Supporting Large Scale eResearch Infrastructures with Adapted Live Streaming Capabilities

Syed Hasan[1], Laurent Lefevre[2], Zhiyi Huang[1] and Paul Werstein[1]

[1] Department of Computer Science
University of Otago,
Dunedin, New Zealand,
Email: shasan|hzy|werstein@cs.otago.ac.nz

[2] INRIA RESO - LIP (UMR CNRS, INRIA, ENS, UCB) - University of Lyon
Ecole Normale Superieure,
46, allee d'Italie - 69364 LYON Cedex 07 - FRANCE ,
Email: laurent.lefevre@inria.fr

Abstract

Large scale e-Research environments face classical distributed challenges: performance, heterogeneous equipment and variable contexts. The users of such infrastructures want to benefit from full interactive environments based on multimedia streams (voice, video, virtual reality) which are difficult to design and support on a large scale basis. In this paper, we present a new approach to support the streaming of live flows between e-Researchers. We show that traditional techniques (using TCP-based live streaming) are unsuitable for infrastructures with long delay and high loss rate. TCP introduces rate oscillations and requires more buffering and bandwidth to sustain a smooth playback. We propose a streaming framework which provides smoother rate control than TCP and improves streaming performance based on cross-layer feedback between the transport protocol and the streaming server. Our solution keeps the buffer usage at the client and server to a minimum level and provides quick rate adaptation. This paper presents simulation results for streaming in different eResearch scenarios.

Keywords: Live streaming, eResearch, Congestion control, Multimedia communication, Streaming media.

1 Introduction

Many eResearch projects are large enough to require the skills and expertise of scientists distributed around the world. Some of these eResearch problems are based on regional and geographical contexts, in which collaboration across distance is critical (Anderson & Kanuka 2003). Recently several projects have created collaborative eResearch tools and infrastructures (Paterson et al. 2007, *Sakai Project* 2005). These projects use the Internet as a viable platform for long distance collaboration among eResearchers. Live streaming is one of the key techniques for disseminating lectures, tutorials and eLearning content in such a collaborative research environment. However the user experience of streaming over the Internet is not always satisfactory. A recent measurement study on Internet streaming has reported that about 13% home and 40% of business streaming sessions suffer various quality degradations (Guo et al. 2006).

Historically the Internet does not provide any Quality of Service (QoS) such as minimum bandwidth or delay in packet transmission. Depending on the level of over provisioning, the available bandwidth may vary significantly at times of congestion. In order to create high quality interactive sessions, the eResearch community must have access to high bandwidth links which may not be always available among all researchers even in today's modern universities. This situation is more complicated when the research project involves collaboration with research facilities in developing countries with comparatively low bandwidth, high inter-link delay and loss rate.

A recent measurement study has reported significant disparity in end-to-end link delay between different parts of the world (ICFA-SCIC 2007). According to that report, the minimum RTT between United States and some East Asian or African countries ranges between 250ms to 400ms. The minimum RTT between United States and most European or Australasian countries varies between 100 to 250ms. However, in some African countries where satellite links are still prevalent, the minimum RTT is above 600ms. As the delay between links increases, interactive communication using traditional techniques becomes more challenging.

In this paper, we illustrate the performance of live streaming in three different eResearch scenarios. The scenarios were chosen to represent links with different RTT groups:

- National eResearch infrastructure : links within New Zealand.
- Large scale eResearch infrastructure : links between Australia/New Zealand and North American/European countries.
- Worldwide eResearch infrastructure : links between Australia/New Zealand and East Asia/African Countries.

eResearch infrastructures can benefit from traditional group communications in networks (like multicast) to allow efficient delivery to large number of clients (see Figure 1). However, in contrast to traditional client-server based live streaming methods, eResearch communities also need to establish point-to-point streaming session between end hosts. The possible scenario is shown in Figure 2. In addition to regular PC based sessions, there are ubiquitous computing devices like PDAs and hand-held mobile devices connected through different access links having various QoS requirements. Often the ubiquitous

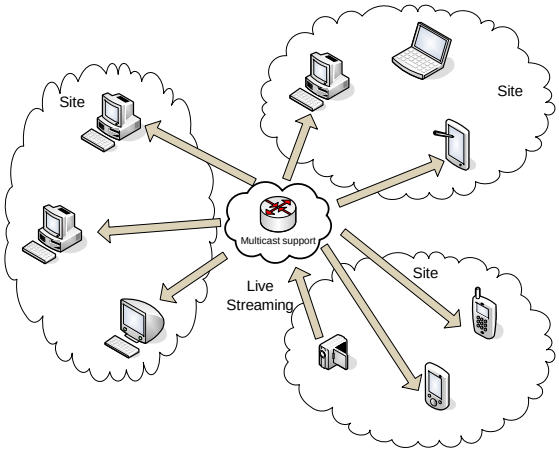

Figure 1: Multicast supported e-Research scenario

computing devices used in eResearch may have limited memory and processing power. As a result, the traditional streaming techniques which require significant computational resources may appear to be insufficient in many occasions. A suitable streaming framework for eResearch scenarios must support devices with limited memory and processing resources.

