2009 International Symposium on Computing, Communication, and Control (ISCCC 2009)
Proc .of CSIT vol.1 (2011) © (2011) IACSIT Press, Singapore

Congestion Control based analysis of High Delay Tolerant network Transport Protocol

Prof Priyanka Sharma [1] and Dr S N Pradhan [2]

[1 2] CSE department, Institute of technology, Nirma University

Abstract. The current congestion control mechanism used in TCP has difficulty reaching full utilization on high speed links, particularly on wide-area connections. One of reason is that regular TCP lays constraint on the congestion windows in order to keep packet drop rate in control, and the packet drop rate needed to fill a Gigabit pipe using the present TCP protocol is below the currently achievable fiber optic error rates. High Speed TCP and Cubic proposed as modifications of TCP's congestion control mechanism to allow it to achieve reasonable performance in high-speed wide are links. Compound TCP which was proposed as modifications of TCP's Congestion control mechanism, and was implemented on windows operating systems. In this project, simulations results showing the performance of High Speed TCP & Cubic and their impact of its use on the current implementations of TCP are presented. Network conditions including different degrees of congestion, different levels of loss rate and two distinct router queue management policies are simulated. We are also demonstrating that link utilization and performance of protocols clearly affected by the variability of RTT of each flows and fairness of other flows on congestion control are different. The performance and Fairness of High Speed TCP and Cubic are compared to the existing TCP. The project also involves a Linux kernel level implementation of Compound TCP along with already provided CUBIC and High Speed TCP, to provide a concrete base to the observations that were made by simulation results.

Keywords: TCP, congestion window, compound TCP, cubic

1. Introduction

The development of network topology led to the appearance of many high speed networks with bandwidth larger than 1 Gbps or even 10 Gbps. Through High-speed Networks, applications like scientific collaboration, telemedicine and real-time environment monitoring can transfer high bandwidth real time data, images and video captured from remote sensors such as satellite and radars. TCP widely adopted as a data transfer protocol in internet, now works well when transfer rate are in the range of 100 bps to 107 bps and round trip time 1 ms to 10 ms but it performs badly in high-speed networks. TCP increase its congestion window by one at every round trip time and reduces it by half at a loss event. In order for TCP to increase its window for full utilization of 10 Gbps with 1500 bytes packets, it requires over 83333 RTTs. With 100 ms RTT, it takes approximately 1.5 h and for full utilizations in steady state, the loss rate can not be more than 1 loss event per 5 * 109 packets which is less than the theoretical limit of the network's bit error rates.

A number of solutions have been proposed to alleviate the aforementioned problem of TCP by changing its congestion control algorithm: High Speed TCP, BIC and CUBIC, Compound TCP, TCP-Hamilton. These new protocols promise to improve TCP's performance in high speed networks significantly. They are called High Speed TCP Variants because they maintain the overall the functionality of TCP such as connection establishment, connection termination and protocol interface to the application layer even though they modify TCP's congestion control algorithm to improve their performance in high speed networks. So it is required that we should understand that the presence of background traffic affects the performance of congestion control.

The behaviour of protocols in various traffic conditions and on various networks parameters is highly volatile.

2. Implementation

2.1. Upgradation of TCP variants

Many researchers around the world are working on development of efficient algorithms for TCP. As a result many new variants of TCP are being published in research community. Two of them are High Speed TCP(HSTCP) and CUBIC, which also have been implemented in Linux. HSTCP is a modification to TCP's congestion control mechanism for use with TCP connections with large connection windows. To address fundamental limitation of TCP, HSTCP is designed to work like TCP in case of normal congestion window size and change congestion control mechanism to give higher throughput in conditions of nominal loss rates and high congestion windows. High Speed TCP increases its congestion window in segments as follows:

$$\mathbf{W \leftarrow W + (a\,(W) / W)}$$

In response to a congestion event, High Speed TCP decreases as follows:

$$\mathbf{W \leftarrow (1\text{-}b\,(W))\,W}$$

For Standard TCP, a(W) = 1, and b(W) = ½.High Speed TCP uses the same values of a(W) and b(W) for W is less than Low Window. CUBIC is yet another improvement to BIC congestion control protocol. The new features of CUBIC protocol are new window growth function, new TCP friendly mode, low utilization detection. The growth function of CUBIC is:

WCUBIC = C (t – K) 3+ WMAX; where, C is a scaling factor and K= (WMAX*â/C) 1/3

The growth of Window at Wmax is slower and away from Wmax it is faster. Hence this ensures that the window remains around Wmax for longer time unlike other TCP algorithms and hence giving more throughputs while maintaining Fairness. Compound TCP uses a synergy between delay and loss for congestion detection to detect congestion and mitigate the RTT unfairness problem. It has a delay based component which works with the standard loss based protocol. In this variant, a new state variable is introduced which controls this delay based component in CTCP. The conventional congestion window remains as which controls the loss-based component in CTCP. Specifically, the TCP sending window is now calculated as follows:

window = min (cwnd + dwnd, awnd), where, awnd is the advertised window from the receiver.

2.2. Simulation

Ns-2 is a discrete event simulator targeted at networking research. The simulation topology chosen uses only single bottleneck line as shown in Figure 1. This topology will provide excellent platform for studying effects of the interaction between flows in network. All traffic in the simulation passes through the bottleneck link and is received by another node in the network. Most flows in the experiments used TCP, but depending on the experiment, these flows were changed a set of parameters. Flows had the ECN bit set to react to ECN-marked packets to coming from the router queue management.

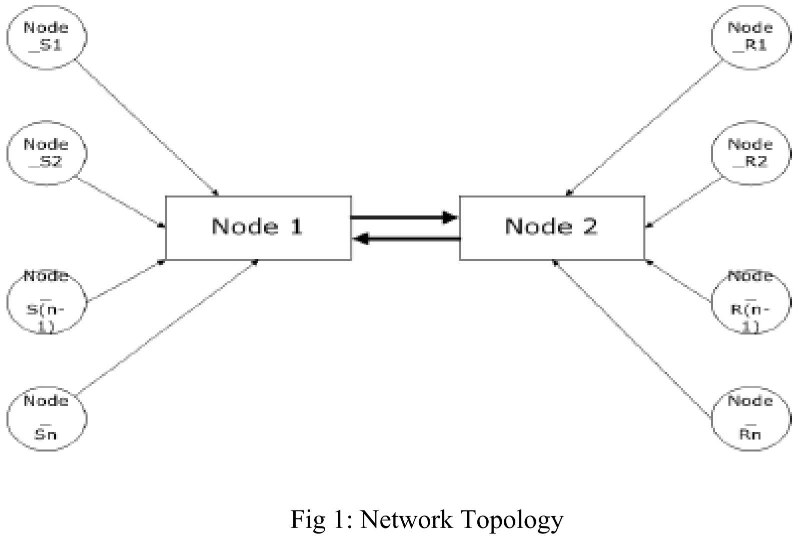

Fig 1: Network Topology

The complete TCP parameter for High Speed TCP and CUBIC are detailed in Table-1.

TCP Parameter	Value of Parameter
ecn_	1
window_	100000
packetSize_	1500
max_ssthresh_	100

TABLE 1: TCP PARAMETER

The ns-2 provides a monitor module to collect data. Two monitors were used in this work, one to monitor the queue in the bottleneck link, tracking the arrival, departure and drop statistics. The other one was used to monitor per flow statistics. There are number of another parameters also used for monitoring data. These parameters should be described as in Table 2. The network conditions will be permitted us to see the variation of the performance of REGTCP and HSTCP flows.

Types	Descriptions
Queue Size	Instant queue size at particular time
Congestion window size	Current window size
Packet arrivals	Total number of packets that have arrived in queue
Packet drops	Number of packets Drops
Packet sent	Number of packet Send
Packet departures	Number of packet departed from the queue
Packets marks	Number of ECN marked packet

TABLE 2: MONITORING STATISTICS DATA

2.3. Kernel Architecture

The kernel module has been developed using Linux kernel 2.6.24.3. The kernel has been configured to execute the modules which is developed to get the values of some parameters of TCP-variants during the experiments. To access the kernel structures and get values, we need to add a hook in the functions that gets called at the time of packet transmission and arrival; we have changed and added the following files for this purpose. The modules added to the Linux kernel are:

1. /sourceroot/include/net/socket_debug.h

2. /sourceroot/net/ipv4/tcp_debug.c   (Inbuilt Module)

3. /sourceroot/net/ipv4/sys_debug.c   (Inbuilt Module)

The modules modified are:

1. /source root/net/ipv4/tcp_ipv4.c

2. /source root/net/ipv4/tcp_output.c

3. /source root/net/socket.c

4. /source root/fs/open.c

5. /source root/fs/read_write.c

6./arch/x86/kernel/syscall_table_32.S

7. /include/asm/unistd_32.h

Compound TCP congestion control algorithm is a patented algorithm and its implementation is not available in Linux. Therefore we have to write compound TCP module in order to analyze its performance. Compound TCP has two components that are used to determine the window size. We also have added the TUBE technology to control gamma dynamically for various types of network.

