International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2017/41293 Volume 4, Issue 1, January – February (2017)

RESEARCH ARTICLE

Evaluation of TCP Congestion Control Modus Operandi in Mesh Networks

Vincent O. Nyangaresi
School of Informatics & Innovative Systems, Jaramogi Oginga Odinga University of Science & Technology, Kenya.
vincentyoung88@gmail.com

Solomon. O. Ogara
School of Informatics & Innovative Systems, Jaramogi Oginga Odinga University of Science & Technology, Kenya.
solomon.ogara@gmail.com

Silvance O. Abeka
School of Informatics & Innovative Systems, Jaramogi Oginga Odinga University of Science & Technology, Kenya.
silvancea@gmail.com

Published online: 27 January 2017

Abstract – In mesh networks, the sender machine is connected to the receiver machine via multiple paths. Efficient transmissions along these paths require proper link choice so as to quickly deliver the packets the destination. Poor link selection can lead to overutilization of some links while the other redundant links remain underutilized. Over-utilized links experience heavy congestions under peak hours. The transmission control protocol (TCP) employs congestion control algorithms to prevent transmitters from overloading the network with data. These algorithms include slow start, congestion avoidance, fast retransmit and fast recovery. The slow start algorithm is utilized during the initial communication phase while congestion avoidance, fast retransmit and fast recovery are reactionary algorithms one packet loss is detected. This paper aimed to analyze the behavior of TCP under these congestion control algorithms in wired mesh networks. The dimensions that were used for this analysis included three way handshake, packet loss, duplicate acknowledgements, segment retransmissions, recovery, I/O plots and time-sequence plots. The objective of this study was to practically understand how the TCP protocol detects and handles network congestions in mesh networks. To achieve this objective, an experimental research design was employed. It involved the practical design of experimental setups that were used to collect data that was analyzed to provide an explanation of the TCP congestion control mechanisms. The results obtained indicate that the TCP first carries out a three handshake before data transmission can take place. It was also observed that the receipt of three duplicate acknowledgements is interpreted by TCP to be packet loss caused by network congestion. Moreover, it was established that TCP initiates fast retransmit and fast recovery when packet loss is detected. The contribution of this paper lies in the fact that it provided a practical understanding of how TCP detects and reacts to mesh network congestion, a concept that is critical to network administrators in their quest for packet loss prevention over the TCP architecture. Towards the end of the paper, suggestions for developing better ways of congestion handling in mesh networks by use of round trip times as a basis for adaptive congestion detection and control are elaborated.

Index Terms – TCP, Congestion, Throughput, Bandwidth, Algorithm.

1. INTRODUCTION

All TCP connections employ the congestion window as one of the dynamics that dictate the number of byte that the sender can transmit without receiving acknowledgments. In [1], it is explained that this window serves to thwart the communication link between the communicating nodes from getting overwhelmed by traffic. Overwhelmed networks lead to packet losses, which according to [2] may be caused by malicious packet dropping and link error.

When connections are established, the congestion window is set to miniature multiple of the maximum segment size (MSS) permitted on those connections. In situations where all the data segments are received and acknowledged, the congestion window is increased by one MSS [3]. This constitutes the slow start phase and it keeps on increasing the congestion window exponentially until a timeout happens or the receiver's window size reaches its limit, called the slow start threshold value (*ssthresh*).

In their study,[4] noted that the receipt of three duplicate acknowledgements in a TCP communication is interpreted as being due to packet loss caused by overcrowded links. Therefore TCP reacts by adjusting the number of bytes that can be sent downwards and the communication enters the congestion avoidance phase. In this stage, the congestion window enlarges linearly at the rate of one MSS per congestion window packets for each acknowledgement. The consequence of this is that the congestion window is incremented by one segment on condition that all segments are acknowledged.

RESEARCH ARTICLE

Unfortunately, using this TCP congestion window adjustment, a malicious node may increase its congestion window hence its bandwidth at the expense of other network nodes. As such, in their study, [5] proposed a dynamic approach for malicious node detection for internet traffic analysis. This is based on the analysis of the node behavior as a basis for allowing or denying connections from a node by computing the posterior probabilities of the factors peculiar to that node.

In TCP Tahoe, in the event of a loss, fast retransmit is entered [6]. In this phase, the slow start threshold value is set to half of the current congestion window while the value of the congestion window is set to one MSS and the communicating parties revert to slow start phase.

Fast recovery is implemented in a variant of TCP known as TCP Reno. In this algorithm, when a packet loss occurs, the slow start threshold value is set to half of the current congestion window while the value of the congestion window is set to one MSS, just like in TCP Tahoe [7]. However, this algorithm omits the slow start phase and enters directly into congestion avoidance phase.

2. MESH NETWORKS COMMUNICATIONS

A mesh network refers to a network where each of the network computers is directly connected to other network computers (fully mesh) or connects indirectly to others (partial mesh) as shown in Figure 1. According to [8], wireless mesh networks offer low up-front cost, reliable coverage, easy network maintenance, dynamic self-organization and self-configuration. These features are superlative for the next generation wireless communication systems since they offer seamless broadband access. Mesh networks are highly fault-tolerant. This so because every computer has multiple possible connections paths to the other computers on the network, so a single cable break will not stop network communication between any two computers.

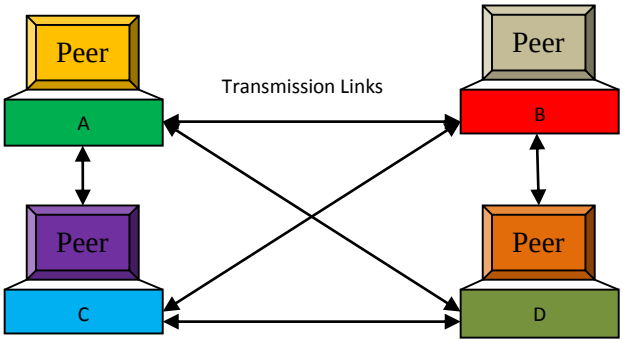

Figure 1: Full Mesh Conceptual Set Up

This figure shows that for machine A to transfer data packets to machine D, there is one direct link that it can use and two indirect links that are still viable for traffic transmissions. The direct link connects A to D through a straight line. Indirect links occur through C and B. All these links are feasible candidates for packet data communication.

In situations where hop count is used as the only criteria for path selection, then since path A---D has less hops than both paths A—B—D AND A—C—D, path A—D will always be chosen for traffic that is meant to traverse A and D. This effectively leads to over-utilization of this route while other rotes lie idle. It is the way the TCP congestion algorithms handles this form of mesh network congestion that this paper sought to investigate and analyze.

For efficient traffic transmission, link A—D should be able to detect and deal with any congestion in an efficient adaptive version.

3. MOTIVATION

In their paper, [9] proposed an energy efficient routing (EER) for reducing congestion and time delay in wireless sensor networks. This approach could lessen the transmission time and time delays for forwarding the packets. This was accomplished using an energy efficient routing protocol, where a discrete delay function is employed to establish new transitional node and then forwarding the network packets through that best intermediate node.

In [10] it is explained that network congestion happens when a network traffic is greater that the network capacity. It can be necessitated by having many data transmitters sending large amounts of data at a very high transmission rate. Network delays cause elongated packet delays and may even cause packet loss as a result of buffer overflow [11].

To address these shortcomings, congestion control algorithms have been developed and implemented in TCP. The congestion control techniques can be viewed as measures to contain network clogging and keep overall data load well below the link capacity [12].

Figure 2 (a) and 2 (b) provide an illustration of the delay and throughput as some of the network congestion performance measures.

