An Experimental Investigation of the End-to-End QoS of the Apple Darwin Streaming Server

Luca De Cicco, Saverio Mascolo and Vittorio Palmisano
{ldecicco, mascolo, vpalmisano}@poliba.it
Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via Re David 200, Italy

***Abstract*— Video content distribution over the traditional best-effort, store-and-forward Internet Protocol is of ever increasing importance due to the great success of new web services such as personal video broadcast or television over IP (IPTV). In this paper we investigate the end-to-end quality of service (QoS) that is provided by the Apple Darwin Streaming Server and the Quick-Time client player in the presence of time-varying available bandwidth and multiple concurrent streaming sessions. The considered end-to-end QoS parameters are loss rates and friendliness experienced in the presence of variable available bandwidth and when multiple QuickTime streaming sessions and/or TCP sessions compete in order to obtain a bandwidth share.**

We found that the Darwin Streaming Server implements a TCP-like congestion control that is more aggressive than TCP; in particular, when more QuickTime flows share the same link with TCP flows, QuickTime gets more bandwidth than TCP. Moreover, in this case QuickTime flows exhibit a higher loss rate than TCP ones.

***Index Terms*— End-to-End QoS, Multimedia Congestion Control, Reliable UDP, Apple Darwin Streaming Server, QuickTime Player**

I. Introduction

Audio/Video content distribution is nowadays a potential killer application for the Internet as it is proved by the great success of YouTube [1] and by the introduction of new applications such as Joost and Babelgum, which aim at providing television distribution over IP. The most part of Internet traffic is still delivered using the TCP transport protocol, which has been the key factor of Internet stability so far. This is the reason for which many Web sites (such as YouTube) that host small length and low resolution videos use only pseudo-streaming technologies that are based on the simple TCP download. In this way the generated traffic is not harmful for the stability of the Internet because the TCP transport protocol implements an effective congestion control algorithm [2]. However, it is not yet clear if the perceived quality is satisfactory for the user. In fact the source of the great success obtained by YouTube is very much likely to be due to the richness of contents and its large user base rather than to the quality of the video delivering. In fact, the TCP window-based congestion control guarantees congestion avoidance by using the additive increase/multiplicative decrease paradigm [2] and reliable delivery of the content through packet retransmissions but not content delivery within delay constraints.

This work was partially supported by the MIUR-PRIN project no. 2005093971 "FAMOUS Fluid Analytical Models Of aUtonomic Systems"

On the other hand, multimedia streaming services can tolerate some low packet loss percentage but require more tight quality of service (QoS) requirements in terms of end-to-end delays and jitter. For this reason the UDP protocol is the preferred transport protocol for multimedia streams, because, as matter of fact, it is a simple packet multiplexer/demultiplexer, where the packet sending rate can be managed at the application level. However, many multimedia applications which use UDP do not implement effective congestion control mechanisms, thus possibly leading to a network *congestion collapse* [3] due to the presence of many unresponsive flows on the same bottleneck link. This circumstance can cause a persistent loss rate, which is an important factor that affects the perceived quality [4],[5].

Several efforts have been made to design multimedia congestion control protocols that are TCP friendly, where friendliness here means that the multimedia flows will share the network bandwidth with TCP flows fairly. The TCP Friendly Rate Control (TFRC) protocol is currently being discussed within the IETF as a possible congestion control algorithm for the transport of multimedia flows [6]. An interesting solution is the Reliable UDP proposed by Apple, which is a TCP-like congestion control protocol that aims at providing a set of QoS enhancements for RTP multimedia flows [7] (see Sec. III for more details)

The Darwin Streaming Server (DSS) is the open source version of the commercial Apple's QuickTime Streaming Server (QTSS) that allows the distribution of streamed multimedia contents over the Internet. The protocols employed by DSS are the standard RTP and RTCP [8]. DSS is based on the same code base of QTSS, but its source code is freely distributed under the Apple Public Source License. Both DSS and the official commercial QuickTime Player (QTP) [9] implement the Reliable RTP congestion control. DSS uses well-known standards (such as RTP, RTCP, SDP and HTTP) for content distribution. Thus, every multimedia player that supports RTP can be used as client.

