Proceedings of the 12th IEEE International Conference on Computer Communications and Networks, ICCCN 2003, Dallas, Texas, October 2003, pp.311-318.

Jitter and Delay Reduction for Time Sensitive Elastic Traffic for TCP-interactive based World Wide Video Streaming over ABone

Javed I. Khan and Raid Y. Zaghal
Networking and Media Communications Research Laboratories
Department of Computer Science, Kent State University
233 MSB, Kent, OH 44242
javed|rzaghal@cs.kent.edu

Abstract--**Interactivity in transport protocol can greatly benefit transport friendly applications generating streaming traffic. Recently we have developed the TCP interactive, which can provide event notification to the subscriber of its communication service. This is operationally state equivalent to the conventional TCP except applications can optionally subscribe, receive, and in real-time react to selected local end-point events. This simple extension opens the horizon for a spectrum of smart application level solutions to be realized for many of the current hard problems including congestion control for time sensitive elastic traffic. We have recently implemented and tested the real system on the Active Network (ABone) testbed for video streaming to worldwide sites. In this paper we share the performance of this system and report potential dramatic improvement in time-bounded streaming traffic.**

I. INTRODUCTION

Interlayer interactivity can be an elegant tool in addressing some of today's standing problems in network transport. In every layer, including application, if subscriber layer of a service is made aware of the lower level service related events in many cases it is possible to device good solutions at subscriber layer or above. Unfortunately, current transport protocols do not have this transparency. Though, the original TCP proposal (RFC007, RFC793) did call for interlayer interactivity, but subsequent implementations ignored it. Some forms of callback facility have only been marginally cited in recent literature, unfortunately without giving the issue any in depth technical consideration. A detail survey is available at [7]. It seems that this critical shortcoming at the interface between existing applications and current networks has made solution to some problems remarkably harder. With the advent of advanced applications and their advanced transport needs current transports are increasingly becoming inadequate. In fact some recent attempts suggested recreating new and much complex transports for them. Unfortunately, majority of these at the end enormously increase the network layer complexity. Such complex permanent addition to network or system layer appears questionable. When their complexities are weighted against their general advantage, these approaches do not seem to be gaining any acceptance. Interestingly, much effective solutions are possible if instead just simple *interactvity* is added at network layers. As we will demonstrate at application (or subscriber) layers not only much more sophisticated and targeted solutions can be designed, but also this simple and intuitive approach can drastically reduce the network system layers complexity. To test the idea, we have recently proposed, implemented and released the *TCP Interactive* [4]. We have also demonstrated a coupling formalism called *Transientware* (T-ware), where sophisticated yet disposable transport enhancing components can easily be invoked upon event at application level [11]. This system makes a subset of its internal event states accessible. A particular problem we try is the congestion management and particularly the one for time-sensitive streaming traffic. Most of the network level schemes for congestion control are based on delaying traffic at various network points. The more classical schemes depend on numerous variants of packet dropping in network, prioritization (graceful delay in router buffer) admission control (delaying at network egress points), etc. However, a key aspect to note in all is that they introduce **time distortion** in the transport pathway of the application. Though this is harmless to time insensitive traffic such as email or ftp, but they distort the temporal characteristics of time sensitive traffic such as multimedia streaming or control data. The recent solutions here too are based on complex network or system layer addition (such as [1]). We demonstrate an interactivity based solution to the problem of congestion management for time sensitive elastic traffic. In contrast to network or system layer solutions, the general principle we follow is simple and intuitive. It seems an effective delay conformant solution for time sensitive traffic may be built if the original data volume can be reduced by its originator-- the application. To demonstrate the efficacy of the principle, we have also designed a corresponding advanced video rate transcoder system [5,6] that works in symbiosis with the network. This transcoder actively participates in a custom symbiotic back-off scheme in application layer with deep application level knowledge resulting in much more effective joint quality/delay sensitive communication. The adaptation is applicable for traffic where it is possible to dynamically adjust the data generation rate. We call it ***elastic traffic***. Most perceptual data, such as audio, video streams generally belongs to this traffic class. The resulting scheme is similar in spirit to the TCP friendly approaches. However, there is a fundamental difference in how it is done. The network or system layers remain as simple as possible. The responsibility of the network layer is simply to pass on only selected end-point events to the applications. Since, the solutions are now implemented at application level therefore these can be made