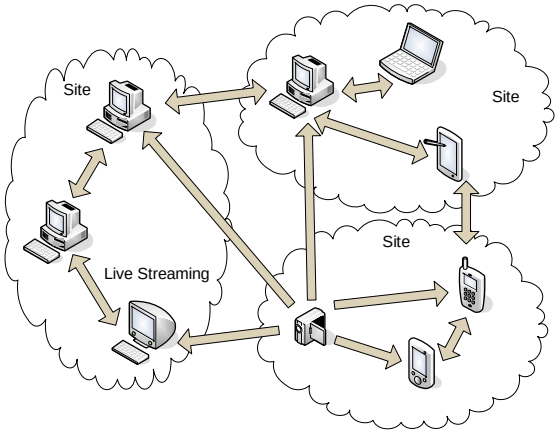

Figure 2: Point-to-Point eResearch scenario

Streaming applications use a playout buffer at the client side to hide the inter-packet delay variation from the playback process. The idea is to prefetch some packets for future playback which protects the stream playback from stalling when the available bandwidth drops below the application's streamed bit rate. Although this mechanism can protect playback disruptions for a brief period of congestion, the size of the playback buffer increases as the inter-link delay and loss increases. For live streaming, the buffering adds delay in playback, and this delay is unacceptable beyond a certain range.

Live streaming applications can tolerate a few packet drops but require in-time delivery of packets. Although UDP is a preferable transport protocol to most streaming applications, very often UDP is blocked by firewalls for security reasons. As a result, TCP is used by almost 70% of live streaming sessions on the Internet (Guo et al. 2006). TCP's congestion control mechanism proactively controls the sending rate of the application. On a single packet loss, TCP cuts the transmission rate by half and blocks the delivery of packets to the receiving application until the lost packet is received through retransmission. Realizing the limitations of TCP, the Internet Engineering Task Force (IETF) is designing a new transport protocol named Datagram Congestion Control Protocol (DCCP) (Kohler et al. 2006) which decouples reliability from congestion control and incorporates TCP-Friendly Rate Control (TFRC) (Handley et al. 2003) as a viable rate control algorithm for multimedia applications. TFRC is a equation based rate control algorithm which provides smoother throughput while being friendly to TCP applications. DCCP is still under development and requires more testing.

In this paper, we conduct experiments with TCP and TFRC for streaming in different eResearch scenarios and propose a framework for Point-to-Point streaming. Our approach makes the streaming application more adaptive by providing fine grain cross layer feedback between the application and the transport protocol. We introduce Dynamic Buffer Active Tuning (DBAT), which monitors the send buffer queue size and provides feedback to the application when the queue size increases beyond a threshold. Using network simulator *ns-2* (ns-2 n.d.), this paper illustrates that the proposed framework requires less buffering delay and improves streaming performance by reducing playback interruptions.

The paper is organized as follows. In Section 2, some background on streaming techniques and the underlying transport protocols are discussed. The proposed framework is presented in Section 3. In Section 4 the experimental results are illustrated. Related work is discussed in Section 5, and Section 6 contains the conclusions and future work.

2 Background

Classical streaming applications support several streamed bit rates in order to match the available bandwidth with the streamed bit rate. In addition, streaming incorporates several quality adaptive techniques. We begin this section with a brief discussion on the application model for streaming, its key performance indicators and present two protocols.

2.1 Streaming Application Model

In traditional streaming solutions, the client and server exchange control packets to negotiate appropriate sending rates. At the beginning of the session, the server uses a packet pair based bandwidth probing technique to determine the available bandwidth and chooses the streaming bit rate accordingly. A playout buffer is used at the client side to reduce the effects of inter-packet jitter. Playback starts as soon as the buffer is filled.

A streaming system goes through three phases:

- *Buffering:* If the size of the playout buffer is large, the initial buffering period is longer, but it protects against playback interruptions when the available bandwidth briefly drops below streamed bit rate. For live streaming, this delay in buffering should be small.
- *Playback:* As long as there are packets at the playout buffer, the client keeps playing at the encoding rate.
- *Rebuffering:* If the buffer gets empty, playback has to stop until the buffer is filled to the threshold level.

The streaming client and server are in a control loop to monitor the packet loss rate and the client side buffer's status. Depending on the available bandwidth, the streaming server may change its streamed bit rate to a new rate whenever the packet loss reaches a predefined threshold or a buffering event occurs. As shown in Figure 3, this loop is decoupled from the rate

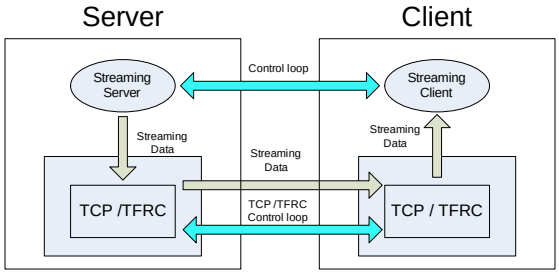

Figure 3: Classical streaming architecture based on TCP / TFRC

control loop of the transport protocol, i.e. there is no exchange of cross-layer information.

2.2 Streaming Performance Metrics

Streaming performance can be evaluated in terms of packet loss rate, rebuffering events, packet loss rate, smoothness in achieved throughput of streamed flows.

- *Number of Packet lost in Burst:* Streaming audio-video applications are able tolerate a few packet losses but streaming performance degrades if packets are lost in burst.
- *Number of Rebuffering Events:* Every time the playout buffer gets drained below a threshold, playback is paused while the buffer is refilled. These abrupt interruptions in playback negetively impact the quality of the streaming session. The number of rebuffering events and the percentage of time spent for buffering can be a good performance indication of streaming service.
- *Average Service Rate:* If the application is able to sustain a high bit rate for long time, the streamed content is of high quality. This improves the users perception of streaming.

A streaming application is able to reduce the number of packet losses and/or the possibility of rebuffering events by quickly adjusting the sending rate. The role of the underlying transport protocol is of utmost importance for such an application. A send buffer is employed to deal with the rate mismatch between the application's sending rate and the transport protocol's allowed transmission rate. This buffering adds end-to-end delay and may become an obstacle for achieving the new streamed bit rate when stream switching occurs. We call this stream switch response time.

