Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 10, October 2014, pg.673 – 679

RESEARCH ARTICLE

Analysis and Experimental Evaluation of Transmit Buffer Information Using Cluster Buffer Routing

N.Ramya[1], M.Geetha[2]

[1]Research Scholar

[2]Assistant Professor Department of Computer Science

[1,2]Muthayammal College of Arts & Science, Rasipuram

[1] ramyavell11@gmail.com

[2] mgeethakolanchi@gmail.com

Abstract- Due to the dramatic price drop of memory network devices are configured with much larger buffer sizes than before in order to prevent packet drop. This undermines the assumptions of the TCP congestion avoidance algorithm which depends on packet loss in order to detect and prevent network congestion. We investigate the effects of this dynamic clustering buffer are throughput and latency of buffer collusion. Our simulations show that increasing buffer sizes reduces throughput slightly, but sharply increases latency. When the link is saturated then large buffer sizes cause dramatic reduction of throughput rates for applications which transmit data only periodically. In addition we show that TCP is in fact remarkably resilient to packet drop and can tolerate very small buffer sizes very well at the benefit of optimal latency.

Index Term- TCP congestion avoidance algorithm, clustering

1. INTRODUCTION

Recently a discussion has been started on performance problems of the Internet [Getty][Cring]. The authors observe poor throughput and high latency when they use high-bandwidth applications from home to servers located on the Internet. Their reasoning is that due to the dramatic price drop of memory network equipment is routinely configured with very large memory buffers. One effect of this is that queues can now grow very long causing high latency. Another effect is that TCP algorithms may no longer work appropriately, causing or aggravating network

congestion. The authors coin the term "Buffer bloat" for this problem. TCP uses a sliding window protocol to reliably transfer data from sender to receiver [RFC793]. The receiver indicates to the sender how much data it is willing to receive by communicating the size of a receive window. The sender buffers data until it has received acknowledgment from the receiver. To avoid overloading the network data transmission is further limited by a congestion control algorithm. TCP congestion control works as follows [RFC5681]. It has a slow start method, where the initial rate of sending data is low. It increases the speed of sending data until a packet drop is detected. TCP increases and decreases the speed of data transmission according to the packet loss to get an optimum data rate which does not cause congestion. Thus TCP relies on the timely detection of packet loss in order to prevent congestion and achieve an optimal transmission rate. The amount of unacknowledged data TCP can send I s limited by the receiver window and the congestion window. Clearly there is no justification for the sender to transmit faster than the rate of the slowest link in the connection as this will only cause congestion. The desired maximum for the amount of unacknowledged data for a TCP sender is therefore the product of the rate of the slowest link times the round-trip time. This is commonly called the bandwidth-delay product. The purpose of the congestion avoidance algorithm can therefore be phrased as to detect the current minimum of all maximum possible transmission rates of all links along the connection path. When an outgoing link of a router becomes congested, but nevertheless the router queues newly arriving packets for this link instead of dropping them then TCP cannot properly detect congestion. TCP may even further increase its transmission rate according to the slow start algorithm. In this situation the round-trip time has already increased. This in turn adversely affects latency sensitive applications. When finally the router buffers become exhausted the router has to drop packets. At this point TCP will detect acknowledgment timeouts and start retransmissions.

When there is a sudden increase in round-trip time TCP will detect this as an acknowledgement timeout, conclude there is network congestion and reduce its transmission rate. However, it is not completely clear how TCP reacts to a situation where a queue of an outgoing link in a router is slowly increasing in length. In such a situation the TCP acknowledgement timeout will not go off. TCP may detect the slow increase in round-trip time, adjust its round-trip estimate and use longer round-trip timeouts. Effectively behaving as if there is no congestion. An interesting idea here is to let TCP tune its window size to approach a minimum round-trip time [CdLnet]. TCP segments are frequently sent in bursts: sequences of segments transmitted with little or no intermediate delay. When such a burst arrives at a bottleneck queuing will result potentially causing congestion. It is known that chances for congestion would be reduced if the sender would pace the transmission at the end-to-end throughput rate using a leaky bucket algorithm. To do this for high transmission rates would incur a lot interrupt overhead [Ant03]. However, the sender could split a large burst into several smaller ones, thus largely avoiding this extra interrupt overhead.

