TCP Behavior over HFC Cable Modem Access Networks

Samir Chatterjee®
Computer Information Systems Department
Georgia State University
Atlanta, GA 30302-4015.

Phone: (404)-651-3886
Fax: (404)-651-3842
Email: schatter@gsu.edu

***Abstract**:*
Hybrid Fiber-Coax based CATV networks has the potential for two-way data services. Such network providers are presently re-inventing the CATV network to provide Internet access. This paper presents a simulation study of the dynamic behavior of TCP over HFC cable modem networks. We believe that this is the first such study as per our knowledge that considers asymmetry and its impact on TCP behavior. The performance of TCP connections with respect to flow control, adaptive retransmission mechanism and slow-start congestion control are studied and analyzed. In particular, we explain the dramatically different behavior and dynamics that are observed for TCP connections using interactive data (remote-login), bulk data (FTP) and web traffic (HTTP). We further study and discuss the impact of several important TCP and system parameters to optimize performance in CATV networks.

® This work has been partially funded through a research course release program from the College of Business Administration at Georgia State University.

1. Introduction

Today's corporations consider information to be a major corporate asset and hence are creating company-wide information exchange utility, called an enterprise network that will provide communications at the corporate and departmental levels. At the same time, changes in the business environment are transforming people's mode of operation and work habits. Increasingly, corporate employees spend time away from their offices and need to access information from remote locations. Working from home and telecommuting are becoming popular modes of operation and hence companies today are extending their enterprise network to include this new segment called home access. Access to corporate data and the Internet are critical to keep workers productive.

The recent popularity of the Internet has resulted in increased demand on our network infrastructure. As new data and multimedia applications come into existence, it places even further constraints on the existing networks. Among recent interest is the growth of Internet access technology from home [1]. Several industries (telcos, cable operators and wireless) have geared up to meet the challenge. At the present time, the CATV operators are reinventing their CATV networks to handle reverse data capability [8] and the telcos are looking towards a new technology called xDSL [2]. Our focus in this paper is on the former type of network.

There is a defensive aspect to cable's motivation for transitioning to residential broadband. This service is a necessary defensive measure against the threat of direct broadcast satellite (DBS) and new telco services. In particular, DBS has signed over five million subscribers in three years, threatening cable's customer base for basic television and premium pay channel service. Extending the capabilities of cable TV networks from current broadcast video business to high-speed Internet access seems natural. Cable has a large capital base to use as leverage for incremental applications. In addition to being the incumbent, cable has specific technical and business advantages with which it can succeed in providing a wide-range of high-speed home services.

Cable service is available to nearly one hundred percent of residences in the U.S and serves nearly 65 million homes. Worldwide there are 157 million subscribers. Widespread coverage helps attract advertising. In the U.S, cable sells $6 billion in annual advertising for national and local spot ads. Cable has more bandwidth than current telephone access networks and wireless networks. The speeds are so high that content providers who deliver over cable are considering changing their content specifically to exploit the speed of the cable, for example, by adding more audio and visual content to web pages. Finally, the top cable operators in the United States are important content providers. TCI owns Liberty Media; Time Warner own Warner Brothers, Turner Broadcasting, and a host of magazines (*Time, Life, People, Sports Illustrated, and Fortune*). When content providers own cable properties, they have guarantees of a high-speed outlet for their visual content, and plentiful content to occupy the many channels cable affords.

The Internet uses the TCP/IP protocol suite for supporting data transport [3, 4]. The performance of these protocols over such new access networks remains largely unknown. It is important that we study and analyze the behavior of these protocols over cable networks. The main contribution of this work is an exhaustive simulation study of TCP's behavior over a Hybrid Fiber-Coax

(HFC) cable modem-based access network. We provide insight into the different behavior of applications that use TCP over HFC. While we do not advocate any drastic changes to TCP protocol at this time, our analysis clearly points out that the HFC MAC protocol and TCP can interact in unexpected ways. Based on our analysis, the paper also suggests various parameters that can be tuned so that one obtains superior performance over cable based networks.

The rest of the paper is divided into the following sections. In section 2, we address the importance and relevance of this problem. Section 3 presents a brief overview of HFC cable modem networks. It also discusses the recent IEEE 802.14 standards. In section 4, we highlight the key features and algorithms that TCP uses which are later used throughout the paper. In section 5, we present the simulation model and the results. Finally, we suggest some improvement schemes and conclude with future work in section 6.

2. Why is this study important?

We strongly believe that performance study of the behavior of TCP/IP over such new access networks is important for both technical and business reasons. First, several cable operators are becoming ISPs to generate alternate sources of revenue. In the U.S, it is estimated that as much a $1 billion of revenue from cable modem could be generated in 1998. Hence these vendors must know more about the performance bottlenecks in a large-scale shared media cable. Besides there are many technical reasons to study this:

- ♦ An HFC network is a highly asymmetrical network with large bandwidth downstream (30 Mb/s) and a relatively small upstream bandwidth (2 Mb/s) which is shared. Since TCP's performance depends on Round Trip Time (RTT) and acknowledgments, its impact on such access network is unknown.
- ♦ TCP is the primary mechanism of congestion control on the Internet. It is expected that the upstream cable access channel may encounter high rates of congestion as the penetration rate (or take rate) of subscribers goes up.
- ♦ HFC network is a split metropolitan network in which every coordination is done via a head-end device. This causes unnecessary delays, which could affect network application performance. This also forces new ways of doing address resolution (ARP) [17].
- ♦ Improvements prompted by a long history of investigation [10, 11, 12, 14, 15] have led to a continuous expansion of the operating conditions under which TCP works well. There is no study to date available in the public literature that reports how well TCP performs on HFC networks. The performance of common client-server application over HFC was reported in an earlier work [17].

3. Hybrid Fiber-Coax and cable modem networks

Historically, CATV networks were developed as one-way broadcast systems using a tree-and-branch architecture [1]. The system consists of a head-end (H/E) station that collects the TV programs, usually through satellite down-link, and then distributes them over coaxial cables along the branches of a tree-like network to customer sites. The distance between H/E and remote customers can be several tens of kilometers. Hence to compensate for signal attenuation, several one-way analog amplifiers were deployed. Over time, the main coaxial trunk was replaced with high-reliability, low attenuation fiber. This trunk extends well into a community and is terminated in a fiber node. From there, coaxial segments are run into the neighborhood and tapped into the homes. HFC systems have the potential for two-way data services. A two-way HFC network has a bandwidth of 750 MHz, in which the range between 5 MHz and 42 MHz is used for upstream transmissions, while the range of 54 MHz and above for downstream transmission. The spectrum between 54 MHz and 450 MHz will remain undisturbed to carry existing analog and TV transmissions while the frequencies above 450 MHz will provide enhanced switched and broadcast data services.

