WWJMRD 2018; 4(11): 1-7
www.wwjmrd.com
International Journal
Peer Reviewed Journal
Refereed Journal
Indexed Journal
Impact Factor MJIF: 4.25
E-ISSN: 2454-6615

Enem Theophilus Aniemeka
Department of Communication & Information Technology, Air Force Institute of Technology, Nigerian Air Force Base, Kaduna, Nigeria

Correspondence:
Enem Theophilus Aniemeka
Department of Communication & Information Technology, Air Force Institute of Technology, Nigerian Air Force Base, Kaduna, Nigeria

Transmission Control Protocol's Initial Congestion Window for Web Latency Reduction and Speedy Flows

Enem Theophilus Aniemeka

Abstract
Transmission Control Protocol (TCP) is a connection-oriented protocol which was designed to provide a reliable transfer of packet data across the Internet. As a result of the increased usage of the Internet, it became very necessary to increase the Internet's initial congestion window (ICW). The problem of TCP was that its flow control algorithm in use was not suitable for short-lived links. Moreover, the slow start of the TCP prevented the optimal performance of the network from being attained, because the Internet web connection time tends to be very short. This causes the Slow Start phase to close; thereby making the ICW to adversely affect the number of round trip times (RTT) a Hypertext Transfer Protocol (HTTP) request/response requires. This study aimed at developing an initial congestion window for the TCP to enhance web latency reduction and speedy flows. This study developed a TCP congestion window model that determined the effect of the increase in congestion window size on the latency of Web requests for links with various RTT, Bandwidth (BW) and Bandwidth-delay product, so as to reduce the latency. A non-TCP congestion control algorithm on a Router was developed to improve the efficiency of a retransmission timer. Optimum Network Engineering Tools (OPNET 17.5) was used as the simulation tool for modeling of the ICW for TCP, while the implementation of algorithm was done using Microsoft Visual C++. The experiment employed a topology with a client-server connected to the router and an IP Cloud with a Digital Subscriber Line (DSL) link transmitting at 1.5MB/sec. Packets of various data sizes were transmitted while varying the ICW values. The methodology used demonstrated the ICW for TCP on a web content application. Also, a non-TCP congestion control algorithm was implemented on the router to optimize the retransmission timer and make the acknowledgement of request to be faster in order to reduce latency and increase speedy flows of traffic on the web. The result showed that the developed TCP model in varying the congestion window size from 2 to 48 was achieved. The modified Random Early Detection (RED) algorithms were able to reduce the transmission timer and also increase the congestion window size. Hence, the result showed that the RTT, throughput and web latency of windows 8/8.1 gave the best result compared to other operating systems (OS). The packet loss reduced to 18 bytes and the RTT was reduced to 10ms at TCP congestion window of 32. Hence, the TCP congestion window was increased to 32 segments.

Keywords: Web Latency, Speedy Flows, TCP, Protocol, Congestion

1.0 Introduction
Different version of TCP in a local network with lossy link have been compared and analyzed in (Wang, et al, 2011). Several schemes have been reported in (Ajax, 2014), to alleviate the effect of losses on TCP performances over wireless network (or networks with lossy links). Slow Start is the main algorithm embedded in a TCP and it is one basic requirement for its implementation. The name Slow Start was coined by John Nagle in 1987 in his mailing list message to Internet Engineering Task Force (IETF), (Jacobson and karel, 2012). Whenever a TCP connection is first established, the Slow Start algorithm initializes a congestion window of one, two or ten segment, (Corbet, 2012; Runa, et al, 2012) which is the amount of data (maximum segment size) the receiver initialized during the first establishment. The moment there is an acknowledgement from the receiver, the congestion window will increase by one segment for each acknowledgement returned to the sender.

Subsequently for every receiver's acknowledgement, the congestion window grows exponentially, two, four, eight, sixteen, and thirty-two and so on. The increase in size will continue until there is packet drop, which occurs probably because the buffer has filled-up. When packets drop occurs it will initiate a timeout at the sender's side and the connection will leave Slow Start to enter TCP congestion avoidance.

