Multi-layer Active Queue Management and Congestion Control for Scalable Video Streaming

[1]Seong-ryong Kang, [1]Yueping Zhang, [2]Min Dai, [1]Dmitri Loguinov *
Texas A&M University, College Station, TX 77843
[1]{skang, yueping, dmitri}@cs.tamu.edu, [2]min@ee.tamu.edu

Abstract

Video streaming is becoming an increasingly important part of the present Internet. To guarantee a high-quality streaming environment to end users, many video applications require a strict form of network QoS that is not available in the present Internet. Thus, to supplement the best-effort model of existing networks, we study a new video streaming framework that allows applications to mark their own packets with different priority and use multi-queue congestion control inside routers to effectively drop the less-important packets during buffer overflows. We describe priority AQM algorithms that provide "optimal" performance to video applications under arbitrary network loss and study a variation of Kelly's congestion control in combination with our framework. We call the combined architecture PELS – Partitioned Enhancement Layer Streaming.

1. Introduction

Typical video applications transport multimedia data that are highly sensitive to quality-of-service (QoS) characteristics (e.g., delay or packet loss) of their end-to-end path and often require better than simply best-effort services from the network before they can offer a high-quality streaming environment to end users. In response to this demand, significant research effort went into improving the best-effort model of the current Internet [2, 4, 6, 7, 10, 32].

One dimension of this related work supplements the best-effort model with network QoS that guarantees a "better than best-effort" performance to end flows (these methods loosely fall under the umbrella of DiffServ [2, 4]). The other, more recent dimension includes various Active Queue Management (AQM) algorithms [6, 7, 10, 11, 20, 32] that are able to provide QoS services to the flows with much less overhead than the more traditional mechanisms like DiffServ or IntServ [3].

However, none of the existing QoS methods provide a scalable, low-overhead, low-delay, and retransmission-free platform required by many current real-time streaming applications. To fill this void, this paper investigates novel AQM algorithms that not only can provide a provably "optimal" performance under random loss, but also possess very low implementation complexity.

* This work was supported in part by the NSF under grant ANI-0312461.

One of the characteristics of video packets that does not match the best-effort service is that they often carry information of *different* importance. Thus, video applications can clearly differentiate between the more-important and the less-important packets. In all layered video coding schemes, the base layer is more important than the enhancement layer. Furthermore, the lower sections of the enhancement layer are more important than the higher sections because their loss renders all dependent data virtually useless. Thus, treating all video packets equally (as in the current best-effort Internet) usually leads to significant quality degradation during packet loss and low useful throughput during congestion, both of which cause video streaming to become unappealing in practical settings.

With the presence of unequal importance among video packets, the first goal of this work is to achieve "high end-user utility," which means that the majority of packets that are transmitted across the bottleneck link must carry *useful* information that can be decoded by the receiver. In video applications that use motion compensation and variable-length coding (VLC), a single lost packet in the base layer may affect several frames and render them all useless even though some of them arrive to the receiver without any loss. Furthermore, the enhancement layer is not immune to packet loss either since strong dependence between the coded data allows packet loss to affect consecutive chunks of data that are significantly larger than those actually lost in the network. Hence, even under moderate packet loss, the bottleneck link may be used to transmit a large number of packets that eventually get dropped by the decoder.

In addition to high utility, many interactive applications (such as video telephony) further require low end-to-end delays to deliver high application-layer performance to the user. Additional problems with delays arise during retransmission of lost packets since all video frames have strict decoding deadlines. During heavy congestion (especially along paths with large buffers), the RTT is often so high that even the *retransmitted* packets are dropped in the same congested queues [21]. As a result, the receiver in such scenarios must ask for multiple retransmissions of each lost packet, which often causes the retransmitted packets to miss their decoding deadlines. Thus, our second goal is to provide a *retransmission-free* network service to video flows. This direction generally aligns well with FEC-based approaches, except our goal is to avoid all bandwidth overhead associated with error-correcting codes and occupy network channels only with the actual video data.

To improve the quality of video delivered over the Internet, we investigate a new streaming framework in which each application marks its own packets with different priorities and uses AQM inside routers to effectively drop the less-important packets during congestion. Such preferential (instead of random) dropping of packets allows the application to maintain a much higher quality of video for the end user compared to similar scenarios in a best-effort network. We also find that the use of multi-queue AQM allows scalable video applications to maintain high useful link utilization *without retransmitting any of the lost packets* or sending any error-correcting codes. Thus, we achieve both goals of high utility and low end-to-end delay.

While our implementation relies on Kelly's utility-based controllers [17], it is important to realize that the proposed framework can be used with any congestion control (including end-to-end methods such as AIMD, TFRC, or even TCP) and can be deployed in the current Internet with minimum modifications to the existing infrastructure.

The rest of paper is organized as follows. Section 2 discusses the background and related work. Section 3 presents an analysis of video streaming in best-effort and AQM environments. Section 4 describes the proposed video streaming framework including FGS-layer packet marking techniques and optimal router queue management mechanisms. Section 5 studies congestion control applied to the proposed framework. Section 6 discusses simulation results and Section 7 concludes the paper.

2. Background and Related Work

Many studies are conducted to improve the best-effort model of the current Internet by supplementing it with network QoS. Some of them focus on AQM [6, 7, 10, 32] that provides unequal treatment to flows, while others range from offering hard guarantees in the form of IntServ [3] to more scalable models such as DiffServ [2, 4]. We briefly overview some of the more recent and promising approaches.

2.1. Priority QoS Methods

Several studies investigate the performance of video streaming over the DiffServ architecture. Gurses *et al.* [12] use a temporally-scalable H.263+ video scheme and three-color markers (TCM) that allow ingress routers to promote packets (i.e., increase their priority). However, this work does not employ congestion control or allow the end flows to benefit from unequal priority of the packets since DiffServ can arbitrarily remark them according to ingress/egress policies of peering ISPs. Shin *et al.* [30, 31] study the problem of "optimal" assignment of relative priority indexes to video packets depending on their impact on the quality of received video. Besides using a fairly complex packet prioritization scheme, the work does not use congestion control or discuss how the network should treat marked packets. Zhao *et al.* [35] employ MPEG-4 FGS for video streaming and use several computationally intensive packet prioritization schemes, but also without studying network support of the proposed architecture.

