Technical Report 2003-02-02
Internetworking and Media Communications Research Laboratories
Department of Computer Science, Kent State University
http://medianet.kent.edu/technicalreports.html

Performance Analysis of 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

Last Update: March 2002

Abstract-- *Interactivity in transport protocol can greatly benefit transport friendly applications. We envision a transport mechanism, which is interactive and can provide event notification to the subscriber of its communication service. We then show a friendly adaptive MPEG-2 video transcoding scheme, which directly interacts with the transport protocol and adjusts its production rate whenever a new event is received from the transport layer. We have recently implemented this concept system. The implementation has two components-- an interactive transport protocol over FreeBSD called iTCP and, a novel symbiotic MPEG-2 full logic transcoder. We experimented the real system on the Active Network (ABone) using selected nodes in the U.S. and Europe. In this report we present the performance of this system and report potential dramatic improvement in time-bounded video delivery.*

1. Introduction

Congestion control for time-sensitive multimedia traffic has remained a difficult problem. Most of the mechanisms for congestion control those have been proposed to date 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 of a vast majority of these schemes is that they introduce **time distortion** in the transport pathway of applications. Though time distortion does no harm to time insensitive traffic such as email forwarding or ftp data, but they work completely against the application if the traffic is time sensitive such as multimedia streaming or control data.

For last few years it has been felt that for multimedia applications, the applications themselves have to be more integrated in the solution. Particularly promising are the research in the new TCP friendly paradigm [KeWi00, ReHE00, SiWo98, PrCN00]. [SiWo98] presented a TCP rate-based pacing mechanism that particularly takes note of document transfer characteristics. [ReHE00] discussed a general framework where applications can control rates based on their end-to-end measurements (similar end-to-end technique is used in RealPlayer). There are also fully application level proposals. Due to the lack of convenient means to obtain network states several works suggested [BrGM99, Wolf97] sending multilevel redundant information for video. Also several other works investigated combining application specific information from several streams into one clearinghouse architectures for aggregated congestion control. For example, Congestion Manager [ABCS00, BaRS99] is a system layer component. It provisions aggregated congestion control when multiple streams from the same end-point attempt to send via a separate program called Congestion Manager (CM). [SiWo98] proposed building TCP friendly application where application relies on real-time transport protocol (RTP) mediated end-to-end measurement. CM tries to minimize congestion by coordination between multiple sending streams. [PrCN00] used multiple probing mechanics for aggregate congestion control.

There has been several promising work on network or system level issues to increase TCP friendly-ness. Though, the paradigm of ‘friendly applications’ almost by definition shifts a major part of the congestion management responsibility to the applications, interestingly relatively very few work exists that seriously looked into the corresponding issues that arise in an actual time-sensitive application while taking advantage of the suggested ‘friendliness’. The dynamics of the two systems can lead to stability issues. Time sensitive applications themselves have substantial complexity in adapting. Rate adaptation for any advanced multimedia application in general is quite complex. It requires sophisticated layer 4+ techniques. It is highly unlikely that multimedia rate adaptations for any performance coding schemes (such as MPEG) can be performed at predominantly network or system layer.

It seems that an alternate strategy for time sensitive multimedia traffic should be the multimedia knowledge enriched rate control, which can work in symbiosis with the network condition. We have recently implemented an MPEG-2 ISO-13818-2 [ISO96, KYGP01, KhYa01, KhGu1] video streaming system, and a novel interactive version of TCP called TCP Interactive. 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[1].

However, a key element in any such scheme is that the application must be notified. Unfortunately, today's transport protocols do not support any interactivity with applications. It seems such non-interactivity has been inherited from the early days of networking interface research, when the applications were simple and did not require sophistication. In this report we will show that transport interactivity can bring major benefit to high performance and demanding applications. The particular scheme we propose here has the following novel aspects compared to other recent works:

- First, we suggest an active and direct notification mechanism by the underlying transport protocol, rather than using indirect end-to-end feedback tools. If there is any congestion, we propose an interactive transport protocol, which can directly notify the application.
- To demonstrate the efficacy of the principle, we have designed a corresponding video rate transcoder system that works in symbiosis with the network. This transcoder actively participates in a custom symbiotic *exponential-back-off and additive-increase* like scheme in application layer with deep application level knowledge. (This is also one of the first to our knowledge) resulting in much more effective joint quality/delay sensitive communication.
- The resulting scheme is similar in spirit to the TCP friendly approaches. However, there is a fundamental difference in how it is done. We expect network (or system) layers to remain as simple as possible. The means and techniques for rate reduction remain with the producer application. The responsibility of the network layer is simply to pass on only selected end-point events to the applications.

[1] It is interesting to note, that the idea of application and network symbiosis have been mentioned for quite some time. However, almost no study exists which has focused on it.

```
initially, cwnd = 1 (one segment);
win_size = min (cwnd, snd_wnd);
When congestion occurs:
   ssthresh = max(win_size/2, 2);
   if congestion was due to timeout
      cwnd = 1;
   for every ACK received:
      if (cwnd <= ssthresh)
         /* perform slow start */
         cwnd =  2 * cwnd;
      else
         /* perform congestion avoidance */
         cwnd = cwnd + segment_size;
```

Figure-1. Slow Start/Congestion Avoidance algorithm (SSCA).

As, we will show the scheme is not only intuitive and simple, but also surprisingly effective compared to many other recently proposed schemes, which involve much more complex system/network layer reorganization.

The result presented in this report is not simulation; rather report from a real implementation of the concept system that we have completed very recently. The implementation has two components-- an interactive transport protocol over FreeBSD that we called iTCP and, a novel symbiotic MPEG-2 full logic transcoder [KYGP01, KhYa01], which is capable of working in tandem with the interactive transport. The transcoding model has been developed by closely following the MPEG-2 Test Model 5 (TM5). MPEG-2 TM-5 signifies a real video coder

```
When a 3rd duplicate ACK is received:
   ssthresh = max(2, min(cwnd, snd_wnd)/2);
   Retransmit missing segment;
   cwnd = ssthresh + 3;

Each time another duplicate ACK arrives:
   cwnd = cwnd + 1;
   transmit a new segment;

When a new ACK arrives:
   /* one RTT after retransmission –
      fast recovery */
   cwnd = ssthresh;
```

Figure-2. Fast Retransmit/Fast Recovery algorithm (FRFR).

with substantial complexity of itself. While the detail can be found in [Mpeg00], we describe the salient part of the rate control architecture that is critical to this symbiosis in [KhGR02].
The report is organized in the following way. In the next section, we first provide an overview of the congestion control mechanism in TCP and highlight segment loss events. In section 3 we present our interactive model. Here we explain event subscription, notification, and handling in the iTCP model. Finally, in section 4 we share performance of the scheme on the Active Network (ABone). We experimented the system on several ABone nodes in the U.S. and Europe.

2. Congestion Control in TCP

TCP is a connection-oriented unicast protocol that offers reliable data transfer as well as flow and congestion control. TCP maintains a congestion window that controls the number of outstanding unacknowledged data packets in the network. Sending data consumes slots in the window of the sender and the sender can send packets only as long as free slots are available. When an acknowledgment (ACK) for outstanding packets is received, the window is shifted so that the acknowledged packets leave the window and the same number of free slots becomes available.

2-1. Congestion Control Algorithms

On startup, TCP performs slow-start, during which the rate roughly doubles each roundtrip time to quickly gain its fair share of bandwidth. In steady state, TCP uses an additive increase, multiplicative decrease mechanism (AIMD) to detect additional bandwidth and to react to congestion. When there is no indication of loss, TCP increases the congestion window by one slot per roundtrip time. In case of packet loss indicated by a timeout, the congestion window is reduced to one slot and TCP reenters the slow-start phase. Packet loss indicated by three duplicate ACKs results in a window reduction to half of its previous size. Therefore, the two principal mechanisms that TCP uses to detect network congestion are (i) when the *retransmission* timer times out and (ii) the arrival of duplicate ACKs. Two algorithms then contribute to the TCP congestion control behavior; these are the classic algorithm of slow-start and congestion-avoidance [Jac88], and the augmentation of fast-retransmit and fast-recovery [Jac90]. Figure-1 and Figure-2 below respectively shows the relevant details of the two algorithms.

2-2. Congestion Control Events

Table-1 below lists six events that internally occur when the TCP invokes a congestion control algorithm. Although many other TCP events might occur during a TCP session (e.g., flow control events or connection establishment and termination events), we are only interested in these congestion control events.
In Table-1, the column labeled (**SSCA**) means that the event takes place in the Slow Start/Congestion Avoidance algorithm, and the label (**FRFR**) means that the event takes place in the Fast Retransmit/Fast Recovery algorithm. These events are also presented in Figure-3 below. The graph given in Figure-3(a) shows the sequence of events of the *SSCA* algorithm and how they affect the effective bandwidth available for TCP. Figure-3(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

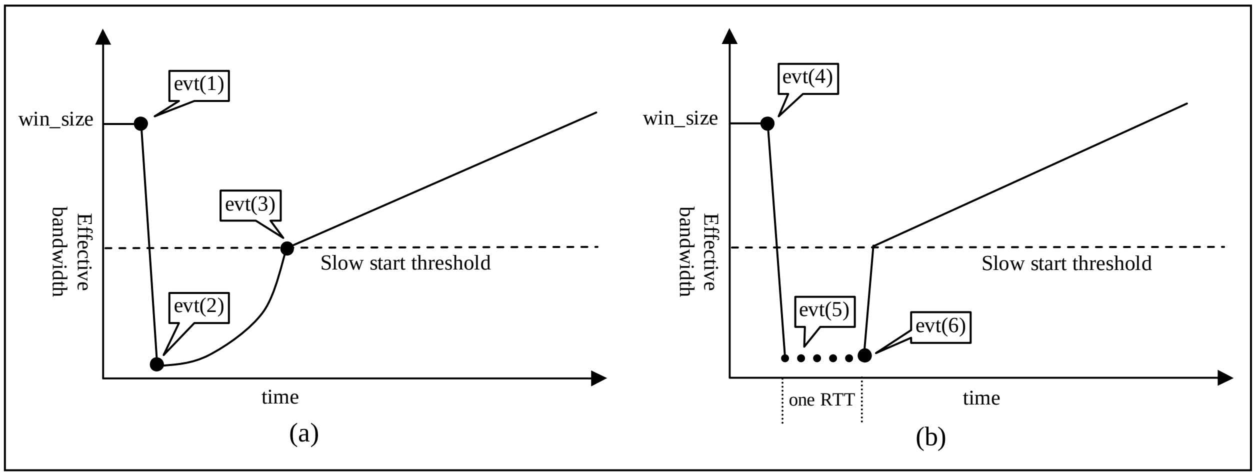


Figure-3. Effective bandwidth changes due to TCP congestion control internal events.

events is made accessible via the interface. An application instance typically subscribes even to a subset of these accessible events. In Table-1 the column (**Sub**) shows the subscribable events in our design.

3. Interactive TCP Model

3-1. Event Subscription

Figure-4 below 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. The model allows one user process to subscribe with multiple sockets and with different set of events per socket. The socket itself can support notification service for multiple subscribing processes. Each individual subscription is handled individually in the socket layer even if the same process made two or more subscriptions.

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.

3-2. Event Notification

The basic scenario of the event notification mechanism proceeds as follows: An entity called *Event-Monitor* runs in the iTCP kernel space and monitors all subscribed events for every socket (2).

Assume at some point event `(evt)` occurs in socket `(sock)`. The Event-Monitor sends a signal to the *Singnal-Handler* (3a), and at the same time it writes the socket descriptor of the socket `(sock)` in the process structure `proc{}` 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 must know first which socket generated the event. To do this, it uses the probing API to read the socket descriptor of the socket `(sock)` from the process structure. Then, it uses the probing API to access the socket `(sock)` and get the relevant information about the outstanding subscription of event `(evt)`. The information retrieved includes the event type and the name of the Event-Handler (4a,b). At this point the Signal-Handler is ready to invoke the appropriate *Event-Handler* (5).

