PERFORMANCE ANALYSIS OF LINUX TCP CONGESTION CONTROL TECHNIQUES

Usman Ahmad, Md Asri Bin Ngadi*
Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia,
UTM Johor Bahru, 81310, Johor, Malaysia.
[e-mail: usman.ahmad82@yahoo.com, dr.asri@utm.my.]
*Corresponding author: Md Asri Bin Ngadi

ABSTRACT*—Transmission Control Protocol (TCP) is a core protocol of the Internet protocol suit, which aims to provide the data reliability transmission among computers. Long distance networks spanning several continents are growing in importance and many multinational companies are now centralizing their data centers for economical reasons. While high performance of TCP in these networks is critical for the effective operation of their data centers, it is commonly reported that TCP substantially underutilized network bandwidth in these environments. Effective congestion control in the network is critical issue for the efficiency of network resources. Since after development of the TCP, many TCP congestion control techniques are proposed to solve the congestion issue for many network conditions (e.g., wired networks, wireless networks and satellite links). Now days, Linux operating system has 14 different congestion control techniques inside the TCP. The aim of this paper is to offer a comparative analysis of behavior of these 14 Linux congestion control techniques, in term of congestion window behavior, rate of packet loss and throughput in wired network scenarios.*

Index Terms*—*Congestion window, Packet loss rate, throughput, wired network

1. INTRODUCTION

The transmission control protocol (TCP) plays a critical role in the Internet infrastructure by providing a connection-oriented, reliable, byte-stream service [1]. TCP uses three way handshaking for connection establishment and four way handshaking for connection termination. Secondly, it creates a flow and error control mechanism. TCP uses a sliding window to achieve flow control and uses time-out, acknowledgment packet and re-transmission for error control. TCP was originally designed for short distance local area networks (LAN), but distance among nodes and capacity of bandwidth increased rapidly. Reliable communication and congestion control are two major functions of TCP. TCP provides congestion control mechanisms to prevent the network from congestion collapses [2,3,4,5] and for achieving high performance. To avoid such kind of congestion collapses, TCP congestion control uses packet conservation principle [2]. According to the this principle packet will not enter in the network until last transmitted data have been acknowledge or lost. To avoid the congestion sender controls the amount of data by using a variable called *(cwnd)* and this variable determine the limit that how much data need to be send by the sender. Receiver side also inform to sender about this capacity by a variable called *(rwnd)*, with the help of these two variables TCP send minimum amount of data congestion window *(cwnd)* and receiver window (*rwnd)*.

There are four basic algorithms inside the congestion control mechanism which are slow start, congestion avoidance, fast re-transmit and fast recovery. TCP slow start algorithm is used to find out the unknown capacity of the available bandwidth. In slow start phase sender increase the size of congestion window by one after receiving each ACK and after that increases exponentially per RTT. The Equation 1, shows the exponential increment after receiving ACK. The slow start exits and enter into the congestion avoidance phase when the value of the congestion window (*cwnd*) increases than the given variable called slow start threshold (*ssthresh*). In congestion avoidance phase sender side increases the size of congestion window by 1/cwnd after each ACK. That means congestion window increases gradually by one packet per every round trip time (RTT), on the other hand the size of congestion window decreases by half of its original size after

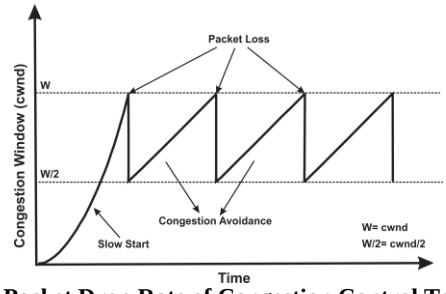

Figure 1: Packet Drop Rate of Congestion Control Techniques

$$ACK : cwnd \leftarrow cwnd + 1 \qquad (1)$$

receiving three duplicates or packet loss.The typical behavior of congestion window is shown in Figure 1. The Equation 2 and 3 shows the behavior of congestion window during congestion avoidance after receiveing ACK and after loss event.

