Rate Based Congestion Control over High Bandwidth/Delay Links[1]

Yunhong Gu ygu@cs.uic.edu
Xinwei Hong xwhong@lac.uic.edu
Marco Mazzucco marco@dmg.org
Robert Grossman[2] grossman@uic.edu

Laboratory for Advanced Computing, University of Illinois at Chicago
M/C 249, 851 S Morgan St, Chicago, IL 60607

Abstract

As the network bandwidth and delay increase, traditional dominant transfer protocols on the Internet like TCP have shown inefficiency and instability to many applications involved in huge amounts of data transfer. We present in this paper a rate based congestion control algorithm as an application level solution. This algorithm is deployed in our SABUL high performance data transfer protocol, which uses UDP to transfer data and TCP to transfer control information.

SABUL sends data packet strictly based a smooth rate, and this rate is tuned to avoid congestion by the algorithm discussed in this paper. Such a rate based congestion control is TCP friendly and it has better bandwidth utilization than TCP. SABUL is a proposed for huge amounts of data transfer in high bandwidth long delay networks.

Keywords: Congestion Control, Rate Control, SABUL, TCP Friendliness, High Performance Networks

1. Introduction

It has been well known that as bandwidth and delay increase, TCP becomes inefficient and instable. This phenomenon often occurs over today's very popular optical links. For instance, the transfer speed for a TCP connection with default configuration over a spare OC-12/110ms RTT link is only about 5 Mbps (see figure 2). At the same time, more and more applications require data transfer speed in Gbps, such as tele-immersion, distributed scientific computing, etc.

The TCP protocol, designed in those days when the network bandwidth was counted by Kbps, although is still the dominant transfer protocol on the Internet, has exposed its shortcomings over such high speed/long delay networks as optical or satellite links.

The problems are mainly raised from the window based congestion control algorithm. First of all, the AIMD algorithm will waste long time to speed up to fulfill the bandwidth. Second, the TCP throughput is inversely proportional to RTT, which may both lead to efficiency and fairness problem. Connections with large RTT cannot fulfill the available bandwidth over high-speed links; connections with short RTT that share a same segment of the link will obtain more bandwidth resources. Third, the TCP congestion control is not explicitly related to the loss rate, but loss is not a binary variable, and adjusting to the sending rate should depend on the current network situation. Fourth, the AIMD algorithm will halve the data rate when it detects a segment loss, which has been proven to be a catastrophe of the high bandwidth links [25]. Finally, the TCP congestion control is dependent on the numbers of the flows. Suppose there are N connections sharing the same link, all the connections will increase the sending rate by one segment every RTT, so the overall increase of all TCP flows is a function of N.

For many years network researchers have been finding improvements to the TCP efficiency and stability. As the research results, different versions of TCP including Tahoe, Reno, Vegas [10], SACK [20, 21] and Westwood [11] have been put out. These TCP variants brought significant improvements by introducing improved congestion control, selected acknowledgment, fast recovery, etc. However, the original windows

[1] This work is supported by grants from NSF under contract No. ANI-9977868.
[2] Robert Grossman is also with Two Culture, Inc.

based algorithm is kept unchanged, as it is concluded in RFC 2581 [13]. The efficiency and fairness problems caused by such kind of congestion control still exist in the most recent TCP implementation.

ECN [9] introduces new idea to the TCP congestion control by notifying the sender explicit router information so that the congestion control can be more efficient. The XCP [1] expands ECN and further introduces the separation of efficiency control and fairness control.

Although both theoretical analysis and simulation of ECN and XCP give good results, they are not likely to be deployed in the near future since it needs ISPs to upgrade the routers. Moreover, the popularity of optical networks is intending to modify the traditional routing architecture with GMPLS [24]. There is a kind of requirement from the users of the high-speed networks for a pure application level solution.

It is naturally to use UDP with rate based congestion control. UDP provides an IP-like interface to application level with error detection. New protocol can be deployed over it. Moreover, we hope to use the rate based control to eliminate the problems brought by the windows based control.

The application level congestion control algorithm that will be described in this paper distinguished itself from the former work such as TFRC [7] is that it sends one packet every time slot whose length is calculated strictly based on a tuned data sending rate. While those window based protocols, no matter AIMD or equation based control they use, try to throw all the data in the window to the network and wait to next round. Rate based control generates a smooth data flow, similar to the traffic shaping mechanism in ATM [26]. This good character is much better than the bursting flow of TCP.

This rate control algorithm is part of our SABUL high performance transfer protocol [14], so we called it the SABUL congestion control algorithm, or SABUL algorithm. In the rest of the paper, the phrase of "SABUL protocol" can have a broad meaning of any protocol using the SABUL algorithm, or the SABUL protocol itself, according to the context.