much more sophisticated, and reengineeringable. As, we will show the overall solution is not only intuitive and simple, but also surprisingly effective compared to many other recently proposed schemes, which have involved much more complex system/network layer reorganization. We have tested the entire system network for symbiotic video streaming to worldwide destinations using the Active Network Backbone (ABone) testbed, which has been recently developed with new facilities for automatic deployment of new protocols. It currently has 100+ nodes of active routers. In this paper we report performance of the system on the ABone. The paper is organized as follows; next we show TCP's congestion control internal events and the event model of our TCP-interactive protocol. A complete discussion of TCP-interactive can be found in our technical report [7]. In section 3 we explain the ABone testbed of our experiment, and in section 4 we present the experiment results.

II. THE TCP-INTERACTIVE EVENT MODEL

A. TCP's Congestion Control Internal Events

Table-1 lists six events that internally occur when the TCP invokes a congestion control algorithm. The column labeled **SSCA**, and **FRFR** respectively mean that the event takes place in the Slow Start/Congestion Avoidance algorithm, and in the Fast Retransmit/Fast Recovery algorithm. These events are also presented in Figure-1. The graph given in Figure-1(A) shows the sequence of events of the *SSCA* algorithm and how they affect the effective bandwidth available for TCP. Figure-1(B) shows the same sequence for the *FRFR* algorithm. However, in general design we expect only a subset of the internal events that constitutes a protocol will be of interest to the subscriber application. Only a subset of the internal events is made accessible via the interface. In Table-1 the column (**Sub**) shows the subscribable events in our design. The details of TCP congestion control algorithms can be found in [2] and [3].

B. Event Subscription

Figure-2 shows the conceptual model of our interactive protocol. Once it establishes a TCP connection, the user process starts by binding the TCP kernel with a set of chosen events from Table-1 using a subscription API that extends the standard socket API. Each subscription binds the subscribed event with a predetermined user-supplied *Event-Handler*. The Event-Handler will be invoked as a user process when the subscribed event occurs in the kernel space. Also, the subscription mechanism itself is dynamic; it allows the subscribing process to subscribe to new events or cancel subscription (unsubscribe) to previously subscribed events at any time during the lifetime of the TCP connection.

C. Event Notification

An entity called *Event-Monitor* runs in the TCP-interactive kernel space and monitors all subscribed events for every socket (2). When event `(evt)` occurs in socket

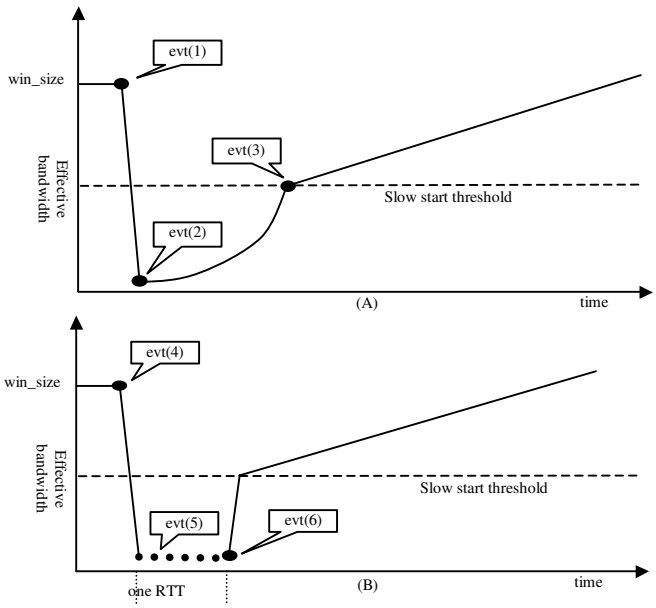

Figure-1. Effective bandwidth changes due to TCP congestion control internal events.

