Simulation Based Analysis of TCP Friendly Rate Control in Wired Environment

P. Balakoteswara
M. Tech (Computer Science), CSE Dept., JNTUA College of Engineering, Anantapuramu, A.P, India.

C. Shoba Bindu
Associate Professor, CSE Dept., JNTUA College of Engineering, Anantapuramu, A.P, India.

Abstract – The traditional Internet is primarily designed to support non-real time traffic such as data in text and images. Now a days and in coming few years the real-time streaming media traffic such as video and audio traffic is going to become dominant traffic type. The bulk insertion of additional video traffic by many applications over Internet can cause a serious problem called congestion collapse which leads to performance degradation particularly for real time traffic. So this congestion should be controlled for getting better performance in multimedia applications. Transmission Control Protocol (TCP) supports congestion control techniques but up to some limits in higher rates and not suitable for real time media traffic. User Datagram Protocol (UDP) doesn't support any congestion control mechanisms and also unreliable, causing instability in the Internet. TCP Friendly Rate Control (TFRC) protocol a reliable, congestion control and end-to-end protocol, is more suitable for real-time streaming data because of smooth sending rate and friendliness with TCP flows achieved it. This paper shows the complete study of TFRC and performance of TFRC is compared with TCP Flows such as TCP New Reno, TCP Vegas, TCP Reno, and TCP Tahoe and also with UDP in wired environment.

Index Terms - TCP Friendly, Congestion Control, media traffic, TFRC, NS-2.

1. INTRODUCTION

Today's Internet is in search of a reliable and congestion control protocol even for multimedia traffic without losing its fair share of bandwidth over Internet. The present Internet infrastructure only provides best-effort service for non-real time traffic. Most of the major traffics of today's Internet are covered by real time streaming media applications such as video on demand, game playing, video conferencing systems etc. So this much of heavy video traffic on Internet is causing congestion collapse [1].

Transmission Control Protocol (TCP) [2] is a reliable and congestion control protocol [10] [20] but at higher rates it won't show its effectiveness. TCP Congestion Control mechanism reduces the sending rate by half in response to even a single packet drop. It shows unwanted aggressiveness when congestion occurs which multimedia applications doesn't want. To avoid [17] this situation different variations of TCP were proposed for congestion control [18] such as TCP-New Reno [3], TCP-Vegas [4], TCP-Reno [5], and TCP-Tahoe [6] etc. All of these try to modify the Additive Increase Multiplicative Decrease (AIMD) [16] [19] strategy to improve TCP performance. TCP was not primarily designed to support streaming media traffic, the saw-tooth behavior of TCP effects the perceived video quality. So, TCP need more improvements to support congestion control for real-time streaming media traffic.

User Datagram Protocol (UDP) [7] is selected for data transfer in higher rates even though it is unreliable. UDP doesn't support any congestion control mechanism this can lead to instability in the network. To solve all these problems a new protocol is designed by Internet Engineering Task Force (IETF) called TCP Friendly Rate Control (TFRC) [8]. TFRC is a reliable protocol for streaming applications and also it supports congestion control mechanism very effectively. It is an equation based approach rather than window based.

The rest of this paper is: Section 2 introduces the various TCP variants briefly that are previously published under related work. Section 3 describes our TFRC mechanism. Section 4 describes the performance metrics that are used to evaluate TFRC. Section 5 presents the simulation environment. Section 6 presents Results and Analysis of TFRC with other TCP Flows and UDP. Finally, Section 7 presents the conclusions and future enhancements.

2. RELATED WORK

Before Coming to TFRC there were a lot of research work has been done. There were many changes applied to the TCP to improve for supporting real-time media traffic. Some of those efforts are shown below.

- TCP TAHOE
- TCP RENO
- TCP VEGAS
- TCP NEW RENO

2.1 TCP-Tahoe

It is a simple modification of TCP protocol which includes Fast Retransmit mechanism. In Tahoe [6] triple duplicate Acknowledgements (ACKs) are treated as the same as a time

out then Tahoe will perform Fast Retransmit. In Fast Retransmit, the slow start threshold set to half of the current window size, reduces congestion window to 1 Maximum Segment Size (MSS), and it will reset to slow-start state.

