On the Interactions Between Layered Quality Adaptation and Congestion Control for Streaming Video

Nick Feamster, Deepak Bansal, and Hari Balakrishnan
MIT Laboratory for Computer Science
{feamster, bansal, hari}@lcs.mit.edu
http://nms.lcs.mit.edu/projects/videocm/

Abstract

This paper uses analysis and experiments to study the impact of various congestion control algorithms and receiver buffering strategies on the performance of streaming media delivery. While traditional congestion avoidance schemes such as TCP's additive-increase/-multiplicativedecrease (AIMD) achieve high utilization, they also cause large oscillations in transmission rates that degrade the smoothness and perceptual quality of the video stream. We focus on understanding the interactions of a family of congestion control algorithms that generalize AIMD, with buffer-based quality adaptation algorithms for hierarchically-encoded and simulcast video. Our work builds on and extends the results of Rejaie et al. [19]; we find that the combination of a non-AIMD algorithm that has smaller oscillations than AIMD and a suitable receiver buffer allocation and management strategy provides a good combination of low playout delay and TCP-friendly congestion control. The paper describes these mechanisms and the results of experiments conducted using a prototype video server for MPEG-4 video, showing that our approach can improve the interactivity and adaptivity of Internet video.

1 Introduction

Streaming video and audio are an increasingly important component of the Internet. Unlike traditional broadcast media like television or cable, Internet conditions change with time, often rapidly. As many others do, we believe that a framework for streaming media over the Internet that is adaptive to varying network conditions is preferable to one that is not. This is important because it is widely believed that the stability of the modern Internet is in large part due to the cooperative behavior of the end hosts implementing the window increase/decrease algorithms described in [1, 11].

Unlike bulk file transfers, however, streaming video seeks to achieve smooth playback quality, rather than simply transmit at the highest attainable bandwidth. This therefore calls for suitable mechanisms to smooth the playback rate, which would otherwise end up oscillating when a stream is sent over a traditional additive-increase/multiplicative-decrease (AIMD) algorithm as in TCP [9, 18]. The process of probing for bandwidth and reacting to observed congestion induces oscillations in the achievable transmission rate, and is an integral part of the nature of many end-to-end congestion management algorithms.

In addition to performing congestion control that interacts well with other flows on the Internet, a streaming media server should adapt the quality of its transmission based on the available bandwidth. That is, a video server should scale the quality of transmission depending on prevailing network conditions. This mechanism is known as *quality adaptation* and can be performed in a number of ways. One option, called *simulcast*, encodes the bitstream at various target bitrates and switches between the previously encoded layers as the available bandwidth changes. An alternative quality adaptation scheme uses *hierarchical encoding* [12, 13, 25], where the video stream is encoded at a base layer and one or more enhancement layers, which can be combined to render the stream at higher quality. As the available bandwidth varies, the number of enhancement layers is adjusted by the server.

Rate oscillations degrade the quality of received video because they require more buffering to sustain a smooth playout and often result in variable quality, which is visually unappealing to the receiver. Overcoming the oscillatory nature of AIMD congestion control to smoothen bitstream can be done in two ways:

- Alternatives to AIMD congestion control [5, 9, 20].
- Use a combination of quality adaptation and receiver buffering [18].

The video server can transmit data according to a congestion control algorithm that results in oscillations of a smaller magnitude than AIMD, such as equation-based congestion control [9], TEAR (TCP Emulation at Receivers) [20], or binomial congestion control [5].

Alternatively, quality adaptation and buffering at the receiver can be used to smooth short-term oscillations caused by AIMD, as described by Rejaie et al [17]. Receiver buffering not only reduces jitter, but if enough data are buffered, also enables the receiver to sustain momentary drops in the sending rate by playing out of its buffer at a higher rate than the server is currently sending. This buffer accumulates when the sender is transmitting faster than the receiver is playing out, by not aggressively adding layers whenever any spare bandwidth becomes available. Rejaie et al. observe that receiver buffering in conjunction with quality adaptation can reduce the effect of oscillations that result from AIMD.

This paper builds on and extends the results of Rejaie et al. in three key directions:

1. We propose mechanisms for smoother playout of layers with a class of non-AIMD congestion control algorithms.
2. We present rules for quality adaptation for simulcast, hierarchical encoding, and an optimistic immediate adaptation algorithm.
3. We provide a mechanism for performing quality adaptation in combination with non-AIMD congestion control algorithms.

To our knowledge, this is the first exploration of receiver buffering for quality adaptation with non-AIMD congestion control and non-hierarchical layering using simulcast. In particular, alternate congestion control schemes such as binomial congestion control can reduce oscillations in the layers, and when used in conjunction with receiver buffering, yield further benefit in terms of interactivity.

Using binomial congestion control in conjunction with buffered quality adaptation for hierarchical encoding results in less required buffering to sustain a loss at any given layer than does AIMD. This smaller amount of required buffering results in a higher level of interactivity, due to the reduced delay before playout can happen. Furthermore, the use of binomial congestion control with quality adaptation results in a faster convergence to transmission of the correct number of layers, thus providing higher perceptual quality at the receiver.

Based on our new quality adaptation rules, we have conducted experiments over a network that emulates the conditions a streaming video server might face on the Internet. Our experiments are conducted using a prototype MPEG-4 server we have designed and implemented, using RTP for delivery, RTCP for feedback, and the Congestion Manager [3, 4] for congestion control. We investigate the effects of various quality adaptation mechanisms and congestion control algorithms. We find that binomial congestion control can provide benefits over AIMD, regardless of the quality adaptation mechanism used by reducing the amount of oscillation in the video server's sending rate. In addition, we find that binomial congestion control can provide significant benefits when used with buffered quality adaptation of hierarchically encoded video by reducing the amount of required buffering at the receiver in order to play out at any given layer.

The rest of the paper is organized as follows: in Section 2, we survey related work; in Section 3, we discuss the algorithms used and present the quality adaptation rules for binomial controls; in Section 4, we present our system architecture, implementation, and the results of our experiments.