`(sock)`. The Event-Monitor sends a signal to the *Signal-Handler* (3a), and at the same time it writes the socket descriptor of the socket `(sock)` in the process structure of every process that subscribed with this socket. Also, it marks all subscriptions of event `(evt)` in the socket `(sock)` as outstanding and need to be handled (3b).

When it receives a signal, the Signal-Handler uses the probing API to know which socket needs attention. Then, it uses the probing API again to access the socket `(sock)` and get the relevant information about the outstanding subscription of event `(evt)`, which includes the event type, and the name of the Event-Handler (4a,b). Event-Handlers are usually small programs supplied by the user. One Event-Handler is forked by the Signal-Handler per signal to take some action knowing that the event `(evt)` has just occurred in the kernel space (e.g. reduce outbound bit rate to the transport layer).

III. EXPERIMENTAL SETUP

A. The Abone Testbed

The ABone [10], developed under the DARPA Active Network program forms a virtual network infrastructure on which a growing set of active network components can be tested and experimentally deployed. ABone is an operational network and provides an Internet wide network of routing as well as processing capable nodes. Providers can contribute confederation of computing capable nodes. Independent application involving multiple trust domains can be securely launched and executed. For our experiment, we wanted to run our video players and the symbiotic transcoder on an TCP

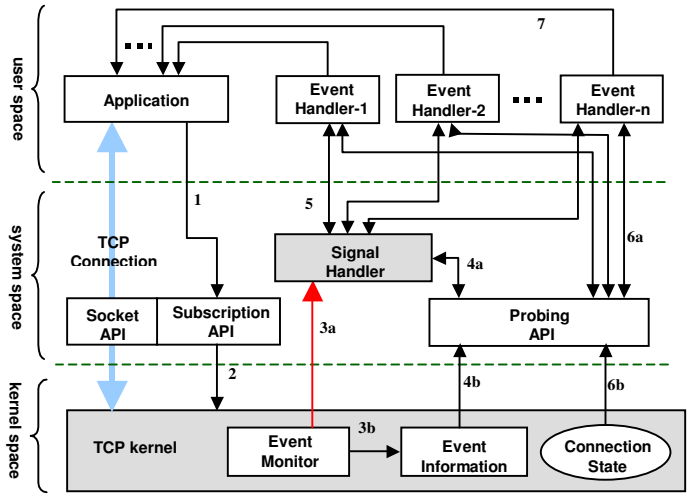

Figure-2. The TCP-interactive extension. The added registration API allows demanding applications to subscribe to events and probe additional event data.

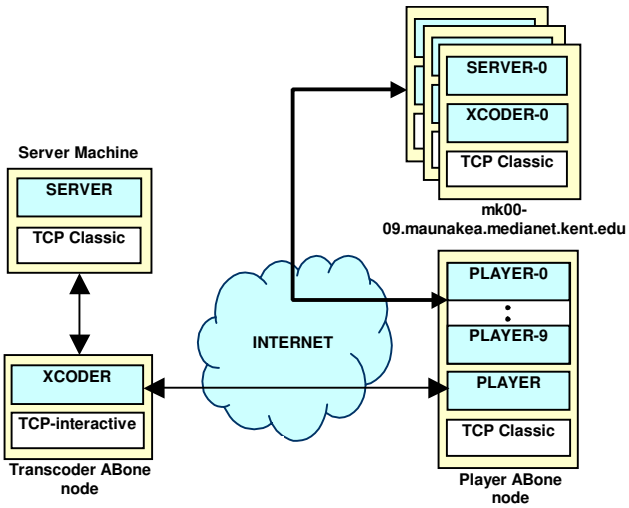

Figure-3. Experiment setup. The transcoder runs on the TCP-interactive ABone machine and the player runs on a remote ABone node. The mk00-09 cluster generates background cross traffic.

Table-1. TCP's Congestion Control Internal Events.

