Effect of TCP Buffer Size on the Internet Applications

Imtiaz A. Halepoto[1], Nazar H. Phulpoto[2], Adnan Manzoor[2], Sohail A. Memon[3], Umair A. Qadir[2]
[1]Department of Computer Systems Engineering, QUEST Nawabshah, Pakistan
[2]Department of Information Technology, QUEST Nawabshah, Pakistan
[3]Department of Mathematics, SALU Khairpur, Pakistan

***Abstract*—The development of applications, such as online video streaming, collaborative writing, VoIP, text and video messengers is increasing. The number of such TCP-based applications is increasing due to the increasing availability of the Internet. The TCP protocol, works at the 4th layer of the Internet model and provides many services such as congestion control, reliable communication, error detection and correction. Many new protocols have been proposed such as stream control transmission protocol (SCTP) with more features compared to the TCP. However, due to the wide deployment, TCP is still the most widely used. TCP creates the segments and transmit to the receiver. In order to prevent the errors TCP saves the segments into the sender buffer. Similarly, the data is also saved at the receiver buffer before its transmission to the application layer. The selection of TCP sender and receiver buffer may be varied. It is very important because many applications work on the smartphones that are equipped with a small amount of memory. In many applications such as online video streaming, some errors are possible and it is not necessary to retransmit the data. In such case, a small buffer is useful. However, on text transmission the complete reassembly of message is required by the TCP before transmission to the application layer. In such case, the large buffer size is useful that also minimizes the buffer blocking problem of TCP. This paper provides a detailed study on the impact of TCP buffer size on smart-phone applications. A simple scenario is proposed in NS2 simulator for the experimentation.**

Keywords*—TCP; sender buffer; receiver buffer; stream control transmission protocol (SCTP); error detection and correction

I. Introduction

The OSI model in the Internet provides a step by step characterization of the computer and telecommunication systems. Transport layer in the OSI model is one the main layers. It provides congestion control, error control, flow control, a stronger checksum and many other features. In a summarized way, the transport layer works for successful delivery of a process from a sender to a receiver. All of these features are provided the TCP protocol [1]. The two other famous protocols of the transport layer are the User Datagram Protocol (UDP) [2] and SCTP [3] (see also [4]). The UDP is mainly beneficial for applications such as video streaming. It is a less complicated protocol due to its header format but not preferable for applications where the reliability is mandatory. The SCTP is a new protocols and it is still in development phase. The key features in the design of the transport layer are the following:

- **Out-of-order delivery** for faster data transmission to the application layer. In this mode, SCTP the receiver does not wait for complete message it simply forwards the data as soon as it is received. This feature is also available in the UDP. However, it is not available in TCP. One of the main application of out-of-order data is the online video or audio streaming. However, in both the sequenced-data delivery and out-of-order data delivery in online streaming may loss few of the segments. But the overhead of sequencing overhead in out-of-order data is less.
- **Connection-orientation** feature is available in the TCP and SCTP but not in UDP. By this features both the sender and receiver initiate a connection establishment procedure before the data transmission. In UDP all the data units travel independently and forwarded by different routers.
- **Connection formation** is initiated by the SCTP and TCP before the data transmission. The connection formation procedures requires verification of sender and receiver, which also improves the security of the protocols. TCP and SCTP uses 3-way and 4-way handshake procedures for connection formation. However, there is no service of connection formation in UDP.
- **Connection termination** is also completed by the TCP and SCTP after the successful transmission of data from sender to receiver. By this method the sender and receiver agree to close the session. There is no connection termination service provided by the UDP.
- **Reliability** by means of acknowledgment to the sender. This feature is available in TCP and SCTP but not in UDP. One the main factor that affects the buffer size is reliability. For example, a sender keeps a copy of the transmitted segment in the sender buffer until the acknowledgment is received or the time to acknowledgment expires.
- **Flow control** to maintain the data transmission rate of the sender. By flow control the protocol reduces the chances of network congestion and other others errors such as data overflow. With the flow control, TCP tries to maintain a synchronization between the sender and the receiver. For example, the synchronization is needed when the sender is very fast compared the receiver.

