Hybrid Internet Simulation Testbed

by Mingyan Liu, Manish Karir, Majid Raissi-Dehkordi,
John S. Baras

CSHCN T.R. 99-23
(ISR T.R. 99-2)

Hybrid Internet Simulation Testbed

Mingyan Liu, Manish Karir, Majid Raissi-Dehkordi,
John S. Baras
Center for Satellite and Hybrid Communication Networks
University of Maryland
College Park, MD 20742 [*]

Abstract

Internet technology as a widely accepted modern telecommunication standard has been widely extended to combine with numerous other technologies, e.g., satellite, ATM, wireless. This is what we term as Hybrid Internet. Along with this technology emerging, various enhancements and alterations of standard TCP/IP for different purposes have been proposed and studied intensively. More and more frequently we are facing the question of how to choose from these different schemes to design a system for a particular purpose, which would inevitably involve the interaction and trade-off study. We believe that simulation is a powerful tool for this type of work. In this paper, we describe our implementation of a Hybrid Internet testbed which includes a series of traffic models and TCP/IP enhancements. The goal of our work is to make a set of reusable modules upon which we can build complex systems to study the standard protocols and their variations. We also present application examples using these module components.

[*] This work was supported by the Center for Satellite and Hybrid Communication Networks, under NASA cooperative agreement NCC3-528

1 Introduction

Internet technology as a widely accepted modern telecommunication standard has been widely extended to combine with a variety of different technologies such as satellite, which results in an integration of technologies – Hybrid Internet.

However, traditional standard TCP can lead to very poor performance when used with other technologies. Take TCP over satellite for example, due to long-delay, large delay-bandwidth product and high BER of satellite links, standard TCP congestion control and retransmission strategy become inadequate or even inappropriate. Various enhancements to standard TCP for different purposes have been proposed and studied intensively [1, 2, 3]. At the same time, emerging of technologies also results in a variety of new applications and more complicated traffic patterns.

More and more research has shown that simple Poisson traffic assumption is far from accurate for real Internet traffic and thus may result in invalid analysis and conclusion. Poisson assumptions greatly underestimate the burstiness in TCP traffic, instead, they are shown to be self-similar. According to [4], the only thing that is close to Poisson in Internet traffic is the user initiated TCP session arrivals, such as remote login and file transfer, and that the packet/data arrivals within each session is much burstier than Poisson. Therefore, we need more accurate source models than Poisson assumption for simulation purposes.

Motivated by the above, our work is aimed at building a set of more realistic source models and protocol models. The simulation environment we choose to use is OPtimized Network Engineering Tool (OPNET). The advantage of using OPNET is that there are existing TCP/IP protocol models and this provides us with the base to build the enhanced models upon.

In this project, we implemented a generic on-off source model, which we used to build a self-similar traffic generator. The correctness of the self-similar traffic generator is tested as we describe in detail in later sections. We also combined the self-similar traffic source with the OPNET HTTP module so that we can generate HTTP files whose sizes are long-range dependent. We also have a model generating traffic that is of fractional brownian motion. Other traffic models we have include Auto-regression, Weibull, Log-normal and Markov Modulated Possion Process.

Figure 1 is the module stack of TCP/IP implementation of client in OPNET [5].

In building protocol models, our work adopts this module stack structure and maintains the same module interfaces between layers. Our major focus is on “tcp” and “ip” modules of this stack, which are replaced by new ones in which enhancement features are implemented with minimal changes to module interfaces, so that these modules can be easily replaced and reused. We will frequently refer back to this picture in later sections.

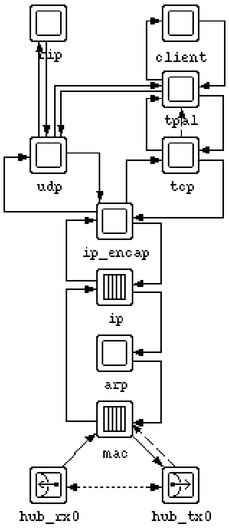

Figure 1: TCP/IP protocol stack

Protocol enhancements to TCP include Fast retransmit and Fast recovery, Selected ACKnowledgment (SACK), Forward ACKnowledgment (FACK), window scaling option, fine resolution time stamp, and TCP spoofing/splitting. Modification to IP include flow classification module, scheduling algorithms like First Come First Serve(FCFS), Round Robin(RR), Start-time Fair Queuing(SFQ), and buffer management schemes like Tail Drop, Drop from Front, Longest Queue Drop(LQD), Random Longest Queue Drop(RLQD), Random Early Discard(RED) and Probabilistic Fair Drop(PFQ). In addition to that, we also implemented Forward Erasure Correction(FZC) protocol booster. Modifications are made to either "tcp" module or "ip" module. The spoofing/splitting model also includes changes to "ip_encap". In most cases new promoted model attributes are required at the network or simulation level [5]. Some of these features have been implemented by Mil3 in the newest version of OPNET to be released. We started most of this work when only version 4.0 was widely available.

