The Minimal Buffering Requirements of Congestion Controlled Interactive Multimedia Applications[Θ]

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 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 oscillate rates according to network conditions. To produce a smooth perceptual quality, multimedia applications use buffering and rate adaptations to compensate 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 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 found that the buffering requirement is proportional to the parameters of AIMD algorithm and quadratic to the application's sending rate and round-trip-time; and we verify this relationship through experiments. Our result indicates that adaptations of maximizing throughput are not suitable for interactive applications with high rate or long round-trip-time.

1. Introduction

Interactive multimedia applications, such as videoconferencing and IP telephony, are becoming an important component of the Internet. Unlike traditional broadcast networks used for television or cable, the modern Internet is highly dynamical and is characterized with rapidly changing conditions. Applications must use congestion control protocols to react to the dynamics of the Internet, in order to keep the Internet's stability [FF99].

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 could cause significant delays to applications. In recent years, researchers have proposed various TCP-friendly congestion control protocols, such as equation-based congestion control [FHPJ00] and general *additive-increase-and-multiplicative-decrease (AIMD)* based congestion control [YL00]. These TCP-friendly congestion control protocols have significantly improved the performance of multimedia applications over the Internet [RHE99a], 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

[Θ] This work was supported in part by DARPA/ITO under the Information Technology Expeditions, Ubiquitous Computing, Quorum, and PCES programs, and in part by NSF Grant CCR-9988440 and ECS-9988435.

huge varieties of its applications, their users, and usage patterns [AP99, PKC96]. As a result of these varieties, the Internet bandwidth share of an application keeps varying with time. In addition to the Internet dynamics, congestion control protocols have to probe the bandwidth share for applications, because no information about bandwidth sharing is directly available to congestion control protocols. The process of probing for bandwidth and reacting to observed congestion induce oscillations in the achievable transmission rate, and is an integral part of the nature of all end-to-end congestion management algorithms.

Applications generally use buffering at the receiver side to smooth the oscillations of the data arriving rate (the network transmission rate), because users of interactive multimedia applications prefer a smooth playback quality rather than playing back at the network transmission rate. In addition to buffering, multimedia applications adjust their playback qualities based on the available transmission rates. 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 [JE96, KW99, BLM01].

In this paper, we study the buffering requirements of different *adaptation policies*, which are application's ways of estimating the network transmission rate and adjusting its playback rate. Adaptation policies have significant impacts on the buffering requirements. A sluggish adaptation that loosely tracks the network transmission rate would require a large amount of buffering to sustain the application's playback when the transmission rate goes lower; an aggressive adaptation tracking the network transmission rate closely requires less buffering.

In particular, we notice a trend of research efforts toward an adaptation policy that tries to fully utilize the available bandwidth while preserves a smooth playback quality. Several existing research work [JE96, KWLG01, RHE99b] have designed mechanisms, such as smart buffer managements and fine-grained adaptations, to push the adaptation toward the direction of maximizing throughput. These work are mainly in the context of streaming media over the Internet, with a clear benefit – bandwidth efficiency. Without inspecting the effect of this adaptation policy, people might think 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; and depending on an application's delay constraint, this delay 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 as the increments of its rate and round-trip-time (RTT). This result indicates that using a small increment parameter 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 how it adapts; in Section 3, we describe the general AIMD algorithm, and

[1] Appendix A.1 presents the rule of choosing AIMD parameters for TCP-friendliness.

the analytical derivation of buffering requirement; in Section 4, we present the experiment architecture and results. Section 5 concludes the paper and describes some future work.

2. Buffering and Adaptations

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

2.1 Application Structure

Figure 2.1 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 through 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 under the limit of congestion control protocol and is put into a receiver side buffer. The data sink fetches data from the buffer and presents it to users.