Many of the features of the transport layer protocols are presented in Fig. 1. Despite all the good features of SCTP, the TCP is currently fully operational over the Internet. In

TCP, the segments that are in queue for transmission and the segments that are revived are stored in a memory called the TCP sender and receiver buffer. Buffer size play major role in the performance of TCP. If the buffer size is too small the TCP would be unable to complete the message by combining the segments at the receiver. It also leads to no buffer space for the parallel TCP flows of other applications. Such type of buffer condition is called the receiver buffer blocking. Similarly, on the sender side, if the buffer size is small it affects the transmission rate due the less number of segments within the sender buffer. With the increasing number applications such as video streaming, messengers, online chats, VoIP, collaborative scientific projects, wireless sensors and monitoring. It is difficult to decide the buffer size requirement. Because some of the applications require reliability and some of them do not. Further, the development of smart-phone apps is also increasing. It is difficult to determine which type of data processing will be carried by the apps at the time of development, because of the real time data processing and software updates. In order to help the developers of smart-phone apps, the consideration of buffer size for the protocol is necessary. Additionally, the researchers are working on the parallel data transmission by using more than one NIC cards. By parallel transmission throughput increases by the factor depending on the number of NIC cards. For the parallel data transmission a new version of TCP is under development. It is called the Multipath TCP (MPTCP) [5]. This research work aims to provide the experimentation of the TCP protocol with various buffer size. The proposed scenario is composed of multihop network. The background traffic is also added in order to make the scenario more like as real life networks where the bandwidth is occupied by the several number of users. The experimentation results are also useful for evaluation of MPTCP. The simulation is carried in NS2 with varying size of the sender and receiver buffer.

The rest of the paper contains the related work in Section II. The experimental setup and configuration details in Section III. The analysis on the basis of the results is presented in Section IV. The conclusions are summarized in Section V.

II. Related Work

The choice of buffer size affects the performance of TCP. For example, if the receiver buffer size equal to the 50 segments. If the there are two processes transmitted by the sender. Each of the process contains 30 segments. In simultaneous transmission of both the processes the receiver will be occupied by the 50 segments. 25 segments from each of the process. Both the processes are received incomplete. The receiver will be waiting for the remaining segments and none of the process will be delivered to the application layer. This kind of situation is called the receiver buffer blocking. Many researchers reported the problem of receiver buffer blocking while using TCP [7], [8], [9], [18], [19]. The researchers also suggested the use of retransmission policies for the transmission missing data of one process. However, such retransmission polices are beneficial for the parallel transmission of data by using more than one link. For one link between a sender and receiver the role of retransmission policy slightly improves performance.

The buffer splitting techniques were proposed by the researchers in [10] and [11]. They proposed two kinds of splitting. First, that equally divide the buffer space in the number of destinations or paths. In real life data from different paths to receiver take different time. So, on the slow path (path of smaller bandwidth or longer propagation delay) data transmission may affect the data that is already in the receiver buffer. On the other side the faster paths occupy more buffer space and may reach to the buffer overflow. Second, the technique, which divides the buffer into parts for the different processes according to outstanding data. The outstanding data is the data that already transmitted by the sender but not yet acknowledged. The work in [12] suggested the use of available buffer space in acknowledgment segment, because this value represents the exact free space of the buffer. Normally, TCP uses the advertised buffer space in the acknowledgment segment. The relationship between the buffer size and the round trip time (RTT) is investigated by Want *et al.* [13]. According the their findings the relationship is linear.

Feature	SCTP	TCP	UDP
Connection-oriented	Yes	Yes	No
Full-duplex	Yes	Yes	Yes
Reliable Data Transfer	Yes	Yes	No
Partial reliable data transfer	Optional	Yes	No
Ordered data delivery	Yes	Yes	No
Unordered data delivery	Yes	No	Yes
Flow control	Yes	Yes	No
Congestion control	Yes	Yes	No
ECN capable	Yes	Yes	No
SACK	Yes	Optional	No
Preservation of Message Boundaries	Yes	No	Yes
Path MTU Discovery	Yes	Yes	No
Application PDU Fragmentation	Yes	Yes	No
Application PDU Bundling	Yes	Yes	No
Multistreaming	Yes	No	No
Multihoming	Yes	No	No
Protection against SYN flooding attacks	Yes	No	N/A
Allows half-closed connections	No	Yes	N/A
Reachability Check	Yes	Yes	No
Pseudo-header for Checksum	No	Yes	Yes
Time Wait State	for vtags	for 4-tuples	N/A

Fig. 1. Summary of the services provided by the transport layer protocols [6].