Event	Meaning	Description	SSCA	FRFR	Sub
1	Retransmission timer timed out	Possibly congested network or the segment was lost.	X		X
2	A new ACK was received	Increment snd_cwnd either exponentially (if less than sstheresh) or linearly otherwise.	X		
3	snd_cwnd has reached the slow start threshold ssthresh	Switch incrementing snd_cwnd from exponential to linear.	X		
4	A third duplicate ACK was received	A segment was probably lost, perform fast retransmit.		X	X
5	A fourth (or more) duplicate ACK was received	One segment has left the network; we can transmit a new segment.		X	
6	A new ACK was received	Retransmitted segment has arrived at the destination and all out of order segments buffered at the receiver are acknowledged.		X	X

Table-2. ABone nodes used to run the player in the experiment.

ABone node	IP	Country	Number Of Hubs	RTT (ms)		
				Avg	min	max
abone.fokus.gmd.de	193.175.135.49	Germany	21	144	131	216
galileo.cere.pa.cnr.it	147.163.3.12	Italy	20	287	266	339
abone-01.cs.princeton.edu	128.112.152.62	NJ, USA	15	51	46	69
dad.isi.edu	128.9.160.202	CA, USA	16	65	65	68

Table-3. Event and timing information. Table (A) shows the results of the iTCP runs, while table (B) shows the results of the TCP-classic runs.

ABone node	Num. Of Events	Ref. Jitter	IAT per frame	Flight per frame	Time per video	Frames per Second
fokus (denmark)	2	7.14943	0.11561	0.34110	115.61562	8.70646
galileo (italy)	1.2	7.46529	0.10758	0.27807	107.58549	9.34654
princeton (usa)	1.2	3.51220	0.11010	0.22855	110.16415	9.22516
isi (usa)	1	5.60420	0.10740	0.21521	107.42165	9.37435
AVERAGE	**1.4**	**4.94894**	**0.108413**	**0.25202**	**108.42889**	**9.31169**

(A)

ABone node	Num. Of Events	Ref. Jitter	IAT per frame	Flight per frame	Time per video	Frames per Second
fokus (denmark)	1	18.28416	0.14174	0.33439	141.75998	7.90342
galileo (italy)	2	36.89013	0.15723	0.35755	157.25488	6.55484
princeton (usa)	1.4	6.617037	0.11241	0.23079	112.41928	8.97858
isi (usa)	2.2	10.95021	0.12270	0.30014	122.70665	8.23842
AVERAGE	**1.64**	**15.95234**	**0.12927**	**0.27918**	**129.28553**	**8.12794**

(B)

interactive system from a selected set of ABone nodes and measure the performance of the video session. The ABone provided a convenient testbed for us. We simply sent our video player to one of the ABone's trusted code server at (http://bro.isi.edu/KENT). Then we configured and registered our TCP-interactive machine (*kawai.medianet.kent.edu*) as a primary node on the ABone to run the TCP Intertactive and the symbiotic trascoder. The server remained in a traditional node. The ABone allowed the automatic loading of the sessions on designated machines worldwide. In addition to the TCP-interactive machine we have a cluster of 10 registered ABone nodes at Kent State University (*mk00-mk09.maunakea.medianet.kent.edu*). Four of these nodes run on FreeBSD and the rest run on Linux.

B. Experiment Setup

This experiment describes the performance for the case of a MPEG-2 ISO/IEC13818-2 (176x120) resolution video encoded with base frame rate of 2 Mbps at main profile on the symbiotic transcoder. Figure-3 illustrates the experiment setup. The video server runs on a classic TCP machine (*manoa*) and feeds the video stream into the transcoder, which runs on the TCP-interactive machine (*kawai*). The cluster (*mk00 – mk09*) was used to generate background cross traffic while the video is playing. We run the player on a selected remote ABone node using the Anetd LOAD command from (*kawai*). We repeated the experiment on four ABone nodes, two in the two coasts of the USA and two in Europe. All four nodes are shown in Table-2.

In all runs, the transcoder subscribes with TCP-interactive for two events: REXMT (retransmit timer out) and DUPACK (third duplicate acknowledgment). Also, we always turned on the event notification property of the TCP-interactive. The only controlled parameter that we changed was the reduction property of the signal handler. When the reduction flag was set (*symbiosis=on*), the signal handler invokes the event handler to reduce the bit rate of the decoder.