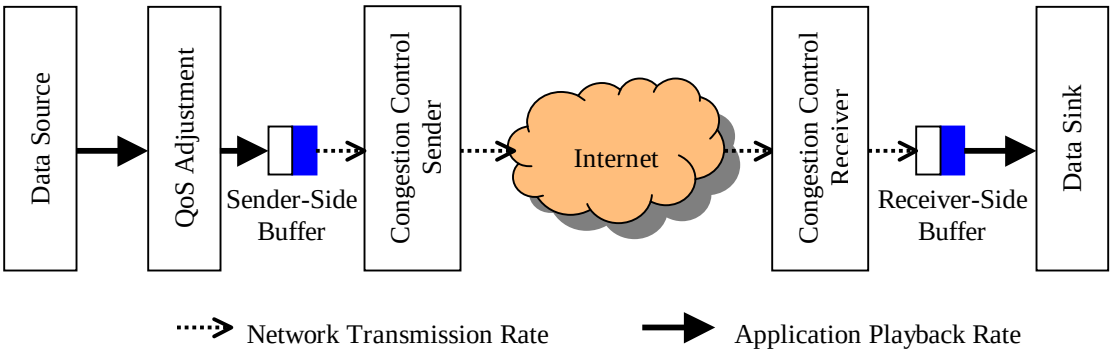

Figure 2.1: A QoS-Adaptive Application over the Internet

The transmission rate over the Internet oscillates over time. To achieve a stable playback quality at the data sink, a receiver-side buffering and a sender-side adaptation are used in this structure. We assume that a constant playback quality of the application maps to a constant bit rate (CBR)[3]. 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, so that the sink can keep playing 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 would 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 network rate drop 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 managements can be found in recent research work [KWLG01, RHE99b].

---

[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.

[3] In reality, a constant quality could map to a variable bit rate [KW99], which will be more complex but won't invalidate the buffering delay derivation in this work.

To reduce the buffering requirements, the target application makes QoS adaptation to adjust 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 work have shown ways of making fine-grained rate adaptations to the available bandwidth. For example, Jacobs et al. [JE96] adapt encoding parameters according to the available bandwidth; Krasic et al. [KW99] 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 [BLM01] to multicast environments.

2.2 Adaptation Policies

For layered-based adaptations, adaptation policies are rules determining when a layer should be added or removed. Buffering requirements at the receiver side are closely related to how the application adapts its rate. In this section, we use examples to show this relationship.

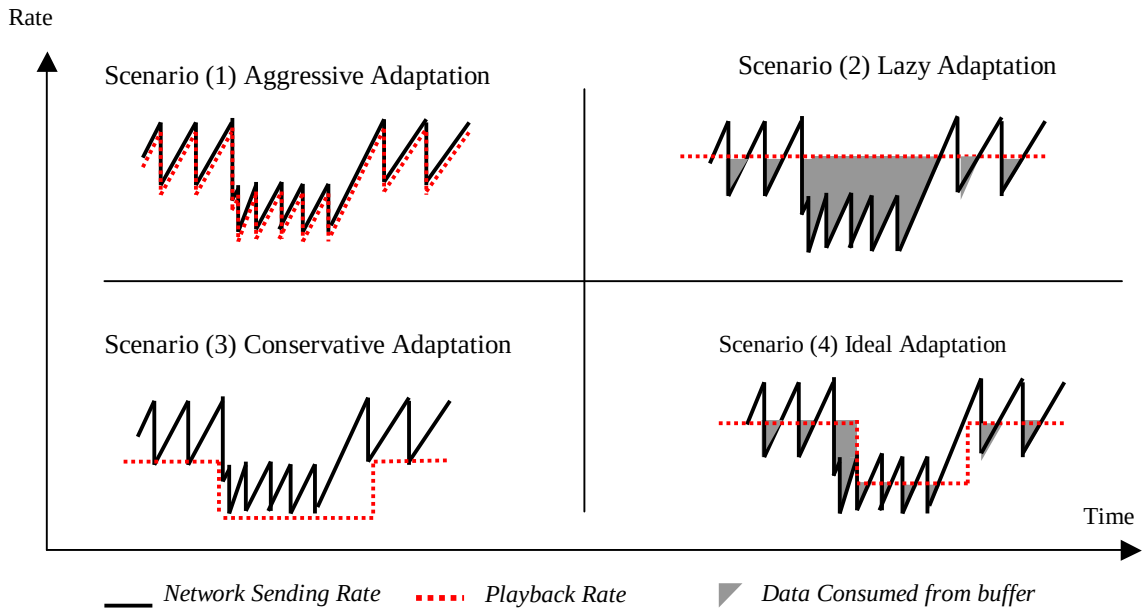

Figure 2.2: Buffering Requirements of Different Backing-off Scenarios

Figure 2.2 shows the running scenarios of four adaptation policies with the same saw-tooth shape transmission rate, which is a typical result of using up to the rate achievable by AIMD-based congestion control protocols.