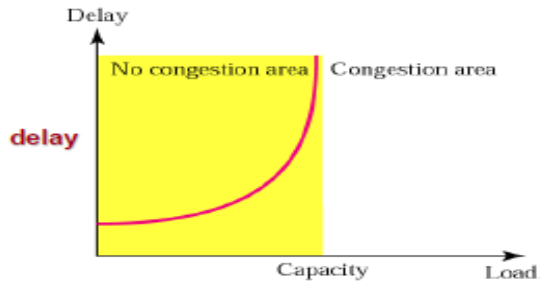

Figure 2 (a): Delay: Network Congestion Performance Measures [3]

RESEARCH ARTICLE

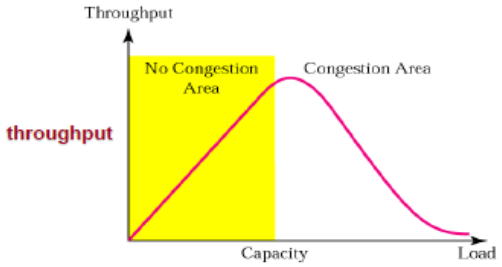

Figure 2 (b): Throughput: Network Congestion Performance Measures [3]

Figure 2(a) shows that there is an exponential increase in network transmission delays as the traffic advances towards network capacity. On its part, Figure 2(b) demonstrates that there is exponential reduction in data throughput as the traffic exceeds the network capacity.

In circumstances where a data segment is delayed, the sender fails to receive acknowledgements in time and therefore ends up retransmitting the packets [13]. The effects of these retransmissions is that there will be the data queues longer and causes further delays, which is in essence, congestion of data at the receiver.

On the other hand, when the traffic becomes more than the network capacity, the queues become jam-packed and the network devices have to discard some packets [14].

This can be demonstrated by Figure 3. As illustrated by this figure, the receiver has a maximum capacity of 4K (4 x 2^{10}= 4096 bytes). This means that it will receive data up to the capacity of 4K, after which it will be forced to drop the incoming packets.

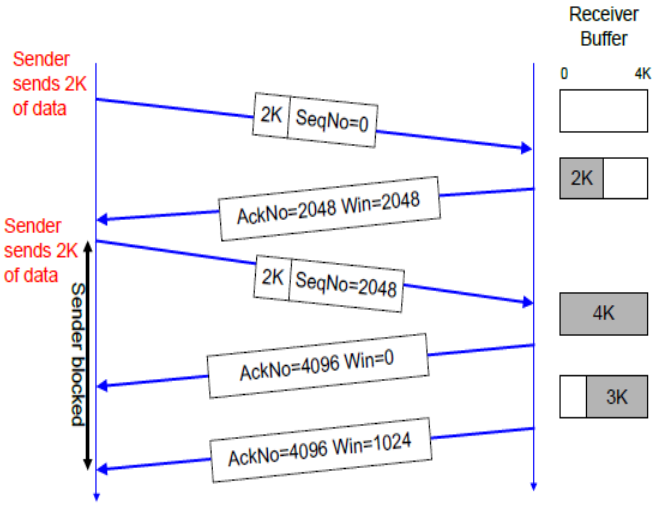

Figure 3: Receiver Buffer Full

Note that acknowledgement number 4096 is accompanied by a window size of 0, which informs the sender that the receiver window is full and therefore it should not transmit any packet. However, if the sender ignores this instruction, then the receiver will be forced to drop the transmitted packets.

When packets are dropped, they will never be acknowledged and the sender will enter fast retransmit and fast recovery to deliver the lost packets [15]. A novel idea will be for the transmitter to wait till the receiver passes the transmitted data to the applications, after which this data will be moved from the buffer, hence creating more space in the buffer.

This is illustrated by the last acknowledgement, where the receiver creates a window size of 1K (IK=2^{10}=1024 bytes). This is why it is now advertising a window of 1024 bytes.

4. PROCEDURE

In a mesh network, computers are connected directly to one another. This network has high redundancy as a packet can take one of the many routes from the source to the destination. To investigate how TCP handles congestion control in a mesh network, two sets of experiments were carried out.

The first set up involved the transfer of a text file between two laptop computers via the hypertext transfer protocol (HTTP). The second experiment involved the downloading of a *243 MB* video file located in *a Wamp server* of one of the laptops. Figure 4 shows part of the mesh network that was employed in this study.

Figure 4: Experimental Set Up

The set up consisted of four laptops labeled A, B, C and D, connected via an Ethernet cable, through a four-port switch.

RESEARCH ARTICLE

Conceptually, this resembles the conceptual full mesh network given in Figure 1.

The file to be transferred, named *'Server'* was created in *Wamp* server and stored in a folder called *ADAPTIVE_TCP_CONGESTION_CONTROL_ALGORITH*. The machine hosting this *Wamp* server was given IP address of 192.168.1.1. Figure 5 shows this file residing in a server.

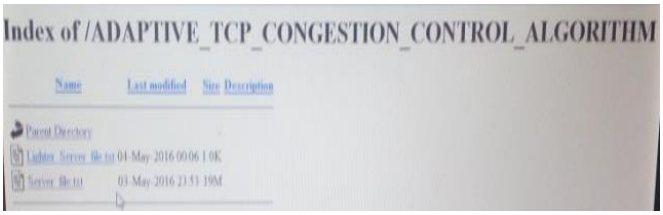

Figure 5: File Residing in *Wamp* Server

The laptop to which the file was to be transferred to was given an IP address of 192.168.1.10. After this, the file URL was entered in the browser of the receiver upon which the file was clicked and the *'save'* option was selected to start the download process. The wireshark network analyze software was employed to capture the transfer of this file between the two laptops.

It was realized that file transfer through HTTP took a very short duration and the analysis of the required parameters could not be possible. Therefore, a *243 MB* video, which took a little longer duration, was downloaded while wireshark was used to monitor the packet transfer between the two laptops. Figure 6 shows the video downloading in progress.

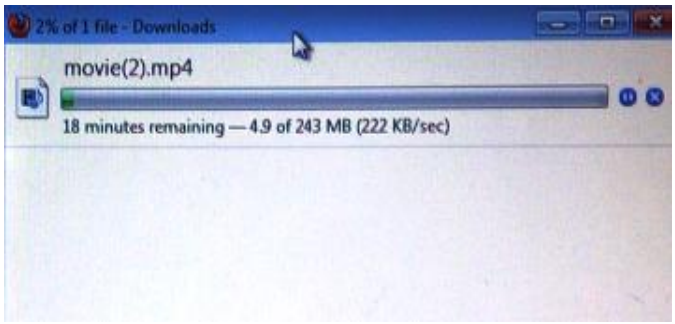

Figure 6: Video Download via HTTP

The downloading speed was 222 KB per second, which meant that the video took an average of [243 x1000/222] = 18 minutes to complete. This was enough time to finish all the required analysis.

5. EXPERIMENTAL RESULTS

The sub-sections below give a detailed description of the data that was obtained from the experimentations that were carried out. This include the verification that TCP employs three way handshake, slow start, congestion avoidance, fast retransmit and fast recovery algorithms as reactionary measure to network congestions.

5.1. TCP Three Way Handshake

The sender starts the conversation with the receiver by sending a SYN packet, sequence number 10200. The receiver responds with a SYN +ACK packet, sequence number 10201, indicating that it is ready for the data exchange. The receiver receives this acknowledgement and proceeds to make a request for the video that is to be downloaded. This constitutes the three way handshake as illustrated by Figure 7.

