User-Centric Evaluation of TCP-friendly Congestion Control for Real-Time Video Transmission

Michael Welzl[1] (corresponding author) and Werner Stadler[2]

[1] Institute of Computer Science
Distributed and Parallel Systems Group
University of Innsbruck
Technikerstr. 25, A-6020 Innsbruck, Austria
michael.welzl@uibk.ac.at, Phone: +43 (512) 507-6110, Fax: -2977
[2] Institute for Telecooperation
University of Linz
Altenberger Str. 69, A-4040 Linz, Austria
wvrn@wvrn.cc

Abstract. As more and more TCP-friendly congestion control mechanisms are introduced to the research community, it becomes increasingly difficult for designers of adaptive multimedia applications to decide which one should be used. We argue that important parameters such as the smoothness of existing mechanisms should be evaluated from a user-level perspective and describe how we have undertaken this effort for real-time video transmission.

Keywords: TCP-friendly Congestion Control, User test, Streaming Video

1 Introduction

The concept of Quality of Service (QoS) exists at various logical layers that may or may not correspond with OSI layers. At the physical layer, for example, the quality of a link may be a function of cable properties — the signal-to-noise ratio would be an important parameter in this context. From the user perspective, the QoS of a link is the quality of the service (such as the comprehensibility of spoken words in the case of a telephone), which is in fact a function of all involved elements — the application and the transport layer as well as the network, data link and physical layers. From the perspective of a network researcher however, the quality of a congestion control mechanism could, for example, be described in terms of responsiveness, fairness and stability [21].

In addition to the common requirement of TCP-friendliness if a mechanism is to be used in the Internet, it is generally known that a smooth rate is helpful for adaptive multimedia applications [15]. Some mechanisms show a smoother rate than others, but they may have certain other disadvantages. We believe that the quality that can be attained with existing TCP-friendly congestion

control mechanisms should be evaluated from the perspective of end users in order to provide some guidance to application programmers; in particular, with the advent of the Datagram Congestion Control Protocol (DCCP) [16] and its numerous possible embedded mechanisms, such guidance seems to become a necessity.

As a start, we have focused on a single mechanism and carried out user tests to evaluate the perceptual quality that corresponds with the rate smoothness of a video stream. Our tests were based on the assumption that rate fluctuations will somehow correspond with visible quality fluctuations; this is probably true for interactive applications (e.g., video conferencing) but not for unidirectional applications, where large buffers could be used to compensate for rate fluctuations.

We provide an overview of TCP-friendly congestion control mechanisms in the next section. In sections 3 and 4, the test environment and setup (types of movies with parameter settings and questionnaire) are described; the results are given in section 5 and section 6 concludes.

2 TCP-friendly Congestion Control: a Brief Overview

Perhaps the simplest commonly known TCP-friendly protocol is the *Rate Adaptation Protocol (RAP)*: it imitates the behavior of TCP in a rate-based manner but does not retransmit lost packets. AIMD ("Additive Increase, Multiplicative Decrease") behavior is not achieved by changing a congestion window but through calculation [14]. In [13], RAP is used as part of an end-to-end architecture for real-time multimedia data transmission.

In the case of TCP and RAP, AIMD corresponds with additively increasing by a factor $\alpha = 1$ (one packet per round-trip time (RTT)) if no loss occurs and multiplying the rate with a factor $\beta = 0.5$ in response to a lost packet. The stability of AIMD does not rely on these factors. Yang and Lam introduce *General AIMD Congestion Control (GAIMD)* [20] as a TCP-friendly generalization of AIMD; their finding is that it is possible to achieve a TCP-friendly rate with different values for α and β as long as the relation

$$\alpha = \frac{4(1-\beta^2)}{3} \tag{1}$$

holds. This means that in order to remain fair towards TCP, one must consider a trade-off between the responsiveness and smoothness of an AIMD algorithm — for example, for $\beta = 7/8$, α must be set to 0.31. Fig. 1 depicts this relationship between the GAIMD increase and decrease factors.

