Performance Analysis of TCP in a Reliable Connections Environment with UDP Flows using OPNET Simulator

Friday Yakubu, S.E Abdullahi and P.E Aigbe

Abstract— The widely use transport protocol on the Internet is TCP/IP and its usage is mainly based on its high dynamic nature of adaptability to any kind of network capacity. The Internet technology today runs over large different link technologies with vastly different characteristics. In spite of the recent sudden increase in accessibility of broadband Internet access to improve the transport protocols communication, business organization, computer laboratory in learning environment, and public organization have fairly small-bandwidth links compare to the network environment that host the requested content. With the explosive growth of the Internet application traffics, the network traffic congestion grows and available limited resources becomes a bottleneck. The performance of the network eventually decline because of traffic congestion among other challenging issues, bringing the need for outstanding means of improving the TCP performance. Therefore, it becomes necessarily for any kind of network environment, to carry out a TCP performance analysis and identify the possible means of obtaining an optimal performance. This paper seeks to reflect the performance analysis of TCP in a network environment of multiple reliable connections with UDP traffic flows.

Keywords - TCP Implementations, TCP performance, Application Traffics, OPNET Simulator, Packet Dropped, Retransmission Count

1. Introduction

The growth in network technology has made the internetworking to be flexible enough to incorporate the changing network environments of the past few decades. The Internet technology today runs over a large different link technologies with vastly different characteristics in terms of bandwidth and communication rate. With the explosive growth of the Internet, as users need for accessing the Internet increases, in a limited Bandwidth environment, the network traffic congestion grows and available limited Bandwidth shrinks. The performance of the network eventually decline because of traffic congestion among other challenging issues, bringing the need for viable solutions. Therefore, it becomes necessarily to study the performance of TCP at the individual flows share link. This paper seeks to carry reflect the performance analysis of TCP in a network environment of multiple reliable connections with UDP traffic flows.

2. Transmission Control Protocol and User Datagram Protocol

According to Gilbert, TCP is a reliable connection-oriented protocol that includes a built-in capability to regulate the flow of information, a function referred to as flow control [7]. The flow of information is managed by TCP, by increasing or decreasing the number of segments that can be outstanding at any point in time. For example, under periods of congestion when a station is running out of available buffer space, the receiver may indicate it can only accept one segment at a time and delay its acknowledgment to ensure it can service the next segment without losing data. Conversely, if a receiver has free and available buffer space, it may allow multiple segments to be transmitted to it and quickly acknowledge the segments [7]. The original TCP specification is described by Postel [12].

On the other hand, User Datagram Protocol (UDP) is a connectionless, best-effort, non-error checking transport protocol [7]. There is no handshaking between sending and receiving transport layer entities before sending a segment. UDP was developed in recognition of the fact that some applications may require small pieces of information to be transferred. The use of a connection-oriented protocol would result in a significant overhead error checking to the transfer of data. Therefore, UDP transmits a piece of information referred to as a UDP datagram without first establishing a connection to the receiver. The protocol is also referred to as a best-effort protocol [7]. To ensure that a series of UDP datagram are not transmitted into a black hole if a receiver is not available, the higher layer in the protocol suite using UDP as a transport protocol will wait for an acknowledgment. If one is not received within a predefined period of time, the application can decide whether to retransmit or cancel the session.

3. TCP Implementations

The original TCP implementations used window size based flow-control to control the use of buffer space at the receiver and retransmission after a packet drop for reliable delivery. The implementations did not include dynamic adjustment of the flow-control window in response to congestion collapse such as classical congestion collapse [11] due to unnecessary retransmission of segments, fragmentation congestion collapse [9] resulting from Maximum Transfer Unit (MTU), and undelivered-segments congestion collapse that occurs when networks overloaded with packets that are discarded before they reach the receiver [3].

The exponential growth in the Internet usage has caused a great increase in the number of TCP implementations. However, the development of TCP must handle changes that may enforce any network setup into congestion collapse [4]. TCP has experienced number of changes in its primitive design, during its development process. Many versions of TCP exist today and each TCP implementation employs congestion control algorithms as described in section 2.4. The available implementations include TCP Tahoe and TCP Reno.