The detail of the algorithm will be depicted and analyzed in the rest of the paper. First we will give the requirements, use cases, and suppositions of our new congestion control algorithm in section 2. The detailed algorithm is given in section 3. The following two sections show the theoretical analysis and experimental results. The paper is concluded in section 6 with a brief look to the future work.

2. Description of the Problem

The flow pattern in the environments of high speed/long delay networks is not similar to the conventional public networks. Generally only small numbers of connections with large amounts of data payload share the bandwidth. The available bandwidth may changes rapidly (e.g., when certain data transfer connection enters or leaves), but not very often. Such a scenario is typically in those networks for applications like scientific computing, grid computing, high resolution streaming video, etc. In the public Internet and its backbone, the available bandwidth is only a small portion of the physical bandwidth, and it changes slowly according to the time of the day with small oscillation [19].

The SABUL congestion control algorithm is designed for the former situation to overcome the TCP problems, while we will see it is also applicable to the latter scenario.

The main objectives of the SABUL algorithm are efficiency and fairness. The efficiency objective requires the algorithm can tune the data sending rate to maximum available bandwidth as soon as possible (convergence). The sending rate should then keep its optimal value with small oscillation until the external environments changes (e.g., new connections enter). In addition, reliability is another important requirement. The algorithm should produce acceptable value at any time.

The fairness objective is a necessary condition for the protocol with SABUL algorithm to be used in public networks. The connections using SABUL algorithm should share the bandwidth fairly with other connections using the same or different congestion control algorithms. Obviously here the different algorithm is the TCP congestion control algorithm in most of the cases. However, we argue that sometimes it is impossible to share exactly the same bandwidth between connections using different algorithms. We will give the definition of TCP friendliness in section 4.

Especially, we hope to keep the algorithm independent of network delay, or round trip time (RTT). The unfairness between TCP connections with different RTT is one of the most arguable TCP drawbacks. The sending rate should only depend on the loss rate, or the current available bandwidth, no matter how long the network delay is.

These objectives together with the application-level character cover all the four criteria in [23].

Rate based congestion control try to adjust the interval of the two packets sending event every round. The change of the interval is equivalent to the change of the sending rate. As we have mentioned, this process is quite different with the window based congestion control, which only control the total number of the packets can be sent in one round.

At the application level we regard the network between end points as a black box. We suppose the router will treat the UDP and the TCP flows equally [18]. The RTT is also supposed to be a known value (it can be evaluated and updated during the transfer) and it includes the packet processing time at the end nodes. In addition, we suppose the routers serve the packets in a FCFS (first come, first serve) manner.

The table below lists the symbols will be used in the paper.

Table 1. Symbols used in the algorithm

Symbol Name	Meaning	Comment
γ	Sending rate	Number of packets per second
δ	Packet sending interval	
ρ	Loss rate	
α	Maximum tolerant loss rate	
T	Rate control interval	
V	Valve of the number of loss events before an increase to δ is performed	
L	Number of loss events since last rate tuning	L is cleared after a rate change
B	Available network bandwidth	

3. The SABUL Congestion Control Algorithm

The objectives listed in section 2 are reached in several aspects in the SABUL algorithm, respectively. Such mechanisms as bandwidth probing and fast loss recovery are used to guarantee the convergence requirement. Regular rate tuning is according to the loss rate. The fairness objective is well served in the rate decrease formulae.

In this section we will give the algorithm directly with explanation, implementation, and other issues followed. Before that, we give a brief introduction to the SABUL protocol as the background knowledge.

The SABUL protocol uses UDP to send its data packets and TCP to send feedback in the reverse direction. It is a reliable data protocol. The sequence numbers of lost packets are sent back to the sender as soon as they are detected. And the lost packets are resent as soon as the sender receives the feedback. The acknowledgement of arrived packet are sent back together every fixed time interval or at some special events. The packets arriving rates is feedback every rate control interval. A timer is kept in the sender side to detect expiration event.

3.1 The Rate Control Algorithm

The SABUL algorithm is described in formula (1) to (4):

$$\delta_{n+1} = \delta_n (1 + k_1(\rho - \alpha)) + c, if\ \rho > \alpha \quad (1)$$

$$\delta_{n+1} = \delta_n (1 + k_2(\rho - \alpha)), if\ \rho < \alpha \quad (2)$$

$$\delta_{n+1} = \delta_n + d, if\ \rho = \alpha \tag{3}$$

$$\delta_{n+1} = \delta_n + b, if\ L > V \tag{4}$$

If we translate the formulae to use sending rate instead of sending interval, we get

$$\gamma_{n+1} = \frac{T\lambda_n}{(T + c\gamma_n)(1 + k_1(\rho - \alpha))}, if\ \rho > \alpha \tag{5}$$

$$\gamma_{n+1} = \frac{\lambda_n}{(1 + k_1(\rho - \alpha))}, if\ \rho < \alpha \tag{6}$$

$$\gamma_{n+1} = \frac{T\lambda_n}{T + d\gamma_n}, if\ \rho = \alpha \tag{7}$$

$$\gamma_{n+1} = \frac{T\lambda_n}{T + b\gamma_n}, if\ L > V \tag{8}$$

Note although this is similar to MIMD format, it is NOT. First of all, the decrease formula (5) is non-linear. Moreover, the increase and decrease parameters are according to the loss rate.