This paper is organized as follows. In Section 2 we describe the components we implemented in OPNET as well as tests that we did to verify the correctness of these modules. We give an application example using some of these modules in Section 3. Section 4 concludes the paper and proposes future work.

2 Testbed Components

In this section, we present the simulation model and experimental results on selected components. We did not provide detailed description on the ones that we think are simple and well-known. However, we give references for interested readers.

2.1 Markov Modulated Poisson Process (MMPP) traffic source

A general n-state MMPP is completely determined by two matrices as shown below. The state transition rate matrix Λ defines the underlying Markov chain which takes λ_{ij} as the transition rate from state i to state j. The arrival rate matrix A gives the poison arrival rates corresponding to different state of the Markov chain.

$$\Lambda = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdot & \cdot & \lambda_{1n} \\ \lambda_{21} & \lambda_{22} & \cdot & \cdot & \lambda_{2n} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \lambda_{n1} & \cdot & \cdot & \cdot & \lambda_{nn} \end{bmatrix},$$

$$A = \begin{bmatrix} a_{11} & 0 & \cdot & \cdot & 0 \\ 0 & a_{22} & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & a_{nn} \end{bmatrix}.$$

A two-state MMPP model was implemented using the above formula, with user definable parameters. This model can also be easily extended to multiple states.

2.2 Self-similar traffic source

This model is an aggregation of multiple on-off sources. Each of these on-off source models generates packets at a constant rate during the on period and go back to silent during the off period. The duration of on and off periods are of Pareto distribution, parameter of which is tunable.

Here we are showing the traffic traces of a source consisting of 20 such on-off source models in Figure 2 and the zooming-in trace (on a different time scale, Figure 3). The horizontal axis is time in second and the vertical axis is the packet count for every second.

To verify the model, we collected the data traces generated by OPNET and estimated the parameter used to generate the traffic. The estimator uses the Maximum Likelihood Whittle Estimation and Fractional Gaussian Noise model. We used a parameter of value 1.2 for the Pareto distribution to generate the traffic, which translates to H (Hurst Parameter) of 0.9. The estimated H we got from the Whittle estimator is 0.9035.

2.3 Fractional Brownian Motion(FBM)

The number of arrivals up to time t is [6]

$$X(t) = mt + k \cdot B_H(t),$$

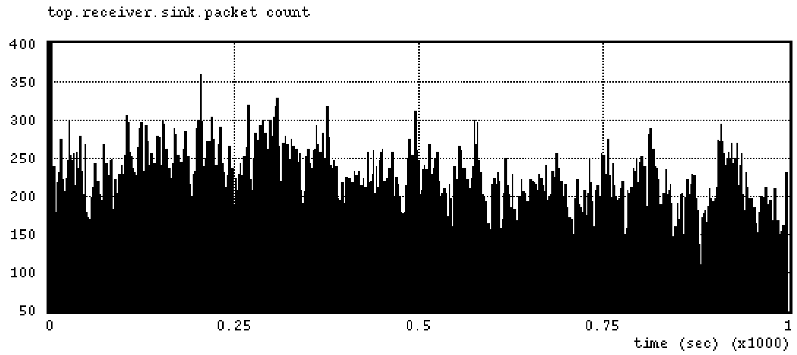

Figure 2: Traffic trace of 20 on-off sources

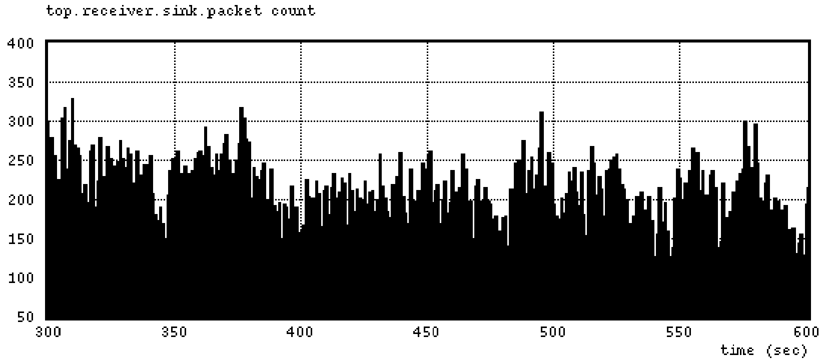

Figure 3: Zooming in of 20 on-off sources

where m is the rate and $B_H(t)$ is Fractional Brownian Motion, This arrival process is modeled using dynamic processes.