3-3. Event Handling

There are *n* Event-Handlers, enough to satisfy all subscriptions made by the user process. 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

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

layer).
Our probing API allows the Event-Handler to probe additional information about the state of the TCP connection (6a,b). This information includes some parameters from the TCP control block such as: send window size (`snd_wnd`), congestion-control window size (`snd_cwnd`), threshold for slow-start (`snd_ssthresh`), current retransmit value (`t_rxtcur`), and round-trip time (`t_rtttime`). The Event-Handler can use some of these parameters to calculate a new sending rate that guarantees certain delivery time bound of the traffic. The model allows the Event-Handler to probe the Kernel any time to get the updated values of these parameters, and it allows the network administrator to restrict access to kernel data by each Event-Handler.

4. Experiment Results

4-1. Using the Active Network

An important feature of our experiment is using a real implementation of the proposed transport protocol (iTCP) and the MPEG-2 transcoder. Furthermore, we wanted to run the experiment on the real Internet environment. To conduct our experiment we wanted to run our video player on a number of remote hosts around the world and measure performance in each case. We could have done this by "telneting" to those remote nodes. But this would have required preparation and communication with people around the world to setup accounts and administer them. Furthermore, this will not be flexible nor practical if we decide to switch to a new set of remote nodes. Therefore, we decided to use the Active Network.
In Active networks, the routers or switches of the network can perform customized computations on the messages flowing through them. These networks are active in the sense that nodes can perform computations on the contents of the packet. As far as we are concerned, we wanted to be able to run our video player on a selected set of ABone nodes and measure the performance of the video session. In that respect, the Active Network (ABone) provided a convenient testbed for us to run the experiment. We simply sent a modified version of our video player to the ABone administrator at the ABone Coordination Center (ABOCC) to be placed on the trusted code server at (http://bro.isi.edu/KENT). Then we configured and registered our iTCP machine (kawai.medianet.kent.edu) as a primary node on the ABone.
In addition to the iTCP machine we have 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. At the time of our experiment (Feb. 2003) there were 24 Linux nodes, 5 Solaris nodes, and 12 FreeBSD nodes registered on the ABone. Since our player was compiled on Linux, we could use Linux nodes only.

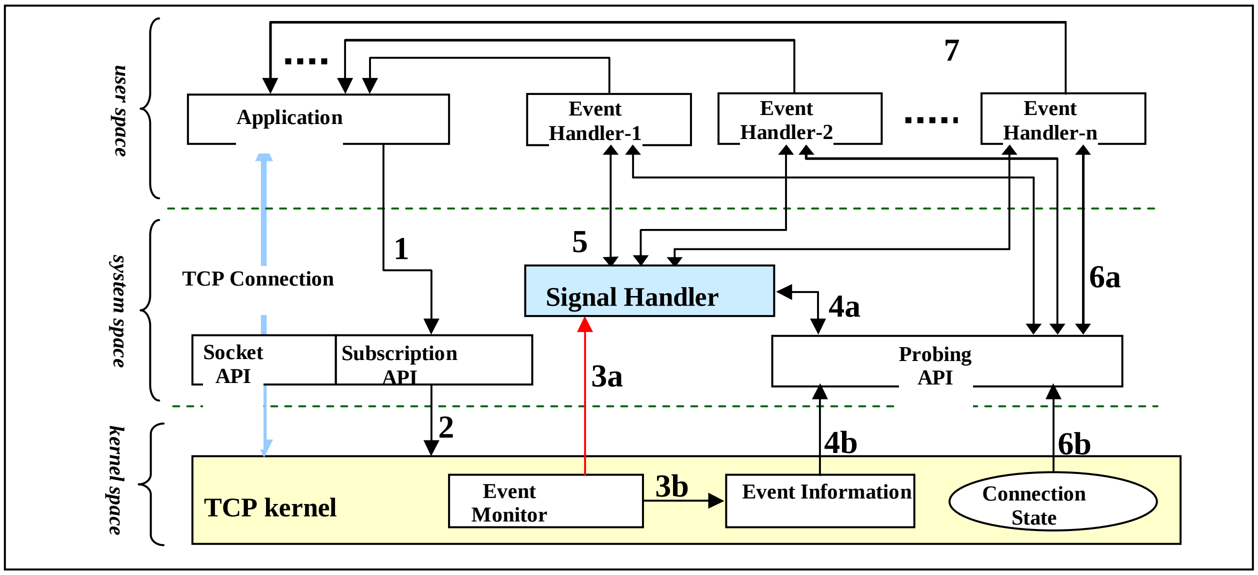


Figure-4. The TCP interactive extension. The added registration API allows demanding applications to subscribe to events and probe additional event data.

4-2. 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-5 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 iTCP machine (kawai). This machine is registered as a primary node on the Active Network (ABone). We also have ten other machines (mk00 – mk09) that are registered on the ABone as well. We used this cluster of ABone nodes 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 five ABone nodes, three in the US and two in Europe. All five nodes are shown in Table-3.

In all runs, the transcoder subscribes with iTCP for two events: REXMT (retransmit timer out event) and DUPACK (third duplicate acknowledgment event) see Table-4. Also, we always turn on the event notification property of the iTCP. 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. Otherwise (when symbiosis=off), the signal handler just records the event type and time in a log file.

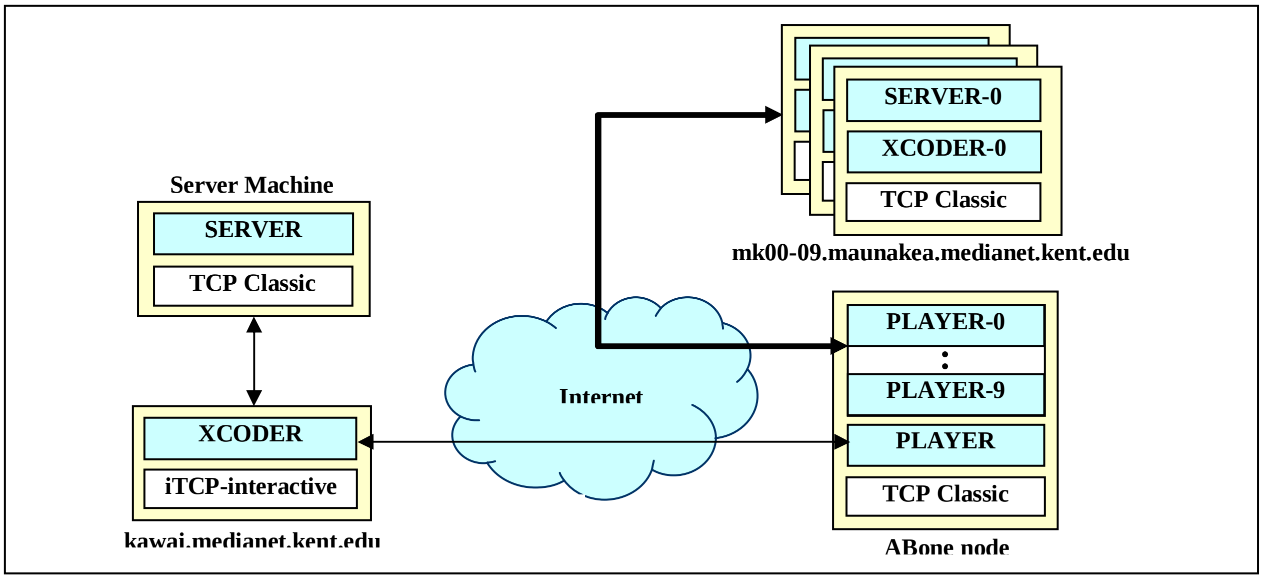


Figure-5. Experiment setup. The transcoder runs on the iTCP machine and the player runs on a remote ABone node. The mk00-09 cluster generates background cross traffic.

We repeated the experiment ten times with each remote ABone node from Table-3, five times in the (symbiosis=on) mode and five times in the (symbiosis=off) mode.

In each run we recorded two log files, one on the transcoder side (kawai), and one on the player side (the remote ABone machine). We retrieved the latter log file using the Anetd GET command. The transcoder 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. Also, when an event signal is received from the iTCP, the signal handler records its type and timing. On the player side, the log file only records the arrival time of each frame.

In the following discussion we will regard the (symbiosis=on) mode to resemble iTCP and the (symbiosis=off) to resemble classic TCP. We made this resemblance since the (symbiosis=off) mode adds only the event notification property to the TCP. This small overhead is irrelevant and can be ignored in the overall system performance analysis.

4-3. Network and Video Parameters

Table-5 shows several parameters to measure end-to-end performance at both the application and network

levels on the five target ABone nodes. Part (a) of the table represents the results for the iTCP 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 eight parameters in Table-5: average number of events, average referential jitter per frame, average inter-arrival time per frame, average SNR(Y) block per frame, average bits per frame, average flight time per frame, average time to transmit/play the entire video (1000 frames), and average frames per second.
To facilitate comparison between the two cases, we converted each parameter from the Table-5 into a bar graph. We show these bar graphs in Figure-6. In each bar graph the x-axis represents the five target ABone nodes and the y-axis represents the measured parameter. First, we notice that the average number of congestion events for both TCP and iTCP modes on all ABone nodes were relatively close (1.4 on iTCP vs. 1.6 on TCP). This observation justifies the comparison and enables us to make the assumption that both modes were running under similar network circumstances. Direct observation of these bar graphs reveals the advantage of the iTCP mode over TCP mode.

4-4. Symbiotic Rate Control and Event Trace

Figure-7 shows the symbiotic frame rate transcoding for five runs on the target ABone nodes. The symbiosis occurred due to the joint rate specification at the rate control logic at the symbiosis unit and in the transcoder. Each case plots the incoming video frame sizes, the target *rate retraction ratio* specified by the symbiosis controller, and the resulting outgoing frame rate generated by the transcoder. Congestion events at the TCP (e.g. timeout event) resulted in the symbiosis unit to modify the rate according to the lazy-binary-back-off rule. A retraction ratio (Alpha) of 0.55 was used. Though, the final generation rate varied widely from frame to frame due to their frame type, but the general trend followed the specified target bit rate.

4-5. MPEG-2 Frame Transport Efficiency: Delay and Jitter

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-8 we plot the delay experienced by the video frames. To avoid any impact of the frame coding and generation delay, we let the transcoder first run in local mode and output its stream to a local disk as soon as it is produced. We then computed the frame delivery time with respect to the frame interval delay in local mode transcoding.
Each part [(a)-(e)] of Figure-8 plots 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, sym=on) and five other runs where the symbiosis is turned off (marked as R1-R5, sym=off). In the figure we plot the (sym=on) or iTCP runs in the shades of red and the (sym=off) or TCP runs in the shades of blue.
As it can be shown after each congestion burst, TCP continuously fell behind. The delay built up and hardly it could recover. This is shown by the step jumps in the delay line. The iTCP also suffered some step buildup, but it in most cases it was much smaller and it could recover after few seconds. Furthermore, in many runs of the TCP 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 iTCP runs the line always followed the expected trend after the step jump.
In Figure-9 we plotted the jitter experienced by the frames in the five 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. As shown the iTCP drastically reduced the jittery behavior.

Subscribe flag (iTCP) = on
Event reception flag (EVENT) = on
Rate reduction flag (SYMBIOSIS) = on/off
Reduction Factor (ALPHA) = 0.55
Subscribed events = REXMT | DUPACK
Frame size = 176 x 120
Number of Frames = 1000

Table-4. Experiment and video parameters. Only the reduction flag (SYMBIOSIS) was changed in different runs.

4-6. Observation at Application Level

In the above discussion we illustrated how the symbiosis mechanism worked from the video transport protocol (MPEG-2) and the network transport protocol (TCP) layers beneath it. In this plot we will illustrate how this mechanism appears from the very top-- in the application layer itself. An application receives and delivers uncompressed frames. The performance metric this end-system uses is the temporal and spatial quality difference between the transmitted and the reproduced uncompressed video frames at both ends. The underlying MPEG-2 transport protocol and the network layer TCP together provides the transport. The specific compression, windowing etc. and other detail mechanisms are external techniques to the end systems.

In Figure-10 each frame is plotted as a point in the video quality/frame delay plane. Each plot in the figure represents the average of five runs on the specified ABone node. As can be seen from the region of the two QoS distributions, in classical TCP (sym=off), although frames have been generated with SNR quality ranging between 13-39 dB, but many of these frames were lost in transport, and were never delivered. In contrast, the proposed iTCP (sym=on) can deliver all the frames guaranteed at 10-38 dB quality. All plots show the Y block quality. Table-6 shows the average for all Y, U, and V blocks. Fundamentally, what TCP interactive has offered is a qualitatively (as opposed to the quantitative improvements offered by any unaware solution) new empowering mechanism, where the catastrophic frame delay can be traded off for acceptable reduction in SNR quality.