We argue that increase only one segment per round is too conservative in high bandwidth long delay networks. In the other hand, a constant large increase (more than one segment) is not proper, since we need to increase the rate faster in a lower speed connection and tune the performance in a finer granularity in a faster connection. We use loss rate in the parameters is also because the loss is not a binary variable. A small loss may be only caused by a busy end system, while a large loss can indicate that new connections come in. Simply dropping the rate with an arbitrary value is not a sound solution.

The same consideration appears in TCP congestion control [13], where it uses *SMSS* * *SMSS* / *cwnd*[3] to increase the window during congestion avoidance.

During the decrease phase we also use similar formulae. The constant *c* used in the formula is to keep faster connection dropping the rate with a larger ratio, by which to guarantee the fairness between different connections.

Formula (3) is to break the equilibrium at $\rho = \alpha$. Equilibrium is violence of fairness. Suppose that two SABUL connections with different initial rate enter the same link and get same loss rate of α, then these unfair situation will be kept until external environments change.

Formula (4) is a fast reaction to the loss. This is very important to both efficiency and fairness.

3.2 Calculation of the Loss Rate

The accuracy of the loss rate is critical to the SABUL algorithm. To get a highly accurate loss rate needs to maintain a record for every sent packet and acknowledgement in that period; however, the cost of this method is too high an overhead to most of the system.

There are two simpler methods to calculate the loss rate. One is to feedback the number of the total received packets (RECV) since last rate control event. The sender then use this received packets number and the total sent packet number (SENT) it records to calculate the loss rate.

$$\rho = (SENT - RECV) / SENT \tag{9}$$

[3] *SMSS* is the size of the segment, and *cwnd* is the current window size.

Formula (9) can generate negative value when received numbers is greater than sent (this is possible since packets sent during previous round can reach in the current round). This formula needs to be enhanced by an exponential smooth moving average function

$$\rho_{n+1} = \rho_n * \text{weight} + (\text{SENT} - \text{RECV}) / \text{SENT} * (1 - \text{weight}) \quad (10)$$

The other method is to count the total numbers of the lost packets (LOSS) from the loss feedback at the sender side. The loss rate is then calculated by

$$\rho = \text{LOSS} / \text{SENT} \quad (11)$$

Both of them have advantages and disadvantages. Formula (10) will lead to a smooth change of the loss rate and help to shape smooth throughput. But (11) is more sensitive to the loss and help to make fast reaction. However, since loss packets in previous round may arrive after the sending rate has been changed, (11) may cause unnecessary rate decrease.

SABUL algorithm uses the second method, which is proved to be better by the experiments.

3.3 Rate Control Interval

In SABUL we use constant control interval T. The reason is to eliminate the bias between connections with different RTT. There are tradeoffs for the constant interval instead of using RTT or a variant interval. Except for the elimination of bias of fairness, constant interval is also easy to implement, especially at the application level, calculation of the RTT may suffer from both the end system utilization (e.g., in a high load system the received packet processing can be delayed) and network rerouting. However, the largest disadvantage of using constant interval is the bias of the loss rate calculation when the interval is less than the RTT.

3.4 Implementation issues

The time unit in the algorithm is sensitive to the performance. In our current implementation we use CPU clock cycle as the minimum unit. It is converted to microseconds as a float number in the formulae. The high accurate timer is not available in most of the regular operating systems today. Implementations can use busy waiting or hardware timer such as Intel APIC onboard timer, which is very popular on Intel architecture motherboards.

High precise timer maybe one of the obstacles that prevent rate based protocol to be implemented in regular operating system in the past, but the hardware today has made it possible.

Finally, we list the constants used in SABUL in table 2.

Table 2: Constants in SABUL Algorithm

T	20000 microseconds	
α	0.001	
k_1	0.125	
k_2	10.0	
c	2.0	Suppose the interval is in the unit of microseconds, otherwise convert the number accordingly
b	0.5	Same as above
d	0.1	Same as above
V	10	

4. Theoretical Analysis

4.1 Efficiency

Without the limit of the window size, SABUL can reach any bandwidth provided by the network. The increase formula (2) can probe the available bandwidth rapidly but still keeps finer granularity when it reaches higher rates (because the interval becomes smaller). Note that the "slow start" mechanism is useless here.

