2012 International Conference on Communication Systems and Network Technologies

Throughput Based Comparison of Different Variants of TCP in Optical Burst Switching (OBS) Network

Sagar H. Sodhatar
Electronics and Communication Department
L. D. College of Engineering
Ahmedabad, India
sagar.sodhatar@gmail.com

Rohit B. Patel
E.C. Department
U. V. Patel College of Engg.
Kherva-Mehssana, India
rbp_ec@yahoo.co.in

Janardana V. Dave
Electronics and Communication Department
L. D. College of Engineering
Ahmedabad, India
prof_jvdave@hotmail.com

Abstract—**Optical Burst Switching (OBS) is a promising switching paradigm for all-optical WDM networks. It combines advantages of both Optical circuit switching (OCS) and Optical packet switching (OPS) and avoids the disadvantages. In this paper, we present a dumb-bell topology as our simulation network. We are using three different variants of TCP that are TCP-Tahoe, TCP-Reno and TCP-New Reno. This study represents results of throughput from an experimental study of TCP source variants, Tahoe, Reno and New Reno. We measure the throughput of each variant by considering the network parameters such as, bandwidth, packet size, congestion window size and queue-limit. Here we construct a standard dumb-bell topology. The dumb-bell topology is used to study the effect of a bottle-neck link shared by many sources. All simulations performed using NS-2 network simulator. After getting the trace for each scenario we analyzed the behavior of three TCP variants and results show that New Reno gives better throughput for wired OBS network in predefined simulation criteria.**

Keywords-Optical Burst Switching (OBS); Network Simulator version-2 (NS-2); TCP variants; Dumb-bell topology; Congestion Window (cwnd).

I. INTRODUCTION

There has been a phenomenal increase in the demand for bandwidth-hungry applications and services. The increase in demand for bandwidth over the years due to rapid growth in the number of Internet users and increase in bandwidth intensive applications such as voice-over-IP, video conferencing, interactive video on demand, and many other multimedia applications [1]. To meet the ever growing demand of bandwidth, copper cables were replaced by optical fibers in both the access networks as well as in the backbone networks [2]. Optical data communication has been acknowledged as the best solution to meet the present bandwidth requirement of users and supporting future network services. This is because theoretically optical fiber has the ability to support bandwidth demand up to 50 THz.

Light wave has higher frequency and hence shorter wavelength, therefore more bits of information can be contained in a length of fiber versus the same length of copper. Apart from this, optical fiber provides extremely low bit-error rate of the order of 10^{-12}. Optical signals are immune to electrical interferences. Fiber cables are much more difficult to tap than copper wires, so there is a security advantage in optical communication.

The first generation optical networks, fibers were used as point-to-point connections. The entire bandwidth available for transmission was not fully exploited. This is because electronic equipments operate at an order of gigabits per second, whereas the fiber has a bandwidth of terabits per second. This mismatch between electronic speed and the optical bandwidth is called electronic bottleneck. Representative of first generation optical networks are SONET/SDH. In second generation Wavelength Division Multiplexing (WDM) technology were deployed to overcome the problem of electronic bottleneck. WDM is the optical version of frequency division multiplexing (FDM). WDM divides the available bandwidth of a single fiber into a number of non-overlapping wavelength channels. Each of the wavelength channels operate at the electronic speed. Several signals are transmitted at different wavelengths in a single fiber at the same time. Thus, WDM encapsulates many virtual fibers in a single fiber. The main advantages of WDM technology are transparency, scalability and flexibility.

II. INTRODUCTION TO TCP

The Transmission Control Protocol was created in the 1970s to make connections across the Advanced Research Projects Agency-net (ARPAnet) and to replace the Network Control Protocol. In 1978 the Internet Protocol (IP) was added to TCP to take over the routing of messages, which resulted in the TCP/IP protocol suite. In 1981 the RFC 793 [3], Transmission Control Protocol, was published. TCP is a communication protocol which is connection-oriented and has a reliable delivery. The application layer sends a bytestream to transport layer, where TCP divides the stream into segments. TCP then sends these segments to the Network layer, where IP handles the sending across the network. To keep track of the packets and keep them in order on the other side, TCP gives each packet a sequence number, which upon reception by the receiving TCP module is acknowledged. If no acknowledgement is received by the sending side within a reasonable time, the sender presumes the packet is lost and resends the packet. With the usage of a checksum, which is computed by the sender and included in the header of the TCP packet, the receiver checks if a packet is damaged by computing the checksum and comparing its own with the one sent along with the packet.