We repeated the experiment ten times on each remote ABone node from Table-2, five times in the (*symbiosis=on*) mode and five times in the (*symbiosis=off*) mode. In each run we recorded the following information for each frame in the video stream: *frame number, departure time, target bits, actual bits, and SNR values for Y, U, and V blocks.*

III. EXPERIMENT RESULTS

A. Events and Timing

Table-3 shows several parameters to measure end-to-end performance at both the application and the network levels on the four target ABone nodes. Part (a) of the table represents the results for the TCP-interactive mode were symbiosis was applied and part (b) represents the results for the TCP classic mode. Each value in the table is an average of five runs on the specified ABone node. We show six parameters in Table-3: *average number of events, average referential jitter per frame, average inter-arrival time per frame, average flight time per frame, average time to transmit/play the entire video (1000 frames), and average frames per second.* The Observation reveals the advantage of TCP-interactive over TCP-classic. We can see clearly that TCP-interactive managed to substantially reduce the jittery behavior in all four cases and reduce the overall delay of the video session. This improvement was achieved since TCP-interactive managed to contain the buffer buildup and thus speedup the delivery of the frames that suffered from congestion.

B. Delay and Jitter Model

Since the focus of this paper is on real-time streaming traffic, we now define flow characteristics mathematically.

Jitter is introduced into the traffic as a result of unexpected delay experienced by the flow on network routers. Assuming that all packets of the flow have the same size S, and that packet j, $j = 0, 1, 2, \ldots, n$ arrives at the receiver node at time t_j. If the expected arrival time of packet j is e_j we define the referential jitter *RefJitter(j)*, experienced by packet j, $j = 0, 1, 2, \ldots, n$ to be the difference between t_j and e_j.

$refjitter(j) = t_j - e_j$

The value of e_j can be calculated by applying the best-case scenario of the flow assuming perfect throughput of the end-to-end network path and zero packet loss. Average jitter experienced by the flow is calculated by:

$$avgjitter = \frac{1}{n}\sum_{j=1}^{n} refjitter(j)$$

If packet j arrives early then *refjitter(j)* will be negative. When a packet arrives early its contents can be buffered at the receiver. Then the player (or the consumer program at the client node) can fetch each frame when its turn comes for display. When *refjitter(j)* is positive, this means that packet j has arrived late and the player will have to pause while waiting for the requested frame to arrive.

C. Frame Arrival Delay

Now we show the impact of TCP interactivity on frame arrival delay at the remote player. We took frame wise detail event trace of what happens to the first 1000 frames of the video at both sending and receiving ends. For a given discard threshold time in the receiving end we also traced which frame was successfully received or not at the receiving end of the MPEG-2 player. In Figure-4 we plot the delay experienced by the video frames. Each plot (A)-(D) of Figure-4 shows the frame arrival time for one of the target Abone nodes. In each case we show five runs where the symbiosis property of the transcoder is activated (marked as R1-R5, *symbiosis=on*) and five other runs where the symbiosis is turned off (marked as R1-R5, *symbiosis=off*). In the figure we plot the (*symbiosis=on*) or TCP-interactive runs in the shades of red and the (*symbiosis=off*) or TCP-classic runs in the shades of blue.As it can be shown, after each congestion burst, TCP-classic continuously fell behind. The delay built up and hardly it could recover. This is shown by the step-jumps in the delay