3.1 TCP Tahoe/Reno

The fundamental algorithm for congestion avoidance and control was first introduced by Jacobson [8]. The implementation of the algorithm, called TCP Tahoe, was based on the principle of *conservation of packet* and it has introduced significant improvements for working over a shared network [5]. When TCP connection is established at the available resources capacity, a packet is not set in transit into the network until sent packet leaves the network. In other word, a packet can be set in transit only when the sender receives an acknowledgement from the receiver indicating successful delivery of the packet. The acknowledgements clock the outgoing packets because an acknowledgement means that a packet was taken off the transmission medium and resources [5]. TCP Tahoe includes Slow-Start, Congestion Avoidance and Fast Retransmit. The Slow Start algorithm was introduced to control transmission following any detected congestion.

The modification of how TCP Tahoe response to detection of loss packet through duplicate ACKs is what leads to the implementation of TCP Reno [6]. The concept is that if packet loss is detected via a duplicate ACK before TCP timeout expired, TCP sender retransmits the loss packet by performing a Fast retransmit algorithm without waiting for RTO and enters into its Fast recovery stage until the non duplicate ACK that acknowledges the entire transmit window of data is received. If it does not receive such an ACK, TCP Reno experiences a timeout and enters the slow-start state as implemented in TCP Tahoe. TCP Reno adds Fast Recovery to TCP Tahoe.

3.2 TCP New Reno/Vegas

TCP New-Reno is a modified version of TCP Reno. In TCP Reno, TCP sender would leave Fast Recovery on the receipt of first ACK that acknowledges new data. This algorithm recovers loss packet efficiently, if there is only one lost packet. However, the algorithm fails from multiple packet recovery within a sliding window [6]. Therefore, an enhanced version of Fast recovery algorithm was proposed to address the short comings in TCP Reno [6]. The algorithm proposed that a smaller value for threshold causes early termination of the slow start stage and also slow increase of the congestion window. In the other way round, a larger value causes the sender to overwhelm the network with packets, causing congestion.

The idea behind TCP New Reno is that during the Fast Recovery, the TCP sender does not exit its Fast Recovery stage on receiving partial ACK until all the data packets which were outstanding at the time entered Fast Recovery. Thereby recovering from multiple packet loss in a single window of data and exits its Fast Recovery phase either on receiving the ACK that acknowledges entire data within that window or on occurrence of retransmission timeout.

TCP Vegas was presented before New Reno, SACK and FACK were developed [2]. Vegas is a TCP implementation which is builds on the fact that proactive measures rather than reactive against packet losses. TCP Vegas is fundamentally different from other TCP variants in that it does not wait for loss to trigger congestion window reductions. It employs an alternative strategy in that it tries to predict when congestion is about to happen and adapts its window to compensate. This is a proactive approach as it attempts to reduce its sending rate before packets start being dropped by the network. TCP Vegas keeps track of the time each segment is sent. When an ACK arrives, it estimates RTT as the difference between the current time and the recorded timestamp for the relevant segment. TCP Vegas has not been widely implemented and is not universally accepted by the Internet community and is still a subject of much controversy.

3.3 TCP SACK/FACK

The default TCP acknowledgment behavior is to acknowledge the highest sequence number of in- order bytes. This default behavior is prone to cause unnecessary retransmission of data, which can exacerbate a congestion condition that may have been the cause of the original packet loss. A proposed modification to TCP, selective acknowledgement (SACK) allows a TCP receiver to acknowledge out-of-order segments selectively rather than just cumulatively acknowledging the last correctly received in-order segment [10]. SACK option is used by TCP receiver to inform the TCP sender that a non contiguous segment of data has been received and it is queued. So the sender need retransmit only the packets that have actually been lost [6], [1]. When a retransmitted packet is itself dropped, the SACK implementation detects the drop with a retransmit timeout, retransmitting the dropped packet and then slow-starting. If all of the outstanding data is ACKed (received ACK sequence is greater than *recover*), the sender exits fast recovery and continues in congestion avoidance. To use SACK, both the TCP sender and receiver must support the feature and must enable it by negotiating the SACK-Permitted option during the connection establishment. Adding the SACK option to the TCP flavours such as TCP Tahoe or Reno, does not change their basic underlying congestion control algorithms. The information about missing sequence numbers is transmitted to TCP sender using three SACK blocks with each ACK, using the rules outlined by Mathias [10].