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, layers are dropped until the application's sending rate is equal to or lower than the network transmission rate. This adaptation policy does not require any receiver side buffering but results quality variations.

Scenario (2) illustrates a lazy adaptation policy that is opposite to the aggressive one and produces a very stable playback rate. The policy does not adjust application's sending rate according to the available bandwidth in the network. However, it requires to buffer a large

amount of data to compensate the rate variations 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 in 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 the adaptation chooses layers to have a rate lower than the lowest rate of the recent saw-tooth shape. With this policy, applications require no receiver-side buffering, and give users a relative stable playback rate. However, this policy doesn't let the application use all the achievable transmission capacity detected by the congestion control protocols.

Scenario (4) presents an adaptation policy that is called *ideal adaptation* by us. It is ideal because we assume it knowing ahead the rate of one saw-tooth in the future. Since it knows one saw-tooth ahead of the future, 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 the throughput. The buffering requirement by this ideal adaptation policy is the amount of data to smooth one saw-tooth of the network transmission rate.

2.3 Cost for the Ideal Adaptation

This ideal adaptation might not be a preferred adaptation policy by applications. However, as an extreme case of adaptations, it exposes the minimal buffering requirement for maximizing the throughput.

For other "realistic" policies, if they push the application rate to the upper bound of the network transmission rate achievable by congestion control protocols, they require at least the same amount of buffering of using this ideal adaptation. This buffering cost is caused by the mechanisms that smooth out the inherent rate oscillation in the congestion control protocols. This buffering is required because the application does not want to oscillate its quality with the rate of congestion control (such as the rate variations within one saw-tooth). This buffering delay could be significant depending on the application's sending rate and round-trip-time. At the time this buffering delay cost is too much, this "ideal" adaptation would no longer be ideal to interactive applications at all. We give a simple derivation for this inherent 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 to avoid congesting the network. It is a window-based congestion control protocol, which uses a congestion window to limit the maximum amount of data sent out by the application within one round-trip-time.

In this section, we first describe the GAIMD algorithm, and then derive the minimal buffer requirement to smooth the rate oscillations in AIMD-based congestion control protocols.

3.1 GAIMD Algorithm

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

$$\begin{cases} \text{(1) Additive Increase:} & W_{t+RTT} \leftarrow W_t + \alpha \times MSS; \quad \alpha > 0 \\ \text{(2) Multiplicative Decrease:} & W_{t+\delta} \leftarrow \beta \times W_t; \quad 0 < \beta < 1 \end{cases} \tag{3.1},$$

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[4]. α and β are parameters of AIMD algorithm, which control the paces of the additive increase and multiplicative backing off respectively. The rate behavior of GAIMD algorithm is similar to the saw-tooth shape of TCP congestion control, which uses an AIMD(1, ½).

3.2 Minimal Buffering Requirement

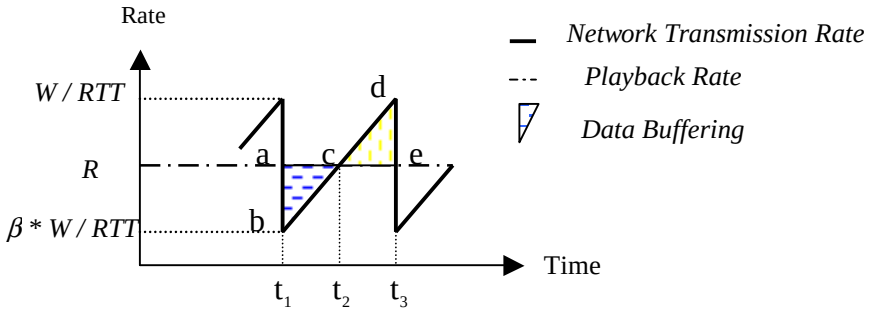

Figure 3.1: Buffering Requirement of an AIMD-based Congestion Control

To determine the buffering requirement for smoothing the rate oscillations, we need to describe how the rate of AIMD-based protocol evolves along time. Figure 3.1 shows a AIMD flow with a playback rate R. For an AIMD flow, the achievable rate in a RTT is its window size divided by the RTT, and the window size evolution of a GAIMD flow is controlled by the algorithm stated in (3.1): 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.

