The Minimal Buffering Requirements of Congestion Controlled Interactive Multimedia Applications

Kang Li[1], Charles Krasic[1], Jonathan Walpole[1], Molly H.Shor[2], and Calton Pu[3]

[1]Oregon Graduate Institute, Department of Computer Science and Engineering
{kangli, krasic, walpole}@cse.ogi.edu
[2]Oregon State University, Electrical and Computer Engineering Department
shor@ece.orst.edu
[3]Georgia Institute of Technology, College of Computing
calton@cc.gatech.edu

Abstract. This paper uses analysis and experiments to study the minimal buffering requirements of congestion controlled multimedia applications. Applications in the Internet must use congestion control protocols, which vary transmission rates according to network conditions. To produce a smooth perceptual quality, multimedia applications use buffering and rate adaptations to compensate these rate oscillations. While several adaptation policies are available, they require different amounts of buffering at end-hosts. We study the relationship between buffering requirements and adaptation policies. In particular, we focus on a widely pursued policy that adapts an application's sending rate exactly to the average available bandwidth to maximize throughput. Under this adaptation policy, at least a minimal amount of buffering is required to smooth the rate oscillation inherent in congestion control, and we view this minimal buffering requirement as a cost of maximizing throughput. We derive the minimal buffering requirement for this policy assuming that applications use an additive-increase-and-multiplicative-decrease (AIMD) algorithm for congestion control. The result shows the relationship between parameters of AIMD algorithms and the delay cost. We show that the buffering requirement is proportional to the parameters of the AIMD algorithm and quadratic to the application's sending rate and round-trip-time. We verify this relationship through experiments. Our results indicate that adaptation policies that maximize throughput are not suitable for interactive applications with high bit rates or long round-trip-times.

1. Introduction

Interactive multimedia applications, such as videoconferencing and IP telephony, are becoming important components of the Internet. Unlike traditional broadcast networks, the modern Internet is highly dynamic and is characterized by rapidly changing conditions. Applications must use congestion control protocols to react to the dynamics of the Internet in order to maintain its stability [1].

TCP is the de-facto standard transport protocol for bulk data transfer in the Internet. However, it does not work well for interactive multimedia applications. Its retransmissions and drastic rate adjustments can cause significant delays for applications. In recent years, researchers have proposed various TCP-friendly congestion control protocols, such as equation-based congestion control [2] and general *additive-increase-and-multiplicative-decrease (AIMD)* based congestion control [3]. These have significantly improved the performance of multimedia applications over the Internet [4], and flows of these protocols interact well with other TCP traffic. However, using TCP-friendly congestion control reduces but does not remove the oscillations in the transmission rate.

The rate oscillations of congestion control protocols are unavoidable because of the Internet dynamics and the nature of congestion control algorithms. The Internet dynamic is a result of the huge variation in applications, users, and usage patterns [5,6]. As a result, the Internet bandwidth share of an application varies with time. In addition, congestion control protocols must probe the network for available bandwidth. 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 all end-to-end congestion management algorithms.

Multimedia applications often use buffering at the receiver side to smooth these rate oscillations because users prefer smooth playback rates to the variable rate of the network transmission. In addition to buffering, multimedia applications adjust their playback quality based on the available transmission rate. This mechanism is known as quality-of-service (QoS) adaptation, which can be performed to adjust an application’s sending rate (as well as playback rate) in a number of ways [7, 8,9].

In this paper, we study the buffering requirements of different *adaptation policies*. An adaptation policy is an application’s way of estimating the network transmission rate and adjusting its transmission rate to match. Adaptation policies have significant impacts on buffering requirements. A sluggish adaptation policy that loosely tracks the network transmission rate requires a large amount of buffering to sustain the application’s playback rate when the network transmission rate drops. On the other hand, an aggressive adaptation that tracks the network transmission rate closely requires less buffering.

We have noticed a trend of research toward adaptation policies that try to fully utilize the available bandwidth while preserves a smooth playback quality. Several existing papers [7, 10, 11] have described mechanisms, such as smart buffer management and fine-grained adaptation, to push the adaptation toward the direction of maximizing throughput. These works are mainly in the context of streaming media over the Internet, and aim to optimize bandwidth efficiency. Without inspecting the detailed effects of this adaptation policy, one might consider using it for interactive multimedia applications. However, we believe there is a cost associated with fully utilizing the achievable transmission rate. This cost is the buffering delay required to smooth the inherent rate oscillations of congestion control protocols. For interactive multimedia applications this cost may not be affordable.

