Characterizing End-to-End Delay Performance of Randomized TCP Using an Analytical Model

Mohammad Shorfuzzaman and Mehedi Masud
Computer Science Department
Taif University, Taif, Saudi Arabia

Md. Mahfuzur Rahman
Dept. of Computer Science and Information Technology
Hajee Mohammad Danesh Science and Technology University
Dhaka, Bangladesh

***Abstract*—TCP (Transmission Control Protocol) is the main transport protocol used in high speed network. In the OSI Model, TCP exists in the Transport Layer and it serves as a connection-oriented protocol which performs handshaking to create a connection. In addition, TCP provides end-to-end reliability. There are different standard variants of TCP (e.g. TCP Reno, TCP NewReno etc.)which implement mechanisms to dynamically control the size of congestion window but they do not have any control on the sending time of successive packets. TCP pacing introduces the concept of controlling the packet sending time at TCP sources to reduce packet loss in a bursty traffic network. Randomized TCP is a new TCP pacing scheme which has shown better performance (considering throughput, fairness) over other TCP variants in bursty networks. The end-to-end delay of Randomized TCP is a very important performance measure which has not yet been addressed. In the current high speed networks, it is increasingly important to have mechanisms that keep end-to-end to delay within an acceptable range. In this paper, we present the performance evaluation of end-to-end delay of Randomized TCP. To this end, we have used an analytical and a simulation model to characterize the end-to-end delay performance of Randomized TCP.**

***Keywords*—Randomized TCP, end to end delay, congestion window, TCP pacing, propagation delay, Markov chain.**

I. Introduction

Transmission Control Protocol (TCP) plays a very important role in data transmission over network. TCP offers a connection oriented delivery service to the end user applications and it provides reliable data flows between two processes running on two end systems. TCP implements data retransmission mechanism for the packets that are lost during transmission and thus ensures guaranteed data transfer between the sender and the receiver. TCP is intelligent enough to understand the loss of packets that are already sent by implementing *timeout* timer at the sender and by using duplicate Acknowledgements messages. TCP implements slow start and congestion avoidance phases to handle the congestion of the network [1]. However, it cannot fully avoid the packet losses which can greatly degrade the throughput quality of the transmission system. This problem becomes very severe when the network is bursty. The network may have bursty traffic very often due to buffer overflow and the limited buffer size of intermediate routers. Researchers found that if the router buffer size can be made equal to the product of link bandwidth and average Round Trip Time (RTT) of flows passing through the router, then the packet loss ratio will be reduced and this will improve the end-to-end delay performance [2]. But this large amount of memory implementation will not be feasible and costly as well. Research is also going on to modify some aspects of TCP algorithm addressing this problem. TCP pacing is proposed in which successive data packets are transmitted with some time intervals in between which can avoid sending packets in bursts. TCP pacing uses the last Round Trip Time (RTT) to adjust not only its next congestion window size (how much) but also the time (when) of sending. The congestion window is the amount of data the sender is allowed to send to the network at a time. Randomized TCP improves the paced TCP to achieve better result. In [3], Chandrayana et al. has proposed Randomized TCP which randomizes the packet sending time at sender. Randomized TCP is very similar to paced TCP but in paced TCP packet sending time at TCP sources are equally spaced for all flows where in Randomized TCP the packet sending times are scheduled at different intervals or randomly for the TCP flows. Randomized TCP solves the phase effect and biasness problems which are still present in TCP pacing. Randomized TCP performs better than paced TCP in improving network throughput, fairness, timeouts and losses. But all these proposals do not address the end-to-end delay performance achieved by Randomized TCP [3]. To this end, this paper evaluates the performance of Randomized TCP in end-to-end delay. It is highly desired that the Randomized TCP shows better end-to-end delay performance than paced TCP. The evaluation shows both network and application layer performance of Randomized TCP.

The remainder of the paper is organized as follows. We have discussed the necessary background in Section 2. Related work relevant to the proposed research and the research problem are presented in Section 3. Simulation and analytical modeling are presented in Section 4 and Section 5 respectively. Finally, Section 6 concludes the paper with some future work.