The root process is a $M/G/\infty$ model in which the number of busy servers are counted whenever there is an arrival or departure. Service time is Pareto. For every fixed small interval, the average number of busy servers n are calculated over that interval and passed on to the child process. The child process then generate packets at a constant rate of $m+n$, until the beginning of next interval when a new value of n is calculated and process starts over again.

2.4 Other Traffic Models

Other traffic models Auto-regression, Pareto, Weibull and Log-Normal. These models are simple and easy to implement (they can be made into function calls), and we will not discuss in detail here.

2.5 Fast Retransmit Fast Recovery and SACK/FACK

These options are also implemented in tcp_conn process. Details regarding Fast Retransmit Fast Recovery(FRR) can be found in [7], and for SACK, [8]. Apart form SACK, we need a mechanism to use information provided by SACK. Forward Acknowledgment algorithm [9] is known to be the most efficient algorithm to use SACK information for fast recovery in case of multiple segment losses. We implemented both.

In FACK, the start of the retransmission phase is at most after receiving three duplicate ACKs but can happen sooner by using the SACK information. After the loss detection, the congestion window is halved and the lost segment is retransmitted. If following this retransmission the available transmit window is still not zero, the probable further lost segments which are reported by the received SACK blocks are transmitted before new data. This is the main difference between FACK and FRR algorithms. In FRR, since no information on additional segment losses is available, after the retransmission of first lost segment the new segments are sent. It is not until after receiving another set of three duplicate ACKs that the second lost segment is retransmitted.

Figures 4-6 show the behavior of the different algorithms with sent sequence number, received sequence number and the congestion window size, from top down, respectively. In Figure 4 the behavior of the original TCP protocol is shown after a segment loss. Figures 5 and 6 show the response of FRR and FACK algorithms to multiple segment losses respectively.

In these experiments two sets of deterministic segment losses occur where in each set three segments are dropped. In both cases the exponential growth of congestion window during the initial slow-start and the linear growth during the congestion avoidance phase are observable. Before the first loss, the connection has reached the steady state with congestion window taking its maximum value. After the detection of loss congestion window is halved and the lost segment are retransmitted.

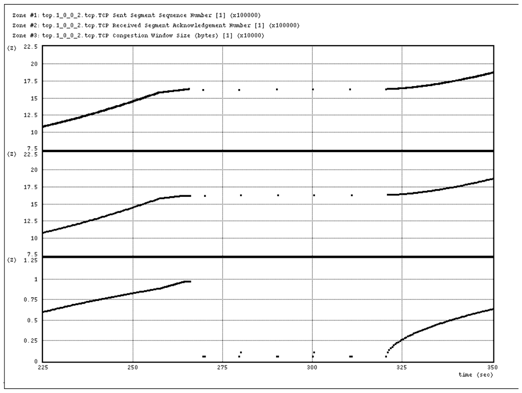

Figure 4: Plain TCP with single loss

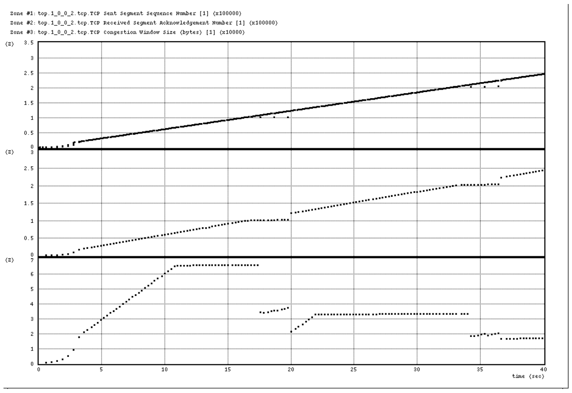

Figure 5: Fast retransmit fast recovery with multiple losses in consecutive windows

2.6 Large Window and Time Stamp Options

[3] Both these options are implemented in tcp_conn process and involves relatively minor changes. In both cases, TCP segment packet format were changed to include a window scaling factor field in the large window option and to include time stamp and time stamp echo fields in the time stamp option.