To transmit and receive digital data over a cable network, one needs a cable modem. Since HFC offers a shared medium environment, the upstream bandwidth is to be shared by several cable modems accessing from homes. The 40 MHz band upstream can be divided into multiple upstream radio frequency (RF) digital channels 1 to n, each carrying a digital bandwidth in the range 1.6 Mb/s to 10 Mb/s, for example, using a quadrature phase shift keying (QPSK) modulation scheme. A Media Access Control (MAC) protocol is needed to share upstream bandwidth [5, 6, 7]. To achieve widespread acceptance, both cable modems and the HFC networks on which they operate must be standardized. The IEEE 802.14 working group is working towards the development of physical (PHY) and MAC[1] layer communication protocols for HFC networks [9].

4. TCP algorithms

The Transmission Control Protocol (TCP) is the primary transport protocol in the TCP/IP protocol suite [3, 4]. It implements a reliable byte stream over the unreliable datagram service provided by IP. As part of implementing the reliable service, TCP is also responsible for flow and congestion control: ensuring that data is transmitted at a rate consistent with the capacities of

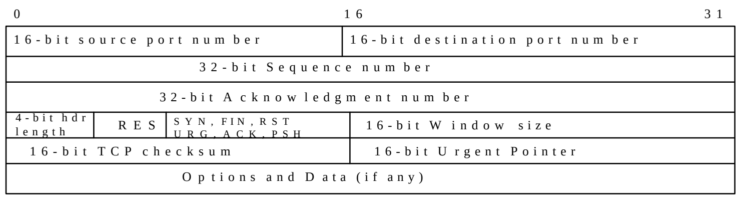
0
16
31
16-bit source port number
16-bit destination port number
32-bit Sequence number
32-bit Acknowledgment number
4-bit hdr length
RES
SYN, FIN, RST URG, ACK, PSH
16-bit Window size
16-bit TCP checksum
16-bit Urgent Pointer
Options and Data (if any)

Figure1: TCP segment header

both the receiver and the intermediate links in the network path. Since there may be multiple TCP connections active in a link, TCP is also responsible for ensuring that a link's capacity is responsibly shared among the connections using it. As a result, most throughput issues are rooted in TCP. The TCP segment header is shown in Fig. 1.

[1] Industry also supports the Multimedia Cable Network System (MCNS) Data over Cable Service Interface Specification (DOCSIS)[http://www.cablemodem.com].

We will study TCP's behavior using the following themes:

Sequence Numbers and Sliding/window: TCP keeps track of all data in transit by assigning each byte a unique sequence number (Figure 1). The receiver acknowledges each byte by sending an acknowledgment which is a cumulative indication of all received data up to a particular byte number. TCP allocates its sequence number from a 32-bit wraparound sequence space. To ensure that a given sequence number uniquely identifies a particular byte, TCP requires that no two bytes with the same sequence number be active in the network at the same time.

Flow Control/Window Size: The flow control mechanism used by TCP is known as a credit allocation scheme. The receiver needs to adopt some policy concerning the amount of data it permits the sender to transmit. Hence every TCP segment contains a window field which allows the receiving TCP to control how much data is being sent at any given time. The receiver advertises a 16-bit window size to the sender. This window size reflects the current buffer space available within the receiver to hold incoming segments. As the receiving TCP entity begins to deliver byte streams to the application, more buffer space is released and hence bigger window sizes may be advertised. The window measures, in bytes, the amount of unacknowledged data that the sender can have in transit to the receiver. A conservative approach is to only allow new segments up to the limit of available buffer space. However such a conservative flow control scheme may limit the throughput of the transport connection in long-delay situations. The receiver could potentially increase throughput by optimistically granting credit for space it does not have. For example, if a receiver's buffer is full but it anticipates that it can release space for 1000 bytes within a RTT delay, it could immediately send a credit of 1000. If the receiver can keep up with the sender, this scheme may increase throughput and can do no harm. If the sender is faster than the receiver, however, some segments may be discarded, necessitating a retransmission.

Retransmission timer: In order to ensure that all data sent by one end of the connection is received by the other, TCP must retransmit segments that have been lost by the network. TCP sets a timer (the retransmission timeout) when data is sent; if no acknowledgment is received for the sent data before the timer expires, it is assumed that the segment was lost and the data is sent again. TCP uses an adaptive timer mechanism. The following Jacobson/Karels algorithm is used to measure timeout RTO.

$$\textit{Difference} = \textit{Sample RTT} - \textit{Estimated RTT}$$
$$\textit{Estimated RTT} = \textit{Estimated RTT} + (g * \textit{Difference})$$
$$\textit{Deviation} = \textit{Deviation} + h(|\textit{Difference}| - \textit{Deviation})$$
$$\textit{RTO} = \textit{Estimated RTT} + f * \textit{Deviation}.$$

g is the RTT gain, h is the deviation gain and ϕ is RTT deviation coefficient.

Congestion control/Slow Start: A number of recent approaches have been suggested to manage the send window for TCP [10, 11]. The size of TCP's send window can have a critical effect on whether TCP can be used efficiently without causing congestion. The larger the send window used in TCP, the more segments that a TCP source can send before it must wait for an ACK. In the ordinary course of events, the self-clocking nature of TCP paces TCP appropriately. However, when connection is first initialized, it has no such pacing to guide it. A procedure known as *slow-start* is recommended. TCP makes use of congestion window, measured in segments rather than octets. At any time, TCP transmission is constrained by the following relationship:

awnd = MIN[credit, cwnd]

awnd = allowed window in segments that TCP will currently allow
cwnd = congestion window in segments.
credit = the amount of unused credit granted in the most recent ACK in segments. The value is obtained from the window field advertised by the other end.

When a new connection is opened, the TCP entity initializes *cwnd* = 1. That is, TCP is only allowed to send 1 segment and then must wait for acknowledgment before transmitting a second segment. Each time an ACK is received, the value of *cwnd* is increased by 1, up to some maximum value. In effect, the slow-start mechanism probes the internet to make sure that it is not sending too many segments into an already congested environment. The term *slow-start* is a bit of a misnomer, because *cwnd* actually grows exponentially.

<u>Dynamic window sizing on congestion</u>: Consider a TCP entity that initiates a connection and goes through the slow-start procedure. At some point, either before or after *cwnd* reaches the size of the credit allocated by the other side, a segment is lost (timeout). This is a signal that congestion is occurring. It is not clear how serious the congestion is. Therefore, a prudent procedure would be to reset *cwnd = 1* and begin the slow-start process all over. While this may seem reasonable, Jacobson [10] points out that "it is easy to drive a network into saturation but hard for the net to recover." In other words, once congestion occurs, it may take a long time for the congestion to clear. Thus, the exponential growth of *cwnd* under slow-start may be too aggressive and may worsen the congestion. Instead, Jacobson proposed the use of slow start to begin with, followed by a linear growth in *cwnd.*

Note that two successive releases of the Berkeley TCP code, referred to as *Tahoe* and *Reno*, use a technique called *fast retransmit* and *fast recovery* that have been shown to improve performance under certain situations. We do not use these in our implementation and hence are not discussed.