In this paper we have used the official DSS and QTP for investigating the effectiveness of Reliable UDP congestion control algorithm in the presence of changing available bandwidth and/or packet losses. Moreover, we have employed VideoLAN Player [10] which does not implement Reliable UDP in order to perform a comparative test with QTP. The goal of these investigations is to evaluate how the congestion control algorithm implemented by Reliable UDP allows the sending rates be managed in order to match the available bandwidth when multiple streaming sessions and/or TCP connections share the same link, thus revealing intra-protocol and inter-protocol fairness behavior.

The rest of the paper is organized as follows: Section II presents the previous work on streaming server performance evaluations; Section III provides a brief description of the Reliable UDP protocol; Section IV describes the considered experimental testbed and the scenarios; Section V reports the experimental results we have obtained and finally Section VI draws the conclusions.

II. Related Work

Many recent investigations have focused on multimedia streaming client/server applications. In [11] an investigation on the Internet streaming quality and efficiency is performed by collecting connection data from thousands of broadband home users accessing both on-demand and live streaming media. Authors have found that input rate adaptation, which is implemented in media authoring, is poorly utilized, particularly when a pre-buffering phase is used. The pre-buffering phase (also called *Fast Streaming*) is widely used and much quality degradation has found to be caused by re-buffering events.

Authors of [12] present an evaluation of RealVideo streaming over UDP and over TCP. It has been found that RealVideo over UDP does not respond to Internet congestion by adapting the sending and/or the encoding rate. In particular, under very constrained bandwidth conditions RealVideo UDP streams does not share the bandwidth fairly with concurrent TCP connections. Moreover, authors report that only the 35% of RealServer implement some form of encoding scalability (called *Media Scaling*), and less then the 50% of the clips were using more then 4 encoding levels so that they can only adapt to the available bandwidth coarsely.

In [13] an investigation of the Windows Streaming Media (WSM) is performed in order to analize content multiple encoding. They found that, if the network capacity is lower than the minimum available encoding level, WSM produces high packet loss rates exhibiting unfairness with concurrent TCP flows.

In [14] a comparative analysis of RealPlayer, Windows Media Player and Quicktime is performed in an emulated environment. They used a UDP cross traffic generator concurrent with each multimedia stream flow in order to emulate a network congestion condition. Authors have found that Quicktime provided the lowest packet loss rate among the other applications, thus indicating that DSS performs an effective congestion control algorithm as compared to the other media streaming solutions.

In this work we aim at performing an extensive investigation of DSS by testing it on different scenarios in order to:

- evaluate how Reliable UDP reacts to congestion episodes
- estimate the friendliness between more concurrent QuickTime multimedia flows and between QuickTime flows and TCP flows

III. Apple's Reliable UDP

Reliable UDP is a set of extensions to the RTP protocol designed in order to provide retransmission and congestion control mechanism to the unreliable UDP protocol. These extensions allow multimedia streams to behave like TCP flows, while providing soft real-time features. The Apple's version of Reliable UDP implements a congestion control based on the Additive Increase/Multiplicative Decrease approach: the sender maintains a *congestion window* (CWND) such as the one used by the TCP congestion control; during the *slow-start* phase, for each ACKed packet, CWND increases by 1 segment, whereas during the *congestion avoidance* phase CWND increases by 1 segment every *round trip time* (RTT). When a timeout expires (which in the DSS implementation is always equal to 250 ms) or 3 duplicate ACKs (3DUPACKs) are received the slow-start threshold is set to 3/4 of CWND and CWND is halved. This behaviour makes Reliable UDP more aggressive than TCP when a loss event is detected, because when 3DUPACKs are received TCP halves the slow-start threshold and enters the congestion avoidance phase, whereas Reliable UDP enters the slow-start phase and then the congestion avoidance phase.

The results obained by our evaluation confirm this behaviour and we show that Reliable UDP tends to use more bandwidth with respect to TCP concurrent flows.

IV. Experimental testbed

The testbed we have set up is made of a Linux machine, hosting the DSS, and a Windows machine where the QuickTime and VideoLAN players have been installed. The Linux machine has beed equipped with the tool `ipqshaper`) that is able to set link capacity, delays, packets loss rates. `ipqshaper` uses the Application Programming Interface (API) provided by `Netfilter` [15] in order to redirect the packets sent or generated by the applications to a user-space drop-tail queue, where traffic shaping and measurement are performed.