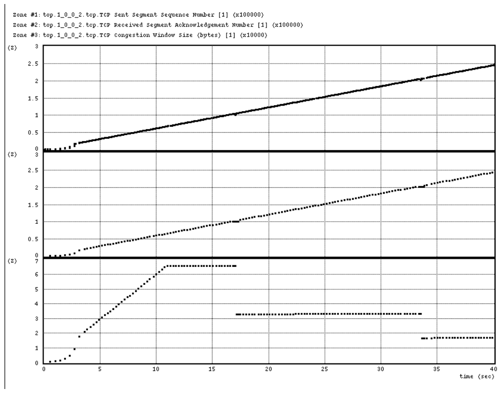

Figure 6: FACK with multiple losses in consecutive windows

2.7 Scheduling and Buffer Management at IP Layer

In order to evaluate various scheduling and buffer management strategies, and how they may improve TCP/IP performance, we implemented a flow classifier that performs per flow queuing at IP layer.

Drop from Tail is the default buffer management scheme in OPNET. We have added Drop from Front proposed by [10], which is based on the interaction of buffer management policies and TCP's fast recovery fast retransmit mechanism. If a packet is discarded at the head of the queue then duplicate acknowledgments are sent one buffer drain time earlier, triggering TCP's fast recovery fast retransmit instead of timeouts. We also added the option of Longest Queue Drop (LQD) [10], which suggest that in the presence of per flow queuing, the drop from front strategy can be applied to the longest queue. A modification of this scheme is Random Longest Queue Drop (RLQD), which performs LQD on a randomly chosen queue from a set of non-conforming queues. We also implemented Probabilistic Fair Drop (PFD) proposed in [11], which drops packets from the queue which utilizes the maximum normalized buffer share.

First Come First Serve is the default scheduling available in OPNET IP layer. We added Round Robin scheduling as well as Start-time Fair Queuing (SFQ). This is an algorithm which approximates Weighted Fair Queuing, but is computationally less complex and demonstrates better bounds on worst case delay and short term unfairness [12].

2.8 Protocol Booster

By definition, a Protocol Booster adds, deletes, or delays messages of an existing protocol, but does not originate or terminate that existing protocol. Boosters are transparent to the protocol being boosted. Boosters are robust protocol adaptors that can reside anywhere in the network or end systems and are design to dynamically improve(boost) the performance or features of an existing protocol. More information can be found in `http://govt.argreenhouse.com/pboosters/` and [13].

We implemented the Forward Erasure Correction(FZC) [14] booster in OPNET. This protocol booster was designed to enhance performance of TCP/UDP over wireless channels. It includes two elements: the sender adds parity packets into the data packet stream, and the receiver regenerates missing/lost packets from the parity packets. Accordingly this has two implications: the reduced numbers of losses and the reduced "goodput" because of parities. This is implemented in OPNET by modifying two pipeline stages: the error correction pipeline stage and the transmission delay pipeline stage. We simulate the effect of this protocol booster by specifying a level of correction, which is a model attribute, and the reduction in data rate resulting from it.

Figure 7 is an example of how using the FZC booster increased the number of correctly received packets.

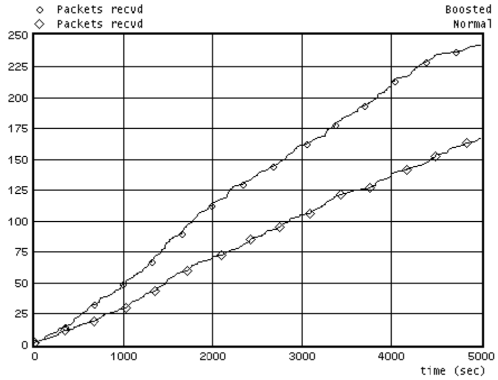

Figure 7: Packets received with/without booster

2.9 TCP spoofing and splitting

By definition, “spoofing” usually means faking an IP address and “splitting” refers to breaking up a TCP connection. As it is impossible to implement splitting without spoofing, we implemented both and both terms are used interchangeably here. Connection splitting is normally used on a ground gateway guarding the satellite channel. The motivation behind is that we want the long delay of satellite channel to be transparent to end users, and that the satellite channel be able to have different TCP options from that of terrestrial part of the TCP connection. Therefore the gateways should have the functionality of splitting the end-to-end TCP connection, faking the IP address of end hosts and acknowledging end hosts as if it were the other end of the connection, and storing data to transfer them onto the next connection. By splitting the end-to-end connection, we can essentially have different retransmission, congestion control and flow control schemes at different parts along the connection.