Two well-known mechanisms use completely different approaches to achieve a smooth rate while maintaining a TCP-friendly rate allocation:

- The well known *TCP-friendly Rate Control (TFRC)* protocol uses an equation which models the steady-state behavior of TCP:

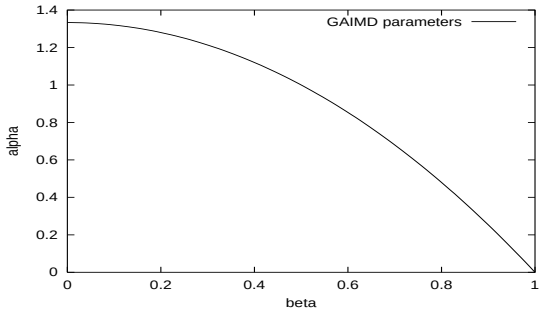

Fig. 1. GAIMD parameters

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{2}$$

Since a TCP-friendly (*TCP-compatible*) flow can be defined as a flow that, in steady-state, uses no more bandwidth than a conforming TCP running under comparable conditions, such a flow should not exceed the sending rate T, which is a function of the packet size s, round-trip time R, steady-state loss event rate p and the TCP retransmit timeout value t_{RTO} and is given in bytes/sec.

The most critical of these parameters is the "steady-state loss event rate"; any number of losses within a round-trip time are considered a single "loss event" in TFRC. The parameter is calculated using the "Average Loss Interval" method, which is a specially weighted moving average of the loss rate. The aggressiveness of TFRC can be tuned by varying the length of the loss interval, but generally, TFRC adapts to traffic fluctuations slowly and achieves a much smoother rate than RAP. More details can be found in [15].

– *TCP Emulation at Receivers (TEAR)* is a sophisticated mechanism where receivers perform calculations that senders would normally do based on the information in acknowledgments. TEAR assumes *rate independence*: the probability of experiencing a packet loss within a certain window of consecutively transmitted packets should not depend on their transmission rate. It achieves a smooth rate because it seldom sends rate updates to the sender; in simulations, the rate of TEAR appeared to be even smoother than the rate of TFRC [6].

There are several other notable TCP-friendly Congestion Control mechanisms — examples include *LDA+*, which controls the sender rate via RTCP packets and uses the "Packet Pair" network measurement method to increase the rate proportional to the bottleneck bandwidth; it is efficient in terms of network utilization and fairness towards TCP [17]. Another one is *Binomial Congestion Control* [2], which is a nonlinear generalization of AIMD and can

be seen as an extension of GAIMD. The interested reader is pointed to [8] for a much more detailed overview.

3 Test Environment

AVCS, the "Adaptive Video Communication System", is a video communication software that uses the TCP-friendly congestion control mechanisms RAP (fully functional) or TFRC (currently only partially working), depending on the choice of the user, to transmit adaptive video data. Since the goal of AVCS is not to provide the user with good perceptual quality but with obvious and immediate feedback, the data stream is adapted to the available bandwidth by changing the resolution of the video image with the finest possible granularity.

We deliberately refrained from using a more realistic video encoding to adapt the video quality to network bandwidth: if we had chosen, say, MPEG, we would have faced an immense choice of possibilities — e.g., changing the frequency of I-, P- and B-frames or altering the compression. Each method has its own advantages and disadvantages and brings about new issues; even if we had managed to choose the most "realistic" codec of the day, the longevity of our results would have been seriously impacted. The goal was therefore not to choose a "good" or "realistic" encoding scheme. Rather, we assumed that bandwidth fluctuations generally affect the perceptual quality of individual video frames and tried to make such quality changes as evident as possible.

Eventually, users should decide why they prefer one mechanism over another — here, the idea is that any upper layer adaptation mechanism will eventually cause the quality to degrade in response to reduced bandwidth in one way or another. Monitoring the quality perceived by users could therefore lead to general statements like "quickly fluctuating quality is worse than constantly poor quality", which could be used as guidance when choosing a congestion control mechanism.

The five node testbed depicted in figure 3 was constructed as a usage scenario for AVCS; we refrained from using tools such as dummynet [4] or NISTnet [11] for the sake of simplicity and controllability. While the tool can act as both a sender and receiver at the same time, we have decided to use it in a unidirectional manner for the time being. Since it is not intended to study the LAN behavior of the protocols but the behavior in the face of a single bottleneck, it is important to avoid Ethernet collisions.[3] The network topology being a 10 Mbit/s Ethernet bus, the only way to generate background traffic without having it collide with AVCS payload is to generate it directly at the router, where both flows enter the queue of the outgoing network interface.

Background traffic was generated in the form of UDP packets using the "MGEN" traffic generator [10] and resembled wide-area network traffic aggre-

[3] According to [5], such a scenario can be used to emulate WAN behavior provided that self-similar background traffic is used; we neglect Internet path changes because they generally occur on timescales that are several orders of magnitude greater than a RTT and therefore irrelevant for our study [19].