II. Background

A. Congestion Window

Each sending endpoint of a TCP connection possesses a buffer for storing data which will be transmitted over the network. On the other hand, receiver side buffering helps the application to read the data only when it is ready. This also lets network transfers take place while applications are busy with other processing, improving overall performance. TCP uses "flow control" to avoid overflowing the receiver side buffer. In this case, TCP sends a fixed number of packets at a time and the size is usually known 'congestion window'. This congestion window (cwnd) contains the amount

of data that may be transmitted from the sender buffer. TCP uses "congestion control" mechanism to dynamically control congestion window size. In TCP, the window size depends on the congestion of the network. If there is any packet loss due to congestion in the network then the window size becomes very small otherwise it increases for each successful packet transmission. This congestion control window have a great impact in the performance of network. In TCP pacing and Randomized TCP, the sender delays the sending of packets in addition to the changing of congestion window when it identifies any congestion in the network.

B. TCP Algorithms

TCP is a very dynamic and reliable congestion control protocol. It uses acknowledgments and the TCP acknowledgments created by the destination are returned to the source. TCP acknowledgments help the sender to know whether the packets are well received or not. In TCP transmission, lost packets are interpreted as congestion signals. There are a number of standard variants of the TCP protocol, such as Tahoe, Reno, New Reno, Sack and Vegas. One of the main differences between these TCP versions lies in their methods of recovering from packet loss due to network congestion. In general, they differ in the sender side algorithms. FAST TCP is a new high speed TCP protocol which uses the experienced queuing delay of the packets to adjust the congestion window size. In our work, we have used TCP NewReno as a general variant of TCP to compare the end-to-end delay performance of Randomized TCP. A detailed account of some of these protocols is given below.

While TCP Reno produces less bursty traffic than TCP Tahoe, it is much less robust towards phase effects. The latter refers to unpredictability in performance resulting from very small differences in the relative timings of packet arrivals for different connections sharing a link. Both versions of TCP appear to have significant drawbacks as a means of providing data services over multimedia networks, because random loss resulting from fluctuations in real-time traffic can lead to significant throughput deterioration in the high bandwidth-delay product. The performance is degraded when the product of the loss probability and the square of the bandwidth delay product is large.

For high bandwidth-delay products, TCP is hideously inequitable towards connections with higher propagation delays: for multiple connections sharing a bottleneck link, the throughput of a connection is inversely proportional to (a power of) its propagation delay. It is worth expounding that random loss causes performance decline in TCP because it does not permit the TCP window to reach high enough levels to allow superior link utilization. On the other hand, when the TCP window is already big and is causing congestion, random early drops of packets when the link buffer gets too filled can in fact improve performance and lessen phase effects [4].

Relatively earlier simulation studies of TCP-tahoe include [5], [6], [7]. Simulations for the simple multi-hop network considered in [6] showed the oscillations in window sizes and the inequality of TCP towards connections traversing a larger number of hops. In [5], the authors consider a number of TCP connections sharing a bottleneck link. There is no queueing of acknowledgements, and sources are assumed to have data to send at all times. As stated earlier, [7] considers the effect of two-way traffic.

The unfairness of TCP-tahoe against connections with large round-trip delays and against connections traversing a big number of congested gateways has also been demonstrated in other current studies of TCP-tahoe. The heuristic analysis shows that, for multiple connections sharing a bottleneck link, the throughput of a connection is inversely proportional to its round-trip time. Oscillatory behavior and inequality towards connections with superior propagation delays have also been noticed in an earlier analytical study of feedback-based congestion control which uses a continuous-time approximation to the dynamic behavior of a rate-based scheme.

Another adaptive window flow control scheme is proposed in [8], [9]. The proposed window adaptation mechanism operates in a high bandwidth-delay product, and is based on asymptotics derived from a queueing model of the network. It has the disadvantage of requiring more central synchronization than TCP: the adaptation algorithm for every link must be acquainted with the distinctiveness of the bottleneck link for that connection and the relative propagation delays of the other connections sharing that link. However, the adaptive mechanism itself is much smoother than the drastic window size changes in TCP, so that a decentralized adaptive scheme based on a similarly smooth mechanism may defeat the weaknesses in TCP while not requiring the type of acquaintance understood in [8], [9].