As it is shown in Figure 1, the packets generated by DSS are redirected to `ipqshaper`'s queue, where traffic shaping is performed. The egress packets from DSS are sent to the QuickTime/VideoLAN players (P_1,..., P_n) that are installed on the Windows machine. `iperf` (T_1,..., T_n) has been installed in order to generate TCP concurrent flows. The queue size has been set equal to $10\,KB$, and a $10\,ms$ delay was applied to all outgoing packets.

It should be noticed that the experimental testbed described above is strictly equivalent to the one that could be obtained using a Dummynet-like router [16], the only difference being that in our case we are able to use only two hosts instead of three.

The video flows have been generated by using the benchmark video sequence *Foreman*, which has been encoded in MPEG4 format, resulting in an average bitrate of $242\,kb/s$. We used the commercial version of QuickTime Player in order to produce the correct *hinted* .mov files, required by the DSS for the streaming.

The experimental scenarios we have tested are made of:

1) A single QuickTime flow with a super imposed constant bandwidth limitation;

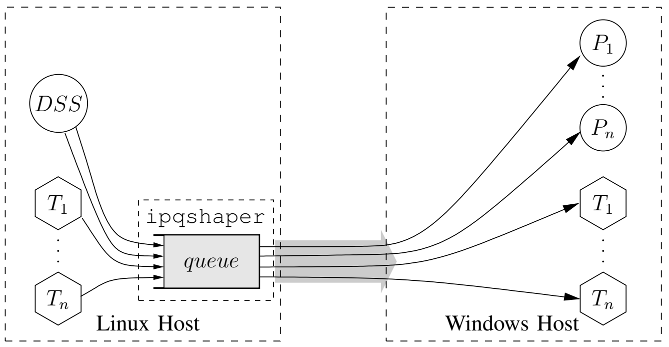


Fig. 1. Experimental testbed

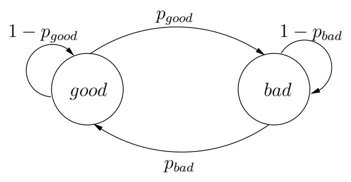


Fig. 2. Gilbert loss model

2) Multiple QuickTime and TCP concurrent flows with a constant bandwidth limitation;
3) Single and Multiple QuickTime and TCP flows with a variable bandwidth limitation;
4) Single and Multiple Quicktime and TCP flows with a super imposed loss rate.

The scenario (4) has been realised by using the Gilbert model in order to emulate loss events that normally affect noisy channels (e.g. IEEE 802.11a/b/g connections). This model uses a two-states Markov chain: a *good* and a *bad* state. When the process is into the *good* state, no loss events are generated. When in the *bad* state, the arriving packets are dropped with probability 1. Figure 2 shows the transition probabilities. It can be proved that the expected values are: $l_{good} = \frac{1}{1-p_{good}}$ and $l_{bad} = \frac{1}{1-p_{bad}}$. The values $l_{good} = 11.63$ and $l_{bad} = 1.78$ used here are the ones reported in [17] for IEEE 802.11 connections.

Goodput, throughput and loss rate shown in test results are defined as follows:

$$\begin{aligned} \text{goodput} &= \frac{\Delta sent - \Delta loss}{\Delta T} \\ \text{throughput} &= \frac{\Delta sent}{\Delta T} \\ \text{loss rate} &= \frac{\Delta loss}{\Delta T} \end{aligned}$$

where $\Delta sent$ is the number of bits sent in the period ΔT, $\Delta loss$ is the number of bits lost in the same period. We have considered $\Delta T = 0.5\,\text{s}$ in our measurements.

V. Results

In this section we present the test results we have obtained in our experimental investigation of DSS.