The work on RTT and the other path performance characteristics such as bandwidth is investigated by the researchers in [14]. The technique of buffer splitting at the sender and receiver is employed in order to reduce the buffer blocking problem. The splitting is performed on the basis of the RTT. The destinations with longer RTT value (slow paths) are allowed to use the small portion of the buffer size. Whereas, the destinations with shorter RTT value (fast paths) are configured to use the large portion of the buffer space both at the sender and receiver. The similar work to improve the performance by minimizing the buffer blocking is also presented in [15], where the technique of transmission scheduling are proposed. Scheduling of different data flows is based on a priority value,

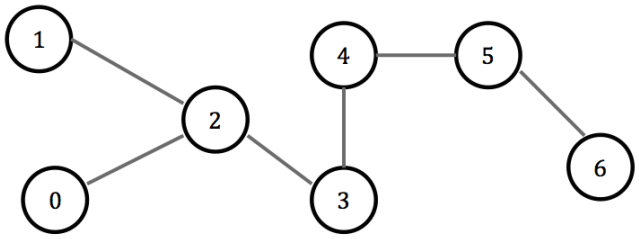

Fig. 2. Network Topology 1: Multihop network.

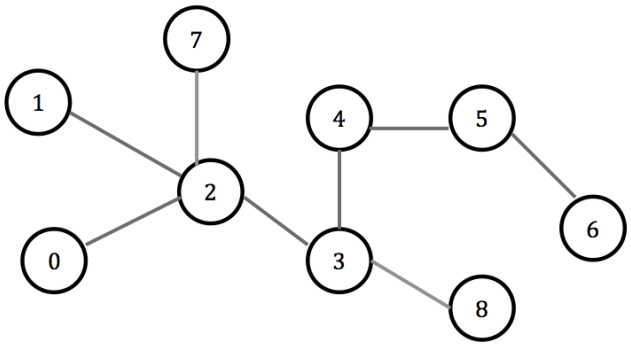

Fig. 3. Network Topology 2: Multihop network with background traffic.

which is calculated by using the outstanding data of each flow. The researchers in [16] suggest that the design of a routing protocol is very important.

III. TCP Implementation and Configuration

The NS2 [17] is used for the implementation and evaluation of the TCP. For the installation of NS2, the Ubuntu 14.04 OS is installed in the virtualbox. The multihop network is proposed for the experimentation. In all of the experiments the throughput is measured by assuming node 0 as source and node 6 as destination. Each of the simulation is repeated 10 times and the results are collect on the basis of average values. The implementation code of TCP is already available in the NS2, however its configuration is required according to the proposed topologies given below. The key parameters of the simulation are presented in Table I.

A. Topology 1

In this topology seven nodes are configured and attached as shown in Fig. 2. Node 0 and 1 are configured with TCP agents as source nodes. Node 6 is the destination node and it is configured with TCP agent as sink. In this topology the main connection that is monitored for throughput is 0–6, whereas 1–6 is just adding an additional TCP flow.

B. Topology 2

In the second topology, nine nodes are used. Nodes 7 and 8 are attached with the intermediate nodes 2 and 3. The main purpose these additional nodes is to provide the background traffic to the TCP. Topology 2 is shown in Fig. 3. The remaining parameters and their configuration are left on the default values present in NS2.