C. TCP pacing and Randomized TCP

TCP's congestion control mechanisms can not avoid bursty traffic flows on high-speed networks. Bustry Traffic produces higher queueing delays, more packet losses, lower throughput. TCP pacing is aimed at reducing the burstiness of TCP traffic and the impact of limited buffers in routers. In TCP pacing, the packet loss is less and the competing flows face less queueing delays. In addition to setting TCP's congestion window which tells about $how\ much\ to\ send$, TCP pacing also fixes the time $when\ to\ send$ the packets. In TCP pacing, the sender sends successive data packets with some equally spaced time intervals, Δ=$\frac{RTT}{cwnd}$. In this case, Round Trip Time (RTT) and cwnd are the key components to adjust the sending time of packets. Though TCP pacing reduces packet losses and queueing delays, this scheme is not efficient in addressing TCP's "phase effect" problem and it also has biasness against long flows. When there are a number of competing flows, phase effect causes a specific section of competing flows to experience recurrent drops. Randomized TCP is found efficient to address this problem and it also reduces the baisness against long flows. In Randomized TCP, the sender sends successive data packets with randomly spaced time intervals, Δ=$\frac{RTT(1+x)}{cwnd}$, where x follows a uniform distribution. Randomized TCP also shows more fairness over Paced TCP when multiplexed with other standard TCP variants (e.g. TCP New Reno) [3]. All the researches include traffic performance such as fairness, throughput of Randomized TCP. But more application-layer performance like end-to-end delay has not been addressed. We have evaluated the end-to-end delay performance of Randomized TCP in this paper. In a recent work, the authors in [10] propose a transport layer solution to

TCP called TCPRand. The idea is to have a randomization of TCP payload size, which breaks synchronized packet arrivals between flows from different input ports. The simulation results show that TCPRand ensures the improvement of TCP fairness with negligible overheads in all test cases. Prakah et al. [11] investigate the TCP outcast problem in data center applications with many-to-one traffic pattern scenario. The research results found out that with excessive traffic flows, drop-tail queueing may drop a series of consecutive packets at each input port having a port blackout. Wu et al. [12] focus on incast congestion control for TCP in data center networks using a similar traffic pattern scenario.

D. End-to-end delay

End to end delay can be defined as the time taken by a packet to reach the destination after it has started from the source. If the network is not busy then there will not be any queueing delay. But if the network has bursty traffic then the queueing delay also affects the end-to-end delay performance. Processing time of the routers, transmission delay also affect the end-to-end delay of packets. We have used an analytical model to characterize the end-to-end delay performance of Randomized TCP and also evaluate the results with simulation.

E. NS-2 Simulation Environment

NS-2 is a discrete event simulator which has mainly targeted at networking research. NS-2 provides substantial simulation support for wired and wireless (local and satellite) networks. NS-2 provides tools to create any desired networking environment and also to import any networking aspects (e.g. TCP, routing, and multicast etc.) for data transmission in that environment. NS (version-2) is developed under the VINT project as a joint effort by UC Berkeley, USC/ISI, LBL, and Xerox PARC. NS-2 is an object oriented simulator and was written in C++ with OTcl as a front-end. The simulator handles two class hierarchy: the C++ class hierarchy (compiled hierarchy), and the class hierarchy within the OTcl interpreter (interpreted hierarchy) [13]. We have used NS-2 network simulator for experiments.

III. Related Work

Many applications rely on TCP since TCP has its own congestion control mechanism and does not bother the network for that. In [14], self regulating TCP acknowledgement pacing scheme has been proposed. This work tells about the ACK pacing technique to reduce the data loss due to congestion in TCP. In TCP each packet needs to be acknowledged by the receiver. But in acknowledge pacing scheme, the acknowledges are delayed by the receiver. ACK pacing uses a matrics (network load dynamics) to make the receiver understand about the congestion and allows the receiver acts upon this situation very smartly. Thus, the authors [14] show the implementation issues and better performance results of self-regulating ACK pacing.