Among non-DiffServ methods, Kuzmanovic *et al.* [19] propose TCP-LP, which provides a TCP-like low-priority service that seeks out bandwidth left-over from the high-priority streams. Tang *et al.* [33] present a video streaming protocol that uses low-priority dummy packets to probe for new bandwidth. The dummy packets are sent only upon packet loss and only for the duration of one round-trip delay. Hurley *et al.* [13] propose ABE (Alternative Best Effort) that requires applications to choose between two conflicting types of service (i.e., low delay or low packet loss). A similar approach is used in BEDS (Best Effort Differentiated Service) [8].

Finally, we should note that Internet-2's QBSS (QBone Scavenger Service) [28] is similar to our approach as it provides service differentiation by allowing end flows to mark their own packets with the low-priority bit. However, the current QBSS does not support more than two priorities or directly benefit video traffic.

2.2. Active Queue Management

Active Queue Management (AQM) schemes perform special operations in the router to achieve better performance for end flows. These operations include dropping random packets (e.g., RED), re-arranging the order in which packets are served (e.g., WFQ), and randomly marking packets from more aggressive flows (e.g., ECN). While WFQ focuses on providing fairness to competing flows [6, 32], RED/ECN attempt to avoid congestion by randomly dropping or marking packets with a certain probability that increases with the level of congestion [7, 10, 11]. As such, these methods are not specifically tailored to multimedia applications and thus cannot directly improve video quality of Internet streaming.

Additional studies combine congestion control with AQM to provide robust and smooth controllers since routers can detect network conditions more accurately than end systems. Lapsley *et al.* [20] study optimization-based congestion control and propose router-based Random Early Marking (REM) that works with cooperating end-flows to maximize their individual utilities. Katabi *et al.* [16] present XCP (eXplicit Congestion notification Protocol) that conveys information about the degree of congestion in network paths to application sources using two separate AQM controllers for utilization and fairness. Several other studies include Kelly's optimization methods [14, 17, 18, 26] and Low's work [22, 23, 24, 25].

2.3. Structure of MPEG-4 FGS

Recall that FGS (Fine Granular Scalability) [29] is the streaming profile of the ISO/IEC MPEG-4 standard, which is a method of compressing residual video signals into a single enhancement layer that provides a flexible and low-overhead

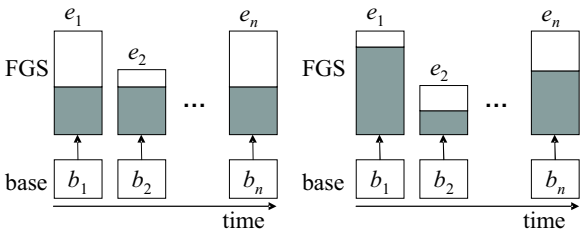

Figure 1. Scaling of MPEG-4 FGS using fixed-size (left) and variable-size (right) frames.

foundation for scaling the enhancement layer to match variable network capacity during streaming. The FGS layer is typically coded at some fixed (very large) bitrate R_{max} and can be re-scaled to any desired bitrate by discarding a certain fraction of each FGS frame.

Fig. 1 illustrates the operation of MPEG-4 FGS. The figure shows individual frames from the base layer and the corresponding FGS layer. The shaded parts of the enhancement layer are the fractions of each frame taken by the server as part of its rate-scaling mechanism during streaming. Depending on the optimization goals of the server, it can transmit a fixed fraction of each frame or use rate-distortion (R-D) models to dynamically determine the desired amount of data in each frame (interested readers are referred to [5] for more details).

3. Analysis of Video Streaming

In the first part of this section, we investigate probabilistic characteristics of video streaming performance under random packet loss. We study a best-effort network, in which routers drop video packets uniformly and randomly during congestion. Recall that many studies of Internet QoS attempt to improve TCP performance by changing drop behavior of the network from bursty to uniformly random [10, 11]. Thus, it can be argued that future networks will deploy such packet drop mechanisms more often than the current Internet. Therefore, we assume an independent loss model with exponential tails of burst-length distributions (rather than a heavy-tailed model, which is commonly observed in FIFO queues) and use it throughout the paper.

In the second part of this section, we overcome the drastic reduction of video quality in best-effort streaming and show that *preferential* packet drops can in fact provide "optimal" performance to the end-user. Thus, following the best-effort analysis, we study priority-based AQM that supports preferential streaming and compare it with the best-effort scheme.

Finally, we should note that although quality degradation of multimedia streaming in best-effort networks is well documented, the novelty of this section lies in the derivation of the exact closed-form expressions for the penalty inflicted on scalable flows under uniform packet loss and the novel associated discussion that is also useful for understanding "optimality" of AQM in later sections.

3.1. Best-Effort Streaming

In this section, we investigate the effect of random packet drops on video quality using the example of MPEG-4 FGS (similar results apply to non-FGS layered coding)[1]. We start by examining the probabilistic characteristics of packet drops in an FGS frame, derive the expected amount of *useful* data recovered from each frame, and define the effectiveness of FGS transmission over a lossy channel.

Assume that long-term network packet loss p can be modeled by a sequence of independent Bernoulli random variables X_i. Each X_i is an indicator function that determines whether packet i is lost or not: $X_i = 1$ *iff* packet i is dropped in the network. Then $P(X_i = 1) = 1 - P(X_i = 0) = E[X_i] = p$ is the average packet loss. Even though this model is a great simplification of real networks and results in the probability of obtaining a burst of length k proportional to e^{-k} (i.e., the tail of burst sizes is exponential), it suffices for our purposes (see the discussion on RED/ECN earlier in this section).

Next assume that FGS frame sizes H_j are measured in packets and are given by *i.i.d.* random variables with a probability mass function (PMF) $q_j = P(H_j = k), k = 1, 2, \ldots$. The exact distribution of $\{H_j\}$ depends on the frame rate, variation in scene complexity, and the bitrate of the sequence. The question we address next is what is the expected amount of useful packets that the receiver can decode from each frame under p–percent random loss? Thus, our goal is to determine the expectation of Y_j, which is the number of consecutively received packets in a frame j.