5. Conclusions and Current Work

In this report, we have presented a case of rate symbiosis mechanism in line with current advances in TCP friendly systems. We have presented the case through a simple ‘interactive’ generalization of the classical transport control protocol, and a novel implementation of a symbiotic MPEG-2 transcoder. We collected the results of our experiment by running the video session on the global Active Network (ABone) testbed.

Glossary of figures and tables in this report

ABone node	Evt. num	Ref. Jitter	IAT per frame	SNR per frame	Bits per frame	Flight per frame	Time per video	Frames per Second
fokus	2	7.1494305	0.115616	21.942224	49737.387	0.34110329	115.615629	8.70646714
galileo (italy)	1.2	7.46529	0.107585	18.423304	36672.216	0.27807673	107.585495	9.34654553
columbia (usa)	1.6	1.0135707	0.101358	22.51238	50173.551	0.19717850	101.357553	9.90596598
princeton (usa)	1.2	3.5122045	0.110104	22.463844	50331.645	0.22855333	110.164151	9.22516667
isi (usa)	1	5.604205	0.107402	20.712062	47694.519	0.21521991	107.421659	9.37435276
AVERAGE	**1.4**	**4.9489401**	**0.108413**	**21.210763**	**46921.864**	**0.25202635**	**108.428897**	**9.31169962**

(A)

ABone node	Evt. num	Ref. Jitter	IAT per frame	SNR per frame	Bits per frame	Flight per frame	Time per video	Frames per Second
fokus	1	18.284166	0.14174	23.5559	54672.969	0.33439687	141.759976	7.90342341
galileo (italy)	2	36.890137	0.157235	23.5559	54672.969	0.35754888	157.254885	6.55484516
columbia (usa)	1.6	7.0201728	0.112287	23.5559	54672.969	0.17304129	112.286873	8.96444653
princeton (usa)	1.4	6.6170372	0.112419	23.5559	54672.969	0.23079274	112.419281	8.97858152
isi (usa)	2.2	10.950218	0.122707	23.5559	54672.969	0.30013761	122.706648	8.23842417
AVERAGE	**1.64**	**15.952346**	**0.129278**	**23.5559**	**54672.969**	**0.27918348**	**129.285533**	**8.12794416**

(B)

Table-5. Network and video parameters. Table (A) shows the results of the iTCP runs, while table (B) shows the results of the classic TCP runs.

abone7.cs.columbia.edu			
	Y	**U**	**V**
	23.5559	11.42286	11.89164
	22.27272	10.73152	11.17327
	20.91348	9.93609	10.29408
	22.2639	10.79802	11.37637
	19.9173	9.434132	9.625847
Average	**21.78466**	**10.46453**	**10.87224**

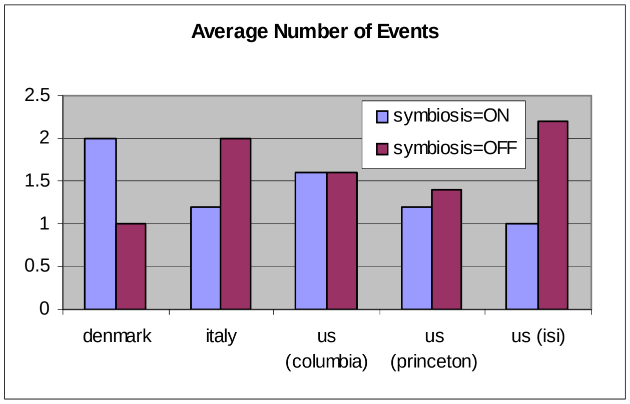
abone.fokus.gmd.de			
	Y	**U**	**V**
	21.7502	10.4313	10.88387
	22.55195	11.00852	11.37276
	22.48072	10.8563	11.33524
	20.32865	9.53193	9.7699
	22.5996	10.98443	11.52106
Average	**21.94222**	**10.5625**	**10.97657**

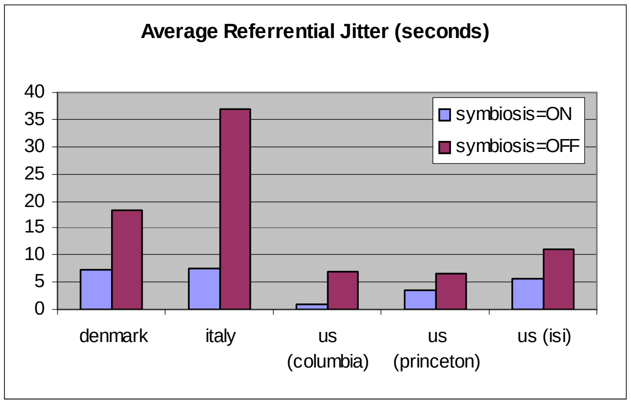
galileo.cere.pa.cnr.it			
	Y	U	**V**
	14.20791	6.309307	6.57389
	18.78038	8.54067	8.73877
	17.55128	7.75856	7.91798
	18.88435	8.58182	8.78481
	22.6926	10.91146	11.37236
Average	**18.4233**	**8.420363**	**8.677562**

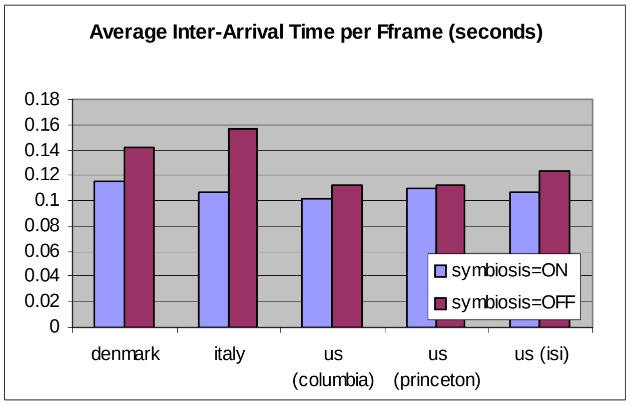
dad.isi.edu			
	Y	**U**	**V**
	22.6875	**10.9431**	**11.38217**
	20.24505	9.4674	9.70093
	14.81236	6.360895	6.513461
	22.2595	10.78451	11.22879
	23.5559	11.42286	11.89164
Average	**20.71206**	**9.795753**	**10.1434**

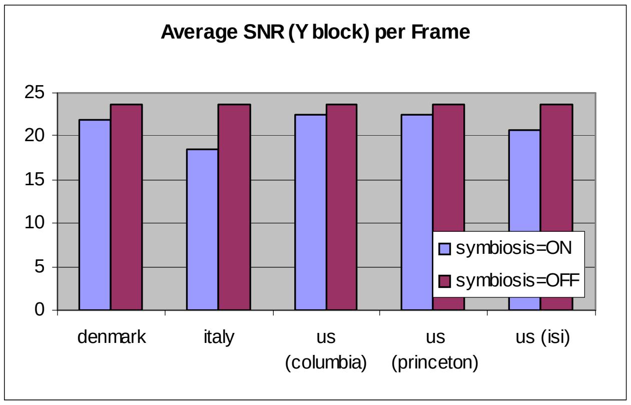
abone-01.cs.princeton.edu			
	Y	U	**V**
	23.5559	11.42286	11.89164
	20.59231	9.73216	10.03474
	23.0545	11.18908	11.68592
	22.24371	10.60695	11.10733
	22.8728	11.05248	11.5538
Average	**22.46384**	**10.80071**	**11.25469**

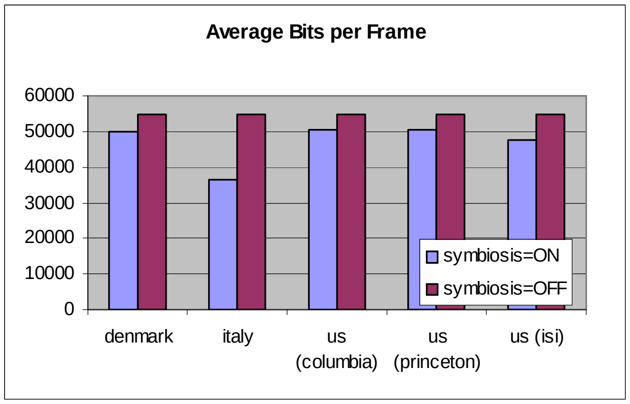
Table-6. The average picture quality for Y, U, and V components on five runs of iTCP.

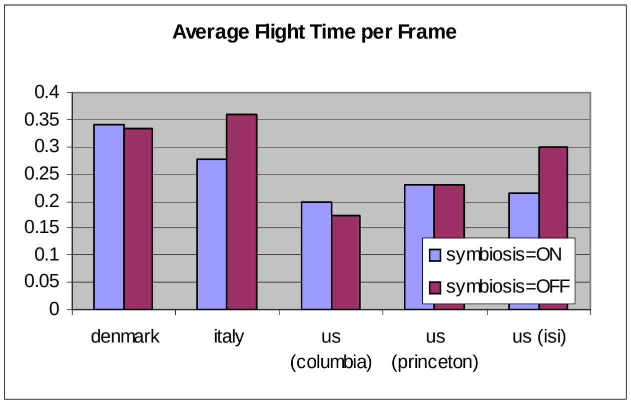

Average Number of Events
2.5
2
1.5
1
0.5
0
symbiosis=ON
symbiosis=OFF
denmark
italy
us (columbia)
us (princeton)
us (isi)

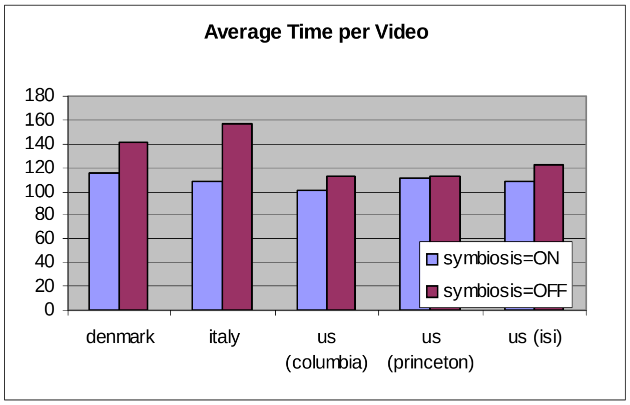

Average Referrential Jitter (seconds)
40
35
30
25
20
15
10
5
0
symbiosis=ON
symbiosis=OFF
denmark
italy
us (columbia)
us (princeton)
us (isi)

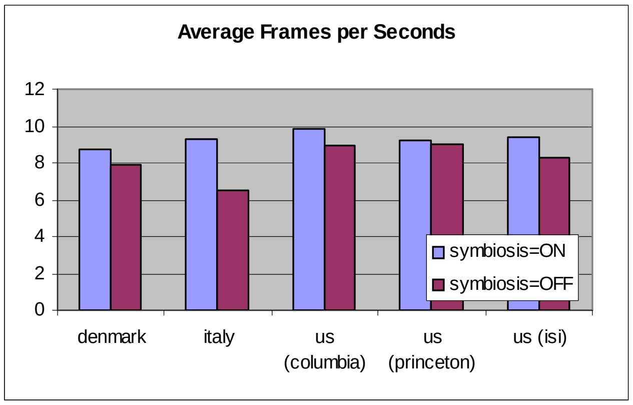

Average Inter-Arrival Time per Fframe (seconds)
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
symbiosis=ON
symbiosis=OFF
denmark
italy
us (columbia)
us (princeton)
us (isi)


Average SNR (Y block) per Frame
25
20
15
10
5
0
symbiosis=ON
symbiosis=OFF
denmark
italy
us (columbia)
us (princeton)
us (isi)


Average Bits per Frame
60000
50000
40000
30000
20000
10000
0
symbiosis=ON
symbiosis=OFF
denmark
italy
us (columbia)
us (princeton)
us (isi)


Average Flight Time per Frame
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
symbiosis=ON
symbiosis=OFF
denmark
italy
us (columbia)
us (princeton)
us (isi)