5. Simulation

Most of the results in this paper are derived from simulations done using a tool called OPNET [16]. The various simulation parameters that were used is summarized in Table 1 while the HFC cable modem network simulation model is shown in Figure 2. The network consists of a H/E node and 5 home stations exchanging messages over HFC access network. Each home node uses a cable modem and their entire protocol stack is also shown in Fig. 2.

Each home node is busy running some application that uses TCP over the HFC cable link. Three client-server applications namely, Remote-login, File transfer and Web browsing are modeled. Remote-login (RL) sessions represent interactive data application that uses TCP. It is characterized by terminal traffic characteristics that typically include login rate, command rate and login duration. RL application typically uses TCP’s PUSH feature to send each command as quickly as possible.

Link bandwidths	Upstream = 2 Mb/s
	Downstream = 30 Mb/s

Distance of H/E to first home	10 km
Max segment size MSS	536 bytes
TCP buffer size	65,535 bytes
RTT gain	0.125
Number of home nodes	5
Applications	Remote-login (TELNET), File transfer (FTP), Web browsing (HTTP)
HFC MAC protocol	IEEE 802.14
H/E router buffer size	512 packets (typical)
Simulation duration	4000 seconds

Table 1: General Simulation Parameters

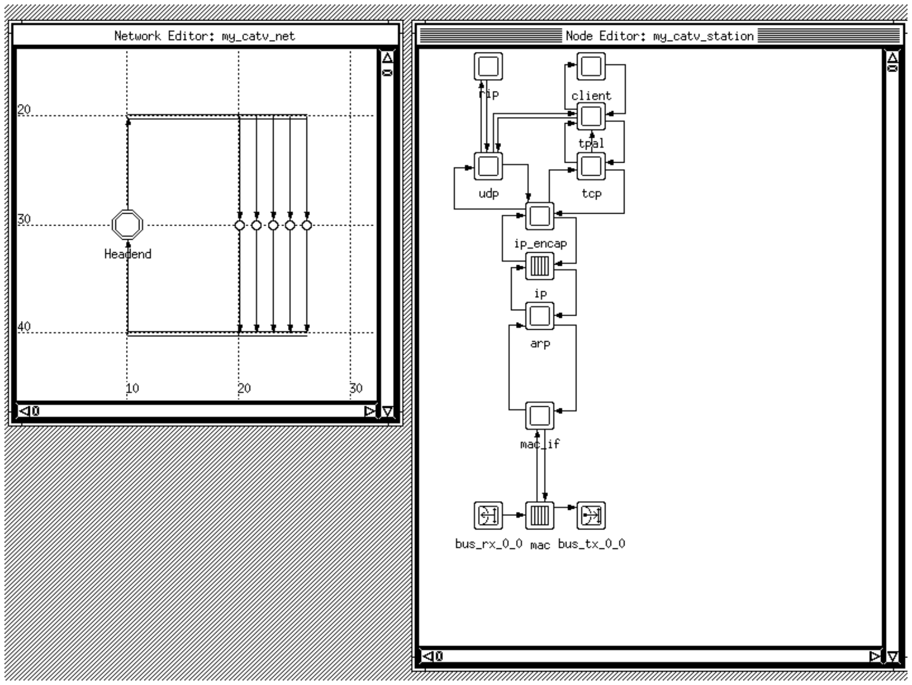

Figure 2: Simulation model & protocol stacks

The FTP application on the other hand models a more bulk data transfer scenario that also uses TCP. This application is typically characterized by a file transfer rate, average file size and the get/put direction of file upload or download. An FTP application typically drives TCP to keep the pipe full as much as possible. We also used a web browser application that models HTTP traffic. In each session, the client process can establish multiple connections to servers, send multiple request commands for HTML pages and inline objects and process the request response.

Results and discussion:
In Fig. 3 and Fig. 4, we observe the dynamics of TCP for interactive and bulk data connections. The behavior of a remote-login session that uses TCP and starts at 942.0 sec and ends at 3841.2

sec is captured in Fig. 3. In a RL application, each keystroke is sent from client to the server 1 byte at a time and the server echo the characters back. Normally, the server piggyback's the ack of received data[2] along with the echoed data into one TCP segment and hence we refer to these as delayed ack's. Treating a command request as a unit of transfer creates another problem for RL and that is it creates several tiny segments (1-byte of data with 40 bytes of TCP and IP header overhead). These tinygrams cause congestion on MAN and WAN. Nagle algorithm is used to tackle this problem in which a TCP connection can have only one outstanding small segment that has not yet been acknowledged. So small amounts of data are collected by TCP and sent in a single segment when the ack arrives.

In Panel #1 of Fig. 3, we show the sent sequence number for the TCP entity at the home node. Command request segments are generated and sent but the long horizontal steps clearly show retransmission of the same sequence number segments. The RL session is facing congestion and loss due to which interactive data have to be retransmitted causing unnecessary delays. The graph has a step function nature but we never see any negative slope. This is because Nagle's algorithm is being used in which only one tinygram is allowed to be send until its ACK returns. Hence the longer horizontal steps signify retransmission of the lost segments. Panel #2 shows the measured RTT along with the RTO used by TCP for timeout. Note that while several thousands of TCP segments are sent, only a few are used as samples for measuring RTT. The gaps in the RTT samples are caused by retransmissions and Karn's algorithm prevents us from updating our estimators until another segment is transmitted and acknowledged. Also note that TCP's calculated RTO is always a multiple of 500 ms due to timer granularity. Panel #3 shows how the cwnd varies with time for the duration of the connection. It starts at 1 segment (MSS = 536 bytes) and exponentially tries to grow using slow-start but as seen fails due to heavy congestion at the H/E which causes cwnd to drop to 1 segment. It never really recovers from slow start and hence cannot achieve high throughput. In Panel #4, note the size of the receivers advertised window which is initialized at 65,536 bytes but does not change much since tinygrams from RL can be quickly processed and hence the buffer always has enough space to hold arriving segments.

All these parameters contrast very heavily for bulk data transfer using FTP as seen in Figure 4. The particular TCP connection over which the files were transferred lasted between 1021.6 sec and 1946.4 sec. So this is a relatively short connection when compared to RL. As seen from Panel #1, the sent sequence numbers increases linearly with no segment being retransmitted. Note that there are no sequence number wrap-ups while data is in transit. While this is a problem for long propagation-delay paths [13] but for CATV links, PAWS (protection against wrapped sequence numbers) may not be required. This FTP session gets a fair allocation of the bandwidth.

In Panel #3 of Fig. 4, we see *cwnd* for this ftp transfer which clearly shows the slow-start procedure rising exponentially. This is an ideal behavior expected from TCP. Comparing Panel #4 with that of the RL session, it is clear that the receivers advertised window changes significantly during the connection dropping to 35,000 bytes or less. This is because FTP can keep the pipe full with steady streams of data and hence the receiver's buffer fills up before

[2] In our implementation, we transfer commands between the terminal and the server with the average command size from terminal being 60 bytes

application can read data. It is evident from the results that bulk data transfer performs far better than interactive data transfer.

