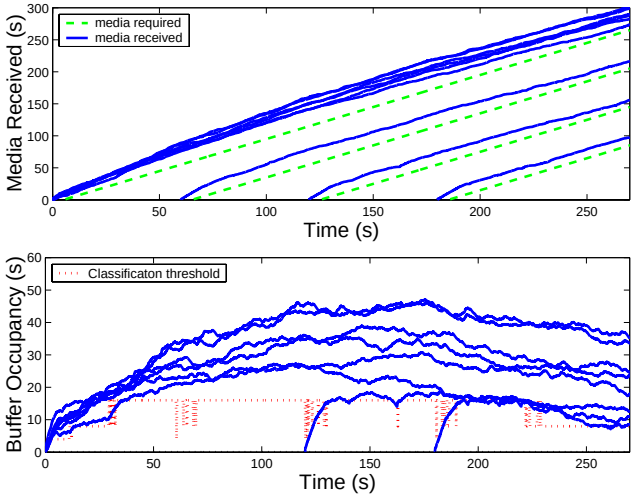

Fig. 7. *Using a maximum of 30% premium traffic, PBE-STP reduces playback disruptions, but maintains large spread in the amount of client buffer occupancy.*

The corresponding results for λ^* of 3 $Mbps$ are shown in Fig. 7. We observe two improvements from FIFO and priority queuing. First, no playback disruption is observed. Second, extended periods of zero throughput observed in Section 4.2 are not present. Nevertheless, there is generally a rather large spread in the level of client buffer occupancy across the different sessions. The dashed line in the bottom plot of Fig. 7 indicates the classification threshold at different times. When a new stream starts, it is classified as premium due to its low client buffer occupancy. As a result high instantaneous throughput is achieved which eventually lead to a buffer occupancy level above the classification threshold. Nevertheless, in *steady-state* operation, apart from the short times when new flows are injected, all sessions are classified as best-effort, and the system effectively behaves as FIFO. This is because we quantized the buffer occupancy according to Fig. 4. When the lowest label with significant amount of traffic has a rate above our target high priority traffic rate, λ^* or 3 $Mbps$, the rule according to Equation 2 is to not transmit high priority traffic. Despite the fact that the PBE-STP is behaving as FIFO most of the time, the ability to provide an initial build-up of buffer for new flows effectively reduces playback disruption.

Generally, it would be desirable to assign packets to "fill the high priority pipe". However, the simple deterministic classification rule given in Equation 2 may fail to fully utilize the amount of high priority traffic that is available, λ^*. One possible implementation to increase utilization is to randomly assign packets with label T to "fill the pipe", i.e., until the amount of high priority traffic meets λ^* [9]. It turns out that such an implementation may lead to lower overall throughput. This is because traffic for TCP flows with label T will be transmitted in both the low and high priority classes, resulting in large random jitter that decreases TCP throughput. Specifically, a TCP flow that *consistently* receives better QoS can sustain a higher throughput, however the same statement may not be true if only a portion of a flow experiences better QoS. This motivates our use of Equation 2 which guarantees consistency in packet classification when the threshold is constant.

4.4. Hybrid Priority Queues

The two-class priority queue in Section 4.3 enjoys only limited success. The primary drawbacks include the overhead to estimate traffic load corresponding to different labels, and more importantly, the steady-state FIFO behavior discussed in Section 4.2. Another approach to prevent starvation is to employ hybrid priority queues that switch between round-robin and priority queues in a periodic fashion. Specifically in the context of Fig. 3, we consider a hybrid queuing discipline in node R that maintains 8 queues corresponding to the 8 possible labels. Packets are served according to two alternating rules so that every K packets served according to priority queuing are followed by one packet served according to round-robin. The corresponding results for $K = 15$ are shown in Fig. 8. Compared to approaches in Sections 4.1 to 4.3, there is generally less playback disruptions, and the spread in client buffer occupancy is much smaller. The remaining client buffer spread in the bottom plot of Fig. 8 is due to the quantization effect of label assignment according to Fig. 4, where packets for flows with 16 to 32 seconds of client buffer are assigned an identical label.