The Forward Acknowledgment (FACK) algorithm proposed, aims at better recovery from multiple losses [10]. In FACK, TCP maintains two additional variables: the forward-most segment that has been acknowledged by the receiver through the SACK option and retransmitted data that reflects the amount of outstanding retransmitted data in the network. Using these variables, the sender can estimate the actual quantity of outstanding data in the network and can inject new data if allowed by receiver's window. TCP FACK regulates the amount of outstanding data in the network to be within one segment of CWND, which remains constant during the Fast Recovery phase.

4. Analysis using Simulator

The use of network simulator to emulate an existing network environment has many merits. With the network simulator, the elements properties, and the emulated network architecture can be controlled. The predefined test configurations may also run automatically, which makes the execution of the performance measurements more convenient. The most widely known simulator, OPNET was selected and used for the analysis. OPNET is a graphic network simulator that provides a set of tools for network modelling, displaying statistics. Many predefined types of nodes are present and almost all widely used protocols and technologies are supported. The supported operating system is Windows.

4.1 The Targeted Environment

The target environment of study is a LAN network environment generating UDP and TCP traffic flows. The network environment consists of ten client hosts each communicating with one of the three different server hosts via a wireless link and a last network element as shown in figure 1.

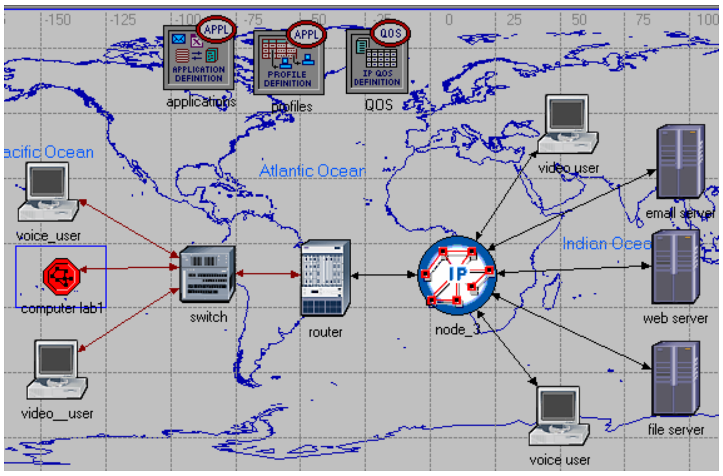

Figure 1: Emulation environment.

The study concentrated on the behaviour of a client TCP in the LAN, in the presence of multiple TCP and UDP traffic flows. The LAN has bandwidth of 100 megabits, wireless link data rate of 64,000 bps and router buffer of 2MB. The network is assumed to provide traffic flows significantly faster than what the wireless link can transmit. Figure 1 illustrates the elements which are emulated from the target network environment.

OPNET software was used to emulate the target network environment. In the emulation environment, performance tests were made using ten client hosts on the network and two different hosts were set to generate voice and video streams. Each of the ten hosts generates TCP traffic. Traffic shaping was not imposed in the network set up. The traffic flows are forwarded base on Best-effort scheme.

The network hosts were assumed to be running the Windows operating system. Each of the client hosts and the server hosts are the TCP endpoints communicating with each other using TCP connection. The TCP enhancements that are used in the performance tests are implemented only in the endpoint hosts.

The wireless link is emulated by issuing appropriate data transmission rate and propagation delay. Link layer retransmissions are emulated by causing an error delay for a packet. During the error delay no packets are released from the link receive buffer. The last-hop router queue is emulated using the input queue. The traditional first-in-first-out scheduling is used on the input queue. If there is no room for incoming packet in the input queue, emulator router discards the packet.

