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@ens-lyon.fr

Abstract

Large scale e-Research environments face classical distributed challenges: performance, heterogeneous equipments and variable contexts. But, the users of such infrastructures want to benefit from full interactive environments based on multimedia streams (voice, video, VR.) which are difficult to design and support on a large scale basis. In this paper, we present a new approach to efficiently 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 playout. We propose a streaming framework which provides a smoother rate control than TCP and improves streaming performance based on cross layer feedback between the transport protocol and 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 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[3]. Recently several projects have been undertaken to create collaborative eResearch tools and infrastructures[17][2]. 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 degradationes[9].

Historically the Internet does not provide any Quality of Service (QoS) i.e guarantee in minimum bandwidth and 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 have reported significant disparity in end-to-end link delay between different parts of the world[8]. According to that report the minimum RTT between United States and East-Asian or some African countries ranges between 250ms to 400ms. The minimum RTT between United States and most European or Australasian countries varies between 100 to 250ms.But in some other African countries, where satellite link is 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-Asian/African Countries.

eResearch infrastructures can benefit from traditional group comunications in networks (like multicast) to allow efficient delivery to large numbers of clients (see Figure 1).

But, in contrast to traditional client-server based live streaming methods, eResearch community needs 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

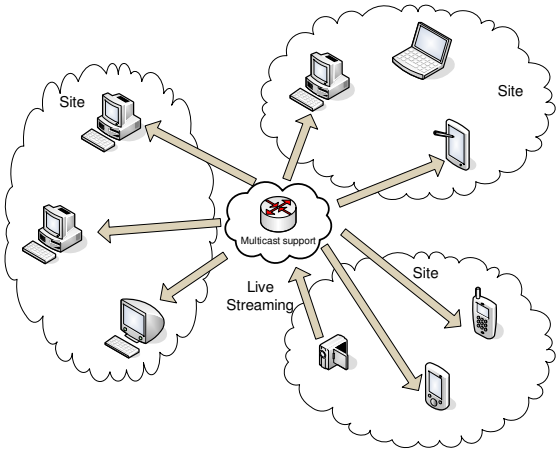

Figure 1: Multicast supported e-Research scenario

ubiquitous computing devices like PDAs and handheld mobile devices connected through different access links having different QoS. In many cases the ubiquitous computing devices used in eResearch may have limited memory and processing power. As a result the traditional streaming techniques which require significant computational ressources may appear to be insufficient in many occasions. A suitable streaming framework for eResearch scenarios must support devices with limited memory and processing resources.

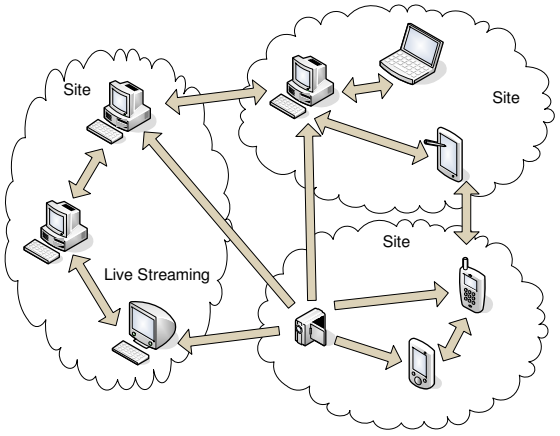

Figure 2: Point-to-Point eResearch scenario

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 and thus protecting 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, as the inter-link delay/loss increases the amount of required playout buffer also increases. For live streaming the buffering adds delay in stream playback and this delay is unacceptable beyond a certain range.

Live streaming applications can tolerate 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% live streaming sessions on the Internet[9]. TCPs congestion control mechanism pro-actively 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)[7] which decouples reliability from congestion control and incorporates TCP-Friendly Rate Control(TFRC)[10] 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 eReseacrh 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 threshhold. Using network simulator *ns-2*[1] 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. Some related work is discussed in section 5 and section 6 contains the future work and conclusion.

2 Background

Classical streaming applications supports multiple level of streamed bit-rate in order to match the available bandwidth with the streamed bit-rate.

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 some 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 full upto a certain threshold.

