An Empirical Study of the Congestion Response of RealPlayer, Windows MediaPlayer and Quicktime

Sven Hessler
Institute of Computer Science
University of Innsbruck
6020 Innsbruck, Austria
sven.hessler@uibk.ac.at

Michael Welzl
Institute of Computer Science
University of Innsbruck
6020 Innsbruck, Austria
michael.welzl@uibk.ac.at

Abstract

Properly reacting to congestion is known to be a requirement for maintaining the stability of the Internet; in addition, UDP based applications should do so for their own sake if they are delay- or loss-sensitive. We measured the responsiveness of the three popular streaming media applications RealPlayer, Windows MediaPlayer and Quicktime with a varying amount of cross traffic and present a comparison of the results.

1. Introduction

Streaming media applications that do not react to congestion endanger the stability of the Internet [2]; a single unresponsive UDP data flow can significantly degrade the throughput of thousands of responsive flows. There is a wealth of possibilities to realize Internet-compatible (i.e. TCP-friendly[1]) congestion control, as has been illustrated by the large number of related research efforts, e.g. RAP [11], TFRC [3], LDA+ [13], TEAR [12] and Binomial Congestion Control [1]. Additionally a large amount of research has been carried out on application layer issues, e.g. how to map a video stream onto an underlying congestion control mechanism [10], or how to separate video data into distinct layers with varying degrees of importance [6]. An overview of *adaptive applications* which utilize such methods can be found in [14].

With the advent of the Datagram Congestion Control Protocol (DCCP) [5], which drastically facilitates embedding a congestion control mechanism within one's streaming media application, there appears to be no excuse for developers not to do so. An in-depth survey of the responsiveness of streaming media is given in [7, 8, 9, 15]. While the authors of [8, 15] investigate the performance of RealPlayer, and Windows MediaPlayer [9] in an isolated environment, only Li et al. [7] compare two streaming applications (MediaPlayer vs. RealPlayer). Though, all the aforementioned research lacks the answer to the practical question a content provider might ask: "What program (RealPlayer, MediaPlayer, Quicktime) is the best if congestion occurs?". As a sensible extension to the research already done and to provide content providers with applicable information to make the right decision of what to use, we analyzed the behavior of three major streaming video applications (RealPlayer, Windows MediaPlayer, and Quicktime) under similar network conditions and present a comparison of the results in this document. The organization of this paper is as follows. Section 2 describes our test environment and discusses how the cross traffic and the payload is generated. In section 3 we present the outcome of our tests. Section 4 summarizes the paper and discusses further directions of research.

2. Test environment and setup

The intention of this paper is to analyze the responsiveness of three popular streaming video application within an isolated test environment. We were not particularly interested in how the programs behave in a realistic environment; rather we wanted to tease out their behavior if congestion occurs. The testbed (fig. 1) is simple and quite similar to that presented in [9]. It interconnects five machines using Fast-Ethernet links ($100\,\mathrm{Mbps}$).

Two PCs running Windows 2000 act as streaming server and -client, respectively. A PC running Linux (RedHat 8.0, Kernel v2.4.18) with two network interfaces is used as a router. At one of the router's legs (the one which is connected to the sink) the available traffic rate was limited to $1\,\mathrm{Mbps}$ by using the `tc` (Traffic Control) Linux command and Class Based Queuing with only one class. We do not

[1] A stream is regarded as TCP-friendly if its bit-rate does not exceed the maximum bit-rate of a TCP connection under the same network conditions [9].

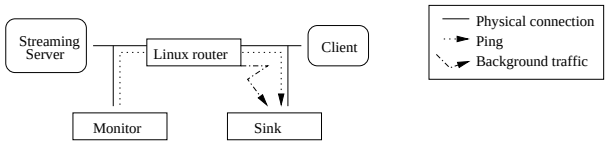

Figure 1. The testbed

use Token Buckets due to their influence on traffic characteristics, e.g. adverse effect on TCP [4]. The monitor and the sink, both running Linux (RedHat 8.0, Kernel v2.4.18) and using tcpdump[2], are used to measure the state of congestion by sending a ping from the monitor to the sink. Furthermore they are used to monitor the sender rate and the current throughput after congestion, respectively. Loss is calculated as the difference between the bytes sent (logged by the monitor) and the throughput (measured by the sink).

