AMSE Association for the Advancement of Modelling & Simulation Techniques in Enterprises
IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Modelling, Measurement and Control A
Vol. 92, No. 1, March, 2019, pp. 30-36
Journal homepage: http://iieta.org/journals/mmc_a

An Adaptive Congestion Control Algorithm

Lal Pratap Verma[1*], Indradeep Verma[2], Mahesh Kumar[3]

[1] Moradabad Institute of Technology, Moradabad, Uttar Pradesh 244001, India
[2] IIMT College of Engineering, Greater Noida, Uttar Pradesh 244001, India
[3] Jaypee University of Engineering and Technology, Guna, Madhya Pradesh, India

Corresponding Author Email: er.lpverma1986@gmail.com

https://doi.org/10.18280/mmc_a.920105

Received: 25 October 2018
Accepted: 18 March 2019

Keywords:
TCP, congestion control, CWND, RTT, RTO, network, transport protocol, internet

ABSTRACT

There are various applications running over the Internet which generates a huge amount of traffic. So, Transport Layer takes the responsibility to manage this traffic and provides reliable, connection-oriented, end-to-end packet delivery service. Transmission Control Protocol (TCP) is a Transport Layer protocol, which provides these services. Each TCP variant provides a solution for specific problems. Utilization of the available bandwidth of the path with respect to the received ACK remains a challenge in long delay network. This paper presents a delay based congestion control approach which tries to maintain the data transmission according to the available capacity of the path. Simulation results show that proposed approach provides better results in terms of packet loss, throughput and inter-protocol fairness as compare to other protocol.

1. INTRODUCTION

Transmission Control Protocol (TCP) [1] contributes a lot to the remarkable growth of the Internet. As a result, the data transmission rate increased significantly. TCP implements the window-based flow control mechanism for congestion control. It uses two state variables to maintain the transmission between sender and receiver are Congestion Window (CWND) and Slow Start Threshold (SSTHRESH). The sender starts sending data according to the minimum of CWND and AWND (receiver advertised window).

The aim of CWND is to prevent the sender to send more than the network capacity in current load condition. Slow Start threshold provides a threshold value for congestion control mechanism. The objective of modifying the CWND is to adapt the current network status. This is to be done by using lost packet. The sender detects packet loss by Timeout mechanism or by duplicate acknowledgement. When the sender detects the packet loss, it retransmits the lost packet. Due to retransmission of the packet, the TCP sender assumes that there is congestion in the network. So, the TCP source modifies the CWND and SSTHRESH value according to the TCP protocol. Standard TCP [2] use additive increase and multiplicative decrease (AIMD) algorithm to adjust the CWND. This algorithm increases the CWND very slowly (Additive increase) and decreases it quickly (multiplicative decrease) according to acknowledgement (ACK) received by TCP source. In high-speed network condition, this algorithm is not able to fully utilize the available bandwidth of the network. A major challenge in such network condition is to quickly adapt the transmission rate according to the available bandwidth of the path.

Internet of Things (IoT) is the global network platform which provides interconnection between different types of devices having capability to transmit data over the network. The devices can be a computing device link personal computer, mobile phone, tablet, any object, animals or human having unique identity (IP address) over the Internet. There are many types of application protocols supported by IoT to provide data transmission between different devices. The XMPP (Extensible Messaging and Presence Protocol) [3], RESTful HTTP [4], and MQTT (MQ Telemetry Transport) [5] are IoT application protocols which use the Transmission Control Protocol (TCP) to offers the data transmission between different devices. Thus, the role of TCP is very important in the growth of IoT.

There are many types of popular TCP variants [6, 9, 22] available today to provide various types of congestion control mechanism. All the TCP variance deals with different TCP problems like efficiency, fairness, RTT fairness, stability, and reliability, and all the TCP variants achieved their respective objective. Utilization of the available bandwidth of the path with respect to the received ACK remains a challenge in the large delay network. Additionally, as number of devices increase over the internet, the network congestion also increases. Thus, this paper presents a new approach of congestion control to handle the large round trip delay and provide better bandwidth utilization as compared to other TCP variants.

The rest of the paper organized as follows: Section 2 presents literature reviews of various TCP variants while section 3 presents a new delay-based congestion control algorithm. The performance evaluation of the proposed algorithm is presented in section 4 while section 5 concludes the overall performance of proposed method.