Fig. 2. Screenshot of the Adaptive Video Communication System (AVCS)

gates on a single link[4] — the bandwidth follows values that were generated with the "fft_fgn" self similar time series generator described in [12] and scaled to represent varying amounts of background traffic (numbers 1-5 in fig. 4).

We believe that one can safely assume that it is very improbable to achieve TCP-friendliness with varying environment conditions by luck. Based on this assumption, the fact that the total throughput of 10 minute AVCS tests with RAP and five different degrees of background load equals FTP downloads under similar conditions clearly shows that our RAP implementation (a Windows NT port of the original *ns* code) is at least roughly TCP-friendly. The result is depicted in fig. 4[5]; here, "RAP frame" — special version of RAP that would only adapt the rate at the beginning of a new frame — is obviously not TCP-friendly.

AVCS was developed as part of a diploma thesis [18] under the primary author's supervision; it provides numerous additional features such as MJPEG (per-frame JPEG) compression, precise timing and a large number of possible parameter settings. AVCS is open source software and was designed using a layered architecture that facilitates enhancement and extension (e.g., it would be straightforward to replace the video functionality in AVCS with audio). It is available from [1].

[4] While this traffic may resemble the kind of traffic encountered at an Internet backbone router, it might not be representative for an end-to-end Internet connection. However, since such a connection looks different depending on when it is used and where it is used from, we decided to accept this simplification.

[5] The throughput of RAP with background traffic levels 1 to 5 was 107%, 105%, 100%, 100% and 103% of FTP throughput, respectively; these deviations were probably caused by effects beyond our control such as Windows operating system timing.

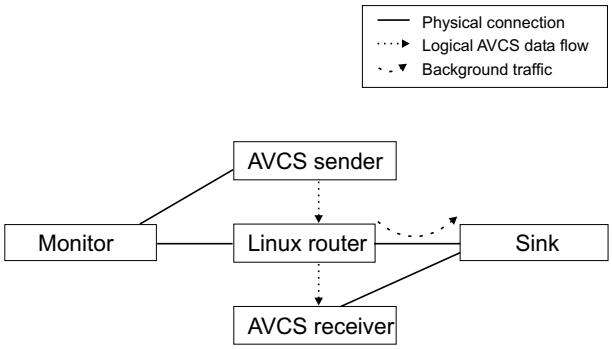

Fig. 3. The testbed

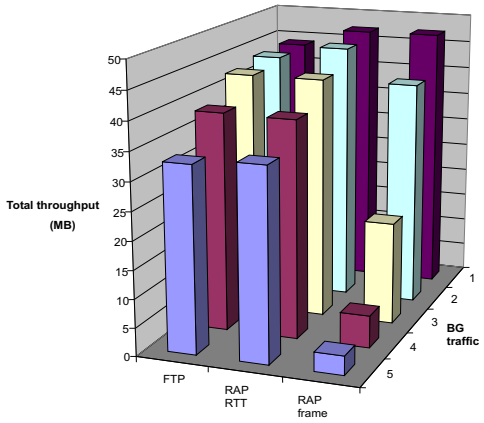

Fig. 4. Total throughput of mechanisms in AVCS

4 Test setup

We decided to choose RAP (with standard per-RTT adaptation) and tune its increase and decrease parameters based on equation 1 in three ways:

1. *Normal:* $\alpha = 1.0$, $\beta = 0.5$
2. *Smooth:* $\alpha = 0.31$, $\beta = 0.875$
3. *Reactive:* $\alpha = 1.3125$, $\beta = 0.125$

Variant 1 corresponds with standard TCP behavior, variant 2 is less aggressive and decreases the rate by a smaller value in response to packet loss, and variant 3 is more aggressive but has a drastic reaction to congestion. Background traffic was moderate (scale number 3 in fig. 4). In order to obtain reproducible results, the video was not taken from a live webcam but from a digital video in our library entitled "Ergonomie — der menschliche Aspekt der EDV" (Ergonomics

— the human aspect of IT). From this video, we chose three different sequences with a duration 39, 21 and 36 seconds, respectively. These sequences were:

- An introduction, consisting of seemingly random pictures of computer equipment (filmed in a way that makes it somewhat hard to identify at first sight)
- A survey, consisting of portrait shots of people answering questions
- The credits sequence, which looks like standard movie credits scrolling across the screen