In this paper, we derive the minimal buffering required to smooth the inherent rate oscillations of a congestion control protocol. We assume applications use general AIMD (GAIMD) based congestion control protocols. GAIMD congestion control protocols use TCP's AIMD algorithm but with an arbitrary pair of increase/decrease parameters (α,β). Throughout this paper, we use *AIMD(α,β)* to indicate a GAIMD-based congestion-controlled flow with (α,β) as parameters. For example, TCP's congestion control uses AIMD(1,1/2).

Our result shows that the minimal buffering requirement is proportional to the increment parameter α when the AIMD-based congestion control is TCP-friendly[1]. And more importantly, the buffering requirement increases quadratically with increases in rate and round-trip-time (RTT). This result indicates that using a small increment parameter α can reduce the buffering requirement, but the effect is limited as rate or RTT increases.

The rest of the paper is organized as follows. In Section 2, we describe the architecture of our target application, and explain how it adapts. In Section 3, we describe the general AIMD algorithm, and present an analytical derivation of its buffering requirement. In Section 4, we present our experimental architecture and results. Finally, Section 5 concludes the paper and outlines some future work.

2. Buffering and Adaptations

This section presents the structure of our target application, outlines how adaptation and buffering are used, and then describes the relationships between various adaptation policies and their minimal buffering requirements.

2.1 Application Structure

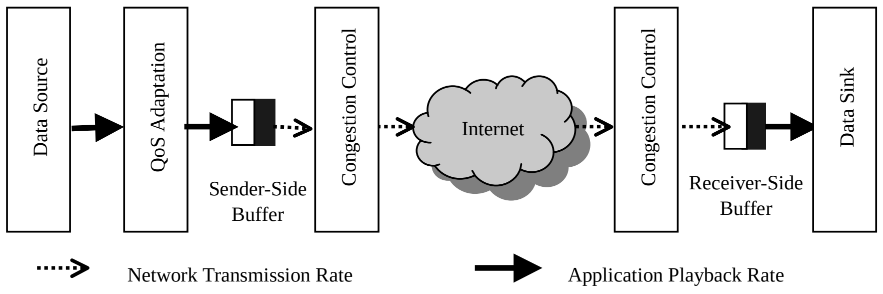

Fig.1. A QoS-Adaptive Application over the Internet

[1] The rule of choosing AIMD parameters for TCP-friendliness is presented in [12].

Figure 1 describes our target application's structure[2]. It includes a data source (e.g., a video camera) and a data sink (e.g., a display) connected via the Internet. The sender side generates data on the fly and sends data to the congestion control protocol through a buffer. Data is transmitted over the Internet, with transmission rate limited by the congestion control protocol, and is put into a receiver side buffer. The data sink fetches data from the buffer and presents it to users.

The transmission rate over the Internet oscillates over time. To achieve a stable playback quality at the data sink, receiver-side buffering and a sender-side adaptation mechanism are used. For simplicity, we assume that a constant playback quality (in application terms) maps to a constant bit rate (CBR). Thus, the users' preference of constant playback quality maps to the preference of a constant draining rate from the receiver-side buffer.

The receiver delays the start of playback at the data sink side until enough data has been accumulated in the receiver-side buffer, to allow the sink to keep playing for a while even when the network transmission rate drops below the playback rate. As long as the network transmission rate can catch up before the receiver-side buffer reaches empty, the user would not perceive any network rate oscillation. Once the transmission rate is higher than the playback rate, the buffer will start to fill again.

Determining what data to send and how to fill the buffer is complex. Applications require smart buffer filling strategies so that all buffered data are useful to compensate for network rate reductions in the future. Since the buffer management is not the focus of our work, we simply assume that the application can fully utilize all the buffered data. Studies of smart buffer management strategies can be found in recent research work [10, 11].

To reduce the buffering requirements, the target application makes QoS adaptation to adjust its sending rate according to the network transmission rate. We assume that the adaptation is fine-grain layer-based, and the application can adapt its rate closely to the network transmission rate. Several research works have shown ways of matching application rates to network rates using fine-grained adaptations. For example, Jacobs et al. [7] adapt encoding parameters according to the available bandwidth, Krasic et al. [8] propose a priority-based encoding mechanism and make a scalable rate adjustment for video streams, and more recently, Byers et al. apply a fine-grained rate adaptation [9] to multicast environments.

