SABUL: A High Performance Data Transfer Protocol

Yunhong Gu, *Student Member*, Xinwei Hong, *Member*, Marco Mazzucco, and Robert Grossman, *Member, IEEE*

***Abstract* -- The paper describes SABUL, an application level data transfer protocol for data intensive applications over high bandwidth-delay product networks. SABUL is designed for reliability, high performance, fairness, and stability. This uni-directional protocol uses UDP to transfer data and TCP to send back control messages. A rate based congestion control that tunes the inter-packet transmission time helps achieve both efficiency and fairness. To remove the fairness bias between flows with different network delays, SABUL adjusts its rate control at uniform intervals, instead of at intervals determined by round trip time. The protocol has demonstrated its efficiency and fairness features in both experiments and practical applications.**

***Index Terms* -- SABUL, transport protocol, rate control, bandwidth-delay product, high performance data transport**

I. INTRODUCTION AND RELATED WORK

With the rapid increase of network bandwidth and the emergence of new routing/switching technology, data transfer protocols are becoming bottlenecks for many applications. This is particularly common in scientific computing and data intensive grid applications [2].

Although TCP is still dominant in the Internet, the drawbacks inherent in its window based congestion control mechanism prevent its use in high bandwidth-delay product (BDP) environments. The AIMD (additional increase multiplicative decrease) algorithm takes too long a time to discover the available bandwidth [5, 6]. Meanwhile, the link error prohibits TCP from obtaining high throughput [9], especially in wireless networks. In addition, there is fairness bias between TCP flows sharing the same bottleneck with different round trip times (RTTs) [9]. The performance of applications involving multiple TCP streams is sometimes limited by the slowest one [17].

Network researchers have been improving TCP for many years and have published a series of TCP variations, including high speed TCP [14] and scalable TCP [15]. Meanwhile, ECN [12] and XCP [1] have also been put out as open looped congestion control methods. However, these solutions are not expected to be deployed widely in the near future due to the changes that may be required in the infrastructure.

The authors are with the National Center for Data Mining, University of Illinois at Chicago. M/C 249, 715 SEO, 851 S MORGAN ST, CHICAGO, IL, 60607. Robert Grossman is also with the Two Cultures Group. (email: gu@lac.uic.edu, xwhong@lac.uic.edu, marco@dmg.org, grossman@uic.edu)

The work is supported by NSF under grant ANI-9977868.

A solution at the application level is provided by parallel TCP implementations, such as PSockets [7] and GridFTP [8], which obtain high throughput with multiple parallel TCP connections. In practice, however, parallel TCP requires extensive tuning and displays performance shortcomings in lossy, wide area networks [13].

Rate based protocols have been regarded as a better solution for congestion control than window based protocols [3]. This idea can be found in NETBLT [10], VMTP [11], and more recently, Tsunami [16].

This background motivated us to design and develop a high performance application-level reliable data transfer protocol, named SABUL, or simple available bandwidth utilization library. It uses UDP with a rate based congestion control mechanism. In section 2 we will describe the details of the SABUL protocol. The simulation and experimental results will be discussed in section 3. The paper is concluded in section 4 with a brief look at future work.

II. PROTOCOL SPECIFICATIONS

A. Design Rationale

SABUL is designed to be a reliable transfer protocol, which means loss detection and retransmission are needed.

To achieve high performance SABUL must discover available bandwidth and react to congestion as soon as possible. Meanwhile, it should be lightweight with small packet and computation overhead.

Fairness is necessary for SABUL to be accepted on public networks. First, all SABUL flows, independent of initial rates and network delays, should reach similar rates ultimately. Second, SABUL should be TCP friendly. However, this is in conflict with the previous rules since TCP throughput is dependent on RTT and not efficient over high BDP links. We have to make a trade-off: the TCP friendliness rule should be obeyed in small BDP links where TCP can work well; otherwise SABUL can allocate more bandwidth than TCP but should leave acceptable space for TCP to increase. In fact, most times a single TCP flow can only utilize a small portion of the bandwidth over high BDP links [5].

B. General Architecture

SABUL uses two connections: the control connection over TCP and the data connection over UDP. Note that the *data connection* is only a logical abstract, and it is not connected physically since UDP is connectionless. By using TCP in the control connection we want to reduce the complexity of reliability control.

A SABUL session is uni-directional. Data can only be sent from one side (the sender) to the other side (the receiver) over UDP and the control information is only from the receiver to the sender over TCP. The sender initializes the connection, waits for the receiver to connect to it and constructs the

control connection. The data connection is built up following a successful control connection.

C. Packet Formats

There are three kinds of packets in SABUL. The application data is packed in the DATA packet with a 32-bit sequence number. The other two are control reports. ACK packet is positive acknowledgment telling the sender that the receiver has received all the packets prior to the sequence number it carries. NAK packet is negative acknowledgment that carries the number of lost packets and their sequence numbers (loss list). All packets are limited to MTU (maximum transfer unit) size such that they will not be segmented.

D. Data Sending and Receiving

Both the sender and the receiver maintain a list of the lost sequence numbers sorted ascendingly.