2.2 TCP-Reno

By adding an additional functionality to TCP-Tahoe TCP-Reno [5] was developed. Here along with Fast Retransmit, Fast Recovery also added. In this mechanism if three duplicate ACKs are received, Reno will halve the congestion window instead of setting it to 1 MSS like Tahoe and sets slow start threshold to the new congestion window, performs a Fast Retransmit and enters a phase called Fast Recovery. If an Acknowledgement times out, slow start phase is used as it is with Tahoe.

In Fast Recovery state, TCP retransmits the missing packet that was indicated by three duplicate Acknowledges, and wait for an ACK of the whole transmit window before returning to congestion avoidance phase. If there is no ACK, TCP Reno experience a time out and enters the slow start state.

2.3 TCP Vegas

Until the mid-1990s, all of the TCP's set time outs and measured round trip delays were based upon only the last transmitted packet in the transmit buffer. But in TCP-Vegas [4] time outs were set and round trip delays were measured for every packet in the transmit buffer. In addition, TCP-Vegas uses additive increases in the congestion window.

2.4 TCP New Reno

TCP-New Reno [3] improves retransmission during the fast recovery phase of TCP Reno. It adds a small change to the Reno algorithm at the sender. The change is the sender's behavior during fast recovery when a partial ACK is received.

The partial ACK do not acknowledges all the packets that were out standing at the start of the fast recovery period but acknowledges some of them. This means that there were multiple loses in the same window of data.

In TCP-Reno, partial ACKs take TCP out of Fast Recovery by deflating the usable window back to the size of congestion window. In TCP-New Reno, partial ACKs do not take TCP out of Fast Recovery. Instead, partial ACKs received during Fast Recovery are treated as an indication that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. Thus, when multiple packets are lost from a single window of data, New Reno can recover without a retransmission time out, retransmitting one lost packet per RTT until all of the lost packets from window have been transmitted. TCP-New Reno remains in fast recovery until all of the data outstanding when Fast Recovery was initiated has been acknowledged.

After all these efforts a new protocol TCP Friendly Rate Control (TFRC) is designed and explained in next section.

3. PROPOSED MODELING

3.1 TCP Friendly Rate Control

TFRC [12] [13] is an end-to-end rate based congestion control reliable protocol to suite multimedia applications. TFRC is more suitable for real-time streaming data because of smooth sending rate achieved by it. TFRC is capable of dynamically adjusting next sending rate at the sender side when congestion occurs. The equation [11] [14] used by TFRC is shown in equation 1.

$$X = \frac{s}{RTT\sqrt{\frac{2bp}{3}} + RTO\left(3\sqrt{\frac{3bp}{8}}\right)p(1+32p^2)} \qquad (1)$$

X is the sending rate in bytes/seconds
s is the segment size in bytes
RTT is the round trip time in seconds
b indicates the number of packets acknowledged by a single ACK
p is the loss event rate (between 0 and 1)
RTO indicates the retransmission time-out in seconds (4RTT)

TFRC is a receiver driven mechanism, with the calculation of the congestion control [21] data (i.e., the loss event rate) in the receiver rather in the sender. The receiver must continuously maintain and update the loss event history data structure and continuously process the loss event rate and send it to sender as soon as it observes an increase in the loss event rate as a feedback.

3.2 Performance Metrics

End-to-End Delay : The average delay of all the packets while travelling from source node to the destination node.

Packet Loss Ratio : The ratio of number of lost packets to the sum of number of packets received and number of lost packets.

Packet Delivery Ratio : The ratio of total number of packets successfully received by the destination nodes to the number of packets sent by the source nodes.

4. SIMULATION ENVIRONMENT

The performance of the proposed TFRC is evaluated using the network simulator version ns-2.35 [9] [15], and the simulation results compared with all other TCP variants and UDP. The network topology used in this paper is a simple network dumbbell topology as shown in fig. 1.