A. *TCP Incarnations*

TCP is a layer-4 protocol in the 7-layer Open Systems Interconnection (OSI) model. TCP sits on top of the Internet Protocol (IP), and is used by applications to communicate between servers and clients. TCP is a connection-oriented and robust transmission algorithm, in that it establishes a connection between client and server before sending data, and in that it acknowledges all the data that is sent, retransmitting any data lost along the way. TCP is commonly used for applications such as mail (IMAP, SMTP, POP3) file transfer (FTP), file sharing(peer-to-peer), web-surfing (HTTP), music downloads (iTunes) and video downloads (VoD, YouTube). In the beginning of the Internet, i.e. ArpaNET, TCP was very simple and would start by transmitting a whole window's worth of data. Under congestion conditions which started to emerge around 1986, intermediate routers were forced to discard many packets. TCP, as it existed at the time, would actually double the rate at which packets were sent, pushing the entire network into a state of congestion collapse, where the network effectively ground to a halt, with the operational capacity of networks dropping by a factor of one thousand.

B. *Slow start*

The slow start mechanism was a direct attempt to avoid congestion collapse by increasing the packet rate of a connection in a controlled fashion – slowly at first, faster later on - until congestion is detected (i.e. packets are dropped), and ultimately arrive at a steady packet rate, or equilibrium. To achieve this goal, the designers of the slow start mechanism chose an exponential ramp-up function to successively increase the window size. Slow start introduces a new parameter called the congestion window or *cwnd*, which specifies the number of packets that can be sent without needing a response from the server.

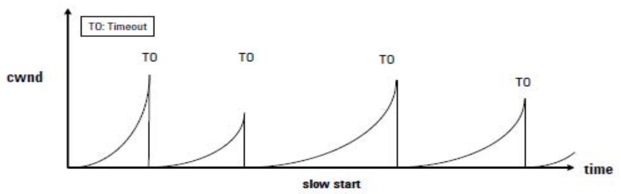

Figure 1. TCP congestion window dynamics with slow start [6]

TCP starts off slowly (hence the name "slow start") by sending just one packet, then waits for a response (an ACK) from the receiver. These ACKs confirm that it is safe to send more packets into the network (i.e. double the window size), rather than wait for a response packet by packet. The window size grows exponentially until a router in between the sender and the receiver discards (drops) some packets, effectively telling TCP through a time-out event that its window size has grown too large. Figure 1 illustrates how the TCP window size is adjusted dynamically over time with the slow-start mechanism. Slow start is usually characterized as a "congestion control" mechanism, and is commonly implemented in modern implementations of TCP.

C. *Congestion Avoidance*

When a network is congested, queue lengths start to increase exponentially. Congestion avoidance was devised as a technique to signal packet loss via time-outs and make TCP throttle back quickly (more quickly than queue lengths are growing), with the objective of stabilizing the whole system. Near the point of congestion, overly aggressive increases in connection bandwidth can drive the network into saturation, causing the "rush-hour effect". To avoid the rush-hour phenomenon, the congestion avoidance mechanism increases bandwidth additively rather than exponentially. When congestion is detected via a timeout, bandwidth is scaled back aggressively by setting *cwnd* to half the current window size. Congestion avoidance is sometimes characterized as being an additive-increase, multiplicative-decrease (AIMD) technique. While slow start and congestion avoidance were devised separately and for different purposes, they are almost always implemented together. The combined algorithm introduces a new variable called *ssthresh* (slow-start threshold), that effectively determines which mechanism is used. As shown in Figure 2, if *cwnd* is less than or equal to *ssthresh*, then TCP is in slow start. If *cwnd* is greater than *ssthresh*, then TCP is in congestion avoidance.