A canonical streaming application emits packets at a constant rate. The transport protocol is responsible for sending the packets from the send buffer to the network interface. Packets are queued at the send buffer when the application's packet generation rate is lower than the transmission rate of the underlying transport protocol. As the feedback delay between the client and the server increases, the client's feedback becomes outdated, and the application level rate adaptation mechanism fails to react soon enough to reduce packet loss and rebuffering events. Some mechanism for reducing this feedback delay will be hugely beneficial.

2.3 Transport Protocol for streaming

2.3.1 TCP

It is well known that TCP's congestion control mechanism is vital for the scalability of the Internet (Jacobson 1988). In order to avoid congestion and ensure fairness among competing flows, TCP controls the sending rate of the application using an Additive-Increase-Multiplicative-Decrease (AIMD) algorithm. TCP keeps an estimate of the available bandwidth for the next RTT using a variable known as a congestion window.

Although UDP is preferable to most streaming applications, TCP is used more often. However the reliable, in order and congestion controlled service model of TCP is inappropriate for streaming flows which require more control and flexibility over its flows. The following are the main obstacles for streaming using TCP:

- *Information Hiding:* TCP hides the loss rate and RTT information from the application.
- *Buffering Delay:* TCP's window based congestion control mechanism requires a send buffer at the application to transport layer interface for briefly storing the in-flight packets as well as enough new packets to saturate the congestion window in the next flight.
- *Abrupt Rate Controlling:* TCP's AIMD reduces the application's sending rate by half on a single packet loss. The application does not get enough time to adapt the sending rate. As a result, a large number of packets are buffered at the send buffer.
- *Head-of-line Blocking:* Whenever a packet loss is detected, TCP's in order delivery mechanism blocks the delivery of received packets to the client until the lost packet is delivered through retransmission.

Although the effects of TCP's rate fluctuation due to congestion control can be reduced using the client side playout buffer, buffering becomes insufficient to reduce the effects of rate variation as the link delay increases. For live streaming, it is challenging to use TCP if the link-delay and/or loss rate is significant.

2.3.2 TFRC

TCP-Friendly Rate Control (TFRC) (Handley et al. 2003) is a rate control algorithm which provides smoother throughput by reacting more slowly to packet loss while being friendly to other TCP flows. In order to be a good network citizen, a deployable congestion control algorithm should be friendly to TCP flows. A flow is TCP-friendly if its average sending rate is no more than a TCP flow running between the same links. A TFRC sender calculates the TCP throughput using a TCP equation (Padhye et al. 1998) based on the receiver's feedback of loss rate, received packet rate, and the RTT information.

TFRC has been incorporated as an alternative congestion control algorithm for the newly standardized Datagram Congestion Control Protocol (DCCP) (Kohler et al. 2006). DCCP provides an unreliable service with reliable connection establishment and option negotiation states. Applications using DCCP have the option to choose different congestion control mechanisms for each direction. Currently only two types of congestion control have been standardized: TCP-like and TFRC.

Due to its smoother rate control, TFRC requires less playout buffer space than TCP. However various studies report poor performance of TFRC based audio-video transmission. For streaming application, even though TFRC reacts slowly on congestion events, the sender only reacts based on the receiver's feedback. An earlier feedback on congestion events

would give more time for the sender to change its sending rate.

3 Proposed Framework: Dynamic Buffer Active Tuning (DBAT)

In this section, we discuss our active queue management mechanism named Dynamic Buffer Active Tuning (DBAT). The goals of DBAT are as follows:

- *Reduce buffering delay:* DBAT keeps the send buffer queue at a minimum level.
- *Provide feedback to the application:* DBAT sends feedback to the application depending on the level of send buffer queue size.
- *Preferential treatment of marked packets:* When the send buffer queue size increases beyond a certain threshold, DBAT only sends the marked packets.

3.1 Motivation

The motivation for designing DBAT is to make the streaming application more adaptive and reactive. The traditional method of changing streamed bit rate based on packet loss rate and client side buffer underflow is inefficient. By the time the application reacts to the changing available bandwidth, it might be too late due to the delay in the feedback loop and send buffer.

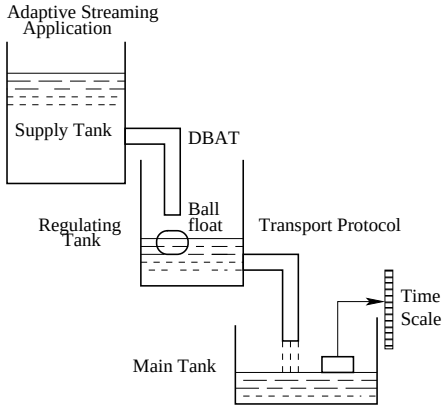

Figure 4: Water Clock Model

The idea behind DBAT can be easily understood by looking at the water clock model shown in Figure 4 (Luenberger 1979). In order to maintain a constant flow rate into the main tank of the clock, the water level at the regulating tank is held nearly constant. This constant level is achieved through a float valve, which is essentially a feedback mechanism. Water from an external supply enters the regulating tank through a pipe. As the water level at the regulating tank rises, it forces the floating ball to tighten against the pipe opening, reducing the input supply rate. When the level drops, the input rate increases. In our streaming architecture, DBAT plays the role of the regulating tank to keep the packet transmission rate at a constant level.

3.2 DBAT Architecture

As shown in Figure 5, DBAT couples the application's control loop with the transport protocol's control loop. Upon connection establishment, the application informs the transport protocol about its bit rate requirement and the transport protocol tries to sustain that rate. Note that streaming applications are data limited and as such cannot grab the available capacity. Knowing the application's desired rate, the transport protocol limits its sending rate up to a certain range.