In [15], Rezdan et al. tell about TCP Westwood which uses Bandwidth Share Estimate (BSE) along with RTT to set the pacing interval for sender side TCP pacing. TCP Westwood keeps the bottleneck service rate in BSE parameter. There are two major phases of TCP: slow start phase and congestion avoidance phase. In this work, TCP pacing is only effective in the slow start phase. In [16], Garetto et al. present a way to analyze the bursty TCP traffic in wide area networks and present ways to characterize "TCP pacing". The authors use a simple analytical model to show the traffic (e.g. packet loss) produced by a large number of TCP connections. In [17], Chang et al. find out that paced TCP performs better channel reuse than TCP Reno.

In [18], ElRakabawy et al. introduce gateway adaptive pacing scheme which is mostly effective for the flows from wired sender to wireless receivers. The gateway device implements a pacing queue and it sends the queued data obtained from the wired sender with understanding of the current transmission rate of the wireless network. This approach includes some transport layer functionality to the IP layer in the Internet gateway. In our work, TCP NewReno is used as standard TCP protocol.

Enachescu et al. has found that buffer size of routers can be made very small if some performance of link layer can be sacrificed [19]. FDL (Fiber Delay Line) provides limited buffering capacity. In [20], TCP pacing is measured using RTT/cwnd which shows TCP pacing application performance in the FDL buffers. In this work, the authors find that TCP Pacing reduces packet loss-rate by decreasing the burstness of packet arrival and achieves higher throughput. [21] shows the effects of small buffer in standard TCP as well as TCP pacing. In our evaluation, It is found that TCP pacing is very useful in small buffer routers. In recent work, the authors in [22], [23] have focused on multipath TCP traffic which enables hosts to send data over multiple paths and has use cases on smartphones and datacenters. The research results confirm that multipath TCP operates successfully over the real Internet even with the middleboxes.

Now, we describe the problem definition for our current research. TCP does not bother network to handle the flow control and congestion control rather it is handled by the end systems. But inside the network the congestion mainly occurs due to buffer overflow at routers. When the network is congested, the TCP sender only adjusts the congestion window and as a result it cannot avoid the packet losses. So, it is very important to make the TCP sender to delay in sending packets when the network has bursty traffic. If the TCP sender delays in sending the packets, then it can reduce packet losses. TCP pacing has addressed this issue and Randomized TCP improves the limitations of paced TCP. In this paper, We have evaluated the end-to-end delay performance of Randomized TCP through simulation and analytical modeling.

IV. Simulation Modeling

A. Simulation Setup

We used NS-2 network simulator to simulate the network environment. As shown in Figure 1 [24], we considered the topology of Abilene Network (Internet2). In Abilene network, there are 11 nodes [node(0,1,2,...10)] and 14 links. There is a bottleneck link between node(2) and node(6). The channels between node(2) and node(6) have only 45Mbps bandwidth whereas all other channels possess 155Mbps bandwidth. We used the table I for assigning the propagation delay of each channel. In simulation, we used 'DropTail queues' and 'fixed

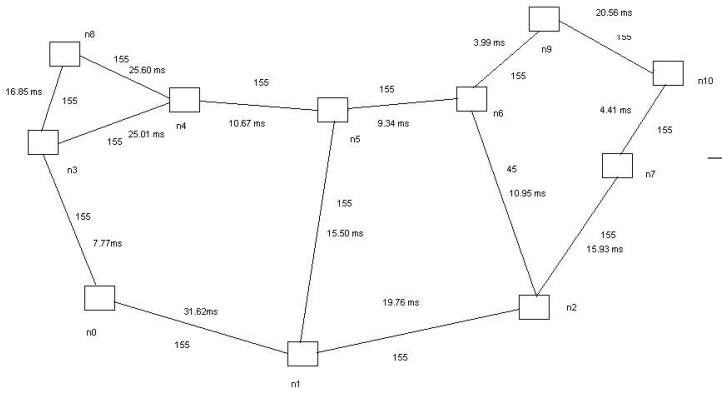

Fig. 1: Network Topology [24]

Source	Destination	Pro. Delay
n(0)	n(1)	31.624ms
n(0)	n(3)	7.772ms
n(1)	n(2)	19.756ms
n(1)	n(5)	15.504ms
n(2)	n(6)	10.950ms
n(2)	n(7)	15.938ms
n(3)	n(4)	25.010ms
n(3)	n(8)	16.852ms
n(4)	n(8)	25.608ms
n(4)	n(5)	10.674ms
n(5)	n(6)	9.340ms
n(6)	n(9)	3.990ms
n(7)	n(10)	4.412ms
n(9)	n(10)	20.464ms

TABLE I: Propagation Delays [24]

routing'. Drop Tail queue drops newly arriving packets until the queue finds enough room to accept incoming traffic. In fixed routing, there is a unique path for each source-destination pair.