Average Time per Video
180
160
140
120
100
80
60
40
20
0
symbiosis=ON
symbiosis=OFF
denmark
italy
us (columbia)
us (princeton)
us (isi)


Average Frames per Seconds
12
10
8
6
4
2
0
symbiosis=ON
symbiosis=OFF
denmark
italy
us (columbia)
us (princeton)
us (isi)

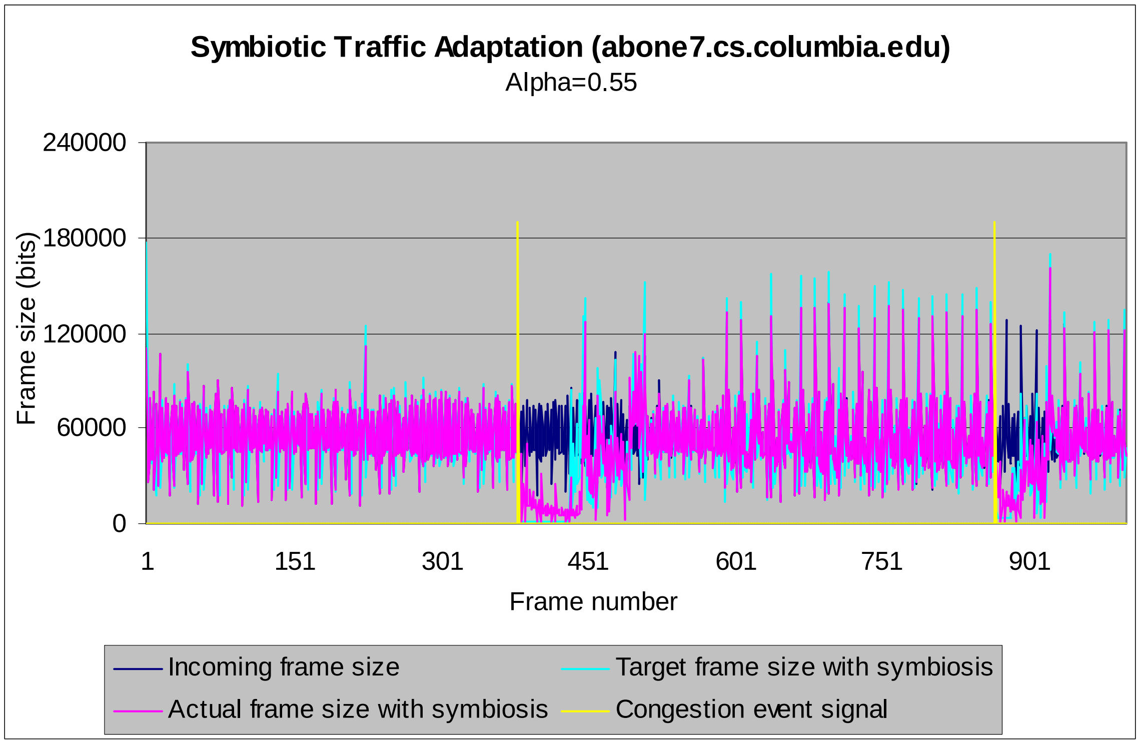


Figure-7. (A)

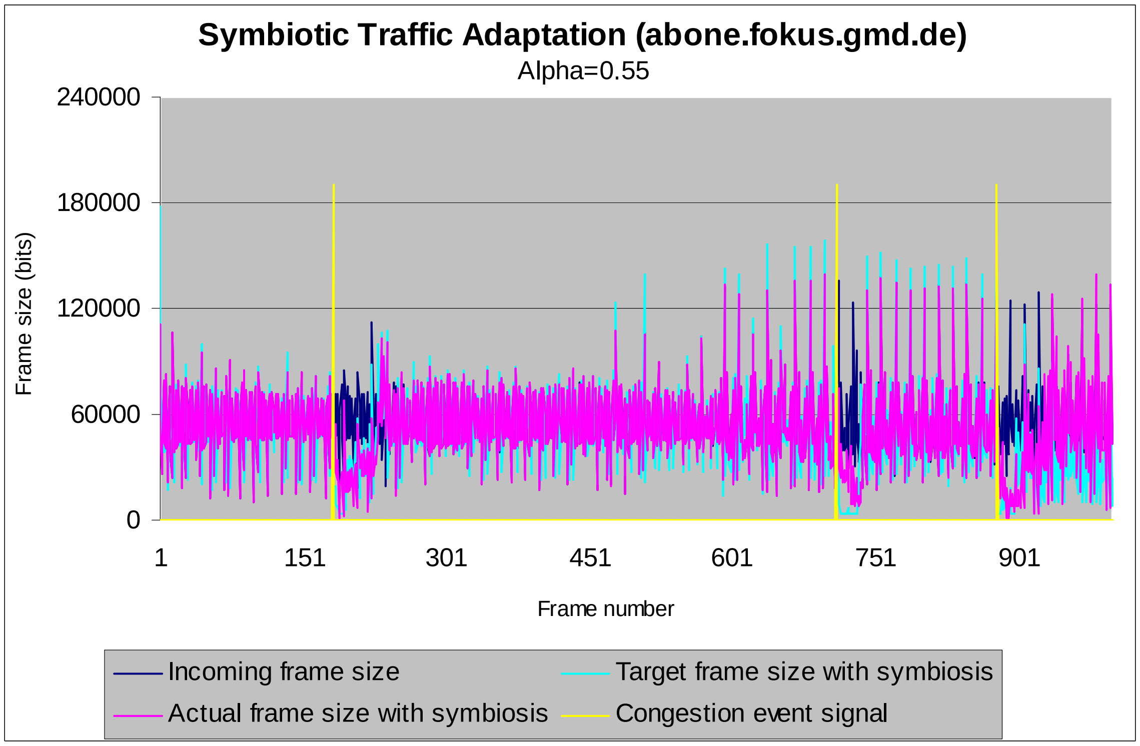


Figure-7. (B)

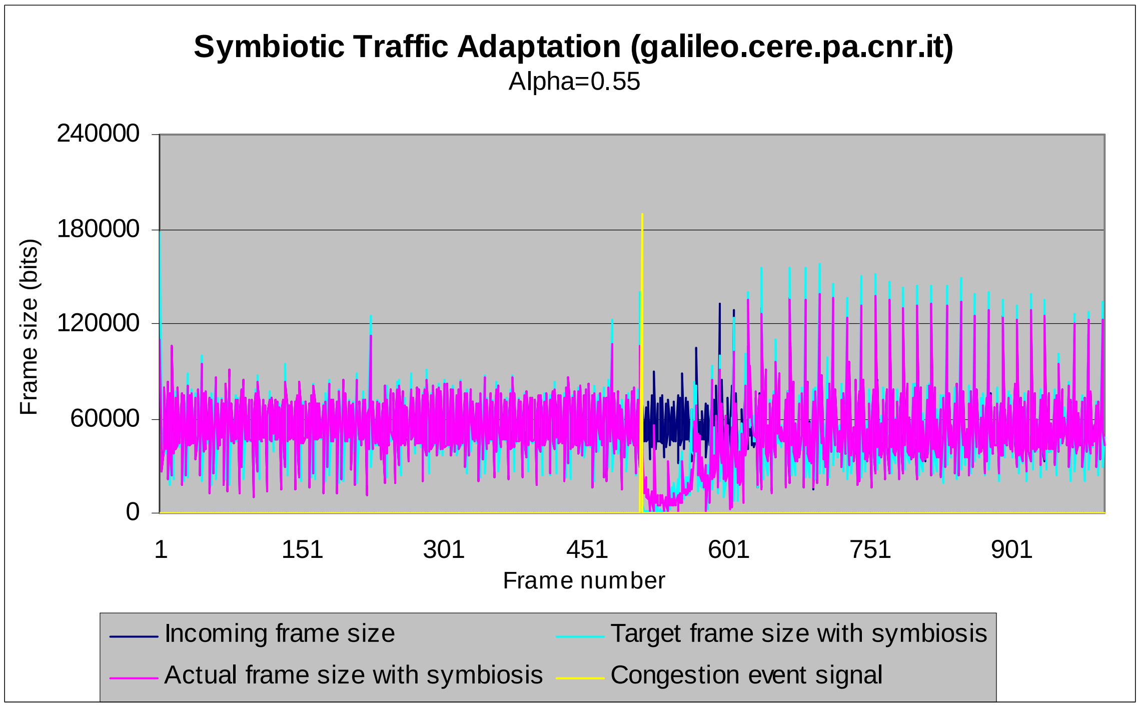


Figure-7. (C)

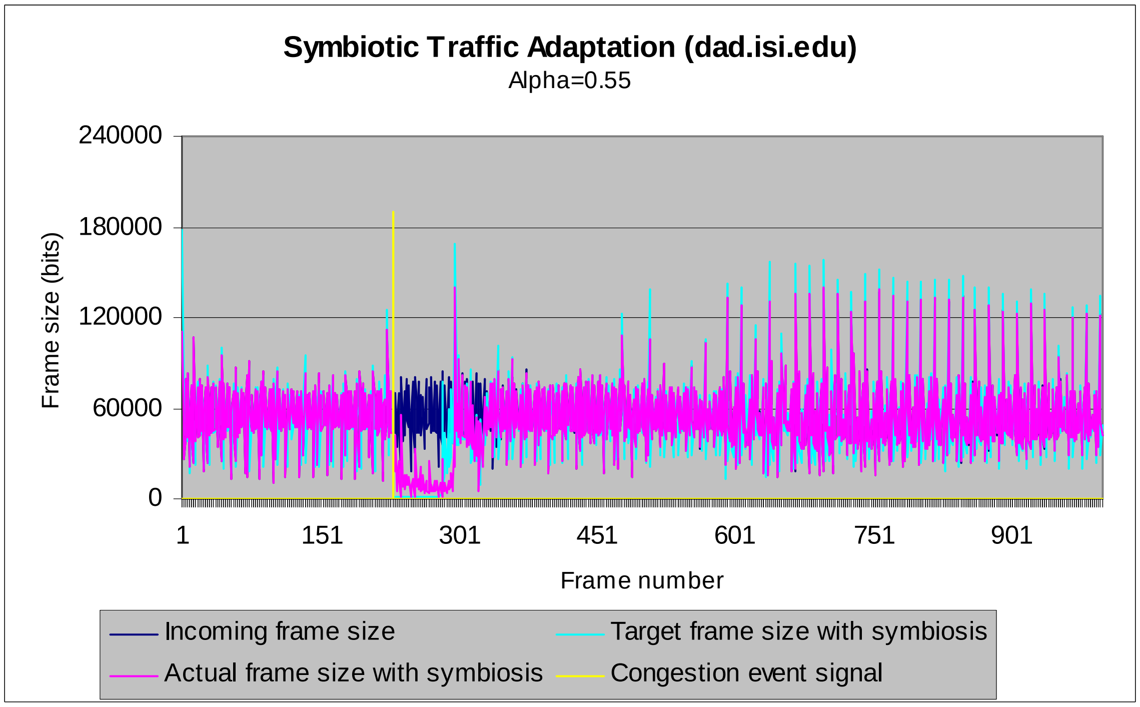


Figure-7. (D)

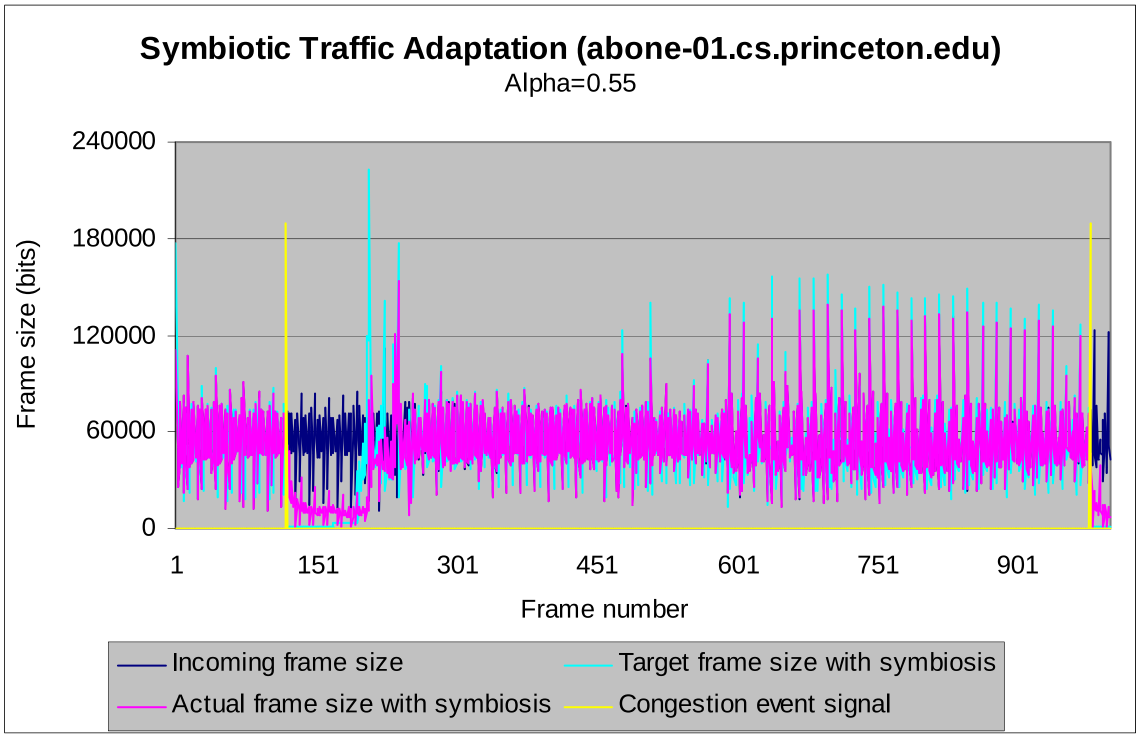


Figure-7. (E)

Figure-7. The binary-back off rate reduction in transcoder. Each figure plots the incoming frame sizes, the event driven target rate (retraction ratio) specified by the symbiosis unit, and the resulting output rate from the transcoder.