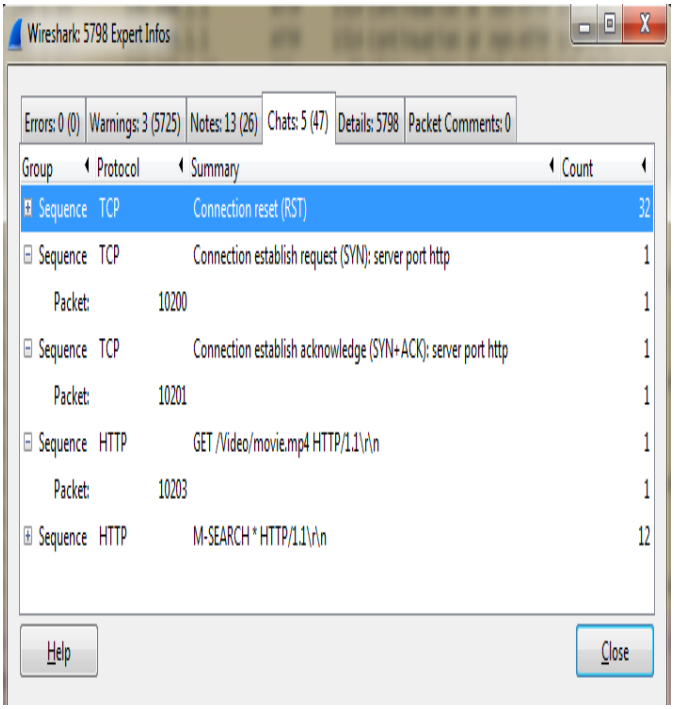

Figure 7: Three Way Handshake

This information was obtained by clicking on the *'Analyze'* menu and selecting *'Expert Info'*. This confirms the fact that indeed the communicating systems must initially have a three way handshake before any data communication can take place between them.

5.2. Slow Start Phase

After the three way handshake, the connection has been established and the TCP begins to slowly establish the network bandwidth in order to avoid transmitting too much data on the network that might be dropped. It was observed that for packet number 12, its sequence number was 1(relative sequence number) and the bytes in flight were 1460. In slow start phase, each time an acknowledgement is received, the congestion window is incremented by 1 MSS (where 1 MSS=1460 bytes). This was confirmed by observing packet number 12 and packet number 13 in Figure 8.

RESEARCH ARTICLE

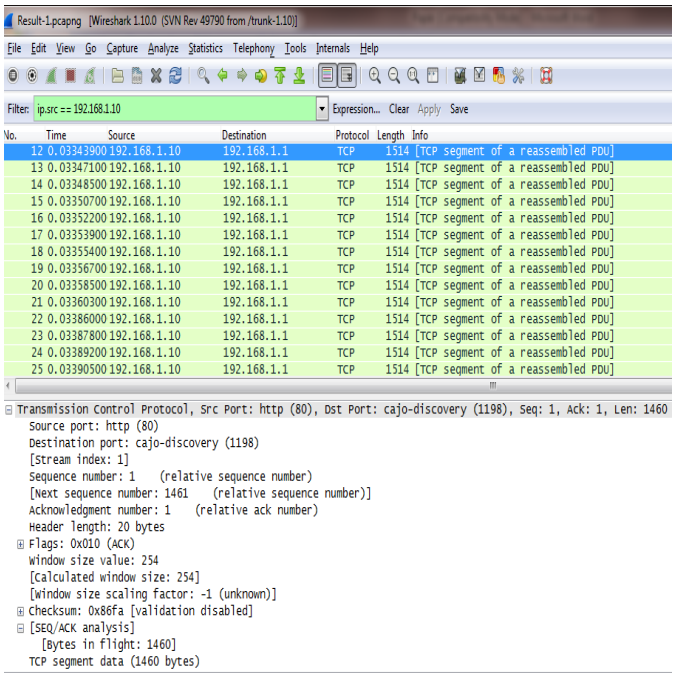

Figure 8: TCP Slow Start Phase

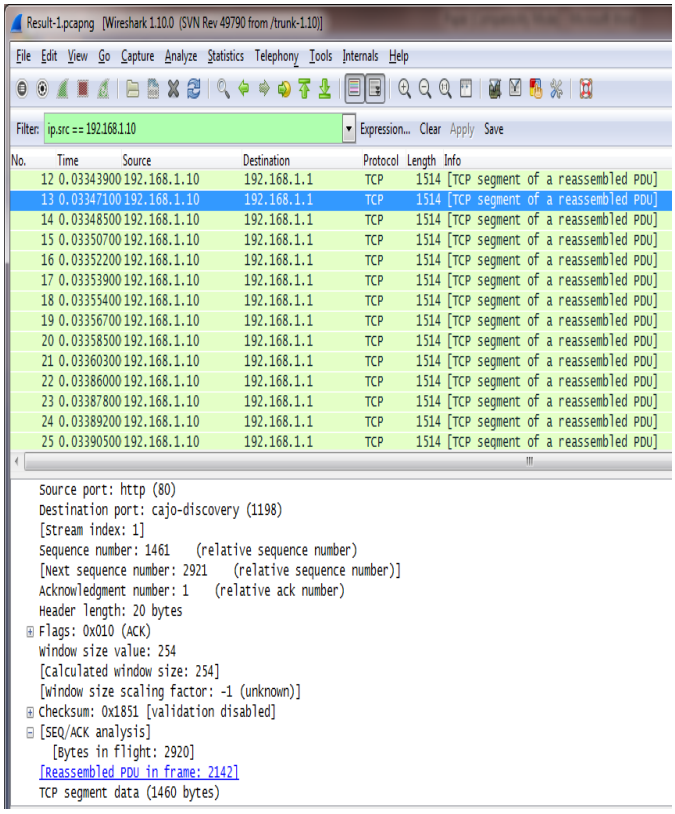

Figure 9: Congestion Window after ACK for Packet 12

It was observed that the advertised window size by the receiver was 254 while the bytes in flight were 1460. The current sequence number is 1, the source port is 80 while the destination port is 1198. Since for slow start phase: Congestion window=congestion window +MSS, then it is expected that when packet 12 is acknowledged, the total bytes in flight will be 2920. Figure 9 shows the details obtained for packet 13.

This Figure 9 illustrates that for packet 13, the total number of bytes in flight is now 2920. The current sequence number is 1461 while the next sequence number is 2921. This is actually in agreement with the calculated value for the new congestion window of 2920.

Therefore, it is expected that for packet 14, the congestion window will be 2920+1460, giving a value of 4380 for the bytes in flight. This is confirmed by Figure 10.

Once again, the values agree with the theoretical values of 4380 for the bytes in flight, current sequence number of 2921, and next sequence number of 4381. It was observed that the value of the congestion window continued to rise up to a value of 65011 for the calculated window size for packet number 297 shown in Figure 11.

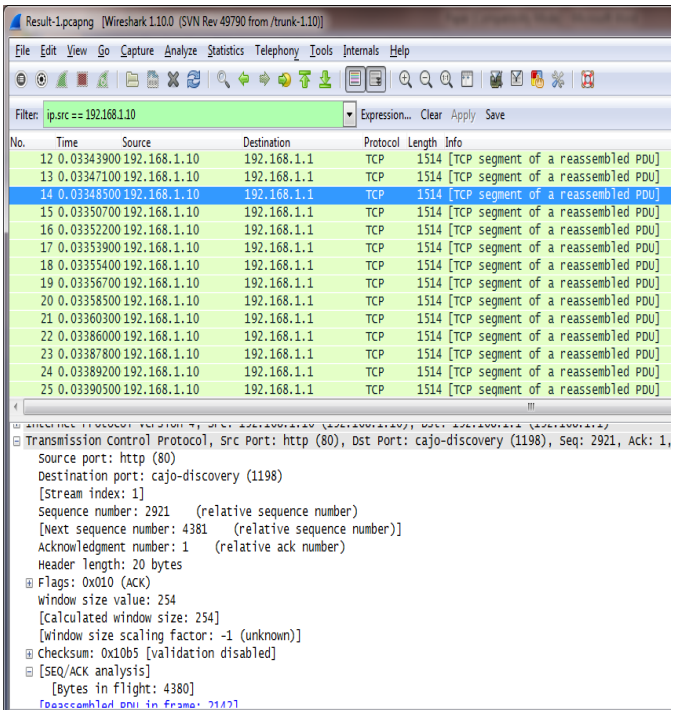

Figure 10: Congestion Window after ACK for Packet 13

After this, the value of bytes in flight went down to 1460 bytes for packet number 298, confirming the fact that the TCP had entered into congestion avoidance phase similar to that of TCP Tahoe, where the congestion window is set to 1 MSS when congestion occurs. During the slow start phase, the source can send data up to the least of the value of the congestion window and the receiver advertised window. This is the lower bound of the sender's TCP window size.