A lot of paper efforts had been put in TCP congestion control with special interest on TCP Congestion Avoidance. However, current study has shown that the major interest should be directed to NON-TCP solution to congestion control (Kristoff, 2015) with more emphasis on Router. Therefore, this paper work is focused at developing a TCP congestion window model that will determine the effect of the increase in congestion window size on the latency of Web requests for various links and also develop Non-TCP congestion control algorithm on a Router. Packet drops are known to be the major cause of congestion in the network and its effective management will go a long way in minimizing congestion on the Internet. The network speed will improve better if Router handles the packet drop than in the current TCP. They will implicitly communicate to the sender through the retransmission timer. There are two approaches to Network Congestion control: End-to-End and Network control. Our focus is on the Network control. Network congestion control uses active queue management to monitor and control congestion, through the use of probability in queue length (Athanasiou, 2012, Kurose, et al, 2014).

The mode of monitoring the average queue size at the router and informing connections of congestion is based on the facts that packets from different connection must always queue at the router using FIFO scheduling (except in the scenario where priority is given). In time of congestion, the probability that the router notifies connections to reset its transmission timer is proportional to that connection ‘share of the bandwidth through the router (Ma, et al, 2011 & Mardini, et al, 2011). Whenever a packet is sent out, TCP starts a timer, if the timer expires without an acknowledgement; it is assumed that packet is lost. However, this might take the TCP a long time to notice that a packet is lost and take action. A feedback from the Router algorithm to the sender through the use of Transmission timer will save congestion build-up in the network and round trip time (RTT). TCP sender can perceive congestion when there is a timeout or 3 duplicate acknowledgements (Slow Start). A new approach to packet drop problem is that the moment there is a packet drop the Router algorithm should cause the retransmission timer to expire immediately (Reset), instead of waiting for the default 1sec to elapse. (Allman, et al, 2013)

To improve TCP performance, a number of proposals have been suggested including Persistent Hypertext Transmission Protocol (HTTP), Transaction TCP and Sharing TCP control blocks. The aim is to adapt the resource usage of a flow to the network’s available resources which are to be shared fairly between competing flows.

2.0 Review of Related Works

The original version of TCP congestion control specified that the congestion window should start at the Maximum Segment Size (MSS) of the connection (Bellavista, et al, 2016), which is computed from the peer’s MSS advertisement obtained during synchronization (SYN) handshake. Mark Allman in 2013, proposed performance enhancement that raises the initial window from 1 MSS sized segment to roughly 4KB. He found out that an initial window of 2 to 4 segments, as suggested by (Allman, et al, 2013), decreased the time of short transfer when compared to 2 and 3 segments.

In September, 2013, Request for Comment (RFC) 2414 suggested experimentation with an initial window of 2 to 4 MSS, depending on the range of MSS. In October, 2011, RFC 3390 finally upgraded the general recommendation (Allman, et al, 2011). According to RFC 5681, the typical machine connected using approximately 1500 bytes Maximum Transmission Unit (MTU) links, should be three (3) MSS or about 4300 bytes, which is for standard Ethernet (Kristoff, 2015).

Recently, the quest to further increase TCP’s initial window continued. In December, 2014 Internet Engineering Task Force (IETF), the body responsible for judging the usefulness or best practice when new features are proposed together with Internet Congestion Control Paper Group (ICCRG) discussed four (4) proposals in connection to increase in TCP’s initial window. A team of staff from Google, H.K Jerry Chu, Nandita Dukkipati, Yuchung Cheng, and Matt Mathis, proposed (Chu, et al, 2014) the increase of initial window to 10 segments. They supported the claim with analytical and experimental work that displayed significant improvements in completion times for typical sized web transfer with small negative effects. A second proposal by (Allman, 2014), suggested the increase of initial window over time, beginning with 6 segments in 2011, incrementing to 15 segments by 2019. The third proposal (Yourtchenko, 2014) suggested a negotiation of initial congestion window between both parties of the TCP connections. Finally, (Touch, 2012) sent a draft that proposed adaption of initial window over a long term based on slow additive increase, and multiplicative decrease when it detects that initial window is too large.

In April, 2013, RFC 6928 was issued as an experimental request for comment based on (Chu, et al, 2013). In November, 2015, Mark Allman published a draft (Allman, 2015) which suggested the removal of initial window restriction and allow the sender to send as many packets as it wants under certain conditions. He pointed out that if the initial window is greater than 10 segments, the packets must be evenly paced over the first round-trip-time.