A streaming server goes through three phases:

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

Streaming client and server sit on a control loop to monitor the packet loss rate and client side buffers status. Whenever the packet loss rate crosses a pre-defined threshold or a re-buffering event occurs, depending on the available bandwidth the streaming server might change streamed bit-rate and starts streaming at the new rate. As shown in figure 3 this

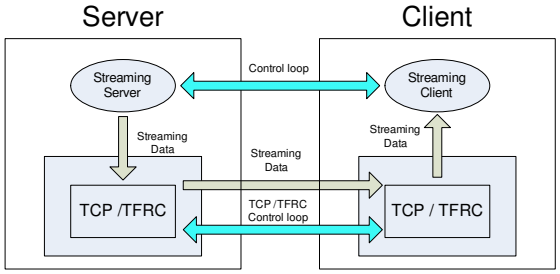

Figure 3: Classical streaming architecture based on TCP / TFRC

loop is decoupled from the rate 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 re-buffering event, packet loss rate, smoothness in achieved throughput of streamed flows.

- *Number of Packet lost in Burst:* Streaming audio-video applications are able tolerate few packet losses but streaming performance degrades if packets are lost in burst.
- *Number of Re-buffering Events:* Every time the playout buffer gets drained below the threshold playback is paused until the buffer becomes full again. This abrupt interruptions in playback drastically impacts the quality of the streaming session. The number of re-buffering 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 to a high streamed bit rate for long time the streamed content is of high quality and this improves the users perception of streaming.

A streaming application is able to reduce the number of packet loss and/or the possibility of re-buffering events by quickly adjusting the sending rate. The role of the underlying transport protocol is utmost important for such an application. A send buffer is required to deal with the rate mismatch between the applications sending rate and the transport protocols 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. When the applications packet generation rate is less than the transmission rate of the underlying transport protocol packets are queued at the send buffer. But as the feedback delay between the client and server increases the client's feedback becomes outdated and the application level rate adaptation mechanism is unable to react soon enough to reduce packet loss and re-buffering 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[11] mechanism is vital for the scalability of the Internet. 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 congestion window.

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

- *Information Hiding :* TCP hides the loss rate and RTT information from the application.
- *Delay Buffering :* 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 Controling :* TCPs AIMD cuts down the applications 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 Introducing :* Whenever a packet loss is detected TCPs 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, as the link delay increases the buffering becomes insufficient to reduce the effects of rate variation. For live streaming it is challenging to stream on TCP if the link-delay and/or loss rate is comparatively significant.

2.3.2 TFRC

TCP-Friendly Rate Control(TFRC)[10] is a rate control algorithm which provides smoother throughput by reacting slowly on packet loss rate while being friendly to other TCP flows. Since most applications on the Internet are TCP based, 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[16] based on receiver's feedback on loss event 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)[7]. DCCP provides an unreliable service with reliable connection establishment and option negotiation states. Applications using DCCP has the option to choose different congestion control mechanism for each direction. Right now only two types of congestion control has been standardized, TCP-like and TFRC.

Due to the smoother rate control TFRC requires less playout buffer space than TCP. But various studies have reported poor performance of TFRC based audio-video transmission. For streaming application even though TFRC reacts slowly on congestion events

the sender can only reacts based on receivers feedback. An early feedback on congestion events will give more time to the sender for rate adaptation.

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 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 upon 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.

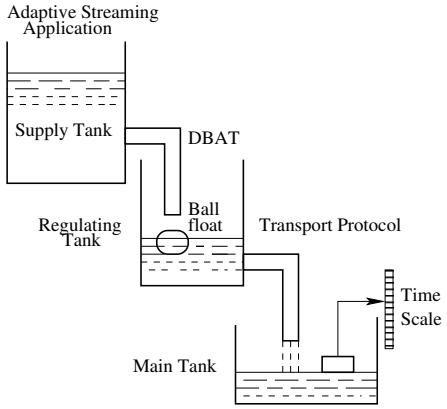

Figure 4: Water Clock Model

The idea behind DBAT can be easily understood by looking at the water clock model as shown in the figure 4[5]. 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. When 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 applications control loop with the transport protocols control loop. Upon connection establishment the application informs the transport protocol about its streamed bit-rate and the transport protocol tries its best to sustain that rate. It is noted that streaming applications are data limited and as such cannot grab the available capacity. Knowing the applications desired rate, the transport protocol limits its sending rate up to a certain range.

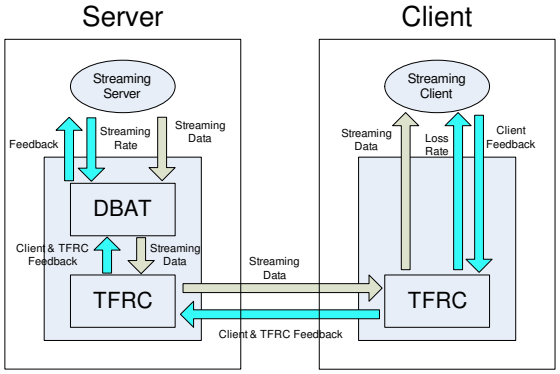

Figure 5: DBAT Architecture

DBAT monitors the send buffer queue size and sends upcall 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.