2.1. Cross traffic generation

Cross traffic was generated as a Constant Bit Rate (UDP) data flow of $1000\,\mathrm{byte}$ packets using the `mgen` traffic generator[3]. It was sent from the router to the sink, which means that it could not cause collisions but only lead to congestion in the queue of the router's outgoing network interface. We generated four classes of cross traffic:

little	a minute of $60\,\mathrm{packets\,per\,second}$ (pps)
medium	a minute of $90\,\mathrm{pps}$
heavy	a minute of $150\,\mathrm{pps}$
burst	$200\,\mathrm{pps}$ for a duration of $720\,\mathrm{ms}$

We customized these rates according to all three investigated applications and the network setup so that little cross traffic slightly impacts the application's behavior. Additionally, initial `mgen` packets were used to synchronize the test machines.

2.2. Generation of the payload

We used the same video clip (a trailer of "Matrix Reloaded") for all tests in order to obtain comparable results. The original movie was encoded with DivX 5.02 using a bit rate of $863\,\mathrm{kbps}$. We chopped the video to $2\,\mathrm{minutes}$ and saved it as an uncompressed AVI file. RealPlayer and Windows MediaPlayer required the stream to be saved in different scaled-down versions in order to be able to react to bandwidth fluctuations. We tried to keep all settings as close to their default values as possible and selected unaltered original templates for those video streams (variants with 43, 58, 282 and $548\,\mathrm{kbps}$ for Windows MediaPlayer and the templates "$0-20\,\mathrm{kbps}$", "$20-34\,\mathrm{kbps}$", "$34-225\,\mathrm{kbps}$", "$225-450\,\mathrm{kbps}$" and "above $450\,\mathrm{kbps}$" for RealPlayer). In the case of Quicktime, we activated the options "prepare for Internet streaming", "stream with control track" and "optimize control track for server". All servers and players were set to use UDP only.

Encoder	Server	Player
RealPlayer		
Helix Producer (v9.0.0.972)	Helix Universal Server Basic (v9.0.3)	RealPlayer (v10 Beta)
Windows MediaPlayer		
Windows Media Encoder (v9.0)	Helix Universal Server Basic	Windows MediaPlayer (v9.0)
Quicktime		
Quicktime Player Pro (vD-6.5)	Darwin Streaming Server (v5.0.1.1)	Quicktime Player (vDE-6.5)

Table 1. Used streaming software

3. Results

We carried out a total of 12 measurements as the product of a single video stream, three streaming applications, and four classes of cross traffic. Additionally for each type of cross traffic an FTP download was performed to figure out whether the tested applications were TCP-friendly or not. The transfer behavior of the TCP stream (FTP download) at each level of cross traffic is used as reference curve when compared with the streaming applications. The evaluation of the acquired results of each trial yielded three types of diagrams.

1. *Sender rate* — The difference between the sender rate (measured at the monitor) and throughput (measured at the sink) could roughly be interpreted as loss. It is a good sign the difference is small, that means both lines overlap.

2. *Packet transmission rate* — Since the packet size varied, the number of transmitted packets per second may not be the same as the data rate in bit per second.

3. *State of congestion* — The measurement of the RTT (by sending a ping from the monitor to the sink) illustrates the state of congestion. Generally spoken, the higher the RTT, the higher the congestion.

[2] `http://www.tcpdump.org`
[3] `http://mgen.pf.itd.nrl.navy.mil/mgen.html`

The sender rate and throughput of RealPlayer (fig. 3), Windows MediaPlayer (fig. 8), and Quicktime player (fig. 10) in the presence of medium cross traffic are shown and compared to that of TCP (fig. 2). The fact that the gap between the graph of the sender rate and the throughput of TCP overlap almost perfectly indicates that there was no loss. On the other hand, the streaming applications experienced loss with different magnitudes, whereby the number of lost packets in the RealPlayer and the Windows MediaPlayer setup is much higher than in the Quicktime test (c.f. tab. 2). In what follows, we present an in-depth analysis of the behavior of each streaming application. The cumulative results for each setting are listed in tab. 2.