2.2 Adaptation Policies

For layer-based adaptations, adaptation policies are rules determining when a layer should be added or removed. Buffering requirements are closely related to how the

[2] We assume the application has only one-way traffic. A typical interactive application usually involves two-way traffic, which can be divided to two applications with one-way traffic but with tight dependency on each other.

application adapts its rate. In this section, we use examples to show this relationship. Figure 2 shows four adaptation policies with the same saw-tooth shape transmission rate, which is typical of AIMD-based congestion control protocols.

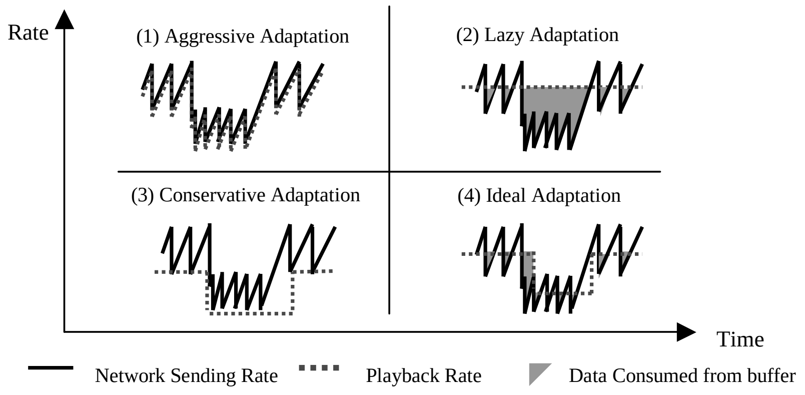

Fig. 2. Buffering Requirements of Different Backing-off Scenarios

Scenario (1) shows an aggressive adaptation policy that closely tracks the network transmission rate: whenever the instant transmission rate is one layer higher than the current application sending rate, a layer is added; whenever the instant transmission rate is lower than the current application sending rate, a layer is dropped until the sending rate is equal to or lower than the network transmission rate. This adaptation policy does not require any receiver side buffering but results in frequent quality variations.

Scenario (2) illustrates an unresponsive (lazy) adaptation policy that is opposite to the aggressive one illustrated in scenario (1), and produces a very stable playback rate. The policy does not adjust the application's sending rate according to the available network bandwidth. However, it requires a large amount of buffered data to compensate for the network rate variations, even when it chooses a playback rate that is close to the average network transmission rate.

Scenario (3) shows a conservative adaptation policy that always sends data at a rate lower than or equal to the lowest transmission rate in the recent history. This policy makes a layer adjustment decision at every time the congestion control backs off its rate, and maintains a sending rate that equals the lowest rate of the recent saw-tooth shape. With this policy, applications require no receiver-side buffering, and give users a relatively stable playback rate. However, this policy doesn't let the application use all the achievable transmission capacity detected by the congestion control protocol.

Scenario (4) presents an ideal adaptation policy. It is called ideal because it assumes advance knowledge of the network behavior, one saw-tooth ahead of time. Since it has future knowledge, it can choose the average of the next saw-tooth as its sending rate. Therefore it achieves a stable quality (in the next saw-tooth period) and maximizes throughput. The buffering requirement for this ideal adaptation policy is the amount of data needed to smooth one saw-tooth of the network transmission rate.

2.3 Cost for the Ideal Adaptation

This ideal adaptation is not a realistic adaptation policy for applications. However, it presents an interesting case for studying buffering since it exposes the minimal buffering requirement for maximizing throughput. We give a derivation for this buffering requirement in Section 3.

3. Buffering Requirement for General AIMD Congestion Control

An AIMD-based congestion control protocol uses a General AIMD algorithm to limit its sending rate in order to avoid congesting the network. It is a window-based congestion control protocol. That is, it uses a congestion window to limit the maximum amount of data sent out by the application within one round-trip-time.