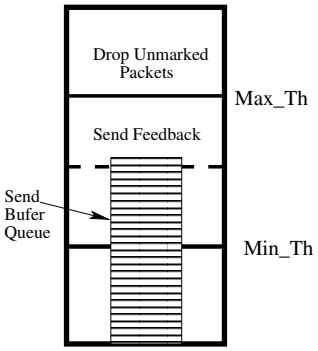

Figure 6: DBAT Queue

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 a feedback is sent to the sender. The number of feedback is limited to at most one per RTT. If *Qavg* grows beyond the maximum threshold then only marked packets are transmitted. The algorithm is quickly presentd in Figure 7.

4 Experimental Results

In this section we present the experimental results for different representative eResearch scenarios. We use network simulator, *ns-2* [1] as our preferred vehicle for simulation. Currently the standard *ns-2* distribution does not have any streaming module included. However we found a streaming module named *Goddard* [12] which is suitable for our experiments. We integrated Goddard in *ns-2* and conducted experiments using it. *Goddard* is based on the behaviors of Real Networks and Windows Streaming Media[6]. During streaming Goddard client and server re-select

```
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
if last_feedback_time > RTT
send feedback
```

Figure 7: DBAT Algorithm

the streamed bit-rate in response to network packet loss or re-buffering events that occur when the client playout buffer get emptied. *Goddard* server supports multi bit-rate streaming. For ease of simulation we only 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 Goddard client and server to exchange media frames. By default the *ns-2* implementation of TFRC has a 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 adjustable send buffer size. To the best of our knowledge we are the first to conduct experiments involving the interaction of streaming application with TFRC in *ns-2*.

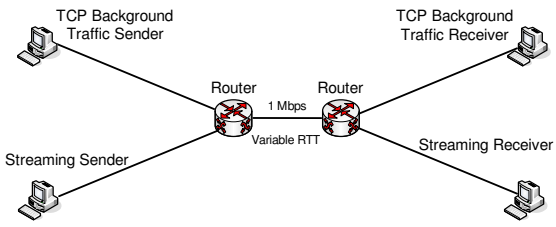

Figure 8: Live streaming simulation topology

We use the dumbbell topology for simulating Point-to-Point live streaming in eReseach sessions(Figure 8). Depending on the scenario, we set the link delay. But in all cases our streaming flow competes with a TCP flow and 20 short TCP flows representing FTP and the web traffic respectively. The FTP flow starts at 0.1 sec and stops at 200the

The bottleneck link is 1 Mbps and we vary the link delay depending 200ms delay. The various. In all cases one streaming flow is competing with a background FTP flow and HTTP flows. The HTTP traffic is generated using empirical data provided by *ns*. The FTP application starts at 0.1 seconds 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 only.

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

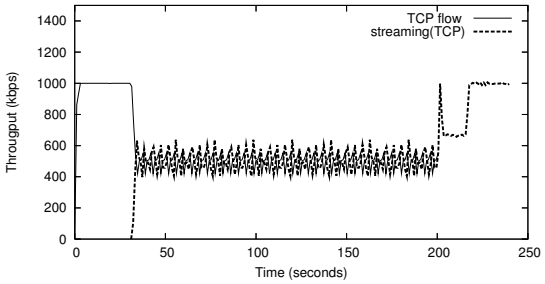

Figure 9: Streaming throughput with 20ms RTT

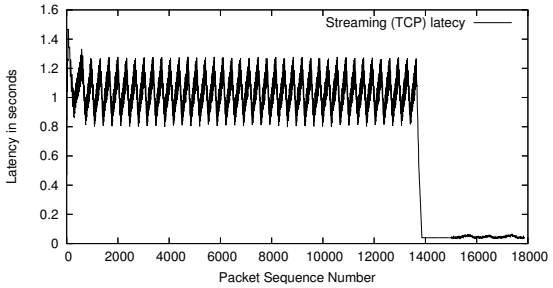

Figure 10: Streaming latency with 20ms RTT

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

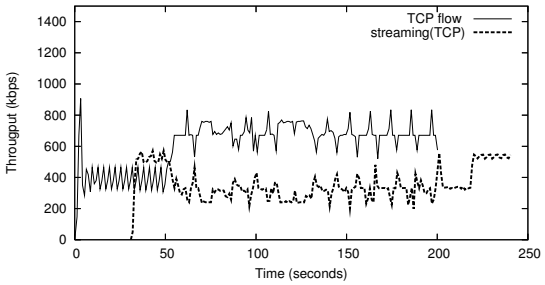

Figure 11: Streaming throughput with 150ms RTT

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

5 Related Work

In an experimental study Balan et al.[18] reports that voice quality is not improved when TFRC is used for rate control. Wang et al.[19] 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.[9] presents 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