RESEARCH ARTICLE

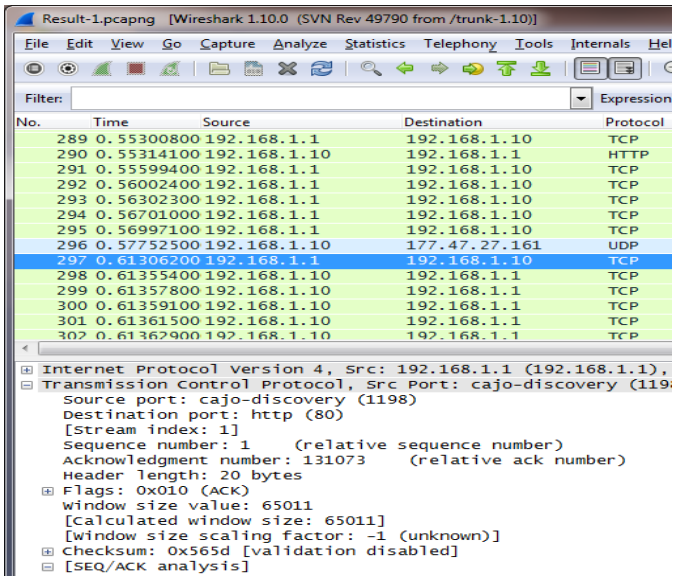

Figure 11: Determination of the Slow Start Threshold Value

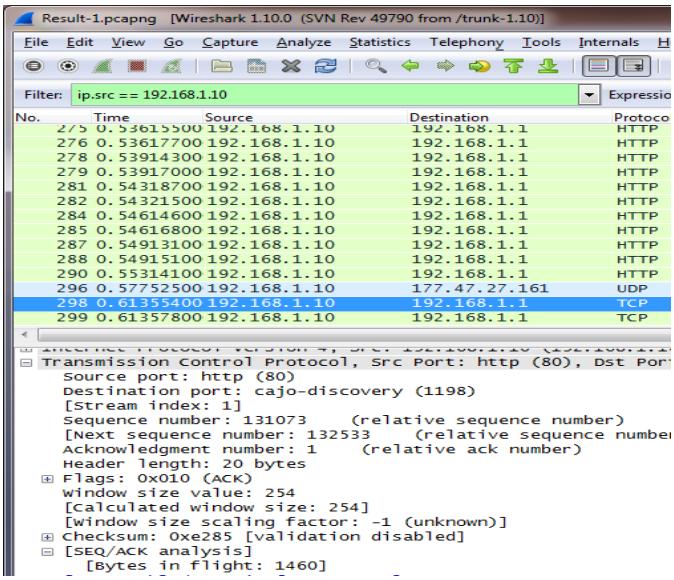

Figure 12: Onset of Congestion Avoidance Phase

After packet number 297, there is drop of byte in flight to a value of 1460. Therefore, the calculated window size for packet number 297, which s 65011 bytes, is the slow start threshold value (ssthresh). This is a close approximation to the theoretical value of 65535 bytes.

Indeed as shown in Figure 12, the value of bytes in flight dropped to a value of 1460, which is equivalent to 1 MSS. The receiver advertised window also falls down to 254 bytes, down from 65011 bytes further confirming that congestion avoidance phase has been entered.

5.3. I/O For Congestion Window Determination

When the congestion window reaches 65535 bytes (slow start threshold value), the TCP window is at full capacity as shown in Figure 13. At this window size, the sender is effectively blocked and the receiver cannot accept any data. This is confirmed by the statement, *'previous segment not captured'*, meaning that this segment was dropped and therefore need to be retransmitted. A plot of amount of data that the sender has in flight against time can be used to approximate the value of the congestion window.

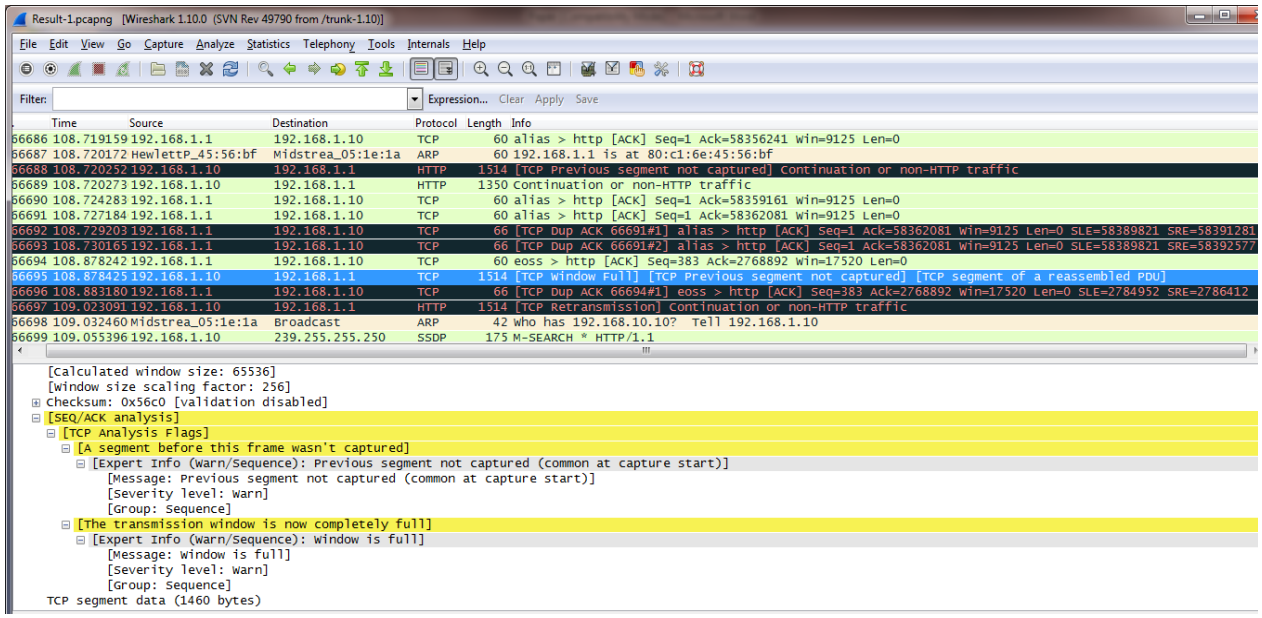

Figure 13: Congestion Window Full

RESEARCH ARTICLE

This is because this window size cannot be determined directly from the Wireshark traces. The trace of bytes in flight against time is shown in Figure 14.

In essence, the I/O graph displays the TCP receiver advertised window size over a given period of time. It has already been established that the *ssthresh* value is nearly *65535* bytes. Using a scale of *1* to *1000* puts this figure to a value of *655.35* bytes as shown in the graph above.

Therefore in situations where the amount of data that is transmitted across the network reaches the size of the receive window, the slow start algorithm is abandoned and the flow of data is influenced by the receiver using the advertised window size.