Akamai (2015) Second Quarter Release, state of the Internet revealed that the average global bandwidth connection speed is 5.1Mbps with more than 60% of clients having bandwidth above 10Mbps, while the usage of narrowband (<256Kbps) has shrunk to about 3% of clients. Since short flows do not enjoy fairness in the current initial congestion window of 3 segments therefore; most applications now devised their own mechanisms for faster download of Web pages. Popular Web browsers, including Internet Explorer 11.0, Firefox 41 and Google's Chrome open up to six TCP connections per domain (IIiyan, 2014; Ajax, 2014 ; Robert, 2017) partly to increase parallelism and avoid head-offline blocking of independent Hypertext Transfer Protocol (HTTP) request/responses, but mostly to boost start-up performance when downloading a Web page. A Google team in 2014 proposed an increase in TCP’s initial congestion window value to 10 segments (Nandita, et al, 2014). The same team submitted an internet-draft copy to the Internet Engineering Task Force for consideration and possible implementation (Nandita, et al, 2012). They conducted some analysis with Web traffic using initial value of 10, and came up with the findings that the completion time of flows significantly decreased. Though, they noticed a negative impact on retransmission rate.

3.0 Methodology

This work focused on the various analyses of the experimental data as functions of traffic characteristics, bandwidth (BW), round-trip-time (RTT), and bandwidth-delay product (BDP).The design tool used is optimum network engineering. Both routing and TCP algorithm was implemented to reduce web latency and increase speedy flows. For the experimental analysis, it used a simple topology with a clients-server connected to the router and an IP cloud with a DSL link transmitting at 1.5MB/sec

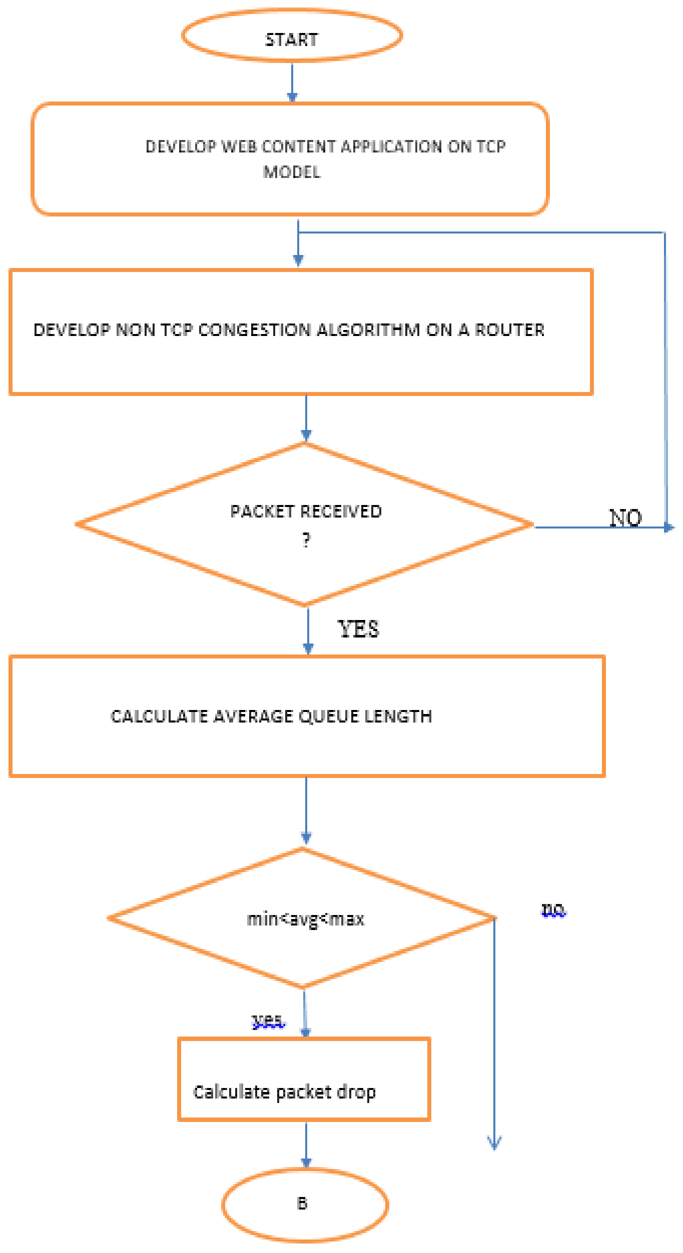
START
DEVELOP WEB CONTENT APPLICATION ON TCP MODEL
DEVELOP NON TCP CONGESTION ALGORITHM ON A ROUTER
PACKET RECEIVED ?
NO
YES
CALCULATE AVERAGE QUEUE LENGTH
min<avg<max
no
yes
Calculate packet drop
B