With an ideal adaptation, the application playback rate is the average of the achievable transmission rate:

$$R = (\frac{W}{RTT} + \beta \frac{W}{RTT}) / 2 \qquad (3.2),$$

The application fetch data from the receiver-side buffer in this rate R, but the network delivers data to the buffer in 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 3.1, which is:

$$\Delta abc = \frac{1}{2\alpha MSS} \times (\frac{1-\beta}{1+\beta})^2 \times R^2 \times RTT^2 \quad (3.4).$$

The details of the derivation are in Appendix A.2.

From this simple derivation, we can 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 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 (3.4), an application will have a stable playback quality within one saw-tooth period. If the bandwidth share is very stable and saw-tooth shape is evenly repeated along time, then the application keeps a stable quality all the time and utilizes its entire bandwidth share.

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

However, in the Internet, even a relative stable bandwidth share would not produce a regularly repeated 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, application changes its playback quality at every saw-tooth period. If application prefers a more stable playback quality, it should buffer more data for the rate oscillations caused by closely spaced back-offs.

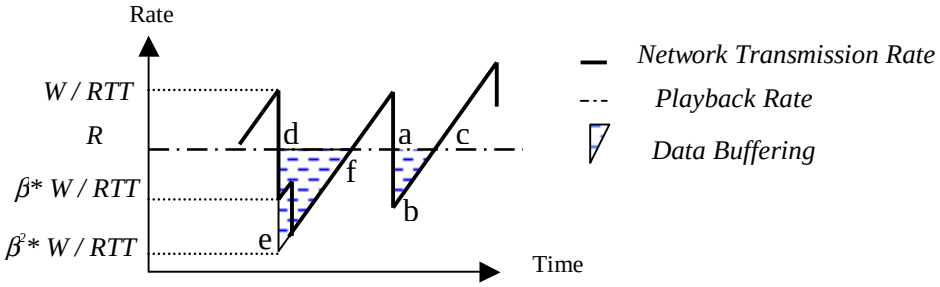

Figure 3.2: Buffering Requirement for Two Continuous Back-offs

Figure 3.2 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

$$\Delta def = (1+2\beta)^2 \Delta abc = \frac{1}{2\alpha MSS} \times (\frac{1+\beta-2\beta^2}{1+\beta})^2 \times R^2 \times RTT^2 \quad (3.5).$$

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 work [FHP00, YL00] have studied how to make AIMD-based congestion control friendly to other 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 Appendix A.1. With this α and β relationship, we can refine the buffering requirement to smooth the inherent oscillation of AIMD-based TCP-friendly congestion control as:

$$\Delta abc = \frac{\alpha}{18MSS} \times R^2 \times RTT^2 \quad (3.6),$$

and the buffering requirement to smooth out two continuous backing off as:

$$\Delta def = (1+2\beta)^2 \Delta abc = (\frac{9-\alpha}{3+\alpha})^2 \times \frac{\alpha}{18MSS} \times R^2 \times RTT^2 \quad (3.7).$$

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 ns simulator [NS].

We use the simple topology[5] shown in Figure 4.1, which has N nodes on each side of a bottleneck link. The bottleneck link uses RED queue management with ECN [FJ93]. Every pair of nodes (S_i,R_i) corresponds to a flow which is either a ECN enabled AIMD-based flow or a UDP flow.

[5] The simulation is available for download at http://www.cse.ogi.edu/~kangli/buffering_delay.html

The number of flows, the bottleneck link bandwidth and its delay are set to different values to produce various experiment setups. Their values 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 the rate available to the flow. Second, after we have the whole trace of the achievable rate by the AIMD congestion control, we simulate application's adaptation behavior with the rate trace as the 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.