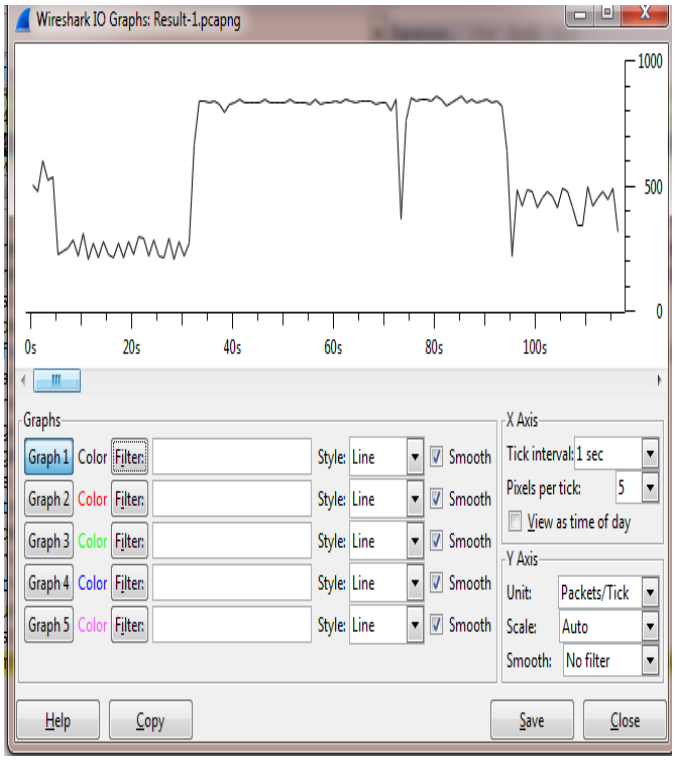

Figure 14: I/O Graph

5.4. Fast Retransmit Phase

The fast retransmission occurs when packet loss is detected. This is normally prompted by the receipt of three duplicate acknowledgments. Figure 15 shows that indeed there was duplicate acknowledgments, indicated by *'Duplicate ACK(#1)'*, *'Duplicate ACK(#2)'* and '*Duplicate ACK(#3)'*.

Upon collapsing the first duplicate, the information in Figure 16 was obtained. This figure shows that among others, packet *66692*, *66696* and *66798* were duplicate acknowledgements. To get more information on the first duplicate *66692*, this packet number was double clicked to reveal the information in Figure 17.

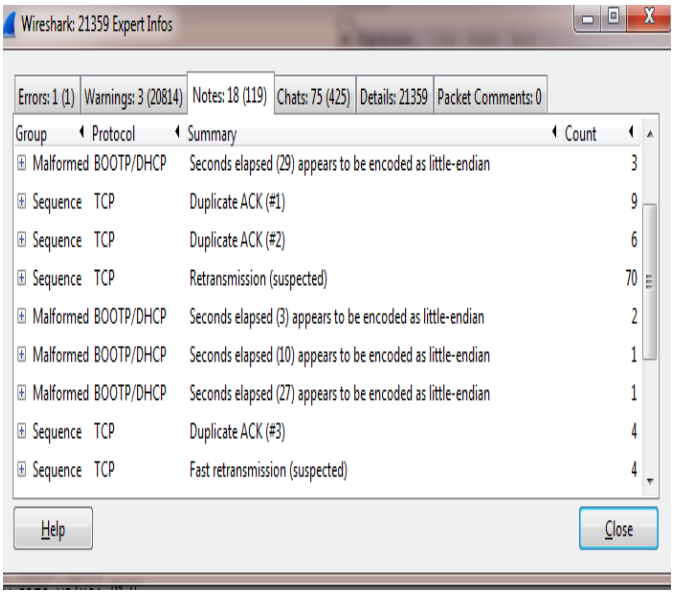

Figure 15: Duplicate Acknowledgements

This Figure 17 gives much information concerning which particular frame has received duplicate acknowledgement.

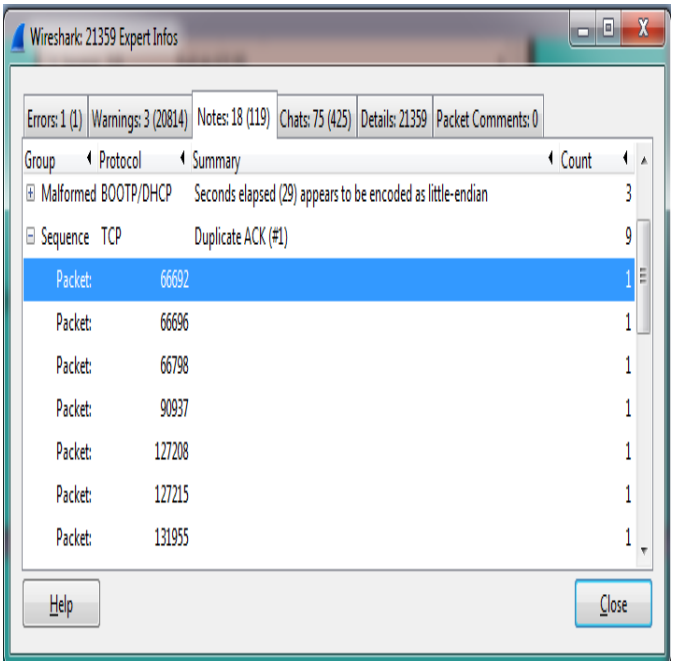

Figure 16: Duplicated ACK Packet Numbers

For this case, it is indicated that frame *66691* has received a duplicate acknowledgement. The expected sequence number is *58362081*. After three acknowledgements (Packets numbers *66691*, *66692*, *66693*) for the same sequence number are received at the sender, the transmitter carries out an immediate retransmission of the missing segment with packet *66697* as shown in Figure 18.