Lemma 1: Assuming independent Bernoulli packet loss with probability p, the expected number of useful packets in an FGS frame is:

$$E[Y_j] = \frac{1-p}{p} \sum_{k=1}^{\infty} \left(1 - (1-p)^k\right) q_k. \quad (1)$$

Proof: See [15]. ■

Throughout the rest of the paper, we examine one particular distribution of $\{H_j\}$, in which all FGS frames have the same fixed size H. Under these conditions, (1) becomes:

$$E[Y_j] = \frac{1-p}{p} \left(1 - (1-p)^H\right). \quad (2)$$

This model is compared to actual simulation results in Table 1 for $H = 100$. As the table shows, even under a reasonably low packet loss of 1%, the expected number of useful packets in each frame is only 62; however, the decoder successfully receives (on average) a total of 99 packets per frame. Furthermore, under moderate loss of 10%, only 9 useful packets are recovered from each frame, while a total of 90 packets per frame are transmitted over the bottleneck link.

1 Further note that motion-compensated enhancement layers suffer even more degradations under best-effort loss and are not modeled in this work. However, the expected amount of improvement from QoS in such schemes is even higher than that in FGS.

H	Packet loss p	Simulations	Model (2)
100	0.0001	99.49	99.49
100	0.01	62.78	62.76
100	0.1	8.99	8.99

Table 1. Expected number of useful packets.

Furthermore, as streaming rates become higher (and H becomes larger), $E[Y_j]$ tends to $(1-p)/p$ and the recovered (useful) percentage of each frame tends to zero. This is shown in Fig. 2 (left) for $p = 0.1$, in which the number of useful packets in the best-effort case quickly saturates at $(1-p)/p = 9$ as H becomes large. The same side of the figure also plots the number of packets that could have been recovered in the "optimal" case, where all $H(1-p)$ packets are useful in decoding (which is clearly the best possible scenario under p–percent packet loss).

To quantify the effect of FGS packet transmission on video quality, we define *utility* U of received FGS video as the ratio of the average number of FGS packets used in decoding a video frame (i.e., $E[Y_j]$) to the total number of received FGS packets (i.e., $H - pH$):

$$U = \frac{E[Y_j]}{H(1-p)} = \frac{1-(1-p)^H}{Hp}, \qquad (3)$$

where the last expansion holds for the constant frame size model in (2). For instance, we get $U = 0.1$ with p = 0.1 and H = 100, which means that only 10% of the received FGS packets are useful in enhancing the base layer. This result is further illustrated in Fig. 2 (right), which plots the utility of best-effort streaming and the "optimal" utility for different values of H and $p = 0.1$. As the figure shows, the utility of best-effort video drops to zero inverse proportionally to the value of H, which means that as $H \to \infty$ (i.e., sending rates become higher), the decoder receives "junk" data with probability 1.

3.2. Optimal Preferential Streaming

In this section, we discuss the "optimal" streaming method that can provide high end-user utility and significant quality improvement along AQM-enabled network paths. In order to achieve the maximum end-user utility (i.e., $U = 1$), routers must drop the upper parts of the FGS layer during congestion and transmit only the lower parts since consecutive lower portions of the FGS layer can enhance the base layer, while any gaps in the delivered data typically render the remainder of the layer useless. Fig. 3 depicts the difference between the ideal and random drop patterns in an enhancement frame and shows that all dropped packets must occupy the upper portion of the FGS layer to achieve optimality.

Since for a given drop rate p, the "optimal" AQM scheme drops pH packets from the upper part of the FGS frame and protects the remaining $H(1-p)$ packets, all received FGS packets are consecutive and thus can be used to enhance the base layer. Hence, the utility of this framework is always one regardless of the values of p or H. For example, assuming the same scenario as in the best-effort case ($p = 0.1$, $H = 100$), preferential streaming delivers *ten times more useful packets* than best-effort streaming. The main question now is whether optimal streaming is possible in practice and how to achieve it using scalable AQM methods. We address this issue next.

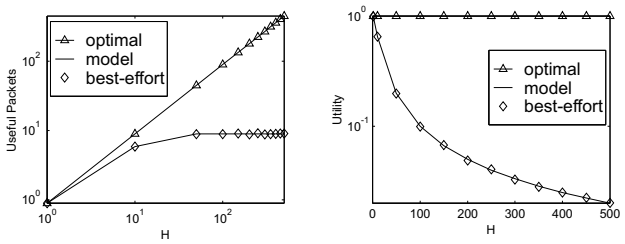

Figure 2. The number of useful FGS packets in each frame (left). Utility of received video (right).

4. Preferential Video Streaming Framework

In this section, we introduce a new video streaming framework called *Partitioned Enhancement Layer Streaming* (PELS) that operates in conjunction with priority-queuing AQM routers in network paths. In the PELS framework, applications partition the enhancement layer into two layers and voluntarily mark their packets using different priority classes, allowing the network routers to discriminate between the packets based on their priority (no per-flow management is required).

Recall that coded video frames carry information that has different importance to the end user – the lower layers are more sensitive to packet loss than the higher layers. The base layer (being most sensitive) is *required* for displaying video appropriately at the receiver and thus is transmitted using the highest priority class. This ensures that the base layer is dropped only when the entire FGS layer is discarded by the routers.

The reason for splitting the FGS layer into two priorities is also simple to understand. Bytes in the lower part of the FGS layer are more important than those in the higher part because the former includes the information needed to properly decode the latter. Due to this nature of FGS streams, dropping packets randomly (as in the best-effort network) does not properly protect the lower parts of FGS even under moderate congestion. Hence, to protect the lower portions of FGS frames and drop the upper parts, preferential treatment of not only the base layer, but also the enhancement layer is highly desirable.

4.1. Router Queue Management

In this section, we discuss queuing disciplines necessary to support PELS and how applications should assign priority to their packets. To separate video traffic from the rest of the

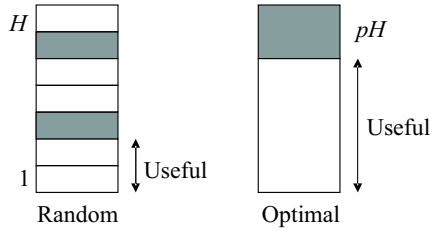

Figure 3. Useful data in each frame under random (left) and ideal (right) loss patterns.