Oscillations still exist in SABUL. However, the oscillation is very small near the steady status, at which point the loss rate is near to zero, and the rate change is very small due to (5) – (8).

Figure 1 shows the theoretical sending interval and sending rate changes of a single SABUL connection.

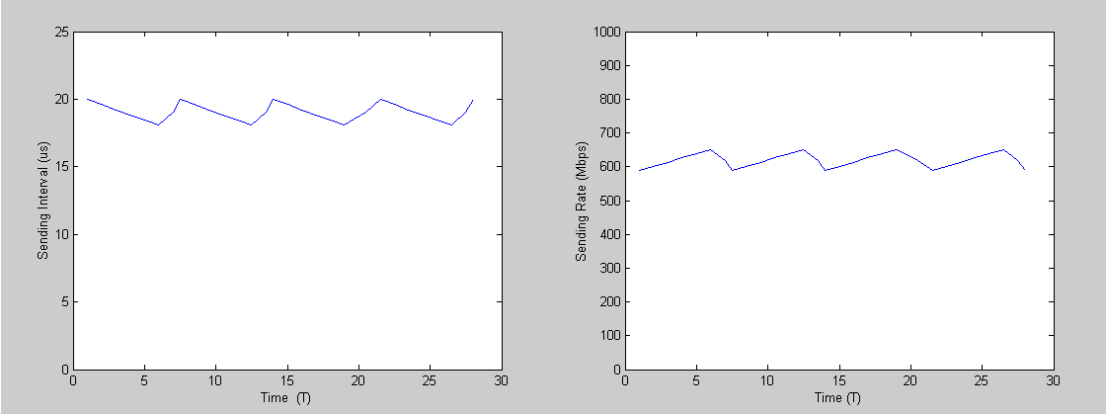

Figure 1. Changes of SABUL Sending Interval and Sending Rate

Suppose there are N intervals for one iterate period. The sending interval and rate change from δ_0 (γ_0) to δ_n (γ_n). The sending rate will keep increasing without loss until it reaches γ_{n-1} according to (2) and generate a loss rate of ρ, then the rate decreases to γ_n according to (1) and (3). From these analysis we have

$$\gamma_n = \gamma_0 \tag{12}$$

$$\delta_n = (\delta_{n-1} + \left\lfloor \frac{\rho B}{V} \right\rfloor b)(1 + k_1(\rho - \alpha)) + c \tag{13}$$

$$\gamma_{n-1} = B\,(1 + \rho) \tag{14}$$

Formula (13) and (14) are approximate formulae, because they suppose that ρ keeps unchanged during the last interval.

From (12, 13, 14) we can calculate the average throughput of SABUL R:

$$R = (\frac{TB(1+\rho)}{TY + (XYb + c)B(1+\rho)} + B(1-\rho))/2 \tag{15}$$

where $X = \left\lfloor \frac{\rho B}{V} \right\rfloor$, and $Y = 1 + k_1(\rho - \alpha)$.

Considering that $\rho \approx 0$, formulae (15) can be further transformed to

$$R = (\frac{TB}{T + cB} + B)/2 \tag{16}$$

Remember that $B = T\,(1 + \rho)/\delta_{n-1}$, which further lead to our conclusion:

SABUL will obtain almost optimal performance (B) provided that

$$\rho \approx 0 \ \textit{and}\ c << \delta \tag{17}$$

It is not surprised that SABUL makes better bandwidth utilization than TCP. In figure 5 (left) two SABUL and two TCP obtain a total average throughput of 699Mbps (191+ 193 + 159 + 156), while four TCP only obtain 538Mbps (141, 131, 135, 132).

4.2 Fairness

SABUL is self-fair, i.e., all the SABUL sections in the same network will share the bandwidth approximately equally independent of the network delay and the initial sending rate.

When a SABUL connection with higher initial rate S1 and another with lower initial rate S2 enter the same link, their bandwidth ratio will keep unchanged if there is no loss (during the increase phase). During the decrease phase, formulae (1) and (5) will keep S1 dropping faster than S2. This will finally lead to a fair bandwidth share between S1 and S2. Moreover, S1 will receive more loss packets than S2, provided all SABUL connections have same loss rate. According to formula (4), S1 will take (4) more than S2, so S1's sending rate drops faster. We use several experiments to validate this informal proof in figure 3.

Because in long delay network, TCP throughput may be limited by its window size, we use the following definition to express the fairly bandwidth sharing between TCP and other protocol, i.e., TCP friendliness.