Until now we have only talked about the sender's role in causing or avoiding congestion, but the receiver can also play a role. The receiver advertises a receive window. If the receiver has knowledge about the round-trip time and the bandwidth then it could potentially help prevent the sender from sending at too fast a rate by limiting the size of the receive window. This is what Linux does by means of the tcp_moderate_rcvbuf configuration variable. The receiver could also help reduce the bushiness at the sender by moderating the rate at which it opens the receive window.

2. RELATED WORK

Interactive multimedia applications, such as videoconferencing and IP telephony, are becoming an important component of the Internet. Unlike traditional broadcast networks used for television or cable, the modern Internet is highly dynamical and is characterized with rapidly changing conditions. Applications must use congestion control protocols to react to the dynamics of the Internet, in order to keep the Internet's stability [FF99].

TCP is the de-facto standard transport protocol for bulk data transfer in the Internet; however, it does not work well for interactive multimedia applications. Its retransmissions and drastic rate adjustments could cause significant delays to applications. In recent years, researchers have proposed various TCP-friendly congestion control protocols, such as equation-based congestion control [FHPJ00] and general additive-increase-and-multiplicative-decrease (AIMD) based congestion control [YL00]. These TCP-friendly congestion control protocols have significantly improved the performance of multimedia applications over the Internet [RHE99a], and flows of these protocols interact well with other TCP traffic. However, using TCP-friendly congestion
control reduces but does not remove the oscillations in the transmission rate.

The rate oscillations of congestion control protocols are unavoidable, because of the Internet dynamics and the nature of congestion control algorithms. The Internet dynamic is a result of the huge varieties of its application, their users, and usage patterns [AP99, PKC96]. As a result of these varieties, the Internet bandwidth share of an application keeps varying with time. In addition to the Internet dynamics, congestion control protocols have to probe the bandwidth share for applications, because no information about bandwidth sharing is directly available to congestion control protocols. The process of probing for bandwidth and reacting to observed congestion induce oscillations in the achievable transmission rate, and is an integral part of the nature of all end-to-end congestion management algorithms.

In this paper, we derive the minimal buffering required to smooth the inherent rate oscillations of a congestion control protocol. We assume applications use general AIMD (GAIMD) based congestion control protocols.

- GAIMD congestion control protocols use TCP's AIMD algorithm but with an arbitrary pair of increase/decrease parameters (a,ß).
- Throughout this paper, we use AIMD to indicate a GAIMD-based congestion-controlled flow with (a,ß) as parameters.

For example, TCP's congestion control uses AIMD.

Our result shows that the minimal buffering requirement is proportional to the increment parameter a when the AIMD-based congestion control is TCP-friendly. And more importantly, the buffering requirement increases quadratic ally as the increments of its rate and round-trip-time (RTT). This result indicates that using a small increment parameter can reduce the buffering requirement, but the effect is limited as rate or RTT increases.

The rest of the paper is organized as follows: in Section 2, we describe the architecture of our target application, and how it adapts; in Section 3, we describe the general AIMD algorithm, and the analytical derivation of buffering requirement; in Section 4, we present the experiment architecture and results. Section 5 concludes the paper and describes some future work.

3. Congestion Avoidance Algorithm

The amount of unacknowledged data TCP can send is limited by the receiver window and the congestion window. Clearly there is no justification for the sender to transmit faster than the rate of the slowest link avoidance algorithm can therefore be phrased as to detect the current minimum of all maximum possible transmission rates of all links along the connection path in the connection as this will only cause congestion. The desired maximum for the amount of unacknowledged data for a TCP sender is therefore the product of the rate of the slowest link times the round-trip time. This is commonly called the bandwidth-delay product. The purpose of the congestion when an outgoing link of a router becomes congested, but nevertheless the router queues newly arriving packets for this link instead of dropping them then TCP cannot properly detect congestion. TCP may even further increase its

transmission rate according to the slow start algorithm. In this situation the round-trip time has already increased. This in turn adversely affects latency sensitive applications. When finally the router buffers become exhausted the router has to drop packets. At this point TCP will detect acknowledgment timeouts and start retransmissions. When TCP senders simultaneously start retransmissions this may even add to the congestion [SS97]. When there is a sudden increase in round-trip time TCP will detect this as an acknowledgement timeout, conclude there is network congestion and reduce its transmission rate. However, it is not completely clear how TCP reacts to a situation where a queue of an outgoing link in a router is slowly increasing in length.