2 Related Work

In recent years, much attention has focused on developing congestion control algorithms for streaming media applications. Early proposals for multimedia congestion control [10, 16, 18, 23, 23, 24] were essentially variants of TCP without the in-order, reliable delivery of data, semantics associated with TCP. More recently, proposals like TFRC [9], TEAR [20], and binomial controls [5] have focused on reducing large oscillations associated with TCP's congestion control. In TFRC, the sender explicitly adjusts its sending rate as a function of the measured rate of loss events based on the TCP-friendly equation developed in [15]. In the TEAR protocol, the receiver emulates the congestion window evolution of a TCP sender. The receiver maintains an exponentially weighted moving average of the congestion window, and divides this by the estimated round trip time to obtain a TCP-friendly sending rate. Binomial controls proposed in [5] generalize TCP's increase/decrease rules to derive a family of TCP-friendly congestion control algorithms with a varying degree of oscillation. We discuss binomial congestion controls in more detail in Section 3.

Rejaie et al. propose a quality adaptation scheme using receiver buffering for AIMD-controlled transmission and playback of hierarchically-encoded video [17, 19]. Long-term coarse-grained adaptation is performed by adding and dropping layers of the video stream, while using AIMD to react to congestion. Receiver buffering alleviates the short-term variations in the sending rate caused by the oscillatory nature of AIMD. A new layer will be added only if, at any point, the total amount of buffering at the receiver is sufficient to survive an immediate backoff and continue playing all of the existing layers plus the new layer, and the instantaneous available bandwidth is greater than the consumption rate of the existing layers, plus the new layer. When the total amount of buffering falls below the amount required for a drop from a particular rate, then the highest layer is dropped. Additionally, buffer space is allocated between layers so as to place a greater importance on lower layers, thereby protecting these layers upon a reduction in the transmission rate. The conditions for the addition and deletion of layers and inter-layer buffer allocation for AIMD and hierarchical encoding are described in [19].

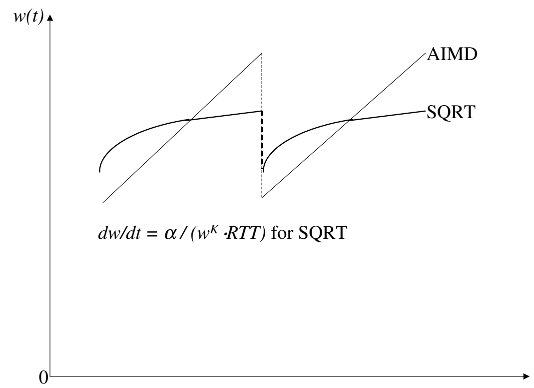

Figure 1. Window evolution vs time for SQRT and AIMD congestion controls. $w(t)$ is the window value at time t.

3 Algorithms

This section describes the algorithms for congestion control and quality adaptation used in our work.

3.1 Binomial congestion controls

Binomial congestion controls generalize TCP's increase/decrease rules using the following equations:

$$\begin{aligned} \text{I: } & w_{t+R} \leftarrow w_t + \alpha / w_t^k; \alpha > 0 \\ \text{D: } & w_{t+\delta t} \leftarrow w_t - \beta w_t^l; 0 < \beta < 1 \end{aligned} \quad (1)$$

k and l are the parameters of binomial controls and w_t is the instantaneous window value, which governs the transmission rate. For $k = 0,\ l = 1$, we get AIMD used by TCP; for $k = -1, l = 1$, we get MIMD (multiplicative increase/multiplicative decrease used by *slow start* in TCP [11]); for $k = -1, l = 0$, we get MIAD; and for $k = 0, l = 0$ we get AIAD, thereby covering the class of all linear algorithms.

In previous work [5], we showed that a binomial congestion control satisfying the (k, l) rule , i.e. $k+l = 1$, is TCP-friendly. Further, one member of this family, SQRT ($k = l = 0.5$) appears attractive for streaming media delivery due to its smaller magnitude of oscillations. In SQRT, the reduction in transmission rate is proportional to $\sqrt{w}$, whereas in AIMD the reduction is proportional to w. However, the potential benefits of such an algorithm and its impact on quality adaptation algorithms have not been studied, nor has its interaction with layered media delivery. Figure 1 shows the nonlinear evolution of the congestion window for the SQRT control algorithm.

3.2 Quality adaptation

To appropriately determine the quality of video that should be sent, rules must be defined in order to determine when a layer should be added or removed. These rules will vary depending on the methods of quality adaptation and congestion control that are employed.

In the case of instantaneous adaptation, the layer switching rules are simple: simply send video at the highest possible layer that can be sent at any given time. In this case, the magnitude and frequency of oscillations in the congestion control algorithm will govern the magnitude and frequency of layer switching and thus affect the perceived quality of video at the receiver.

To reduce layer switching, some hysteresis may be used, either by delaying switching to a higher quality video (until there is some confidence that the higher quality video can be sustained) or by buffering data at the receiver for future playout. These quality adaptation algorithm however, depends on whether the data is being simulcast or available as hierarchically encoded.

Rejaie et al. describe a quality adaptation scheme for hierarchically encoded data and AIMD congestion control, where a new layer is added only when the two conditions hold [19]:

$$\begin{aligned} R &> (n_a+1)C \\ \sum_{i=0}^{n_a-1} buf_i &\geq \frac{((n_a+1)C-\frac{R}{2})^2 T}{2\alpha} \end{aligned} \qquad (2)$$

Here, R is the current transmission rate, n_a the number of currently active layers, C the bandwidth/layer, buf_i the amount of data buffered for layer i, α the rate of linear increase in bandwidth, and T the round trip time. These rules ensure that the server adds a new layer only when:

1. The instantaneous available bandwidth is greater than the consumption rate of the existing layers plus the new layer, and,

2. There is sufficient total buffering at the receiver to survive an immediate backoff and continue playing all the existing layers plus the new layer.

The layers are dropped when any of the above rules is not satisfied. Additionally, [19] derives the optimal allocation of buffers at the receiver among various layer based on the observation that the distribution of buffering must provide maximal protection against dropping layers for any likely pattern of short-term reduction in available bandwidth. Since for hierarchically encoded data, the presence of lower layer data is essential for playing out data from higher layer, this implies that

1. Allocating more buffering for the lower layers not only improves their protection but also increases efficiency of buffering, and

2. Buffered data for each layer cannot provide more than its consumption rate(i.e. C). Thus, there is a minimum number of buffered layers needed to cope with short-term reductions in available bandwidth for successful recovery. This minimum is directly determined by the reduction in bandwidth (or the number of cutoffs, called *smoothing factor* in [19]) that we intend to absorb by buffering.