A protocol P is TCP friendly if the coexisted TCP connections share approximately the same amount of bandwidth as if P is a TCP connection.

If there are M P connections and N TCP connections sharing a link, and each TCP get a bandwidth of B1; on the other hand, if there are M+N TCP connections in the same link, and each TCP get a bandwidth of B2, we say P is TCP friendly if and only if B1 >= B2.

It is NOT sound to use formulae (15) to judge if SABUL is TCP friendly. The reason is that SABUL shapes a smooth flow but TCP produces a bursting flow. When SABUL and TCP share the same link, SABUL will suffer more loss than TCP. TCP try to pour all the packets in its window into the network in an aggressive way, thus the latter arriving SABUL packets will be discarded if the queue of the router is overflow or reach a maximum limit (e.g., in RED queue). When SABUL receives packet loss it decreases the sending rate, while TCP will keep increasing its window until its own packet first gets lost. TCP then halve its window and repeat the increase. This process will lead to a balance between SABUL and TCP.

We use a simple model to examine how the bandwidth is shared between SABUL and TCP. Suppose one SABUL connection with rate R and one TCP connection with window W share a link with bandwidth B. In addition, SABUL and TCP use same packet size, and R, W and B are all counted by number of packets. We suppose the gateway always discards the latest packet.

We label the TCP packet and the SABUL packet with T and S, respectively. At any instant, the source (the node where TCP and SABUL stream are both generated or where they are jointed) can only deliver 1 packet, either T or S. If there is a conflict, each of the packets has a probability of 50% to be delivered. The delayed packet is then delivered in next time slot. The source generates an output sequence in the form of

T, T, S, T, S, … T, T

TCP will reach its highest throughput when the last packet of the sequence is a T; otherwise it keeps increasing W. We now show that the largest number of T in the sequence is 2B/3. Since 50% of the Ts are delayed, so the following equation is satisfied if the last packet is T:

$$W + W/2 = B \tag{18}$$

The TCP throughput than can be obtained by

$$(B/3 + (B/3 + 1) + \ldots + 2B/3) * RTT / ((B/3) * RTT) = B/2 \tag{19}$$

According to formulae (16), the throughput of SABUL is a little lower than the rest B/2, or the TCP throughput. So SABUL is TCP friendly.

An interesting conclusion is that if TCP cannot reach 2B/3 throughput, it will not be affect by SABUL. All these judgments are validated by the experiments in figure 4.

5. Experiments Results

Series of experiments have been done to examine the performance of SABUL algorithm in high speed network testbeds with bandwidth ranges from OC-12 to 10 GigE. These testbeds include StarLight, SARA, OmniNET, and some Intenet2 nodes. Early experiments were also done on NS-2 simulator.

The experiments are focused on the efficiency and fairness features of the algorithm. In the experiments below, physical bandwidth is only a reference and an upper limit of throughput. The actual throughput is also affected by the end system (e.g., CPU, NIC, etc) and the routers (both hardware limit and soft policy of management to different flows).

All TCP experiment results are obtained by Iperf [27] with default configurations. Although modifications to such TCP configuration as the window size can improve the performance, our experiments did not do that because manually tuning is not practical in almost all the applications in real world.

Figure 2 is the testing running single SABUL over spare networks (three charts above). The sending rate shows that the bandwidth utilization of SABUL is independent of the network delay and reaches the transfer speed limit probed from Iperf. The three charts below in Figure 2 are the counterpart experiments of TCP performance in the same network. The TCP performance decreases rapidly as the network delay increases, which is well known today.