$$ACK : cwnd \leftarrow cwnd + \frac{1}{cwnd} \qquad (2)$$

$$Loss : cwnd \leftarrow \frac{1}{2} \times cwnd \qquad (3)$$

For the reliable and better transmission, TCP must have a mechanism for detecting packet loss and re-transmit it. Before sending the data TCP associates a unique number with data called sequence number and receiver sends the ACK packet with next sequence number to the sender after receiving the packet. This is called cumulative ACK and when the packet loss, receiver send the duplicate ACK to sender with same sequence number. There is a re-transmit timer maintained by TCP, when ever sender receives cumulative ACK, TCP resets the timer. TCP assumes packet is lost when the sender does not receive any ACK during the given time and re-transmit it again. Re-transmit the data after timeout is very inefficient

because the re-transmission time out (RTO) is longer than RTT and sender sites idle and wait for the timeout without sending any data.

TCP fast re-transmission [7] mechanism reduces the time that a sender waits before re-transmitting a lost packet. Thus when receiver receive abnormal data or out of order data then it immediately sends the duplicate ACK to the sender and if sender receives three ACK that means packet loss, then the sender immediately re-transmit the lost segment to the receiver without any wait. After lost segment sent by fast re-transmit the fast recovery is used to make TCP operates in congestion avoidance phase instead of slow start, which improves the performance of TCP.

Many of high speed TCPs are proposed to improve the efficiency of network and classified into three main categories namely, loss-based, delay-based and loss-delay based. In loss-base protocols packet loss is used as indication of congestion in the network. Some proposed protocols in this category are HS-TCP [8, 9, 10, 11, 12, 13]. In delay-based protocols only delay or round trip time (RTT) information is used for the indication of congestion. Many delay-based protocols are proposed such as [14, 15, 16]. Loss and delay based protocols are combine both delay and loss information to detect congestion in the network. Some proposed protocols in this category are [17] and [18].

2. RELATED WORK

Tahoe was the first TCP variant proposed with congestion control technique. One of the main principal of its operation is that the rate of new packet sent into the network must be close to the rate at which the receiver returns the acknowledgments. In versions earlier than Tahoe sources and multiple segments into the network until the advertised receiver window is full. This over aggressive behavior may cause problem when there are slower links between the sender and receiver. TCP Tahoe includes three new techniques, which are slow start, congestion avoidance and fast re-transmit. TCP Tahoe also adds a new window to the sender side called congestion window (*cwnd*). TCP sender side must not transmit more than the minimum of the congestion window and receiver window (*rwnd*). TCP Tahoe, even with its limitations, was a major breakthrough in congestion control and played a critical part in the prevention of congestion collapses in the Internet. Tahoe sets the guidelines and principles for which virtually every new TCP implementation should respect or, at least consider. Tahoe is followed by Reno [3].

Reno used many characteristics of Tahoe but proposed two new changes to improve the performance of Tahoe. TCP Reno inceases its congestion window by Equation 4 and upon packet loss decreases it congestion window by Equation 5. Reno changes the approach of Tahoe of detection of three acknowledgments. The basic idea of Reno is that after detecting packet loss through duplicate acknowledgments, the Reno did not use the technique of Slow Start, it continues the sending data. The main thing is that sender still receiving acknowledgments and still send packets, until the load of congestion is not going to heavy. Reno introduced a new approach called fast recovery. According to Reno technique when duplicate acknowledgment will receive then fast re-transmit technique will activated and resend the lost packet. This approach is very near to Tahoe, after that Reno will start fast recovery. In the fast recovery process the half value of congestion window is saved in *ssthresh* and new value of the congestion window is set to half of the congestion window and add three times the segment size. This can be done because the three acknowledgments received by sender is like three packet left the network [2]. In Reno algorithm when re-transmit time will expire then the Reno will enter in slow start phase, which is the phase of Tahoe. The technique of Reno is better as compared to Tahoe regarding single packet loss in congestion window, but like Tahoe the Reno is not behave well when more that one packets loss in single window. Many researchers optimized the standard TCP protocol by modify the congestion control technique to prevent the network by congestion collapses. The problem of congestion collapses happens when there is no any congestion control technique and end to end flow mechanism which tells the network about lost packets [19] and this may become a very sever problem as a large capacity of bandwidth can be wasted due to these lost packets. TCP congestion control techniques perform well when the available bandwidth is low but in the presence of high bandwidth it does not perform well [20], this is due to long time to increase the congestion window size which can not fully utilize the available bandwidth [21].