2. RELATED WORK

A congestion control algorithm for TCP is proposed by [6] known as TCP-Tahoe introduces Slow Start, Congestion Avoidance, and Fast Retransmission technique. But TCP-

Tahoe reduces the congestion window to one when packet loss is detected and lead to significant throughput degradation. This serious problem is identified by [2] and revised the original Slow Start and Congestion Avoidance method by introducing the major congestion event and minor congestion event. A major congestion event is identified by Retransmission Timeout (RTO) and a minor congestion event is identified by three duplicate acknowledgements. When a TCP sender receives three duplicate acknowledgements, it concludes that the packet is lost. So, the TCP sender retransmits the lost packet. Due to retransmission of the packet, the TCP sender assumes that there is congestion in the network. So, the TCP sender modifies the Cwnd and Ssthresh values by the half of the current CWND. For optimization of Fast Retransmission method of TCP-Tahoe, TCP-Reno uses Fast Recovery method. In this method, TCP sender does not exit Fast Recovery mode until it receives a non-duplicate acknowledgment. TCP Reno improves the performance of TCP when minor congestion detected.

TCP Reno differentiates the major and minor congestion event in the network but it cannot identify that which type of congestion event occurs when multiple packet loss occurs as a single congestion event. TCP Reno reacts on such type of congestion event as a heavily loaded network condition and reduces the cwnd according to a number of packet drops. It reduces the performance of TCP Reno significantly.

Floyd et al. [7, 8] identified this problem of TCP Reno and proposed its revised version known as TCP New Reno. In this TCP, Fast Recovery algorithm has been modified to overcome this problem. It resolves this problem by restricting the current Fast Recovery phase cannot change until all the acknowledgement receives corresponding to current CWND. For this purpose, TCP New Reno uses a new state variable that records the sequence number of the last data packet sent before starting the Fast Recovery phase. This state variable helps TCP New Reno to identify the partial acknowledged and new acknowledged data packet. After receiving a new acknowledgement, TCP concludes that all data packets are reaching to destination successfully. Now, its exit from the Fast Recovery and start sending data packet according to Congestion Avoidance phase. Recording the sequence number before entering into Fast recovery phase is the solution of unwanted CWND reduction. But in some cases, when the timeout occurred during the Fast Recovery phase, then an unnecessary CWND reduction may still occur. A solution of the problem is that remember the highest sequence number send after each timeout and discard all the duplicate acknowledgement that is lower sequence number than the highest sequence number.

So, TCP New Reno resolves the problem of multiple packet loss as a single congestion event and improves the performance. It also maintains fairness of the flow as TCP Reno.

Another solution to multiple loss is proposed by Mathis et al. [9] is called TCP SACK. This protocol provides the ability for the receiver to report the number of successfully delivered data packets. By using this information TCP sender can calculate a block of the lost packet (gap in sequence number of the acknowledgement) and retransmit it quickly.

Mathis and J. Mahdavi [10] proposed another technique based on SACK with new congestion control mechanism. FACK (forward acknowledgement) maintain three state variables, H-highest sequence numbers, F-forwarded most sequence number, and R- the number of retransmitted packets. A relation of H-F+R can be utilized by the sender decides to either send new data or not.

Brakmo and Peterson [11] proposed a new congestion avoidance technique called TCP Vegas based on TCP Reno. The main logic of the Vegas congestion control algorithm is for estimation of the used buffer size of the bottleneck link of the path by measuring RTT. TCP Vegas compute the difference of the expected flow rate and Actual flow rate and adjust the congestion window accordingly. When TCP Vegas estimate the delay of the network path, TCP Vegas use RTTmin. Hasegawa et al. [12] recognized this serious problem of TCP Vegas and proposed another version of TCP is called Vegas+. It assumes initially Vegas friendly environment and applies bottleneck buffer size estimation to control the congestion window. When Vegas+ detect an unfriendly environment, it supports Reno algorithm.

Floyd [13, 14] proposed High-Speed TCP (HS-TCP) for high-speed network. HS-TCP replace standard New Reno increasing coefficient in Congestion Avoidance and decrease factor detect minor loss during the Fast Recovery phase. HS-TCP has the problem of fairness of RTT with a different flow. Kelly [15] proposed a Scalable TCP (STCP) as a replacement of HS-TCP to solve the effectiveness problem in high-speed long delay network. STCP use MIMD concept to increase and decrease the congestion window. STCP experience some critical problem like enter-fairness and constant congestion. Leith et al. [16, 17] proposed another congestion control algorithm HTCP which remove the enter-fairness problem of STCP and HS-TCP. The main idea of HTCP is that congestion window increases in n steps in Congestion Avoidance phase.