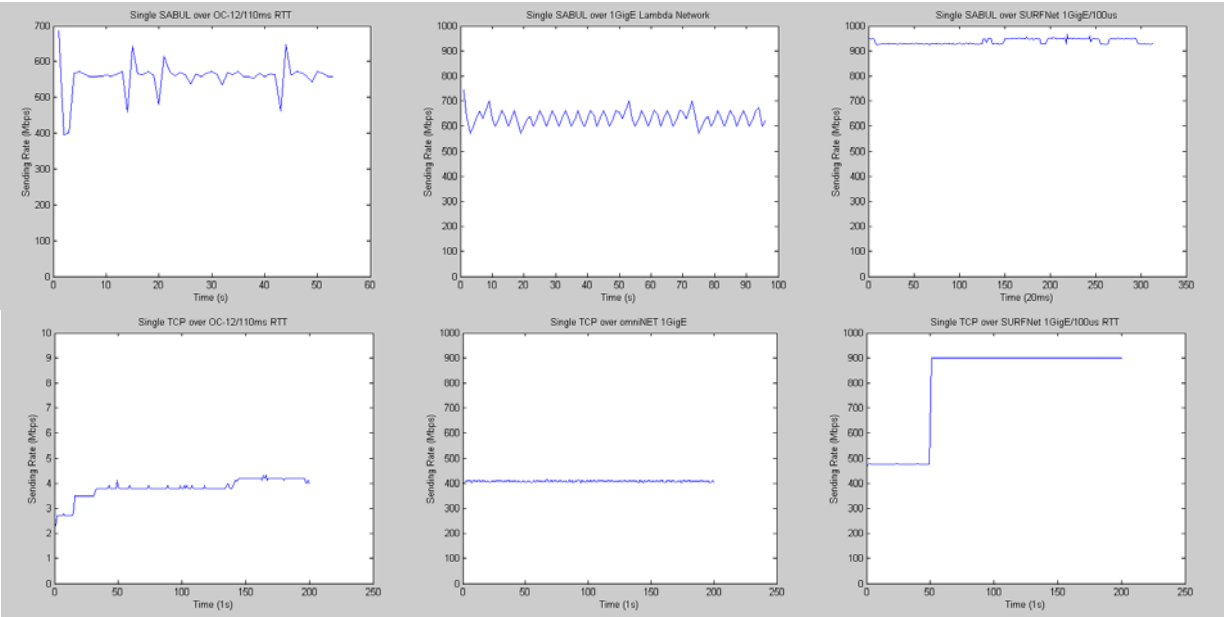

Figure 2: SABUL Efficiency Experiments

Figure 3 shows the efficiency and fairness for multiple SABUL connections. The efficiency and fairness objectives are well kept in this scenario. In the two charts above both of the connection have same initial rate, while in the two below, they have different initial rate, and in the right chart the two connections send data in opposite directions. These experiments were running over omniNET, SARA and StarLight, receptively.

Figure 4 shows the testing of the TCP friendliness of the SABUL algorithm. We run two SABUL connections and two TCP connections on both short RTT (100us) and long RTT (100ms) links. The 4-TCP connection experiments are the counterpart in the same network configuration. In the left two charts, the average throughputs of two SABUL and two TCP are 159, 156, 191, and 193Mbps respectively, which the four TCP throughputs in the same environment are 141, 131, 135, and 132Mbps respectively. This also validates that SABUL has better bandwidth utilization than TCP. In the two charts right of the figure, the TCP throughput is limited by the long RTT, and it is not affected by SABUL, though the latter reaches an aggregate throughput of 960Mbps. These experiments were done over StarLight and StarLight-SARA receptively.