In this simulation, nodal delays are neglected since processing delays incurred by packets are comparatively very less than propagation delays in channels. Using sample run, we characterized the traffic for the network to obtain expected utilization of each channel. NS-2 is an open source simulator and we modified the NS-2 source code (existing TCP implementation) to implement Randomized TCP. For this, we modified the 'delay' variable in send-much function (in tcp.cc) by introducing uniform randomization. TCP uses the 'delay' variable to measure the next packet sending time when the network is bursty. TCP fixes up the delay value as $\frac{RTT}{cwnd}$ but we introduced a uniform random variable with this to obtain the characteristics of Randomized TCP. So in case of Randomized TCP, the delay value is measured by $\frac{RTT(1+x)}{cwnd}$, where x follows a uniform distribution. TCP uses 't_rtt_' variable for saving RTT values and 'cwnd_' variable saving for current congestion window size. TCP NewReno

Queue Size	Randomized	NewReno
20 pkts	49.97 ms	51.51 ms
35 pkts	46.86 ms	48.47 ms
50 pkts	45.29 ms	46.76 ms
65 pkts	44.02 ms	44.90 ms
5000 pkts	43.49ms	45.79 ms

TABLE II: End-to-end Delays for different queue size

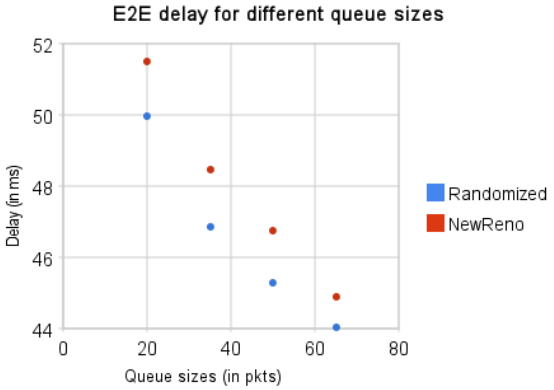

Fig. 2: End-to-end delays for different queue size

is an already developed module in NS-2. We used TCP NewReno to compare the performance of Randomized TCP. In all cases, delay is calculated as the difference between successful reaching time of each packet at destination and the starting time from the source. We have only considered the 'tcp packets' for calculating end-to-end delay, and the 'ack packets' and 'connection establishment packets' are not considered for delay calculation. We have varied the total number of flows in the network to control the burst level of the network.

B. Experiments and Results

First of all, we measured the end-to-end delays of Randomized TCP and TCP NewReno by varying the queue size of channels. The results shown in Table II tell that Randomized TCP's performance is peak over TCP NewReno when the queue size is very small. With the increase of queue sizes, the delay difference between Randomized TCP and TCP NewReno becomes narrow or small. In case of infinite buffer, Randomized TCP also shows good performance over TCP NewReno. In all these cases, we fixed up the total flow number as 90 and each packet size as 1000 Bytes.

In the next experiment, we fixed up the queue size as small (size as 35) and have varied the total number of flows to change the amount of burst in network. It is found that, for this small queue size and in high bursty traffic, Randomized TCP is far better than TCP NewReno. But in infinite buffer size (we used queue size as 5000) consideration, the burstiness cannot make significant delay difference between Randomized TCP and TCP NewReno. We depicted the results in Table III and in Table IV.