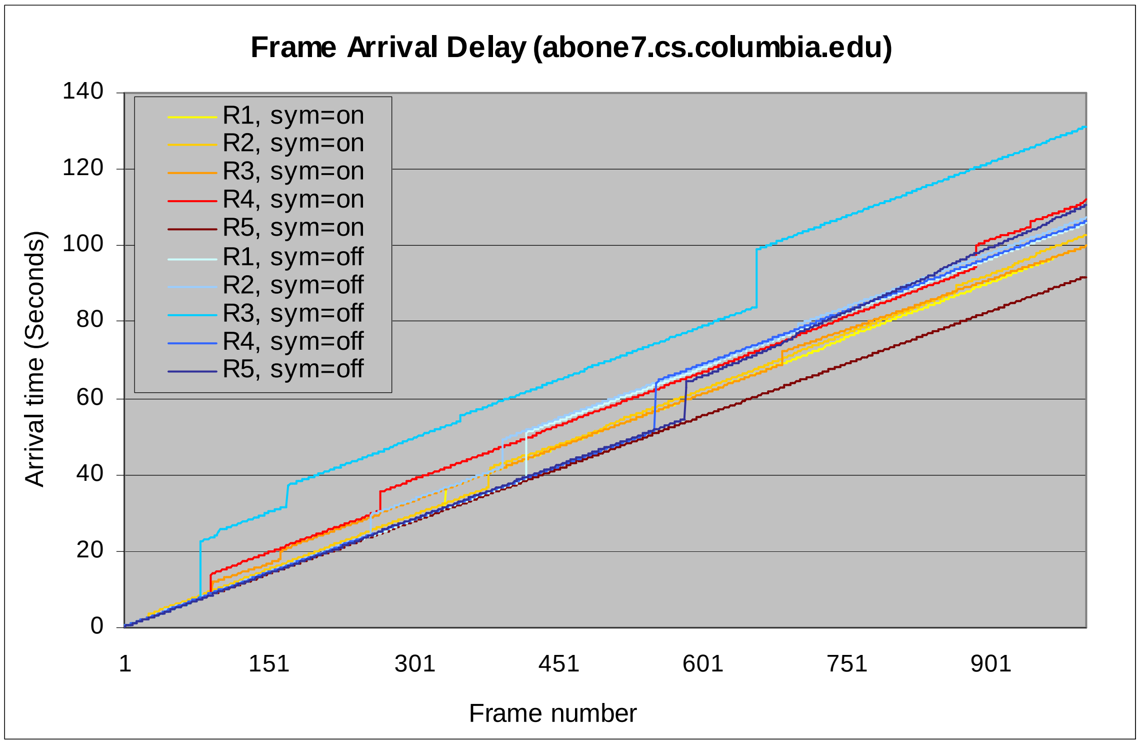


Figure-8. (A)

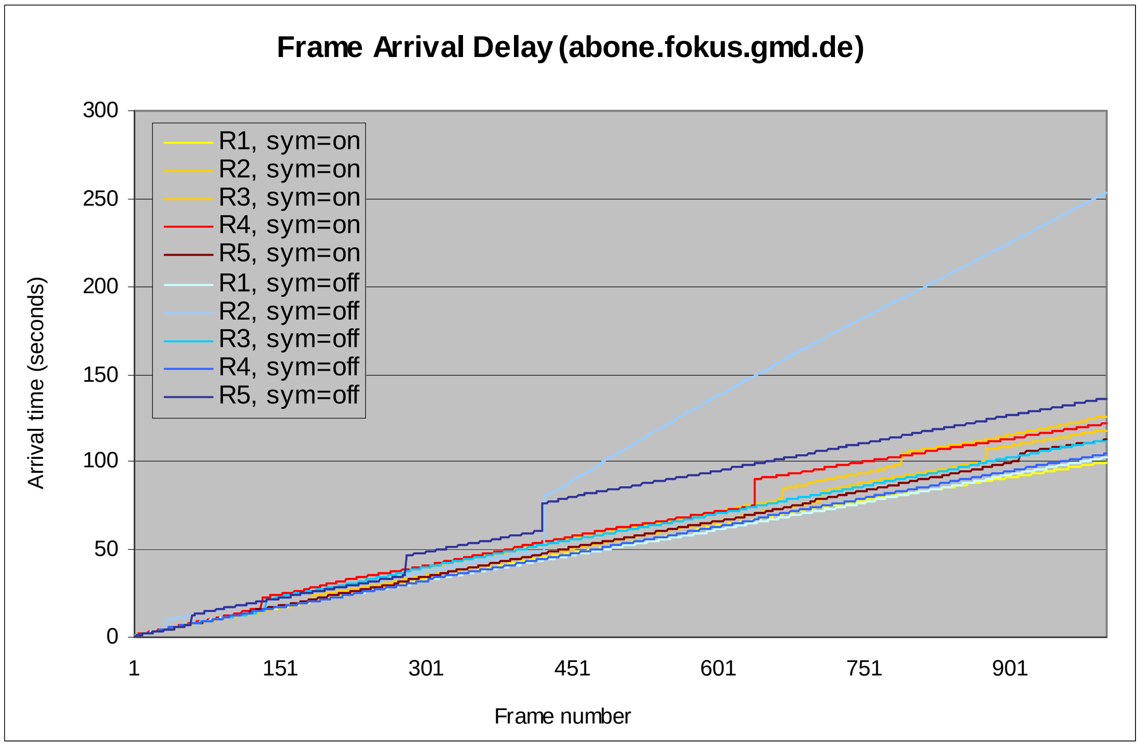


Figure-8. (B)

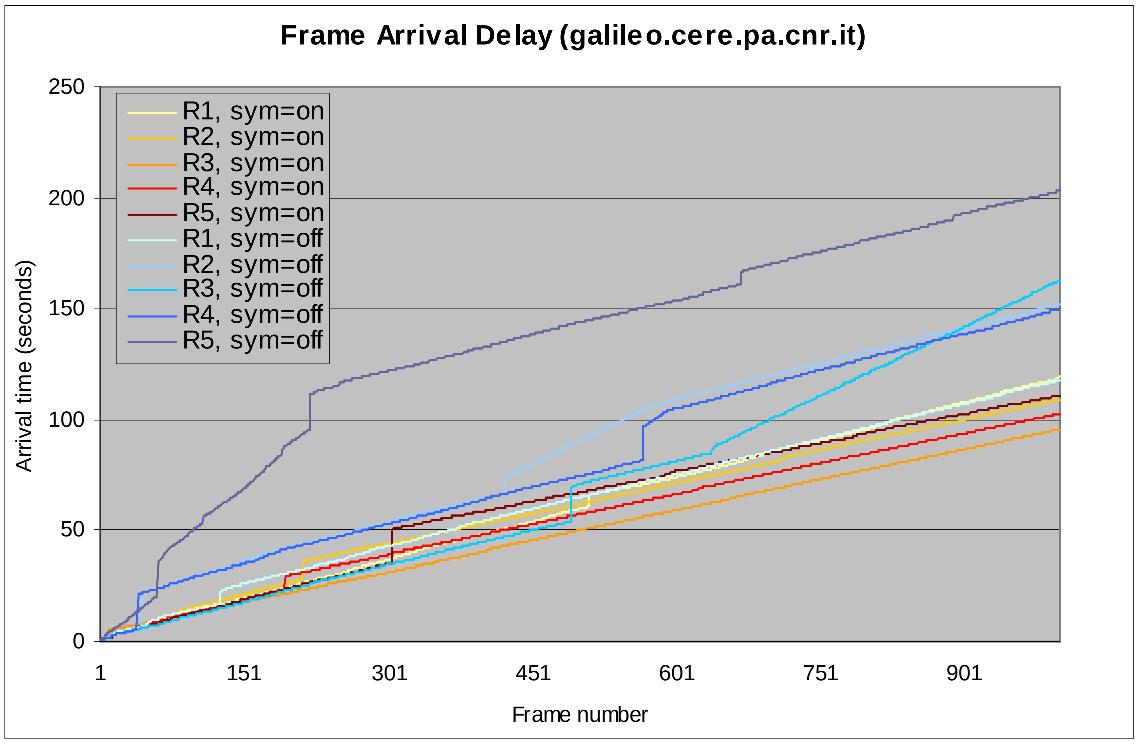


Figure-8. (C)

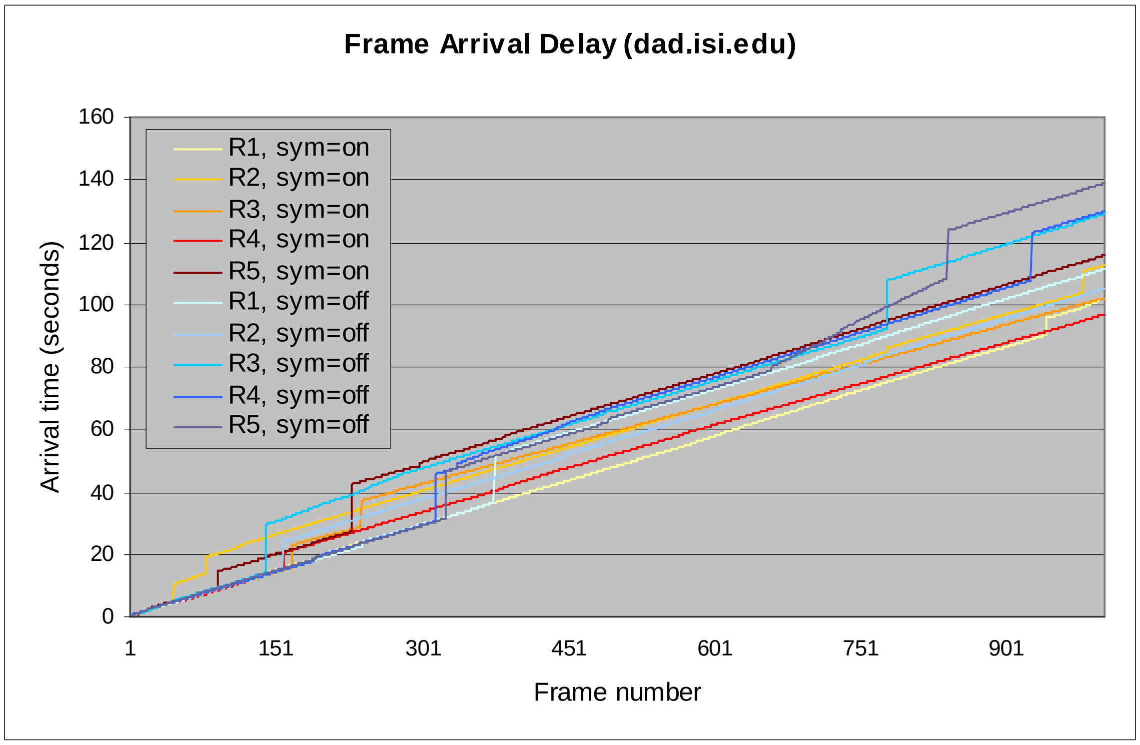


Figure-8. (D)