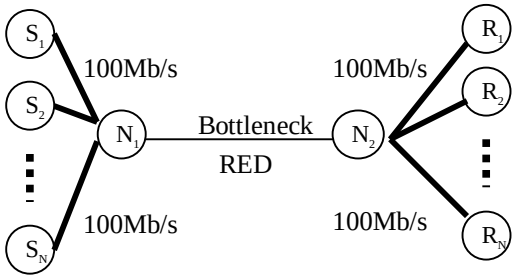

Figure 4.1: Basic Experiment Topology

4.1 Comparisons of Various Adaptation Policies

The first experiment we conducted is to illustrate the difference of 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 a short 10 seconds burst to 600Kbps.

For this particular rate trace, we plot the rate behaviors of our simulated layered application. Figures 4.2 – 4.5 show the application rate together with the network transmission rate for each adaptation policy. We summarize the result of this experiment in Table 4.1.

Table 4.1: 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

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 length 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 ideal and lazy adaptation policies.

Besides the buffering requirement, Table 4.1 also summarizes the bandwidth efficiency and numbers of rate adjustment happened during the experiment period shown in Figures 4.2 – 4.5. Clearly the conservative adaptation has a relative stable playback quality, but a low bandwidth efficiency. All the other three policies have a high bandwidth efficiency. The reason of 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 existing layer rates.