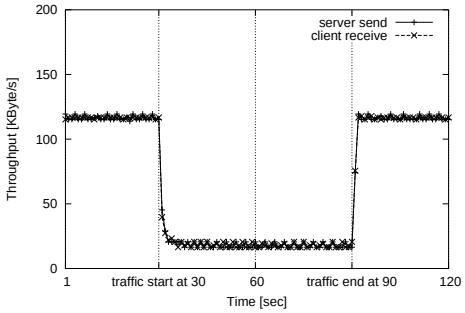

Figure 2. TCP (FTP): sender rate and throughput with medium cross traffic

3.1. RealPlayer

Without cross traffic, RealPlayer sent at an average rate of $90\,\mathrm{kbps}$ and reduced its rate after approximately $45\,\mathrm{s}$. This might be related to the separate buffering and playout phases of the player that were discovered in [9]. When cross traffic came into play and congestion occurred, RealPlayer reacted strangely: first, it increased the rate but reduced it shortly after (fig. 3). This short burst led to congestion (fig. 4) and loss, as indicated by the diverging lines. At one point, the two lines cross in fig. 3 — this is due to traffic that was stored in the queue but eventually reached the sink. In general, during congestion periods, there is a visible distance between the two lines. It is particularly interesting that, after the cross traffic stopped, it took approximately $20\,\mathrm{s}$ until the player began to increase its rate again, and the final rate at the end of the measurement was nowhere near its original rate in the case with little cross traffic (fig. 5). With medium cross traffic, the rate dropped to zero and was not increased again until the end of our measurement. This effect coincides with the packet size. After approximately $40\,\mathrm{s}$, the player significantly reduced it from $1200\,\mathrm{byte}$ down to less than $500\,\mathrm{byte}$; we believe that this is due to congestion. This is underlined by the fact that packet size reduction occurred with any cross traffic. The higher the cross traffic, the higher the drop if congestion occurred and the lower the average packet size thereafter. Burst traffic did not significantly affect the RealPlayer's behavior.

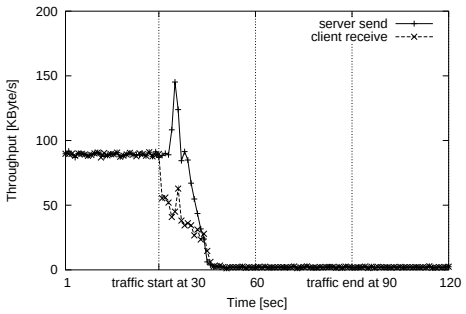

Figure 3. RealPlayer: sender rate and throughput with medium cross traffic

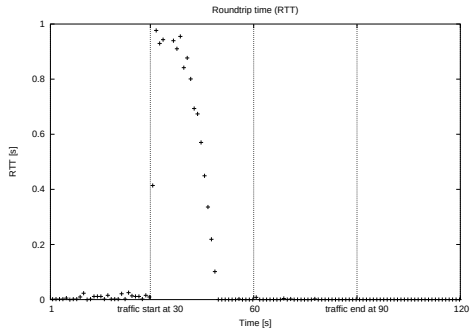

Figure 4. RealPlayer: RTT measurement with medium cross traffic

3.2. Windows MediaPlayer

In contrast to RealPlayer, the sending rate of Windows MediaPlayer fluctuated even without cross traffic. The variation of the average packet size was similar to RealPlayer but it did not show any response to congestion at any level of cross traffic; neither did the sender rate (fig. 7). Figure 8 illustrates a general problem of Windows MediaPlayer that was observed in all our tests with cross traffic: its rate was somewhat steady but constantly a bit too high. This led to congestion (fig. 9) and loss. After congestion, it increased the sending rate to a level that it never had before — possibly, as a compensation for loss and to fill the

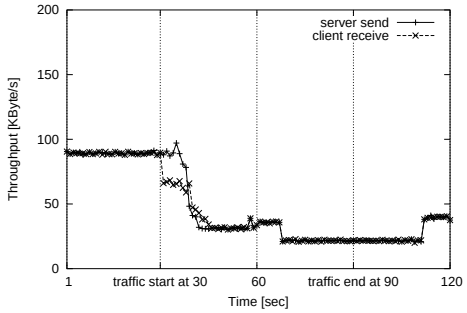

Figure 5. RealPlayer: sender rate and throughput with little cross traffic

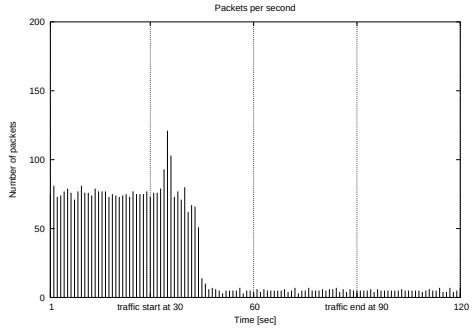

Figure 6. RealPlayer: packets sent per second with medium cross traffic

buffer. In the scenario with heavy cross traffic, the rate suddenly dropped to almost zero after approximately $80\,\mathrm{s}$, and the player stopped to send and gave an error message after about $110\,\mathrm{s}$.

3.3. Quicktime

Quicktime showed a strongly fluctuating sending rate even without cross traffic. The average packet size varied slightly stronger than that of RealPlayer or Windows MediaPlayer. It did not show any response to congestion at any level of cross traffic. Contrary to the average packet size the sending rate was adjusted in response to congestion by Quicktime (fig. 10). Quicktime reduced the sending rate significantly when congestion occurred and increased the rate quickly after congestion was over. Also, the two lines strongly overlap, which means that there was hardly any loss. This behavior is more pronounced in fig. 11, indicating that Quicktime responds well to congestion in the network.