Why does RL and FTP TCP sessions exhibit dramatically different behavior? How much of this are due to the HFC MAC protocol characteristics? What are the implications? When an interactive RL connection shares upstream bandwidth with a bulk transfer FTP connection, the interactive session suffers in performance. There could be several reasons for that. First note that TCP connection arrival follows some stochastic process and hence the exact behavior may depend on the currently existing connection mix through the upstream. Besides in a HFC MAC, cable modems contend for getting reservation slots to transmit their data. A bulk data application can easily fill up segments up to MSS and request guaranteed slots for transmission while it is harder for interactive application to guess *a priori* how much to reserve. When cable modems send their reservation requests through contention mini-slots, it is often difficult to choose the necessary reservation for interactive data. Hence to improve the performance of interactive data transfer, some better scheduling algorithm may be required at the H/E to better utilize TCP. This has major implications since telecommuting is an application that MSO's are betting on for the penetration of the cable modem business.

We also had a web browser client connecting to a web server. Every TCP connection was very short lived for HTTP and completed even before the slow start algorithm has finished (see Fig. 5). The user will never experience the full link bandwidth. All the transfer time will be spent in slow start. This problem is particularly severe for HTTP that is notorious for starting a new TCP connection for every item on a page. This poor protocol design is a (major) reason Web performance on the Internet is perceived as poor: the Web protocols never let TCP get up to full speed. We expect this to remain true even over HFC access networks.

There are at least four system resources that could affect performance: maximum segment size (MSS), size of the IP buffer at H/E, TCP's receive buffer size and server processing speed at H/E. In Fig. 6, we study the impact of MSS on a TCP connection's throughput and delay. The MSS is the largest "chunk" of data that TCP will send to the other end. When a connection is established, each end can announce its MSS. The resulting IP datagram is normally 40 bytes larger: 20 bytes for the TCP header and 20 bytes for the IP header. In general, the larger the MSS the better, until fragmentation occurs. This may not be true however in all cases. A larger size allows more data to be sent in each segment, amortizing the cost of IP and TCP headers. For FTP over TCP (see Panel #1), a larger value of MSS results in better TCP throughput. The TCP end-to-end delay varies inversely with MSS and smaller MSS (536 bytes) have much higher delays. Choosing the right MSS is always difficult. The MSS lets a host limit the size of datagrams that the other end sends it. When combined with the fact that a host can also limit the size of the datagrams that it sends, this lets a host avoid fragmentation when a host is connected to a network with a small MTU. The only way to avoid fragmentation is to discover the path MTU which remains tricky even now. While MSS clearly impacts FTP transfers, it may not affect interactive transfers such as RL (see Panel #2 and #4). In a RL TCP connection, only a few bytes are transferred and hence smaller MSS segments perform better both in throughput and delays than larger MSS since the large MSS is really never utilized with user data.

We can determine the maximum possible throughput on a TCP connection. It depends on the window size, propagation delay, and data rate. In TCP, window size and sequence numbers refer to individual octets. We use the following notation:

ω = TCP window size (octets)
ρ = Data rate (bps) at TCP source available to a given TCP connection
δ = Propagation delay (seconds) between TCP source and destination over a given TCP connection.

For simplicity, let us ignore the overhead bits in a TCP segment. Suppose that a source TCP entity begins to transmit a sequence of octets over a connection to a destination. It will take δ seconds for the first octet to arrive at the destination and an additional time δ for an acknowledgment to return. During that time, the source, if not limited, could transmit a total of $2\rho\delta$ bits, or $\rho\delta/4$ octets. In fact the source is limited to window size of ω octets (advertised by H/E) until an acknowledgment is received. Accordingly, if $\omega > \rho\delta/4$, the maximum possible throughput can be achieved over this connection. If $\omega < \rho\delta/4$, then the maximum achievable normalized throughput is just the ratio of ω to $\rho\delta/4$.

For a CATV network, ρ = 2 Mbps in the upstream assuming that a home node has full bandwidth access. With a propagation delay of 5 μsec/km and assuming a home node to be at a distance of 10 km from the H/E, then δ = 50 μsec. Hence $\omega > \rho\delta/4$ = 25 bytes. The maximum window size is 2^{16} - 1 = 65,535 octets should suffice for most applications. Fig. 7, shows the impact of typical TCP receiver buffer sizes (window) on overall performance. Clearly, the above shows the maximum throughput achievable in theory but in reality due to sharing of several TCP connections in the upstream link, the actual throughput would be less. We used two standard values for TCP receive buffer, those of 4096 bytes and 65,535 bytes. As seen, FTP connections achieve higher throughput and lower delay for 65,535 byte receive buffers in comparison to shorter buffer sizes. But the impact of any window size on RL TCP connection is not that clear. It may at times achieve higher throughput with smaller TCP receive buffer size. Again a possible cause of this could be smaller datagram sizes and Nagle's algorithm restricting the number of datagrams to be sent before ACK arrives.

Bandwidth	Delay x Bandwidth Product
T1 (1.5 Mbps)	18 KB
Ethernet (10 Mbps)	122 KB
T3 (45 Mbps)	549 KB
FDDI (100 Mbps)	1.2 MB
OC-3 (155 Mbps)	1.8 MB
OC-12 (622 Mbps)	7.4 MB
OC-24 (1.2 Gbps)	14.8 MB

Table 2: Required window size for 100 ms of RTT

In Fig. 8, we study and analyze the impact of IP buffer size at the H/E on overall performance. One easy way to support a large number of users (and hence TCP connections) is to provide more storage in the network with either higher bandwidth or more buffering within the H/E IP processor. We considered buffering because it is more practical. Even with small number of connections, bottleneck routers should have at least one delay-bandwidth product of buffering. Table 2 shows the delay *x* bandwidth product for several common network technologies encountered in daily life.

A TCP connections one-way propagation time in a HFC link is 50 μsec. A 2 Mbps link can only forward 25 bytes of data during that latency time from a home node to the H/E. So theoretically, the IP buffer within the H/E need only be of size 25 bytes, which is not even a packet (assuming minimum IP datagram to be of size 576 bytes). But for all practical purposes, we must have sufficient buffer sizes so that it does not cause unnecessary packet loss (when multiple TCP clients transmit simultaneously) that could result in TCP retransmission and thereby wasting bandwidth. Note that the application empties the TCP buffer and hence the server processing speed (in jobs/sec) also plays a critical role in overall performance (see Fig. 9). While buffer requirements shown above in Table 2 for various networking technologies assume symmetry and equal requirements at either end of the connection, the asymmetric speeds of an HFC access network impose different buffering requirements at the H/E and at each home node. This is because at the downstream direction, H/E can transmit at full speeds of 30 Mbps giving rise to a bandwidth *x* delay product of 375 bytes. Hence each home node should have a buffer of at least 375 bytes. This gives a total of at least 400 bytes that any TCP sender must keep in its window if it wishes to keep the network busy.