RESEARCH ARTICLE

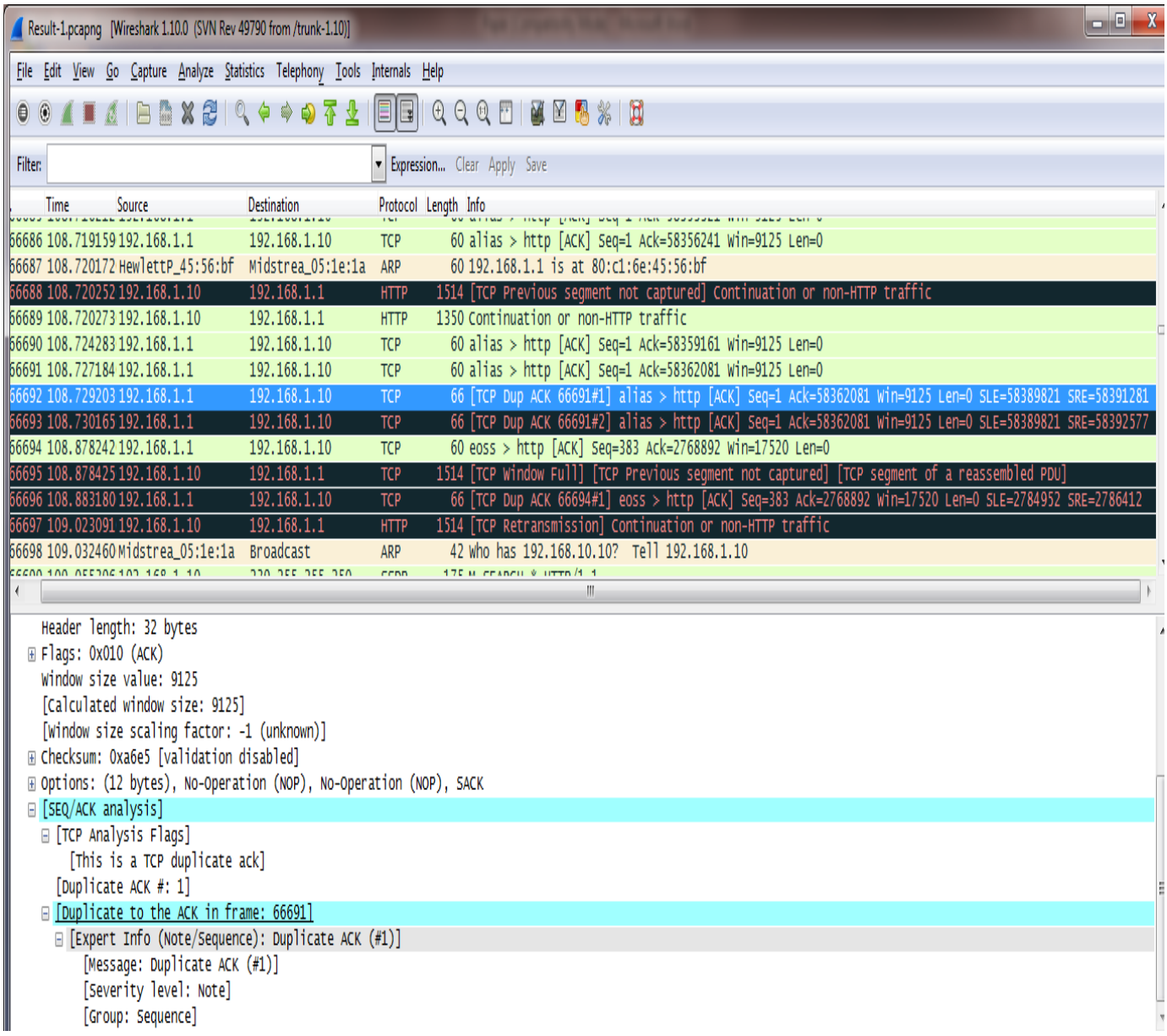

Figure 17: Details of Duplicated Acknowledgements

Interesting to note is that the Selective Acknowledgements (SACKS) option is set by the receiver, indicating that the sender is informed of the data that has been received out of order, by use of left edge and right edge SACKs (SLE and SRE). From this figure SLE=*58389821* while SRE =*58391281*.

The start of a block is indicated by SLE (Left edge) while the end of a block is indicated by SRE (Right edge). Consequently, the data source can only retransmit the missing data segments.

Other packets were also retransmitted as shown by Figure 19. This figure demonstrates that packets *666697, 66700, 66702* among others were fast retransmitted by the sending laptop.

These are the packets that arrived out of order or lost and were therefore received duplicate acknowledgments, prompting their retransmissions.

When TCP enters the fats retransmit phase, the missing segments are retransmitted before the retransmission timer expires.

This algorithm also sets the value of *ssthresh* to half the current congestion window, as well setting the current congestion window to a value that is 3MSS more than *ssthresh*. Since TCP with SACK is an extension of the TCP Reno algorithm, it does retain the slow start and retransmit features of Reno.

RESEARCH ARTICLE

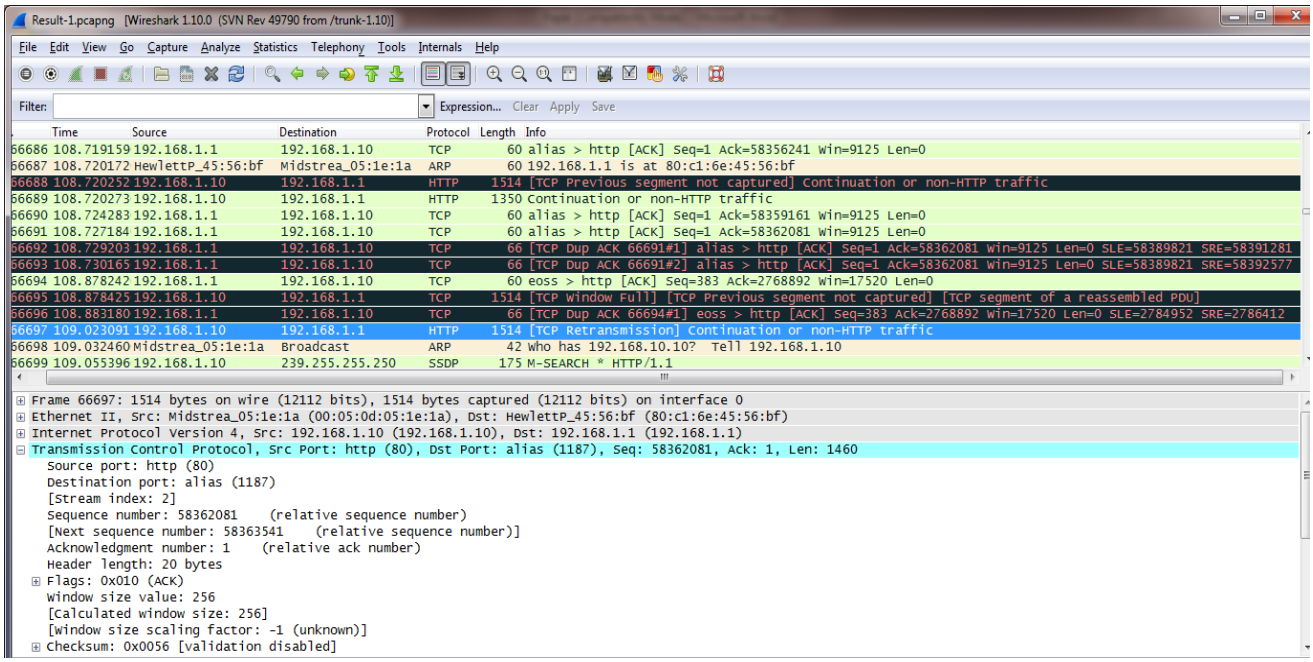

Figure 18: Retransmission of Packet with Sequence Number 58362081

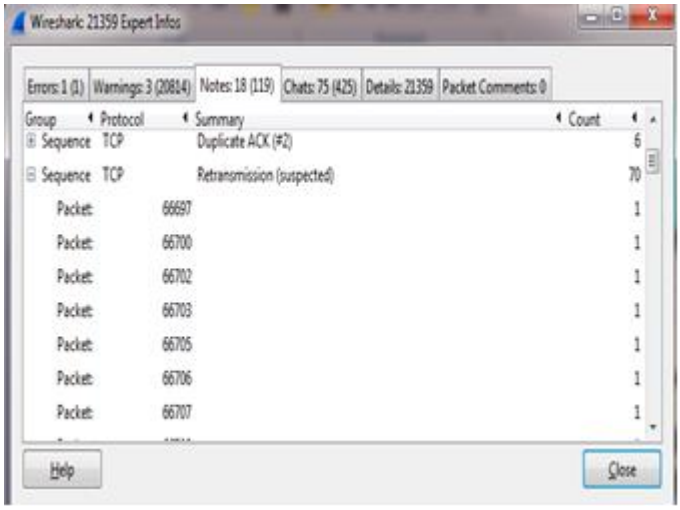

Figure 19: Retransmission of More Packets

5.5. Fast Recovery

This phase controls the transmission of new data, after the retransmission of missing segments, until the first non-duplicate acknowledgement has been received at the sender. Figure 20 shows massive retransmissions of missing segments.

After the acknowledgements of all these retransmissions, fast recovery takes control until the new segments send are acknowledged. This means that when packet number *66723* is acknowledged, fats recovery steps in. packets number *66724* to *66727* all acknowledge previously retransmitted packets. However, packet number *66743* is an accumulative acknowledgement and it acknowledges all pending packets up to packet number *66730*.