We generated movie files from these sequences with each of the GAIMD parameter variants, yielding a total of 9 output files (and a total duration of 14.4 minutes). These files were later shown to the test persons. The somewhat short duration of the individual sequences was necessary in order to keep the test persons motivated — we made an effort to keep the total test duration in the range of approximately 20 minutes. The following facts were explained to the test persons beforehand:

- *There is a problem for streaming media with small and fluctuating bandwidth in the Internet; a compromise (reduced quality) is necessary in order to keep the transmission fluent.*
- *This test is not about different video rendering mechanisms — it is about data transmission mechanisms and their (unavoidable) impact on the quality of a video stream.*
- *Since we refrained from using compression in order to visualize rate fluctuations as clearly as possible, the quality of these videos is generally worse than what you normally see with commercial streaming video applications. Please do not rate the absolute quality in comparison with such applications; rate the relative quality of the mechanisms in comparison with each other instead.*

They had to rate each of the sequences as "good", "mediocre" or "poor" and provide an absolute qualitative ranking of the 9 sequences. The goal of asking these questions was to find differences in the users' perception and evaluation of the increase/decrease parameter variations — the final question about the absolute ranking should help to verify the results of the relative "good" to "poor" rankings. Additionally, we wanted to figure out whether the users' answers remain similar with different video material.

The questionnaire and the video files are available from [1]; we will try hard to ensure their accessibility as long as possible.

5 Results

We conducted our study with 34 computer science students at the University of Innsbruck; the typical test person was aged 20 and male. To evaluate the questionnaire results, "good", "mediocre" and "poor" quality were mapped to a range of (1..3). The results are listed in table 1 and the average is depicted in fig. 5.

Sequence	Average	Median	Stddev
Intro, standard	2.74	3	0.57
Intro, smooth	2.12	2	0.73
Intro, reactive	2.03	2	0.76
Survey, standard	2.71	3	0.68
Survey, smooth	1.91	2	0.62
Survey, reactive	2.38	2	0.6
Credits, standard	2.82	3	0.58
Credits, smooth	2.12	2	0.69
Credits, reactive	2.44	3	0.70

Table 1. User test results (relative ranking)

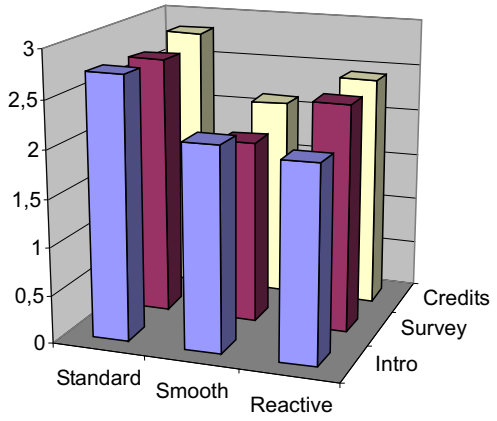

Fig. 5. User test results (average relative ranking)

From looking at the values, we can see two interesting facts: i) both the reactive and the smooth variants appeared to work better than the standard (TCP-like) congestion control variant, and ii) the smooth variant seems to be the "winner". However, the reactive variant appeared to yield better results for the introduction, which we believe to be due to the fact that the better (albeit fluctuating) quality of the individual frames made it easier for our test persons to identify the computing equipment in this sequence. Since the main purpose of the study was to figure out whether a smoother rate is generally advantageous in the absence of large buffers, we decided to test the thesis:

> Are the results obtained with a smooth rate significantly better than the other results?

by grouping the values from the absolute ranking (final question) pairwise with the mechanism parameter variation and applying a two-sided T-test. The average and standard deviation of the input data for the test are listed in table 2; the

Mechanism	Average	Stddev
Standard	5.4948	2.5704
Smooth	4.3232	2.5105
Reactive	5.4286	2.6124

Table 2. User test results (absolute ranking)

formula

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - \mu}{\sqrt{(\frac{1}{n_1} + \frac{1}{n_2})\frac{(n_1-1)\hat{\sigma}_1^2 + (n_2-1)\hat{\sigma}_2^2}{n_1+n_2-2}}} \tag{3}$$

was used [3], whereby s^2 is the estimator for $\hat{\sigma}^2$:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{4}$$

and $\mu = 0$ (the averages $\bar{x}_1$ and $\bar{x}_2$ are compared), $n_1 = n_2$ is the sample size (both 99 in our case) and $\bar{x}_1$ and $\bar{x}_2$ are the average samples 1 and 2, respectively.