Assuming IP packets of size 576 bytes, we tested the performance for various buffer sizes as shown in Fig. 8. Here we plot the effective TCP throughput, which is the average bps forwarded to the application layer by the TCP layer in the H/E node for all connections. At higher upstream load (caused by higher file transfer rates), the total throughput is good with even small number of buffers (80 packets/buffer). As shown above in the calculation, there are always more buffer space than total TCP's sending packets. The other panel shows the effective TCP delay versus load. A small IP queue under high load may result in higher packet loss rates which in turn causes TCP to time-out and retransmit. This leads to higher TCP delays.

In Fig. 9, we show the impact of server processing speed on the aggregate TCP throughput and delay of all connections within the H/E server. Specifically, two job rates of 1000 and 5000 jobs/sec were chosen. For underutilized upstream links, it does not seem to make too much of a difference while at extremely high link utilization[3], higher server speeds is expected to give better system performance.

6. Conclusions and future work

This paper is the first attempt to understand the behavior of TCP over cable modem networks. The salient features of HFC network presents interesting challenges to TCP. Our main research contributions are as follows:

1. Interactive data versus bulk data transport over HFC MAC links have very different behavior which is due to both TCP's inherent slow-start and congestion avoidance schemes as well as the nature of the MAC protocol. In particular we showed that interactive RL applications

[3] We ran out of memory in our simulation at very high loads and hence could not project the graphs shown to higher link utilization levels.

may under-perform largely due to the fact that it is difficult for such applications to reserve *a priori* the number of upstream slots. On the other hand, a bulk data application (such as FTP) can easily fill up to a MSS and yield better throughput and delay performance. This suggests two possible avenues for further research. First, scheduling algorithm at the MAC level needs to be investigated in light of how TCP may interact and have an impact for the high-level application. Second some form of QoS features has to be overlaid in the architecture.

2. We also showed that web traffic using TCP are very short-lived and hence TCP always remains in slow-start and never achieve full speed. We are aware that this problem is now being alleviated by HTTP 1.1 specification but thorough testing still remains to be done.
3. TCP is typically unfair towards connections with higher propagation delays and this could cause performance problems when multiplexing short and long-haul traffic on HFC upstream links. For guaranteed performance in highly utilized networks, each TCP connection should be given reserved buffer and bandwidth resources throughout the network. Typically, the resource allocation would be determined at connection set up and enforced at routers using per connection queuing. Since administering the resources allocated to every best effort connection may be excessively expensive, a more feasible option for HFC service providers would be to use some form of bandwidth allocation algorithm per class within the H/E. Using such a scheduling that does load balancing, the unfairness we pointed out for RL, FTP and Web traffic could be improved if not eliminated all together.
4. To test the viability of carrying traffic with various latency requirements on a single upstream channel, Broadcom recently modeled a set of scheduling implementations in conjunction with fragmentation support using a modified version of the MCNS MAC [18]. The modifications included an improved framing and scheduling algorithm, an additional binary Quality of Service (QoS) attribute for subscriber cable modems and support for source fragmentation. The scheduling algorithm used the polling mechanism suggested by MediaOne and MIT. Cable modems with a high QoS requirement were polled periodically by the H/E to allow them to request bandwidth. The resulting requests were given higher priority than those from best effort modems in granting bandwidth in upcoming frames. The improvement in performance gained through polling, prioritization and fragmentation have been shown. Packet delays are well bounded with an upper bound near 20 ms (as required for voice). However, the interactions of TCP/IP over the modified HFC MAC with the new polling scheduling are not yet studied.

There is room for more work on flow control mechanisms that maintain low loss rate regardless of load. As more HFC networks are deployed, more analysis of Internet traffic is needed especially to look at aggregate TCP flows and how they actually share the upstream link bandwidth. Also, the effects on performance as more homes come on-line need to be investigated. The behavior of large TCP flows over the bottleneck upstream link can also be explored. As a future work, we would like to investigate the impact of TCP in supporting real-time audio and video applications that can be run from home over HFC networks.

Acknowledgments: The author would like to thank Pradeep Singh of Mil3 for valuable discussions during the course of this work.

References:

[1] A. Paff, “Hybrid Fiber/Coax in the public telecommunications infrastructure”, IEEE Communications, April 1995.

[2] P. Kyees, R. McConnel, K. Sistanizadeh, "ADSL: A new twisted-pair access to the Information High-way", IEEE Communications, April 1995.

[3] Douglas E. Comer. *Internetworking with TCP/IP, Vol. 1:Principles, Protocols, and Achitecture*, 3rd Edition, Prentice Hall, 1996.

[4] W. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, MA:Addison-Wesley, 1994.

[5] D. H. Su, N. Golmie, G. Pieris, S. Masson, “Preliminary simulation results of the MAC protocol proposals”, IEEE 802.14/96-0126, May 1996.

[6] D. Sala, J. Limb, “A protocol for efficient transfer of data over fiber/cable systems”, *Proceedings of IEEE INFOCOM*, 1996.

[7] J. E. Dail, M. A. Dajer, Chia-Chang Li, P. D. Magill, C. A. Siller, K. Sriram, N. Whitaker, “Adaptive Digital Access Protocol: A Mac Protocol for Multiservice Broadband Access Networks”, IEEE Communications, March 1996.

[8] E. J. Hernandez-Valencia, “Architectures for Broadband Residential IP Services Over CATV Networks”, IEEE Network, Jan/Feb 1997.

[9] IEEE 802.14, “Cable TV MAC/PHY protocol working group functional requirements”, October 19, 1994.

[10] V. Jacobson, "Congestion Avoidance and Control", *Proceedings of SIGCOMM'88*, Computer Communication Review, August 1988.

[11] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP", in *Proceedings of SIGCOMM'96*, Stanford, CA, August 26-30, 1996.

[12] K. Fall, S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP", Computer Communication Review, Vol. 26, No. 3, July 1996.

[13] C. Partridge, T. Shepard, "TCP/IP performance over Satellite Links", IEEE Network, September/October 1997.

[14] T. V. Lakshman, U. Madhow, "The performance of TCP/IP for networks with high-bandwidth-delay products and random loss", IEEE/ACM Transactions on Networking, Vol. 5, No. 3, June 1997.

[15] R. Morris, "TCP behavior with many flows", *Proceedings of ICNP97*, Atlanta, GA, Oct. 1997.

[16] **OPNET 4.0**, *Mil3 Inc.,* Washington DC, 1998.

[17] S. Chatterjee, "Performance results and operational issues of client-server IP applications over HFC cable networks", *Proceedings of 6th IEEE Singapore International Conference on Networks'98 (SICON'98)*, Singapore, June 29-July 3, 1998.

[18] Thomas Quigley and David Hartman, "MCNS Data-Over-Cable Protocol", Communications Engineering Design, March 1998.