5. QUANTITATIVE RESULTS

In this section, we quantify the performance improvement of PBE-STP over FIFO queuing. Since the effectiveness of TCP feedback control scheme depends strongly on propagation delay, we are interested in comparing PBE-STP against FIFO under different amounts of propagation delay in addition to different levels of network load. The hybrid queue implementation of PBE-STP in Section 4.4 is employed. In order to gain precise control of propagation delay, results in this section are generated using the discrete-event network simulator *ns-2* [14].

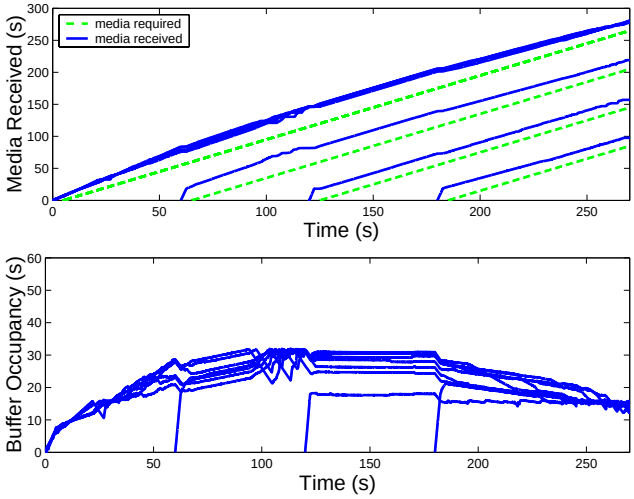

Fig. 8. *Using hybrid priority queue, PBE-STP achieves small spread in client buffer occupancy as well as few overall number of playback disruptions.*

5.1. Simulation Setup

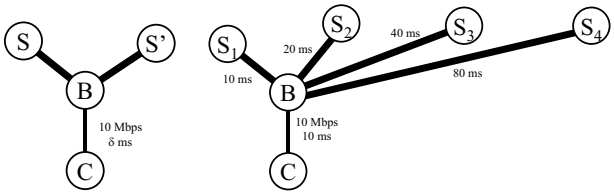

Fig. 9. *Simulation topologies. Unless otherwise indicated, links are 100 $Mbps$ and with negligible delay.*

The network topologies in Fig. 9 are used for simulations in this section. Media traffic originating from various nodes with initial designation of S is streamed via TCP through a common resource bottleneck B to clients in node C. PBE-STP is implemented at node B. For media traffic, we assume Poisson arrival of intensity α new sessions per second. For each new session, we choose a constant media bit-rate from 100, 300, and 500 $kbps$ uniformly at random. The media duration is drawn uniformly according to an empirical distribution obtained from actual media access logs of over 1.2 million accesses from HP Media Solutions. By varying the arrival intensity α, we can achieve different levels of *Loading Factor*, which is the average aggregate streaming rate to the bottleneck bandwidth. Note that loading factor is an average quantity, so that short timescale congestion may still occur at loading factors below 1, and vice versa.

Since it may take a long time for the media traffic to build up and reach "steady state", we introduce a batch of

media traffic starting at time 0 to model the effect of media flows that started at negative times. Note that the initial traffic will have a bit-rate distribution identical to that of the regular media traffic. Assuming independence, the expected number of initial streaming sessions N can be calculated as the product of the average duration d and the arrival rate α of the regular media traffic. Given the PDF of the duration of regular media flow f (given by the empirical distribution discussed above), it can be shown that the PDF for the duration distribution of the initial media flows, f_i, is given by:

$$f_i(x) = \int_x^\infty \frac{1}{y} f(y) dy$$

All traffic is media traffic carried by TCP and no unresponsive UDP flows are employed for two reasons. First, the presence of unresponsive flows unduly complicates fair comparison between FIFO and PBE-STP, since any difference in quality of service received by unresponsive flows under the two schemes has to be accounted for. Second, the empirical distribution of media access duration discussed above has long tail with concentration of very short accesses. As a result, the generated traffic naturally contains both short and long time-scale fluctuation in load.

The streaming clients implement a five second initial buffer (pre-roll buffer) so that playback will start after a five second delay at startup. Also, each time a playback disruption occurs another five second delay for re-buffering will take place.