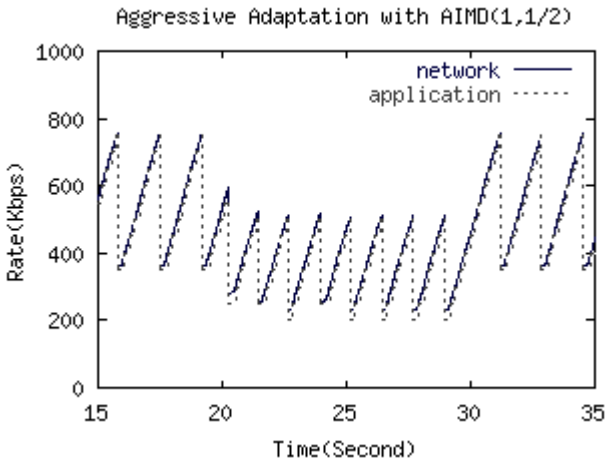

Figure 4.2: Aggressive Adaptation

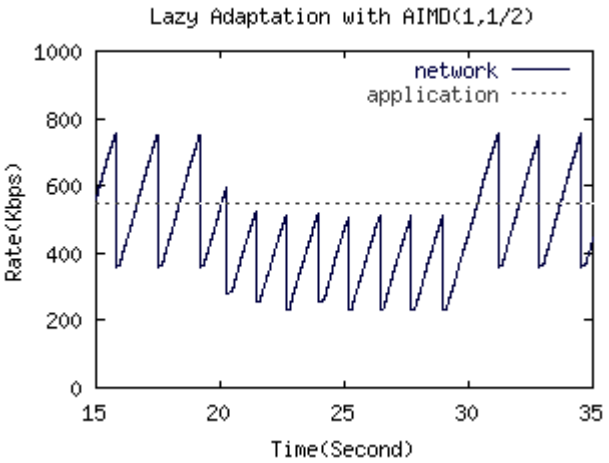

Figure 4.3: Lazy Adaptation

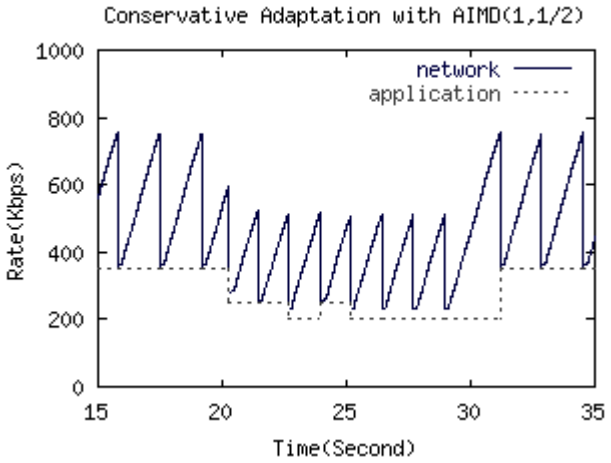

Figure 4.4 Conservative Adaptation

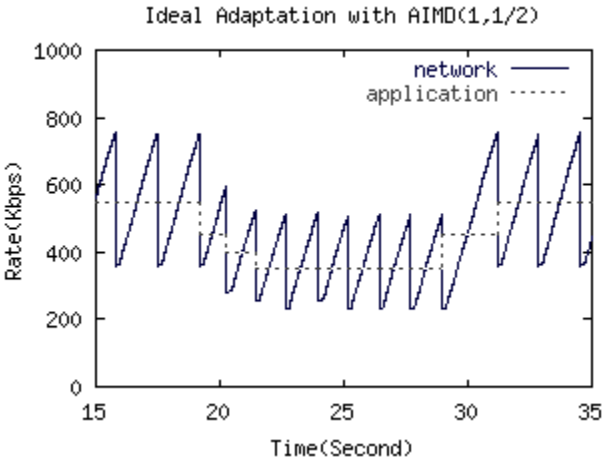

Figure 4.5: Ideal Adaptation

4.2 Buffering Requirements of the Ideal Adaptation Policy

In this experiment, we verify the buffering requirement relationship described by (3.4). 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 4.6. 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 4.7.

The experiment result shows AIMD parameters has an effect on the minimal buffering requirement. For example, a 1Mbps AIMD(1,1/2) flow on a 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 [C96]. Choosing a small AIMD parameter pair (α,β) is able to reduce the buffering delay experienced by the flow. 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.

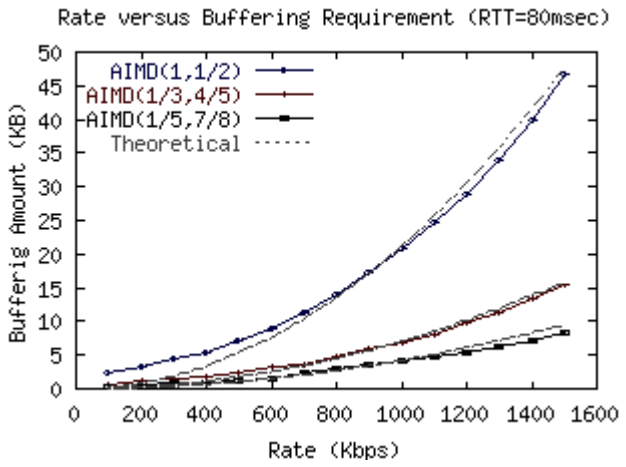

Figure 4.6 Rate versus Buffering

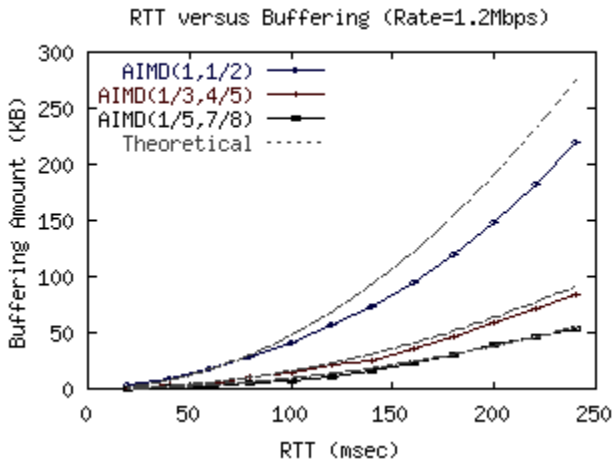

Figure 4.7 RTT versus Buffering

However, the experiment result also shows the buffering requirement increases quadratically with rate and RTT, which is problematic for interactive applications with high rate and long RTT. In Figure 4.6, 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 resulted 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 coast-to-coast 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.

Most of the buffering requirement results in this experiment are smaller than the ones predicted by (3.4). We believe one reason is that the implementation of AIMD actually increases its rate sub-linearly rather than linearly, where the derivation of (3.4) assumes that the additive part of 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 early claim that interactive applications do not prefer paying the cost of the buffering delay to maximize their throughputs. On the contrary,

we believe interactive applications should send in a lower rate than the rate detected by the congestion control protocols in order to avoid any buffering delay.

5. Conclusion and Future Work

In this paper, we have addressed the minimal buffering requirement of adapting the application data rate to the average available bandwidth, which maximize the multimedia application's throughput. The minimal buffering requirement is used to compensate the rate oscillation of congestion control protocols. For AIMD-based congestion control protocols, the required buffering is at least the amount of buffer required to smooth the saw-tooth rate shape.