3.1 GAIMD Algorithm

GAIMD generalizes TCP's AIMD algorithm in the following way:

$$\begin{cases} \text{Additive Increase:} \quad W_{t+RTT} \leftarrow W_t + \alpha \times MSS; \quad \alpha > 0 & (1) \\ \text{Multiplicative Decrease:} \quad W_{t+\delta} \leftarrow \beta \times W_t; \quad 0 < \beta < 1 & (2) \end{cases}$$

in which W_t is the congestion control's window size (in bytes) at time t, RTT is the round-trip-time, and MSS is the packet size[3]. α and β are parameters of the AIMD algorithm which control the paces of the additive increase and multiplicative back off respectively. The rate behavior of the GAIMD algorithm is similar to the saw-tooth shape of TCP congestion control, which uses an AIMD(1, ½).

3.2 Minimal Buffering Requirement

To determine the buffering requirement for smoothing the rate oscillations, we need to describe how the rate of an AIMD-based protocol evolves over time. Figure 3 shows an AIMD flow with a playback rate R. For an AIMD flow, the achievable rate in a single RTT is its window size divided by the RTT. The window size is controlled by the GAIMD algorithm as follows. If the window size before a back off is W, the achievable network transmission rate for this flow periodically varies from β*W / RTT to W/RTT.

[3] We assume the congestion control protocol uses a constant packet size and a constant RTT.

▬ Network Transmission Rate - - - Playback Rate ◸ Data Buffering

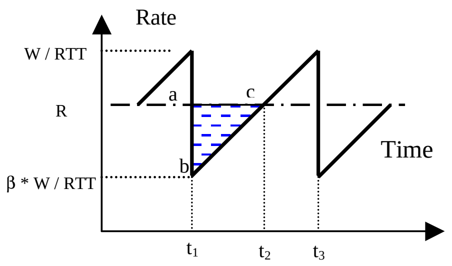

Fig. 3. Buffering Requirement for one Back-off

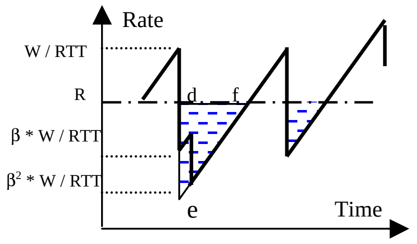

Fig. 4. Buffering Requirement for Two Continuous Back-offs

According to the ideal adaptation policy, R is the average of the achievable transmission rate. The application fetches data from the receiver-side buffer at this rate, but the network delivers data to the buffer at a rate of the saw-tooth shape. Therefore, the data buffering required to smooth the rate oscillations in one saw-tooth is equal to the area of triangle Δabc in Figure 3, which is:

$$\Delta abc = \frac{1}{2\alpha MSS} \times (\frac{1-\beta}{1+\beta})^2 \times R^2 \times RTT^2 \qquad (3)$$

The details of the derivation are in the technical report [12].

From Equation (3), we see the buffering requirement is related to the selection of AIMD parameters (α,β). More importantly, this buffering requirement is in proportion to the square of the rate and RTT, which is significant for high rate and long RTT applications. This result indicates that interactive applications might not want to fully utilize all the available bandwidth in order to avoid this buffering cost.

With the amount of buffering indicated by Equation (3), an application will have a stable playback quality within one saw-tooth period. If the bandwidth share is very stable and the saw-tooth shape is uniform over time, then the application keeps a stable quality all the time and utilizes its entire bandwidth share.

However, in the Internet, even a relatively stable bandwidth share would not produce a uniform saw-tooth shape. Very often, back-offs come closely to each other for a while, and spread sparsely for another while. With the ideal adaptation, the application changes its playback quality at every saw-tooth period. If the application prefers a more stable playback quality, it should buffer more data for the rate oscillations caused by closely spaced back-offs.

Figure 4 shows an example of two closely spaced back-offs. If an application wants to keep a stable playback quality when two back-offs happen continuously, the buffering requirement would be at most be the area of triangle Δdef, which is

$$\Delta def = (1+2\beta)^2 \Delta abc = \frac{1}{2\alpha MSS} \times (\frac{1+\beta-2\beta^2}{1+\beta})^2 \times R^2 \times RTT^2 . \qquad (4)$$

Similar derivations can be applied to the buffering requirement that is used to smooth more than 2 continuous back-offs.

3.3 Buffering Requirement for AIMD-based TCP-friendly Congestion Control Protocols

Early research [13,3] has studied how to make AIMD-based congestion control friendly to TCP traffic in the Internet. A simplified result from the TCP-friendliness study can be expressed as a constraint on its α and β parameters: $\alpha = \frac{3(1-\beta)}{1+\beta}$. The derivation is available in [12]. With this α and β relationship, we can refine the buffering requirement in Equation (3) as:

$$\Delta abc = \frac{\alpha}{18MSS} \times R^2 \times RTT^2 \cdot \qquad (5)$$

4. Experiments

We make several experiments to verify our derivation of minimal buffering requirements with various pairs of AIMD parameters. All these experiments are conducted in the ns simulator [14].