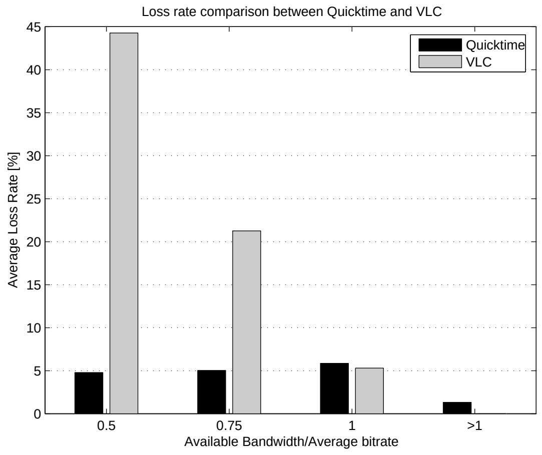


Fig. 3. Loss rate comparison between Quicktime and VLC

A. Darwin Streaming Server evaluation

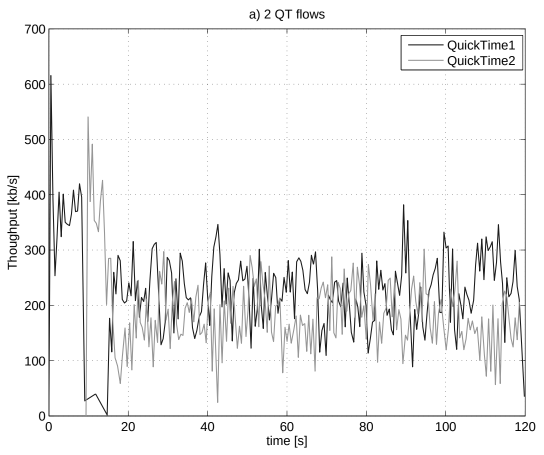
1) VideoLAN vs QuickTime Player : In this scenario we have evaluated the impact of using QTP - that supports Reliable UDP protocol - w.r.t. using a generic player, such as VLC, that does not support such protocol. We compared QTP and VLC using the same available bandwidth. Using a large available bandwidth of $800\,kb/s$, the average streaming rate (without any loss events) was $264\,kb/s$ with VLC. We used a value of $B_{avg} = 320\,kb/s$ as a safe approximation of the required bandwidth for one streaming flow.

We used 4 values for the available bandwidth: $0.50 \cdot B_{avg}$, $0.75 \cdot B_{avg}$, B_{avg} and $> B_{avg}$, i.e. $800\,kb/s$. Figure 3 shows the average loss bitrate obtained with QTP and VLC, respectively. It can be seen that VLC loss rate percentages are roughly equal to the super-imposed bandwidth reductions, whereas, on the other hand, the QTP loss rate is always around 5%. This result shows that the Reliable UDP congestion control effectively succeeds at providing a low loss rate by adapting the sending rate to the available bandwidth.

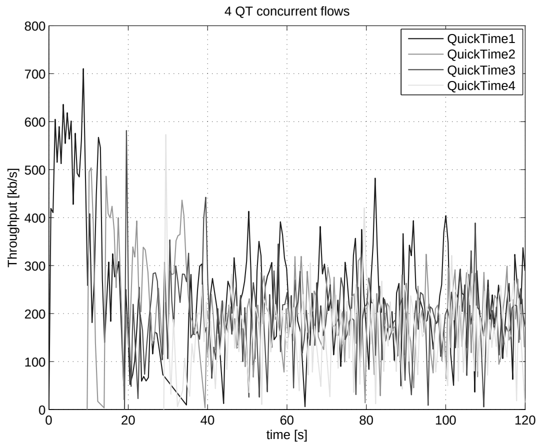
2) Two and four QuickTime concurrent flows: In this scenario we tested DSS in presence of 2 or 4 concurrent streaming sessions over the same link with super-imposed bandwidth reductions in order to evaluate the fairness of Reliable UDP.

In the case of 2 concurrent QTP streaming sessions we imposed a bandwidth equal to $0.75 \cdot B_{avg} \cdot 2 = 480\,kb/s$, whereas in the case of 4 concurrent sessions we imposed a bandwidth equal to $0.75 \cdot B_{avg} \cdot 4 = 960\,kb/s$. Each connection starts after a $10s$ delay from the other in order to avoid overlaps between the pre-buffering phases.