4.2 Internet Application traffics

The most commonly used applications in the target network environment are those that require Internet access, which we call *the Internet applications* and Internet application traffic must traverse via the last router, the IP32 cloud, and finally via the Internet Servers gateway to reach the Internet Servers. The Internet applications for the study are web browsing, video conference, and FTP, e-mail.

OPNET Modeler provides standard built-in models for software applications such as *web (HTTP)*, *e-mail* and *FTP* which can be easily configured to simulate applications used in the subnets. Thus, these applications are implemented and configured via OPNET's Custom Application feature. Table 1 summarizes the application definitions.

Table 1: Application Definitions

Application	Meaning
e-Mail (light)	Indicate light activities of e-mail application
e-Mail (heavy)	Indicate heavy activities of e-mail
FTP (light)	Indicate light file transfer activities

FTP (heavy)	Indicate heavy file transfer activities
Web browsing (light)	Indicate light browsing activities
Web browsing (heavy)	Indicate heavy browsing activities

To set-up and configure any application in OPNET, the Application Configuration and Profile Configuration modules was added to the emulated target network environment. The Application Configuration module contains the application definitions, while the Profile Configuration module contains the profiles of user behavior, e.g. describes how the users employ the applications defined in the Application Configuration module.

4.3 Metrics of the Performance Analysis

Retransmissions: One of the interesting metric for the analysis is the number of retransmission which does strongly affects the throughput of a TCP connection. When the number of retransmission is reduced, the throughput of the TCP connection most significantly improves. In this analysis, retransmission is triggered by packet drop at the last-hop router or by a retransmission timeout caused by excessive delay.

Dropped packets: another metric is the number of dropped packets which has direct effect on the number of retransmissions. This metric gives additional information needed to derive the number of unnecessary retransmissions. Because packets drop take place only at the last-hop router as a result of buffer overflow, this metric measures the severity of congestion at the last-hop router.

5. Baseline Configuration for the Performance Analysis

The main objective is to inspect the performance of TCP in a MRC with the presence of UDP traffic flows over wireless link and router of a limited buffer memory, and to inspect certain details of TCP behaviour more closely. Buffer size of 8mb and data rate of 64,000 kb were used as baseline for the performance analysis. However, three different router buffer sizes with three different data rates were later used in the analyses. The primary interest of the study is not to find exactly the optimal buffer size and data rate for the different scenarios. Hence the selection of the three different buffer sizes and data rates are to limit the number of test runs.

SACK implementation was used as the baseline technology for establishing a reliable connection. Using the same modelled environment, TCP Reno and NewReno implementations were later, at different scenarios configured. Similarly, the selection of the TCP platforms is to report which among offers better performance.

5.1 TCP Technologies Performance Analysis Using Different Router Buffer Size (ETTDB)

In theory, it is expected that when buffer size is increased, the expected result would be a significant increase on the TCP performance, because buffer overflow that leads to packet drop will decrease. The selected TCP technologies discussed in section 5 were tested on the same test-bed to see the effect of TCP technology on the performance using different router buffer size and to find possible causes of change in the behaviour. After running the three simulations with same data rate of 64,000bps, the best scenario from each simulation were selected and simulated again.

Table 2: Simulation Configuration for **ETTDD**

Simulation1: SACK		Simulation2: NewReno		Simulation3: Reno	
Scenarios	Router buffer size	Scenarios	Router buffer size	Scenarios	Router buffer size
1	8mb	1	8mb	1	8mb
2	16mb	2	16mb	2	16mb
3	32mb	3	32mb	3	32mb
4	64mb	4	64mb	4	64mb

Table 2 summarizes the TCP implementation and router buffer size configurations for the simulations. The best scenarios are scenarios2 from simualtion1, scenario3 from simulation2, and scenario3 from simulation3.