Caini and Firrincieli [18] proposed another congestion control algorithm called TCP Hybal. This technique resolves RTT unfairness problem by introducing a modification of New Reno's Slow Start and Congestion Avoidance phase. A scaling factor is calculated by relation RTT/RTTref where RTTref is the reference RTT has value 25ms. It provides better results in terms of RTT friendliness, but it increases the aggressiveness of the flow. Jin et al. [19, 20] introduce FAST TCP that provide time-based congestion window update base on the delay of the network. FAST-TCP defines a fixed rate congestion window update. It provides a better result in terms of inter-fairness, RTT fairness, stability, and scalability. Baiocchi et al. [21] proposed another congestion control algorithm called YeAH-TCP. It combines packet loss detection and measurement of RTT. It improves the inter-fairness and RTT fairness as compare to HS-TCP and STCP. Ha et al. [22] proposed a TCP-CUBIC for congestion control which is an enhanced version of BIC-TCP [23] TCP-CUBIC grows its window to the midpoint between the last maximum window size where the packet was lost and the last minimum window size; it did not lose any packet. TCP-CUBIC has two profiles concave and convex for window increase. It uses RTT-independent growth function that maintains scalability, RTT-fairness, and Intra-fairness, but it is not able to utilize available network resources and suffers from a greater number of packet losses. Wang et al. [24] plan to take the advantage of FAST-TCP [19-20] and TCP-FIT [25], and proposed a new congestion control algorithm called FAST-FIT. It uses the FAST-TCP growth function to maintain data flow and uses a TCP-FIT technique to adjust the CWND. It shows better results in terms of inter-protocol fairness (with TCP-Reno), high bandwidth utilization in wired and wireless environments.

Wang et al. [26] proposed Fair TCP that provides the initial congestion window value based on network status. It increases

the congestion window on the basis of RTT. It provides better results in terms of inter-protocol fairness, increases transmission efficiency and RTT fairness. Sharma and Kumar [27] suggested a new adaptive congestion control scheme in mobile ad-hoc networks to adjust the transmission rate of path. However, Verma and Kumar [28] suggested new adaptive data chunk scheduling policy for concurrent multipath transfer (CMT-SCTP).

During the analysis of these protocols, the authors identified that TCP suffer from heavy packet loss in large delay network. As a result, such variants [22, 25] suffer from unnecessary timeout and CWND reduction. Such variants [22, 25] are not able to utilize available bandwidth and also suffer from inter-protocol fairness. Additionally, as numbers of device increases over the Internet, network congestion also increases [29, 30]. Therefore, we need a congestion control policy which manage the traffic according to availability of the bandwidth.

3. PROPOSED WORK

The path delay variation is an important factor in TCP which reflect the correct network status. Let Pi be the paths used for data transmissions and the round trip delay of path is defined as Di. If delay Di of path Pi changes, it means that bottleneck queue size of path Pi also changes. It means that path traffic intensity also changes. Therefore, this paper introduces a new delay-based TCP variant, which uses RTT variation as congestion detection factor. The proposed technique is the extension of TCP New Reno in terms of delay based TCP. It uses all the policies of TCP New Reno with some small change in fast recovery phase. Apart from TCP New Reno policies, it has a new congestion detection method.

The RTT is in important factor used by all the TCP variants to analyze the delay. It includes queuing delay, transmission delay, and propagation delay.

$$RTT_{min} = P_d + T_d + Q_{min} + P_r \quad (1)$$

$$RTT_i = P_d + T_d + Q_d + P_r \quad (2)$$

where, RTTi is current RTT, RTTmin is a minimum RTT, Pd is propagation delay, Td is a transmission delay, Pr is a procession delay, Qd is a current queuing delay of the path, and Qmin is a minimum queuing delay.

The propagation delay, processing delay and transmission delay remain almost same for all conditions, but the queuing delay change when the queue size changes. When, queuing delay changes RTT also changes correspondingly. If the authors adjust the transmission rate of the TCP with respect to RTT change, then it will quickly adopt the network condition and improve the available bandwidth utilization and reduce the number of packet loss. Therefore, the authors are introducing a threshold calculation method for RTT, which provide information about when TCP change the transmission rate.