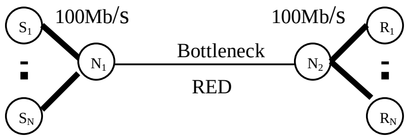

Fig. 5. Basic Experiment Topology

We use the simple topology shown in Figure 5, which has N nodes on each side of a bottleneck link. The bottleneck link uses RED queue management with ECN [15]. Every pair of nodes (S_i,R_i) corresponds to a flow which is either an ECN enabled AIMD-based flow or a UDP flow. The number of flows, the values of the bottleneck link bandwidth and its delay are stated within each experiment.

Each experiment includes two steps. First, we run a non-adaptive infinite source application over an AIMD flow to monitor available rate for the flow. Second, after we have the whole trace of the achievable rate by the AIMD congestion control, we simulate the application's adaptation behavior with this available bandwidth, and compare the buffering requirement of different adaptation policies. In this step, we use a simulated adaptive application, which is a fine-grain layer-encoded application with a rate range of 100Kbps to 1.5Mbps, in constantly spaced layers of 50Kbps.

4.1 Comparisons of Various Adaptation Policies

The first experiment we conducted illustrates the buffering requirements and bandwidth efficiency for various adaptation policies. In this experiment, the bottleneck link bandwidth is set to 1Mbps with 40ms delay. To produce regularly behaved saw-tooth rate shape we run a single AIMD(1,1/2) flow with a 256B packet size. Parallel with this AIMD(1,1/2) flow, a UDP flow runs through this bottleneck link. We adjust the UDP flow's rate to control the available bandwidth of the AIMD(1,1/2) flow. In this experiment, the UDP flow is set to 400Kbps CBR except for a short 10 seconds burst to 600Kbps.

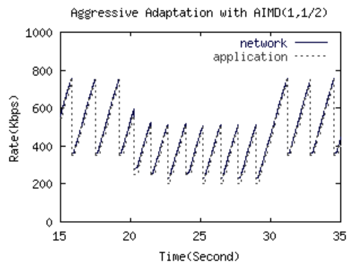

Fig. 6. Aggressive Adaptation

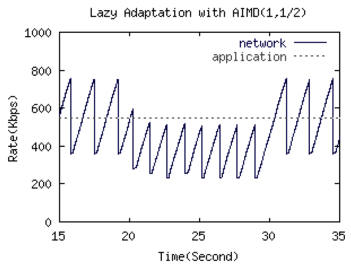

Fig. 7. Lazy Adaptation

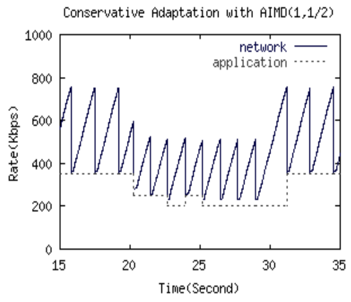

Fig. 8. Conservative Adaptation

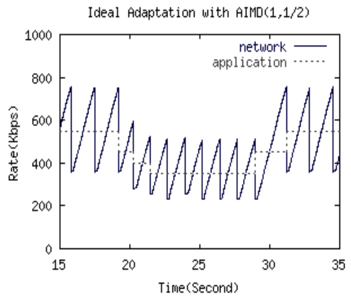

Fig. 9. Ideal Adaptation

Figures 6 – 9 show the application rate together with the network transmission rate for each adaptation policy. We summarize the result of this experiment in Table 1. For the buffering requirement, both aggressive and conservative adaptation policies keep the application's sending rate lower than the available network transmission rate, thus they don't need any receiver side buffering. The lazy adaptation has a relatively large buffering requirement, which is related to the duration of transmission rate degradation. In this experiment, a 300KB buffer is about 5 seconds delay for the application. For the ideal adaptation, it requires 7.8KB to smooth its saw-tooth size, which is about 100ms for the AIMD flow with a 600Kbps sending rate. Any other adaptation policy that maximizes the throughput would experience a delay between the delays of the ideal and lazy adaptation policies.