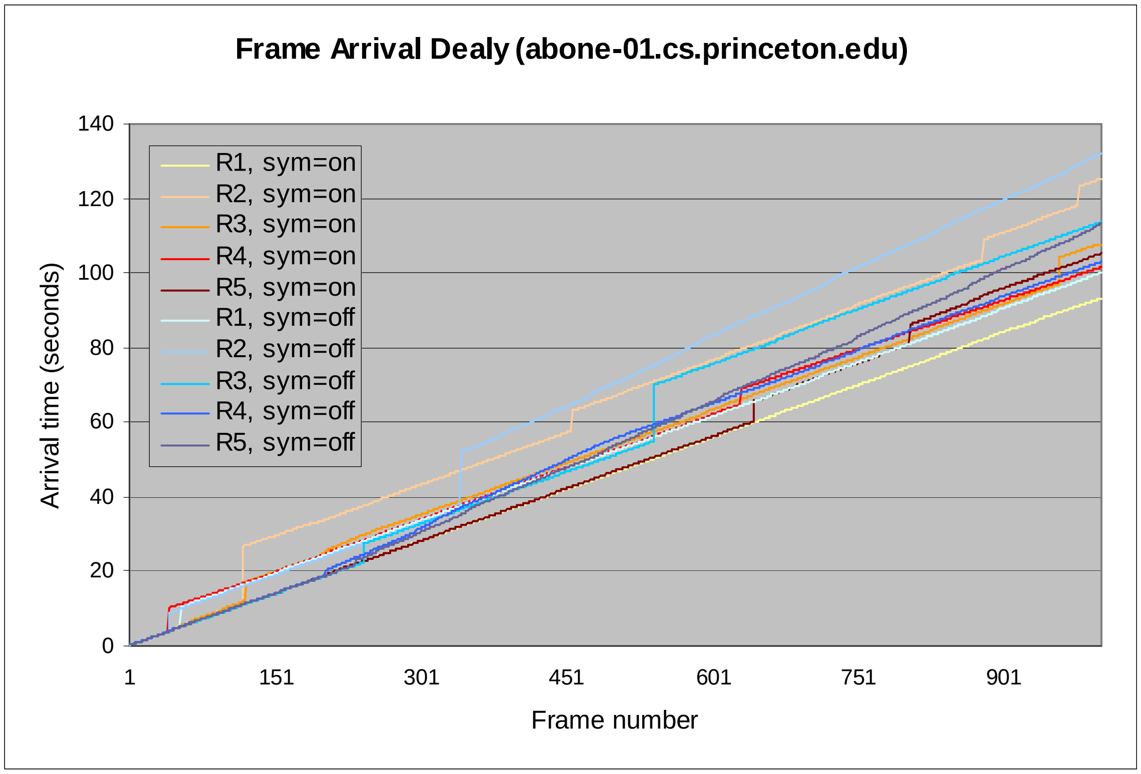


Figure-8. (E)

Figure-8. The figure plots the arrival time of the frames. For ideal case it should be linear. However, with each timeout event the backlog increased (observed as the step jumps in delay). The iTCP helps in gradually reducing these step delays for consecutive loss events.

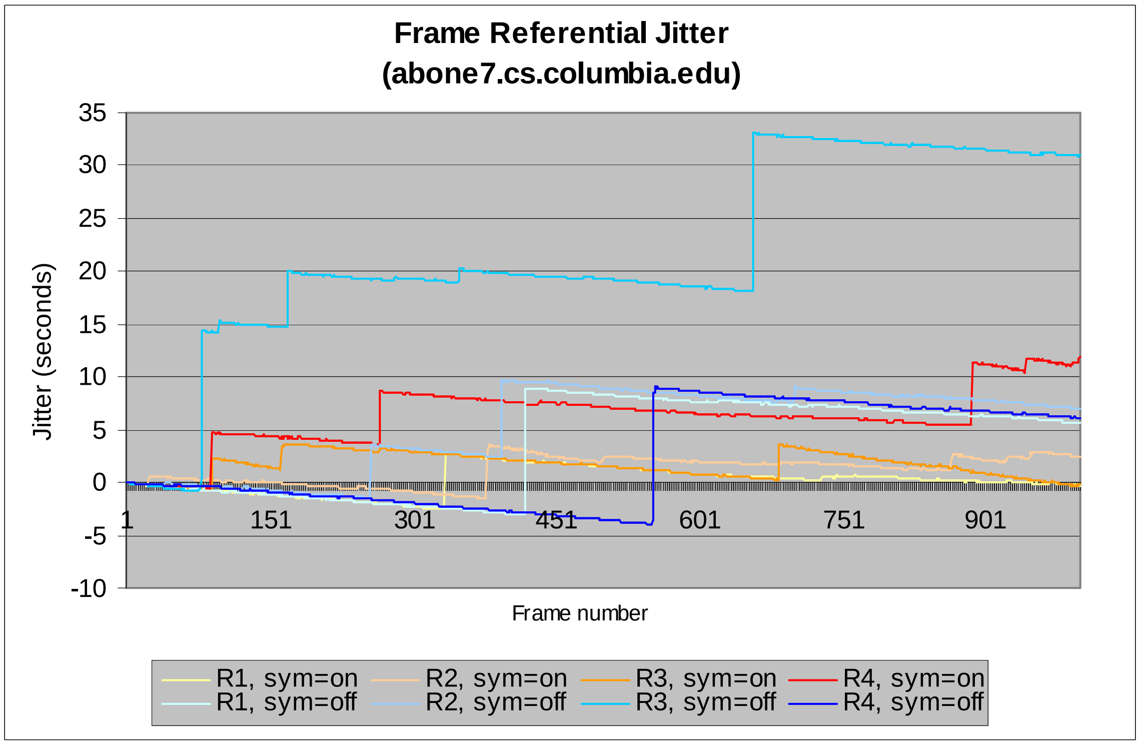


Figure-9. (A)

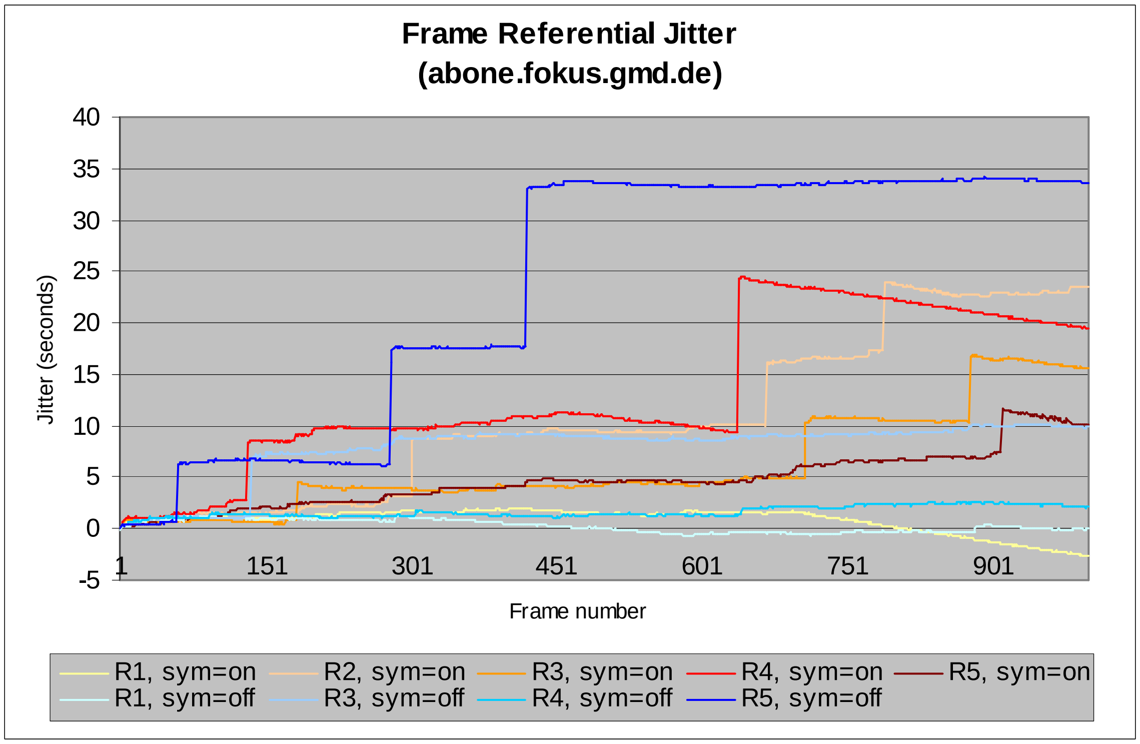


Figure-9. (B)

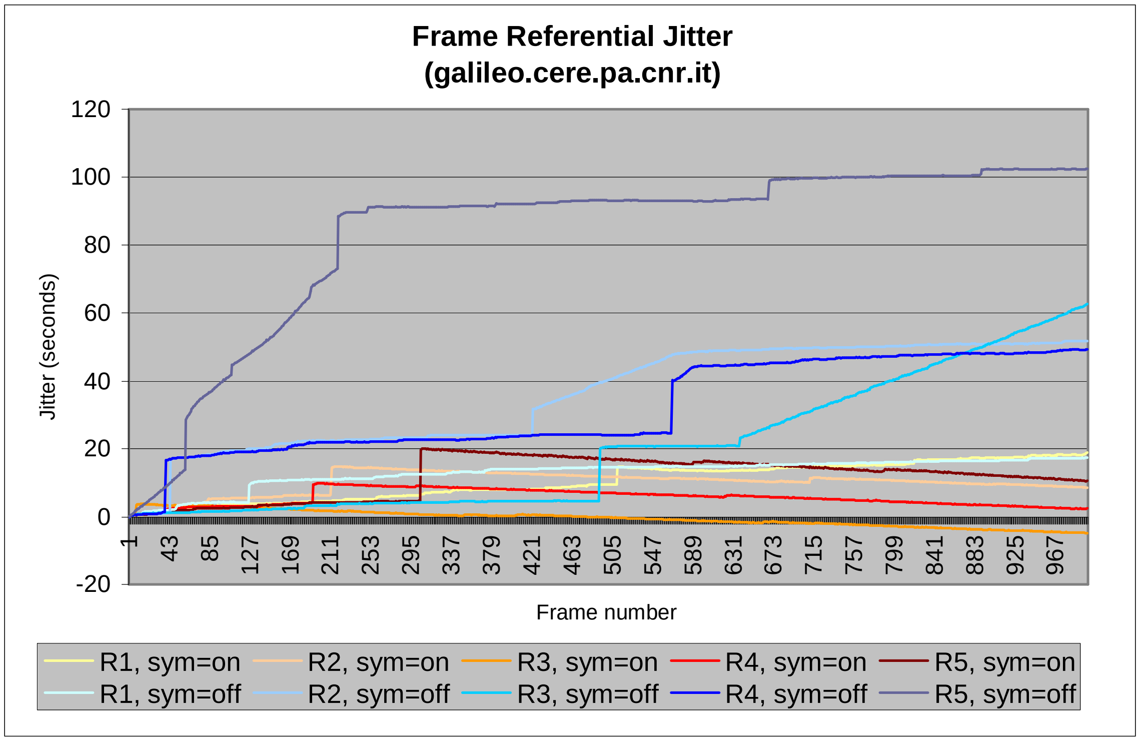


Figure-9. (C)