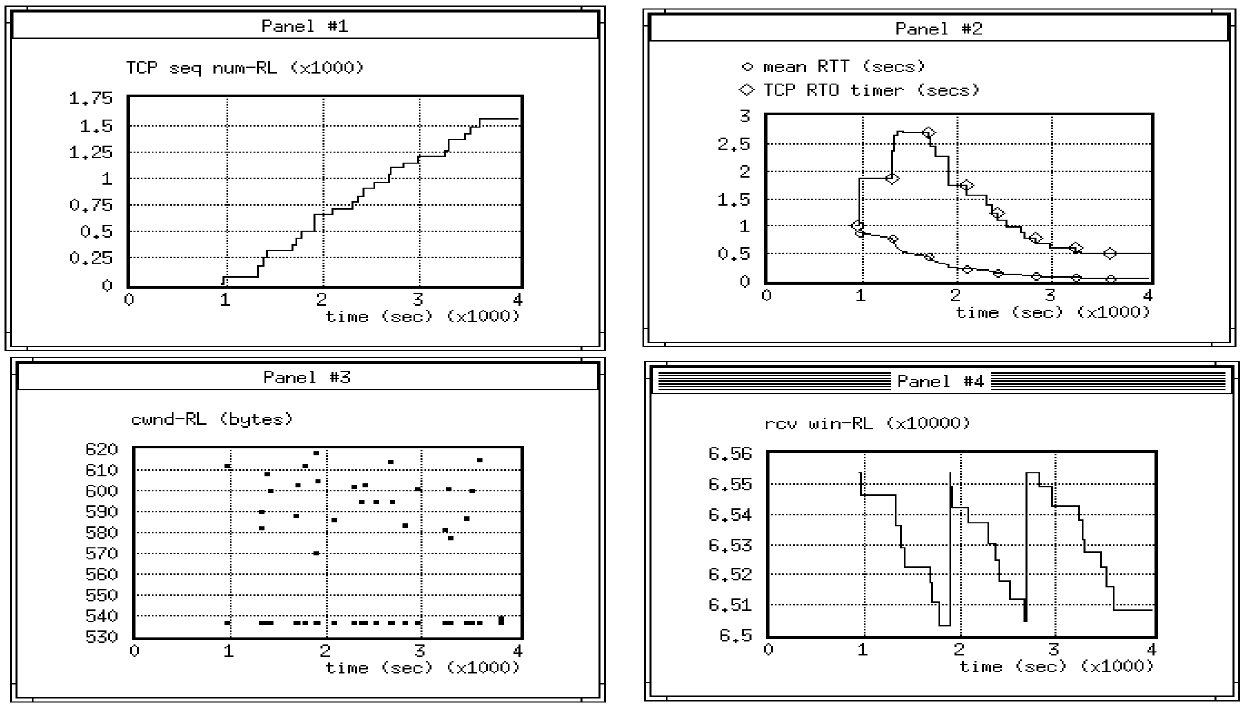

Figure 3: TCP connection behavior for interactive data (Remote-login application)

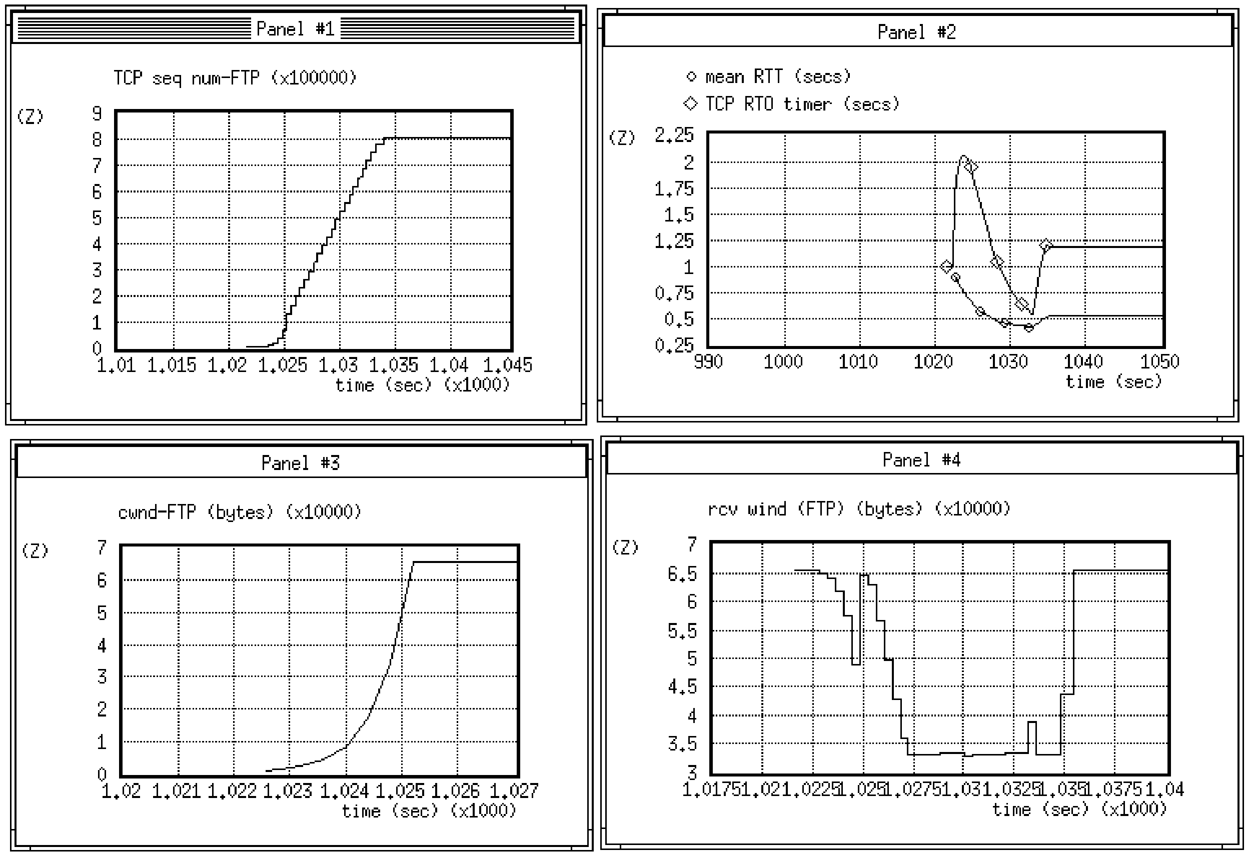

Figure 4: TCP connection behavior for bulk data (FTP application)