Fig 1: a Conceptual Framework

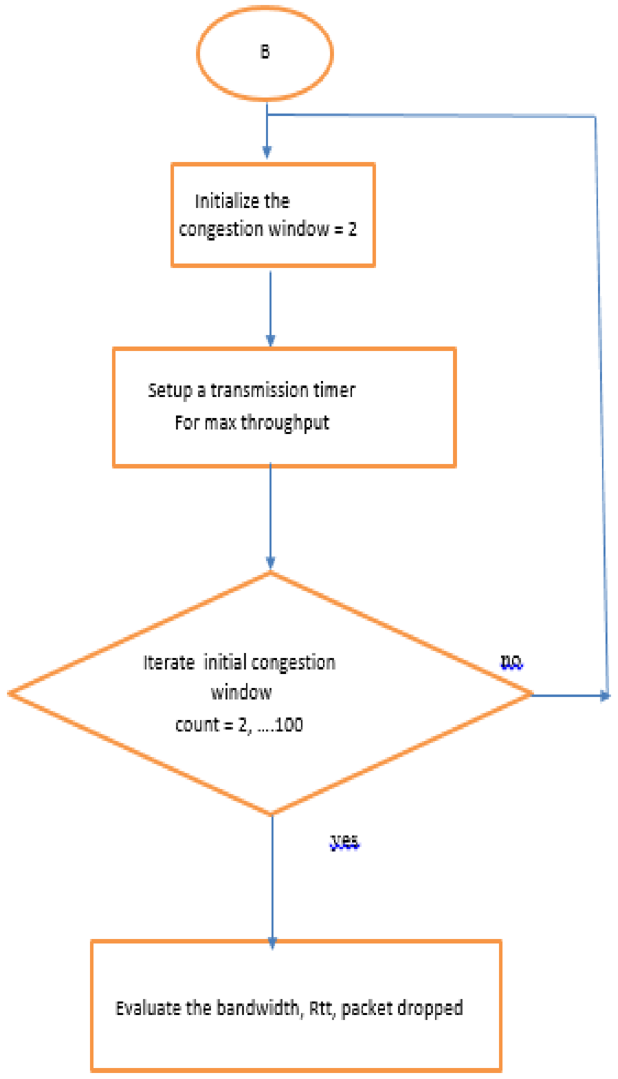
B
Initialize the congestion window = 2
Setup a transmission timer For max throughput
Iterate initial congestion window count = 2, ….100
no
yes
Evaluate the bandwidth, Rtt, packet dropped

Fig.1a.Conceptual Framework

From figure 3.1 above, the procedure for the conceptual framework is as follows:
Step 1: A web content application was developed on the TCP (transmission control protocol model)
Step 2: A non TCP congestion algorithm was developed on the router
Step 3: the program checked if packet has been received
Step 4: if step 3 is no, goto step2
Step 5: Average Queue length was calculated
Step 6: check if average queue size greater than minimum and less than maximum
Step 7: determine the packet drop
Step 8: initialize the initial congestion window =2
Step 9: Setup a transmission timer for max throughput
Step 10: Iterate initial congestion window and varies it from 2 to 100
Step 11: if step 10 is not successful, goto step 8
Step 12: determine the bandwidth, roud trip time and packet drop.

Development of a Client-Server model on TCP
The essence of this stage is to determine the throughput of a TCP connection using client-server technology. The algorithms for the development of the client- server model are as follows:
Step 1: A web application which uses TCP was developed.