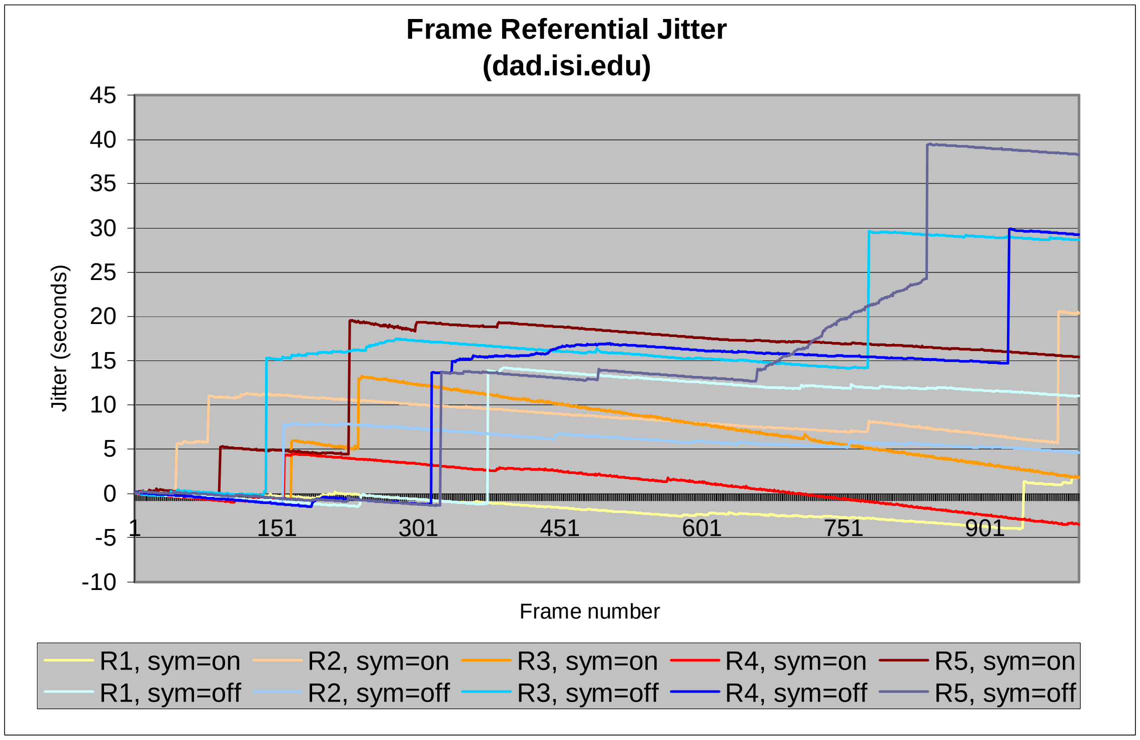


Figure-9. (D)

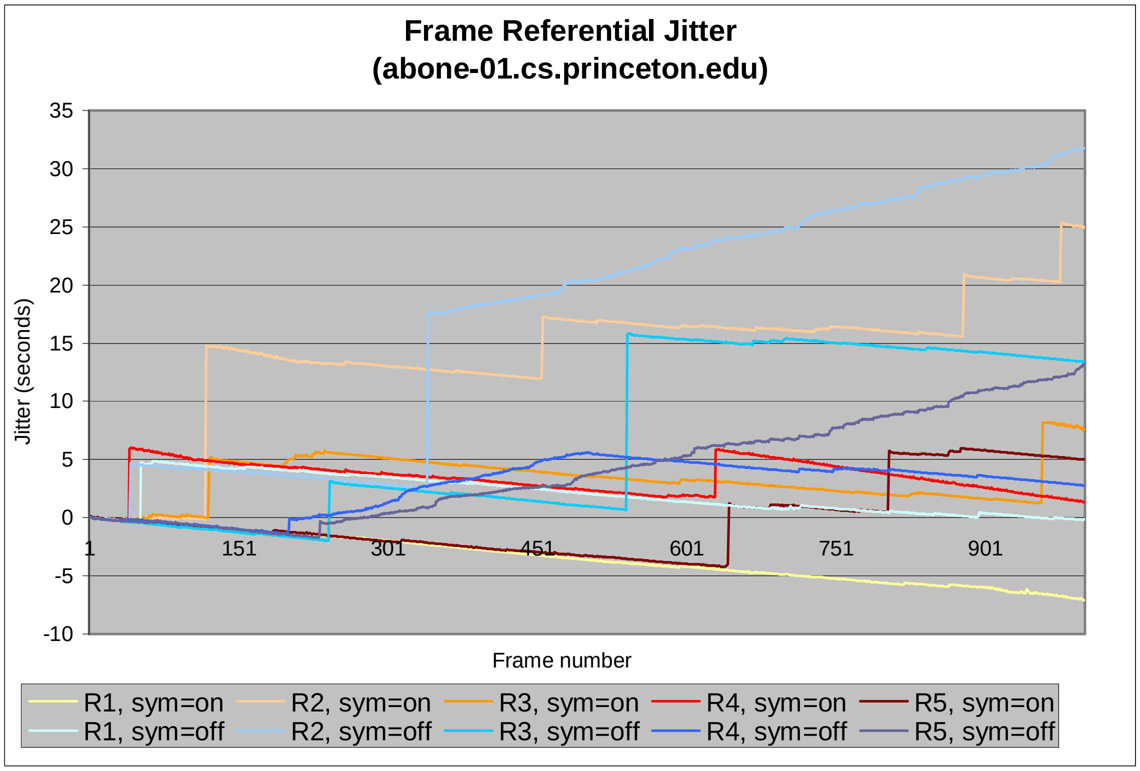


Figure-9. (E)

Figure-9. The figure plots the per frame jitter for the same set of experiments. A negative jitter however, means the frame arrived earlier than its ideal time. As shown, the classical TCP fell behind with each loss event.

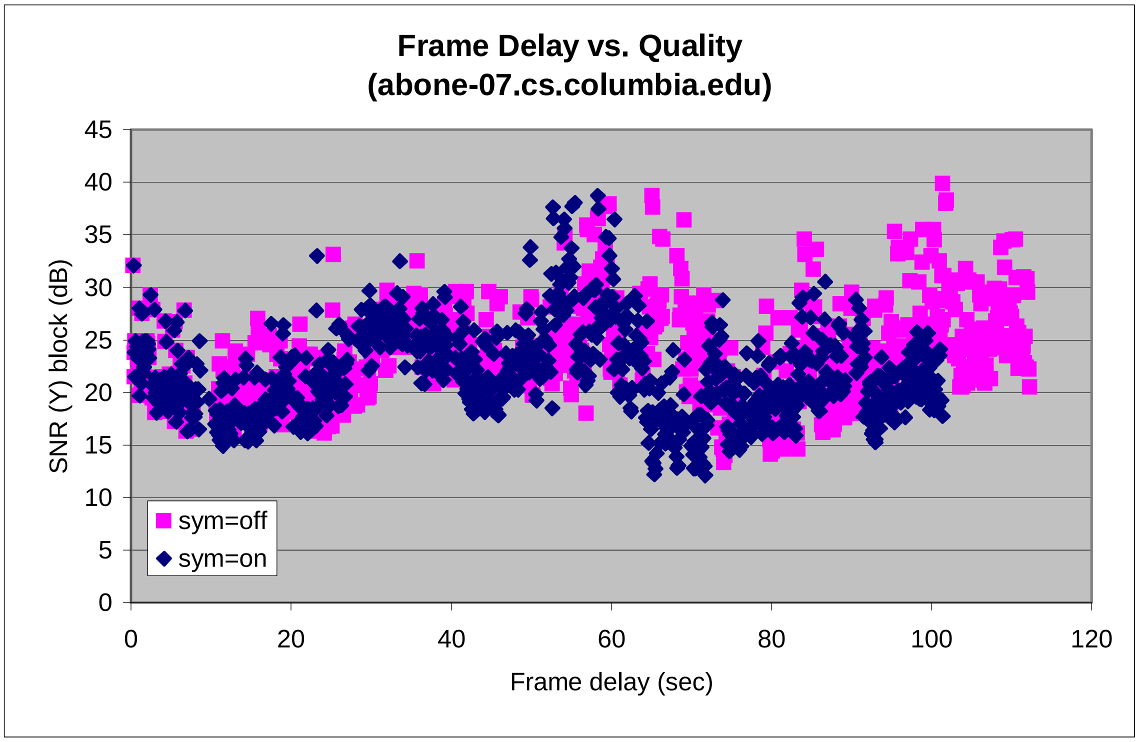


Figure-10. (A)

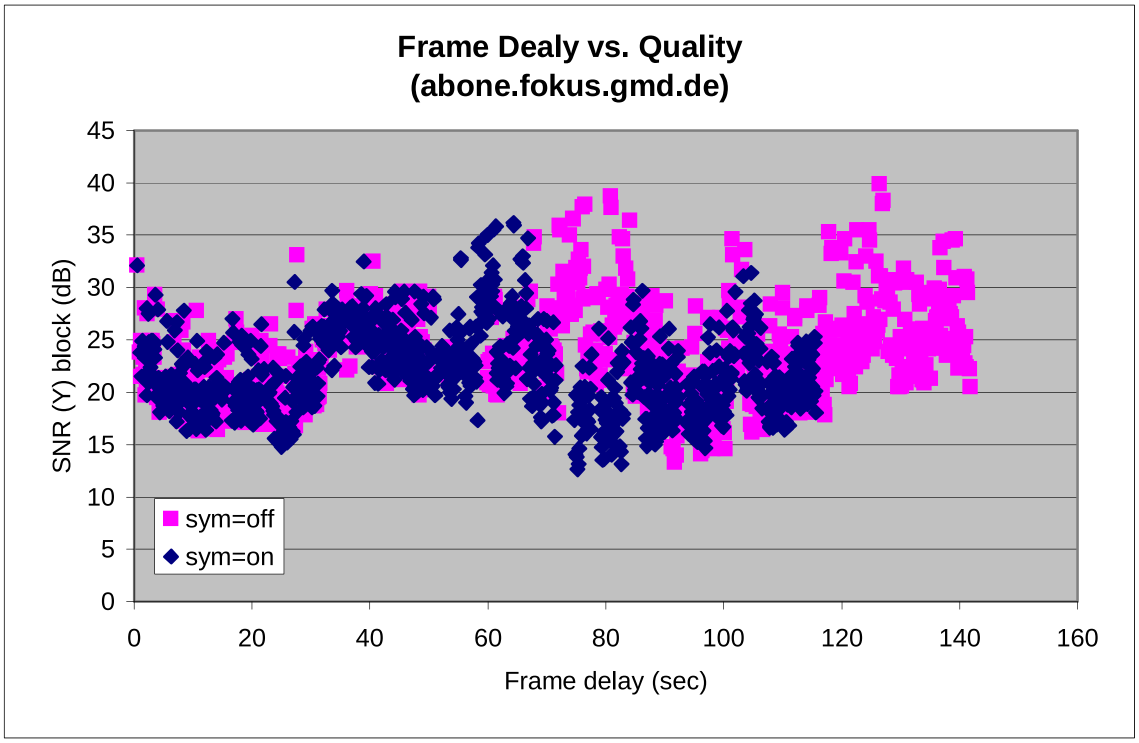


Figure-10. (B)

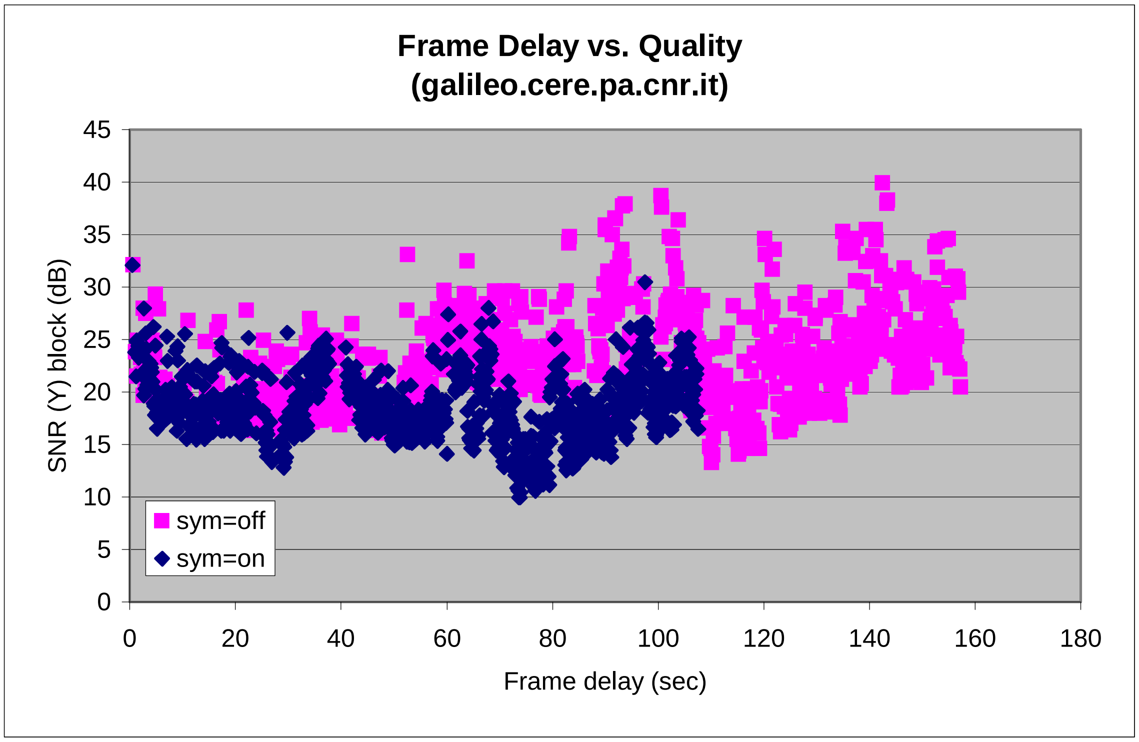


Figure-10. (C)