D. *Fast Retransmit*

Early TCP was designed with a simple retransmission timer that was very coarse. If packets were lost due to network congestion, considerable idle time was wasted before TCP would start transmitting again. Fast retransmit is an attempt to accelerate TCP's responsiveness through the use of ACK packets. As a TCP receiver receives each packet, it is continually sending ACKs back to a TCP sender. A TCP receiver sends back duplicate packets when a packet is lost, or packets are received in the wrong order. If three duplicate ACKs are sent back from receiver to sender, an assumption is made that packets have been lost in the network and they must be re-transmitted. A TCP sender then re-transmits the segment that has been lost without waiting for a re-transmission timer to expire.

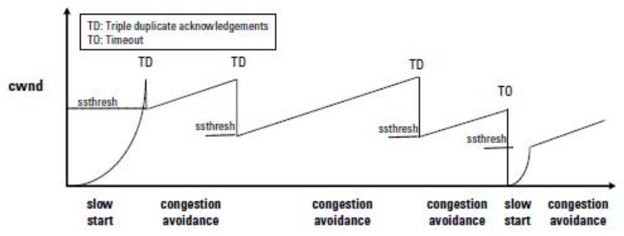

Figure 2. TCP congestion window dynamics with slow start and congestion avoidance [6]

E. *TCP Tahoe*

When the first three of these techniques (slow start, congestion avoidance, fast retransmit) are used together, that implementation is nicknamed “Tahoe”.

F. *Fast Recovery*

Fast recovery works hand in hand with fast retransmit to improve the responsiveness of TCP once congestion has occurred. Fast recovery introduces the concept of partial acknowledgements, which are used to notify that the next in-sequence packet has been lost so it can be re-transmitted immediately. Furthermore, instead 0 of going back into slow-start mode, fast recovery jumps TCP into congestion avoidance mode. The congestion window is also reduced to the slow start threshold *(ssthresh)* instead of dropping all the way back to a window size of one packet. The four techniques above are described in IETF RFC 2581[4].

G. *TCP Reno*

When the first four of these techniques (slow start, congestion avoidance, fast retransmit and fast recovery) are used together, that implementation is nicknamed “TCP Reno”.

H. *Modified Fast Recovery*

Engineers noticed that packets tended to be dropped from queues in bursts, and sought to improve TCP Reno’s performance in that situation, i.e. specifically when multiple packets were dropped from the same window. Modified fast recovery introduces a new variable called recover and is detailed in RFC 3782[5].

I. *TCP New Reno*

When TCP uses the modified fast recovery scheme along with the four congestion techniques above, that implementation is nicknamed “TCP New Reno”.

III. Simulation Scenario

In this paper we present a dumb-bell topology as our OBS network. The network to be simulated is depicted in Figure3.

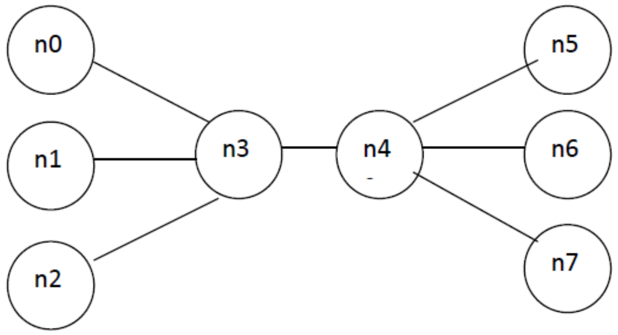

Figure 3. Simulated Network (dumb-bell topology)