As it is shown in Figure 4, during the first $10\,s$ of each connection, each flow tends to take up all the available bandwidth in order to fill the playout buffer (over-buffering phase). After this phase, DSS tries to allocate for each flow a bandwidth equal to the average streaming rate B_{avg}, but the first flows tends to occupy more resources, as it is shown in Table I. Moreover, the second flow experiences a high loss rate, i.e. the adopted congestion control is aggressive and tries



(a)



(b)

Fig. 4. Two (a) and four (b) QuickTime concurrent flows

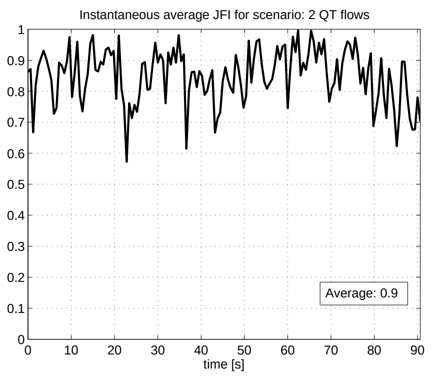
to reallocate bandwidth continuously.

In order to measure the fairness, we have evaluated the instantaneous Jain Fairness Index (JFI) [18] for each connection. As it is shown in Figure 5, the instantaneous JFIs oscillate in the range $[0.5, 0.98]$ around their average values.

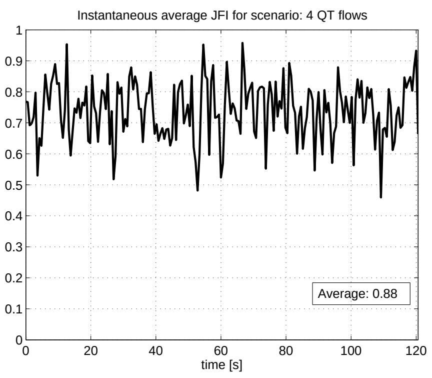
3) One and four QuickTime concurrent flows sharing a square wave available bandwidth: In this scenario we have used an available bandwidth that varies as a square wave in the range $[100, 1600] \cdot N\, kb/s$ (where N is the number of concurrent flows, i.e. $N = 2$ or $N = 4$) with a period which is set equal to $40\, s$ in order to investigate how DSS adapts his sending rate.

As it is shown in Figure 6, DSS adapts its sending rate in order to match the available bandwidth, using a pre-buffering phase every time spare bandwidth is available. After each pre-buffering phase, the throughput matches again B_{avg}.

In the case of four QuickTime concurrent flows, it results that the flow that have been started first takes more bandwidth, whereas the others experience a higher loss rate (Table I).



(a)



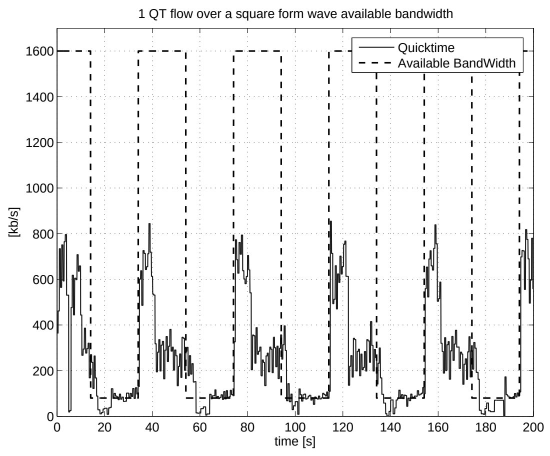
(b)

Fig. 5. JFI index for two (a) and four (b) QuickTime concurrent flows

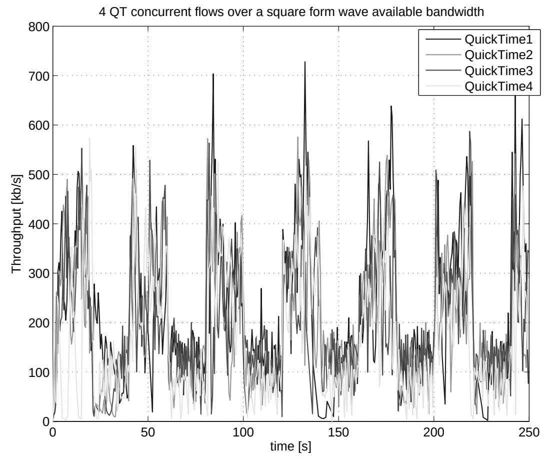
Experiment	Flow	TP (kb/s)	LR (kb/s)	JFI
2 QT, constant bw	QT1	208.34	18.36	0.98
	QT2	171.97	26.83	
4 QT, constant bw	QT1	217.81	30.56	0.96
	QT2	183.37	37.15	
	QT3	178.14	48.21	
	QT4	154.34	51.54	
1 QT, variable bw	QT	257.67	10.91	-
4 QT, variable bw	QT1	212.52	24.96	0.99
	QT2	202.49	31.35	
	QT3	195.53	36.32	
	QT4	163.05	28.73	
1 QT, lossy link	QT	203.45	25.70	-

TABLE I
DSS TEST RESULTS WITH AVERAGE THROUGHPUTS (TP), LOSSRATES (LR) AND JAIN FAIRNESS INDEXES (JFI).