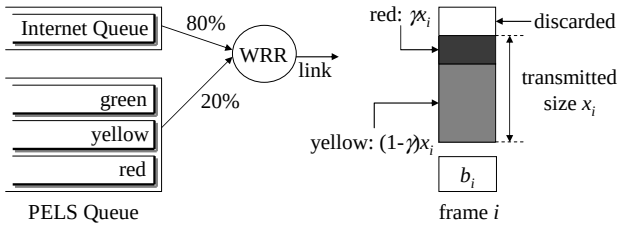

Figure 4. Router queues for PELS framework (left). Partitioning of the FGS layer into two layers and PELS coloring of FGS packets (right).

flows, the proposed PELS architecture must maintain in each network router two types of queues – the PELS queue and the Internet queue. The PELS queue is further subdivided into green, yellow, and red priority queues to service marked multimedia packets, while the Internet queue serves all other (non-multimedia) Internet traffic in a regular FIFO fashion. To ensure that network bandwidth is shared "fairly" between PELS applications and other Internet traffic, we employ weighted round-robin (WRR) scheduling between the PELS and Internet queues. Recall that WRR can provide a desired level of fairness between several types of traffic by allocating a certain fraction of the outgoing link to each queue as shown in Fig. 4 (left). This allows de-centralized administrative flexibility in selecting the weights and assigning proper "importance" to different classes of traffic.

It is easy to see that the PELS queue must employ a *strict* priority queuing discipline to maximize the resulting video quality for a given total throughput budget. Since the higher parts of the FGS frame cannot be decoded without the presence of the lower parts, each router has no reason to transmit the higher parts before sending the lower ones. This implies that network routers must use queuing mechanisms that do not allow low-priority packets to pass until *all* high-priority packets are fully transmitted. Note that, in general, strict priority queueing is frowned upon since it leads to starvation in low-priority queues and denial-of-service effects for certain flows; however, this situation does not arise in PELS since each flow sends a certain amount of high-priority (i.e., green) packets and always receives non-negligible service from the network. In fact, starvation in low-priority (i.e., red) queues is equivalent to 100% loss in these queues and has very little effect on the resulting quality since it affects only the upper parts of each enhancement frame (more on this below).

Since PELS application sources can arbitrarily mark their packets, we must next ensure that no end-user gains anything by marking *all* of its FGS packets with high priority (i.e., green). Such "misbehaving" sources will increase congestion in the green queues, which will result in (uniform) random losses in their base layers and will quickly degrade the resulting quality of their own video. Similarly, end-flows have little incentive in sending too many yellow packets or being congestion-indifferent. Thus, if each application is a selfish, independent entity that attempts to maximize the utility of its video at the receiver, it will send red packets to probe for congestion and back-off (i.e., reduce the total sending rate) during the loss of any red packets to protect the yellow/green queues from upcoming congestion.

We should make several other interesting observations. First, notice that PELS assumes certain stationarity of the end-to-end path (all packets take the same route) and the presence of PELS-enabled AQM at the *bottleneck* router. The former assumption is common to all flows using congestion control (i.e., multi-path routing and/or route changes make the control loop produce unpredictable results). The latter assumption is very relaxed since it does not require *all* routers to deploy PELS at the same time. Our second observation is that priority queuing in PELS is low-overhead, flexible, does not require support from DiffServ or use of per-flow management, and can be implemented using priority queues available in many existing router hardware/software solutions. Finally, PELS does not require communication between routers and leaves the decisions of how to mark packets to the end-user (i.e., pushes complexity outside the network).

4.2. FGS Partitioning and Packet Coloring

In a practical network environment (such as the Internet), packet loss and available bandwidth are not constant and change dynamically depending on cross traffic, link quality, routing updates, etc. Hence, streaming servers must often probe for newly available bandwidth as part of congestion control and continuously send low-priority packets under the assumption that these probes (and only they) will get lost during congestion.

Fig. 4 (right) illustrates one possible partitioning of FGS bytes into two priority classes (i.e., yellow and red) that can achieve "optimal" utility discussed in section 3.2. The figure shows that the server sends x_i bytes from each enhancement frame i (where x_i is given by congestion control and is derived from R_{max} using rate scaling algorithms [5]). The transmitted section of each FGS frame is divided into two segments – the lower segment of size $(1-\gamma)x_i$ is all yellow and the upper segment of size γx_i is all red. The division into red and yellow packets depends on how conservative (many red pack-

ets and large γ) or optimistic (few red packets and small γ) the server wants to be.

In an ideal network with stationary packet loss p in the enhancement layer, the server can select γ such that γx_i is equal to px_i. This will ensure that *all* red packets are lost and that exactly $(1-p)x_i$ yellow packets are recovered for decoding (this is the best scenario under any circumstances). In practice, however, keeping red packet loss p_R at 100% is not feasible since any slight increase in p (caused by a new flow joining the network or change in network conditions) will "spill" the loss into the yellow queue, effectively creating a best-effort FIFO situation in the yellow queue. Thus, proper and dynamic selection of γ is important (see the next section).

The other issue to address is congestion control. Even though red queues can be used to isolate increasing packet loss p without reducing the sending rate of the flow (i.e., by proportionally increasing γ), the resulting situation will lead to "trashing" the network with numerous red packets that eventually get dropped at the bottleneck link. To prevent waste of bandwidth on the path to the bottleneck, the server must implement elastic congestion control and reduce its rate whenever it loses either yellow or red packets (the loss of green packets means that there is not enough bandwidth to support the base layer and no meaningful streaming can continue). Since all flows in our model use PELS and the same congestion control, they all back-off during the loss of red packets and keep the amount of "waste" to a minimum.

4.3. Selection of γ

Recall that partitioning of the FGS layer into yellow/red packets attempts to ensure that only the upper sections of each frame are dropped during congestion; however, the performance of PELS depends on the selection of γ and the level of congestion at the bottleneck link. In order for PELS to be effective, we must ensure that when flows probe for new bandwidth, they do not incur such high levels of congestion as to force packet loss in the yellow priority queue. Hence, given any control interval k with packet loss $p(k)$ in the FGS layer, how can the server make sure that there will be no loss among *yellow* packets during interval $k+1$?

Intuitively, γ should be adjusted according to packet loss measured during interval k to keep the resulting red loss $p_R = px_i/\gamma x_i = p/\gamma$ at a certain threshold p_{thr}. The most optimistic approach suggests $p_{thr} \approx 1$ (which leads to the largest utility $U \approx 1$) and the most pessimistic approach keeps $p_{thr} \approx p$ (which leads to the best-effort utility in the enhancement layer).