The way we implemented spoofing can be best explained in the following diagrams. Figure 8 describes the connection setup and data transfer procedures. Figure 9 describes the connection closing procedure. In both figures we have two end hosts H1 and H2, and a gateway G. We see that the gateway acknowledges both end hosts on behalf of the other.

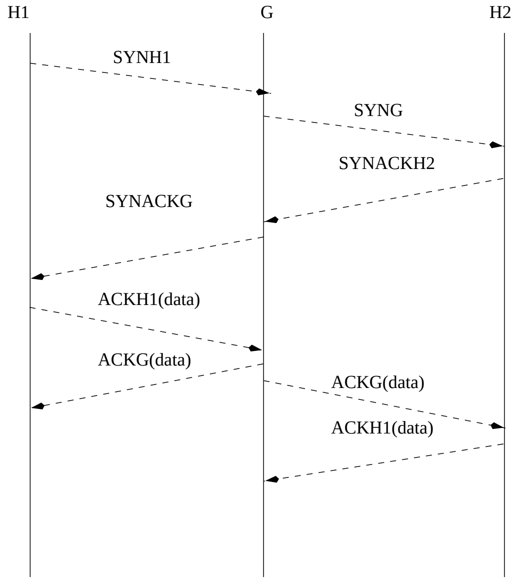

Figure 8: Connection setup and data transfer

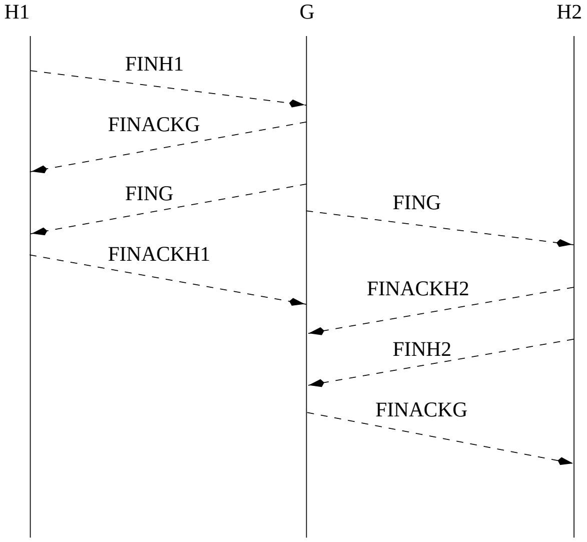

Figure 9: FACK with three losses in one window

This model was implemented by modifying three modules: “ip”, “ip_encap”, and “tcp”. And to use it, it would require that a gateway equipped with a TCP layer, as oppose to normal gateways that only have layers up to IP.

3 Application Example – DirecPC

In this section, we give an example of using some of the modules we built – the DirecPC application model, as shown in Figure 10.

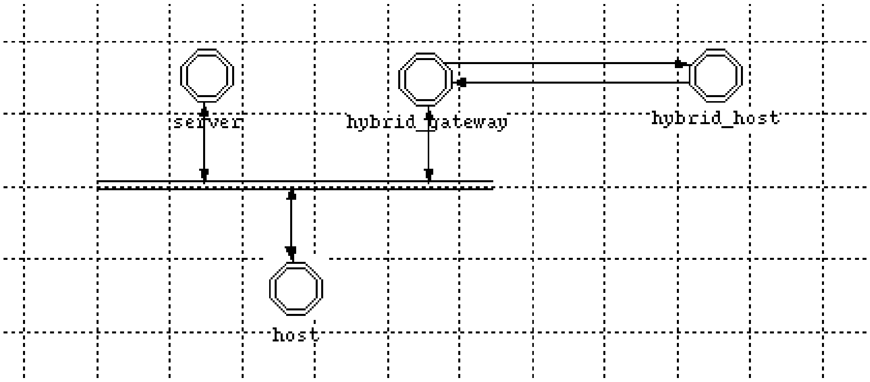

Figure 10: DirecPC Setup

The hybrid host sends request to hybrid gateway via the reverse channel which can be a telephone line or wireless link. The hybrid gateway gets the request and forwards it to the server, gets back data and delivers them to the hybrid host via the forward satellite channel. We put TCP spoofing module into the hybrid gateway to let it acknowledge the hybrid host independently from the server. This way the delay caused by the satellite channel can be hidden from the host.

Figure 11 shows the RTT seen by the normal host and the hybrid host. We can see that after initial delay, the round-trip time seen by both is the same.