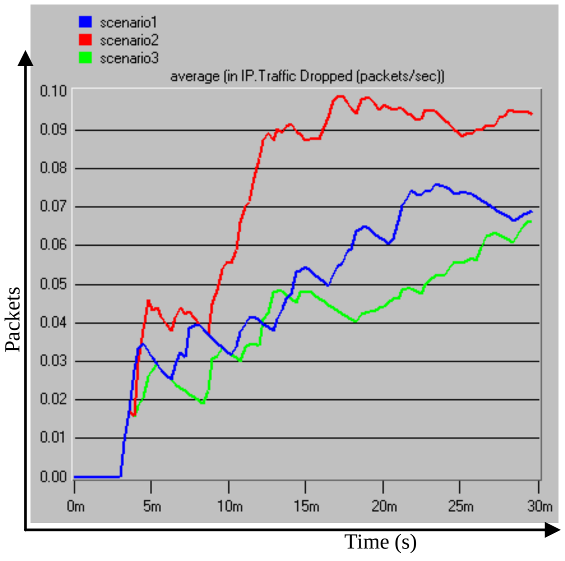

Figure 6: ETTDB Dropped Packets

Simulation1: for SACK		Simulation2: NewReno		Simulation3: Reno	
Scenarios	Data rate	Scenarios	Data rate	Scenarios	Data rate
1 2 3 4	64,000bps 128,000bps 256,000bps 512,000bps	1 2 3 4	64,000bps 128,000bps 256,000bps 512,000bps	1 2 3 4	64,000bps 128,000bps 256,000bps 512,000bps

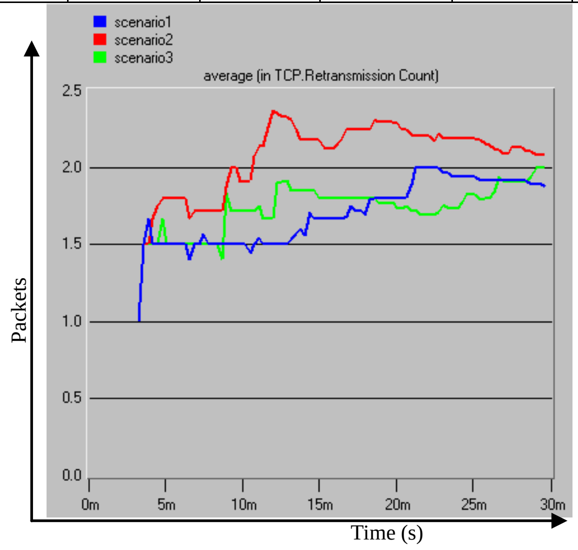

Figure 7: ETTDB Retransmission Count

In these simulation scenarios, the interest is to see how the technology implementation adapts to buffer overflows and link noise in the presence of UDP traffic connection.

The scenarios of the TCP technologies that offered best performance were selected, scenarios renamed and simulated again. The best selected scenario2 renamed as scenario1, scenario3 renamed as scenario2, and scenarios3 was not renamed. The graph pattern of the packets dropped in figure 6 showed that scenario3 using Reno implementation with 32mb offered better TCP performance followed by scenario1 using SACK technology with 16mb. The gradual increase in packets dropped and retransmission in the figures 6 and 7 indicate the effect of overlapping radio channels, signal attenuation and additional noises. They have significant impact on packet losses and retransmission.

5.2 TCP Technologies Performance Analysis Using Different Data Rate (ETTDR)

The selected TCP technologies discussed in section 5 were tested on the same test-bed to see the effect of TCP technology on the performance of the TCP using different data rate over fixed router buffer size of 16mb, and to find possible causes of change in the behaviour.

Table 3 summarises the TCP implementation and data rate configurations for the simulations. After running the three simulations with same router buffer size of 16mb, the best scenario from each simulation were selected and simulated again. The best scenarios are scenarios2 using SACK implementation with data rate of 128,000bps from simulation1, scenario3 using NewReno implementation with 256,000bps from simulation2, and scenario3 using Reno implementation from simulation3. When simulating the best scenarios selected, scenario2 from simulation1 was renamed as scenario1, scenario3 from simulation2 renamed as scenario2, and scenarios3 from simulation3 left as scenario3. Figures 8 and 9 below show the results of the simulation of the best scenarios selected.