3. Experiments and Results

The following section will present an account of the various tools and technologies that have been used to conduct experiments and derive results.

3.1. Software Tools

The software tools that have been used in the system on the Linux Platform (Fedora core 7) are NS-2, Traffic Generator Tools like JPERF, MGEN.

3.2. Results from experiments

The TCP-variants and TCP run individually and for comparing fairness, TCP-variants and TCP run together in network in both conditions. Each set of flows runs with RED and Drop Tail queuing policy in the routers. Each simulation runs for 100 seconds and simulation runs several times for getting most accurate results. For real environment testing, we have created the required setup in the lab. The result displays that TCP variants are more aggressive than REGTCP. Performance of protocols does not totally depend on queuing policy, but it also depends on network condition. It can be clearly observed from results that congestion window size of TCP-variants reaches its limit in just few seconds compared to REGTCP which is require a number of seconds because TCP-variants are more aggressive than REGTCP. It is important to point out here, that other restrictions to TCP reaching a high throughout are: limited TCP buffers, reduced network buffer capacity and a large packet loss in slow start. The REGTCP protocol limits its performance by its slow increase rate of congestion window size; this slow response eventually leads other performance problems. The use of REGTCP produces a higher congestion event compare to TCP variants. Another important aspect to observer is that congestion event rate for HSTCP is never lower than 10-6 and second observation is that congestion events in red queuing policy occurs less often than in Drop Tail policy. We can clearly observe that when packets loss rate is higher than 10-5, link can't be fully utilized by HSTCP. HSTCP performance is below or similar to that REGTCP when packet loss rate is higher than 10-5. The important observation of CUBIC is that in case of small number of flows in network, congestion event rate of CUBIC is lower than REGTCP, but in case of higher number of flows in network, congestion event rate of cubic is equal or more than REGTCP. So, we can say that congestion event rate of CUBIC flows can't be controlled meaning that when number of flows increase, the congestion event rate also increase. For RTT Fairness, HSTCP is not completely fair with other HSTCP flows when RTT is different, but CUBIC and CTCP are completely fair with other flows when RTT is different. The reason behind this is that window growth rate is dominated by the elapsed time. TCP fairness deals that with the affair of fairness in terms of bandwidth allocation under certain protocol setting and network conditions. HSTCP and CUBIC are not fair with other protocols in network. In order to address these problems for fair transfer between TCP-variants, it is necessary to hold bandwidth allocation for other TCP-variants. Compound TCP is completely fair with other TCP-variants in network. It is clearly seen in figures.

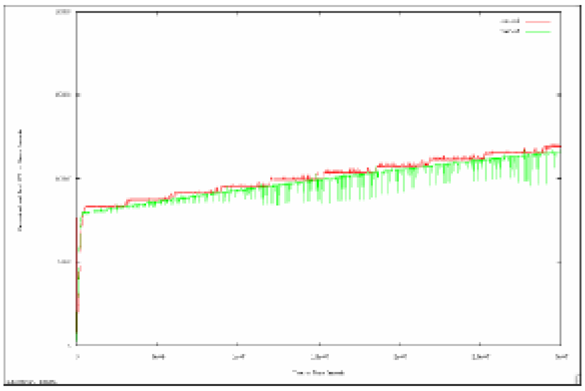

Fig 2: Compound TCP(Smooth) – Homogeneous Flows – RTT(First Flow)