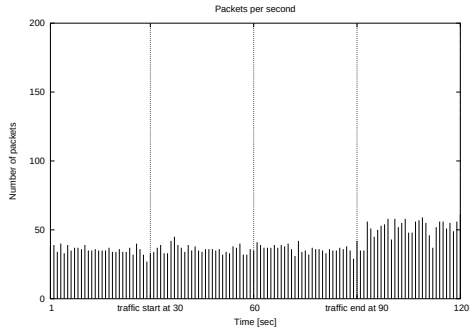

Figure 7. Windows MediaPlayer: packets sent per second with medium cross traffic

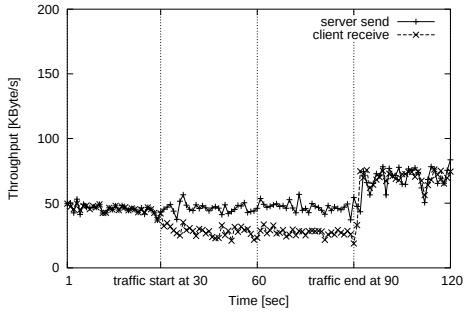

Figure 8. Windows MediaPlayer: sender rate and throughput with medium cross traffic

4. Summary and future work

In this paper we investigated the behavior of three major streaming applications — RealPlayer, Windows MediaPlayer, and Quicktime. We chose a straightforward test environment as a basis for comparison. It could be characterized by similar network conditions in all tests and particularly no traffic artifact caused by uncontrollable influences from the network. Thus, the behavior of the streaming application could easily be compared. Despite this simple setup substantial results were acquired, which we regard as useful for content providers as practical guidance.

All of the investigated applications were TCP-friendly according to the definition in [3]. While it did not have (or strive to achieve) the greatest throughput, no streaming media application lost less than Quicktime. We believe the loss ratio to be the best indicator of how closely a mechanism is able to follow the available bandwidth; from this point of view RealPlayer and Windows MediaPlayer must be judged as poorly performing — whereby the bad results obtained from MediaPlayer could be misleading if they resulted from the incompatibility of the MediaPlayer client

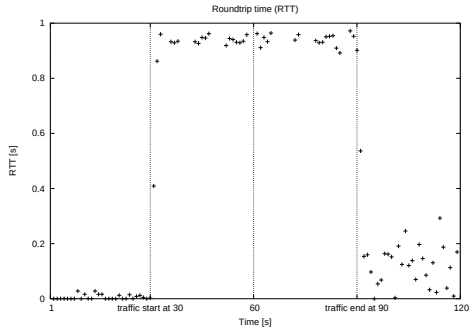

Figure 9. Windows MediaPlayer: RTT measurement with medium cross traffic

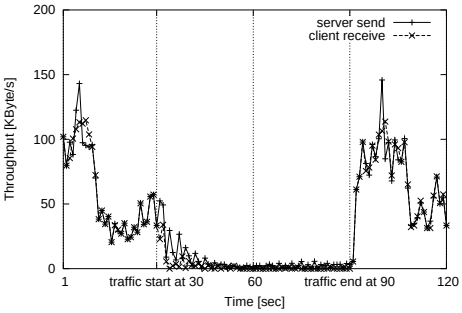

Figure 11. Quicktime: sender rate and throughput with heavy cross traffic

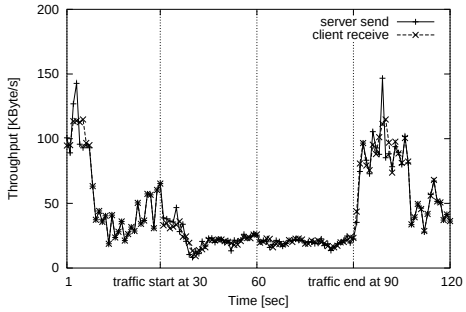

Figure 10. Quicktime: sender rate and throughput with medium cross traffic

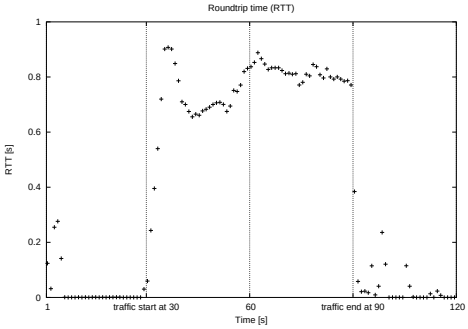

Figure 12. Quicktime: RTT measurement with medium cross traffic

and the freely available Helix Server. Our data concur with our subjective impression: the video stream looked best in all cases with Quicktime.