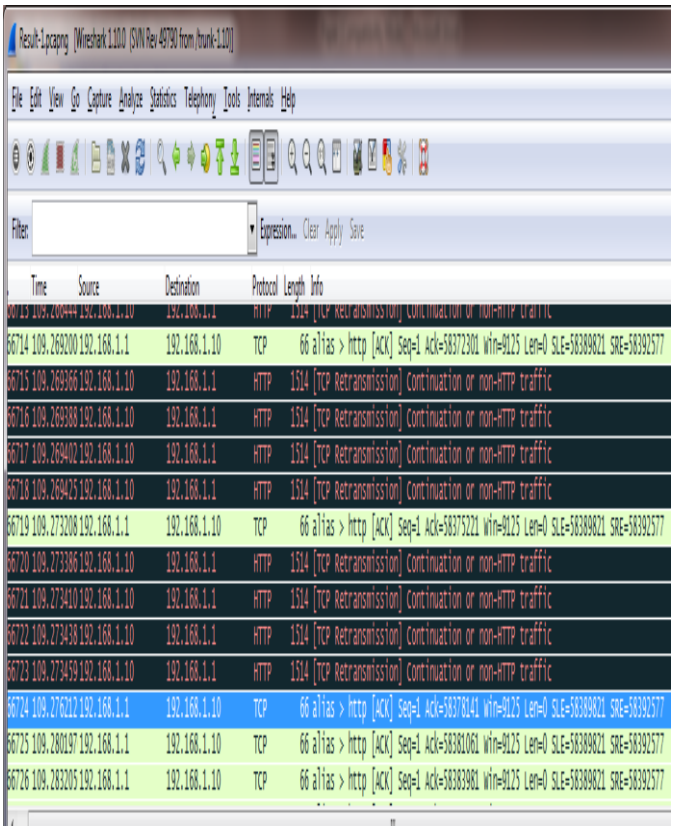

Figure 20: Massive Segment Retransmissions

RESEARCH ARTICLE

Therefore after packet number *66731* is ACKed, fast recovery is initiated. This happens at packet number *66746*, which acknowledges both packets number *66731* and *66732*. Therefore after packet number *66746*, TCP brings into effect the congestion avoidance algorithm.

5.6. Time-Sequence Graph (TCP Trace)

This graph is a plot of sequence numbers against time. It is used to show the maximum bandwidth for the communication link during the period when communication is taking place between end systems. The gradient of this curve gives the network data rates.

It is clear from the graph displayed in Figure 21 that the network bandwidth is not uniform. The implication of this is that the bandwidth keeps changing over time. During the initial stages (Time 0 to 32 seconds), the gradient is low, indicating the transmission of a small number of packets. However, between 32 seconds and 90 seconds Time- duration, the gradient rises steeply clearly indicating high network bandwidth (slow start phase that doubles MSS values). The gradient slags gain between 90 and 110 seconds duration, indicating the congestion avoidance phase.

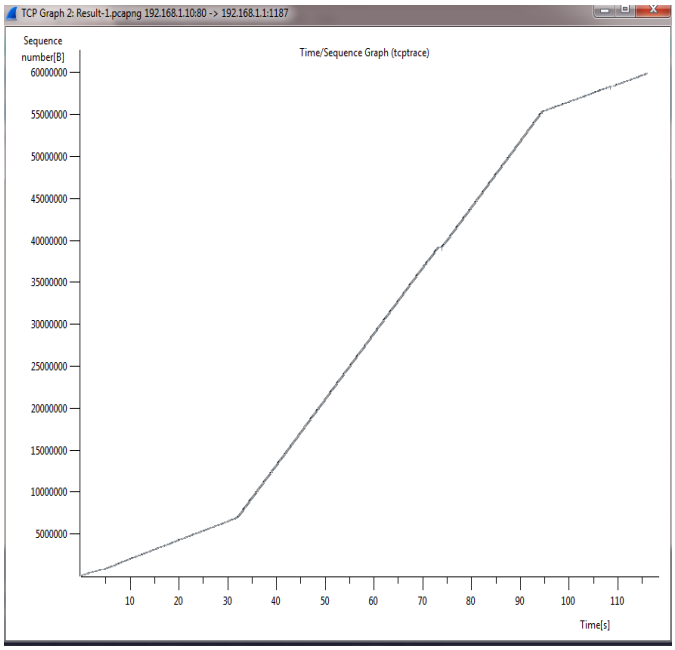

Figure 21: Time – Sequence Graph

6. PROPOSED ROUND TRIP TIME - BASED ADAPTIVE CONGESTION CONTROL

The evaluation of the current TCP congestion control algorithms has revealed that most of them are reactive in nature and involve massive retransmission of packets deemed to have been lost. The criterion for the activation of congestion control is the receipt of three duplicates. In this paper, a novel congestion control mechanism employing round trip times is suggested as the possible solution to address the shortcomings of the current congestion control algorithms.

Four parameters are crucial in this proposed algorithm: the retransmission timer (RT), round trip time (RTT), receiver congestion window (cwnd) and the receiver advertised window (rwnd) as shown in Figure 22.

Normally, TCP maintains a Retransmission Timer (RT) for each connection. This timer is started during a transmission. A timeout of the RT causes a retransmission. The *rwnd* dictates the value of *cwnd*, which is equivalent to the size of the current receiver buffer window.

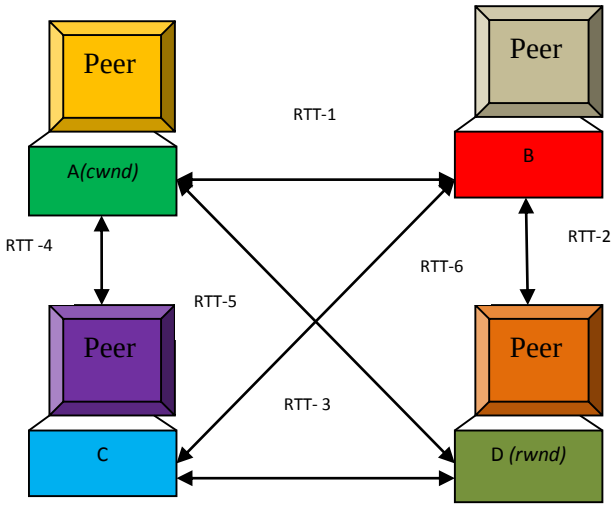

Figure 22: Round Trip-Based Adaptive Congestion Control

6.1. Parametric Selection

The size of *cwnd* determines the number of data packets that could be sent through a given path. The round trip time (RTT) refers to the duration taken by a probe packet to travel from the source to the destination and back.

6.2. Link Selection

In this proposed congestion control algorithm, communication links are evaluated statistically using the RTT value as criteria. A path with the least cost (PLC) in terms of RTT will be the suitable candidate for data packet transmission. In Figure 22, the total RTT values for the possible links in this mesh network are computed as follows:

Path A—D: *Total RTT value = RTT- 5,*

Where RTT-5 is the RTT value for a direct link between A and D.

Path A—B--D: *Total RTT value = RTT- 1 + RTT- 2,*

Where RTT-1 is the RTT value for a direct link between A and B and RTT-2 is the RTT value for direct link between B and D.

RESEARCH ARTICLE

Path A—C--D: *Total RTT value = RTT- 4 + RTT-3,*

Where RTT-4 is the RTT value for a direct link between A and C and RTT-3 is the RTT value for direct link between C and D.

Path A—C—B---D: *Total RTT value = RTT- 4 + RTT- 6 + RTT- 2,*

Where RTT-4 is the RTT value for a direct link between A and C , RTT- 6 is the RTT value for direct link between C and B, and RTT –2 is the RTT value for a direct link between B and D.

6.3. Routing Table Updates

In ideal circumstances, the routing table stores numerous paths which the packet can utilize as it traverses the network. This proposed algorithm will require that the routers maintain optimum paths based on the shortest Total RTT values. Obviously, congestion in a given link will result in longer total RTT values and these changes should be recorded as path updates in the routing table.