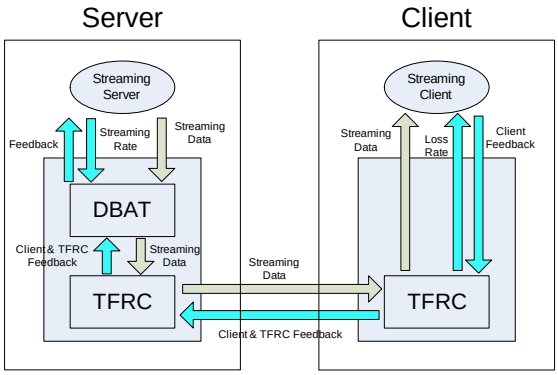

Figure 5: DBAT Architecture

DBAT monitors the send buffer queue size and sends feedback to the sender application when the queue size is increased beyond a threshold. As shown in Figure 6, DBAT keeps a minimum threshold and a maximum threshold for controlling the send buffer queue size. Minimum threshold is calculated by multiplying the streamed bit rate with the delay and the maximum threshold is set to twice the minimum threshold.

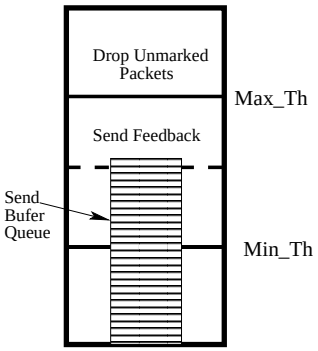

Figure 6: DBAT Queue

On each packet arrival the weighted queue length, *Qavg*, is calculated. If the average queue length increases beyond the mid point of minimum and maximum threshold, feedback is sent to the sender. The amount of feedback is limited to at most one per RTT. If *Qavg* grows beyond the maximum threshold, only marked packets are transmitted. The algorithm is presented in Figure 7.

```
Min_Th = streamed_bit_rate * delay
Max_Th = 2*Min_Th

On each packet arrival
   calculate weighted avg queue length Qavg
If (Qavg > Max_Th)
   preferentially drop unmarked packets
else if ((Qavg > (Max_Th + Min_Th )/2 and
          (last_feedback_time > RTT))
   send feedback
```

Figure 7: DBAT Algorithm

4 Experimental Results

In this section, we present the experimental results for different representative eResearch scenarios. We use network simulator, *ns-2* (ns-2 n.d.) for simulation. Currently the standard *ns-2* distribution does not have any streaming module included. However, we found a streaming module named *Goddard* (Chung et al. 2002) which is suitable for our experiments. We integrated *Goddard* in *ns-2* and conducted experiments using it. *Goddard* is based on the studied behaviors of Real Networks and Windows Streaming Media (Chung & Claypool 2006). During streaming the *Goddard* client and server reselect the streamed bit rate in response to network packet loss or rebuffering events that occur when the client's playout buffer gets emptied. The *Goddard* server supports multiple bit rate streaming. We vary the inter-packet gap to stream at the rate of 80, 120, 240, 320, 640, 960 and 1920 kbps.

Goddard does not have any support for TFRC. We modified the code so that we can use TFRC as a transport protocol for streaming. We found that the TFRC implementation in *ns-2* does not have any real data transmission capability which is required by the streaming module. We changed the interface of this implementation so that data can be transmitted with each packet enabling the *Goddard* client and server to exchange media frames. By default the *ns-2* implementation of TFRC has an infinite send buffer. We introduced a send buffer with adjustable size into TFRC. As for TCP, we modified the full TCP implementation of *ns-2* to support an adjustable send buffer size. To the best of our knowledge, we are the first to conduct experiments involving the interaction of streaming applications with TFRC in *ns-2*.

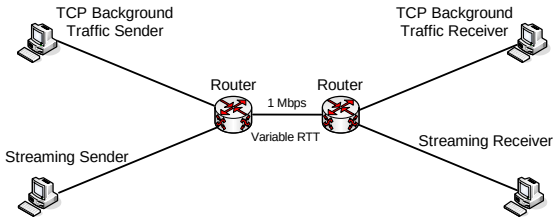

Figure 8: Live streaming simulation topology

We use the dumbbell topology for simulating Point-to-Point live streaming in eResearch sessions (Figure 8). Depending on the scenario, we set the link delay. The bottleneck link is 1 Mbps. In all cases, one streaming flow is competing with a background FTP flow and some HTTP traffic. The HTTP traffic is generated using empirical data provided by *ns-2*. The background FTP and HTTP traffic simulates a real world scenario where most streaming flows compete with web and FTP flows. The FTP application starts at 0.1 second and stops at 200 seconds. The streaming flow starts at 30 seconds and stops at 240 seconds. The bottleneck router queue size is set to twice the bandwidth and delay product of the link. Due to the randomness of the background HTTP traffic, the loss rate of the bottleneck link may vary. Therefore for each scenario, we run the experiment several times and plot the average values. The send buffer size for TCP and TFRC based streaming is set to a fixed value of 64 packets. This is in line with the existing configurations in Linux and Windows systems where the default send buffer is set to 64KB (64 1KB packets).

For each scenario, we determine the throughput and end-to-end application level latency. The throughput is calculated with one second granularity. The latency is calculated by measuring the delay between the time when the streaming sender sends a packet to the transport layer and the time when that packet arrives at the streaming receiver. In all cases, DBAT is used on top of TFRC.

4.1 Scenario 1: National eResearch infrastructure : links within New Zealand.

Often eResearchers need to collaborate on projects undertaken at the national level. The RTT for Scenario 1 is set to 20ms which is an approximation of maximum RTT within New Zealand.