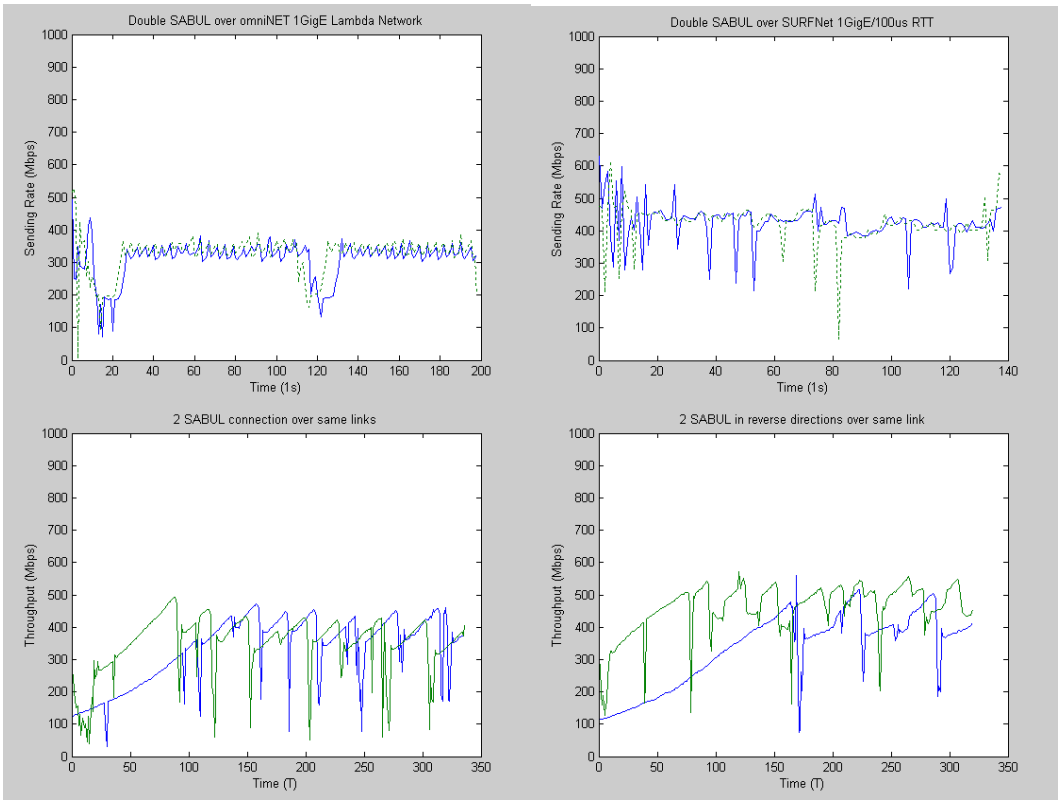

Figure 3: Multiple SABUL Efficiency and Fairness

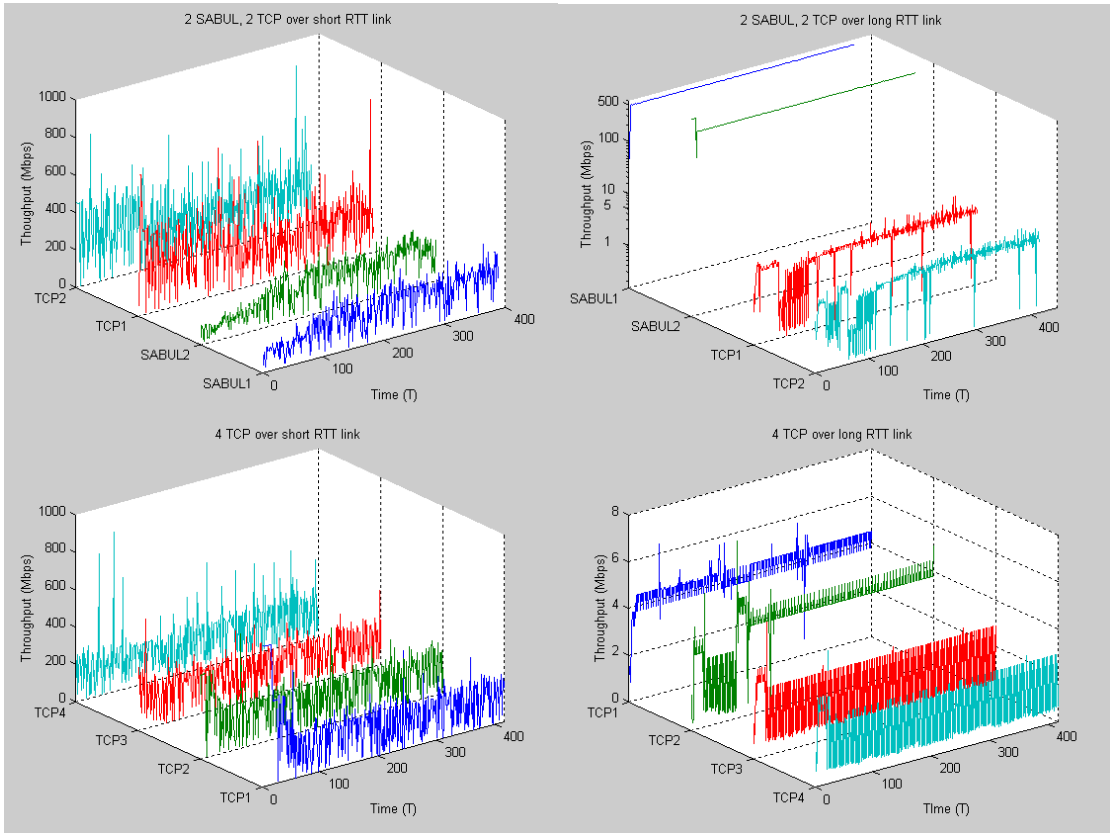

Figure 4: TCP Friendliness

SABUL has been used in several high performance distributed applications, including DSTP [15], Lambda-FTP [16], high performance streaming data merge API [17], and Lambda Mirror (a real-time data replication tool used in distributed database for data protection). These applications are all running on high-speed optical links.

6. Conclusion

By our best knowledge, SABUL is the first released library that uses application level rate based congestion control. We control the interval of the packets sending but not the overall packet numbers. While most of the protocols in TCP/IP networks use the latter method. Rate based control has a good manner of smooth flow shape. It is the elemental reason of why SABUL is friendly to TCP.

We use the loss rate in the rate control functions such that the change of the sending rate can adapt to the network situation in an more accurate way than traditional methods, which regard the network situation as "congestion or not". This leads to a better utilization of the bandwidth. Also, the "slow start" phase is eliminated from our control mechanism. Fast reaction to loss and the fairness between different SABUL connections are well considered in the control algorithm. Finally, another important difference from other algorithms is that SABUL uses constant control interval independent of RTT.

But the most important character of SABUL is that it is designed for high bandwidth networks, where traditional protocols over IP cannot fully utilize the bandwidth. We have deployed SABUL in several practical applications and obtained good performance.