The sender always checks the loss list first when it is time to send a packet. If it is not empty, the first packet in the list is resent and removed; otherwise the sender checks if the number of unacknowledged packets exceeds the flow control window size, and if not, it packs a new packet and sends it out. The sender then waits for the next sending time decided by the rate control. The flow window serves to limit the number of packet loss upon congestion, when TCP control reports can be delayed. The maximum window size can be set up by application, which is suggested to be Bandwidth * RTT (use SYN instead of RTT if SYN > RTT).

After each constant synchronization (SYN) time, the sender triggers a rate control event that will update the inter-packet time.

The receiver receives and reorders data packets. The sequence numbers of loss packets are recorded in the loss list and removed when the resent packets are received.

The receiver sends back ACK periodically if there is any newly received packet. The ACK interval is the same as SYN time. The higher the throughput is, the less ACK packets are generated. NAK is sent once loss is detected. The loss will be reported again if the retransmission has not been received after k*RTT, where k is initialized as 2 and is increased by 1 each time the loss is reported. The increase of k is to avoid that the sender is blocked by continuous arrival of loss report. Loss information carried in NAK is compressed, considering that loss is often continuous.

In the worst case, there is 1 ACK for every received DATA packet if the packet arrival interval is not less than the SYN time; there are M/2 NAKs when every other DATA packet gets the loss for every M sent DATA packets.

E. Rate Control

The constant synchronization (SYN) interval in SABUL is 0.01 second. This number is used to reach an acceptable trade-off between efficiency and fairness (both self-fairness and TCP-friendliness), rather than a theoretical value.

Every SYN time, the sender calculates the exponential moving average of the loss rate. If the loss rate is less than a small threshold (0.1%), the number of packets to be sent in the next SYN time is increased by:

$$inc = \max(10^{\lceil \log_{10}(syn/int) \rceil}/1000, 1/MTU)$$

where *inc* is the number of packets to be increased, *syn* is the SYN time (i.e., 0.01), *int* is the current inter-packet time. The inter-packet interval is then recalculated.

The inter-packet time is increased by 1/8 as soon as the sender receives an NAK packet and

1) If the lost sequence number is greater than the largest sent sequence number when last decrease occurs, or
2) If it is the 2^{dec_count}th NAK since last time condition 1) is satisfied, where *dec_count* is set to 4 once 1) is satisfied and increased by 1 each time 2) is satisfied.

The packet sending is frozen (no data is sent out) for a RTT once condition 1) is satisfied.

The objective of the increase formula is to maintain an acceptable bandwidth share between coexisting TCP and SABUL, while keeping a fast bandwidth discovery independent of network delay. However, it causes unfairness between flows with different initial rates. This problem is alleviated by the decrease formula, supposing all flows sharing the same bottleneck link have the same loss rate in the long run. Flows with higher sending rates will decrease more. In addition, setting *dec_count* as 4 after the first decrease favors lower rate flows. The rate control algorithm is basically a combination of AIMD for fairness and stability and MIMD (multiplicative increase multiplicative decrease) for efficiency.

During high congestion, the sending rate can be decreased continuously. Meanwhile, the increase becomes slower as sending rate decreases. The flow window limits the number of unacknowledged packets and the feedback mechanism limits the frequency of control reports. So congestion collapse is avoided.

The initial flow window is 1 packet, and increases its size to the number of acknowledged packets after each received ACK until it reaches the maximum window size or loss occurs. The flow window is set to the maximum value after slow start phase. The initial sending rate can be set up by application, and it does not increase during slow start phase.

III. SIMULATION AND EXPERIMENTS

Both simulation on NS-2 and experiments on real networks were done to examine the efficiency and fairness of SABUL. In all the simulations, the maximum flow window size is large enough so that it doesn't limit the packet sending.

Figure 1 shows how coexisting SABUL and TCP flows share the bandwidth in the simulation. TCP obtains higher bandwidth in low RTT environments, but SABUL still has an acceptable throughput. In high RTT environments, TCP's performance is poor and SABUL obtains much higher bandwidth. These results comply with our design rationale for TCP friendliness.

Simulation to check the fairness between SABUL flows with different initial rates and RTTs is in Figure 2. The unfairness caused by initial sending rates does exist occasionally. However, it has been constrained by the decrease formula to an acceptable bandwidth share. In Figure 2 we found that the RTT bias is almost completely avoided.

At IGrid 2002 (3rd International Grid Conference) we successfully reached about 2.8Gbps through 3 SABUL connections from StarLight (Chicago) to SARA (Amsterdam,

Holland) [4], which has 10Gbps link capacity (the throughput is limited by the 3 pairs of GigE NICs) with 110ms RTT.