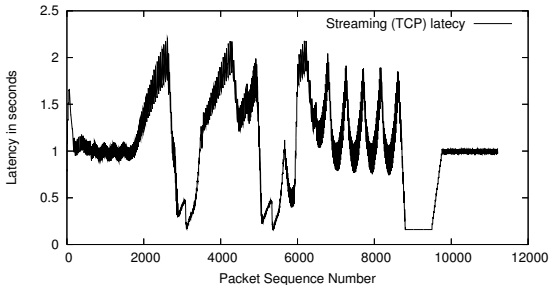

Figure 12: Streaming latency with 150ms RTT

bit-rate stream was around 4 seconds and proposes coordinated streaming, a mechanism that coordinates client side buffering and rate adaptaton to reduce the stream switching delay. Krasic et al.[14] presents a framework for adaptive video streaming based on priority dropping. Chung et al.[12] developed a transport level protocol named Media Transport Protocol(MTP) which removes the burden of in order delivery from TCP. Goel et al.[4] 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 Future Work and Conclusion

Our next steps will concern the exploring of router assistance approach (like XCP[13] needs in terms of fairness with TCP[15]) and investigate the impact on DBAT in terms of responsiveness and reactivity. Morevoer, we will focus on live streaming in group communication (multicast support, see Figure 1) and solve some scalability issues (in terms of memory / buffer usage) by deploying DBAT solutions in clients.

References

[1] ns-2 network simulator.

[2] Sakai project. http://www.sakaiproject.org/.

[3] T. Anderson and H. Kanuka. *E-research: Methods, Strategies, and Issues*, chapter 6, page 73. Allyn and Bacon, 2003.

[4] Kang Li Ashvin Goel, Charles Krasic and Jonathon Walpole. Supporting low latency tcp-based media streams. Technical report, Worcester Polytechnic Institute, May 2002.

[5] D. Bertsekas and R. Gallager. *Data networks*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1987.

[6] Jae Chung and Mark Claypool. Empirical evaluation of the congestion responsiveness of realplayer video streams. *Kluwer Multimedia Tools and Applications*, 2006.

[7] Mark Handley Eddie Kohler and Sally Floyd. Designing dccp: congestion control without reliability. In *ACM SIGCOMM 2006*, pages 27–38, Pisa, Italy, 2006.

[8] ICFA-SCIC Monitoring Working Group. Icfascic network monitoring report. Technical report, International Committee for Future Accelerators (ICFA) - Standing Committee on Inter-Regional Connectivity (SCIC), 2007.

[9] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, Oliver Spatscheck, and Xiaodong Zhang. Delving into internet streaming media delivery: A quality and resource utilization perspective. In *ACM Internet Measurement Conference(IMC)*, Rio de Janeiro,Brazil, October 2006.

[10] Mark Handley, Sally Floyd, Jitendra Padhye, and Joerg C. Widmer. Tcp friendly rate control (tfrc): Protocol specification. Internet Engineering Task Force,RFC 3448, January 2003.

[11] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM*, pages 314–329, Stanford, California, United States., 1988.

[12] Mark Claypool Jae Chung and Robert Kinicki. Mtp: A streaming-friendly transport protocol. Technical report, Technical Report Oregon Graduate Institute School of Science and Engineering, 2002.

[13] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *ACM SIGCOMM*, 2002.

[14] Charles Krasic, Jonathan Walpole, and Wu-chi Feng. Quality-adaptive media streaming by priority drop. In *13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2003.

[15] Dino M. Lopez Pacheco, Laurent Lefevre, and Cong-Duc Pham. 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, May 2007.

[16] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling tcp throughput: a simple model and its empirical validation. In *ACM SIGCOMM*, pages 303–314, 1998.

[17] M. Paterson, D. Lindsay, A. Monotti, and A. Chin. Dart: a new missile in australia's e-research strategy. *Online Information Review*, 31(2):116–134, 2007.

[18] Saverio Niccolini Vlad Balan, Lars Eggert and Marcus Brunner. An experimental evaluation of voice quality over the datagram congestion control protocol. In *IEEE Infocom*, Anchorage, AL, USA, May 2007.

[19] Bing Wang, Jim Kurose, Prashant Shenoy, and Don Towsley. Streaming via tcp: An analytic performance study. In *ACM Multimedia*, New York City, NY, October 2004.