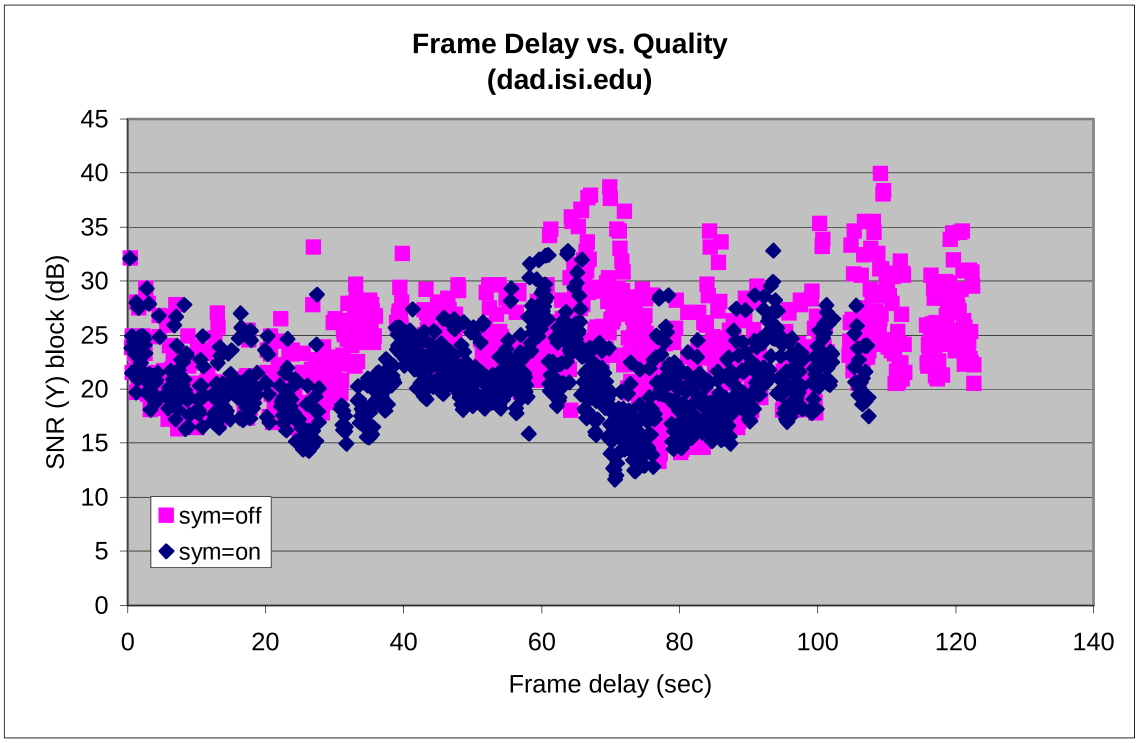


Figure-10. (D)

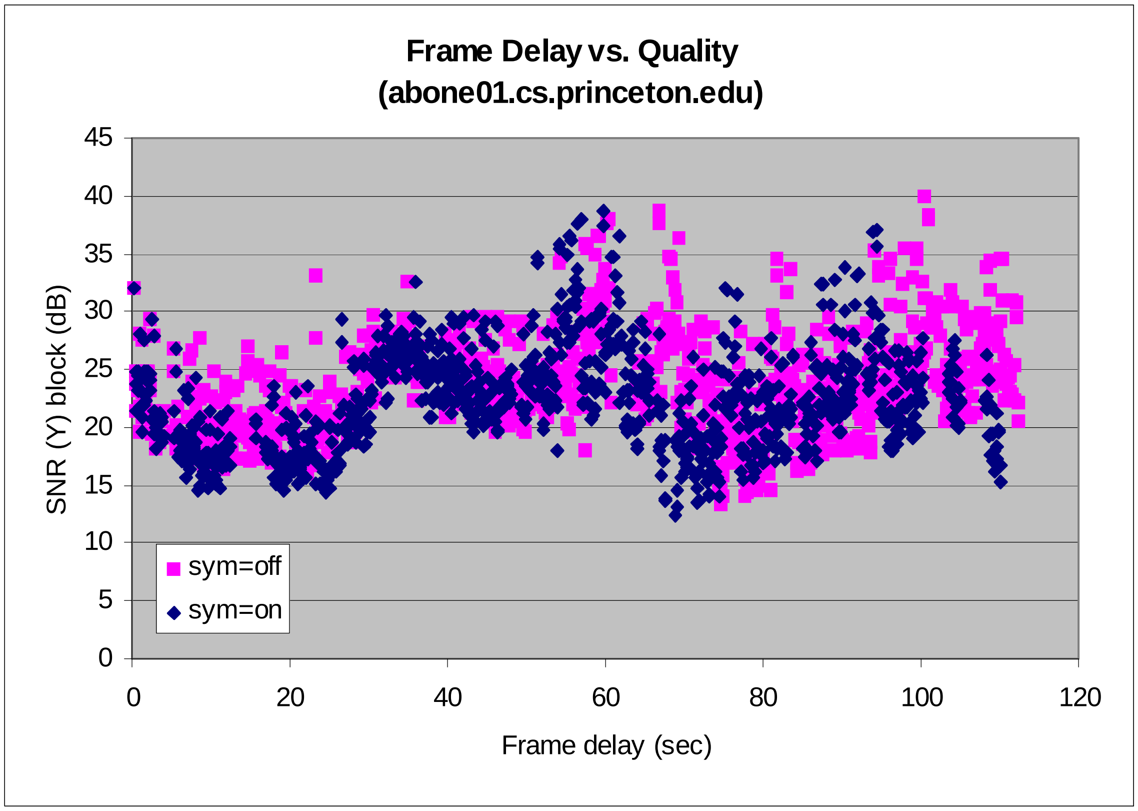


Figure-10. (E)

Figure-10. In each graph the two clusters shows the quality, delay tradeoff offered by the iTCP. The iTCP dramatically reduced frame delivery delay by controlled trade-off of the SNR quality

References

[ABCS00] D. Andersen, D. Bansal, D. Curtis, S. Seshan, and H. Balakrishnan, System Support for Bandwidth Management and Content Adaptation in Internet Applications, Proceedings of the Fourth Symposium on Operating Systems Design and Implementation, OSDI 2000, October 23-25, 2000 San Diego, California.

[AlPa99] Allman, Paxson, et al. TCP Congestion Control, RFC 2581, April 1999.

[BaRS99] Balakrishnan, H., Rahul, H., and Seshan, S., An Integrated Congestion Management Architecture for Internet Hosts," Proc. ACM SIGCOMM, Cambridge, MA, September 1999.pp.175-187.

[BrGM99] Hector M. Briceño, Steven Gortler and Leonard McMillan, NAIVE--network aware Internet video encoding, Proceedings of the seventh ACM international conference on Multimedia, October 30 - November 5, 1999, Orlando, FL USA, Pp 251-260.

[BrOP94] Brakmo, L.S., O'Malley, S.W., and Peterson, L.L, TCP Vegas: New Technique for Cogestion Detection and Avoidance, Proc. SIGCOMM' 94 Conf. ACM, pp-24-35, 1994.

[GuTe98] Wetherall, Guttag, Tennenhouse, "ANTS: A Tool kit for Building and Dynamically Deploying Network Protocols", IEEE OPENARCH'98, San Francisco, April 1998. Available at: http://www.tns.lcs.mit.edu/publications/openarch98.html

[ISO96] Information Technology- Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC International Standard 13818-2, June 1996.

[Jac88] Jacobson, V, "Congestion Avoidance and Control", Proc. SIGCOMM' 88, Conf, ACM, pp-214-329, 1988.

[Jac90] Jacobson, V., "Modified TCP Congestion Avoidance Algorithm," end2end-interest mailing list, April 30, 1990.

[KeWi00] Jun Ke and Carey Williamson / University of Saskatchewan, Towards a Rate-Based TCP Protocol for the Web, Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000.

[KhGu01] Javed I. Khan, Q. Gu, Network Aware Symbiotic Video Transcoding for Instream Rate Adaptation on Interactive Transport Control, IEEE International Symposium on Network Computing and Applications, IEEE NCA' 2001, October 8-10, 2001, Cambridge, MA, pp.201-213

[KHFH96] 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,

[KYGP01] Javed I. Khan, Seung Su Yang, Qiong Gu, Darsan Patel, Patrick Mail, Oleg Komogortsev, Wansik Oh, and Zhong Guo Resource Adaptive Netcentric Systems: A case Study with SONET- a Self-Organizing Network Embedded Transcoder, Proceedings of the ACM Multimedia 2001, October 2001, Ottawa, Canada, pp617-620

[KhY01] Javed I. Khan & S. S. Yang, Resource Adaptive Nomadic Transcoding on Active Network, International Conference of Applied Informatics, AI 2001, February 19-22, 2001, Insbruck, Austria, [available at URL http://medianet.kent.edu/, also mirrored at http:// bristi.facnet.mcs.kent.edu/medianet] (accepted).

[KhGR02] Javed I. Khan, Qiong Gu and Raid Zaghal, Symbiotic Video Streaming by Transport Feedback based quality rate selection, Proceedings of the 12th IEEE International Packet Video Workshop 2002, Pittsburg, PA, April 2002, http://www.pv2002.org .

[KPOY01] Javed I. Khan, Darsan Patel, Wansik Oh, Seung-su Yang, Oleg Komogortsev, and Qiong Gu, Architectural Overview of Motion Vector Reuse Mechanism in MPEG-2 Transcoding, Technical Report TR2001-01-01, Kent State University, [available at URL http://medianet.kent. edu/ technicalreports.html, also mirrored at http:// bristi.facnet.mcs.kent.edu/medianet] January, 2001]

[Mpeg00] MPEG-2 Test Model 5, [URL: http://www.mpeg.org/MPEG/MSSG/tm5/, last retrieved December 25, 2000]

[PeDa00] 19 L. L. Peterson and B. S. Davie. Computer Networks, 2nd edition, Morgan-Kaufmann, 2000.

[PrCN00] Prashant Pradhan, Tzi-cker Chiueh and Anindya Neogi Aggregate TCP Congestion Control Using Multiple Network Probing, Proceedings of the The 20th International Conference on Distributed Computing Systems, ICDCS 2000. 2000.

[ReHE00] Reza Rejaie,Mark Handley, Deborah Estrin, Architectural Considerations for Playback of Quality Adaptive Video over the Internet, Proceedings of the IEEE International Conference on Networks (ICON'00), 2000.

[SiWo98] 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.

[Tene96] Tanenbaum, A. Computer Networks, 3rd edition, Prentice Hall, 1996.

[TSSW97] Tennenhouse, D. L., J. Smith, D. Sincoskie, D. Wetherall & G. Minden., "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, Jan 97, pp 80-86

[Wolf97] Bernd E. Wolfinger, On the potential of FEC algorithms in building fault-tolerant distributed applications to support high QoS video communications, Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing , 1997, Pages 129 – 138