The above observations help them to derive conditions for optimal allocation of buffering for AIMD, as shown in Figure 2.

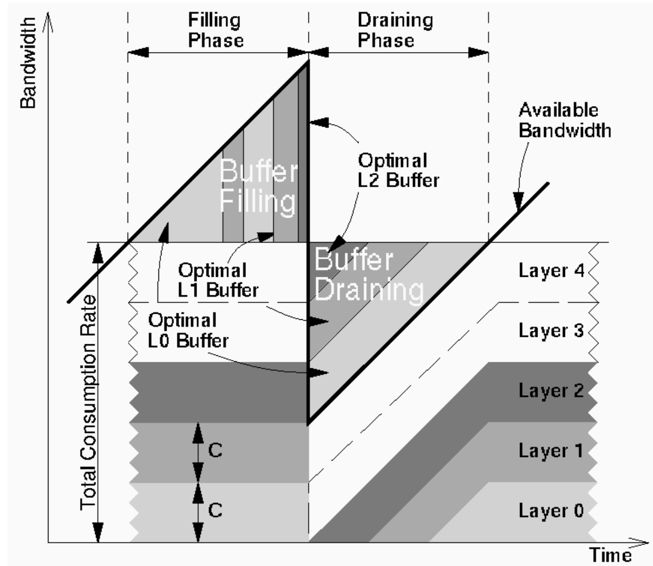

Figure 2. Optimal inter-layer buffer distribution as derived in [17, 19]. Figure from [17], reprinted with permission.

3.2.1 Simulcast

In simulcast, the sender encodes video offline at various transmission rates (each bitrate is an independent stream); in our analysis, we assume a constant rate spacing between successive layers. To reduce variation in the quality of video stream, some hysteresis is required so that transmission is not switched to a higher quality video unless there is some confidence that the higher quality video can be sustained. This implies that a video sender should not send highest quality video based on the instantaneous transmission rate.

We assume that a packet drop due to congestion may happen at any time. Under this assumption, the transmission should be switched to a higher layer only if the higher layer can be sustained after an immediate backoff. Note that unlike hierarchical encoding, buffered data for one layer becomes useless once the layer is switched because the layers are independent. Thus, if buffering is done to provide the hysteresis margin on backoffs, each time the layering is switched, the buffer must be built from the new layer. This results in choppiness in video playout. Thus, for an adaptive simulcast video, we assume that there are no buffers available to alleviate backoff and the resulting drop in the sending rate.

Under the above assumption, a simple method to determine the quality of video data to transmit is to use the instantaneous transmission rate and send the highest layer possible assuming that a backoff happens immediately. Then, for a TCP-friendly binomial control, the video quality that should be sent should be the highest encoding available such that

$$C < R - \beta R^l$$

Here, C is the bit rate at which the video stream is encoded, and R is the instantaneous transmission rate. For TCP-style AIMD, $\beta = 1/2$ and $l = 1$, so

$$C < R/2$$

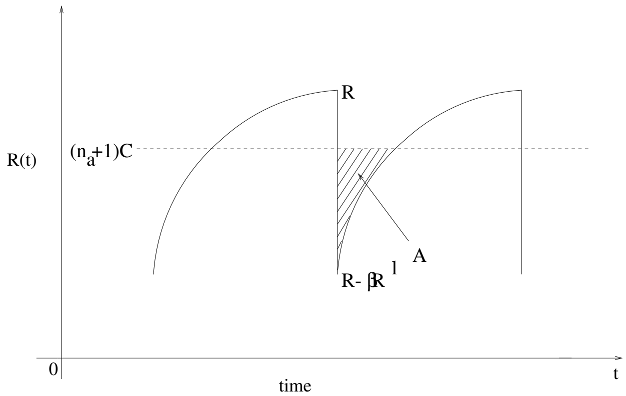

Figure 3. Figure showing the window evolution vs. time for binomial congestion control algorithm

and for SQRT:

$$C < R - \beta\sqrt{R}$$

Thus, using this simple algorithm, SQRT congestion control allows transmission of a higher layer compared to AIMD. Note that the above derivation assumes no buffering exists for a given quality of video at the receiver. If, on the other hand, one is willing to tolerate a certain amount of delay, one may buffer data at the receiver to sustain an immediate backoff.

3.2.2 Hierarchical encoding

In the case of hierarchical encoding, more complex rules govern the quality adaptation. We derive the buffering requirements and inter-layer buffer allocation strategy for binomial control algorithms.

Similar to [19], a layer should be added when the following conditions hold (assuming one backoff, i.e., smoothing factor = 1):

$$\begin{aligned} R &> (n_a+1)C \\ \sum_{i=0}^{n_a-1} buf_i &\geq A \end{aligned}$$

where A is the area of the shaded portion in Figure 3. This area, derived in the appendix, is given by:

$$A = \frac{(n_a+1)C[(n_a+1)^{k+1}C^{k+1} - (R-\beta R^l)^{k+1}]T}{(k+1)\alpha} - \frac{[(n_a+1)^{k+2}C^{k+2} - (R-\beta R^l)^{k+2}]T}{(k+2)\alpha} \quad (3)$$

Setting $k = 0$ for AIMD yields equation 2 as derived in [19]. Since $k = 1/2$ for SQRT algorithms, we get the following conditions for adding a layer:

$$\begin{aligned} R &> (n_a+1)C \\ \sum_{i=0}^{n_a-1} buf_i &\geq \frac{(n_a+1)C[(n_a+1)^{3/2}C^{3/2} - (R-\beta\sqrt{R})^{3/2}]T}{(3/2)\alpha} \\ &\quad - \frac{[(n_a+1)^{5/2}C^{5/2} - (R-\beta\sqrt{R})^{5/2}]T}{(5/2)\alpha} \end{aligned} \quad (4)$$

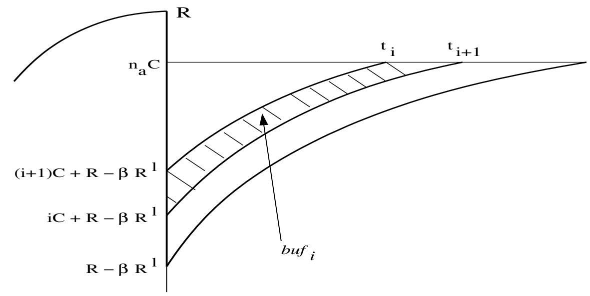

Figure 4. Optimal inter-layer buffer allocation for binomial congestion control.