4) One QuickTime flow over a lossy link: In this scenario we have evaluated the behavior of DSS and QTP in the presence of a lossy link, emulated using a Gilbert model. Figure 7 shows the throughput and the loss rate imposed by the Gilbert model. As it is shown in Table I, in this case the throughput is roughly equal to B_{avg}, i.e. the congestion control implemented by DSS exhibits a fast bandwidth reallocation, likely because Reliable UDP uses a slow-start phase after each loss event.



(a)



(b)

Fig. 6. One (a) and four (b) QuickTime concurrent flows over a square wave available bandwidth

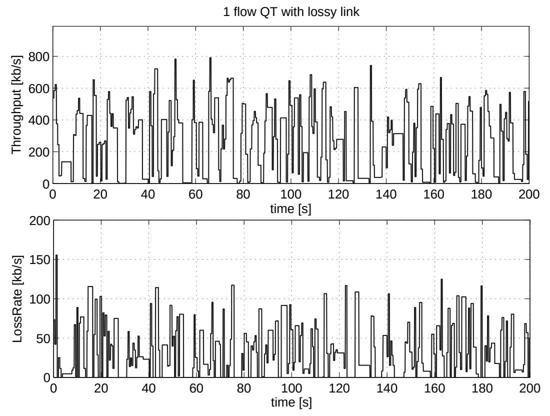


Fig. 7. One QuickTime flow over a lossy link

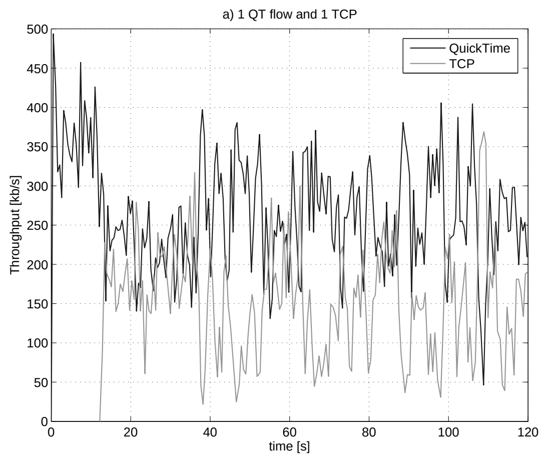


Fig. 8. One QuickTime flow with one concurrent TCP flows

B. Darwin Streaming Server vs TCP

1) One QuickTime flow with one concurrent TCP flows: In order to evaluate the TCP-friendliness provided by QuickTime streams we have set up a scenario in which the link capacity is set equal to to $0.75 \cdot B_{avg} \cdot 2 = 480\,kb/s$. The TCP flow is started $10\,s$ after the QuickTime flow. As it is shown in Figure 8, the TCP flow does not grab the same bandwidth share of the QuickTime flow, thus resulting in a JFI which is equal to 0.86. The values of the average throughputs and loss rates are reported in the Table II.

2) Many QuickTime flows with many concurrent TCP flows: In this scenario we report results obtained about the TCP-friendliness of QuickTime streams using more concurrent connections. In particular we have run the following experiments: 1 QT and 3 TCP streams, 2 QT and 2 TCP streams and finally 3 QT and 1 TCP stream. We imposed an available bandwidth equal to $0.75 \cdot B_{avg} \cdot 4 = 960\,kb/s$. In Figure 9, it is shown that QT flows tend to be unfair when sharing the bottleneck with TCP flows. Table II provides in the case of 2 QT and 2 TCP flows a JFI equals to 0.81, instead in the case of 3 QT and 1 TCP flows the JFI is 0.87.