RTTmax is the maximum round trip time, which is estimated by TCP source. RTTmax is estimated when a packet drop occurred. It means that when the bottleneck queue is full, then it drops the packet (maximum packet gets dropped due to queue over flow). When packet drops from bottleneck queue, at that time queue is full and queuing delay is maximum.

$$RTT_{max} = P_d + T_d + Q_{d\_max} + P_r \quad (3)$$

where, Qd_max is maximum queuing delay. Thus, the threshold for RTT can be calculated based on RTTmax, RTTmin, and RTTi with a scaling factor of τ. Value of τ is determined by performing a number of experiments and set it to 2.5. The threshold for RTT is calculated on each received ACK at TCP source as:

$$RTT_{thresh} = \frac{RTT_{max}+RTT_{min}+RTT_i}{\tau} \quad (4)$$

where, RTTthresh is a threshold for RTT and τ is scaling factor. Now, proposed method adapts path traffic as well as delay variation based on RTTthresh threshold estimated by TCP source. The proposed method may not change the traffic rate when current RTT is greater than RTTthresh. However, it increases the CWND when current RTT is less than RTTthresh. The authors are introducing the functioning of the proposed TCP as:

Initially, TCP starts with TCP New Reno policy like slow start and congestion avoidance. For each receive ACK, TCP calculate the RTTmax, RTTmin, and RTTi. After calculation of the TCP factors, it compares RTTi with RTTthresh. If RTTi is greater than the RTTthresh, then stop increasing cwnd. Else, if RTTi less than RTTthresh, then cwnd increase according to following equation.

$$\partial = \frac{ssthresh}{1+e^{cwnd}} \quad (5)$$

$$cwnd_{i+1} = cwnd_i + \partial + \frac{1}{cwnd_i} \quad (6)$$

This approach worked in congestion avoidance phase only.

Algorithm 1: Delay-based congestion control

1. **For each ACK**
2. **Begin**
3. **If** $ACK\ received$
4. $RTT_i = Now - Packet\_send\_time$
5. **If** $RTT_i < RTTmin$
6. $RTT_{min} = RTT_i$
7. **End If**
8. **If** $RTT_i > RTT_{max}$
9. $RTT_{max} = RTT_i$
10. **End If**
11. $RTT_{thresh} = \frac{RTT_{max}+RTT_i+RTT_{min}}{\tau}$
12. **If** $cwnd_i > ssthresh$
13. **If** $RTT_i > RTT_{thresh}$
14. $cwnd_{i+1} = cwnd_i$
15. **Else If** $RTT_i < RTT_{thresh}$
16. $\partial = \frac{ssthresh}{1+e^{cwnd}}$
17. $cwnd_{i+1} = cwnd_i + \partial + \frac{1}{cwnd_i}$
18. **Else** $cwnd_i \leq ssthresh$
19. $New\ Reno\ Slow\ start$
20. **End If**
21. **End If**
22. **End If**
23. **End**

The advantage of proposed congestion control approach is that the proposed approach adapts the network conditions (congestion or congestion free) quickly and transmit data accordingly. Therefore, proposed approach achieves better performance as compared to other TCP variants. However, if

delay variation of the path changes without congestion then this may cause wrong delay estimation and interpretation. But, in most of the cases delay of the path varies due to congestion. Therefore, such condition is very rear and it may not arise when everything is normal except traffic rate.

4. PERFORMANCE ANALYSIS

This section describes the experimental setup and results obtained for proposed algorithm using NS2 (Network Simulator2) [31]. The simulation topology (dabbled topology) has 8 nodes as shown in Figure 1. The nodes S1, S2 and S3 have been configured to act as sources for TCP or UDP based applications. D1, D2 and D3 are the destinations for aforementioned sources. The network has seven links with different characteristics. Link S1-R1, S2-R1, S3-R2, R1-D3, R2-D1 and R2-D2 are configured to have 10Mbps bandwidth and link R1-R2 has 1.5Mbps. The queue size of each link is kept 50. Propagation delay of the link varies according to simulation requirement. The performance of the proposed TCP is compared with TCP Reni, New Reno, TCP Cubic and TCP Fast-Fit.