Figure 1 shows a simple network topology consisting of six source nodes and six destination nodes. The access link capacities were 2 Mbps and the bottleneck capacity between nodes R0 and R1 also 2 Mbps. The access links delay was 0.002 seconds.

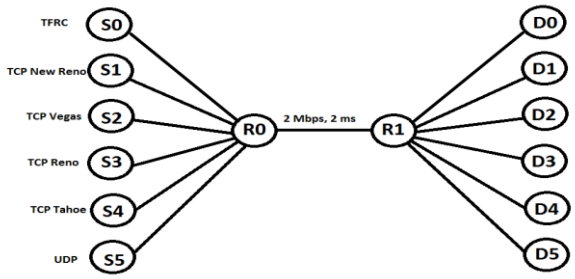

Fig. 1: Simple network dumbbell topology

5. RESULTS AND DISCUSSIONS

In this paper, we compare the performance of TCP-Tahoe, TCP-Reno, TCP-Vegas, TCP-Newrono, TFRC and UDP through Packet Delivery Ratio, Packet Loss Ratio, End-to-End Delay.

In figure 2, it shows end-to-end delay of TFRC is equal to TCP flows up to 4 Mbps, later compare to UDP, TFRC gives far better results. The end-to-end delay of TFRC is less than UDP. The end-to-end delay of TCP New Reno is second best protocol in all TCP flows. TCP Tahoe and Reno are giving least results. Here, we incremented UDP rate and TFRC rate simultaneously and also changing TCP window sizes for all other TCP flows. The reason why TFRC has better end-to-end delay is its dynamically adjusting next sending rate behaviour and no need to wait for feedback of each packet. When ever packet loss occurs then only it wait for feedback and adjusts its next sending rate with out loosing its fair share of bandwidth.

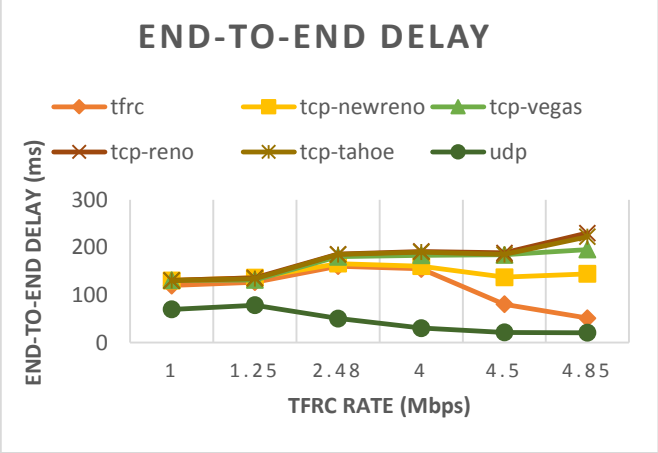

Figure 2: End-to-end delay of TFRC, UDP and TCP variants

In figure 3, it shows packet loss ratio of TFRC is almost friendly with TCP flows up to 4 Mbps, later compare to UDP, TFRC has low packet loss ratio. The packet loss ratio of TCP Newreno is second best protocol in all TCP flows. TCP Tahoe and Reno are giving least results. Here, we incremented UDP rate and TFRC rate simultaneously and also changing TCP window sizes for all other TCP flows. Here, the packet loss ratio of TFRC is far better than UDP because at higher rates also TFRC preserves reliability with low end-to-end deay.