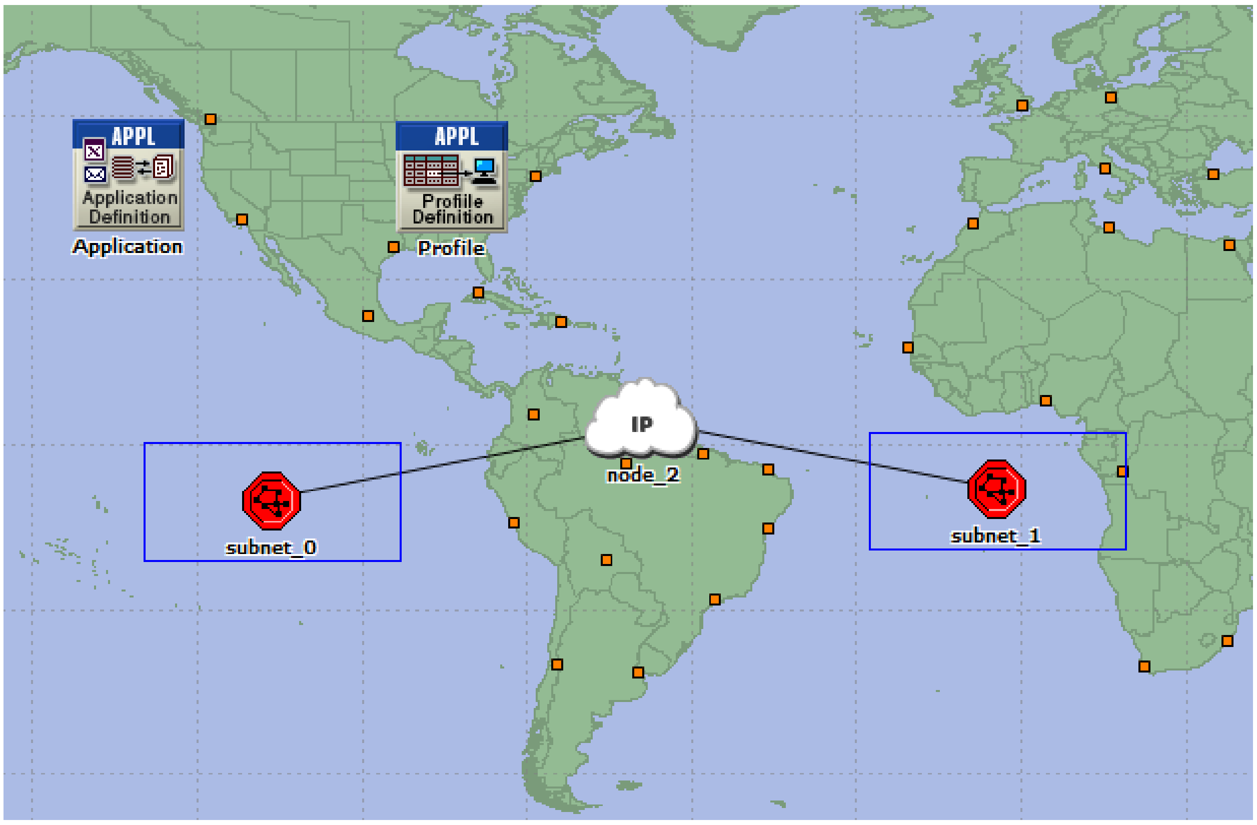

Fig.2: TCP developed Model

The figure 3.2 is the TCP model which was configured with a web application. It involves two subnet. Subnet 0 and 1 Subnet 0 is the server while subnet 1 is the client. Node 2 is the internet services. The application profile was used for development of the web object while the profile was used for the attribute which works with the application. An iteration was develop for the modification of the web latency

The probability that a flow has size s is given by,

$$p(\mathrm{s}) = \frac{\frac{i}{\mu} e^{-\frac{1}{\mu}s}}{e^{-\frac{1}{\mu - e^{-\frac{1}{\mu}\theta,}}}} 1 \le s \le \theta$$

equ 1

Where $$\sum_{s=1}^{\theta} p(\mathrm{s}) = 1.$$

equ 2

Assume that the round trip time RTT is same for all flows sharing a single bottleneck link. This causes the losses to be synchronized in case there is packet loss. As a result, the flows reduce their congestion windows (cwnd) at around the same time.

The slow-start threshold is set to a large value so that the bottleneck for a flow is only the network and not the end host. Furthermore, only packet loss can cause window halving.

3.3 Development of algorithm for slow start count

$$E[\mathrm{S}]_S = \frac{(1-(1-q)^s(1-\mathrm{q})}{q}$$:

equ 3

Let the V be set to ^ V; i.e., a flow starts with IW-size as ^ V. The number of RTTs required to send E[S]s segments was approximated to