Our next step for SABUL is to migrate to completely lambda switched optical networks, where new technologies of traffic engineering is deployed and will bring series of changes to congestion control.

References:

[1] Dina Katabi, Mark Hardley, Charlie Rohrs: Internet Congestion Control for Future High Bandwidth-Delay Product Environments, SIGCOMM 2002

[2] Milan Vojnovic, Jean-Yves Le Boundec: On the Long-Run Behavior of Equation-Based Rate Control, SIGCOMM 2002

[3] Aditya Akella, Richard Karp, Christos Papadimitriou, Srinivasan Sesham, Scott Shenker: Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP, SIGCOMM 2002

[4] Deepak Bansal, Hari Balakrishnan: Binomial Congestion Control Algorithms, IEEE INFOCOM 2001

[5] S. Fredj, T. Bonald, A. Proutiere, G. Regnie, J. Roberts: Statistical Bandwidth Sharing: A Study of Congestion at Flow Level, SIGCOMM 2001

[6] Deepak Bansak, Hari Balakrishnan, Sally Floyd, Scott Shenker: Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms, SIGCOMM 2001

[7] Sally Floyd, Mark Handley, Jitendra Padhye, Jorg Winmer: Equation-Based Congestion Control for Unicast Applications, SIGCOMM 2001

[8] F. Kelly, A. Maulloo, D. Tan: Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability

[9] K. Pamakrishnan, S. Floyd: Proposal to Add Explicit Congestion Notification (ECN) to IP, RFC 2481, Jan. 1999

[10] Jeonghoon Mo, Richard J. La, Venkat Anantharam, Jean Walrand: Analysis and Comparison of TCP Reno and Vegas, INFOCOM 1999

[11] S. Mascolo, C. Casetti, M. Gerla, S. S. Lee, M. Sanadidi: TCP Westwood: Congestion Control with Faster Recovery, UCLA CSD Technical Report #200017, 2000

[12] Mark Allman, Vern Paxson: On Estimating End-to-End Network Path Properties, SIGCOMM 1999

[13] M. Allman, V. Paxson: TCP Congestion Control, RFC 2581, April 1999

[14] Yunhong Gu, Marco Mazzucco, Xinwei Hong, Robert Grossman: Experiences in Design and Implementation of a High Performance Transfer Protocol, submitted to ICDCS '03.

[15] Yunhong Gu: Lambda-FTP. LAC/NCDM Technical Report.

[16] Robert Grossman, Marco Mazzucco: The DataSpace Transfer Protocol (DSTP): A Data Transport Protocol for Data Webs.

[17] Marco Mazzucco, Asvin Ananthanarayan, Robert Grossman, Jorge Levera, Gokulnath Bhagavantha Rao: Merging Multiple Data Streams on Common Keys over High Performance Networks, SuperComputing 2002, Baltimore, November 2002
[18] P. Pieda, N. Seddigh, B. Nandy: The Dynamics of TCP and UDP Interaction in IP-QoS Differentiated Services Networks, *The 3rd Canadian Conference on Broadband Research*, November 1999.
[19] P. Barford and D. Plonka. Characteristics of Network Traffic Flow Anomalies. In Proceedings of the ACM SIGCOMM Internet Measurement Workshop, Nov. 2001.
[20] Mathis, M., and Mahdavi, J., *Forward Acknowledgement: Refining TCP Congestion Control*, SIGCOMM 96, August 1996
[21] Fall, K., and Floyd, S., *Simulation-based Comparisons of Tahoe, Reno, and SACK TCP*. Computer Communication Review, V. 26 N. 3, July 1996, pp. 5-21
[22] Floyd, S., Jacobson, V., *Random Early Detection gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.
[23] Dah-Ming Chiu, Raj Jain, *Analysis of the Increase and Decrease Algorithm for Congestion Avoidance in Computer Networks,* Journal of Computer Networks and ISDN, Vol. 17, No. 1, June 1989, pp. 1-14.
[24] Ayan Banerjee, Kireeti Kompella, Yakov, et. al., *Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements*, IEEE Communications Magazine, January 2001
[25] V. Jacobson, R. Braden, D. Borman, RFC1323 - TCP Extensions for High Performance, May 1992
[26] Hiroyuki Ohsaki, Masayuki Murata, Hiroshi Suzuki, Chinatsu Ikeda, Hideo Miyahara, *Rate-Based Congestion Control for ATM Networks*, ACM SIGCOMM: Computer Communication Review.
[27] NLANR/DAST, UIUC, *Iperf 1.6: the TCP/UDP Bandwidth Measurement Tool*, Retrieved on October 14, 2002, from http://dast.nlanr.net/Projects/Iperf/.