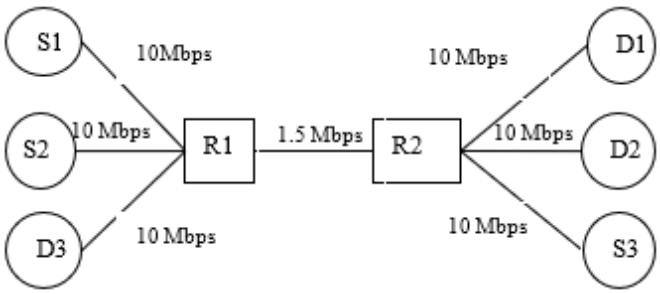

Figure 1. Simulation topology

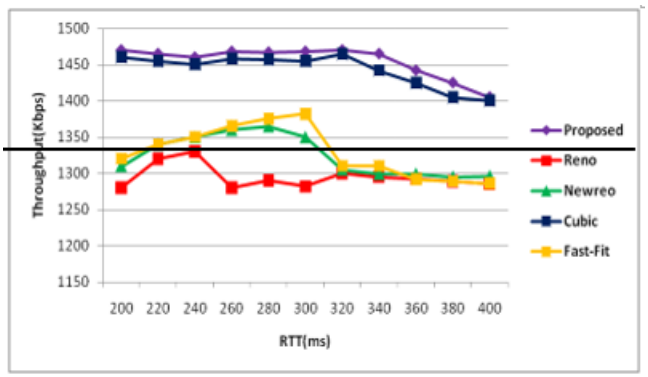

Figure 2. RTT Vs throughput

The authors first analyze the performance of the proposed TCP in variable RTT conditions. During the simulation, queue size of the bottleneck is kept 50 packets, propagation delay varies from 100ms to 200ms, source S1 use CBR traffic on the top of UDP, source S2 use FTP traffic on the top of TCP and the source S3 use CBR traffic in the reverse direction from S1 and S2. Figure 2 shows the throughput variation of TCP variants with respect to RTT. It shows that as the RTT increases, throughput of TCP variants shows variable trends. TCP Reno and TCP Newreno achieves lower throughput as compared to other TCP variants. However, TCP Fast-fit and TCP Cubic show better throughput as compared to TCP Reno and TCP Newreno. Meanwhile, proposed method achieves better throughput as compared to other TCP variants. The proposed method achieves better throughput due to it delay-based transmission rate adaptation policy. As a result, proposed method suffers from a smaller number of packet losses, which directly affect the available bandwidth utilization. Hence, proposed TCP achieves higher throughput than other TCP variants in all RTT variation.

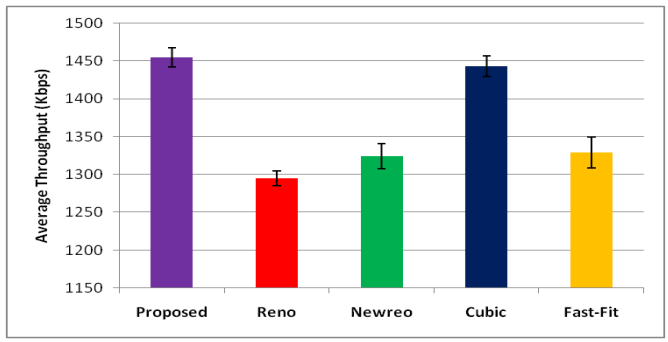

Figure 3. Average throughput with confidence interval

The authors also calculate the average throughput improvement and confidence interval of proposed method, Cubic, Fast-Fit, Newreno and TCP Reno. Figure 3 shows the average throughput and confidence interval (using error bar). It shows that proposed method has better average throughput improvement as compared to other TCP variants. Meanwhile, confidence interval of all TCP variants also confirms that proposed method has better confidence that throughput lies between 1443-1467 Kbps, while other TCP variants have less confidence interval for throughput as compared to proposed method.

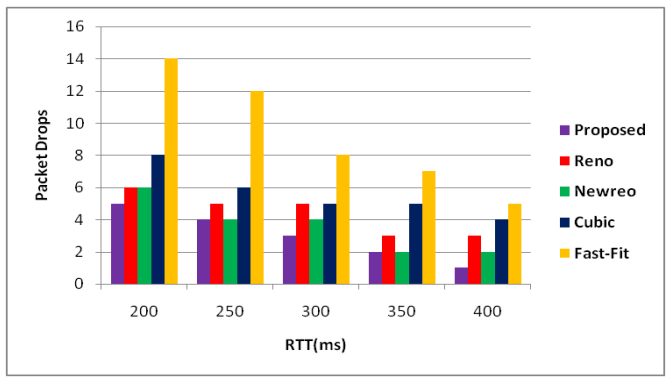

Figure 4. RTT Vs packet drops

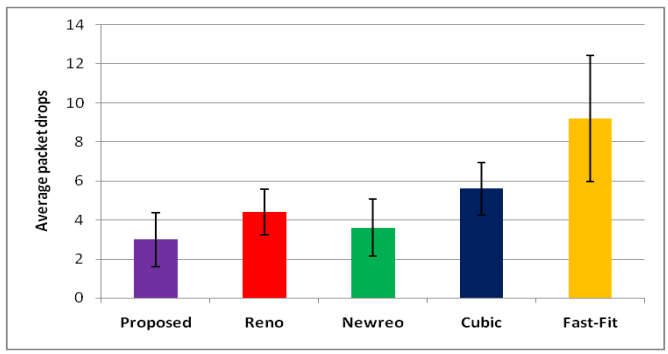

Figure 5. Average packet drops with confidence interval

Now, Figure 4 shows the number of packets drops in variable RTT conditions. It shows that as RTT increases number of packet drops decreases. Figure 4 shows that Fast-Fit has highest number of packets drops in all conditions while proposed TCP has the less number of packet drops as compared to TCP Cubic and Fast-fit, TCP Reno, and TCP Newreno. Figure 5 shows the average packet drops with confidence interval (using error bar). It also confirms that proposed method has less average packet loss as compared to other TCP variants. While, confidence interval also shows that proposed method has better confidence that the packet loss lies between 2-5 while other variant has large confidence that proposed method.