- In such a situation the TCP acknowledgement timeout will not go off.
- TCP may detect the slow increase in round-trip time, adjust its round-trip estimate and use longer round-trip timeouts. Effectively behaving as if there is no congestion.
- An interesting idea here is to let TCP tune its window size to approach minimum round-trip time**.**

4. EXPERIMENTAL RESULTS

We set up a network in Omnet++/INET based on the above model. The client and server are connected via two routers as shown in figure 1. The client is connected to the router through a high bandwidth channel. The router to router connection is through a low bandwidth channel to simulate the congestion bottleneck at the first router.

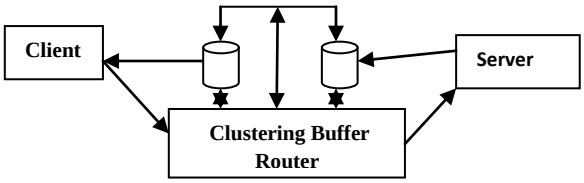

Fig.1: Network Model for simulation

The values for the different network parameters are as follows:

B1 = B3 = 2 Mbps

B2 = 0.5 Mbps

D1 = D3 = 0.002s

D2 = 0.02s

In the rest of the experiment, we have used these values unless otherwise stated. The shortest possible round-trip time of this configuration is 0.024 seconds. In all experiments we used a TCP maximum segment size of 536 bytes. This experiment shows how TCP performance is affected by large buffers. The client runs a TCP application which sends a large amount of data. This application creates heavy traffic in the network and causes congestion at the router. It transfers 1000MB of data to the server. We measured the rate at which the server S is receiving packets from the client. The experiment was run for 1000 seconds. The following table shows the average good put at the server.

Table 1: B2 = 0.5 Mbps

Buffer Size (No. of Frames)	Goodput at Receiver (Bytes/s)	Average RTT(seconds)	Packets dropped
10	54722	0.088	31168
50	53375	0.232	2300
100	53282	0.448	3643
500	53181	1.95	711

The results show a slight decrease of good put with higher buffer size. Remarkable here is that a education of packets dropped doesn't improve the good put, but even reduces it slightly. We repeated this experiment with a different Router1 - Router2 channel speed. The channel bandwidth B2 was changed to 0.1 Mbps. This makes the network more congested.

4.1 Effect of Buffer Size on Latency

In the next experiment, we measured how latency is influenced by different buffer sizes. To measure the latency a client-server pair was added to the network in experiment 1. Client 1 is the same as in the previous experiment which runs the same TCP application. The new client, Client 2 is connected to Router 1. A ping application is running on the client2. This application sends a ping request every 10 ms to the Server 2 which is connected to Router 2. The round-trip time for ping requests, which is the time interval between sending the ping request and receiving the ping reply at the client, has been recorded. It was observed that the RTT is increasing with increase in buffer size.

The ping application can be considered as a representative of an interactive application. The observed denotes the potential latency experienced by an interactive application. This shows how much a latency sensitive application is affected by increase in buffer size.

Fig.2: PingRTT for different Buffer Size

Ping for different Buffer Size

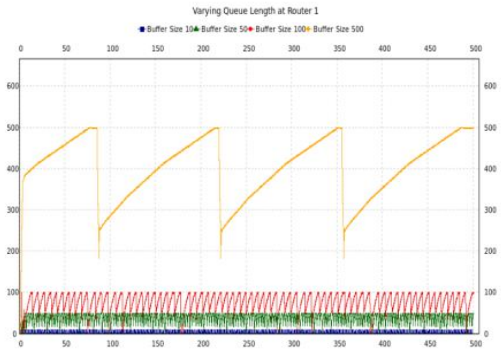

Fig.3: Variation of Queue length over time

4.2 Relation of buffer size, packet drops and Performance.

We want to more closely analyze the effects of buffer size and packet drops on performance. We used the same network model as in experiment 4, but now we fixed the advertised receiver window at the high value of 100 KB and changed the value for the buffer size in the first router in order to observe the average queue length and the number of dropped packets. The buffer size is the maximum number of frames that can be buffered in one direction. Each measurement lasted 180 seconds and transmitted up to 38500 TCP frames from sender to receiver.