6.4. Mode Of Operation

The intent of this paper was to evaluate the modus operand of TCP congestion control algorithms in mesh networks. Due to the limitations noted in the current TCP implementation of congestion control such as heavy packet retransmissions due to the receipt of three duplicates, some of which may be occasioned by network delays (which may ultimately lead to packet re-ordering) rather than packet loss, an adaptive congestion control based on round trip times (RTT) values rather than three duplicates is suggested.

In this new algorithm, the total RTT values will be cached in the routers' memory. Any network delays cause the elongation of RTT values. Suppose a given path is currently selected as the optimum link. If excessive traffic flows though this link, congestion may occur, leading to higher RTT values.

Therefore, a re-calculation of new RTT values may be selected another different path as the optimum one for data transmission. Since these RTT computations will be accomplished statistically at the start and during transmissions, all the mesh network paths will have a better chance of transmitting data. The consequence is that packets belonging to a single message may take different paths to the destination, which leads to faster traffic transmissions.

In so doing, there is an efficient handling of congestion in mesh networks and the many redundant paths will be assured of being used to transmit packets at any particular moment. This will prevent overwhelming few links with data packets while the rest of the links lie idle.

Therefore, instead of waiting for three duplicates to detect congestion, and perform packet re-transmissions, statistical measurements of RTT values will suffice, prompting packet transmissions via links with shorter RTT values.

7. CONCLUSION

The aim of this paper was to evaluate TCP congestion control mechanisms in a mesh network. Parameters such as packet loss, retransmissions, network bandwidth, I/O graphs, Time - sequence graphs were utilized to gauge its performance in these networks. Specifically, the congestion window, slow start threshold value and congestion detection and reaction were used to identify the kind of TCP implemented in the communicating parties.

The behavior of TCP during congestion avoidance was compared to the theoretical established behaviors. It was noted that the experimental values agreed with the theoretical values. In conclusion, it was established that TCP indeed carries out a three way handshake before any data can be sent over the communication links.

Moreover, it was clear that TCP indeed has inbuilt mechanisms for dealing with network congestion, namely slow start, congestion avoidance , fast retransmit and fast recovery algorithms. Due to the noted poor handing of congestion in mesh networks as a result of reliance on the receipt of three duplicates, an adaptive congestion control algorithm employing the round trip times as the criteria was suggested.

In this new algorithm, congestion in one of the mesh links causes its RTT vale to increase. This means that a re-computation of new RTT values may render this link non-optimum, and hence other links with lesser RTT values will be employed to transmit packets. In this way, the overwhelming of fewer inks with traffic at the expense of other links will be avoided. Ultimately, the message packets take different routes and hence link utilization is distributed among the available links. This leads to better control of congestion in mesh networks. Future works lie in the practical implementation of this adaptive round trip time-based congestion control algorithm in real world mesh network.

REFERENCES

[1] H. Dave, V. Gupta, & P. Dihulia, "Performance comparison between TCP sack and TCP Vegas using NS-2 Simulator," International Journal of Computer Application, 68(11), pp. 49-52, 2013.
[2] M. Kavitha, B. Ramakrishnan and R. Das, "A Novel Routing Scheme to Avoid Link Error and Packet Dropping in Wireless Sensor Networks," International Journal of Computer Networks and Applications (IJCNA), 3(4), PP: 86-94, 2016, DOI: 10.22247/ijcna/2016/v3/i4/48569.
[3] N. Vlajic , "TCP: Congestion Control," CSE 3214, PP: 1-20, 2016.
[4] S. Stefan, N. Cardwell, D. Wetherall, and T. Anderson, "TCP Congestion Control with a Misbehaving Receiver," Department of Computer Science and Engineering: University of Washington, Seattle, PP: 1-8, 2015.
[5] R. Shankar and S. Naidu, "A Dynamic Approach of Malicious Node Detection for Internet Traffic Analysis," International Journal of Computer Networks and Applications (IJCNA), 1(1), PP: 33-39, 2014.
[6] Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm," Standards Track, PP: 1-16, 2012.

RESEARCH ARTICLE

[7] A. Walid, Q. Peng, J. Hwang, and S. Low. "Balanced Linked Adaptation Congestion Control Algorithm for MPTCP", Working Draft, IETF Secretariat, Internet-Draft, 2015.
[8] Ahmed and N. Shinde, "A Simulation Technique for Wireless Mesh Networks to Present Its Topology and Evaluate Its Impact on Communication Revolution," Communications and Network, 2016, DOI: 10.4236/cn.2016.81005.
[9] M. Anuba and A. Anuja, "Energy Efficient Routing (EER) For Reducing Congestion and Time Delay in Wireless Sensor Network," International Journal of Computer Networks and Applications (IJCNA), 1(1), PP: 1-10, 2014.
[10] M. Hamzah, A. Hijawi and M. Mohammad, "Performance Analysis of Multi-Path TCP Network," International Journal of Computer Networks & Communications (IJCNC), Vol.8, No.2, PP: 145-147, 2016.
[11] Cao, Yu, Mingwei Xu, and Xiaoming Fu, "Delay-based congestion control for multipath TCP," In Network Protocols (ICNP), IEEE International Conference on, pp. 1-10, IEEE, 2012.
[12] J. VijiPriyal , S. Suppiah, "An Extended Study On Newton Raphson Congestion Control", International Journal of Advanced Research, Volume 4, Issue 2, 2016.
[13] Alrshah, Mohamed A., et al. "Agile-SD: a Linux-based TCP congestion control algorithm for supporting high-speed and short-distance networks." Journal of Network and Computer Applications 55 (2015): 181-190.
[14] Kire Jakimoski, Slavcho Chungurski, Sime Arsenovski, Lidija Gorachinova, Oliver Iliev, Leonid Djinevski and Emilija Kamcheva. "Performance Analysis of Linux-Based TCP Congestion Control Algorithms in VANET Environment," International Journal of Future Generation Communication and Networking *(IJFGCN)*, Vol. 4, Issue 2, 2016.
[15] S. Ogara, "Lecture Notes: Flow Control, Congestion Control and Error Control," PP: 1-40, 2016.

Authors

Vincent O. Nyangaresi is currently a student researcher in areas of data communication and computer networks, network design and administration, distributed systems and information systems security. He has published numerous research articles covering areas such as communication systems, secure network communications, systems acceptance modeling, TCP architecture and design, radio wave propagation, virtualization and cloud computing, among others.

Dr. Solomon O. Ogara, B.Sc. (Egerton), B.Sc. (Arizona), M.Sc. (Dakota), Ph.D. (North Texas) is currently the Chairperson of the Department of Computer Science & Software Engineering, Jaramogi Oginga Odinga University Of Science And Technology. He has worked as an assistant professor of computer information system at Livingstone College. He has taught different computer information systems and networking courses including: Introduction to Computer Information Systems; Object Oriented Programming; Decision Support & Business Intelligence, Computer Architecture & Organization; System Analysis and Design, Web Design using HTML5; Database Management, Enterprise Network Design, Wired, Optical and Wireless Communications; Voice/VoIP Administration; Operating Systems with UNIX and Windows Server; Data, Privacy and Security; Principles of Information Security.

Dr. Silvance O. Abeka is currently a Senior Lecturer and a Dean-School of Informatics and Innovative Systems of Jaramogi Oginga Odinga University of Science and Technology. He worked previously as a Director- Institute of Open and Distance Learning at Africa Nazarene University and also as a Dean- Faculty of Applied Science and Technology of Kampala International University- Dar es Salaam Collage. He holds a Ph.D in Management Information System (MIS), Masters of Science in Computer Science from University of Da es Salaam and Master of Business Administration (Information Technology) from Kampala International University. His research interests include IT innovation adoption, open source software study, IT offshoring, Management Information Systems, Foundations of Network and System Security, Impact of Digital Technologies on Society, Networking Protocols and Topologies, Web- Design and E- Learning Technologies.