Based upon these observations, we seek a middle ground in which p_{thr} can be stabilized between 70 and 90% using simple closed-loop control methods that adjust γ based on the following rules:

- Increase γ when p increases
- Decrease γ when p decreases.

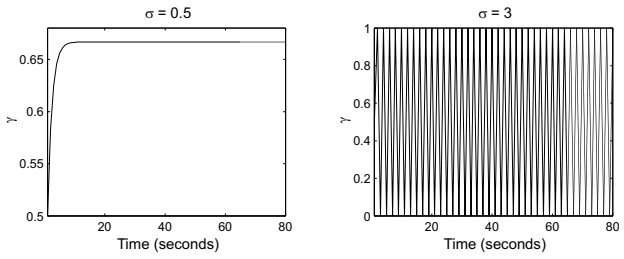

Figure 5. Stability of γ with different σ.

Considering this general intuition, we next investigate a single proportional controller that adjusts γ based on the measured packet loss $p(k)$ and target red packet loss p_{thr}:

$$\gamma_i(k) = \gamma_i(k-1) + \sigma(p_i(k-1)/p_{thr} - \gamma_i(k-1)), \quad (4)$$

where index i represents flow number, $p_i(k)$ is the measured *average* packet loss in the entire FGS layer for flow i during interval k, and σ is controller's gain parameter. Note that $(1-p_{thr})\gamma x_i$ is the amount of cushion left by the server for the yellow packets. For example, $p_{thr} = 0.75$ means that 25% of the red queue works to protect the yellow queue against sudden (unexpected) increase in packet loss.

Note that, in general, the measurement of $p_i(k)$ is coupled with congestion control and should be provided by its feedback loop (we discuss this in section 5). Next notice that the controller in (4) is stable if the following is satisfied.

Lemma 2: The controller (4) is stable *iff* $0 < \sigma < 2$.

Proof: See [15]. ■

Note that in a real network environment, feedback delays are often involved. Assuming arbitrary round-trip delay D_i for flow i, (4) becomes:

$$\gamma_i(k) = \gamma_i(k-D_i) + \sigma(p_i(k-D_i)/p_{thr} - \gamma_i(k-D_i)). \quad (5)$$

Then we have a stronger version of the previous lemma that shows stability of the resulting controller under arbitrary delays.

Lemma 3: The controller (5) is stable *iff* $0 < \sigma < 2$.

Proof: See [15]. ■

Next, we derive the effect that (4)-(5) have on the packet loss in the red queues.

Lemma 4: Assuming stationary packet loss p, both controllers (4)-(5) converge red packet loss p_R to p_{thr}.

Proof: See [15]. ■

To illustrate that selection of σ is important, but not drastically difficult, Fig. 5 depicts the behavior of $\gamma(k)$ with different σ (we use a heavy-loss case with $p = 0.5$ and $p_{thr} = 0.75$ in this example). As the figure shows, $\gamma(k)$ is stabilized at the stationary point $\gamma^* = p/p_{thr} \approx 0.67$ when σ = 0.5, while it is unstable when σ = 3.

The resulting utility of received video in PELS under dynamically changing γ is lower-bounded by the following (assuming that only yellow packets are recovered from the FGS layer):

$$U \geq \frac{H(1-\gamma)}{H(1-p)} = \frac{1-p/p_{thr}}{1-p}. \quad (6)$$

Thus, the utility of PELS is at least 0.96 for $p = 0.1$ and $p_{thr} = 0.75$ and at least 0.996 for $p = 0.01$ and the same threshold. Although PELS does not achieve "optimality" for $p_{thr} < 1$, it comes very close to it and at the same time avoids the pitfalls of the optimal method.

5. Congestion Control for Video

Congestion control is necessary for streaming applications to provide a high level of video quality to end users and avoid wasting network resources with packets that are eventually dropped in congested queues. Many control methods dynamically adjust the sending rate of end-flows based on network feedback and aim to achieve a stable tradeoff between underutilization of resources and network congestion (i.e., packet loss).

Recent studies have focused on developing smooth congestion control schemes for multimedia streaming (e.g., TFRC [9] and binomial algorithms [1]) after AIMD (Additive Increase, Multiplicative Decrease) was unofficially found to be "unacceptable" for video streaming due to its large rate fluctuations. Nonetheless, these new control schemes often do not have stationary points in the operating range of typical applications and continuously oscillate [34].

Among many recent game-theoretic and optimization methods [14, 17, 18, 22, 23, 24, 25, 26], we selected Kelly's congestion control framework (called *proportional fairness* [17]), since it is stable, efficient, and fair under various network conditions. In this section, we study Kelly's controls, apply them to PELS streaming, and investigate whether their performance provides the necessary foundation for achieving our goals of smooth, high-quality video streaming.

In general, it is important to remember that PELS is *independent* of congestion control and can be utilized with any end-to-end or AQM scheme. Thus, the complexity of implementing Kelly controls inside routers should be de-coupled from that of PELS since the latter does not require the presence of the former. Kelly controls are studied here as an example of one possible scheme that supplements PELS with smoothly changing rates. We leave the study of additional methods for future work.

5.1. Continuous-Feedback Control

Although Kelly's controls have attracted significant attention, their application to video streaming is limited to [5] in which Dai *et al*. use an application-friendly form of the controller given by:

$$\frac{dr(t)}{dt} = \alpha - \beta p(t) r(t), \quad (7)$$

where $r(t)$ is the rate of the flow, $p(t)$ is packet loss (feedback from the network), α and β are gain parameters. Since in real applications, rate adjustment is not continuous, we use a discrete form of (7). However, notice that the classical discrete Kelly control studied by [14] and others shows stability problems when the feedback delay becomes large [34]. Hence, we employ a slightly modified discrete version of this framework called *Max-min Kelly Control* (MKC) [34]:

$$r_i(k) = r_i(k - D_i) + \alpha - \beta r_i(k - D_i) p_l(k - D_i^{\leftarrow}), \quad (8)$$

where $r_i(k)$ is the rate of source i during interval k, $D_i^{\leftarrow}$ is the backward delay from the router to source i, D_i is the round trip delay of flow i, and packet loss p_l is fed back from the *most-congested* resource l (this provides max-min resource allocation instead of proportionally fair). The packet loss is computed inside router l at discrete intervals and inserted into all passing packets:

$$p_l(k) = \frac{\sum_{j \in S_l} r_j(k - D_j^{\rightarrow}) - C_l}{\sum_{j \in S_l} r_j(k - D_j^{\rightarrow})}, \quad (9)$$

where S_l is the set of sources sending packets through router l, $D_j^{\rightarrow}$ is the forward delay from source j to the router, and C_l is link capacity of router l. The stability of system (8)-(9) is formalized as follows.