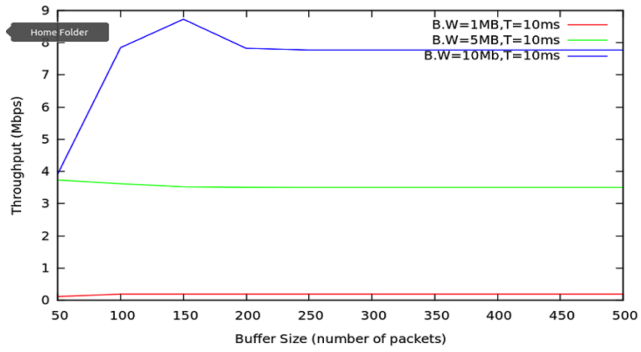

Fig. 4. Experiment 1a.

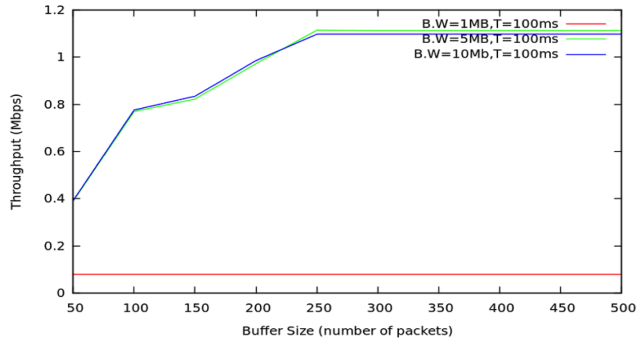

Fig. 5. Experiment 1b.

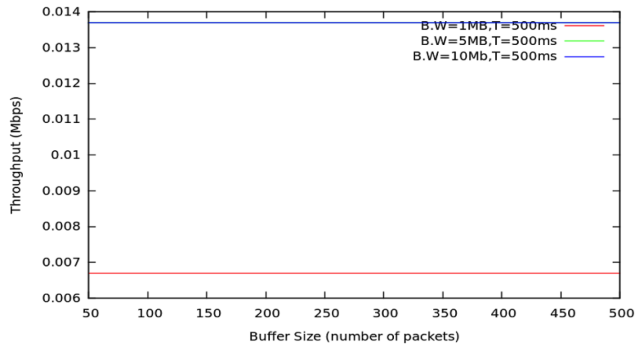

Fig. 6. Experiment 1c.

IV. Results and Discussions

A. Experiment–1

In this experiment, three kinds of simulations are performed each time while changing the values of the propagation delay. The value of sending buffer changes from 50 to 500 number of segments. It is observed that when the delay is small as in Fig. 4, the TCP throughput is directly proportional to the value of bandwidth, i.e. 10Mbps. When the delay increase as in Fig. 5 and 6, the medium is not fully occupied. Hence the throughput at 10Mbps and 5Mbps is also same. However, it is greater than the throughput at 1Mbps. It is also clear that the increase in the buffer size does not significantly affect the data transmission rate. The buffer size of 200–250 is enough to reach the maximum throughput.

TABLE I. PARAMETERS AND VALUES

Experiment	Nodes	Bandwidth(Mbps)	Delay(ms)	No. of Simulations	Network
1. Changes in the sender buffer					Topology–1
1a	0 to 6	1, 5, 10	10	10	
1b	0 to 6	1, 5, 10	100	10	
1c	0 to 6	1, 5, 10	500	10	
2. Background traffic					Topology–2
2a	0 to 6	1, 5, 10	10	10	
2b	0 to 6	1, 5, 10	100	10	
2c	0 to 6	1, 5, 10	500	10	
3. Large Buffer Size	0 to 6	1, 5, 10	10	10	
4. Changes in the Receiver buffer					Topology–2
4a	0 to 6	1, 5, 10	10	10	
4b	0 to 6	1, 5, 10	10	10	
5. Equal Buffer Size	0 to 6	1, 5, 10	10	10	Topology–2

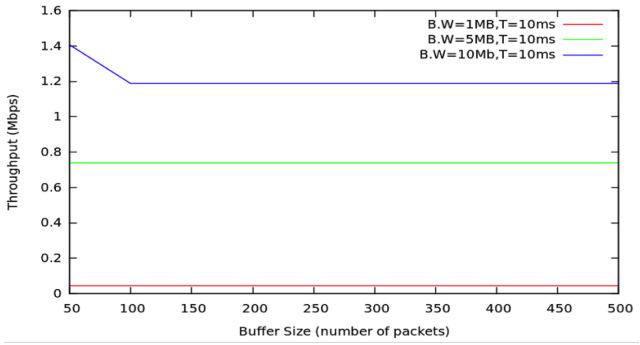

Fig. 7. Experiment 2a.