This network, we are proposing having 8 nodes. The nodes are connected by duplex links. We attach TCP agents to the transmitting node and TCPSink agents to the receiving nodes. The TCP sender sends data to TCPSink agent and processes its acknowledgments. The TCP agent does not generate any application data on its own; instead, the simulation user can connect any traffic generation module to the TCP agent to generate data. Two applications are commonly used for TCP: FTP and Telnet. FTP represents a bulk data transfer of large size, and telnet chooses its transfer sizes randomly from tcplib. These applications work by advancing the count of packets available to be sent by a TCP transport agent. The actual transmission of available packets is still controlled by TCP’s flow and congestion control algorithm. Here, we attach TCP source variant Tahoe to node n0, Reno to node n1 and New Reno to node n2. Nodes n0, n1 and n2 are source nodes and nodes n5, n6 and n7 are destination nodes. The data from all three source nodes are transferred via the link between node n3 and node n4. The link between these two nodes is known as “Bottle-neck link”. The bandwidth and delay of the links between the nodes n0-n3, n1-n3, n2-n3, n4-n5, n4-n6 and n4-n7 is 300Mb and 20ms respectively. The bandwidth and delay of the bottle-neck link is 60Mb and 60ms respectively. We measure the throughput of these three different variants of TCP. Throughput can be defined as how much data can be transferred from one location to another in a given amount of time. Simulation time is taken as 5sec. Simulation process can be given as follow:

- Create a dumb-bell topology as depicted in Figure 3.
- Establish connection between the source and destination nodes.
- Attach the agent to node which will communicate on the behalf of particular node.
- Attach traffic source to node which will generate the real time load (or data).
- Start the simulation process. Start and stop time of the simulation process is defined by the user.
- When the simulation process is complete, NAM and TRACE file has been created by the software.
- NAM file provides the visualization of the topology created.

- TRACE file provides the real time statistical data of the topology created.
- We will collect data in appropriate form using AWK file in store it in text file.
- Plot the data stored in text file using XGRAPH or GNUPLOT function.

In the next section we discuss the numerical results of simulation network. We plot the graphs of Throughput v/s Time for all three different source variants using x-graph command in NS2.

IV. Numerical Results

We start the simulation and got the statistical data in the form of trace file. From these data we plot the graph of throughput v/s time for TCP source variants Tahoe, Reno and New Reno. We start the simulation with packet size is equal to 1460 bytes. We kept maximum congestion window size is equal to 10 for all variants. The all other network parameters are kept constant throughout the simulation i.e. bandwidth and delay of the bottle-neck link. We increase the packet size of the source variant and measure the throughput of the variant. To support our numerical results, we have shown the behavior of *cwnd* as a function of time. The graphs of congestion window v/s time help us to understand the nature of source variants Tahoe, Reno and New Reno. After that, we increase the packet size i.e. 2000, 3000, and 4000. We measure the throughput and analyze the behavior of each source variant. The slow start and congestion avoidance phase of *cwnd* will help us to understand the behavior of throughput as we are increase the packet size of the source variant. The throughput is start increasing and keep on increasing until the congestion is occurred. When congestion occur the all three TCP variants reacts individually. TCP Tahoe uses three techniques to resolve the congestion these are *slow start, congestion avoidance and fast retransmit* [4].

From the results, we can see that Tahoe increases the size of congestion window exponentially. But when congestion occurred, Tahoe decreases the *cwnd* directly to initial window size that is equal to 1. While, Reno and New Reno immediately half the *cwnd* and then reduces it to 1. We see that how all three variants of TCP react to congestion. TCP Tahoe again enters to *slow start* phase after the congestion has been faced. While, Reno and New Reno do not directly enter the *slow start phase* but they will continue transmission by entering into *congestion avoidance phase* [4, 5]. After entering into congestion avoidance phase the throughput is gradually increase which is quite obvious. The logic behind this is that until the congestion has faced the TCP variants increase window size "exponentially" and once the congestion is detected they will increase window size "linearly". From the results of congestion window v/s time, we are able to differentiate between the slow start and congestion avoidance phase of all three TCP variants. The detailed characteristics of three TCP variants are given in [4, 5]. TCP New Reno uses modified fast recovery technique to resolve the congestion and hence gives the better throughput results [5].