5.2. Simulation Results

The following two metrics are used to quantify the performance advantage of PBE-STP over FIFO forwarding:

Disruption Frequency: the number of playback disruptions a media session is expected to experience per second of media playback. For each experiment, we denote the number of playback disruptions and the duration for flow i by n_i and d_i, respectively. We approximate the disruption frequency, F, by:

$$F = \frac{\sum_i n_i}{\sum_i d_i}$$

Required Initial Buffer-time (RIB): the amount of initial buffer time that would allow 95% of flows to finish without encountering any playback disruption. At a given load, this measure signifies the proper amount of initial buffer time, and thus wait-time, that should be used to achieve 95% flows without disruption, had we known the network condition beforehand. For flow i, we denote by r_i^k the time when frame k is received, and by p_i^k the time when frame k is needed for playback. Then the initial buffer time b_i that is required for flow i to avoid playback disruption can be calculated as follows.

$$b_i = \max_k \{(r_i^k - p_i^k)^+\}$$

where

$$(r_i^k - p_i^k)^+ = \max\{r_i^k - p_i^k, 0\}$$

The *Required Initial Buffer-time* is then calculated as the 95 percentile of b_i.

For each test condition, two tests are performed where the exact same media traffic is used to guarantee fair comparison between PBE-STP and FIFO. This means that the set of starting times, durations, and bit-rates for all the media flows are identical for corresponding PBE-STP and FIFO experiments. All simulations are performed for one hour of simulated time.

5.2.1. *Different Network Load*

Using the topology of Fig. 9-(a) with a bottleneck link delay δ of 10 ms, and with sources randomly placed at nodes S and S', we performed 66 independent simulations of one hour simulated time each, at different loading factors between 0.92 and 1.06. For each of the 66 simulations, each consisting of hundreds of media streams, a single disruption frequency and required initial buffer time (RIB) are reported to summarize the service quality at that particular loading factor. The results are given in Fig. 10. We make two observations. First, at loading factors below unity, PBE-STP out-performs FIFO on almost every case using identical media traffic. Specifically, the average disruption frequency at loading factors below one is 0.0072 and 0.0016 for FIFO and PBE-STP, respectively. This means the expected time

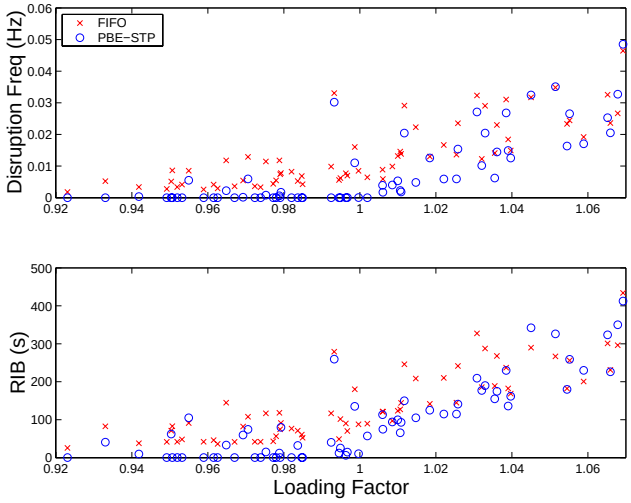

Fig. 10. *PBE-STP achieves significantly lower Disruption Frequency and Required Initial Buffer-time (RIB) than FIFO at loading levels less than unity.*

between playback disruptions is about $1/0.0072$ or 139 seconds for FIFO and 625 seconds for PBE-STP – a 4.5 times improvement. Similarly, at the same range of loading factor, the average RIB for FIFO and PBE-STP are 79 and 28.5 seconds, respectively – a 2.8 times improvement. Second, at loading factor above unity, PBE-STP still outperforms FIFO on average, but the performance improvement diminishes and there are more experiments in which FIFO performs better. Specifically, only an average improvement of 33% and 16% are observed for disruption frequency and RIB, respectively. For loading factors above unity, it is expected that PBE-STP will provide small performance improvements over FIFO, since PBE-STP can neither create resources nor reduce demand. Instead, it realizes intelligent resource management to provide more throughput to streams with higher need at the expense of streams with less need. At the highly loaded regime, very few streams can afford to receive reduced throughput, and the effectiveness of PBE-STP diminishes.