Figures 9, 10 and 11 show the throughput of TCP, TFRC and DBAT based streaming, respectively. In comparison to TCP, TFRC and DBAT provides better streaming throughput. In case of TCP, the available bandwidth is shared equally among the streaming flow and the background FTP flow.

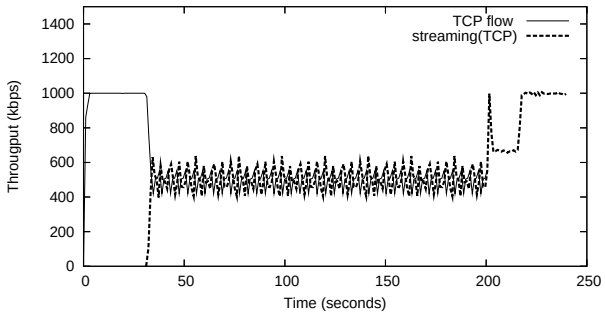

Figure 9: Streaming(TCP) throughput

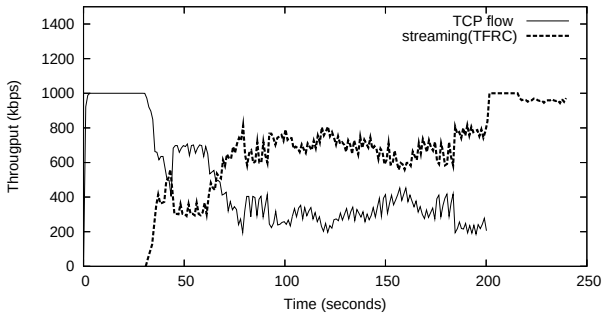

Figure 10: Streaming(TFRC) throughput

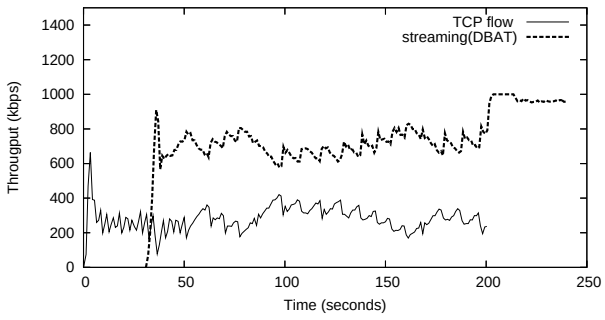

Figure 11: Streaming(DBAT) throughput

Due to the background HTTP traffic, packets are dropped from the bottleneck router causing TFRC and TCP flows to retreat. Since TFRC only reacts to average packet loss rate, the TFRC flow gains more bandwidth than the TCP flows. It can be seen from Figure 11 that the background HTTP flow also affects the FTP throughput during the initial period.

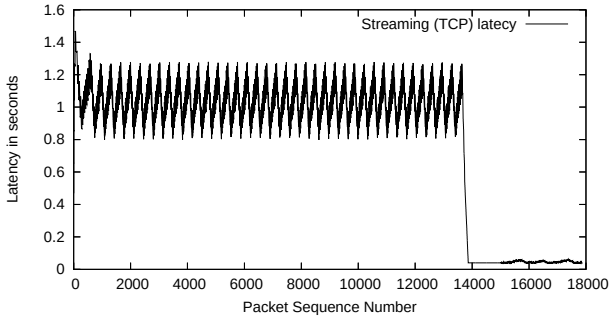

Figure 12: Streaming(TCP) latency

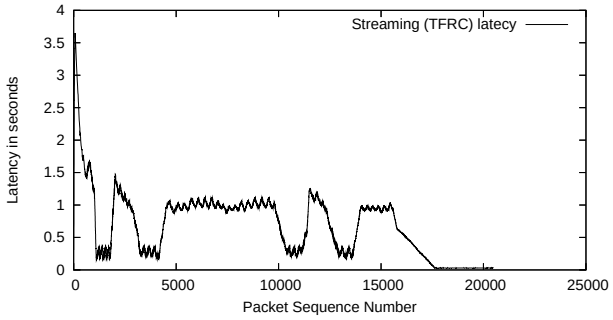

Figure 13: Streaming(TFRC) latency

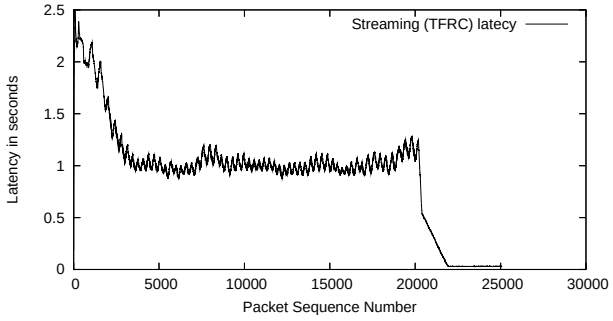

Figure 14: Streaming(DBAT) latency

The latency graphs for TCP, TFRC and DBAT based streaming are shown in Figures 12, 13 and 14 respectively. Due to the fixed size send buffer, latency varies for TFRC based streaming. The end-to-end delay increases is proportion to the size of the send buffer queue. We can see the difference in case of DBAT based streaming. Since DBAT keeps the send buffer queue level at a minimum level, the end-to-end latency is much smoother. The delay variation in TCP based streaming is within an acceptable range.

In this scenario, TCP based streaming is suitable. However DBAT and TFRC also provide good performance. In case of DBAT, the streaming throughput is higher.

4.2 Scenario 2: Large scale eResearch infrastructure : links between Australia/New Zealand and North American/European countries

Due to the geographic location of Australia and New Zealand, any communication with Europe and America involves long delay. The RTT for this scenario is set to 150ms. As shown in Figures 15, 16 and 17, TCP based streaming has lower throughput than TFRC or DBAT based streaming flows. With the increased link delay, the retransmitted packets are less likely to arrive at the client in time and hence rebuffering occurs frequently. This results in lower throughput. Since TFRC or DBAT do not retransmit lost packets, the streamed service rate is higher.