Lemma 5: System (8)-(9) is stable under heterogeneous delays *iff* $0 < \beta < 2$.

Proof: See [34]. ■

We apply (8) for rate control in PELS streaming and investigate its control characteristics including:

- Convergence to a single stationary point
- Fairness between flows.

From (8) and (9), we next derive stationary rates r_i^* of end flows in the equilibrium point and show that (8) has no oscillations in the steady state.

Lemma 6: Regardless of the feedback delay, the stationary rate r_i^* of each flow is:

$$r_i^* = \frac{C_l}{N} + \frac{\alpha}{\beta}. \quad (10)$$

Proof: See [15]. ■

Thus, unlike AIMD or TCP, MKC does not penalize flows with higher RTT and further converges to a single stationary point with no oscillation.

Next notice that priority queueing in PELS imposes increased delays on red packets and that the utilization of each priority class directly affects delay characteristics of *all* queues with lower priority. Since green packets have much smaller queuing delays than yellow or red packets, it is tempting to provide feedback only in green packets. However, since the base layer is sent at significantly lower rates than the enhancement layer, this method introduces unnecessary feedback delays due to large inter-packet spacing of the base layer. Thus, it is easy to conclude that network feedback must be inserted by the router into *all* passing packets (regardless of their color) for timely delivery to the end flows. Below, we discuss methods to discard out-of-sequence (i.e., outdated) feedback that may arrive in red/yellow packets.

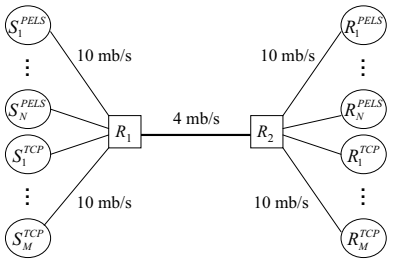

Figure 6. Simulation topology.

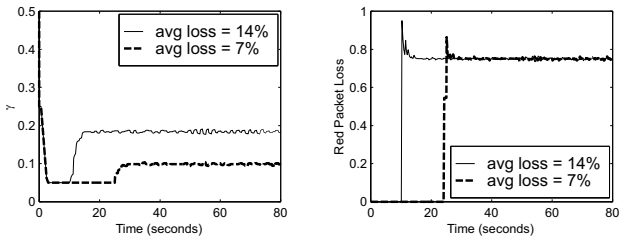

Figure 7. The evolution of γ (left). The corresponding red loss rates (right).

5.2. PELS Implementation

We implemented new agents and a priority-based AQM mechanism for PELS streaming in the ns2 network simulator [27]. PELS application sources mark their packets with three priority levels (i.e., green, yellow, and red) and employ MKC for rate control. Computation of packet loss $p(k)$ is performed by the router on a discrete time scale of T time units and then injected into the header of each packet passing through the router (note that feedback information is a queue-specific metric). Each new computation of $p(k)$ increases router's local epoch number z to prevent sources from reacting to the same feedback more than once as well as to suppress outdated values of $p(k)$ created by re-ordering inside PELS queues. Label (*router ID*, z, $p(k)$) is provided to end flows through the header of the packets queued at the bottleneck link.

Once received by the end-user, feedback $p(k)$ is sent in ACKs to the source, which applies rate adjustments according to (8) as long as it has not seen this feedback before. The use of epoch numbers allows the source to keep the frequency of its control loop in sync with that of the router and ensures stability of the resulting system.

We next describe the above two algorithms in more detail. Upon arrival and queuing of a packet i, the router increments its local counter S by the size of the packet s_i: $S = S + s_i$. Then once every T time units, the router computes new total rate R, new packet loss p, increments its epoch number z, and resets the byte counter:

$$R = \frac{S}{T}, p = \frac{R - C}{R}, z = z + 1, S = 0. \tag{11}$$

To verify the "freshness" of feedback, each PELS source i checks feedback sequence number z in the acknowledgment and ignores feedback with z less than or equal to its current epoch number z_i; otherwise, z_i is set to z and a new sending rate is computed using (8). When there are multiple routers along an end-to-end path, each router compares its p_l with that inside arriving packets and overrides the existing value only if its packet loss is larger than the current loss recorded in the header. End flows use the *router ID* field to keep track of feedback freshness and react to possible shifts of the bottlenecks.

Selection of interval T depends on the desired responsiveness of the PELS framework to network conditions, but does not affect stability of the system as a whole. To analytically reflect the implementation of the PELS framework where the router purposely delays its feedback by T units, we need to modify the model of packet loss in (9) to become:

$$p_l(k) = \frac{\sum_{j \in S_j} r_j(k - T - D_j^{\rightarrow}) - C_l}{\sum_{j \in S_j} r_j(k - T - D_j^{\rightarrow})}. \tag{12}$$

Stability of system (8)-(12) is proved using the same arguments as in Lemma 5 [34] and is omitted from this paper.

6. Simulation Results

In this section, we present simulation results of PELS including the properties of $\gamma(k)$, MKC congestion control, PELS queuing delay, and PELS video quality. We start by describing the simulation setup.

6.1. Simulation Setup

For ns2 simulations, we use a simple bar-bell topology with multiple PELS and TCP sources connecting to a single bottleneck link. As shown in Fig. 6, the capacity of the bottleneck is 4 mb/s, while the rest of the links are 10 mb/s. In all simulations, one video frame (63,000 bytes including the base layer) consists of 126 packets, 500 bytes each (these numbers are derived from MPEG-4 coded CIF Foreman). We mark 21 packets in each frame as green to protect the base layer of the sequence.

Recall that router queues in our framework completely separate the PELS flows from the traffic in the Internet queue. In our simulations, we allocate 50% of the bottleneck link to TCP cross-traffic; however, since the PELS and Internet queues do not affects each other in any way, we only focus on PELS flows and omit discussion of what happens to TCP traffic.