5.2.2. *Different Network Delay*

Since the effectiveness of TCP to effect changes in transmission rate decreases with increasing delay, it is expected that the performance of PBE-STP would as well. To compare the performance of PBE-STP with FIFO at different network delays, we repeat the simulation experiments in Section 5.2.1 with different values of network delay. Specifically, for the topology in Fig. 9-(a), experiments are performed for δ of 5, 10, 20, 40, 80, 160, 320 and 640 ms. The results are summarized in Fig. 11. We see that generally the disruption frequency increases for both PBE-STP and FIFO as propagation delay δ increases. Specifically, when δ is increased from 5 to 640 ms, the disruption frequencies for both PBE-STP and FIFO increase roughly twenty times. Nevertheless, PBE-STP constantly maintains significantly lower disruption frequency. In Fig. 12, we plot the ratio of the average disruption frequency for PBE-STP and FIFO, as a function of different values of delay δ. Separate ratios are reported for loading factors above and below unity. We see that despite the absolute performance degradation associated with increasing feedback latency, the relative performance improvement in disruption frequency stays relatively constant.

5.2.3. *Streams with Mixed Delay*

We next compare the behavior of PBE-STP against FIFO when multiple streaming sessions with different end-to-end network delay are sharing a common resource bottleneck. Specifically, we use the topology of Fig. 9-(b), where media traffic is being generated independently at source nodes S_1 to S_4. The average amount of traffic from the 4 source nodes is the same, and different loading factors can be achieved by

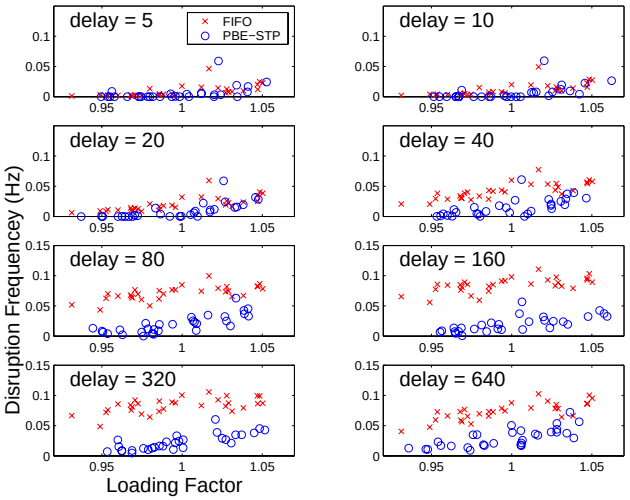

Fig. 11. *Disruption Frequency for FIFO and PBE-STP for different one-way propagation delays.*

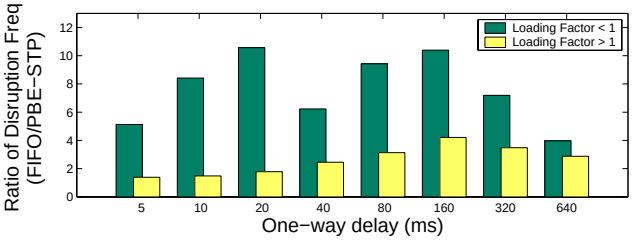

Fig. 12. *Relative performance improvement of PBE-STP over FIFO, as a function of one-way propagation delay.*

increasing the arrival intensity of new flows α, as described in Section 5.1. As before, 66 simulations of one hour duration each are performed at different loading factors between 0.92 and 1.08. For each loading factor, 4 average numbers of disruption frequency are reported, one for each source node S_1 to S_4. The results are shown in Fig. 13. We see that under FIFO, streaming sessions that originate from the distant node S_4 consistently receive poor service. Even at lower loading factors near 0.92, streams from S_4 experience a disruption frequency of 0.05, or roughly a playback disruption every 20 seconds. This is due to the dependency of TCP throughput on round-trip time given by Equation 1, which indirectly assigns higher throughput to streaming sessions with low round-trip-time at the expense of sessions with higher round-trip-time. Such throughput assignment without regard to application needs causes streams from S_4 to suffer playback disruptions even when many streams from S_1 have plenty of buffered data at the playback client. In contrast, under PBE-STP, *the effective or resulting allocation of bottleneck throughput is dependent on application need*. As a result, at loading factors below unity, playback