Figure 18 illustrates the latency for TCP based streaming flows. The latency varies by almost 2 seconds between consecutive packets. Due to this variance, the playout buffer gets empty and streaming pauses for rebuffering.

DBAT based streaming provides the highest streaming throughput. Since DBAT flows are more adaptive, the streaming sender is able to pump the packets at a higher rate without losing many packets in a burst. It can be noted that this time TFRC is fair with the TCP flows.

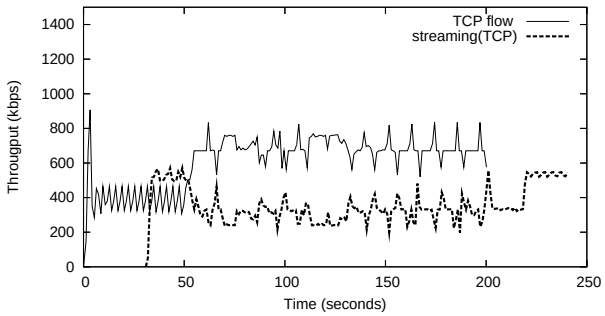

Figure 15: Streaming(TCP) throughput

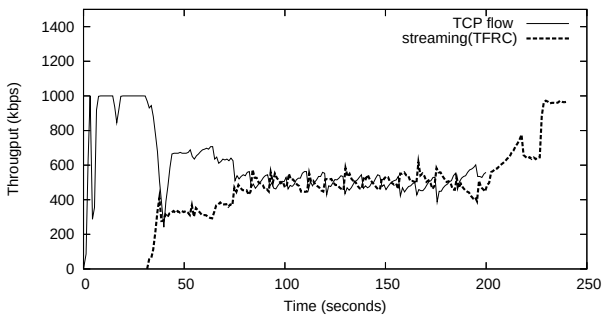

Figure 16: Streaming(TFRC) throughput

Figures 19 and 20 show that more packets are transmitted with DBAT based streaming since DBAT attains a higher packet rate than TCP or TFRC based streaming. In this scenario, TCP performs poorly. To overcome these issues, the streaming client needs more memory for the playout buffer. This additional buffering delay is unacceptable for live streaming sessions. Moreover, allocating the required buffer memory may not be possible in many cases when the client is a hand-held mobile device.

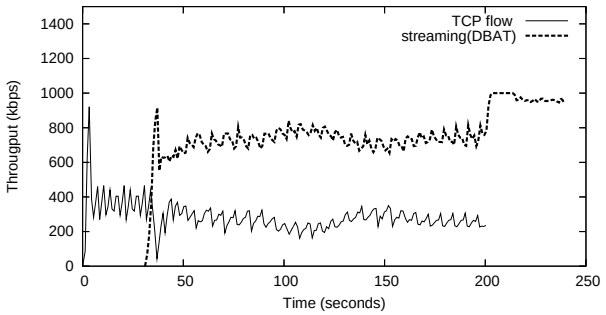

Figure 17: Streaming(DBAT) throughput

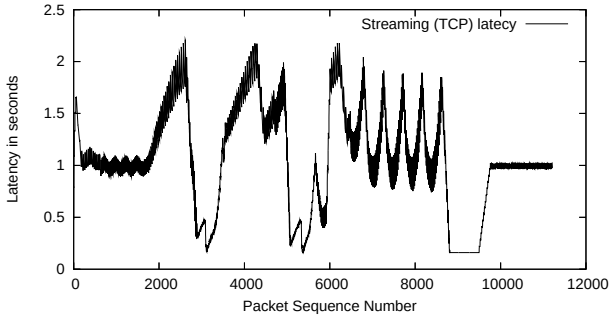

Figure 18: Streaming(TCP) latency

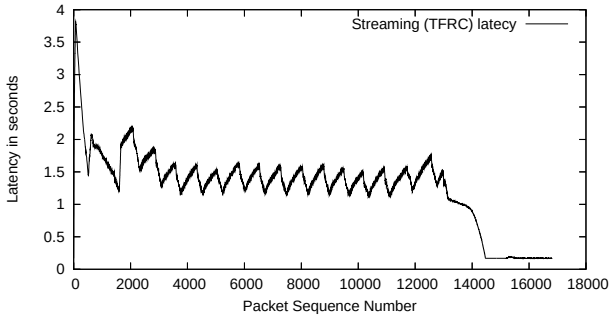

Figure 19: Streaming(TFRC) latency

4.3 Scenario 3: World wide eResearch infrastructure : links between Australia/New Zealand and East Asian/African Countries

Many East-Asian/African countries have limited Internet bandwidth and some of them are connected through satellite links. This type of infrastructure imposes additiona(like XCP (Katabi et al. 2002) needs in terms of fairness with TCP (Pacheco et al. 2007)) challenges in achieving workable streaming performance for eResearch. For this case, the RTT is set to 450ms. Figures 21, 22 and 23 illustrate that TCP based streaming has very low throughput in comparison to TFRC and DBAT based streaming flows.