$$cwnd \leftarrow cwnd + \left(\frac{1}{cwnd}\right) \qquad (4)$$

$$cwnd \leftarrow cwnd - \left(\frac{cwnd}{2}\right) \qquad (5)$$

To solve the issue many high speed variants of TCP has been proposed and implemented such as Reno [14], High Speed TCP [8], Scalable TCP (STCP) [9], BIC TCP [11,10] (HTCP), Compound TCP (CTCP) [18], Fast TCP [16], TCP Illionois [17], and TCP CUBIC [12]. All these congestion control techniques of TCP are proposed according to hardware, network and data transfer demands. Now a days network and hardware is more sophisticated and need to transfer heavy amount of data. The exiting congestion control techniques have many problems related to fairness, rate of packet loss, throughput and low link utilization. Existing congestion control techniques have severe problem of RTT unfairness and achieving full available bandwidth on the given link because the congestion window increase rate gets larger as the window grow [22]. To solve this problem TCP BIC [11] was proposed which also improves the TCP friendliness and bandwidth scalability. The proposed congestion control algorithm uses two types of window size control polices called additive increase and binary search increase. In the presences of large congestion window additive increase with large increment ensure the RTT unfairness and good scalability and when the size of congestion window is small the binary search increase the TCP friendliness [11].

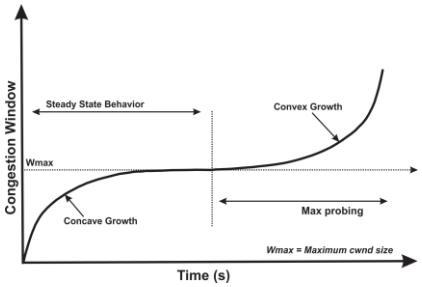

Figure 2: Concave and Convex function of CUBIC

The next enhanced version of BIC is TCP CUBIC [12]. CUBIC replace the concave and convex windows growth function of BIC by Cubic function as shown in Figure 2. The key feature of the TCP CUBIC is that the growth of congestion window depends between two consecutive congestion events. CUBIC replaced by TCP BIC in 2006 as a default TCP for Linux, but TCP CUBIC still have many problems related to slow start phase of TCP, slow convergence, RTT fairness and friendliness. TCP CUBIC is high speed TCP variant which is the default protocol of Linux. In high bandwidth delay product TCP CUBIC may can achieve full utilization of available bandwidth, but TCP CUBIC has shortcomings in term of convergence speed [23].

3. METHODOLOGY

To evaluate the performance of these fourteen congestion control techniques, many experimental tests are conducted by using NS-2 having version 2.35 and Linux operating having version Fedora 16 . NS-2 is widely used simulation tool to evaluate the many performance metrics, like goodput, efficiency, convergence time, fairness and friendliness, behavior of congestion window, packet loss rate and throughput. The main aim of this simulation experiment is to analysis the performance of many congestion control techniques in wired network environment. For the simulation, wired network topology is configured with four nodes. One node represents the sender of the data, second node represents the receiver of data and other 2 nodes are routers between these two sender and receiver nodes for the completion of network topology. A brief discussion about network scenarios is discussed in next subsection.