Table 1 also summarizes the bandwidth efficiency and number of rate adjustments that happened during the experiment period shown in Figures 6 – 9. Clearly the conservative adaptation has a relatively stable playback quality, but a low bandwidth efficiency. All the other three policies have a high bandwidth efficiency. The reason for not using 100% bandwidth is that the application is layer-encoded, and its sending rate can only approximate the available bandwidth with a sum of its existing layer rates.

Table 1. Comparison of Various Adaptation Policies

Adaptation Policy	Minimal Buffer Requirement	Bandwidth Efficiency	Number of Quality Adjustments
Aggressive Adaptation	0	92%	105
Conservative Adaptation	0	58%	5
Lazy Adaptation	> 300KB	92%	0
Ideal Adaptation	7.8KB	92%	5

4.2 Buffering Requirements of the Ideal Adaptation Policy

In this experiment, we verify the buffering requirement relationship described by Equation (3). We use only one AIMD flow with a 256B MSS, and one UDP CBR flow. First, we set the bottleneck link bandwidth to 1.5Mbps with a 40ms one-way delay. We vary the rate of the UDP flow to produce available bandwidth from 100Kbps to 1.5Mbps for the AIMD flow. We run this experiment 3 times with different AIMD flow parameters: (1,1/2), (1/3, 4/5), and (1/5, 7/8). The measured buffering requirements are plotted in Figure 10. Second, we give a 1.2Mbps available bandwidth to the AIMD flow and vary the bottleneck propagation delay from 10ms to 120ms. The result of the buffering requirement versus the RTT is in Figure 11.

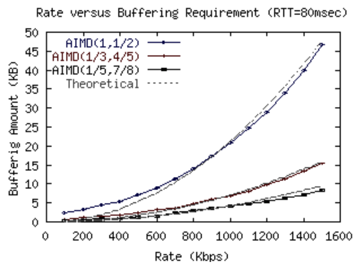

Fig. 10. Rate versus Buffering

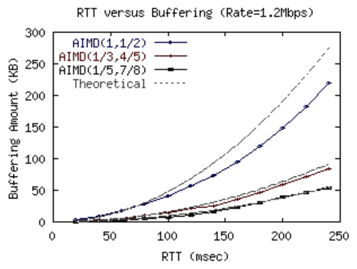

Fig. 11. RTT versus Buffering

The experiment result shows AIMD parameters have an effect on the minimal buffering requirement. For example, a 1Mbps AIMD(1,1/2) flow on an 80ms RTT path requires more than 20KB buffering. This amount of buffering is equivalent to more than 160ms delay for this flow, which is too large for interactive applications [16]. Choosing a small AIMD parameter pair (α,β) allows the buffering delay experienced by the flow to be reduced. For example, by using AIMD (1/5,7/8), the buffering requirement can be reduced to 5KB, which maps to 40ms delay for this flow.

However, the experiment result also shows that the buffering requirement increases quadratically with rate and RTT, which is problematic for interactive applications with high rate and long RTT. In Figure 10, even with AIMD(1/5,7/8), the buffering delay becomes significant as the application's sending rate gets larger.

RTT has a similar effect on the buffering size as flow rate does, but the case is worse because a large RTT for interactive applications usually corresponds to a small buffering delay budget. For flows with a small RTT, for example 20ms, the resulting buffering delay is less than 10ms for a 1.2Mbps data rate. This indicates that the required minimal buffering is not significant for interactive applications on a metropolitan area network or even a WAN between cities not far away. However, it is problematic for interactive applications across oceans or between coasts within a continent (e.g. 80ms RTT in US). For example, for a flow with 100ms RTT and 1.2Mbps data rate, the required buffering delay is about 300ms, which is much more than most interactive applications can tolerant.

The buffering requirement results measured in this experiment slightly differ from the ones predicted by Equation (3). We believe one reason is that RTT is not constant as we assumed in Equation (3). Another reason is that the implementation of AIMD actually increases its rate sub-linearly rather than linearly, where the derivation of Equation (3) assumes that the additive part of the AIMD algorithm behaves linearly.

Even with this sub-linear increment, the buffering requirement is still quadratic to the application's rate and RTT. This result confirms our claim that interactive applications may not always prefer to maximize their throughputs, since they may come at the expense of unacceptable end-to-end delay.