E [T]s = log2 (E[S]s ^ V + 1). equ 4

So, for the given flow-size distribution, the expected number of RTTs is given by,

$$d = \sum_{s=1}^{\theta} E[\mathrm{T}]_s \times p(\mathrm{s})$$
equ 5

Similarly, the expected cwnd of the flow that will cause or face a loss during slow-start phase was written as,

$$E[\mathrm{W}]_S = \left\{ \frac{E[\mathrm{S}]_S +}{2} \right\}$$
equ 6

The expected congestion window l of a flow with certain flow-size from the given distribution was expressed as

$$I = \sum_{S=1}^{\theta} E[\mathrm{W}]_s \times p(\mathrm{s})$$
equ 7

Let S := {Si|i ∈ N } 2 Ng denote the strategy set, where N is the number of different values of the parameter V. Denote by x(t) = {(x1(t), ..., xN (t))|xi(t) ≥ 0, =1 xi(t) = 1} the population profile or the state space, where xi(t) is the fraction of the population using strategy Si at time t.

4.0 Result

4.1 Result Comparison for Slow Start Count

Table 1: Variation of Slow start Initial Count (cwnd) for windows 8/8.1

Slow Initial Count	Packet Loss	RTT
2	23	189
4	41	93
8	52	47
16	93	18
32	18	10
40	126	206
48	176	320

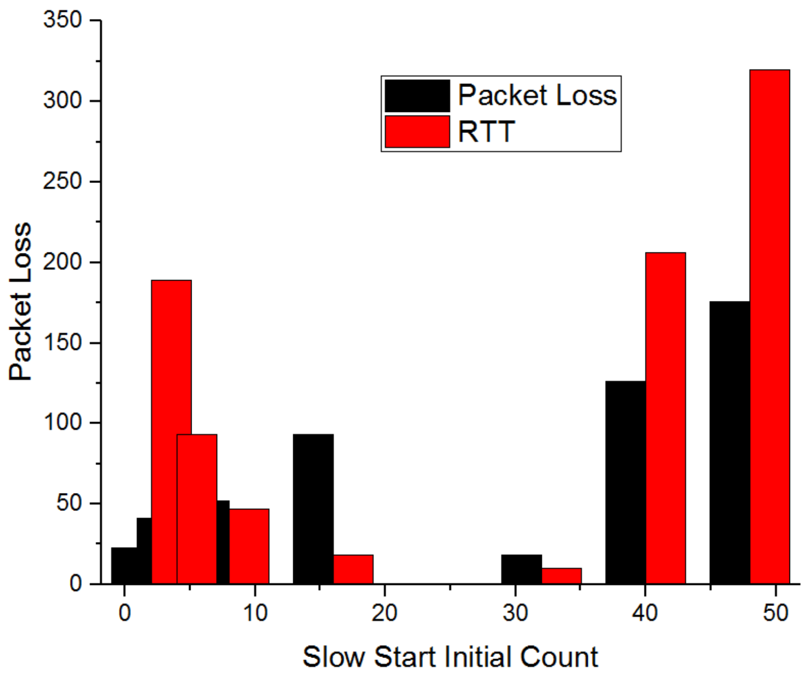

Fig.1: Slow Start Initial Count Variation

Figure 1 above is the slow initial count variation for windows 8/8.1. When the first initial count is 2, the packet loss is 23 and the round trip time is 189ms. As the initial count increases from 2, the round trip time was reducing and packet loss was increasing. When the initial count increases above 32, the round trip time increases with packet loss This implies that window 8/8.1 has been improved to 32 segments which is the maximum TCP congestion window size. This will speed up the web page request with reduction of latency and enhance faster opening of a web page.

Table 2: Packet size = 32MB

Operating system	Throughput(Bps)	RTT(ms)	Web Latency(ms)
Windows 95	1064	10	10
Windows 98	1210	08	09
Window xp	1321	07	07
Windows server	1206	06	09
Windows 7	1462	05	06
Windows 8	1872	04	04
Windows 8.1	2019	03	03
Linux mint 18.3	1892	06	10
Solaris 10	1863	07	08