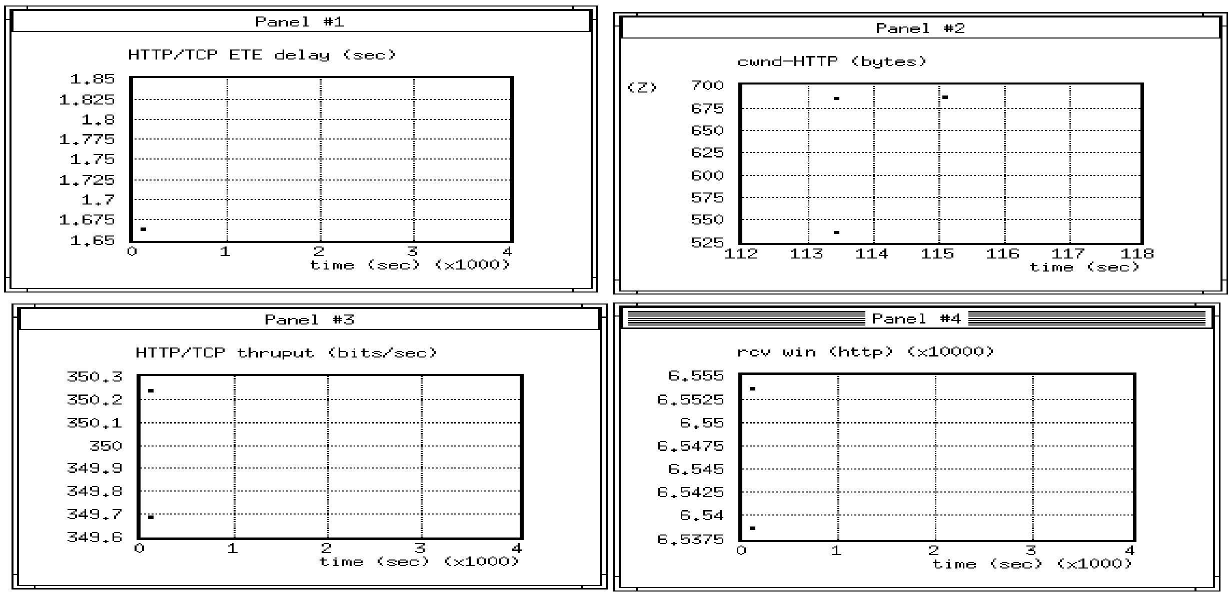

Figure 5: TCP connection behavior for world-wide web traffic (HTTP application)