Now, another simulation has been performed to analyze the performance of proposed TCP in different background traffic environment. In this simulation setup, queue size of the bottleneck is 50, propagation delay is 50ms, and simulation time is 150sec. Figure 6 shows the throughput variation of TCP variants with different background traffic. It shows that as traffic rate increases, throughput of all TCP variants decreases. TCP Reno shows the least utilization for the entire traffic rate while proposed TCP achieves better throughput as compared to other TCP variants. Average throughput and confidence interval of all TCP variants are also shown in Figure 7. This figure shows that proposed method has better average throughput as compared to other TCP variants while confidence interval of proposed method also confirms that it has better confidence interval lies between 930-1330 Kbps.

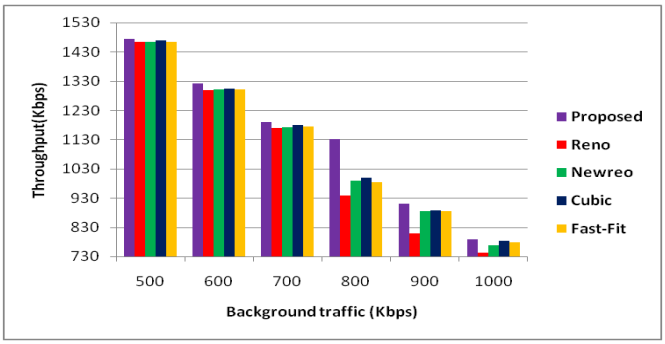

Figure 6. Background traffic Vs throughput

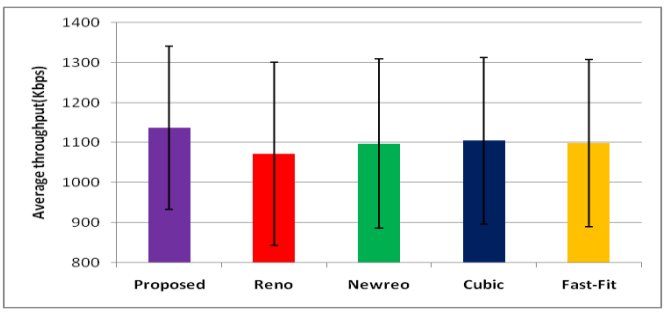

Figure 7. Average throughput with confidence interval

Figure 8 shows the number of packets drops while background traffic varies from 500-1000Kbps.It shows that as traffic rate increases packet drop also increases. Figure 8 shows that Fast-Fit suffers from highest number of packets drops while proposed method shows least number of packets drops as compared to other TCP variants. Figure 9 shows the average packet drops and confidence interval of all TCP variants. It also confirms that proposed method achieves less numbers of packet drops and has better confidence interval compared to TCP Reno, TCP Newreno, Cubic, and Fast-Fit.