Table 3: Simulation Configuration for **ETTDR**

When the best scenarios were simulated, NewReno technology (scenario2) with wireless link data rate of 258,000bps over a router buffer size of 16mb offered the best TCP performance followed by SACK (scenario1) technology with wireless link data rate of 128,000bps over the same router as shown in figure 9. The Reno technology with wireless link rate of 258,000 bps over the same router buffer size yielded the worse performance of the TCP.

The graph pattern of retransmission count in figure 9 indicates that using Reno technology configured in scenario1, the TCP experienced the highest retransmission count throughout the simulation period and SACK technology configured in scenario1 offered better performance of TCP followed by NewReno technology configured in scenario2.

Among the possible reasons for having such patterns of retransmission is due to packets dropped at the router as a result of buffer overflow and probably less effect of link noisy. Finally, the results of the simulations reveal that TCP Implementation using the selected different data rate over the same router buffer size of 16mb yielded different performance of the TCP in the emulated network environment.

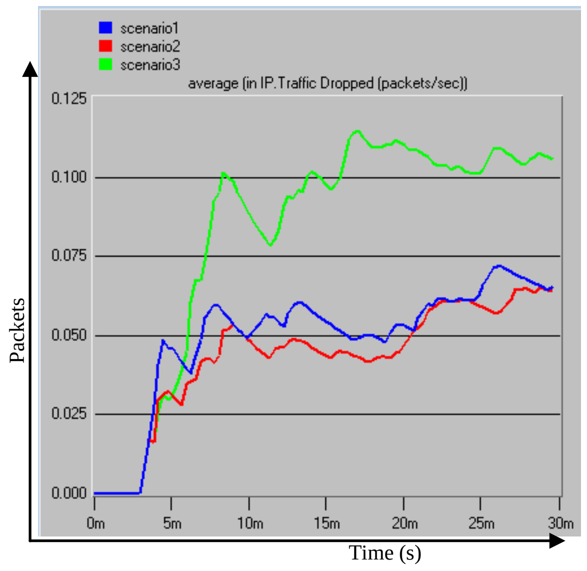

Figure 8: ETTDD Dropped Packets

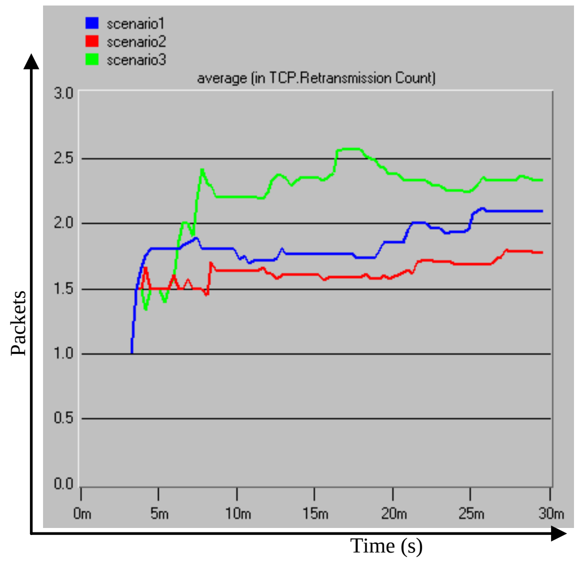

Figure 9: ETTDD Retransmission Count

5.3 Summary of the finding

The focus of this study was to analyze the effects of TCP implementation in a network environment of MRC with UDP traffic flows. In addition, the study also aimed to help researchers who are interested in improving TCP or UDP performance by giving them an accurate insight about TCP implementations behavior. The study used simulation tool OPNET. Six kinds of traffic, three TCP implementations were configured and simulated. Based on the simulation results, the performance of the TCP implementations in the emulated network environment are not the same. SACK implementation with data rate of 128,000bps offered best performance of TCP followed by NewReno technology with data rate of 258,000bps using the same router buffer size of 16mb. Reno platform with data rate of 258,000bps using the same buffer size offered worst performance.