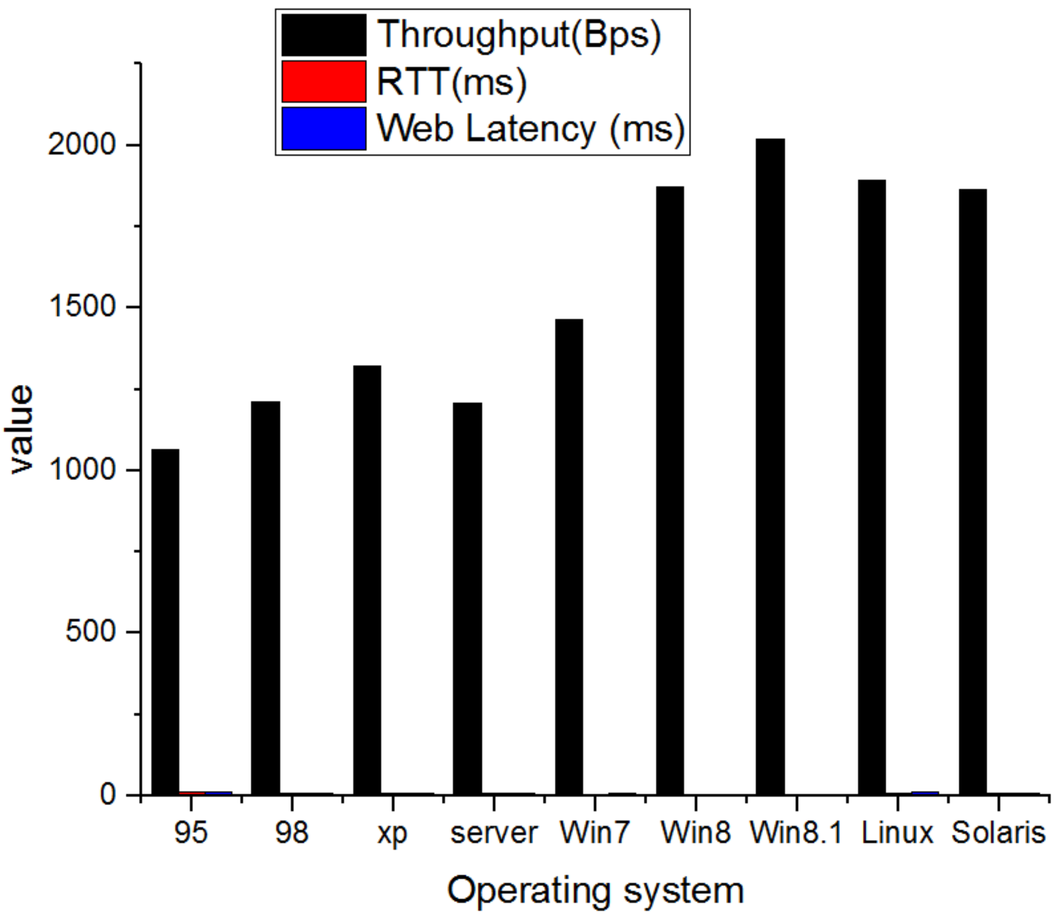

Fig.2: Packet Size at 32MB

Figure 2 above gives the comparison of differnet operating system to know the effect of throughtput, round trip time and web latency when a packet size of 32mb was transmitted on the web page. From the analysis, round trip time and web latency are very small which makes the throughput to be higher and the web page request rate faster. Window 8.1 has the highest throughput compared to Windows 95 which has the lowest. The higher the throughput the faster will be the web page request.

5.0 Conclusion and Recommendation

5.1 Conclusion

This thesis demonstrates the congestion control algorithms implemented by TCP. The study shows a number of scenarios to examine the effect of these algorithms. The performance was tested using HTTP transferring files of different sizes. The performance was examined with different file size. A good estimation of the traffic investigated in the network has been shown using OPNET simulations Transmission Control Protocol's initial congestion window for web latency reduction and speedy flows was implemented successfully. Accurate network bandwidth and resources are important to a variety of network applications. 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 including video and VOIP applications.

5.2 Recommendations

This paper work is recommended to the Internet Engineering Task Force (IETF) group, as they will help to:

1. Increase the Transmission Control Protocol's initial congestion window value (higher initial window) to 32 segments.

2. Benefit users through web latency reduction and speedy flows in their business especially stock market (FOREX) trading where a difference of a few milliseconds in latency can gain (or loss) millions of naira.
3. Benefit internet users, because latency and page load time are factors that influence user satisfaction and increase website visit.