Assuming that we have layers available at all encodings (i.e. continuity assumption), it follows that $n_a + 1$ corresponds to the encoding of the video data at the mean rate of transmission. Thus:

For AIMD:

$$(n_a+1)C = \frac{3R}{4}$$

and for SQRT:

$$(n_a+1)C < R - \frac{\beta}{2}\sqrt{R} \quad (5)$$

The inequality in 5 follows from the concavity of the window vs. time curve 3. Putting these values into equations 5 and 7, we get

For AIMD:

$$\sum_{i=0}^{n_a-1} buf_i \geq \frac{(3R/4 - R/2)^2 T}{2\alpha} = O(R^2) \quad (6)$$

while for SQRT:

$$\sum_{i=0}^{n_a-1} buf_i \geq \frac{(R^{3/2}\beta^2 + o(R))T}{8\alpha} = O(R^{3/2}) \quad (7)$$

Thus, the buffer required for adding a layer with SQRT is significantly lower than the buffer requirements when AIMD is used.

Figure 4 shows the optimal amount of buffering required for layer i with a binomial algorithm. We can express this buffering requirement buf_i, in terms of the layer i, the current rate R, the current layer's rate n_aC, and the parameters for binomial congestion control, α, β, k, and l. This buffer allocation for a scheme which uses binomial congestion control is similar to that of the AIMD scheme [19] and is derived in the appendix.

4 Implementation and experiments

We performed several experiments to assess the performance of various quality adaptation schemes with both AIMD and SQRT congestion control schemes under emulated network conditions. We ran these experiments on a prototype MPEG-4 streaming system that we are developing under Linux.

4.1 System description

Figure 5 shows the architecture of the system that we are developing for MPEG-4 delivery that serves as a testbed for our experiments.

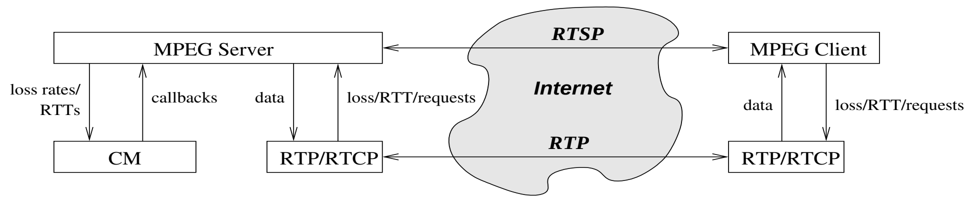

Figure 5. System architecture. Feedback is sent to the streaming application via RTCP, which is used to appropriately adjust the transmission rate.

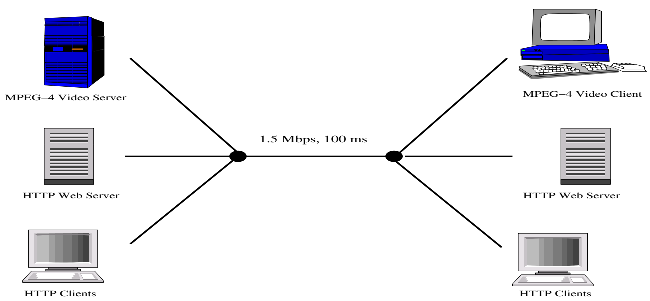

Figure 6. Experimental testbed.

It consists of a video server and a video client. The server listens for requests on an RTSP [22] port and streams requested data to the client via an implementation of RTP (over UDP) [21] that has been extended to support application-level framing and selective reliability. Feedback is provided to the server via RTCP receiver reports and is used to adjust the congestion window size at the server.

The RTSP request from the client for a particular video stream refers to a file containing an index for all the layers corresponding to that video, as well as their bit-rates. Based on its knowledge about the network capacity, the video server chooses the layer to send, and then, switches between different layers as the network conditions change.

Our system supports standards-compatible extensions to RTP/RTCP that allow for the application-level framing of the data with Application Data Units (ADUs) [7]. ADUs enable fragmentation and reassembly of independently processible units of data and also make selective recovery of application specific data units possible at the receiver. For the case of MPEG-4 video streaming, one frame of the compressed video bit-stream (separated by VOP start codes) corresponds to one ADU. These frames are packetized by the sender and then, when they are received by the receiver, are reassembled and passed to the application layer for decoding once the complete frame has been received.

The information provided by the receiver regarding packet loss rates, round trip times etc. is used to provide feedback to the underlying congestion control algorithm implemented using the Congestion Manager (CM) [2, 3, 4]. CM supports both AIMD and SQRT congestion control and an application can specify which congestion control algorithm should be used for its connections. CM also provides callback mechanisms to provide feedback to the application about the network conditions, which enables the application to adapt its content quality to the network conditions. This is done by having the video server make a request to the CM whenever it wants to send data; the CM issues a callback to the server when the connection is permitted to send a packet. The server also periodically queries the CM to determine the current bandwidth of the channel and then sends the appropriate quality of data.

We conducted experiments using the testbed shown in Figure 6. The video server (running on a Pentium 2 Linux 2.2.9 box) streamed data to the receiver (also a Pentium 2 Linux 2.2.9) across a 1.5 Mbps link with 100 ms latency, configured using Dummynet [8]. Background Web cross-traffic was introduced using the SURGE toolkit [6] to emulate the varying network conditions that an actual streaming server might see in the face of competing Web traffic on the Internet. Our test MPEG-4 video stream was encoded at rates corresponding to discrete rates between 100 Kbps and 700 Kbps, in constantly-spaced increments of 100 Kbps.

4.2 Instantaneous adaptation

Figures 7 and 8 show excerpts of traces of an MPEG-4 stream transfer from the video server to the client. In this case, the sender adapts the number of layers based on the *instantaneous* available bandwidth. While this approach has the advantage of adding layers aggressively and quickly taking advantage of bandwidth as it becomes available, it results in rapid oscillations as a result of response to the sawtooth-like behavior of the congestion control algorithms.

An important point to note in these graphs is that SQRT congestion control reduces the magnitude of layer switching in response to changes in available bandwidth. The multiplicative decrease of AIMD algorithms results in drastic reduction in the quality of video sent by the sender immediately following a packet drop. SQRT congestion control on the other hand, exhibits a significantly smaller reduction in the video quality as the sending rate is much smoother.