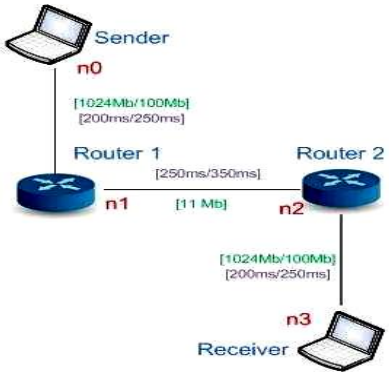

Fig. 3. Network Scenario, Test Bed

4. SIMULATION TOPOLOGY

To simulate the tests, a four nodes test-bed is used as shown in Figure 3, n0 represents data sending node, n3 represents data receiving node and n1, n2 represent routers between sending and receiving nodes. In this scenario, data communication between nodes n0 and n3 is done in four different data flows; detail is given in Table 1. Complete NS-2 wired simulation parameters are shown in Table 2. Routers n1- n2 link speed and queue-limit is 11Mb and 15 respectively. Traffic type, windows size and packet size is Linux TCP, 9000 and 3500 respectively. For all wired scenario simulations, routers link speed, routers queue limit, traffic type, windows size and packet size is constant. In wired scenario flow 1 and flow 2, link speed between nodes and router is 1024 MB, propagation between nodes and routers is 200ms and propagation between routers is 250ms. While in wired flow 1a and 2a, link speed between nodes and router is 100 MB, propagation between nodes and routers is 250ms and propagation between routers is 350ms. FTP traffic is used as main traffic from node n0 to n3. In wired flow 2 and 2a, UDB traffic at rate 10Mb and packet size 3500 is used as extra background traffic. The simulations have been conducted by varying the TCP congestion control techniques, configured with TCP Linux and simulation time is 120 seconds.

TABLE I
NETWORK SCENARIO, SIMULATION PLAN

Flow Information	Main Traffic	Back Traffic
Wired Flow 1 (WrFl-1)	TCP	nill
Wired Flow 2 (WrFl-2)	TCP	UDP
Wired Flow 1a (WrFl-1a)	TCP	nill
Wired Flow 2a (WrFl-2a)	TCP	UDP

4. RESULTS AND DISCUSSION

The simulation results are divided into three parts. In first part the behavior of congestion window is evaluated in wired network scenarios, in second part analysis of packet drop rate is done and in part the throughput of these fourteen variants is evaluated.

TABLE II
NETWORK SCENARIO, SIMULATION PARAMETERS

Parameter Name	WrFl-1 & WrFl-2	WrFl-1a & WrFl-2a
Link Speed n0-n1	1024 Mb	100 Mb
Link Speed n2-n3	1024 Mb	100 Mb
Propagation n0-n1	200 ms	250 ms
Propagation n2-n3	200 ms	250 ms
Propagation n1-n2	250 ms	350 ms
Link Speed n1-n2	11 Mb	11 Mb
Queue Limit n1-n2	15	15
Traffic 1 n0-n3	TCP Linux	TCP Linux
Traffic Type	FTP	FTP
Window Size	9000	9000
Packet Size	3500	3500
Simulation Time	120 sec	120 sec
Traffic 2 n1-n2	CBR (WrFl-2)	CBR (WrFl-2a)
Traffic Type	UDP	UDP
Rate	10 Mb	10 Mb
Packet Size	3500	3500

2. ANALYSIS OF CONGESTION WINDOW BEHAVIOUR

In this section the behavior of congestion window is discussed. The trace files are obtained after simulation by NS-2 and organized in statistical tool SPSS. The resultant graph shows the combine behavior of congestion window for all fourteen TCP congestion control techniques. In Figure 4, the congestion control technique of TCP Hybla increases its congestion window size very aggressively to obtain the given bandwidth as compare to other techniques, which also leads

towards burst packet losses , that's why TCP Hybla is no more in any network. After that the congestion window behavior of TCP Highspeed, Scalable TCP, TCP Veno and YeAH is near to similar, Highspeed TCP is more friendly as compare to other two variants. TCP Vegas and TCP Reno are the oldest congestion control techniques, so the in these congestion control techniques the growth of congestion window is very slow as shown in Figure 4. The congestion window behavior of TCP LP is very similar to to TCP Reno. The behavior of TCP BIC and YeAH TCP is also similar.

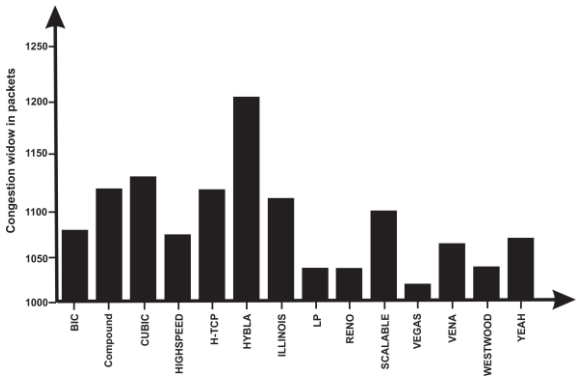

Figure 4: Behaviour of Congestion Control Technique

5. ANALYSIS OF PACKET DROP RATE

In this section packet drop rate is calculated and discussed. The mean of packet drop is taken of all flows and evaluated in SPSS. With same wired network scenario the results of packet drop are taken from flows of fourteen congestion control techniques by using NS-2 and its trace files. Its is observed that TCP Hybla has high packet drop rate while TCP Compound and TCP CUBIC have low packet drop rate as compare to other congestion control technique as shown in Figure 5. The packet drop rate of Highspeed TCP, Hamiton TCP, TCP Illinois and scalble TCP is 70% same, therefore drop rate of TCP Vegas, TCP Westwood and YeAH TCP are 80% same with each other. However we can say that the packet loss rate of TCP BIC, TCP Compund and TCP CUBIC is low as compare to other congestion control techniques.