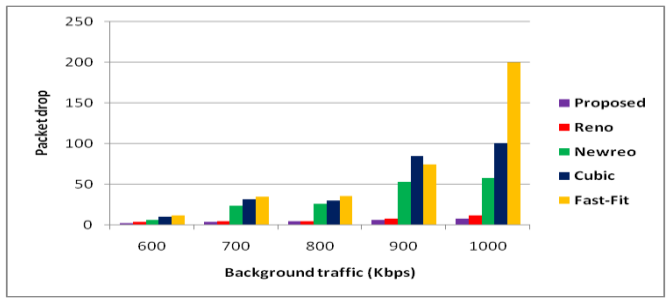

Figure 8. Background traffic Vs Packet drops

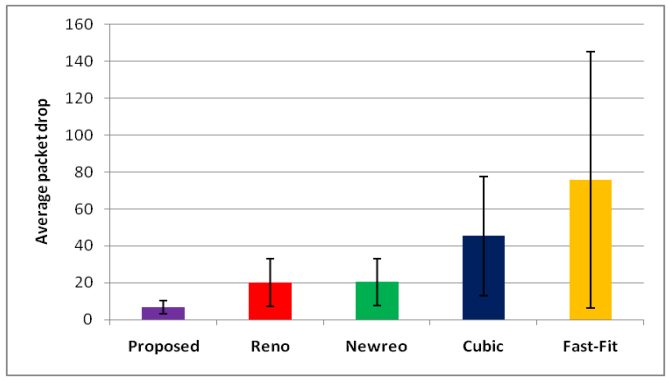

Figure 9. Average packet drops with confidence interval

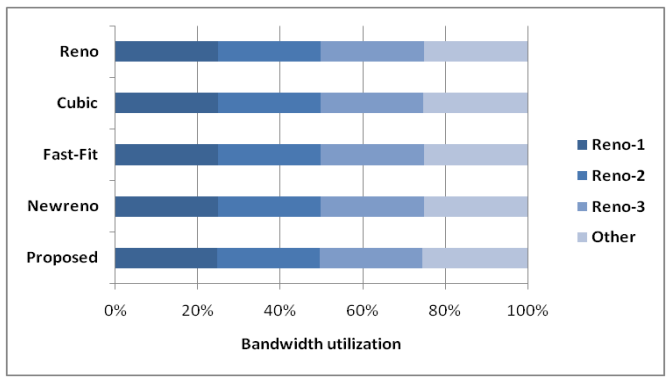

Figure 10. Bandwidth utilization among the competing traffic

The inter-protocol fairness is another important issue with TCP. Therefore, another simulation has been performed to analyze inter-protocol fairness of proposed TCP. In this simulation setup, queue size of the bottleneck is 50, propagation delay is 50ms, and simulation time is 150sec. Figure 10 shows the fairness property of TCP with three TCP Reno flow and one other flow (Proposed TCP, TCP Reno, TCP Newreno, TCP Cubic and Fast-Fit) in terms of available bandwidth utilization. In this figure, first 3 different blocks show the TCP Reno flow share and last block shows the other TCP variants (Proposed TCP, TCP Reno, TCP Newreno, TCP Cubic and Fast-Fit). It shows that the proposed TCP share all most equal bandwidth with each flow, and has better bandwidth utilization than other TCP variants. It concludes that the proposed method has similar inter-protocol fairness lick other TCP variants.

5. CONCLUSION

This paper presents new delay-based TCP variants which uses RTT as a congestion detection factor. It uses the RTT variation as an indicator of congestion. The proposed TCP adapts the network traffic condition quickly to adjust the transmission rate. As a result, it achieves better performance. The simulation results show that the proposed TCP reduces the number of packet drops, improved the bandwidth utilization in variable RTT conditions as well as variable background traffic conditions. It also shows better inter-protocol fairness with standard TCP Reno. In future, this proposed congestion control approach can further extend to test the performance with available IoT application protocols.

REFERENCES

[1] Postel, J. (1981). RFC793-Transmission Control Protocol. Internet Engineering Task Force. From https://www.ietf.org/rfc/rfc793.txt.

[2] Allman, M., Paxson, V., Stevens, W. (1999). RFC2581-TCP congestion control. Internet Engineering Task Force. From https://tools.ietf.org/html/rfc2581.

[3] Saint-Andre, P. (2011) RFC6120-Extensible Messaging and Presence Protocol (XMPP): Core, Internet Engineering Task Force. From https://tools.ietf.org/html/rfc6120.

[4] Fielding, R.T. (2000) Architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California, Irvine.

[5] Banks, A., Gupta, R. (2015) MQTT (MQ Telemetry Transport) http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os.html.

[6] Jacobson, V. (1988). Congestion avoidance and control. ACM SIGCOMM, 18(4): 314–329. https://doi.org/10.1145/52325.52356

[7] Floyd, S., Henderson, T. (1999). RFC2582-the NewReno modification to TCP's fast recovery algorithm. Internet Engineering Task Force. From https://tools.ietf.org/html/rfc2582.

[8] Floyd, S., Henderson, T., Gurtov, A. (2004). RFC3782—the NewReno modification to TCP's fast recovery algorithm. Internet Engineering Task Force. From http://tools.ietf.org/html/rfc3782.

[9] Mathis, M., Mahdavi, J., Floyd, S., Romanov, A. (1996). RFC2018—TCP selective acknowledgment options. Internet Engineering Task Force. From https://tools.ietf.org/html/rfc2018.