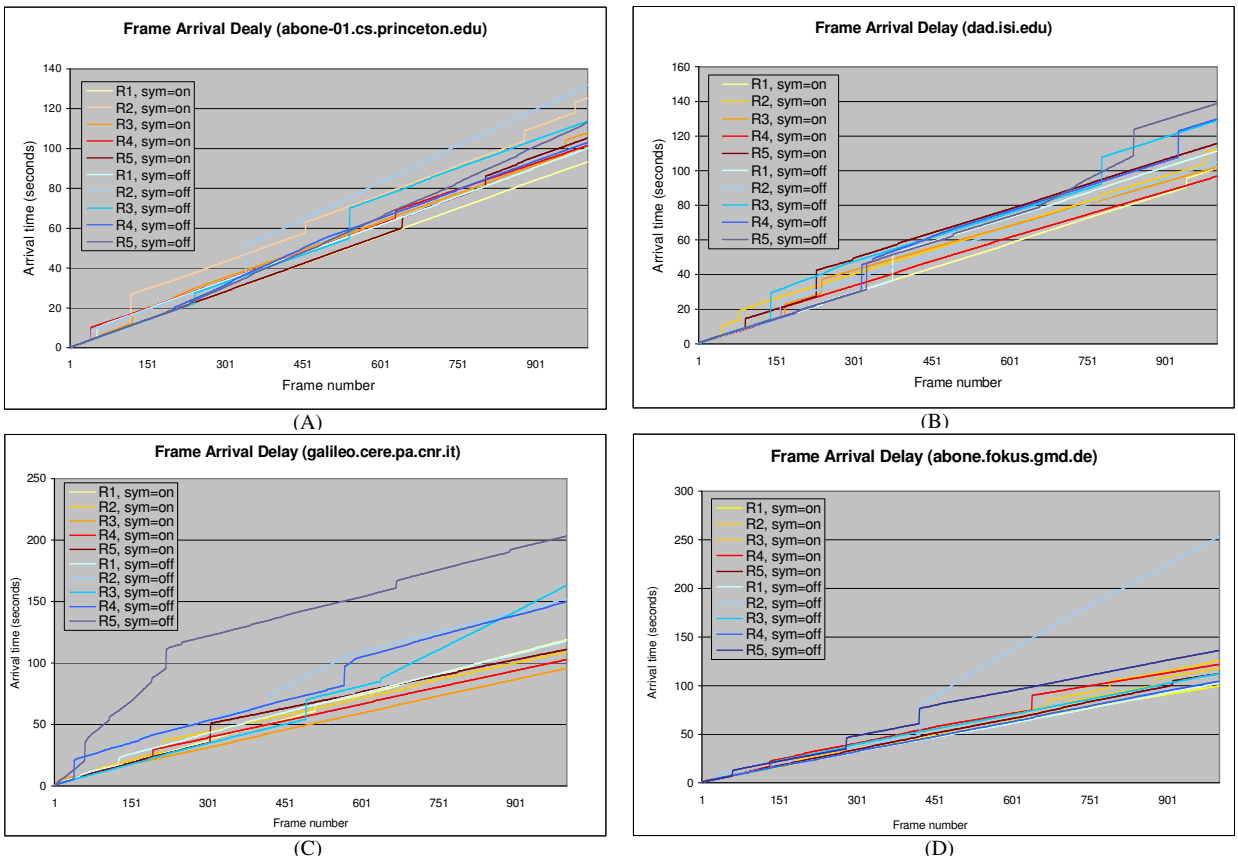

Figure-4. The arrival time of video frames. With each timeout event the backlog increases and can be observed as step jumps in the delay. The TCP-interactive helps in gradually reducing these step-jumps for consecutive loss events.