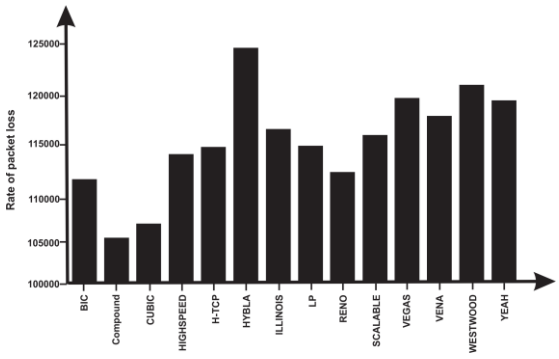

Figure 5: Packet Drop Rate of Congestion Control Techniques

6. ANALYSIS OF THROUGHPUT

In this section throughput is evaluated and discussed. The mean throughput is taken of all flows and evaluated in statistical tool SPSS. With the same network scenario, throughput results are taken from four flows of fourteen congestion control techniques for the analysis. I divided the simulation in two phases. In first phase these techniques are evaluated during the first ten seconds and in second phase these are evaluated during first fifteen seconds. In first phase TCP Hyble showed the highest throughout and TCP Veno showed the lowest throughput in Figure 6. It is surprised that TCP Compound, Highspeed TCP, TCP LP, TCP Reno, TCP Vegas and TCP Westwood shows the same mean throughput during the first ten seconds as shown in Figure 6. The behavior of TCP Hybla, TCP CUBIC, Highspeed TCP is same in throughput analysis and in *cwnd* analysis section.

It is concluded that the behavior of *cwnd* effected the over all throughput of the network. We can not say that TCP Hybla is performing well between all fourteen congestion control techniques because the rate of packet loss of TCP Hybla is very high as compare to other TCP congestion control techniques. So after intensive analysis it is concluded that TCP CUBIC is on the top of all other congestion technique and thats why it is default congestion control technique in Linux operating system.

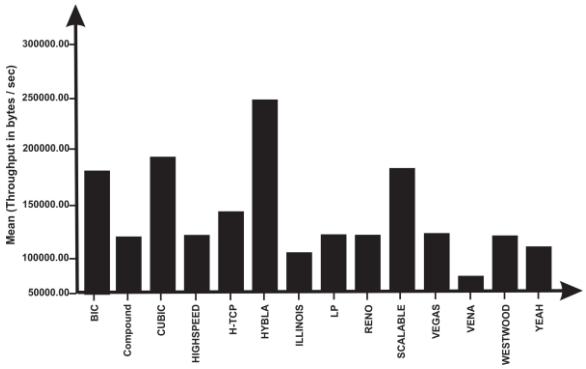

Figure 6: Throughput after first 10 seconds

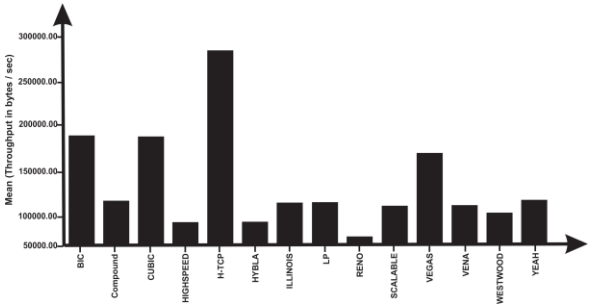

Figure 7: Throughput after first 15 seconds

7. CONCLUSION

TCP is a wide spread protocol on the Internet for the transmission of data. With the passage of time many congestion control techniques are proposed to solve the congestion problem for many network scenarios. As a result

14 congestion control techniques are included in Linux operating system, from the classic technique (TCP Reno), and then many improved versions like TCP Vegas, TCP Veno, TCP Weswood, TCP Illinois, Scalable TCP, YeAH TCP, TCP BIC, TCP CUBIC and other are introduced.