After evaluating these major streaming applications within our straightforward test environment, a couple of questions remain and need further investigation. Future work will cope with the question of how multiple streaming servers affect the results acquired so far, if the MediaPlayer performs better with a more compatible server, and an extension of the test environment.

5 Acknowledgments

This work was funded by the Austrian Science Fund (FWF). We thank Muhlis Akdag for his substantial contributions to this work.

References

[1] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *Proceedings of IEEE INFOCOM*, 2001.

[2] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, August 1999.

[3] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM*, 2000.

[4] G. Huston. Next steps for the ip qos architecture, November 2000. RFC 2990.

[5] E. Kohler, M. Handley, and S. Floyd. Datagram congestion control protocol (dccp), March 2005. Internet-draft draft-ietf-dccp-spec-11.txt.

[6] A. Legout and E. W. Biersack. Plm: Fast convergence for cumulative layered multicast transmission schemes. In *Proceedings of ACM SIGMETRICS*, 2000.

[7] M. Li, M. Claypool, and B. Kinicki. Mediaplayer versus realplayer - a comparison of network turbulence, November 2000. ACM SIGCOMM Internet Measurement Workshop.

[8] A. Mena and J. Heidemann. An empirical study of real audio traffi c. In *Proceedings of IEEE INFOCOM*, pages 101–110, 2000.

[9] J. Nichols, M. Claypool, R. Kinicki, and M. Li. Measurements of the congestion responsiveness of windows streaming media. In *Proceedings of the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2004.

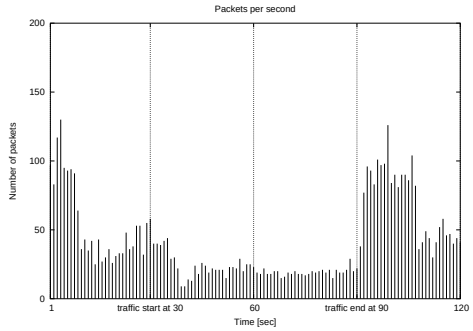

Figure 13. Quicktime: packets sent per second with medium cross traffic

[10] R. Rejaie, M. Handley, and D. Estrin. Quality adaptation for congestion controlled video playback over the internet. In *Proceedings of ACM SIGCOMM*, 1999.

[11] R. Rejaie, M. Handley, and D. Estrin. Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Proceedings of IEEE INFOCOM*, 1999.

[12] I. Rhee, V. Ozdemir, and Y. Yi. Tcp emulation at receivers - fbw control for multimedia streaming. Technical report, Department of Computer Science, NCSU, 2000. http://www.csc.ncsu.edu/faculty/rhee/export/tear_page/.

[13] D. Sisalem and A. Wolisz. Lda+: A tcp-friendly adaptation scheme for multimedia communication. In *Proceedings of ICME*, 2000.

[14] X. Wang and H. Schulzrinne. Comparison of adaptive internet multimedia applications. *IEICE Transactions on Communications*, E82-B(6):806–818, June 1999.

[15] Y. Wang, M. Claypool, and Z. Zuo. An empirical study of realvideo performance across the internet, November 2001. ACM SIGCOMM Internet Measurement Workshop.

Scenario with … cross traffic	sent (kB)	received (kB)	lost (kB)	Loss (%)
FTP Download				
no	13977.9	13977.9	0.0	0.00
little	10433.0	10433.0	0.0	0.00
medium	8069.7	8069.7	0.0	0.00
heavy	6314.6	6303.7	10.8	0.17
burst	13833.8	13833.8	0.0	0.00
RealPlayer				
no	8301.8	8301.8	0.0	0.00
little	5669.3	5546.4	122.9	2.17
medium	3965.2	3414.7	550.4	13.88
heavy	4291.8	3160.1	1131.7	26.37
burst	8372.1	8372.1	0.0	0.00
Windows MediaPlayer				
no	5450.9	5450.9	0.0	0.00
little	6513.2	6032.9	480.3	7.37
medium	6225.2	5110.0	1115.1	17.91
heavy	3722.1	1963.1	1758.9	47.26
burst	5450.7	5448.7	1.9	0.04
Quicktime Player				
no	4859.7	4859.7	0.0	0.00
little	5065.4	5065.4	0.0	0.00
medium	5083.3	5072.1	11.1	0.22
heavy	4145.8	3907.1	238.7	5.76
burst	4637.5	4637.5	0.0	0.00

Table 2. Cumulative results