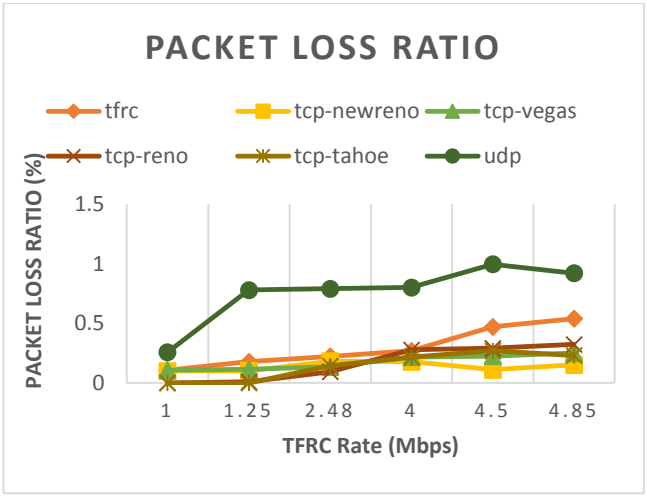

Figure 3: Packet Loss Ratio of TFRC, UDP and TCP variants

In figure 4, it shows Packet Delivery Ratio of TFRC is almost friendly with TCP flows up to 4 Mbps, later compare to UDP, TFRC has good packet delivery ratio. The packet loss ratio of TCP Vegas is second best protocol in all TCP flows. TCP Newreno and Reno are giving least results. Here, we incremented UDP rate and TFRC rate simultaneously and also changing TCP window sizes for all other TCP flows. The packet delivery ratio of TFRC is very good because it always follows the TCP flows at normal rates as we know TCP has good packet delivery ratio and also when compare to udp at higher rates because of its congestion control mechanism TFRC has good packet delivery ratio.

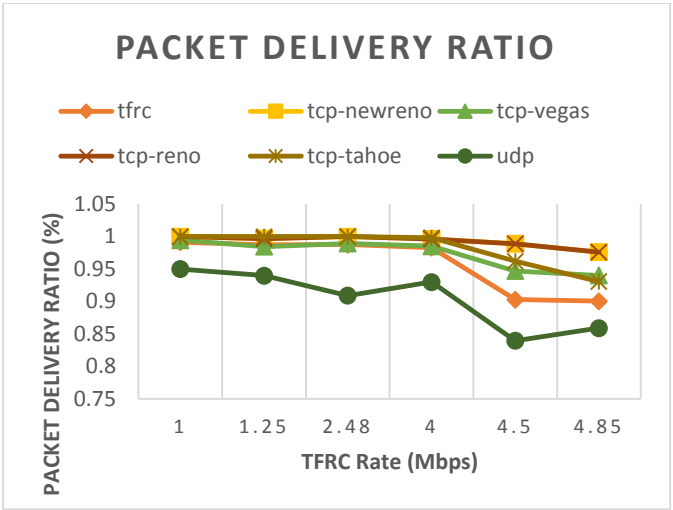

Figure 4: Packet Delivery Ratio of TFRC, UDP and TCP variants

6. CONCLUSION

By examining all these simulation results we can easily say that TFRC will be the best suitable protocol for real-time multimedia traffic. It's dynamically changing rate based congestion control approach is the so far best compare to all TCP flows. TFRC is capable of replacing both TCP and UDP. We can surely say that by observing these results, we can use TFRC for low and high data rates as a most reliable protocol. At higher rates we can say that TFRC is having higher packet delivery ratio. In Future, by shifting the overall receiver functionality like calculating packet loss ratio to sender, the performance of TFRC can be improved by reducing additional load on receiver and the delay for receiving feedback from receiver can be reduced.