3) Many QuickTime flow with many concurrent TCP flows over a square wave available bandwidth: In this scenario we have evaluated the TCP-friendliness of QuickTime streams in presence of time-varying available bandwidth, that varies as a square wave in the range $[100, 1600] \cdot N\,kb/s$ (where N is the number of total concurrent flows). Figure 10 shows the experiment results. In the case of 1 QT and 1 TCP flow, after each over-buffering phase, the QT flow takes a bandwidth roughly equal to B_{avg}, so that the TCP flow (which is a greedy source) can take a higher bandwidth. On the other hand, in the case of 2 QT and 2 TCP flows, the QT flows take always more bandwidth than TCP ones.

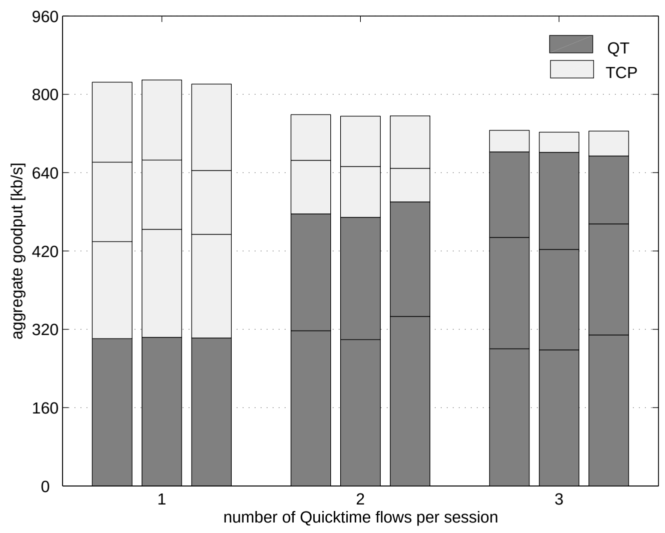
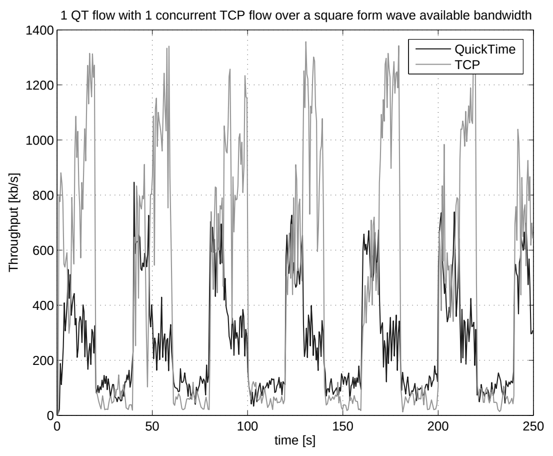


Fig. 9. Many Quicktime flows with many concurrent TCP flows

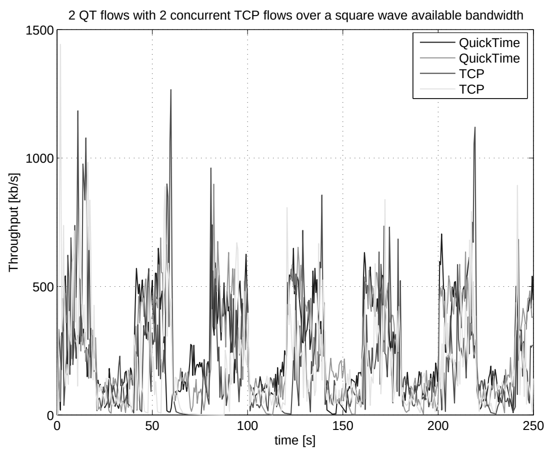
Experiment	Flow	TP (kb/s)	LR (kb/s)	JFI
1 QT + 1 TCP, const bw	QT	265.96	22.24	0.93
	TCP	150.30	13.71	
1 QT + 3 TCP, const bw	QT	250.44	27.09	0.96
	TCP1	198.18	24.45	
	TCP2	150.50	23.94	
	TCP3	179.20	23.61	
2 QT + 2 TCP, const bw	QT1	278.82	22.52	0.81
	QT2	215.62	31.61	
	TCP1	87.70	18.49	
	TCP2	121.14	21.92	
3 QT + 1 TCP, const bw	QT1	249.55	27.32	0.87
	QT2	216.97	36.67	
	QT3	144.67	33.40	
	TCP	93.76	18.46	
1 QT + 1 TCP, variable bw	QT	262.96	15.65	0.93
	TCP	453.86	24.17	
2 QT + 2 TCP, variable bw	QT1	253.24	20.76	0.97
	QT2	252.64	20.28	
	TCP1	195.46	24.23	
	TCP2	172.51	23.55	

TABLE II

DSS VS TCP EXPERIMENTAL RESULTS REPORTING AVERAGE THROUGHPUTS (TP), LOSS RATES (LR) AND JAIN FAIRNESS INDEXES (JFI).