6.2. Stability Properties of γ

In this section, we show simulation results regarding stability of $\gamma(k)$ computed by end-flows using dynamically varying packet loss $p(k)$. Fig. 7 (left) shows the evolution of $\gamma(k)$ obtained by running PELS streaming simulations in ns2 with two different average packet losses and $\sigma = 0.5$. In the beginning, γ drops from the initial value of 0.5 to the lowest possible threshold $\gamma_{low} = 0.05$ since there is no packet loss (i.e., the

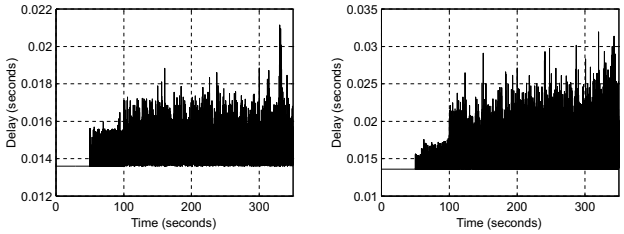

Figure 8. Green (left) and yellow (right) delays.

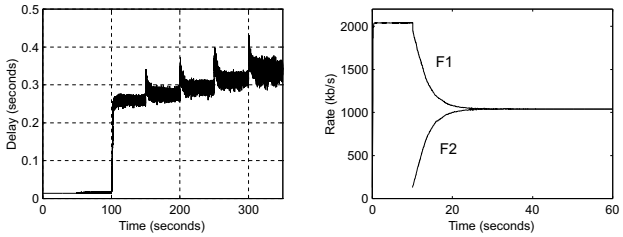

Figure 9. Red packet delays in PELS (left). Convergence and fairness of MKC congestion control (right).

flows slowly probe for new bandwidth). When packets start being dropped during congestion, γ increases until it is stabilized at $\gamma^* = p^*/p_{thr}$. Small oscillations of $\gamma(k)$ after it reaches the stationary point is caused by small variation in feedback $p(k)$.

Fig. 7 (right) illustrates red packet drop rates p_R corresponding to the values of γ on the left side the figure. As shown in the figure, red packet loss is stabilized at the target threshold rate $p_{thr} = 75\%$ regardless of the value of p (i.e., 7% or 14%). Since the red loss never reaches 100%, all of yellow packets are protected and experience (ideal) zero-loss conditions.

6.3. Delay Characteristics of PELS

Recall that AQM routers in the PELS framework employ three priority queues for preferential treatment of green, yellow, and red packets. Fig. 8 illustrates delays of green (left) and yellow (right) packets and Fig. 9 (left) depicts delays of red packets. Theses delays are obtained by running `ns2` simulations in which at every 50 seconds, two new flows entered the system with the initial rate of 128 kb/s (i.e., the rate of the base layer).

First notice that green and yellow packets have very small delays compared to those of red packets. The average delays of green and yellow packets are only 16 and 25 ms, respectively, while the average delays of red packets reach as high as 400 ms. Further notice that after 100 seconds, red packet delays increase every 50 seconds since each new flow further reduces the available bandwidth and increases congestion in the red queue. These results are expected from the use of priority queuing in the routers and have no harmful effect on PELS flows as loss or delays in the red queue have minimum impact on the video quality (in fact, the purpose of red packets *is* to be lost in the network and protect the yellow queue).

6.4. Properties of PELS Congestion Control

We next study characteristics of MKC congestion control coupled with the PELS queuing framework. Fig. 9 (right) illustrates convergence of two PELS flows to 50% of the available PELS capacity (i.e., 1 mb/s each) for $\alpha = 20$ kb/s and $\beta = 0.5$. In the figure, flow F1 starts at time zero with the initial rate r_0 = 128 kb/s and then converges to the full link capacity at around 0.1 seconds exponentially claiming the available bandwidth. It maintains the equilibrium rate until the second flow F2 starts at $t = 10$ seconds (r_0 = 128 kb/s). After another 13 seconds, both flows converge to a fair allocation of link's bandwidth. For additional simulations of MKC with non-equal feedback delays, see [5, 34].

6.5. PSNR Quality Evaluation

In this section, we compare the proposed preferential streaming scheme with the best-effort method using PSNR quality curves. Through simulation, we obtained packet loss statistics of each FGS frame and then applied them to the video sequence offline. We enhanced each base-layer frame using consecutively received FGS packets and plotted PSNR quality curves accordingly. Aggregate packet loss was calculated in the routers at $T = 30$ ms time intervals.

Our main puzzle in this section was to properly select a "generic" brand of best-effort streaming that adequately represents existing (non-QoS) approaches. Although there are numerous methods of streaming video over the Internet (including TCP, FEC-protected transmission, and various non-AIMD methods), we aim to compare PELS with an alternative framework that: 1) does not retransmit any lost packets; and 2) does not send any error-correcting codes. Since no such framework exists to our knowledge, we use AQM-enabled MKC under the assumption that the base layer is "magically" protected at all times. If packet loss is allowed in the base layer and retransmission is suppressed, best-effort streaming simply becomes impossible due to propagation of losses throughout each GOP (Group of Pictures). Thus, we protected the entire base layer in the best-effort case and allowed random loss only in the FGS layer to keep this approach even remotely competitive with PELS.

We first examine PSNR of the Foreman sequence reconstructed with 10% network packet loss (left of Fig. 10). As shown in the figure, best-effort streaming improves the base-layer PSNR by approximately 24% on average, while PELS enhances it by 60%.

Next, we examine the case with higher packet loss. Fig. 10 (right) illustrates the PSNR curve of the same Foreman sequence reconstructed with 19% packet loss. In this case, while the best-effort method improves the base-layer PSNR only by 16%, PELS improves it by 55%.

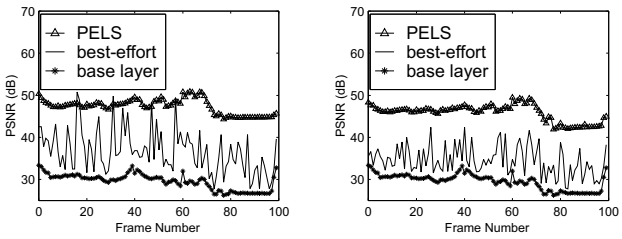

Figure 10. PSNR of CIF Foreman reconstructed with 10% (left) and 19% (right) packet loss.

