IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. VI (Mar-Apr. 2014), PP 49-53
www.iosrjournals.org

Performance Evaluation of High Speed TCP Variants in Dumbbell Network

[1]Subramanya P, [2]Vinayaka K S, [3]Gururaj H L, [4]Ramesh B
[1, 2]Department of CS&E, Malnad College of Engineering, Hassan, India.
[3]Assistant Professor Department of CS&E, Malnad College of Engineering, Hassan, India
[4] Professor, Department of CS&E, Malnad College of Engineering, Hassan, India

Abstract: *Dumbbell topology evolved as a key topology for a large variety of applications ranging from home networking to transportation systems, campus network and medical systems. TCP is one of the core protocols of the internet protocol suite, reliability is the main constraint for the traffic flows in TCP. Even though congestion control mechanisms are implemented, packet losses are encountered which leads to performance degradation. In this paper, the congestion control algorithms are implemented and corresponding results are analyzed in Dumbbell topology, it is typically used to analyze the TCP traffic flows. Fairness of throughput is evaluated for different TCP variants using network simulator.*

Index Terms: *Acknowledgement, Congestion Window, Dumbbell Topology, Slow-start Threshold.*

I. Introduction

Dumbbell network has become popular in recent years due to the high demand for continuous network connectivity. Dumbbell network topology has emerged as a promising wired technology for a large variety of applications, Applications of dumbbell topology range from home networking to transportation systems and medical systems. In other words, dumbbell topology has evolved as a key network to evaluate the traffic flows in TCP variants.

TCP is a connection oriented end-to-end protocol. It has been widely adopted as a reliable transfer protocol for most of the communication network. TCP performs a 3way handshake for the connection establishment. TCPs' primary goal is to provide a connection oriented reliable data transfer service between different applications to be able to provide these services on top of unreliable communication systems. TCP needs to consider data transfer, reliability flow control, and multiplexing and congestion control.
A Network is said to be congested when the traffic offered to it exceeds the available bandwidth capacity. Van Jacobson proposed a new principle called as "conservation of packets", which means that anew packet is not injected into the network until an old packet leaves the network [3].The network is not perfect anda small percentage of packets are lost, either due to network error or due to the fact that there is congestion in the network and the routers are dropping packets. We shall assume that packet losses due to network loss are minimal and most of the packet losses are due to buffer overflows at the router [19]. Thus it becomes increasingly important for TCP to react to a packet loss and take action to reduce congestion.TCP ensures reliability by starting a timer whenever it sends a segment. If it does not receive an acknowledgement from the receiver within the 'time-out' interval then it retransmits the segment.

Van Jacobson [3] proposed three algorithms for congestion avoidance and control: Slow-Start, Congestion Avoidance and Fast Retransmit. Slow-Start algorithm is designed to start the Self-Clocking mechanism. This algorithm quickly fills the empty pipeline (network is viewed as a pipeline) at the beginning of transmission or after a retransmission timeout to bring the connection towards its equilibrium (a connection is said to be in equilibrium if it is running stably with a full window of data in transit). Congestion Avoidance algorithm, also known as Additive Increase/Multiplicative Decrease (AIMD) algorithm, closely obeys the "Conservation of Packets" principle once the connection is in equilibrium. Fast Retransmit algorithm considers duplicate acknowledgements [5] as a sign of packet loss in the network and retransmits the lost packet without waiting for a retransmission timer to expire.

Initially by taking a brief look at each of the congestion avoidance algorithms and noting how they differ from each other in the Dumbbell network. In the end we evaluate the performance of TCP variants in dumbbell network. The performance is measured in terms of overall network throughput.
The remainder of the paper is organized as follows. In Section II we discuss the congestion control mechanisms of each TCP variant. Section III presents different simulation environment designed for Dumbbell networks and describes the performance metrics in detail. Section V discusses the simulation results. Section VI gives conclusion and possible future directions.

II. Tcp Variants

A. TCP Reno

TCP Reno retains the basic principles of TCP such as slow start. However it adds some intelligence over it so that lost packets are detected earlier and the network is not emptied every time a packet is lost. Initially the congestion window is set to 1 [3]. The congestion window size increases exponentially, i.e. the congestion window size is incremented for every acknowledgement received.

$$Cwnd \longleftarrow cwnd + 1 \tag{1}$$

The congestion window size increases exponentially until it encounters the *ssthresh.* Once it encounters the *ssthresh*, the congestion window is incremented as follows.