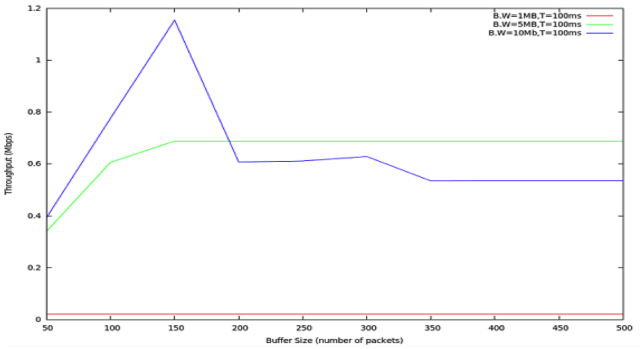

Fig. 8. Experiment 2b.

B. Experiment–2

The topology 2 is used in this experiment. A TCP flow is defined from node 3–8 and its affect on the flow of nodes 0–6 is observed. The same bandwidth and delay values of Experiment–1 are applied. The results trends are very similar to Experiment–1. The results are presented in Fig. 7, 8 and 9. When the buffer is smaller than 200 there is steady improvement in the throughput. However, with a larger buffer of more than 200 packets, the maximum throughput has reached.

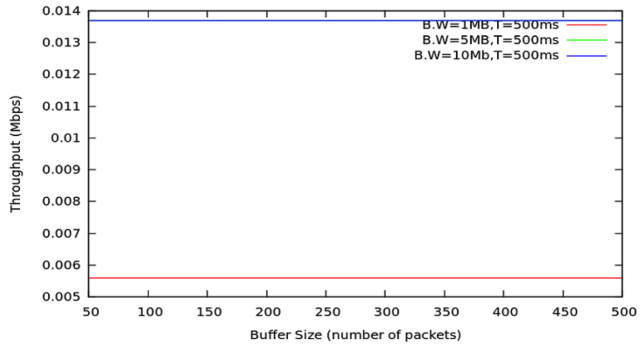

Fig. 9. Experiment 2c.

C. Experiment–3

In this experiment, the simulation of Experiment–2 is extended for a very large buffer size. The buffer size used is from 1000 packets to 10000 packets. The delay is configured to 10ms, however the experiment is repeated with different values of bandwidth, i.e. 1Mbps, 5Mbps and 10Mbps. This expriments proves that the large size of buffer is not useful in the proposed scenario, the throughput remains the same. In Fig. 10, the throughput at 1Mbps, 5Mbps and 10Mbps is 50Kbps, 0.7Mbps and 1.2Mbps.

D. Experiment–4

In this experiment, Topology–2 is used. However the changes are performed in the receiver buffer instead of the sender buffer. According the investigations, when the delay is smaller, the sender transmits data as long as it is available. Due to which there are less occasions of packet loss in the 0–6 TCP flow. So, the output remains at the same value as shown in Fig. 11 and 12, where different values of bandwidth are used. In the case of longer delay of 100ms, the small buffer does not reaches the larger throughput. But as the buffer size increases the the throughput also increases. After the buffer size of 250 packets, the additional space in the buffer space does not increases the throughput.

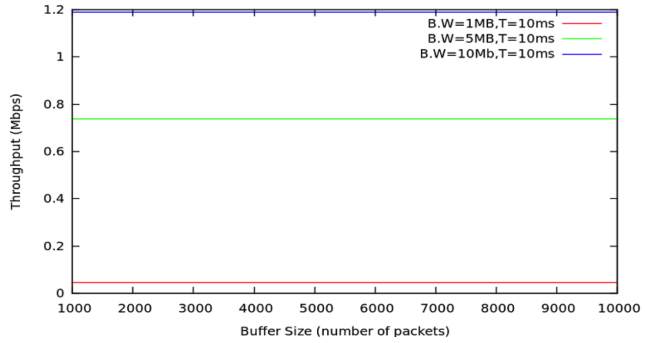

Fig. 10. Experiment 3.