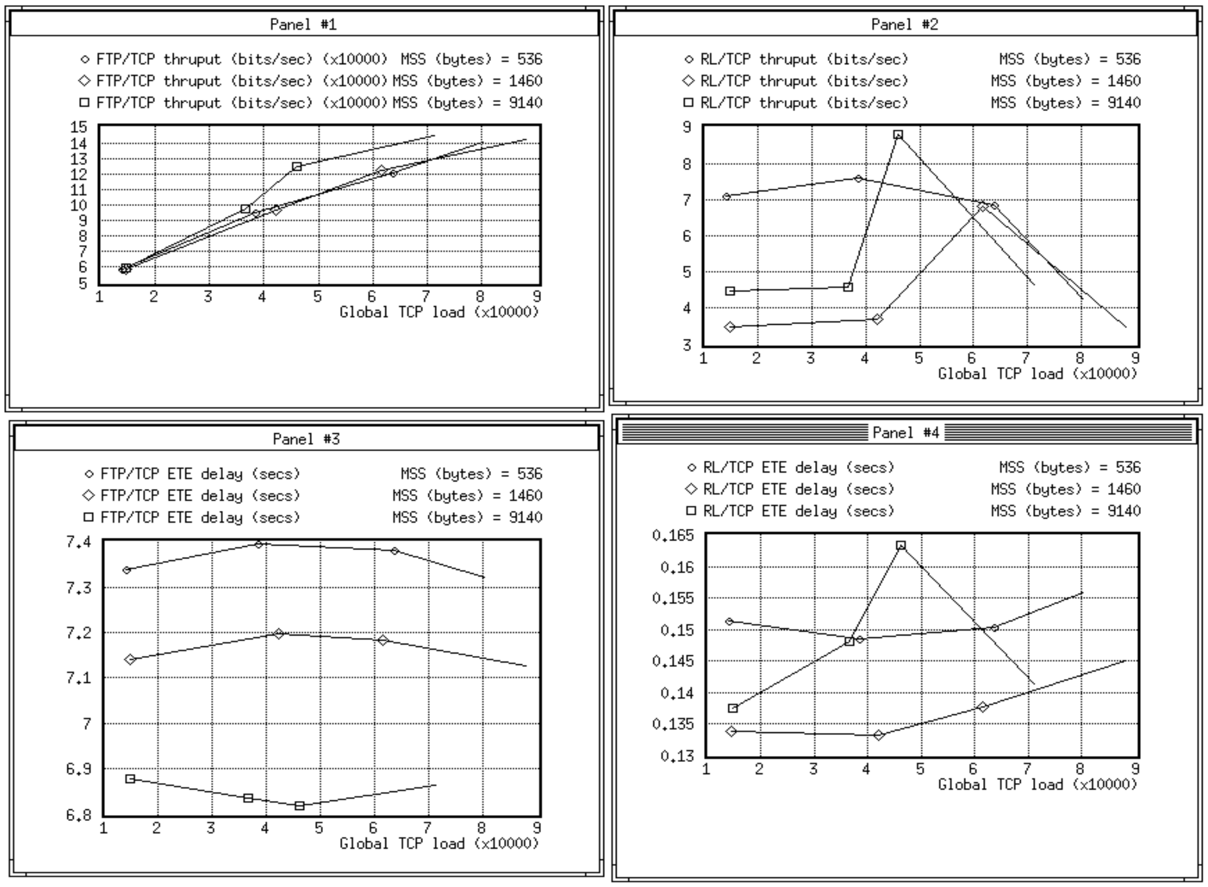

Figure 6: Impact of TCP MSS on throughput

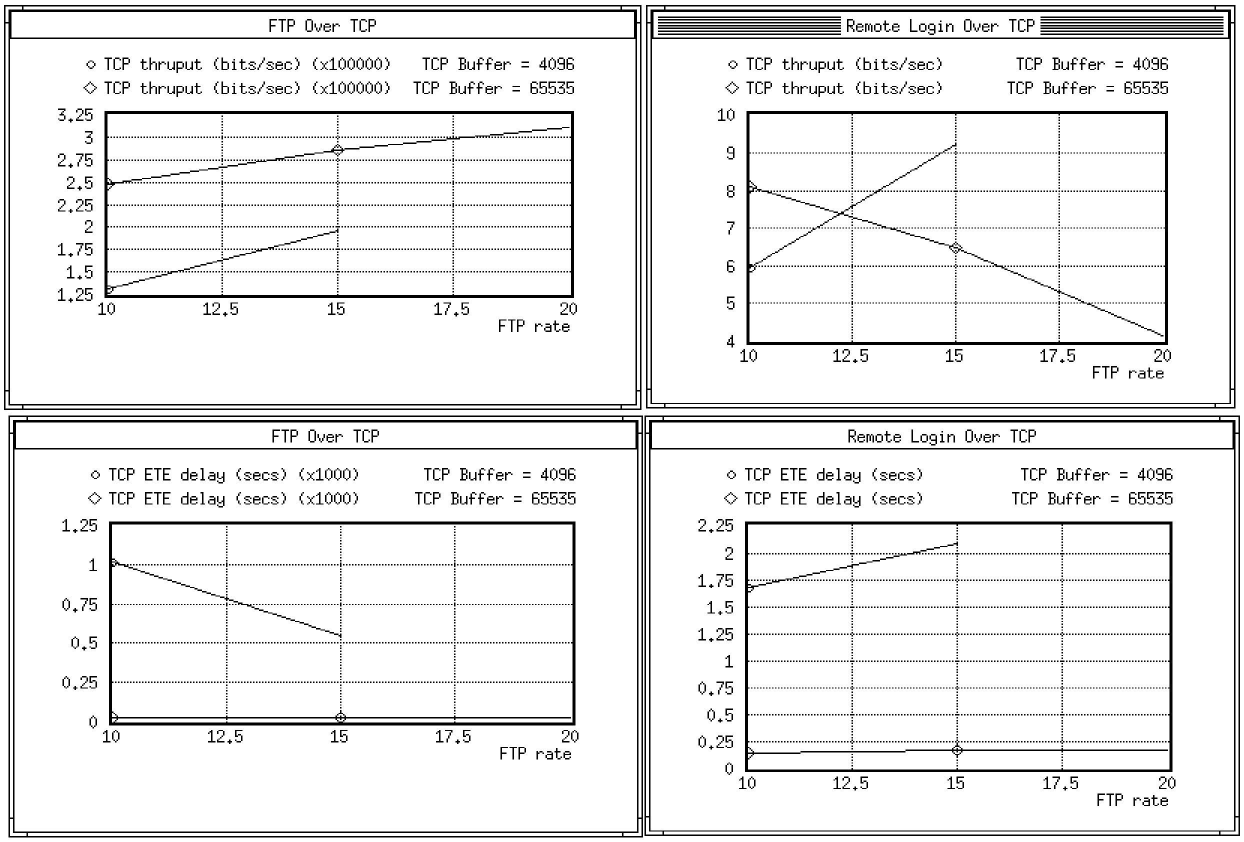

Figure 7: Impact of TCP receive buffer on FTP and RL throughput and delay

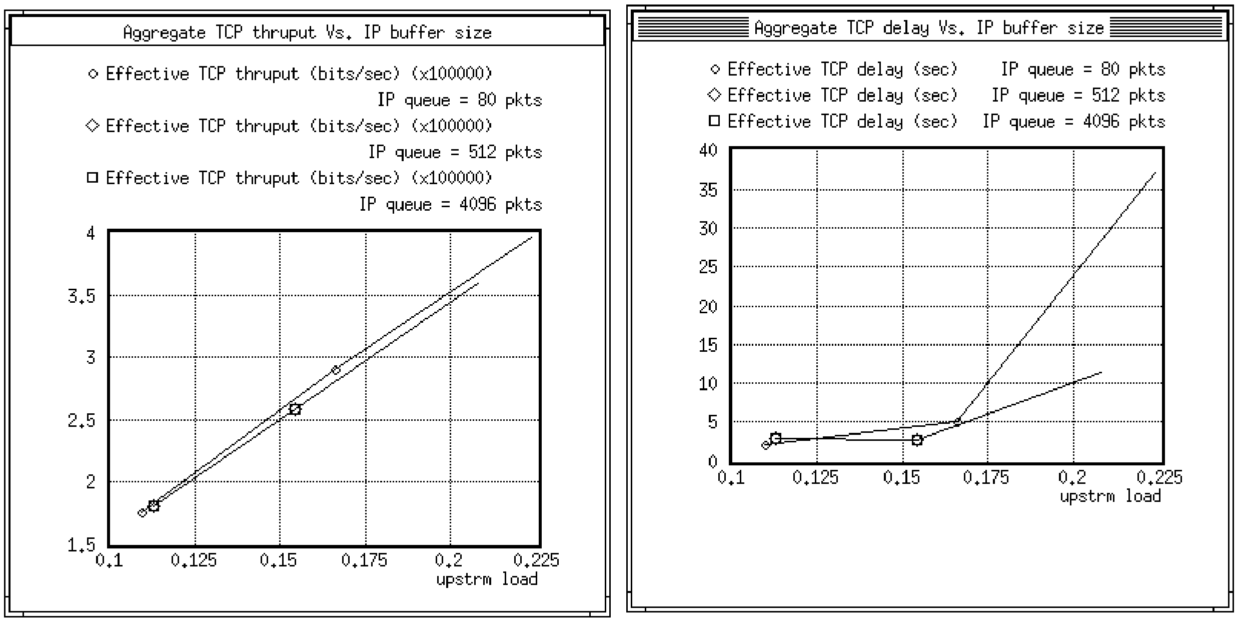

Figure 8: Impact of H/E IP buffer size on throughput

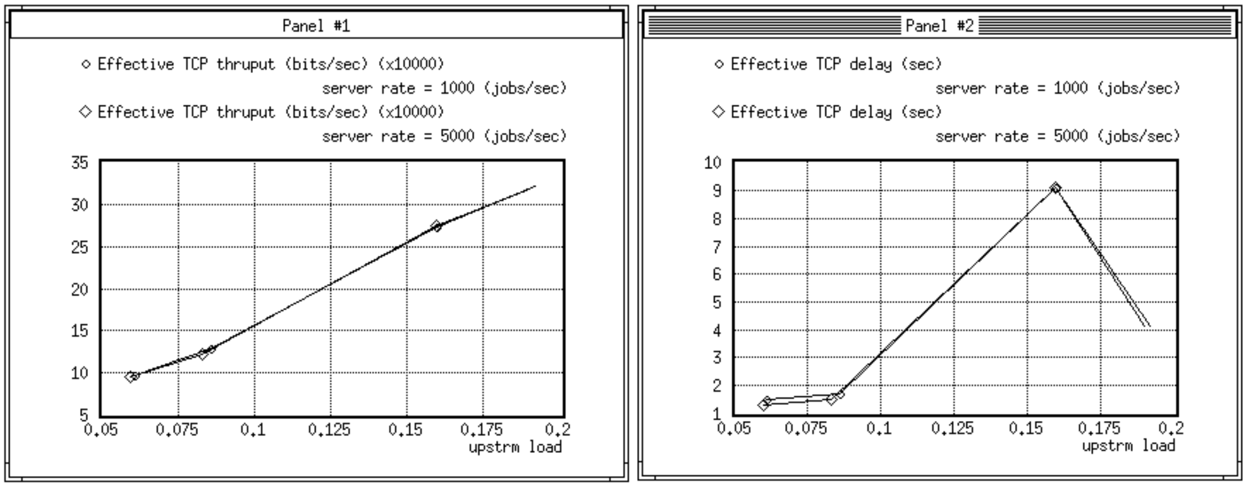

Figure 9: Impact of server processing speed on aggregate TCP throughput and delay at H/E.