$$Cwnd \longleftarrow cwnd + \frac{1}{cwnd} \tag{2}$$

The logic behind Reno is we get duplicate acknowledgements if the next expected packet in the sequence is delayed in the network, the segments reached there in out of order or that the packet is lost. If at all we receive a number of duplicate acknowledgements, it means that the sufficient time has passed even if it takes the longest path in the network, then the probability of the packet being lost is very high. [5] Reno detects congestion as soon as it receives 3 duplicate acknowledgements. Without waiting for the time
Out, thus it retransmits the lost packet without emptying the network. The modification of TCP Reno is that it retransmits the lost packet without setting the congestion window size to 1, which empties the network. It enters into an algorithm which we call “Fast-Retransmit”. The basic algorithm is presented as

Every 3 duplicate acknowledgements are taken to be a segment is lost and we retransmit the lost segment immediately and enters the fast recovery phase.

Ssthresh is set to half of the current window size and *cwnd* is also set to the same.i.e.

$$Cwnd = ssthresh \longleftarrow \frac{cwnd}{2} \tag{3}$$

For each duplicate acknowledgement received increase the cwnd by one. If the increased cwnd is greater than the amount of data in the pipe then transmit a new segment else wait. Thus we don’t empty the pipe, we just reduce the flow. We continue with congestion avoidance phase of Tahoe after that.

Drawback - Reno performs very well over TCP when the packet losses are small. [19] If we have multiple packet losses in a segment, Reno doesn’t perform too well and the performance is as same as Tahoe. The reason is if we have multiple packet drops, the information about packet loss comes only after we receive 3 duplicate acknowledgements. But the information of the second packet loss comes only after the we receive the acknowledgement of the re-transmitted packet i.e. one round trip time.

B. TCP New-Reno

New-Reno is a modification of Reno. TCP
New-Reno detects multiple packet losses, hence it is much more efficient than Reno in the context of multiple packet losses. New-Reno also enters the fast-retransmit phase like Reno. However it doesn’t exit the fast-recovery phase until all the unacknowledged data in the pipeline which was outstanding at the moment it entered the fast recovery phase is acknowledged. [5] Henceforth it overcomes the drawback of TCP Reno of reducing the cwnd multiple times.

The increase and decrease parameters in the congestion window size are same as in Reno which is depicted in (1) and (2).

The fast-retransmit phase is as same as in Reno, the difference comes in the fast recovery phase where it allows the multiple retransmissions of the data. [19] When New-Reno enters the fast-recovery phase it makes a note of the data which is outstanding. Then the fast recovery phase proceeds as in Reno. However when a fresh acknowledgement is received, then there are 2 possibilities.

If it acknowledges all the data which were outstanding when it entered the fast-recovery phase then it exits fast recovery and sets the cwnd to ssthresh which is set to half of the current cwnd. And enters the congestion avoidance phase and continues as in Reno and Tahoe which is depicted in (3).

If the acknowledgement is partial acknowledgement, then it deduces that the consecutive data in the pipe was lost and re-transmits the unacknowledged data and sets the number of duplicate acknowledgements to zero. Then it exits the fast-recovery and enters the congestion avoidance phase.

Drawback–New-Reno suffers from the fact that it has to wait for one round trip time to detect every single packet loss. [5] When the acknowledgement of the re-transmitted data is received, only then we can deduce which other segments of data was lost.

C. High Speed TCP(HSTCP)

High speed TCP was proposed to improve the performance of TCP connections by increasing the size of congestion window. [9] HSTCP overcomes the current congestion control algorithms which limit the network resource utilization by making minor modifications. [2] However TCPs' main aim is to transmit the data in an efficient manner with respect to time and packet drops. HSTCP retains the basic principles such as slow-start and time out with slight changes in it. High speed TCP introduces a relation between the packet drop rate and an average congestion window. Whenever an acknowledgement is received, the congestion window is increased as shown below.

$$cwnd \leftarrow cwnd + \frac{a(cwnd)}{cwnd} \qquad \textbf{(4)}$$

Whenever the congestion is detected, the cwnd is updated as shown below

$$cwnd \leftarrow cwnd - b(cwnd)*cwnd \qquad \textbf{(5)}$$

HSTCP performs as same as the standard TCP when the congestion window is small, the values for a and b are 1 and 0.5 correspondingly. [3] Once the congestion window size is beyond certain threshold the window size a increases more aggressively than the standard TCP, while the value of b dips to 0.1 from 0.5. [8] This behavior helps HSTCP to be compatible with standard TCP flows in networks and also to quickly utilize the available bandwidth in network with large bandwidth delay products.