The average number of layers dropped when a backoff occurs is noticeably smaller with SQRT, because the magnitude of the rate change upon a drop is smaller than in AIMD. Furthermore, SQRT results in a more constant rate of transmission, which reduces the amount of buffering at the receiver to reduce jitter.

Figure 11 shows the frequency of various drop magnitudes for two different bottleneck bandwidths, 1.5 Mbps and 2.0 Mbps. In each case, SQRT did not cause any layer drops of more than one layer. However, AIMD often resulted in many layer drops per backoff. Because small oscillations in layer adaptation (i.e., drops of one layer upon backoff) are more tolerable to the user than large variations in quality, this suggests that SQRT provides much higher perceptual quality to the user by reducing the number of abrupt layer drops. Furthermore, as available bandwidth increases (compare the 1.5 Mbps and 2.0 Mbps histograms), the rate at which the server can transmit data increases, increasing the magnitude of backoff on packet loss. This is because the backoff is proportional to R for AIMD and $\sqrt{R}$ for SQRT, where R is the transmission rate before the drop. Thus, as the available bandwidth increases, the benefit of SQRT congestion control with respect to layer dropping becomes more pronounced.

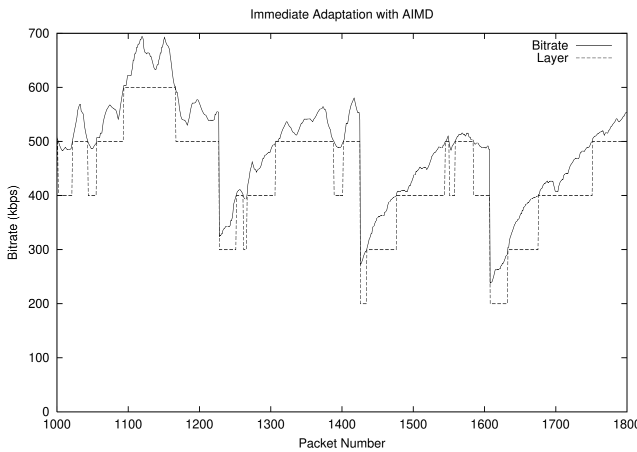

Figure 7. AIMD congestion control with immediate adaptation.

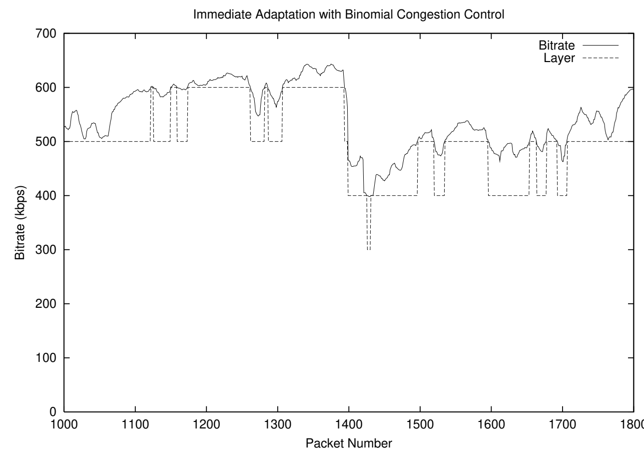

Figure 8. SQRT congestion control with immediate adaptation.

4.3 Simulcast

Figures 9 and 10 show excerpts from a similar MPEG-4 transfer, but with the layering decisions made by the simple simulcast quality adaptation rules. That is, the video server will not add an additional layer unless it can support that layer following one immediate backoff at any given time.

Here, the benefits of SQRT congestion control are significant. SQRT results in a higher average layer sent for a given average sending rate. This is because the sender can send a higher layer and still sustain backoffs, given that the amount by which the server backs off with SQRT is considerably smaller.

Our simulcast rules drastically reduce the frequency of layer changes using AIMD, because a layer is not added until the sender is certain that it can continue to play that layer should a backoff occur at any given time. However, because the amount by which the rate changes via one backoff in AIMD is large, the sender is not able to add layers as quickly. As a result, the target bitrate of the video transmitted by the sender is much smaller than the average available rate, as shown in Figure 9.

Since the amount by which the sender reduces its rate upon packet loss with SQRT is smaller than with AIMD, the sender can add layers more aggressively as the rate increases, since a backoff that may ensue at any given time will not result in as drastic of a rate reduction as with AIMD. As a result, the sender is capable at sending much higher quality video under simulcast using SQRT than with AIMD. This can be seen by comparing Figures 9 and 10.

Thus, for any given rate, simulcast quality adaptation rules in conjunction with SQRT congestion control result in a higher target bitrate video being transmitted for a given transmission rate. This has important applications for streaming video. For one, when starting from slow start, the video application can send the highest quality video more quickly as the rate increases (i.e., faster convergence). Second, given a rate which the sender is capable of sending at any given time, the server can send higher layers under SQRT congestion control, because the magnitude of backoffs is smaller, thus resulting in a higher quality of video received by the client.

Another interesting point to note by comparing Figures 8 and 10 is that the introduction of the more conservative simulcast rule did not have a considerably great effect on reducing the oscillations in layer switching. This appears to be because SQRT backoffs are often rather small, and the instantaneous channel bandwidth seen by the sender undergo transient oscillations due to RTT variations, which

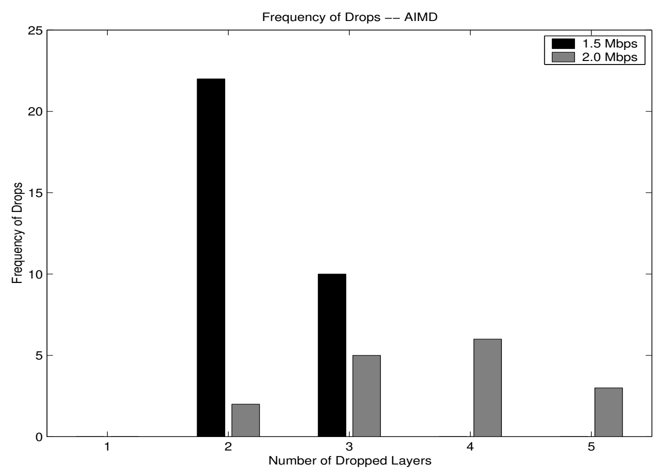

Figure 11. Frequency of layer drops of various magnitudes using AIMD congestion control and instantaneous rate adaptation. For SQRT congestion control, not shown in the figure, all backoffs resulted in dropping exactly one layer. The benefits of SQRT become more significant as bottleneck bandwidth increases and AIMD backoffs become larger.

can change the perceived sending rate temporarily by more than one SQRT backoff. This effect is not seen in AIMD, because these rate variations are smaller than one backoff. One area of future work is in handling these transient variations appropriately in the context of simulcast quality adaptation and binomial congestion control.