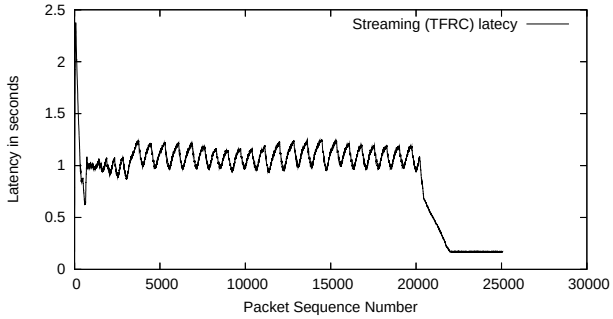

Figure 20: Streaming(DBAT) latency

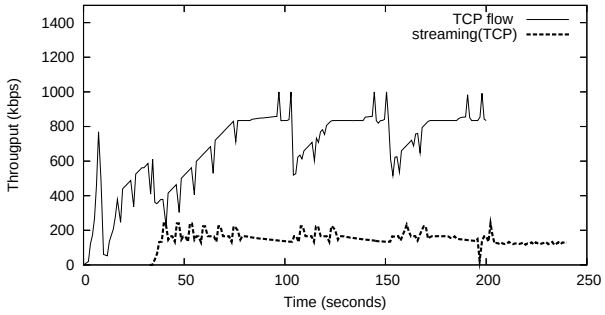

Figure 21: Streaming(TCP) throughput

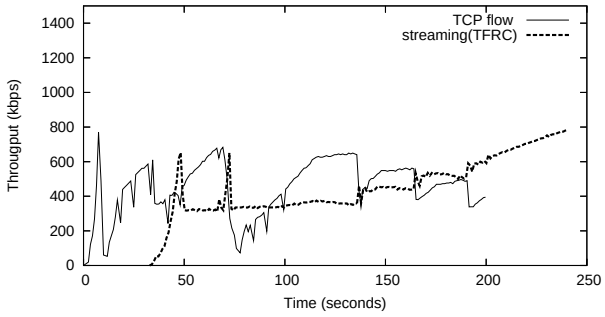

Figure 22: Streaming(TFRC) throughput

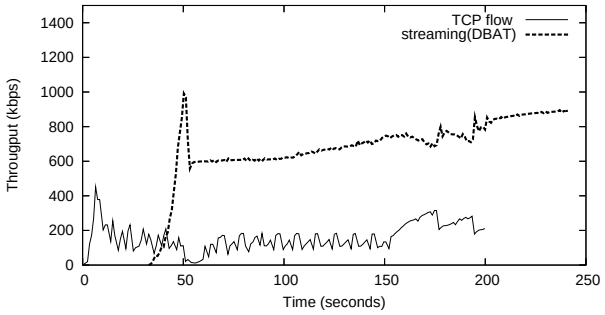

Figure 23: Streaming(DBAT) throughput

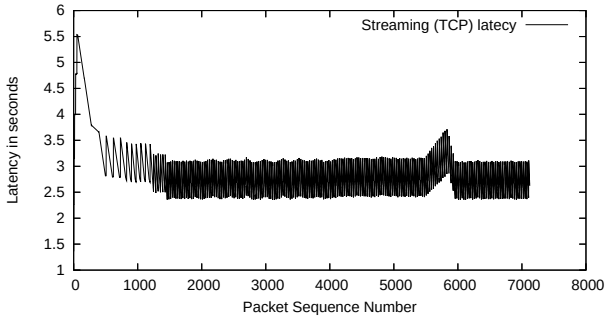

Figure 24: Streaming(TCP) latency

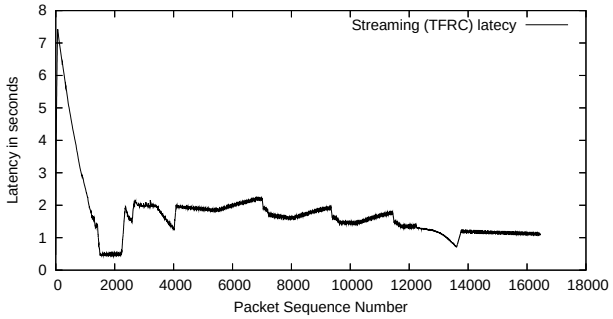

Figure 25: Streaming(TFRC) latency

As shown in Figure 24, inter-packet latency fluctuates significantly when streaming with TCP. Figures 25 and 26 show that TFRC and DBAT based streaming provide better latency performance, respectively. Despite the high inter-link delay, it is possible to experience better streaming playback if TFRC is used instead of TCP based streaming.

4.4 Discussion

The amount of time spent in buffering is the most important performance indicator for streaming sessions since playback remains paused during this period. As a result, the users' perception of streaming is aggravating. Table 1 shows the amount of time spent for buffering with different protocols in our representative scenarios. For Scenario 3, TCP based streaming flows spend 37.80% of its total streaming time in rebuffering. TCP shows good performance in Scenario 1 where the link delay is only 20 ms. This is because in low delay links TCP's acknowledgement based feedback mechanism is effective. TFRC and DBAT reduces the rebuffering period significantly. However the fairness issue between TFRC and TCP flows should be improved.

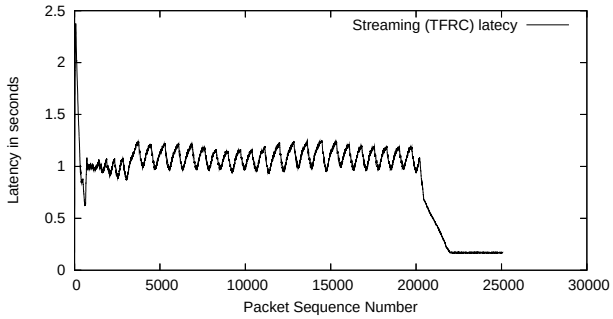

Figure 26: Streaming(DBAT) latency

	Scenario 1	Scenario 2	Scenario 3
TCP	0.39%	10.46%	37.80%
TFRC	0.84%	1.12%	4.44%
DBAT	0.43%	1.01%	4.13%