References

1. Ajax (2014). Connectivity enhancements in windows internet explorer 8. Retrieved from: https://msdn.microsoft.com/en-us/cc304129(v=vs.85).aspx.
2. Akamai (2015). The State of the Internet. http://www.akamai.com/stateoftheinternet. Retrieved 10 July, 2015.
3. Allman, M., Floyd, S. and Partridge, C. (2014). Increasing tcp's initial window, July 2014 – Internet-Draft draft-floyd-incr-init-win-03.txt.
4. Allman, M., Hayes, C. and Ostermann, S. (2013). An evaluation of tcp with larger initial windows. Submitted to acm computer communication review. 40th IETF meeting. Retrieved 9 June, 2013.
5. Allman, M., Paxson, V. and Blanton, E. (2017). Tcp congestion control. IETF. sec. 3.2. RFC 2581. Retrieved 6 April, 2014.
6. Allman, M., Balakrishnan, H., and Floyd, S. (2001). Enhancing tcp's loss recovery using limited transmit. RFC 3042. Retrieved 2 June, 2012.
7. Allman, M., Floyd, S, and Partridge, C. (2011). Increasing tcp's initial window. RFC 3390. Retrieved 7 March, 2012.
8. Allman, M. (2014). Increase of initial congestion window over time. http//tools.ietf/html. Retrieved 4 January, 2014.
9. Allman, M.(2015).Removing Tcp's initial congestion window, November 2015. Internet-draft draft-allman-tcpm-no-initwin-00.txt (work in progress)
10. Athanasiou, G. (2012). Dynamic resource management in 802.11 wireless mesh networks. Journal of Computer Systems, Networks, and Communications, 2012
11. Bellavista, P., Cai, Y. and Magedanz, T. (2011). Recent advances in mobile middleware for wireless systems and services. Mobile Networks & Applications, 16(3), 267-269. Doi:10.1007/s11036-011-0313-7
12. Chu, J., Dukkipati, N., Cheng, Y. and Mathis, M. (2013). Increasing tcp's initial window. IETF request for comment 6928. ISSN: 2070-1721. Retrieved 10 October, 2014.
13. Corbet, J (2012). Increasing the tcp initial congestion window. Vol.4 pg102-113. Retrieved 10 October, 2013.
14. Floyd, S. and Jacobson, V. (1993). Random early detection gateways for congestion avoidance. Ieee/acm transaction on networking. Retrieved 15 May, 2014.
15. IIiyan, P. (2014). HTTP 2.0 is coming, be ready. http://calendar.perfplanet.com/2014/http-2-0-is-coming-be-ready/. Retrieved 5 August, 2015.
16. Jacobson, V. and Karel, M. (2012). Congestion avoidance and control. ACM SIGCOMM Computer Communications Review, 25(1) pp. 157-187. doi:10.1145/205447.205462. Retrievedfrom:paper.microsoft.com/en-us/um/people/cn_slides/jacobson88congestion.pdf.
17. Kristoff, J. (2015). The Transmission control protocol (TCP) and Explicit Congestion Notification, ACM Computer Communications Review, October p. 10-23.
18. Mardini, W. and Alfoul, M. (2011). Modified wrr scheduling algorithm for wimax networks. Network Protocols and Algorithms, 3(2), 24.
19. Nandita, D., Tiziana, R.,Yuchung, C., Jerry, C., Natalia, S., Agarwal, A., Herbert, T. and Jain, A. (2014). An argument for increasing TCP's initial congestion window. ACM SIGCOMM CCR, vol. 40, pp. 26-30.doi:10.1145/1823844.182348.
20. Nandita, D., Mathis, M., Cheng, Y. and Ghobadi, M. (2011). Proportional rate reduction for TCP. Computer-communication net-works: Network Protocols, November 2–4, Berlin, Germany. Retrieved 14 July, 2015.
21. Nandita, D., Cheng, Y., Mathis, M. and Chu, J. (2012).Increasing TCP's Initial Window, Internet-Draft (Experimental).Olsen, M. ACK congestion control.
22. Robert, P,.(2017). SPDY: An experimental protocol for a faster web. http://dev.chromium.org/spdy. Retrieved 15 March, 2014.
23. Runa, B. and Dinil. D. (2013). Evolution of TCP's initial window size. 38 annual IEEE conference on local computer networks. Retrieved 20 March, 2015
24. Touch, J. (2012). Automating the initial window in tcp. Internet draft. Draft-touch-tcpm-automatic-iw-03-txt. IETF secretariat. Retrieved from: https://tools.ietf.org/html/draft-touch-tcpm-automatic-iw.03. 2 July, 2015
25. Yourtchenko, A., Begen, A. and Ramaiah, A. (2014). Two-party selection of TCP initial congestion window estimate. Retrieved from:https://tools.ietf.org/pdf/draft-yourtchenko-two-party-initial-window-00