From the above simulation results, it is found that Randomized TCP is very good in considering end-to-end delay per-

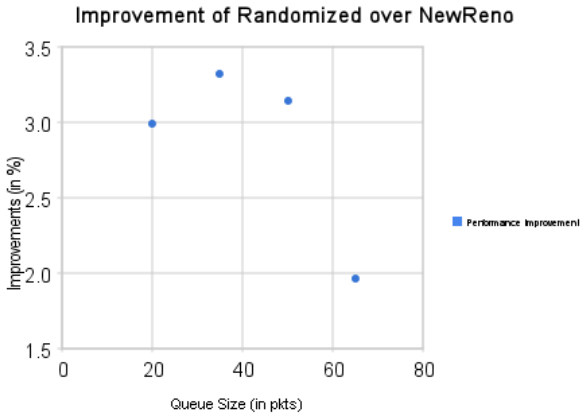

Fig. 3: Performance improvement in Randomized TCP

Total Flow	Randomized	NewReno
45	43.24 ms	43.34 ms
90	46.86 ms	48.47 ms
135	64.34 ms	107.62 ms

TABLE III: E2E Delays for different bursts (Queue size=35: as small size)

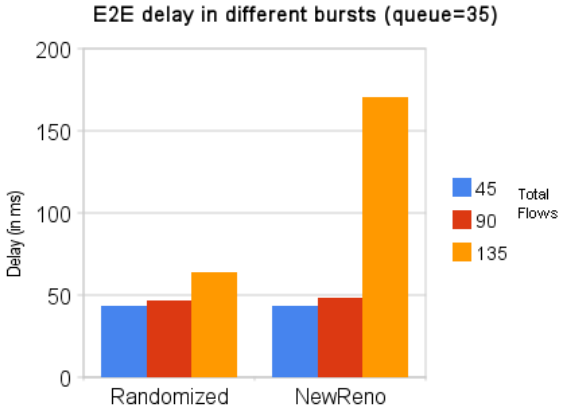

Fig. 4: E2E Delays for different bursts (Queue size=35)

Total Flow	Randomized	NewReno
45	39.78 ms	39.78 ms
90	43.49 ms	45.79 ms
135	57.28 ms	58.38 ms

TABLE IV: E2E Delays for different bursts (Queue size=5000: as infinite size)

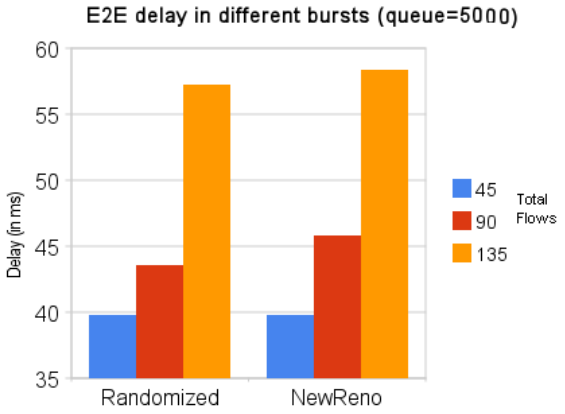

Fig. 5: E2E Delays for different bursts (Queue size=5000)

formance. About 3.25% delay improvements can be achieved for small queue size (Figure 3). Again in bursty network, Randomized TCP also shows better result over TCP NewReno when the channels have small buffer (Figure 4, 5).

V. Analytical Modeling

This analytical model is very similar to the queueing network model proposed by Lam et al. [25]. It is assumed that the routers in the network are indexed by i = 1, 2, 3, 4,...,M. Each router works on First Come First Serve (FCFS) basis and works at constant rate C_i bits per second. There may be more than one flow and the flows are indexed by k = 1, 2, 3,....., K. The first-order Markov Chain with transition probabilities for a particular flow k is modeled as $p^k{}_{ij}$ which is the routing probability to server j from server i and i, j = 1, 2, 3, 4,...., M.