In this paper comparative analysis is offered in term of behavior of congestion window, packet loss and throughput of 14 congestion control techniques in wired network topology. Such kind of Analysis showed that the behavior of congestion window is depend upon the packet losses. In this analysis , TCP CUBIC performed well in term of congestion window behavior and packet loss, that's why it is a default congestion control technique in current Linux operating system.

REFERENCES

[1] Postel, J. Transmission control protocol, (1981)

[2] Jacobson, V. Congestion avoidance and control. In ACMSIGCOMMComputer Communication Review, vol. 18. ACM, 314-329, (1988).

[3] Allman,M., Paxson, V., Stevens,W. et al.. TCP congestion control. Baiocchi, (1999)

[4] Nagle, J. Congestion control in IP/TCP internetworks (1984)

[5] Floyd, S. Congestion control principles. Floyd, (2000)

[6] Rohrer, J. P., Perrins, E. and Sterbenz, J. P. End-to-end disruption-tolerant transport protocol issues and design for airborne telemetry networks. In Proceedings of the International Telemetering Conference,(San Diego, CA) (2008).

[7] Krevat, E., Vasudevan, V., Phanishayee, A., Andersen, D. G., Ganger, G. R., Gibson, G. A. and Seshan, S.. On application-level approaches to avoiding TCP throughput collapse in cluster-based storage systems. In Proceedings of the 2nd international workshop on Petascale data storage: held in conjunction with Supercomputing€™07. ACM, (2007).

[8] Floyd, S. HighSpeed TCP for large congestion windows. (2003)

[9] Kelly, T. Scalable TCP: Improving performance in highspeed wide area networks. ACM SIGCOMM Computer Communication Review. **33(2)**, (2003)

[10] Leith, D. and Shorten, R. (2004). H-TCP: TCP for high-speed and long-distance networks. In Proceedings of PFLDnet, vol. **4** (2004).

[11] Xu, L., Harfoush, K. and Rhee, I. Binary increase congestion control (BIC) for fast long-distance networks. In INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol. **4**. IEEE, Xu,W. (2004).

[12] Ha, S., Rhee, I. and Xu, L. CUBIC: a new TCP-friendly high-speed TCP variant. Selected Areas in Communications, Journal IEEE on. **42(5)**, (2008).

[13] Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M. Y. and Wang, R. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. In Proceedings of the 7th annual international conference on Mobile computing and networking. ACM, (2001).

[14] Floyd, S., Henderson, T. and Gurtov, A. The NewReno modification to TCPs fast recovery algorithm. Technical report. RFC 2582, April, (1999).

[15] Brakmo, L. and Peterson, L. TCP Vegas: End to end congestion avoidance on a global Internet. Selected Areas in Communications, IEEE Journal on. 13(8), (1995)

[16] Wei, D., Jin, C., Low, S. and Hegde, S. FAST TCP: motivation, architecture, algorithms, performance. IEEE/ACM Transactions on Networking (ToN). 14(6), (2006).

[17] Liu, S., Bas Â¸ar, T. and Srikant, R. TCP-Illinois: A loss-and delay-based congestion control algorithm for high-speed networks. IEEE/ACM Transactions on Networking (ToN). **65(6)**, (2008).

[18] Song, K. T. J., Zhang, Q. and Sridharan, M. Compound TCP: A scalable and TCP-friendly congestion control for igh-speed networks. Proceedings of PFLDnet (2006).

[19] Floyd, S. and Fall, K. Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transactions on Networking (TON). 7(4), (1999).

[20] Qureshi, B., Othman, M., Subramaniam, S. andWati, N. A. QTCP: Improving Throughput Performance Evaluation with High-Speed Networks. Arabian Journal for Science and Engineering, (2012).

[21] Pan, X.-z., Su, F.-j. and Ping, L. d. CW-HSTCP: Fair TCP in high-speed networks. JOURNAL-ZHEJIANG UNIVERSITY SCIENCE. **7(2)**, 172 (2006)

[22] Caoa. CUBIC with faster convergence: An Improved CUBIC Fast Convergence Mechanism. (2002)

[23] Leith, D., Shorten, R. and McCullagh, G. Experimental evaluation of Cubic- TCP, (2008).