The results of the test verify the thesis:

> The "smooth" variation is significantly (with a probability of approx. 95%) better evaluated than the other variations.

6 Conclusion

In this article, we described how we carried out a qualitative comparison of three different TCP-friendly congestion control schemes (based on a single mechanism with different parameter settings). The value of our results is clearly limited by the fact that we used only one mechanism, one application and one type of background traffic so far; also, our user group was relatively small and homogeneous. There are, of course, numerous other ways to perform such tests; some methods are described in [7].

Our basic goal was to find an answer to the question: "Is a smooth rate generally better for interactive video applications?", thereby providing some guidance for programmers; our answer is "yes". We believe that the growing transport layer diversity (in particular due to the DCCP protocol [16]) generates a demand for such guidance. This fact is substantiated by the ongoing work towards a DCCP user guide [9]. We regard our own work as being complementary to this DCCP effort. Eventually, we plan to produce several documents that should serve as an increasingly helpful aid for application programmers.

7 Acknowledgments

We would like to thank all the students who were involved in building the testbed and, in particular, Gerhard Stummer who implemented AVCS.

References

1. AVCS. Available from: http://www.welzl.at/research/projects/avcs/.
2. Deepak Bansal and Hari Balakrishnan. Binomial congestion control algorithms. In *IEEE INFOCOM 2001, Anchorage, Alaska*, April 2001.
3. Juergen Borz. *Statistik*. Springer, 1993.
4. Dummynet. Available from: http://info.iet.unipi.it/∼luigi/ip_dummynet/.
5. Faris Faris Ferit Yegenoglu and Osama Qadan. A model for representing wide area internet packet behaviour. In *IEEE IPCCC 2000*, February 2000.
6. Volkan Ozdemir Injong Rhee and Yung Yi. Tear: Tcp emulation at receivers – flow control for multimedia streaming. Technical Report Technical Report, Department of Computer Sciences, NCSU, 2000.
7. A. Watson J. Mullin, L. Smallwood and G. M. Wilson. New techniques for assessing audio and video quality in real-time interactive communication. In *IHM-HCI 2001, Lille, France*, September 2001.
8. Robert Denda Joerg Widmer and Martin Mauve. A survey on tcp-friendly congestion control. *IEEE Network Magazine, Special Issue "Control of Best Effort Traffic"*, 15(3), May 2001.
9. Damon Lanphear. Dccp user guide. Internet-draft draft-lanphear-dccp-user-guide-00.txt (work in progress), October 2002.
10. MGEN. Available from: http://manimac.itd.nrl.navy.mil/mgen/.
11. NISTNet. Available from: http://snad.ncsl.nist.gov/nistnet/.
12. Vern Paxson. Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *Computer Communication Review (CCR)*, 27(5):5–18, October 1997.
13. Mark Handley Reza Rejaie and Deborah Estrin. Quality adaptation for congestion controlled video playback over the internet. In *ACM SIGCOMM 1999, Cambridge, MA.*, September 1999.
14. Mark Handley Reza Rejaie and Deborah Estrin. Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *IEEE Infocom 1999, New York City, New York*, March 1999.
15. Jitendra Padhye Sally Floyd, Mark Handley and Joerg Widmer. Equation-based congestion control for unicast applications. In *ACM SIGCOMM 2000, Stockholm, Sweden*, August / September 2001.
16. Mark Handley Sally Floyd and Eddie Kohler. Datagram congestion control protocol (dccp). Internet-draft draft-ietf-dccp-spec-06.txt (work in progress), February 2004.
17. Dorgham Sisalem and Adam Wolisz. Lda+: A tcp-friendly adaptation scheme for multimedia communication. In *ICME 2000*, 2000.
18. Gerhard Stummer. *Adaptionsmechanismen verteilter Multimedia-Internetanwendungen*. Diploma thesis, Johannes Kepler University of Linz, Linz, Austria, 2001.
19. V. Paxson Y. Zhang and S. Shenker. The stationarity of internet path properties: Routing, loss, and throughput. Technical Report ACIRI Technical Report, icir — i.c.s.i. center for internet research, May 2000.
20. Y. Richard Yang and Simon S. Lam. General aimd congestion control. Technical Report Technical Report TR-2000-09, Department of Computer Sciences, The University of Texas at Austin, May 2000.
21. Min Sik Kim Yang Richard Yang and Simon S. Lam. Transient behaviors of tcp-friendly congestion control protocols. In *IEEE INFOCOM 2001, Anchorage, Alaska*, April 2001.