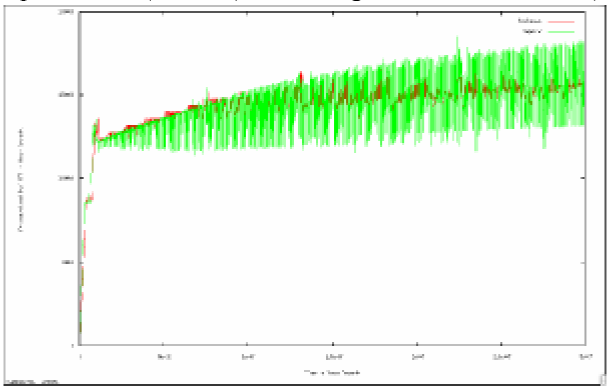

Fig 3: Heterogeneous Flows – High Speed TCP on Background COMPOUND Flows – (TCP - Fairness)

3.3. Conclusion

The work presented in the paper is a combination of both simulations of various variants of TCP and kernel module development to test the same in real environment. The kernel module was developed with Linux kernel 2.6.24.3 and installation took a lot of effort in order to get it running. We designed a tool for our analysis which would give us the results that we wanted. As a collective result of these experiments on 100MBPS real network and Gigabit Network in simulation, we can say that these new congestion control algorithms CUBIC, High speed TCP and Compound TCP are better than Regular TCP algorithms in throughput and bandwidth utilization in heterogeneous and homogeneous network, there are some problems regarding aggressiveness in High Speed TCP and CUBIC like High speed TCP has problem of aggressiveness in TCP and RTT unfairness. Cubic and compound performs fairly well in RTT fairness, but CUBIC still has problem of TCP unfairness. The main reason of TCP unfairness in former two High Speed variants is reactive congestion control mechanisms. The TCP fairness of Compound TCP algorithms is due to its proactive approach where window size is dependent on both congestion and delay. Delay provides measure of congestion in network before it actually happens. So it is able to avoid it in most cases.

4. Acknowledgements

I would like to acknowledge the efforts of my students Parth, Bhautik and Paresh for their contribution towards the work related to this paper.

5. References

[1] Claudia Salzberg Rodriguez, Gordon Fischer and Steven Smolski, The Linux Kernel Primer

[2] Raj Jain, High Speed TCP: Recent Development, Issue and Challenges, Microsoft High-Speed TCP workshop, February 2007.

[3] Thomas Herbert, The Linux TCP/IP stack

[4] Pasi Sarolahti, Mark Allman and Sally Floyd, Evaluating Quick-Start for TCP.

[5] Alex Kesselman and Yishay Mansour, Adaptive AIMD Congestion Control.

[6] Mohamed Tekala and Robert Szaho, Modeling Scalable TCP friendliness to New Reno TCP.

[7] Lan McDonald and Dr. Richard Nelson, Congestion Control Advancement in Linux

[8] Yee-Ting Li, Douglas Leith and Robert Shorten, Experiment Evaluation of TCP Protocols for High Speed Networks

[9] Pradeep Kyasanur, Analysis of TCP-friendly Congestion Control Algorithms

[10] Shirin Ebrahimit- Taghizadeh, Ahmed Helmy and Sandeep Gupta, TCP vs. TCP: A Systematic Study of Adverse Impact of Short-lived TCP flows on Long-Lived TCP flows.

[11] Evandro de Souza, Is High Speed TCP a Way?

[12] Deepak Bansal and Hari Balakrishnan, Binomial Congestion Control Algorithms

[13] Richard J. La, Jean Walrand and Venkat Anantharam, Issues in TCP Vegas

[14] Kun Tan Jingmin Song Qian Zhang, A Compound TCP Approach for High-speed and Long Distance Networks

[15] U. Hengartner1, J. Bolliger1 and Th. Gross, TCP Vegas

[16] Kai Xu, Ye Tian, and Nirwan Ansari, TCP-Jersey for Wireless IP Communications

[17] Wilson Boulevard, RFC793: Transmission Control Protocol J. Li, Y. Jiang, R. Fan. Recognition of Biological Signal Mixed Based on Wavelet Analysis. In: Y. Jiang, et al (eds.). *Proc. of UK-China Sports Engineering Workshop*. Liverpool: World Academic Union. 2007, pp. 1-8. (Use "References" Style)

[18] R. Dewri, and N. Chakraborti. Simulating recrystallization through cellular automata and genetic algorithms. *Modelling Simul. Mater. Sci. Eng*. 2005, **13** (3): 173-183.

[19] A. Gray. *Modern Differential Geometry*. CRE Press, 1998.