Table 5: Relation between buffer size, packet drop and performance

buffer size	throughput Kbit/s	RTT in sec	packet drops	avg.queue length
1	796	0.030	790	0.5
2	880	0.033	725	1.0
3	900	0.035	575	1.5
6	977	0.045	473	3.5
12	984	0.067	261	8
25	982	0.113	132	18
50	979	0.204	100	37
100	984	0.381	31	76

In this table we see that we can severely reduce the router buffer without losing much throughput. There is a throughput-latency optimum between 3 and 6 queue buffers. The large number of packet drops don't affect the throughput much. In the table the highest percentage of dropped sender TCP packets is 2% for a buffer size of 1 frame.

CONCLUSION

We investigate how TCP throughput and latency are affected by router buffer size. Our experiments show that by increasing buffer size throughput is slightly reduced and latency is sharply increased. When the router buffer size increases then one high-throughput application can severely reduce throughput rates for applications which only transmit periodically. Our experiments also show that TCP is remarkably resilient to very small router buffer sizes and to packet drop by routers. By configuring small buffer sizes latency can be substantially improved at a slight cost of throughput. As a corollary of our experiments we show that a receiver can overcome the adverse effects of large router buffers by advertising an optimal receive window. Finding such a value is in principle just as hard as detecting an optimal congestion window size, but the receiver may have additional information, like for instance the speed of its access link or a history of previous connections.

REFERENCES

[1] W. Zhang, M. Branicky, and S. Phillips, "Stability of networked control systems," IEEE Control Systems Magazine.
[2] P. Antsaklis and J. Baillieul, "Special issue on networked control systems," IEEE Transactions on Automatic Control.
[3] F. Lian, F. Moyne, and D. Tillbury, "Performance evaluation of control networks," IEEE Control Systems Magazine, vol. 21, pp. 66–83, 2001.
[4] Z. Jin, S. Waydo, E. Wildanger, and M. Lammers, H. Scholze, P. Foley, D. Held, and R. M. Murray, "Mvwt-ii: The second generation Caltech multi-vehicle wireless testbed." American Control Conference, 2004.
[5] L. Shi, M. Epstein, and R. M. Murray, "Towards robust control over a packet dropping network," in Mathematical Theory of Networks and Systems, 2006.
[6] C. Robinson, G. Baliga, S. Graham, and P. Kumar, "Design patterns for robust and evolvable networked control," in Conference on Systems Engineering Research, 2005.
[7] P. Gupta and P. Kumar, "The capacity of wireless networks," IEEE Transaction on Information Theory, vol. 46, no. 2, pp. 388–404, Mar. 2000.
[8] D. Georgieve and D. M. Tillbury, "Packet-based control: The h2 - optimal solution," Automaitca, vol. 42, pp. 137–144, 2006.
[9] P. A. Kawka and A. G. Alleyne, "Stability and performance of packet- based feedback control over a markov channel," in American Control Conference, 2006.

[10] P. Naghshtabrizi and J. Hespanha, "Anticipative and non-anticipative controller design for network control systems," in Networked Embedded Sensing and Control, P. J. Antsaklis and P. Tabuada, Eds. Berlin: Springer, 2006, vol. 331, pp. 203–218.
[11] A. Bemporad, "Predictive control of teleoperated constrained systems with unbounded communication delays," in Proceedings of the 37th IEEE Conference on Decision and Control , Tampa, FL, USA, Dec.
1998, pp. 2133–2138.
[12] A. Casavola, M. Papini, and G. Franz, "Supervision of networked dynamical systems under coordination constraints," IEEE Transaction on Automatic Control.
[13] B. Mughal, A. Wagan, and H. Hasbullah. E client congestion control in VANET for safety messaging. In Information Technology (ITSim), 2010 International Symposium in, volume 2, pages 654{659. IEEE, 2010.
[14] OMNeT++. Omnet++ network simulation framework. http://www.omnetpp.org/, Feb. 2011.
[15] N. Ramos, D. Panigrahi, and S. Dey. Quality of service provisioning in 802.11 e networks: challenges, approaches, and future directions. Network, IEEE19(4):14{20, 2005.
[16] R. Reinders, E. van Eenennaam, G. Karagiannis, and G. Heijenk. Contention window analysis for beaconing in vanets. In Seventh IEEE International Wireless Communications and Mobile Computing conference, IWCMC 2011, 2011.
[17] D. Rossi, R. Fracchia, and M. Meo. VANETs: Why Use Beaconing at All? In Communications, 2008. ICC'08. IEEE International Conference on, pages 2745{2751. IEEE, 2008.