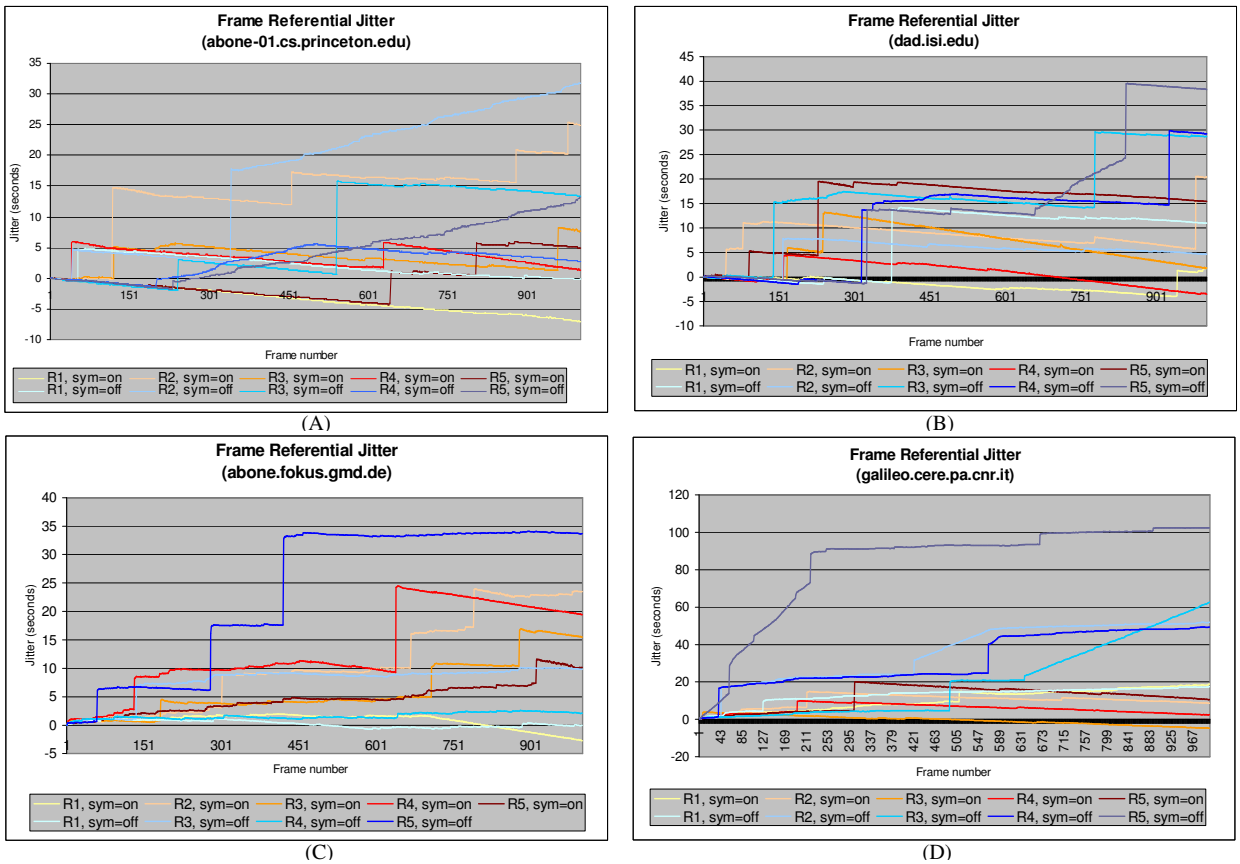

Figure-5. Per frame referential jitter. Negative jitter means that the frame arrived earlier than its ideal time. The classical TCP fell behind with each loss event.

line. The TCP-interactive also suffered some step buildup, but in most cases it was much smaller and it could recover after few seconds. Furthermore, in many runs of the TCP-classic mode, the buildup can be seen as a change in the slope of the line immediately after the step jump (e.g. R2 on '*focus*' and R5 on '*galileo*'). In the TCP-interactive the line always followed the expected trend after the step jump.

D. Referential Jitter

In Figure-5 we plotted the jitter experienced by the frames in the four nodes. We took the difference between the expected ideal arrival time and the actual arrival time for each frame. A negative jitter means the frame arrived earlier than expected. Two things can be noticed in these plots; first, the step jumps in the TCP-interactive runs were generally smaller than those in TCP-classic runs since the frames that faced congestion suffered less delay. This can be clearly seen in plot (A), (C), and (D) of Figure-5. Second, in some cases of the TCP runs, the jittery behavior of the video stream increased immediately after the huge step jump. This can be seen in the plot when the line moves upwards after the step jump and remains in that direction for 50-100 frames period. This behavior is clearly expressed in plot (C), (D), and (E). On the other hand, in all TCP-interactive runs, the line either stayed horizontal, i.e. no change in the jitter after the step jump, or went down as a result of lower jitter. As shown the TCP-interactive drastically reduced the jittery behavior.