Table 1: Total buffering time

5 Related Work

In an experimental study, Balan et al. (Balan et al. 2007) reported that voice quality is not improved when TFRC is used for rate control. Wang et al. (Wang et al. 2004) developed an analytical model for TCP based streaming and concludes that TCP generally provides a good streaming performance when the achievable TCP throughput is roughly twice the media bit rate with only a few seconds of start up delay. Luo et al. (Guo et al. 2006) presented the result of measurement study based on large streaming media work load taken from thousands of broadband home users and business users hosted by a major ISP. It shows that the median time to change to a lower bit rate stream was around 4 seconds and proposes coordinated streaming, a mechanism that coordinates client side buffering and rate adaptation to reduce the stream switching delay. Krasic et al. (Krasic et al. 2003) presented a framework for adaptive video streaming based on priority dropping. Chung et al. (Chung et al. 2002) developed a transport level protocol named Media Transport Protocol(MTP) which removes the burden of in order delivery from TCP. Goel et al. (Goel et al. 2002) proposed a dynamic send buffer tuning approach where the buffer size is kept slightly larger than the TCP congestion window for TCP-based media streaming. Unlike their work, we focus on media streaming on TFRC.

6 Conclusions and Future Work

In this paper, we investigated the performance of streaming on three different eResearch infrastructures. We showed that TCP performs poorly when the inter-link delay is high, but TFRC based streaming improves the performance. We proposed Dynamic Buffer Active Tuning (DBAT) framework and showed that DBAT increases streaming throughput and reduces rebuffering events by making streaming flows more adaptive.

Our future work will involve exploration of a router assisted approach (like XCP (Katabi et al. 2002) needs in terms of fairness with TCP) to ensure fairness (Pacheco et al. 2007). We also will investigate the impact on DBAT in terms of responsiveness and reactivity. Moreover, we will focus on live streaming in group communication as shown in Figure 1 and solve some scalability issues (in terms of memory and

buffer usage) by deploying DBAT solutions in clients. DBAT can be tailored easily to streaming in multicast scenarios by using a layered media encoding format.

References

Anderson, T. & Kanuka, H. (2003), *E-research: Methods, Strategies, and Issues*, Allyn and Bacon, p. 73.

Balan, V., Eggert, L., Niccolini, S. & Brunner, M. (2007), An experimental evaluation of voice quality over the Datagram Congestion Control Protocol, *in* 'IEEE Infocom', Anchorage, Alaska, USA, pp. 2009–2017.

Chung, J. & Claypool, M. (2006), 'Empirical evaluation of the congestion responsiveness of Real Player video streams', *Kluwer Multimedia Tools and Applications* **31**(2), 171 – 193.

Chung, J., Claypool, M. & Kinicki, R. (2002), MTP: A streaming-friendly transport protocol, Technical report, Oregon Graduate Institute School of Science and Engineering.

Goel, A., Krasic, C., Li, K. & Walpole, J. (2002), Supporting low latency TCP-based media streams, *in* 'Tenth International Workshop on Quality of Service (IWQoS)', Miami, pp. 193– 203.

Guo, L., Tan, E., Chen, S., Xiao, Z., Spatscheck, O. & Zhang, X. (2006), Delving into internet streaming media delivery: A quality and resource utilization perspective, *in* 'Proceedings of the 6th ACM SIGCOMM on Internet measurement', Rio de Janeiro, Brazil, pp. 217–230.

Handley, M., Floyd, S., Padhye, J. & Widmer, J. C. (2003), 'TCP Friendly Rate Control (TFRC): Protocol specification', Internet Engineering Task Force, RFC 3448.

ICFA-SCIC (2007), ICFA-SCIC network monitoring report, Technical report, International Committee for Future Accelerators (ICFA) - Standing Committee on Inter-Regional Connectivity (SCIC).

Jacobson, V. (1988), Congestion avoidance and control, *in* 'ACM SIGCOMM', Stanford, California, United States., pp. 314–329.

Katabi, D., Handley, M. & Rohrs, C. (2002), Congestion control for high bandwidth-delay product networks, *in* 'Proceedings of the 2002 ACM SIGCOMM conference', pp. 89–102.

Kohler, E., Handley, M. & Floyd, S. (2006), Designing DCCP: congestion control without reliability, *in* 'ACM SIGCOMM 2006', Pisa, Italy, pp. 27–38.

Krasic, C., Walpole, J. & Feng, W.-c. (2003), Quality-adaptive media streaming by priority drop, *in* '13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)', pp. 112 – 121.

Luenberger, D. (1979), *Introduction to Dynamic Systems*, Wiley New York, chapter 8, p. 297.

ns-2 (n.d.), 'The Network Simulator ns-2', http://www.isi.edu/nsnam/ns/.

Pacheco, D. M. L., Lefevre, L. & Pham, C.-D. (2007), Fairness issues when transferring large volume of data on high speed networks with router-assisted transport protocols, *in* 'High Speed Networks Workshop 2007, in conjunction with IEEE INFOCOM 2007', Anchorage, Alaska, USA, pp. 46 – 50.

Padhye, J., Firoiu, V., Towsley, D. & Kurose, J. (1998), Modeling TCP throughput: A simple model and its empirical validation, *in* 'ACM SIGCOMM', pp. 30–314.

Paterson, M., Lindsay, D., Monotti, A. & Chin, A. (2007), 'DART: A new missile in Australia's e-research strategy', *Online Information Review* **31**(2), 116–134.

Sakai Project (2005), http://www.sakaiproject.org/.

Wang, B., Kurose, J., Shenoy, P. & Towsley, D. (2004), Streaming via TCP: An analytic performance study, *in* 'Proceedings of the 12th annual ACM international conference on Multimedia', New York City, NY, pp. 908–915.