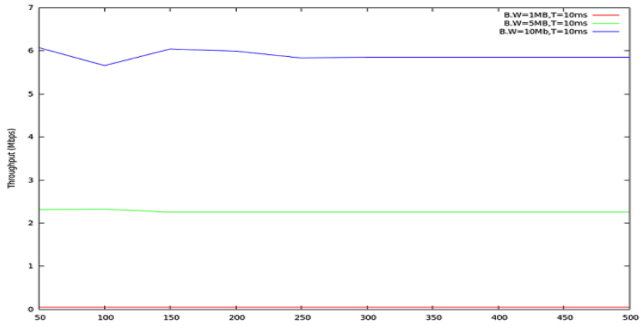

Fig. 11. Experiment 4a.

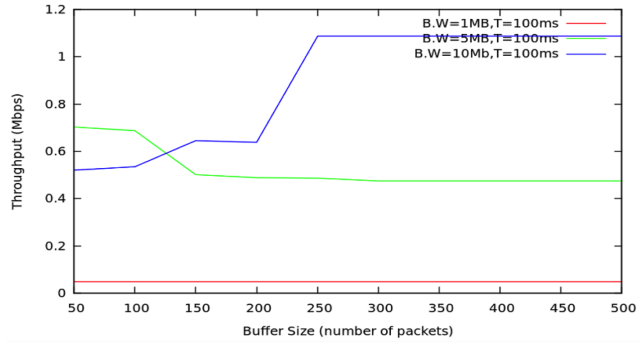

Fig. 12. Experiment 4b.

E. Experiment–5

The last experimentation is carried out by changing the buffer size both at the sender and receiver. In this experiment, the buffer size increases from 50–to–500 packets. It is observed that, when the buffer is small the throughout is less. When the buffer size is large it increase to the throughput but up 100 packets size. After the size of 100, the increase in the buffer space does not increase the throughput. The results of Experiment–5 are shown in Fig. 13.

In all of the experiments after a certain buffer space the performance of TCP remain the same in terms of throughput. According the Experiment–5, the significant amount of the buffer space is 100 packets. That is equal to the bandwidth delay product. In this experiment, the bandwidth delay product is 10Mbps * 10ms = 100K. The suggestion of the buffer size that is twice the bandwidth product is also presented in [6].

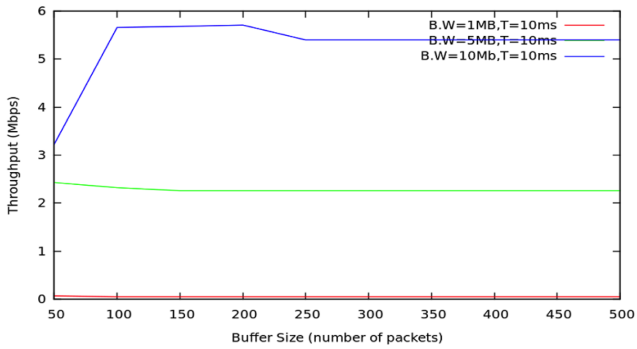

Fig. 13. Experiment 5.

V. Conclusion

The application development using Android or other platforms is increasing. The applications such as video/audio streaming, online collaboration, VoIP, messengers are the need of time. Some of them require sequenced delivery like collaborative writing projects, whereas some of them like online video streaming the sequenced delivery is not the priority. In video streaming the best and fast delivery is important. Many protocols are also available to deal with the sequenced and out-of-order delivery of data such as UDP, TCP and SCTP. TCP is one of the most widely used protocol over the Internet. Depending on the type of application the requirement of the buffer space at the sender and receiver is different. If not considered properly the buffer size the problem such as buffer blocking and buffer overflow may occur. This work provides the details of the experimentation of TCP with different buffer size options. According the results of the simulation over a multihop scenario, the too large buffer size does not increases the throughput. On the other hand the smaller buffer also degrades the performance of TCP. The finds suggest that the buffer size of twice the bandwidth delay product is suitable for the TCP flows. In future, the work may be extended on the upcoming version of TCP called the MPTCP.