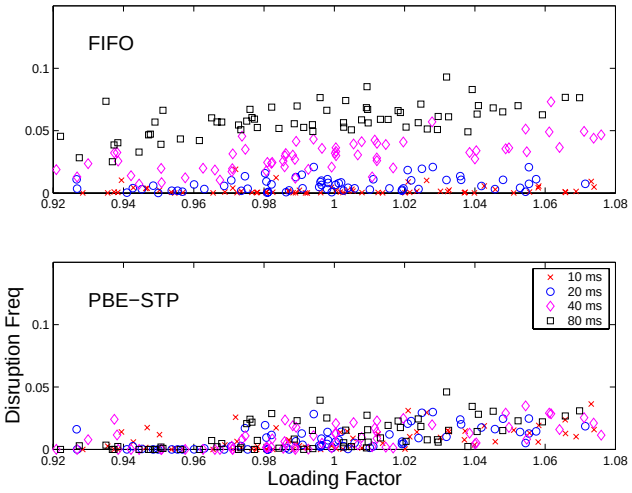

Fig. 13. *Disruption frequencies for streams with different end-to-end delays.*

disruptions are greatly suppressed (as compared to FIFO). At loading factors above unity, PBE-STP maintains roughly uniform service for streams from different source nodes. With FIFO, on the other hand, streams from S_1 continue to enjoy better service at the expense of streams from S_3 and S_4, even when the loading factor exceeds unity.

6. SUMMARY

This paper proposes a system that enables multiple streaming sessions sharing a common resource bottleneck to dynamically receive different amounts of resources based on their respective application needs, where we define the need of each client as the time to buffer underflow and playback disruption. Specifically, each packet of a streaming media session is labeled at the sender with the buffer occupancy of the client, and prioritized scheduling is performed at the resource bottleneck to preferentially transmit packets of sessions with smaller buffer occupancy. This approach provides a method of classifying packets from all streaming sessions in a stateless manner into one of the service classes to effect the desired dynamic resource allocation, and specifically without requiring either explicit per-session resource reservation or across-session (e.g. across multiple senders) coordination. Using this approach, sessions with lower client buffer occupancy receive higher throughput and thereby significantly reduce the number of playback disruptions. Specifically, our experimental results indicate that, as compared to conventional FIFO, our proposed approach increases the mean time between playback disruptions by a factor of 4 for fixed initial buffer time, or reduces the required initial buffer time by a factor of 2.5 for fixed probability of hitting a playback disruption.

7. REFERENCES

[1] S. Blake et al., "An architecture for differentiated services," RFC 2475, http://www.ietf.org/rfc/rfc2475.txt.

[2] M. Mirhakkak, N. Schult, and D. Thomson, "Dynamic bandwidth management and adaptive applications for a variable bandwidth wireless environment," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 1984–1997, Oct 2001.

[3] F. Fitzek and M. Reisslein, "Prefetching protocol for continuous media streaming in wireless environments," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 2015–2028, Oct 2001.

[4] E. Steinbach, N. Färber, and B. Girod, "Adaptive playout for low latency video streaming," *IEEE ICIP*, October 2001.

[5] Conklin, Greenbaum, Lillevold, Lippman, and Reznik, "Video coding for streaming media delivery on the internet," *IEEE Transactions on Circuits and Systems for Video Technology*, March 2001.

[6] W. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. Multimedia*, pp. 172–186, June 1999.

[7] M. Grossglauser et al., "RCBR: A simple and efficient service for multiple time-scale traffic," in *SIGCOMM*, August 1995.

[8] P. Mehra, A. Zakhor, and C. Vleeschouwer, "Receiver-driven bandwidth sharing for TCP," in *INFOCOM*, 2003.

[9] W. Noureddine and F. Tobagi, "Improving the performance of interactive TCP applications using service differentiation," in *IEEE Infocom*, June 2002.

[10] K.K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 2, pp. 158–181, May 1990.

[11] M. Mathis et al., "The macroscopic behavior of the tcp congestion avoidance algorithm," *ACM Computer Communications Review*, pp. 67–82, July 1997.

[12] W. Stevens, *TCP/IP Illustrated*, Addison-Wesley, 1994.

[13] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *ACM SIGCOMM*, September 1999.

[14] "Network simulator 2," http://www.isi.edu/nsnam/ns.