III. CONCLUSIONS

In this paper we have presented the case of the 'interactive' generalization of the classical transport control protocol and its runtime performance by running the video session on the global Active Network (ABone) testbed. Though we have illustrated the concept with TCP protocol, but it should bring benefit irrespective of the actual transport service. We believe that interactivity should be considered as a fundamental feature of transport layer design-particularly which targets advanced applications and systems. The interlayer interactivity can enable systematic and modular building of customizable transport services as well as their incremental case-based enhancements without requiring change in lower layers. Thus, indeed it can reduce overall system complexity. Indeed, there is no reason to believe why many of the currently proposed new network or system layer transport services can not be implemented at application level if the interactivity is provisioned. It should also be pointed out that without transport interactivity a network middle layer constructed over the current transports will also face the same fundamental limitations.

Interestingly, the buffer accumulation problem of TCP often forced the use of otherwise less convenient UDP for multimedia. The solution demonstrated here incidentally also overcomes this particular limitation.

The approach presented emphasizes the advantage of network '*friendly*' applications [8]. However, it also departs significantly from the mainstream TCP friendly systems. First, some of the parameters measured (such as RTT) by end-to-end means in application dependent friendliness might be already available (or are being estimated/tracked) at lower layers anyway. The direct protocol interactivity can avoid potential duplication of efforts, and return these parameters faster. Secondly, it does not add any new major component in network software structure as suggested by other network enforced friendliness techniques. Indeed, several proposed approaches in TCP friendliness failed to gain acceptance because of the enormous complexity they poised to introduce at network or system layer.

What are the potential costs? The augmentation of the notification feature clearly increases the normal mode delay slightly. The actual cost depends on the intensity of coupling. However, as shown by the results-- with a prudent design the impact on the network level transfer rate (based on low layer measurement), if any, can be widely surpassed by the gain made at application layer. However, an interesting safeguard of this scheme is that a wrong design will only affect the application at fault and will have no effect on network or others.

The work has been supported by the DARPA Research Grant F30602-99-1-0515.

References

[1] D. Andersen, D. Bansal, D. Curtis, S. Seshan, and H. Balakrishnan, "System Support for Bandwidth Management and Content Adaptation in Internet Applications", Proc. of the Symp. on Operating Systems Design and Implementation, OSDI 2000, Oct. 2000, San Diego, CA.

[2] V. Jacobson, "Congestion Avoidance and Control", Proc. SIGCOMM' 88, Conf, ACM, pp-214-329, 1988.

[3] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm", end2end-interest mailing list, April 30, 1990.

[4] Javed I. Khan, R. Zaghal, and Q. Gu, "Symbiotic Streaming of Time Sensitive Elastic Traffic on an Interactive Transport", The 8th IEEE Symp. on Computers and Communications - ISCC'2003, Turkey, June 2003, pp1435-1440.

[5] Keesman, Gertjan, Hellinghuizen, Robert Hoeksema, Fokke Heideman, Geert, "Transcoding of MPEG bitstreams Signal Processing: Image Communication", Volume: 8, Issue: 6, pp. 481-500, September 1996.

[6] Javed I. Khan, D. Patel, "Extreme Rate Transcoding for Dynamic Video Rate Adaptation", 3rd Int. Conference on Wireless and Optical Communication WOC 2003, Banff, Canada, July 2003, pp410-415.

[7] Javed I. Khan and R. Zaghal, "Event Model and Application Programming Interface of TCP-Interactive", Technical Report TR2003-02-03, Kent State University, [URL http://medianet.kent. edu/technicalreports.html] March 2003.

[8] Dorgham Sisalem, Adam Wolisz, "Towards TCP-Friendly Adaptive Multimedia Applications Based on RTP", Proceedings of the The Fourth IEEE Symposium on Computers and Communications, 1998.

[9] Javed I. Khan, R. Zaghal & Q. Gu, Dynamic QoS Adaptation for Time Sensitive Traffic with Transientware, 3rd Int. Conference on Wireless and Optical Communication WOC 2003, Banff, Canada, July 2003, pp225-229.

[10] Steven Berson, B. Braden, and L. Ricciulli, "Introduction to the ABone", available at http://www.isi.edu/abone /DOCUMENTS/ABarch/, February 2002.

Proceedings of the 12th IEEE International Conference on Computer Communications and Networks, ICCCN 2003, Dallas, Texas, October 2003, pp.311-318.