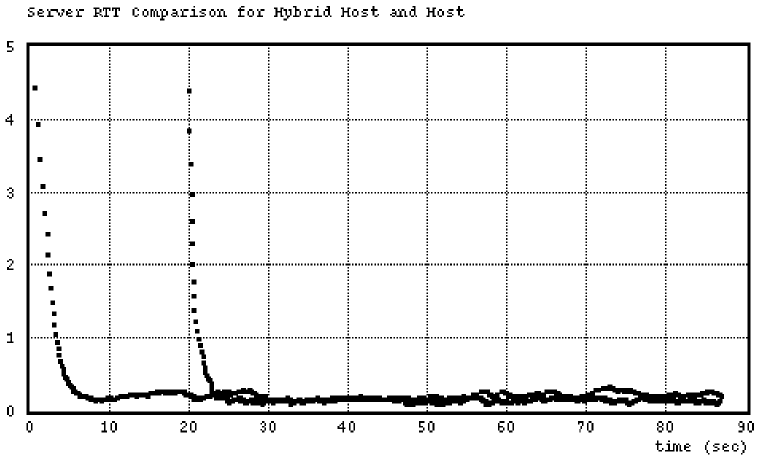

Figure 11: RTT comparison

We also got improved throughput when applying large window option to the satellite link. Similar results and more applications can be found in another paper "A Simulation

Study of Enhanced TCP/IP Gateways for Broadband Internet over Satellite" by Karir, Liu, Barrett and Baras.

4 Conclusion and Ongoing Work

In this paper we presented our work in building simulation modules including traffic generators and TCP/IP protocol enhancements and protocol boosters. We described testing results of some of our work and also gave an example of how reusing these modules can help build up complicated and customized systems.

Currently we are working on validation and verifying applicability of difference traffic models. Our approach is to take real traffic traces, estimate parameters for a specified probabilistic model, use the estimated parameters as input to the corresponding simulation model to generate duplicated traffic traces, and finally we use goodness-of-fit test to examine how close the simulation trace can approximate the real trace. We are also actively working on implementing Ad-Hoc protocols and ACK compression and ACK reconstruction protocol booster.

References

[1] Hugo Grosmangin and John S. Baras. TCP enhancements for the integration of satellite links in then Internet–Modeling and simulation study of Tahoe, Reno and SACK TCP behavior. Technical Report TR 97-212r2, Center for Satellite and Hybrid Communication Networks, Uni versity of Maryland, College Park, 1997.

[2] Lawrence S. Brakmo and Larry L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Interne t. 1991.

[3] R. Braden V. Jacobson and D. Borman. TCP Extensions for High Performance. *IETF RFC 1323*, 1992.

[4] Vern Paxson and Sally Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Trans. on Networking*, 3(3):226–244, 1995.

[5] Mil 3 Inc. *OPNET Modeler, Modeling/Vol.1*, 1997.

[6] G. Samorodnitsky. *Stable non-Gaussian random processes: stochastic models with infinite variance.* Chapman & Hall, 1994.

[7] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. *IETF RFC 2001*, 1997.

[8] S. Floyd M. Mathis, J. Mahdavi and A. Romanow. TCP Selective Acknowledgement Options. *IETF RFC 2018*, 1996.

[9] Mattew Mathis and Jamshid Mahdavi. Forward Achnowledgment: Refining TCP Congestion Control. *Computer Communication Review*, 26(4), 1996.

[10] T. V. Laxman, A. Nierhard, and T. J. Ott. The Drop from Front Strategy in TCP over ATM and Its Internetworking with Other Control Features. *Proc. INFOCOM*, 3, 1996.

[11] R. Vaidyanathan. Issues in Resource Allocation and Design of Hybrid Gateways. *MS Thesis, University of Maryland*, 1999.

[12] P. Goyal, H. Vin, and H. Cheng. Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. *IEEE/ACM Trans. Networking*, 5(5):690–704, 1997.

[13] D. C. Feldmeier, A. J. McAuley, J. M. Smith, D. S. Bakin, W. S. Marcus, and T. M. Raleigh. Protocol Boosters. *IEEE Journal on Selected Areas in Communications*, 16(3):437–444, 1998.

[14] D. Bakin, W. Marcus, A. McAuley, and T. Raleigh. An FEC Booster for UDP Application over Terrestrial and Satellite Wireless Networks. *Proceedings of International Mobile Satellite Conference (IMSC 97), Pasadena, CA*, 1997.