5. Conclusion and Future Work

In this paper, we have addressed the minimal buffering requirements of adapting the application data rate to the average available bandwidth, which maximizes a multimedia application's throughput. The minimal buffering requirement is used to compensate for the rate oscillations of congestion control protocols. We derived the relationship between the minimal buffer requirements and congestion control's AIMD parameters, application rate, and RTT. Our result indicates that choosing an AIMD-based TCP-friendly congestion control with a small increment parameter can reduce the buffer requirement, because the buffer requirement is proportional to the increment parameter. However, the buffer requirement is also proportional to the square of the application's sending rate and round-trip-time. Thus, adapting application sending rate closely to the average available bandwidth is not a preferable adaptation policy for interactive applications with high rate and long RTT.

In this paper, we studied the buffering requirement of AIMD congestion control. Besides AIMD-based congestion control protocols, several other algorithms like binomial congestion control [17], Equation-based congestion control [2], and TCP emulation at receivers (TEAR) [18] have been proposed to reduce the oscillations in the application sending rate. Evaluation of the buffering requirements of multimedia applications using these protocols is one of our targets for future work.

Reference

1. Sally Floyd, and Kevin Fall. “Promoting the Use of End-to-End Congestion Control in the Internet” *IEEE/ACM Transactions on Networking*, August 1999.
2. Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. “Equation-based Congestion Control for Unicast Applications.” In *Proceedings of ACM SIGCOMM 2000*, August 2000.
3. Yang Yang, and Simon Lam. “General AIMD Congestion Control” In *Proceedings of ICNP 2000*, Osaka, Japan, Nov 2000.
4. R. Rejaie, M. Handley, and D. Estrin. “An End-to-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet”. In *Proceedings of IEEE INFOCOM’99*, Mar, 1999.
5. Mark Allman, Vern Paxson. “On Estimating End-to-End Network Path Properties”, In *Proceeding of SIGCOMM’99*, pp. 263-274, 1999.
6. K. Park, G. Kim, and M. Crovella. “On the Relatioinship Between File Sizes, Transport Protocols and Self-Similar Network Traffic”. In *Proceedings of ICNP’1996*.
7. S. Jacobs and A. Eleftheriadis. “Providing Video Services over Networks without Quality of Sevice Guarantees”. In *Proceedings of World Wide Web Consortium Workshop on Real-time Multimedia and the Web*, 1996.
8. Charles Krasic and Jonathan Walpole. “QoS Scalability for Streamed Media Delivery”, OGI CSE Technical Report CSE-99-11, September, 1999
9. John Byers, Michael Luby, and Michael Mitzenmacher. “Fine-Grained Layered Multicast”, In *Proceedings of IEEE INFOCOM 2001*, April 2001.
10. Charles Krasic, Jonathan Walpole, Kang Li, and Ashvin Goel. “The Case for Streaming Multimedia with TCP”, OGI-Tech-Report 01-003, March, 2001.
11. R. Rejaie, M. Handley, and D. Estrin. “Quality Adaptation for Congestion Controlled Video Playback over the Internet”. In *Proceedings of SIGCOMM’99*, Oct., 1999.
12. K. Li, C. Krasic, J. Walpole, M. H. Shor, and C. Pu, “The Minimal Buffering Requirements of Congestion Controlled Multimedia Applications”, OGI-Tech-Report 01-008, June, 2001.
13. Sally Floyd, Mark Handley, and Jitendra Padhye. “A comparison of equation-based congestion control and AIMD-based congestion control.” Under submission. Available at http://www.aciri.org/tfrc.
14. ns: UCB/LBNL/VINT Network Simulator, http://www-mash.cs.berkeley.edu/ns/ns.html
15. S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance”, *IEEE/ACM Transactions on Networking*, vol.1, pp.397-413, August 1993.
16. Stuart Cheshire. “Latency and the Quest for Interactivity”. White paper for the Synchronous Person-to-Person Interactive Computing Environments Meeting, San Francisco, November 1996. Available at http://www.stuartcheshire.org.
17. D. Bansal and H. Balakrishnan. “Binomial Congestion Control Algorithms”, In *Proceedings of INFOCOM 2001*, April 2001.
18. I. Rhee, V. Ozdemir, and Y. Yi. “TEAR: TCP emulation at receivers - flow control for multimedia streaming”. http://www.csc.ncsu.edu/eos/users/r/rhee/WWW/export/tear_page.