It is assumed that the packets from k flows arrive at source node following poisson distribution with γ_k (packets per second), where k = 1, 2, 3, 4,....., K. The total external arrival rate to the network,

$$\gamma = \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 + ... + \gamma_K. \quad (1)$$

For a particular flow k, if γ_k and $p^k{}_{ij}$ are given then at router i the arrival rate of packets (due to a particular flow k), λ_{ik} is identified as

$$\lambda_{ik} = \gamma_k \delta_{ik} + \sum_{j=1}^{M} \lambda_{jk} p^k{}_{ji}, \quad (2)$$

where δ_{ik} is 1 when i is source node of flow k, otherwise δ_{ik} remains zero. So, the total arrival rate at router i (due to all flows),

$$\lambda_i = \sum_{k=1}^{K} \lambda_{ik}. \quad (3)$$

Again, the traffic intensity at router i for a particular flow k,

Total Flow	Analytical	Simulation
45	33.35 ms	39.78 ms
90	51.86 ms	45.59 ms
135	198.48 ms	57.28 ms

TABLE V: Simulation and Analytical Results

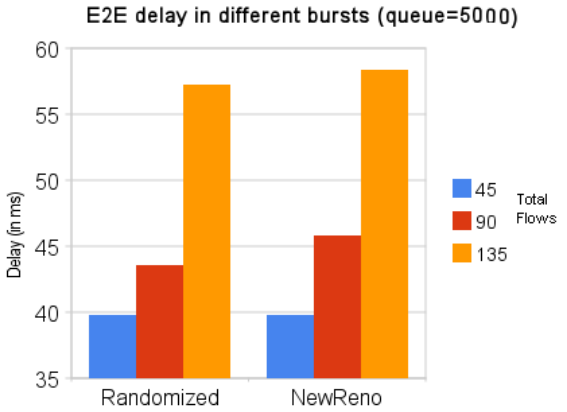

Fig. 6: Analytical vs Simulation Results

$$\rho_{ik} = \frac{\lambda_{ik}}{\mu C_i}, \tag{4}$$

where μ is used as mean in poisson distribution. The total traffic intensity at router i for all flows,

$$\rho_i = \sum_{k=1}^{K} \rho_{ik}. \tag{5}$$

This ρ_i can be used to find out the mean number of packets in transit within the network,

$$\gamma T = \sum_{i=1}^{M} \frac{\rho_i}{1-\rho_i} = \sum_{i=1}^{M} \frac{\lambda_i}{\mu C_i - \lambda_i}, \rho_{ik} = \frac{\lambda_{ik}}{\mu C_i}. \tag{6}$$

So, the mean end-to-end delay,

$$T = \frac{1}{\gamma} \sum_{i=1}^{M} \frac{\lambda_i}{\mu C_i - \lambda_i}. \tag{7}$$

A. Evaluation

The analytical model assumes customer arrival to follow a poisson distribution and also assumes the packet size to follow exponential distribution. But TCP as well as Randomized TCP does not follow these distributions in packet sending. So there arose a big difference in analytical and simulation results. We showed both results in Table V and Figure 6

VI. Conclusion

TCP is an important aspect in Computer Networking. Randomized TCP algorithm finds out a way of obtaining useful and automated services in data communication. This can introduce the best usage of the existing resources. The end-to-end delay evaluation in this paper makes Randomized TCP a more attractive and useful Transport layer solution. We compared the end-to-end delays of Randomized TCP and TCP NewReno by varying the queue size of channels. The results show that Randomized TCP's performance is peak over TCP NewReno when the queue size is very small. With the increase of queue sizes, the delay difference between Randomized TCP and TCP NewReno becomes narrow or small. In case of infinite buffer, Randomized TCP also shows good performance over TCP NewRenoIt. Thus, the users can easily enjoy their required best services through Randomized TCP. As a future work, we are planning to work on different queuing algorithms with Randomized TCP congestion control mechanism and specifically implementing a new queue mechanism which may give better performance in Randomized TCP congestion control than the existing queue mechanisms.

References

[1] J. F. Kurose and K. W. Ross, *Computer Netwoking: A top down approach featuring the internet*. Addison Wesley, 2002.

[2] G. Hasegawa, T. Tomioka, K. Tada, and M. Murata, "Simulation studies on router buffer sizing for short-lived and pacing TCP flows," *Computer Communications*, vol. 31, no. 16, pp. 3789 – 3798, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYP-4SGKBBD-3/2/cd284735365599e5edd17eac89380d2e

[3] K. Chandrayana, S. Ramakrishnan, B. Sikdar, and S. Kalyanaraman, "On randomizing the sending times in TCP and other window based algorithms," *Computer Networks*, vol. 50, no. 3, pp. 422–447, 2006.