From the curves in Fig. 10, we also observe that the PSNR of best-effort streaming varies by as much as 15 dB (even though the sending rates of MKC are perfectly smooth) and provides a highly-fluctuating quality that is similar to that achievable with AIMD [5]. On the contrary, PELS maintains a much higher PSNR throughout the entire sequence and keeps quality fluctuation to a minimum, which can be further reduced using sophisticated R-D scaling methods [5] (not used in this work). Thus, we can conclude that PELS streaming provides an effective and low-overhead QoS foundation for scalable multimedia streaming in the future Internet.

7. Conclusion

This paper studied characteristics of video streaming in best-effort networks and proposed a preferential streaming framework (PELS) that can provide a high level of end-user QoS. We further studied modified Kelly controls in conjunction with PELS and found that they presented a good foundation for future video streaming in AQM environments. Since the PELS framework is independent of congestion control methods employed, it can be further used with a variety of existing and future game-theoretic or optimization-based controllers.

Acknowledgment

The authors are sincerely grateful to Hayder Radha for bringing this problem to our attention.

References

[1] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms," *IEEE INFOCOM*, 2001.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *IETF RFC 2475*, 1998

[3] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," *IETF RFC 1633*, 1994.

[4] D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE/ACM Trans. on Networking*, vol. 6, no. 4, 1998.

[5] M. Dai and D. Loguinov, "Analysis of Rate-Distortion Functions and Congestion Control in Scalable Internet Video Streaming," *ACM NOSSDAV*, June 2003.

[6] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," *ACM SIGCOMM*, 1989.

[7] V. Firoiu, and M. Borden, "A Study of Active Queue Management for Congestion Control," *IEEE INFOCOM*, 2000.

[8] V. Firoiu, X. Zhang, and Y. Guo, "Best Effort Differentiated Services: Trade-off Service Differentiation for Elastic Applications, *IEEE International Conference on Telecommunications* (ICT), June 2001.

[9] S. Floyd and J. Padhye, "Equation-Based Congestion Control for Unicast Applications," *ACM SIGCOMM*, 2000.

[10] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, 1993.

[11] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, October 1994.

[12] E. Gurses, G.B. Akar, and N. Akar, "Impact of Scalability in Video Transmission in Promotion-Capable Differentiated Service Networks," *IEEE ICIP*, 2002.

[13] P. Hurley, M. Kara, J.Y. Le Boudec, and P. Thiran, "ABE: Providing a Low-Delay Service within Best Effort," *IEEE Network Magazine*, vol. 15, no. 3, May 2001.

[14] R. Johari and D.K.H. Tan, "End-to-End Congestion Control for the Internet: Delays and Stability," *IEEE/ACM Trans. on Networking*, vol. 9, no. 6, 2001.

[15] S. Kang, Y. Zhang, M. Dai, and D. Loguinov, "Multi-Layer Active Queue Management and Congestion Control for Scalable Video Streaming," *Texas A&M Technical Report*, January 2004.

[16] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," *ACM SIGCOMM*, 2002.

[17] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society*, 49, 1998.

[18] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," *ACM SIGCOMM*, 2001.

[19] A. Kuzmanovic and E. Knightly, "TCP-LP: A Distributed Algorithm for Low Priority Data Transfer," *IEEE INFOCOM*, 2003.

[20] D. Lapsley and S. Low, "Random Early Marking: an Optimization Approach to Internet Congestion Control," *IEEE ICON*, 1999.

[21] D. Loguinov and H. Radha, "End-to-End Internet Video Traffic Dynamics: Statistical Study and Analysis," *IEEE INFOCOM*, 2002.

[22] S. Low, "Equilibrium Bandwidth and Buffer Allocations for Elastic Traffics," *IEEE/ACM Trans. on Networking*, 2000.

[23] S. Low and D. Lapsley, "Optimization Flow Control–I: Basic Algorithm and Convergence," *IEEE/ACM Trans. on Networking*, 1999.

[24] S. Low, F. Paganini, J. Wang, S. Adlakha, and J. Doyle, "Dynamics of TCP/RED and a Scalable Control," *IEEE INFOCOM*, 2002.

[25] S. Low, L. Peterson, and L. Wang, "Understanding TCP Vegas: a Duality Model," *ACM SIGMETRICS*, 2001.

[26] L. Massoulie, "Stability of Distributed Congestion Control With Heterogeneous Feedback Delays," *IEEE INFOCOM*, 2002.

[27] Network Simulator (ns2). *http://www.isi.edu/nsnam/ns/*.

[28] QBone Scavenger Service. *http://qbone.internet2.edu/qbss*.

[29] H. Radha, M. Schaar, and Y. Chen, "The MPEG-4 Fine-Grained Scalable Video Coding Method for Multimedia Streaming Over IP," *IEEE Trans. on Multimedia*, vol. 3, no. 1, 2001.

[30] J. Shin, J.G. Kim, J.W. Kim, and C.C. Kuo, "Dynamic QoS Mapping Control for Streaming Video in Relative Service Differentiated Networks," *European Trans. on Telecommunications*, vol. 12, no. 3, 2001.

[31] J. Shin, J.W. Kim, and C.C. Kuo, "Quality-of-Service Mapping Mechanism for Packet Video in Differentiated Services Network," *IEEE Trans. on Multimedia*, vol. 3, no. 2, 2001.

[32] M. Shreedhar and G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin," *IEEE/ACM Trans. on Networking*, vol. 4, no. 3, 1996.

[33] J. Tang, G. Morabito, I. Akyidiz, and M. Johnson, "RCS: a Rate Control Scheme for Real-time Traffic in Networks with High Bandwidh Delay Products and High Bit Error Rates," *IEEE INFOCOM*, 2001.

[34] Y. Zhang, S. Kang, and D. Loguinov, "On Delayed Stability and Performance of Distributed Congestion Control," *Texas A&M Technical Report*, 2003.

[35] L. Zhao, J.W. Kim, and C.C. Kuo, "MPEG-4 FGS Video Streaming with Constant-Quality Rate Control and Differentiated Forwarding," *SPIE VCIP*, 2002.