(a)



(b)

Fig. 10. Many QuickTime flow with many concurrent TCP flows over a square wave available bandwidth

VI. Conclusions and Further Work

We have carried out an investigation of the Darwin Streaming Server (DSS) in order to evaluate the behavior of Reliable UDP in the case of time varying network bandwidth and in the presence of multiple concurrent QuickTime and TCP flows.

The main findings of this paper are the following: i) when a client that does not implement Reliable UDP is used , no congestion control is performed at all, and the loss rate is proportional to the available bandwidth/encoding rate ratio; ii) when a Reliable UDP-capable client is used, there is an effective adaptation to the network bandwidth; iii) when more concurrent QuickTime flows share the same link, the available bandwidth is shared equally, albeit exhibiting large oscillations; iv) QuickTime flows are very unfriendly with respect to concurrent TCP flows and they eperience higher loss rate.

References

[1] “YouTube - Broadcast Yourself.” http://www.youtube.com/.

[2] Van Jacobson and Michael J. Karels, “Congestion avoidance and control,” *ACM SIGCOMM Computer Communication Review*, Jan. 1995.

[3] Sally Floyd and Kevin Fall, “Promoting the Use of End-to-End Congestion Control in the Internet,” *IEEE/ACM Transactions on Networking*, 3 May 1999.

[4] L. Ding and R. Goubran, “Assessment of effects of packet loss on speech quality in VoIP,” *Haptic, Audio and Visual Environments and Their Applications, 2003. HAVE 2003. Proceedings. The 2nd IEEE Internatioal Workshop on*, pp. 49–54, 2003.

[5] T. Hayashi, S. Yamasaki, N. Morita, H. Aida, M. Takeichi, and N. Doi, “Effects of IP packet loss and picture frame reduction on MPEG1 subjective quality,” *Multimedia Signal Processing, 1999 IEEE 3rd Workshop on*, pp. 515–520, 1999.

[6] M. Handley, S. Floyd, J. Padhye, and J. Widmer, “TCP Friendly Rate Control (TFRC): Protocol Specification,” Jan. 2003.

[7] *QuickTime Streaming Server Modules Programming Guide*.

[8] “http://developer.apple.com/opensource/server/streaming.”

[9] “http://www.apple.com/it/quicktime/download/.”

[10] “VideoLAN.” http://www.videolan.org/.

[11] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang, “Delving into internet streaming media delivery: a quality and resource utilization perspective,” *Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pp. 217–230, 2006.

[12] J. Chung, M. Claypool, and Y. Zhu, “Measurement of the Congestion Responsiveness of RealPlayer Streaming Video Over UDP,” *Proceedings of the Packet Video Workshop (PV)*, 2003.

[13] J. Nichols, M. Claypool, R. Kinicki, and M. Li, “Measurements of the congestion responsiveness of windows streaming media,” *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, pp. 94–99, 2004.

[14] S. Hessler and M. Welzl, “An Empirical Study of the Congestion Response of RealPlayer, Windows MediaPlayer and Quicktime,” *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC'05)-Volume 00*, pp. 591–596, 2005.

[15] Oskar Andreasson, *Iptables Tutorial 1.2.0*. World Wide Web, http://iptables-tutorial.frozentux.net/iptables-tutorial.html, 2005.

[16] L. Rizzo, “Dummynet: a simple approach to the evaluation of network protocols,” *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[17] J. Hartwell and A. Fapojuwo, “Modeling and characterization of frame loss process in IEEE 802.11 wireless local area networks,” *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 6, 2004.

[18] R. Jain, A. Durresi, and G. Babic, “Throughput Fairness Index: An Explanation,” *ATM Forum Contribution 99*, vol. 45, 1999.