[10] Mathis, M., Mahdavi, J. (1996). Forward acknowledgement: Refining TCP congestion control. SIGCOMM Computer Communications Review, 26(4): 281-291. https://doi.org/10.1145/248157.248181

[11] Brakmo, L., Peterson, L. (1995). TCP Vegas: end to end congestion avoidance on a global Internet. IEEE J. Sel. Areas Communication, 13(8): 1465-1480. https://doi.org/10.1109/49.464716

[12] Hasegawa, G., Kurata, K., Murata, M. (2000). Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet. In Proc. IEEE ICNP, pp. 177-186. https://doi.org/10.1109/ICNP.2000.896302

[13] Floyd, S. (2003). RFC3649—HighSpeed TCP for large congestion windows. Internet Engineering Task Force. From https://tools.ietf.org/html/rfc3649.

[14] Floyd, S. (2003). HighSpeed TCP and Quick-Start for Fast Long-Distance networks (slides). TSVWG, IETF.

[15] Kelly, T. (2003). Scalable TCP: Improving performance in highspeed wide area networks. Computer Communications Review, 32(2). https://doi.org/10.1145/956981.956989

[16] Leith, D., Shorten, R. (2004). H-TCP: TCP for high-speed and long-distance networks. In Proc. of PFLDnet.

[17] Leith, D. (2008). H-TCP: TCP congestion control for high bandwidth-delay product paths. IETF Internet Draft, From http://tools.ietf.org/html/draf-tleith-tcp-htcp-06.

[18] Caini, C., Firrincieli, R. (2004). TCP Hybla: A TCP enhancement for heterogeneous networks. International J. Satellite Communications and Networking, 22: 547-566. https://doi.org/10.1002/sat.799

[19] Jin, C., Wei, D., Low, S., Buhrmaster, G., Bunn, J., Choe, D., Cottrel, R., Doyle, J., Feng, W., Martin, O., Newman, H., Paganini, F., Ravot, S., Singh, S. (2005). FAST TCP: from theory to experiments. IEEE Network, 19(1): 4-11. https://doi.org/10.1109/MNET.2005.1383434

[20] Wei, D.X., Jin, C., Low, S.H., Hegde. (2006). FAST TCP: Motivation, architecture, algorithms, performance. IEEE/ACM Transaction Networking, 14(6): 1246-1259. https://doi.org/10.1109/TNET.2006.886335

[21] Baiocchi, A., Castellani, A.P., Vacirca, F. (2007). YeAH-TCP: Yet another highspeed TCP. In Proc. PFLDnet, ISI, Marina Del Rey (Los Angeles), California, February 2007.

[22] Ha, S., Rhee, I., Xu, L. (2008). CUBIC: A new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating Systems Rev., 42(5): 64-74. https://doi.org/10.1145/1400097.1400105

[23] Xu, L., Harfoush, K., Rhee, I. (2004). Binary increase congestion control for fast long distance networks. In Proc. IEEE INFOCOM, 4: 2514–2524. https://doi.org/10.1109/INFCOM.2004.1354672

[24] Wang, J., Wen, J., Han, Y., Zhang, J., Li, C., Xiong, Z. (2014). Achieving high throughput and TCP Reno fairness in delay-based TCP over large network. Frontiers of Computer Science, 8(3): 426-439. https://doi.org/10.1007/s11704-014-3443-9

[25] Wang, J., Wen, J., Zhang, J., Han, Y. (2010). CP-FIT: An improved TCP congestion control algorithm and its performance. In Proc. of IEEE International Conference on Computer Communication, pp. 2133-2137. https://doi.org/10.1109/INFCOM.2011.5935128

[26] Wang, G., Ren, Y., Li, J. (2014). An effective approach to alleviating the challenges of transmission control protocol. IET Communication, 8(6): 860-869. https://doi.org/10.1049/iet-com.2013.0154

[27] Sharma, V.K., Kumar, M. (2017). Adaptive congestion control scheme in mobile ad-hoc networks. Peer-to-Peer Networking and Applications, 10(3): 633-657. https://doi.org/10.1007/s12083-016-0507-7

[28] Verma, L.P., Kumar, M. (2017). An adaptive data chunk scheduling for concurrent multipath transfer. Computer Standards & Interfaces, 52: 97-104. https://doi.org/10.1016/j.csi.2017.02.001

[29] Mishra, N., Verma, L.P., Srivastava, P.K., Gupta, A. (2018). An analysis of IOT congestion control policies.

Procedia Computer Science, 132: 444-450. https://doi.org/10.1016/j.procs.2018.05.158
[30] Arunmozhi, S, Nagarajan, G. (2018). Quadrature spatial modulation on full duplex and half duplex relaying network. Modelling, Measurement and Control A. 91(4):168-174.
[31] NS Project. (2011). The Network Simulator: ns-2. http://www.isi.edu/nsnam/ns, accessed on June 2017.