6. Conclusion

In spite of the recent sudden increase in accessibility of broadband Internet access, the best part of business organization, computer laboratory in learning environment, and public organization have fairly small-bandwidth links in comparison with the sites hosting beloved content. It is quite common for multiple users of the Internet to get connected at the same time, and running multiple networking applications. Based on the paper work, different type of TCP connections can give different result depending on the network environment. Using selected test cases, TCP SACK yielded the best followed by NewReno. Besides, it is observed that the rate of UDP affect TCP send rates. In the presence of UDP, there was significant decrease in the performance of the TCP.

Authors Profile

Friday Yakubu received the B.Tech. degree in Computer Science from Abubakar Tafawa Balewa University, Bauchi, Nigeria, in 2003, he obtained Masters of Information Management in 2008 and M.Sc. degree in Computer Science in 2011, from A.B.U Zaria. He currently works with Ahmadu Bello University, Zaria as Software Programmer.
Email: Yakfri@yahoo.com

Dr. S.E Abdullahi, Visiting Lecturer, Department of Mathematics, Ahmadu Bello University Zaria, Nigeria.

Aigbe Patience Erinma received her B.sc. degree from University of Benin, Nigeria in 1993, Msc. degree from Ahmadu Bello University Zaria in 2011. She presently works with A.B.U Zaria as Chief System Analyst.

References

[1] Blanton E., Allman M., Fall K., and Wang L. (2003). *A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP*. RFC 3517. Available: http://www.ietf.org/rfc/rfc3517.txt, accessed on 29/08/2010.

[2] Brakmo S.L, O'Malley W.S., and Peterson L.L.(1994). TCP Vegas: *New Techniques for Congestion Detection and Avoidance*. Proceeding of ACM SIGCOMM '94 held at Stanford, California. August, Pp. 158-181. Available:http://www.cs.umd.edu/class/spring2010/cmsc711/vegas.pdf, accessed on 25/08/2010.

[3] Floyd S. and Fall K. (1999). *Promoting the Use of End-to-end Congestion Control in the Internet*. IEEE/ACM Transactions on Networking. Vol. 7, pp. 458-472. Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.6774.pdf, accessed on 1/10/2010.

[4] Floyd S. (2000). *Congestion Control Principles*. RFC 2914. Available:http://www.ietf.org/rfc/rfc2914.txt, accessed on 20/10/2010.

[5] Floyd S. and Henderson T. (1999). *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 2582. Available: http://www.ietf.org/rfc/rfc3390.txt, accessed on 20/10/2010.

[6] Floyd S. and Fall K. (1996). *Simulation-based Comparisons of Tahoe, Reno, and SACK TCP*. Computer Communication Review, July, Vol. 26, No. 3, pp. 5-21. Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.3327.pdf, accessed on 20/10/2010.

[7] Gilbert H. (2002). *The ABC of IP Addressing*. AUERBACH, Baca Raton London, New York Washington, D.C.

[8] Jacobson V. (1988). *Congestion Avoidance and Control*, Proceeding of ACM SIGCOMM '88 held at Palo Alto, California. August, pp. 133-147. Available: http://ee.lbl.gov/papers/congavoid.pdf, accessed on 10/09/2010.

[9] Kent C. and Mogul J. (1987). *Fragmentation Considered Harmful*. Proceeding of the ACM workshop on Frontiers in Computer Communications Technology held at New York, NY, USA. November, pp. 390-401. Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.5308.pdf, accessed on 25/10/2010.

[10] Mathis M., Mahdavi J., Floyd S., and Romanow A. (1996). *TCP Selective Acknowledgment Options*. RFC 2018. Available:http://www.ietf.org/rfc/rfc2018.txt, accessed on 10/08/2010.

[11] Nagle J. (1984). *Congestion Control in IP/TCP Internetworks*. ACM *SIGCOMM* Computer Communication Review held at Palo Alto, California. June, pp. 3-5. Available:http://www.sigcomm.org/ccr/archive/1995/jan95/ccr-9501-nagle84.pdf, accessed on 15/09/2010.

[12] Postel J (1981). *Transmission Control Protocol*. Internet RFC 793. Available: http://tools.ietf.org/html/rfc793, accessed on 25/08/2010.