4.4 Hierarchical encoding

With hierarchical encoding and receiver buffered quality adaptation, the oscillations resulting from sending layers using instantaneous or simulcast rules can be further reduced, as buffering built up at the receiver can be used to play out video at higher layers, even if the rate momentarily drops below the total bit-rate being supported by the layers being sent.

In Section 3, we showed that, for a given rate, SQRT requires much less buffering at the receiver, thus resulting in a higher de-

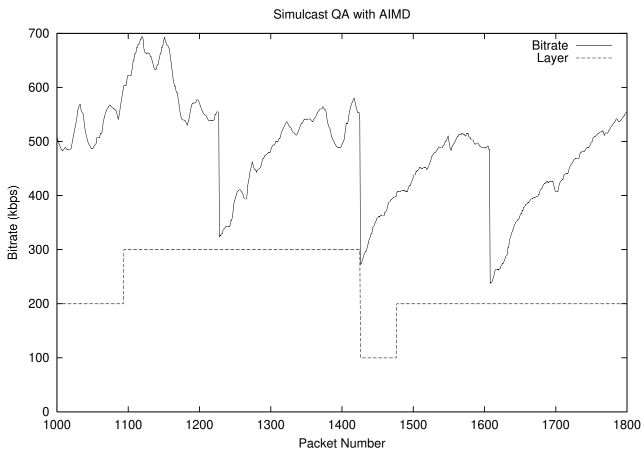

Figure 9. AIMD congestion control with simulcast quality adaptation.

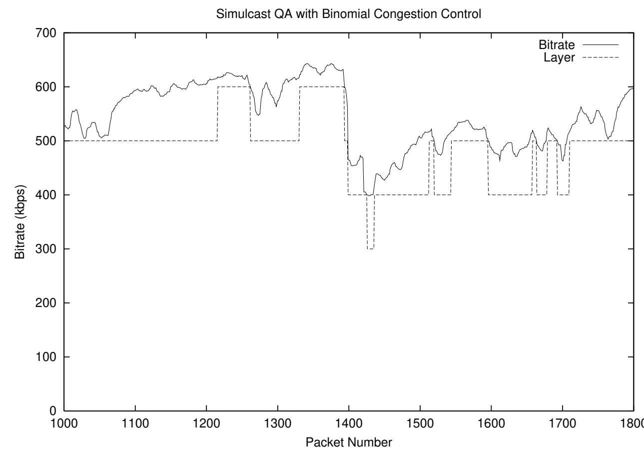

Figure 10. SQRT congestion control with simulcast quality adaptation.

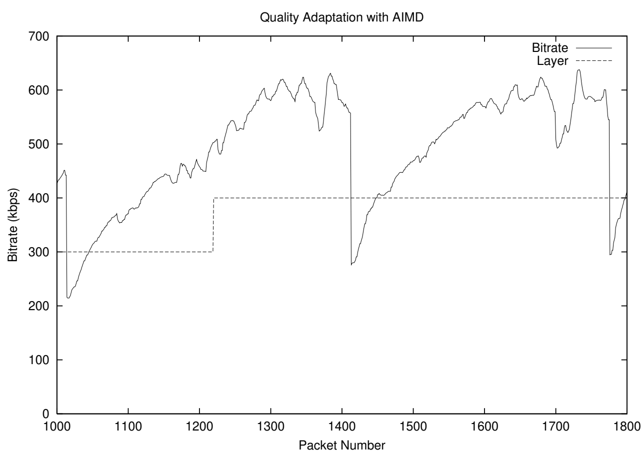

Figure 12. AIMD congestion control for hierarchical encoding with receiver-buffered quality adaptation.

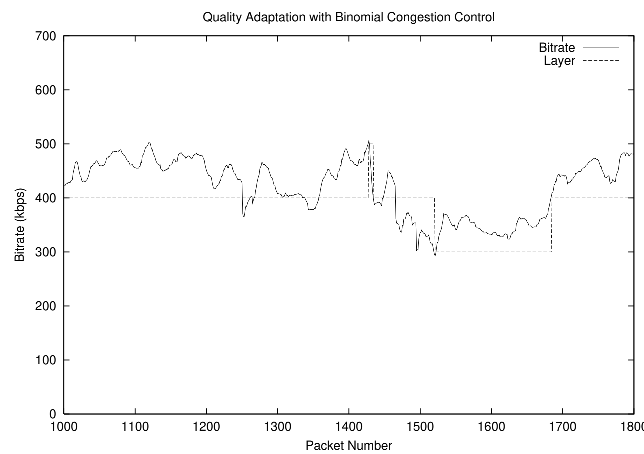

Figure 13. SQRT congestion control for hierarchical encoding with receiver-buffered quality adaptation.

Layer	**AIMD**	**SQRT**
2	76.41	44.97
3	196.25	63.13
4	233.12	99.94

Table 1. Buffering requirements in KBytes for adding layers for AIMD and SQRT algorithms.

gree of interactivity and faster convergence. Table 1 verifies our analytical findings. This trial shows that considerably less receiver buffering is required to add a layer, thus resulting in faster convergence to the appropriate rate and a higher degree of interactivity. As with simulcast quality adaptation, the number (but not magnitude) of oscillations seen by a system employing hierarchical encoding and SQRT is mildly higher compared to AIMD, but these oscillations are offset by higher overall quality of received video at the receiver.

Using hierarchical encoding with receiver buffering, both AIMD and SQRT congestion control algorithms result in convergence to the same transmitted layer, but AIMD is much slower in converging because a lot more buffering must be built up to sustain a backoff at any particular time. Figures 12 and 13 show layered quality adaptation for AIMD and SQRT, configured to sustain up to two immediate backoffs in transmission rate due to (unforeseen) congestion.