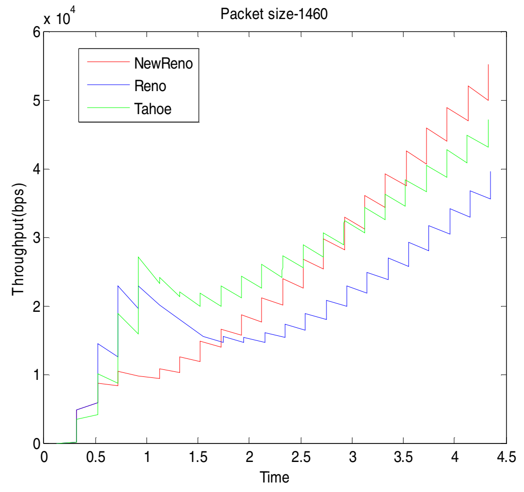

Figure 4. Graph of Throughput v/s Time

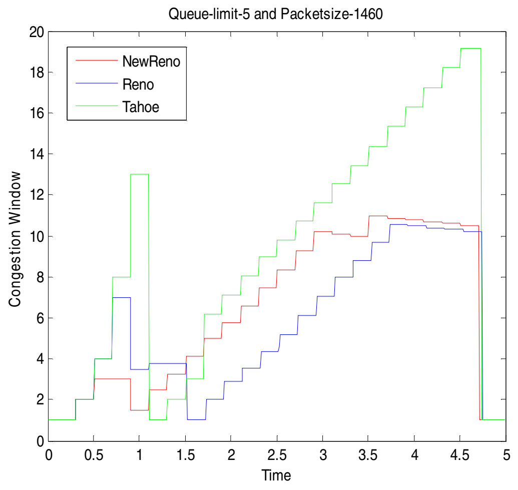

Figure 5. Graph of Congestion window v/s Time

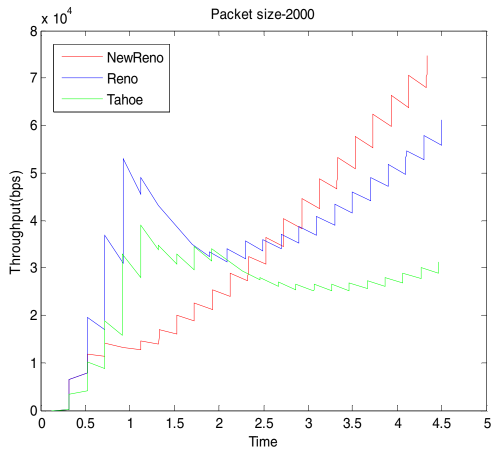

Figure 6. Graph of Throughput v/s Time

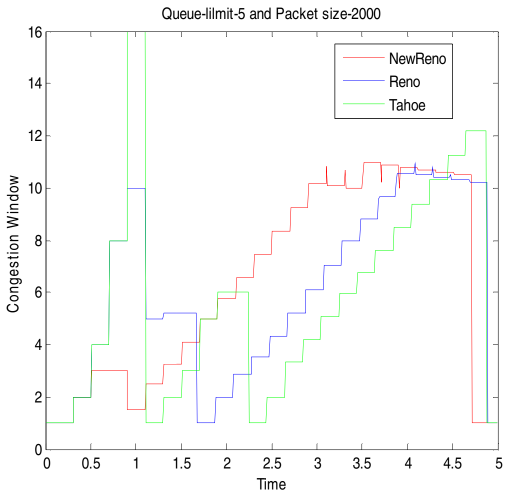

Figure 7. Graph of Congestion window v/s Time

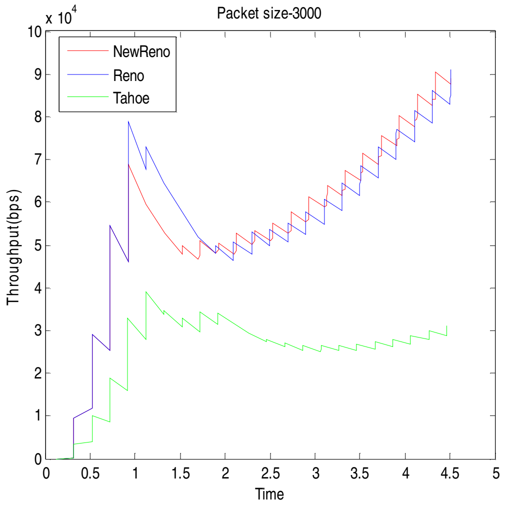

Figure 8. Graph of Throughput v/s Time

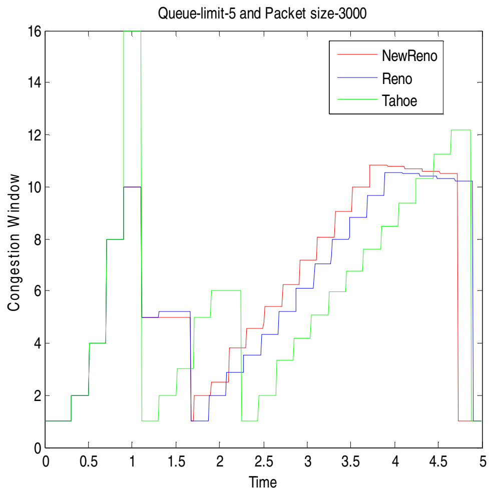

Figure 9. Graph of Congestion window v/s Time

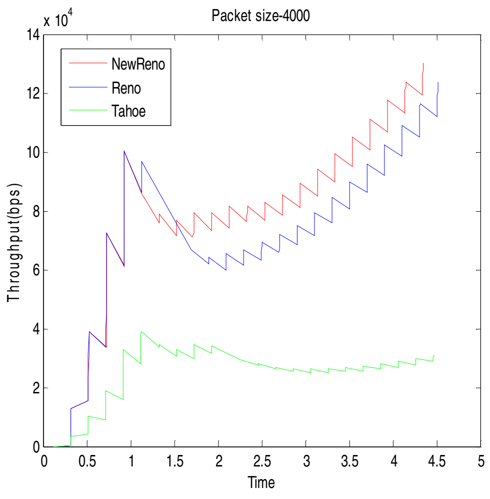

Figure 10. Graph of Throughput v/s Time

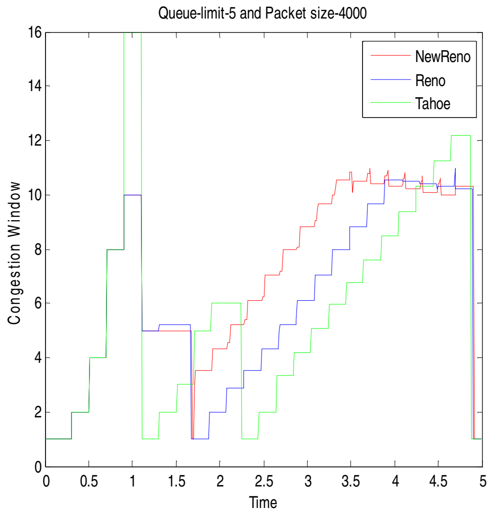

Figure 11. Graph of Congestion window v/s Time

V. Conclusion

From the above experimental results, we conclude that as we increase the packet size of the variant at the source side, throughput of the TCP source variant varies drastically. Throughput of Tahoe decreses upto 82% as we increase the packet size from 1460 to 4000bytes while throughput of Reno and New Reno increase upto 32% and 43% respectively. New Reno uses modified congestion resolution techniques to reduce congestion and gives the better throughput results compared to other two variants. Hence, TCP New Reno performs best among all three variants for wired network in the predefined simulation criteria i.e. Bandwidth of the bottle-neck link (60Mbps), delay time (60ms), maximum value of congestion window(10) and queue-limit(5).

REFERENCES

[1] International Engineering Consortium. The Direction of Optical Networking Market. Available at http://www.iec.orgjonline/tutorials/.

[2] A. K. Turuk and R. Kumar, "A Novel Scheme to Reduce Burst-Loss and Provide QoS in Optical Burst Switching Network". In Proceeding of HiPC-2004, pp. 19-22, 2004.

[3] Defense Advanced Research Projects Agency, Transmission Control Protocol, Arlington, September 1981.

[4] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control", IETF RFC 2581, April 1999.

[5] S. Floyd, T. Henderson and A. Gurtov, "The New Reno Modification to TCP's Fast Recovery Algorithm", IETF RFC 3782, April 2004.

[6] White paper Agilent N2X 2, "TCP and queue Management", © Agilent Technologies, Inc. 2008.