REFERENCES

[1] K. Satyanarayan Reddy and Lokanatha C. Reddy, "A survey on congestion control mechanisms in high speed networks," IJCSNSInternational Journal of Computer Science and Network Security, vol. 8, no. 1, pp. 187 – 195, 2008.
[2] J. Postel, "Transmission Control Protocol", RFC-793, September 1981.
[3] Mohammad Reza and Reza Kourdy, "Tcp-newreno buffer management in network on chip," Journal of Computing, vol. 4, no. 7, pp. 128-130, Jul. 2012.
[4] Mao Kai, "A logarithmic slow-start algorithm of tcp-vegas in ip networks," Applied Mathematics and Information Sciences, vol. 7 no. 2, pp. 599-605, 2013.
[5] M. Nirmala, R.V. Pujeri, "Performance to tcp-vegas, bic, and reno congestion control algorithms on iridium satellite constellations," International Journal of Computer Network and Information Security, vol. 4, no. 12, pp. 40-49, Nov. 2012.
[6] TeamourEsmaili, A. N. rad, and Ghazal Lak, "Effect of multiple fast-retransmission of tcptahoe in decagon noc," Journal of Computing, vol. 4, no. 6, pp. 206-209, Jun. 2012.
[7] J. Postel, "User Datagram Protocol", RFC 768, August 1980.
[8] Handley, Floyd, Widmer and Padhye, "TCP-Friendly Rate Control (TFRC): Protocol Specification", IETF RFC 5348, April 2008.
[9] The VINT Project, "The ns Manual (formerly ns notes and documentation)", http://www.isi.edu/nsnam/ns/ns-ddocumentation.html, November 2011.
[10] Soo-Hyun Choi, "Congestion Control for Real-time Interactive Multimedia Streams," Ph. D. Thesis, University College London, Computer Science Dept., 2010.
[11] Agnieszka Chodorek and Robert R. Chodorek "Streaming Video over TFRC with Linear Throughput Equation," *Advances in Electronics and Telecommunications*, vol. 1, no. 2, Nov. 2010.
[12] Lisong Xu and Josh Helzer, "Media Streaming via TFRC: an Analytical Study of the Impact of TFRC on User-perceived Media Quality," *Computer Networks*, vol. 51, no. 17, pp. 4744-4764, 2007.
[13] S. Tsao, Y. Lai, and Y. Lin, "Taxonomy and Evaluation of TCP-Friendly Congestion Control Schemes on Fairness, Aggressiveness, and Responsiveness." *IEEE Network*, vol. 21, no. 6, , pp. 6-15, 2007.
[14] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, pp. 43 – 56, Aug. 2000.
[15] University of California Berkeley, "The network simulator - ns-2," [Online] http://www.isi.edu/nsnam/ns.
[16] Y. Yang and S. Lam, "General AIMD Congestion Control," *Proc. IEEE ICNP 2000*, Nov 2000, pp. 187–98.
[17] B. Braden, D. Clark, and J. Crowcroft, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Apr. 1998, http://www.ietf.org
[18] D. Bansal *et al*., "Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms," *Proc. ACM SIGCOMM '01*, Aug. 2001, pp. 263–74.
[19] A. Lahanas and V. Tsaoussidis, "Exploiting the Efficiency and Fairness Potential of AIMD-based Congestion Avoidance and Control," *Comp. Networks*, vol. 43, no. 2, Oct. 2003, pp. 227–45.
[20] J. Widmer, R. Denda, and M. Mauve, "A Survey on TCP-friendly Congestion Control," special issue of the *IEEE Network Control of Best Effort Traffic*, vol. 15, no. 3, May–June 2001, pp. 28–37.
[21] G. Jourjon, E. Lochin, and L. Dairaine, "Optimization of tfrc loss history initialization," *Communications Letters, IEEE*, vol. 11, no. 3, pp. 276–278, March 2007.

Authors

Balakoteswara Panchakshari received B.Tech degree in Computer Science and Engineering from Annamacharya Institute of Technology and Sciences, Rajampet, affiliated to JNTUA College of Engineering, Anantapuramu, A.P, India, during 2008 to 2012. Currently pursuing M.Tech in Computer Science from JNTUA College of Engineering, Anantapuramu, A.P, India, during 2013 to 2015 batch. His Area of interests include Computer Networks, Network Security.

C. Shoba Bindu is an Associate Professor of Computer Science and Engineering at Jawaharlal Nehru Technological University College of Engineering, Ananthapuramu. She obtained her Bachelor degree in Electronics and Communication Engineering, Master of Technology in Computer Science from Jawaharlal Nehru Technological University Hyderabad and Ph.D. in Computer Science and Engineering from Jawaharlal Nehru Technological University Anantapuramu. She has published several Research papers in National \ International Conferences and Journals. Her research interests includes network security and Wireless communication systems.