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 multimedia 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 [BB01], Equation-based congestion control [FHPJ00], and TCP emulation at receivers (TEAR) [ROY00] 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 future work targets.

Another future work is about the interaction between application adaptation and congestion control. In this paper, we assume that rate adaptations do not change a flow's behavior. However, the bandwidth sharing system in the Internet is dynamic. The reduction in application's sending rate could reduce the application's competition with other traffic. Studying this interaction and taking its effect into account is another interesting topic that we are studying.

Reference

[AP99] Mark Allman, Vern Paxson. "On Estimating End-to-End Network Path Properties", In *Proceeding of SIGCOMM'99*, pp. 263-274, 1999.

[BB01] D. Bansal and H. Balakrishnan. "Binomial Congestion Control Algorithms", In *Proceedings of INFOCOM 2001*, April 2001.

[BLM01] John Byers, Michael Luby, and Michael Mitzenmacher. "Fine-Grained Layered Multicast", In *Proceedings of IEEE INFOCOM 2001*, April 2001.

[C96] 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

[FF99] Sally Floyd, and Kevin Fall. “Promoting the Use of End-to-End Congestion Control in the Internet” *IEEE/ACM Transactions on Networking*, August 1999. Available at http://www.aciri.org/floyd/papers.html

[FHP00] 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.

[FHPJ00] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. “Equation-based Congestion Control for Unicast Applications.” In *Proceedings of ACM SIGCOMM 2000*, August 2000.

[FJ93] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance”, *IEEE/ACM Transactions on Networking*, vol.1, pp.397-413, August 1993.

[JE96] 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.

[KW99] Charles Krasic and Jonathan Walpole. "QoS Scalability for Streamed Media Delivery", OGI CSE Technical Report CSE-99-11, September, 1999

[KWLG01] Charles Krasic, Jonathan Walpole, Kang Li, and Ashvin Goel. “The Case for Streaming Multimedia with TCP”. Submitted for publication. Available as OGI CSE Technical Report CSE-01-003, March, 2001.

[NS] ns: UCB/LBNL/VINT Network Simulator (Version 2) http://www-mash.cs.berkeley.edu/ns/ns.html

[PKC96] 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*.

[RHE99a] 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.

[RHE99b] R. Rejaie, M. Handley, and D. Estrin. “Quality Adaptation for Congestion Controlled Video Playback over the Internet”. In *Proceedings of SIGCOMM’99*, Oct., 1999.

[ROY00] Injong Rhee, Volkan Ozdemir, and Yung Yi. “TEAR: TCP emulation at receivers - flow control for multimedia streaming”. Technical Report Draft available at http://www.csc.ncsu.edu/eos/users/r/rhee/WWW/export/tear_page

[YL00] Yang Yang, and Simon Lam. “General AIMD Congestion Control” In *Proceedings of ICNP 2000*, Osaka, Japan, Nov 2000.

Appendix

A.1 Average Throughput of a GAIMD Congestion Control

For a flow that uses GAIMD algorithm described in (3.1), we can estimate its throughput given its packet loss probability *p*.

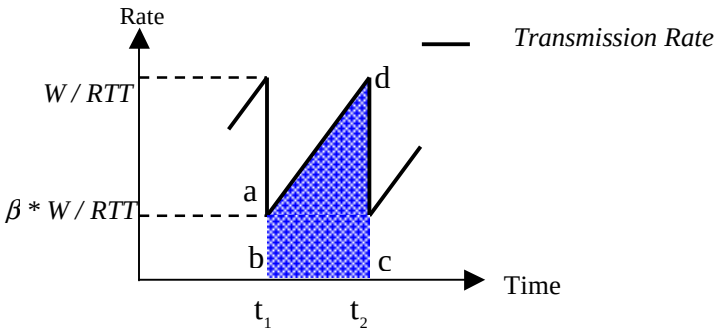

Figure A.1: Throughput Derivation for an AIMD flow

Figure A.1 illustrates the flow's rate oscillations along the time, assuming the flow's packet losses are evenly distributed along the time. Because of these periodic packet losses, the flow's congestion window shows a saw-tooth pattern. We assume the flow's congestion window size reaches W upon the arriving of every packet loss event, and backing off to βW after the event. Thus, the flow's rate keeps oscillating between W/RTT and $\beta W/RTT$.