Thus, not only does AIMD require more buffering at the receiver than SQRT, but it also takes a longer time to play out the layers associated with the average rate. This result indicates that significantly higher interactivity is possible using SQRT congestion control. If a user wants to perform random access on a video stream (forward, rewind, etc.) to a point where no data is buffered for the video stream at any layer, using SQRT congestion control allows the video server to converge and start playing a higher quality of video much more quickly than AIMD does. Using SQRT congestion control also reduces the initial perceived latency at the beginning of a stream before a video clip can start being rendered.

Smooth quality of a received video signal depends on appropriate buffering. In particular, receiver buffering must be large enough to (1) account for network jitter (delay variation), (2) allow time for retransmission of lost packets, and (3) enable quality adaptation.

The buffering required to counteract network jitter is a function of the variance in network delay, where the instantaneous jitter j_i can be expressed as $|(A_i - A_{i-1}) - (S_i - S_{i-1})|$ [14, 21]. Using this, the required buffering to counteract network jitter is $\beta\delta_i$, where δ_i is smoothed jitter; smaller values of β reduce overall delay, and larger values decrease the likelihood of late (hence, effectively lost) packets. Buffering for retransmission of lost packets also depends on the absolute network round-trip time. Buffering for quality adaptation depends on the absolute transmission rate. We have shown that required QA buffering is $O(R^{3/2})$ for SQRT congestion control and $O(R^2)$ for AIMD. The dominant factor for the required buffering thus depends on the relation of the absolute round-trip time to the RTT variance and the absolute transmission rate. As the absolute RTT becomes large with respect to RTT variance, buffering due to retransmission requests will dominate buffering required to counteract jitter, and vice versa.

5 Conclusion

In this paper, we have addressed the issue of how the binomial congestion control algorithms, which reduce oscillations in the transmission rate, can be used by layered streaming applications to improve video quality and interactivity. We studied the interaction between the quality adaptation mechanisms for variable rate and hierarchically encoded data and the underlying congestion control algorithm, building on the work by Rejaie et al [19].

Besides binomial controls, several other algorithms like TFRC and TEAR have been proposed to reduce the oscillations in the sending rate. Evaluating the benefits of these other schemes and comparing them with the binomial algorithms for streaming media delivery is a topic for future work. While congestion control for multimedia is an active topic of research, more research is needed on the streaming application algorithms to adapt to the vagaries of the network and the underlying layers to ensure better user experience for real applications. We believe that our MPEG-4 server, integrated with the congestion manager, provides an attractive platform for such work.

References

[1] M. Allman and V. Paxson. *TCP Congestion Control*. Internet Engineering Task Force, April 1999. RFC 2581.

[2] D. Andersen, D. Bansal, D. Curtis, S. Seshan, and H. Balakrishnan. System support for bandwidth management and content adaptation in Internet applications. In *Proc. Symposium on Operating Systems Design and Implementation*, October 2000.

[3] H. Balakrishnan, H. S. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *Proc. ACM SIGCOMM*, pages 175–187, September 1999.

[4] H. Balakrishnan and S. Seshan. *The Congestion Manager*. Internet Engineering Task Force, Nov 2000. Internet Draft draft-balakrishnan-cm-03.txt (`http://www.ietf.org/internet-drafts/draft-balakrishnan-cm-03.txt`). Work in progress, expires May 2001.

[5] D. Bansal and H. Balakrishnan. Binomial Congestion Control Algorithms. In *Proceedings INFOCOM 2001*, April 2001. Also available as MIT-LCS-TR-806 from `http://nms.lcs.mit.edu/papers/cm-binomial.html`.

[6] P. Barford. SURGE – Scalable URL Reference Generator. `http://www.cs.bu.edu/students/grads/barford/Home.html`, 1998.

[7] D. Clark and D. Tennenhouse. Architectural Consideration for a New Generation of Protocols. In *Proc. ACM SIGCOMM*, pages 200–208, September 1990.

[8] Dummynet. `http://www.iet.unipi.it/~luigi/ip_dummynet`, September 1998.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. *Proc. ACM SIGCOMM*, pages 43–54, September 2000.

[10] S. Jacobs and A. Eleftheriadis. Providing Video Services over Networks without Quality of Service Guarantees. In *WWW Cons. Workshop on Real-Time Multimedia and the Web*, 1996.

[11] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM*, pages 314–329, August 1988.

[12] J.-Y. Lee, T.-H. Kim, and S.-J. Ko. Motion prediction based on temporal layering for layered video coding. In *Proc. ITC-CSCC*, volume 1, pages 245–248, July 1998.

[13] S. McCanne. *Scalable Compression and Transmission of Internet Multicast Video*. PhD thesis, Univ. of California at Berkeley, 1996.

[14] S. McCanne. Scalable multimedia communication with internet multicast, light-weight sessions, and the MBone. Technical Report CSD-98-1002, August 1998.

[15] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A Simple Model and its Empirical Validation. In *Proc. ACM SIGCOMM*, September 1998.

[16] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A Model Based TCP-friendly Rate Control Protocol. In *Proc. NOSSDAV*, July 1999.

[17] R. Rejaie, M. Handley, and D. Estrin. Quality Adaptation for Unicast Audio and Video. In *Proc. ACM SIGCOMM*, September 1999.

[18] R. Rejaie, M. Handley, and D. Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proc. IEEE INFOCOM*, volume 3, pages 1337–1345, March 1999.

[19] R. Rejaie, M. Handley, and D. Estrin. Layered Quality Adaptation for Internet Video Streaming. *IEEE Journal on Selected Areas of Communications (JSAC)*, Winter 2000. Special issue on Internet QOS. Available from `http://www.research.att.com/~reza/Papers/jsac00.ps`.

[20] Rhee, I., Ozdemir, V. and Yi, Y. TEAR: TCP Emulation at Receivers - Flow Control for Multimedia Streaming. NCSU Technical Report, April 2000.

[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. IETF, January 1996. RFC 1889.

[22] H. Schulzrinne, A. Rao, and R. Lanphier. *Real Time Streaming Protocol (RTSP)*. IETF, April 1998. RFC 2326.

[23] D. Sisalem and H. Schulzrinne. The Loss-Delay Adjustment Algorithm: A TCP-friendly Adaptation Scheme. In *Proc. NOSSDAV*, July 1998.