Drawback–HSTCP in fact suffers from the round trip time unfairness and TCP unfairness problems. RTT unfairness is defined as the ratio of *cwnd*in terms of RTT ratio of multiple TCP connections [10]. Moreover, the performance of regular TCP flows is largely affected by the aggressiveness of HSTCP. This is known as TCP-unfairness. [15]

D. Scalable TCP(STCP)

Scalable TCP is similar to the HSTCP's aggressive increase and decrease algorithm. Scalable TCP modifies the standard congestion control algorithms. [16] Instead of halving the congestion window size, every data loss decreases the congestion window by a small fraction until the data loss stops. [3] Once the data loss stops, the rate is ramped at a slow fixed rate. However, the increase and decrease rates in STCP are constant rather than HSTCP's varying current window size. Whenever an acknowledgement is received, the cwnd is updated as.

$$cwnd \leftarrow cwnd + a \qquad \textbf{(6)}$$

When congestion is detected, cwnd is updated as.

$$cwnd \leftarrow cwnd - (b*cwnd) \qquad \textbf{(7)}$$

The values of a and b are fixed to 0.01 and 0.125, the time taken by STCP to double its sending rate at the source is 70 round trip time's for any rate and henceforth the algorithm is scalable. However, TCP-unfairness and RTT-unfairness problems are the major drawbacks like HSTCP[10].

III. Simulation Setup And Methodology

The results in this paper are based on the simulations done on ns-2, a discrete event simulator [18]. We have chosen dumbbell topology for the study.

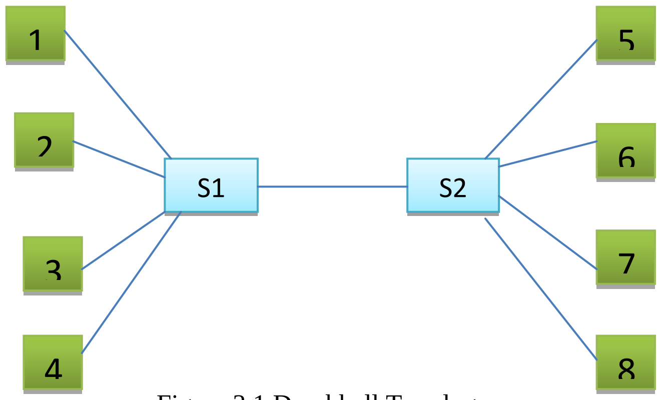

Figure 3.1 Dumbbell Topology

A. Topology

A Dumbbell topology is modeled with the network simulator-2 consisting of 8 hosts and 2 switches. This topology is easily scaled to different sizes. Different resources have their unique communication addresses, so here assumed that all switches has attached processor core as resources therefore treated similarly except that a traffic generator can be attached to resources. Switch, resource and link are three basic elements in the topology. Assume that the each resources has infinite buffer size but finite in switches. It means that the packet being dropped or lost cannot occur in resources but only take place in switches.

B. Communication Links

An inter-communication path between the switches is composed of links. Each node is connected with point-to-point bidirectional links. The bandwidth and latency of the link is configurable. When any link down between two nodes it implies that the packet cannot be travel between these nodes in any direction. Because bidirectional links are actually implemented using a single wire.

C. Communication traffic

The traffic flow in the dumbbell network is simulated and analyzed using different TCP variants such as: TCP Reno, TCP New Reno, HSTCP and STCP. The results and analysis are depicted in the next section.

IV. Results and Analysis

The traffic flow is determined using NS2 simulator. Table I shows the throughput (in Kbps) obtained for each high-speed TCP variant in dumbbell network. Throughput decreases as we increase the number of hops for all high-speed TCP variants as shown in the table. Our studies show that all high-speed TCP variants consistently perform better with dumbbell network.

In this scenario simulation model uses the packet generation of traffic using file transfer protocol (FTP) mechanism. FTP represents a bulk data transfer of large size where the packet size and interval of packet generation are fixed. The following Tcl code shows the traffic configuration setting for FTP packet generation

```
set ftp0 [new Application/FTP]
$ftp0 set packet Size_ 1000
$ftp0 set interval_0.0625
```

In our simulation the packet size is set to 1000 in all the cases. The table 4.1 depicts the results.