Since a GAIMD flow increase its window size by $\alpha * MSS$ per RTT, the time for the flow's congestion window increasing from βW to W can be derived by:

$$t_2 - t_1 = \frac{(1-\beta)W}{\alpha MSS} RTT \qquad \text{(A.1).}$$

The total amount of data sent out during this time (t_2-t_1) is indicated by the area of the shaded region *abcd*, which can be derived as:

$$Area(abcd) = \beta W + (\beta W + \alpha MSS) + (\beta W + 2\alpha MSS) + \ldots + W = \frac{1-\beta^2}{2\alpha MSS} W^2 \quad \text{(A.2).}$$

Since one packet of every $Area(abcd)/MSS$ amount of packets is lost and the packet loss probability is p, we can have

$$\frac{1}{p} = \frac{Area(abcd)}{MSS} = \frac{1-\beta^2}{2\alpha MSS^2} W^2 \qquad \text{(A.3).}$$

Thus, we have

$$W = \frac{\sqrt{\frac{2\alpha}{(1-\beta^2)}} \times MSS}{\sqrt{p}} \qquad \text{(A.4).}$$

Finally, the average throughput of the flow can be derived by:

$$\overline{R} = \frac{Area(abcd)}{t_2 - t_1} = \frac{(1+\beta)W}{2RTT} = \sqrt{\frac{\alpha}{2} \times \frac{1+\beta}{1-\beta}} \times \frac{MSS}{RTT\sqrt{p}} \qquad \text{(A.5).}$$

For TCP with AIMD parameter α=1 and β=½, its throughput can be expressed as

$$\overline{R} = \frac{\sqrt{3/2} \times MSS}{RTT\sqrt{p}} \qquad \text{(A.6)}$$

If an GAIMD flow wants to have the same average throughput as TCP when they share the same *RTT*, packet size *MSS*, and packet losses rate p, the GAIMD flow's α and β parameters have to satisfy the following equations:

$$\sqrt{\frac{\alpha}{2} \times \frac{(1+\beta)}{(1-\beta)}} = \sqrt{\frac{3}{2}} \qquad \text{(A.7)},$$

which can be further simplified as

$$\alpha = \frac{3(1-\beta)}{1+\beta} \qquad \text{(A.8)}.$$

A.2 Buffer Requirement of a GAIMD Congestion Control

The rate of a GAIMD flow varies because of its way probing bandwidth and making congestion avoidance. Once the transmission rate is lower than the receiver play out rate, users will perceive the transmission rate oscillations unless there is receiver side data buffering. Receiver side buffering is a popular way to tolerant this rate oscillation. The amount of receiver-side buffering is needed for the transmission rate to catch up the playing out rate.

Figure 3.1 shows a GAIMD flow with a playing out rate R. We assume the GAIMD flow's transmission rate periodically varies from β * W / RTT to W/RTT.

Since the playing out rate can not be higher than the average the transmission rate, (Otherwise, it will run out the receiver side buffer), the playing out rate is limited by:

$$R = (\frac{W}{RTT} + \beta\frac{W}{RTT})/2 \qquad \text{(A.9)}.$$

With this playing out rate, the required data buffering size to avoid receiver side buffer underflow is equal to the area of triangle abc in Figure 3.1, which is:

$$\Delta abc = \frac{1}{2}[t_2 - t_1][R - \beta\frac{W}{RTT}] = \frac{1}{2}[\frac{(1-\beta)W}{2\alpha MSS}RTT][\frac{1+\beta}{2} \times \frac{W}{RTT} - \beta\frac{W}{RTT}] = \frac{(1-\beta)^2}{8\alpha MSS}W^2 \quad \text{(A.10)}.$$

Since playback rate R is equal to the average of the transmission rate, we have $W = \frac{2}{1+\beta}R \times RTT$, and the receiver-side buffering is

$$\Delta abc = \frac{1}{2\alpha MSS} \times (\frac{1-\beta}{1+\beta})^2 \times R^2 \times RTT^2 \qquad \text{(A.11)}.$$