[4] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions of Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[5] S. Shenker, L. Zhang, and D. D. Clark, "Some observations on the dynamics of a congestion control algorithm," *Computer Communication Review*, pp. 30–39, 1990.

[6] L. Zhang, "A new architecture for packet switching network protocols," Ph. D. dissertation, M.I.T. Lab. Comput. Sci., Cambridge, MA, 1989.

[7] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic," in *Proc. ACM SIGCOMM '91*, 1991, pp. 133–147.

[8] D. Mitra, "Asymptotically optimal design of congestion control for high speed data networks," *IEEE Trans. Commun.*, vol. 40, no. 2, pp. 301–311, 1992.

[9] D. Mitra and J. B. Seery, "Dynamic adaptive windows for high speed data networks with multiple paths and propagation delays," *Computer Networks and ISDN Systems*, vol. 25, pp. 663–679, 1993.

[10] S. Lee, M. Lee, D. Lee, H. Jung, and B. S. Lee, "Tcprand: Randomizing tcp payload size for tcp fairness in data center networks," in *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 1697 – 1705.

[11] P. Prakash, A. Dixit, Y. C. Hu, and R. Kompella, "The tcp outcast problem: Exposing unfairness in data center networks," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2012, pp. 30–30.

[12] W. Haitao, F. Zhenqian, G. Chuanxiong, and Z. Yongguang, "Ictcp: Incast congestion control for tcp in data center networks," in *Proceedings of the 6th International Conference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 13:1–13:12.

[13] F. K and V. K., "Notes and documentaiton, LBNL," http://www.isi.edu/nsnam/ns/.

[14] J. Aweya, M. Ouellette, and D. Y. Montuno, "A self-regulating TCP

acknowledgement (ACK) pacing scheme," *International Journal of Network Management*, vol. 12, no. 3, pp. 145–163, 2002.

[15] A. Razdan, A. Nandan, R. Wang, M. Sanadidi, and M. Gerla, "Enhancing TCP performance in networks with small buffers," in *Proceedings of the 11th IEEE International Conference on Computer Communications and Networks*. IEEE, 2002, pp. 39–44.

[16] M. Garetto and D. Towsley, "An efficient technique to analyze the impact of bursty TCP traffic in wide-area networks," *Performance Evaluation*, vol. 65, no. 2, pp. 181 – 202, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6V13-4NWCGRD-1/2/0fab98f6270f2f0706849a94e42091bd

[17] C.-Y. Luo, N. Komuro, K. Takahashi, and T. Tsuboi, "Paced TCP: A dynamic bandwidth probe TCP with pacing in adhoc networks," in *International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC*. IEEE, 2007, pp. 1–5.

[18] S. M. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with gateway adaptive pacing for multihop wireless networks with internet connectivity," *Computer Networks*, vol. 52, no. 1, pp. 180 – 198, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4PT29C0-2/2/a5366ee033e4e1a9b831190ad1c0b581

[19] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: routers with very small buffers," *SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 83–90, 2005.

[20] O. Masafumi, I. Maroto, and T. Tatsuro, "Application of TCP pacing to optical packet network with Fiber Delay Line buffers," in *Proceedings of the 7th IEEE International Conference on Optical Internet*. IEEE, 2008, pp. 1–2.

[21] D. Wischik, "Buffer sizing theory for bursty TCP flows," in *International Zurich Seminar on Communications*. IEEE, 2006, pp. 98–101.

[22] Q. Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "Evaluating android applications with multipath tcp," *Mobicom 2015*, pp. 230–232, 2015.

[23] B. Hesmans, H. Tran-Viet, R. Sadre, and O. Bonaventure, "A first look at real multipath tcp traffic," *Lecture Notes in Computer Science*, vol. 9053, pp. 233–246, 2015.

[24] T. Lakshman and U. Madhow, "The performance of tcp/ip for networks with high bandwidth-delay products and random loss," 1997. [Online]. Available: citeseer.ist.psu.edu/lakshman96performance.html

[25] U. of Texas TR-069, "Queueing network model," http://www.cs.utexas.edu/ftp/pub/techreports/tr81-167.pdf.