Table 4.1 Comparative Analysis

TCP variants	Packets sent (%)	Packets Lost (%)	If Multiple packets Lost
Reno	53.69%	46.31%	Worst
New-Reno	68.06%	31.94%	Better
HSTCP	56.91%	43.09%	Average
STCP	54.55%	45.45%	Average

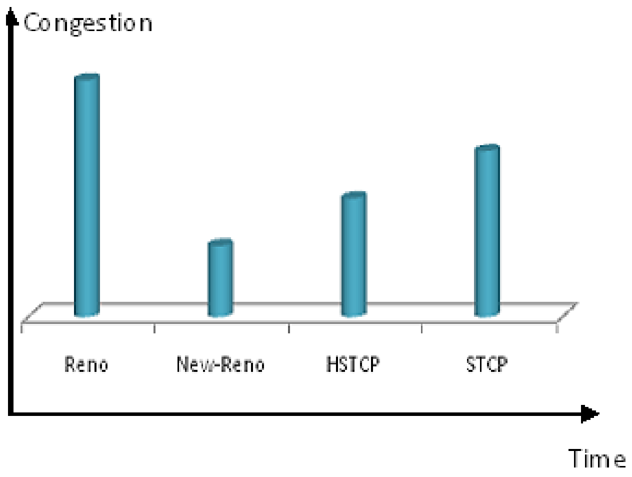

Figure 4.1 Congestion Graph

Later we simulated the High speed TCP variants: HSTCP and STCP. in HSTCP the throughput is low compared to New-Reno, but the sending rate is high. At last we evaluated for STCP for dumbbell topology where the increase and decrease parameters are constant which leads to a low throughput of data.
With these simulation results, the traffic flow of TCP using New-Reno in the dumbbell network out performs compared to other TCP variants.

V. Conclusions

We evaluated the performance of high-speed TCP variants in terms of throughput using dumbbell network. It is observed that the performance of TCP largely depends on the congestion in the network. From our simulation results, TCP New-Reno performs better for congestion control. In this study we have not considered other performance parameters such as Convergence speed, RTT fairness and TCP fairness. In future, we intend to study the performance of high-speed TCP variants with above mentioned parameters to improve the performance of TCP in Dumbbell network.

References

[1] Mohith P. Tahilaini “Comparative study of High Speed TCP variants in Multihop Wireless Networks,” IJCTE, Vol. 5, No. 5,Oct 2013

[2] I. F. Akylidiz, X. Wang, and W. Wang, “Wireless mesh networks: a survey,” Computer Networks, Elsevier, pp. 445-487, January 2005.

[3] T. Kelly, “Scalable TCP: Improving Performance in High Speed Wide Area Networks,” ACM SIGCOMM Computer Communication Review, vol. 33, pp. 83-91, 2003.

[4] V. Jacobson, “Congestion avoidance and control,” Proceedings of SIGCOMM ’88, ACM, Stanford, CA, Aug. 1988.

[5] K. Fall and S. Floyd, “Simulation-based comparisons of Tahoe, Reno, and SACK TCP,” ACM Computer Communication Review, vol. 26, no. 3, pp. 5-21, 1996.

[6] S. Floyd and T. Henderson, “The newreno modification to TCP’s fast recovery algorithm,” Request for Comments 2582, Experimental, April1999.

[7] L. Brakmo and L. Peterson, “TCP Vegas: end-to-end congestionavoidance on a global internet,” IEEE Journal on Selected Areas in Communication, vol. 13, pp. 1465-1480, Oct. 1995.

[8] K. T. J. Song, Q. Zhang, and M. Sridharan, “A compound TCP approach for high-speed and long distance networks,” in Proceedings of PFLDNet, 2006.

[9] S. Floyd, “Highspeed TCP for Large Congestion Windows,” Request for Comments 3649, Experimental, 2003.

[10] L. Xu, K. Harfoush, and I. Rhee, “Binary Increase Congestion Control for fast long-distance networks,” in Proceedings of IEEE INFOCOM,Hong Kong, 2004.

[11] I. Rhee and L. Xu, “CUBIC: A new TCP-friendly high-speed TCP variant,” Proceedings of the third PFLDNet Workshop, France, 2005.

[12] C. Jin, D. X. Wei, and S. H. Low, “FAST TCP: motivation, architecture, algorithms, performance,” in Proceedings of IEEE INFOCOM, Hong Kong, 2004.

[13] TCP Evolution and Comparison, Which TCP will Scale to Meet the Demands of Today’s Internet? Whitepaper, FastSoft, Pasadena, 2008.

[14] G. Holland and N. Vaidya, “Analysis of TCP Performance over Mobile Ad Hoc Networks,” ACM/IEEE MOBICOM ’99, Seattle, Washington,Aug. 1999.

[15] E. D. Souza and D. Agarwal, “A HighSpeed TCP Study: Characteristics and Deployment Issues,” LBNL Technical Report, Berkeley, 2003.

[16] M. Gerla, K. Tang, and R. Bagrodia, “TCP Performance in Wireless Multi-hop Networks,” in Proceedings of IEEE WMCSA ’99, NewOrleans, LA, Feb. 1999.