[24] W. Tan and A. Zakhor. Real-time Internet Video Using Error Resilient Scalable Compression and TCP-friendly Transport Protocol. *IEEE Trans. on Multimedia*, 1(2):172–186, May 1999.

[25] M. Vishwanath and P. Chou. An efficient algorithm for hierarchical compression of video. In *Proc. IEEE International Conference on Image Processing*, November 1994.

A Quality adaptation

A.1 Buffering requirements

For binomial controls, the increase in sending rate (R) per round trip (T) is governed by the following equation ([5]):

$$\begin{aligned}
dR/dt &= \frac{\alpha}{R^k T} \\
A_1 &= \int_{t_1}^{t_2} R dt \\
&= \int_{R-\beta R^l}^{(n_a+1)C} \frac{R^{k+2}T}{(k+1)\alpha} dt \\
&= \frac{[(n_a+1)^{k+2}C^{k+2}]T}{\alpha(k+2)} \\
A &= (n_a+1)C(t_2-t_1) - A_1 \qquad (8) \\
t_2 - t_1 &= \frac{[(n_a+1)^{k+1}C^{k+1} - R^{k+1}]}{(k+1)\alpha} \\
\Rightarrow A &= \frac{(n_a+1)C[(n_a+1)^{k+1}C^{k+1} - R^{k+1}]}{(k+1)\alpha} - \frac{(n_a+1)^{k+2}C^{k+2} - R^{k+2}]}{(k+2)\alpha} \qquad (9)
\end{aligned}$$

A.2 Inter-layer buffer allocation

For binomial controls, the optimal inter-layer buffer allocation is determined by the following equation:

$$buf_i = [(n_a C t_i - N_i) - (n_a C t_{i+1} - N_{i+1})] \qquad (10)$$

where t_i is the amount of time taken for the rate to increase from $(iC + (R - \beta R^l))$ to $n_a C$ along the curve $R_i(t)$. N_i is the area under the curve $R_i(t)$ from the time period 0 to t_i. The buffer allocation for layer i, buf_i is shown by the shaded area in Figure 4. Thus from the equation relating the evolution of rate over time,

$$\begin{aligned}
\frac{dR}{dt} &= \frac{\alpha}{R^k} \\
R(t) &= \left[(k+1)\alpha t + R_0^{k+1}\right]^{\frac{1}{k+1}}
\end{aligned}$$

it is possible to determine an expression for t_i, the time taken for the instantaneous rate R to increase to $n_a C$ for a generic binomial increase by α / R^k.

$$t_i = \frac{1}{\alpha(k+1)} \left[(n_a C)^{k+1} - (iC + (R - \beta R^l))^{k+1}\right] \qquad (11)$$

The curve $R_i(t)$ is defined by the curve which originates at the rate corresponding to i layers above the backoff, i.e., $(iC + (R - \beta R^l))$ and increases in a binomial fashion. The area under the rate curve $R_i(t)$, N_i, can be expressed as:

$$\begin{aligned}
N_i &= \int_0^{t_i} R_i(t) dt \\
&= \int_0^{t_i} \left[(k+1)\alpha t + (iC + (R - \beta R^l))^{k+1}\right]^{\frac{1}{k+1}} \\
&= \frac{1}{\alpha(k+2)} \left[(n_a C)^{k+2} - (iC + (R - \beta R^l))^{k+2}\right] \qquad (12)
\end{aligned}$$

There are two scenarios considered in [18] corresponding to γ immediate backoffs (smoothing factor) and the backoffs uniformly separated. These are the worst cases possible and thus, give an upper bound on the buffer requirements for each layer.

A.2.1 Scenario 1

We can now substitute these results into equation 10 and generalize for γ successive backoffs for Scenario 1 to obtain

$$\begin{aligned} buf_i &= \frac{n_a C}{\alpha(k+1)}[((i+1)C + (R - \gamma\beta R^l))^{k+1} - (iC + (R - \gamma\beta R^l))^{k+1}] \\ &\quad + \frac{1}{\alpha(k+2)}[(iC + (R - \gamma\beta R^l))^{k+2} - ((i+1)C + (R - \gamma\beta R^l))^{k+2}] \end{aligned} \tag{13}$$

There are two important notes with respect to this derivation. The first is that analytically we have shown an upper bound for the buffering required for γ immediate successive backoffs, in reality, the quantity of each successive backoff would be smaller because the backoff is reducing from the already smaller rate. In our implementation, we have calculated exact buffering requirements, but this result cannot be shown in a clean analytically closed form.

Second, it should be noted that the implementation calculates the buffer requirements for transmitting N layers by simply summing the optimal inter-layer buffer allocation requirements for each layer. This strategy was adopted because this was the approach taken in the *ns* simulation code for receiver buffered quality adaptation in Rejaie et al.'s work [18].

A.2.2 Scenario 2

In Scenario 2, the γ backoffs are divided into γ_i initial immediate backoffs (as in Scenario 1), followed by $\gamma - \gamma_i$ successive backoffs to $n_a C - \beta(n_a C)^l$.

The total amount of buffering required for Scenario 2 is the amount of buffering required in a Scenario 1 situation with γ_i backoffs (Equation 12) (denoted by buf_{i1}), plus the amount of buffering required to sustain $(\gamma - \gamma_i)$ successive backoffs from $n_a C$ to $n_a C - \beta(n_a C)^l$ (denoted by buf_{i2}). Thus,

$$\begin{aligned} buf_i &= buf_{i1} + (\gamma - \gamma_i) buf_{i2} \\ buf_{i1} &= \frac{n_a C}{\alpha(k+1)}[(((i+1)C + (R - \gamma_i\beta R^l))^{k+1} - (iC + (R - \gamma_i\beta R^l))^{k+1}] \\ &\quad + \frac{1}{\alpha(k+2)}[(iC + (R - \gamma_i\beta R^l))^{k+1} - ((i+1)C + (R - \gamma_i\beta R^l))^{k+1}] \end{aligned} \tag{14}$$

$$\begin{aligned} buf_{i2} &= \frac{n_a C}{\alpha(k+1)}[(n_a C)^{k+1} - (n_a C - \beta(n_a C)^l))^{k+1}] \\ &\quad + \frac{1}{\alpha(k+2)}[(n_a C)^{k+2} - (R - \beta(n_a C)^l))^{k+2}] \end{aligned} \tag{15}$$