References

[1] Allman, Mark, Vern Paxson, and Ethan Blanton. TCP congestion control. No. RFC 5681. 2009.

[2] Postel, Jon. User datagram protocol. No. RFC 768. 1980.

[3] Borella, Michael S. "System and method for control of packet data serving node selection in a mobile internet protocol network." U.S. Patent No. 7,346,684. 18 Mar. 2008.

[4] Halepoto, Imtiaz Ali, Adnan Ahmed Arain, and Umair Hussain. "Evaluation of multimedia streams in internet applications." Proceedings of the 8th International Conference on Information Systems and Technologies. ACM, 2018.

[5] Ford, Alan, *et al*. Architectural guidelines for multipath TCP development. No. RFC 6182. 2011.

[6] Halepoto, Imtiaz Ali. "Scheduling and flow control in CMT-SCTP." HKU Theses Online (HKUTO) (2014).

[7] Janardhan R Iyengar, Paul D Amer, and Randall Stewart. Performance implications of a bounded receive buer in concurrent multipath transfer. Computer Communications, 30(4):818829, 2007.

[8] Janardhan R Iyengar, Paul D Amer, and Randall Stewart. Receive buer blocking in concurrent multipath transfer. In Global Telecommunications Conference, 2005. GLOBECOM05. IEEE, volume 1, pages 6pp. IEEE, 2005.

[9] Halepoto, I. A., Memon, M. S., Phulpoto, N. H., Rajput, U., Junejo, M. Y. On the use of Multipath Transmission using SCTP. IJCSNS, 18(4), 58. Chicago, 2018.

[10] Hakim Adhari, Thomas Dreibholz, Martin Becke, Erwin P Rathgeb, and Michael Tuxen. Evaluation of concurrent multipath transfer over dissimilar paths. In Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on, pages 708714. IEEE, 2011.

[11] Thomas Dreibholz, Martin Becke, Erwin P Rathgeb, and M Tuxen. On the use of concurrent multipath transfer over asymmetric paths. In Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pages 16. IEEE, 2010.

[12] Rungeler, Irene, Michael Txen, and Erwin P. Rathgeb. "Congestion and flow control in the context of the message-oriented protocol SCTP." International Conference on Research in Networking. Springer, Berlin, Heidelberg, 2009.

[13] Yang, Wang, Hewu Li, and Jianping Wu. "Pam: Precise receive buffer assignment method in transport protocol for concurrent multipath transfer." Communications and Mobile Computing (CMC), 2010 International Conference on. Vol. 1. IEEE, 2010.

[14] Halepoto, Imtiaz A., Francis CM Lau, and Zhixiong Niu. "Management of buffer space for the concurrent multipath transfer over dissimilar paths." Digital Information, Networking, and Wireless Communications (DINWC), 2015 Third International Conference on. IEEE, 2015.

[15] Halepoto, Imtiaz A., Francis CM Lau, and Zhixiong Niu. "Scheduling over dissimilar paths using CMT-SCTP." Ubiquitous and Future Networks (ICUFN), 2015 Seventh International Conference on. IEEE, 2015.

[16] Bhangwar, Noor H., Imtiaz A. Halepoto, Intesab H. Sadhayo, Suhail Khokhar, and Asif A. Laghari. "On Routing Protocols for High Performance." Studies in Informatics and Control 26, no. 4: 441-448, 2017.

[17] Greis, Marc. "Marc Greis tutorial for the ucb/lbnl/vint network simulator ns." (2004).

[18] Halepoto, I.A, Sadhayo, I.H, Memon M.S, Manzoor A, Bhatti, S. Analysis of Retransmission Policies for Parallel Data Transmission. Engineering, Technology & Applied Science Research (ETASR), 8(3), 2018.

[19] Imtiaz A. Halepoto, Adnan Manzoor, Nazar H. Phulpoto, Sohail A. Memon and Muzamil Hussain, Mobility Management Using the IP Protocol International Journal of Advanced Computer Science and Applications (IJACSA), 9(5), 2018. http://dx.doi.org/10.14569/IJACSA.2018.090562