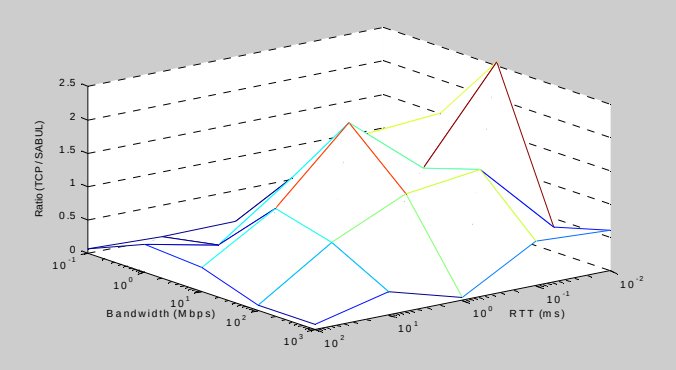

Figure 1. This figure summarizes the results of simulations where TCP and SABUL flows share the same link for various different bandwidth and RTTs. In low BDP environments, TCP obtains more of the available bandwidth. In high BDP environments where TCP is less efficient, SABUL is able to effectively use the available bandwidth.

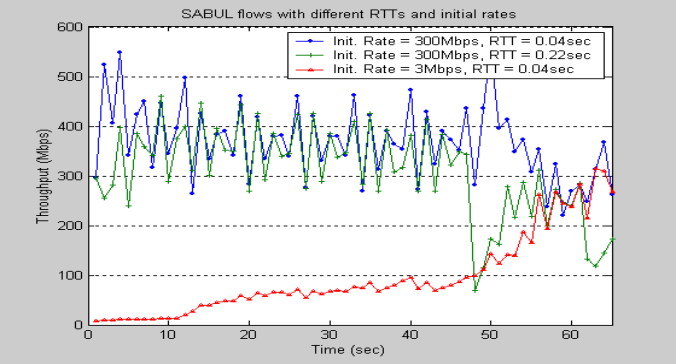

Figure 2. This graph shows the performance of three simulated SABUL flows sharing a 1 Gbps link. The flows have different RTTs and different initial sending rates. The flows all converge at about 280 Mbps, showing that the performance is independent of RTTs and fairly distributes the available 1 Gbps bandwidth.

The SABUL and TCP relationship when sharing the same link in real networks is also examined. In the first experiment, 4 TCP streams and 2 TCP/2 SABUL streams were compared in StarLight Local networks (Table 1), where the link capacity is 1Gbps and the RTT is 0.0004 seconds. Notice that the introduction of SABUL streams doesn't significantly impact the TCP flows, experimentally demonstrate the fairness in small RTT links.

The second experiment shows the performance of a very large number of small TCP flows on a 1 Gbps link connecting Chicago and Amsterdam in the presence of between 0 and 9 SABUL flows (Table 2). The first row shows the number of SABUL flows running, while the second row shows the total bandwidth in Mbps of all 500 TCP flows sharing the link with the indicated number of SABUL flows.

The TCP version used in these experiments is SACK and the buffer size is set to at least bandwidth-delay product.

Table 1. SABUL and TCP coexist on local high speed network

4 TCP flows (Mbps)	226	225	227	225
2 SABUL / 2 TCP (Mbps)	251	237	230 (TCP)	231 (TCP)

Table 2. 500 1MB TCP streams with background SABUL flows

SABUL num.	0	1	2	3	4	5	6	7	8	9
TCP (Mbps)	82	98	78	40	65	37	37	40	36	32

IV. CONCLUSION WITH FUTURE WORK

The objective of SABUL is to provide an application level library for data intensive applications over high performance networks such as computational grids. At the same time, it can coexist with TCP in traditional low BDP environments.

Currently SABUL has been implemented on several platforms and released as an open source project to the public. We have used it in several high performance applications, including high performance FTP, streaming join [17], remote data replication, and striped file transfer.

The possibility of unfairness between SABUL flows still exists, but it is limited to an acceptable ratio. We have not completely removed the effect of network delay. Flows with longer delay react slower to congestion but suffer more loss. These issues will be further examined and solved in our future work.

REFERENCES

[1] D. Katabi, M. Hardley, and C. Rohrs: *Internet Congestion Control for Future High Bandwidth-Delay Product Environments*, ACM SIGCOMM 2002.

[2] I. Foster, C. Kesselman, and S. Tuecke: *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. International J. Supercomputer Applications, 15(3), 2001.

[3] S. Floyd, M. Handley, J. Padhye, and J. Widmer: *Equation-Based Congestion Control for Unicast Applications*, ACM SIGCOMM 2000.

[4] R. L. Grossman, Y. Gu, D. Hanley, X. Hong, D. Lillethun, J. Levera, J. Mambretti, M. Mazzucco, and J. Weinberger, *Experimental Studies Using Photonic Data Services at IGrid 2002*, FGCS, 2003.

[5] Y. Zhang, E. Yan, and S. K. Dao: *A Measurement of TCP over Long-Delay Network*, Proc. of 6th Intl. Conf. on Telecommunication Systems

[6] W. Feng, and P. Tinnakornsrisuphap: *The Failure of TCP in High-Performance Computational Grids*, Supercomputing 2002

[7] H. Sivakumar, S. Bailey, and R. L. Grossman: PSockets: *The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks*. Supercomputing 2000

[8] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke: *Data Management and Transfer in High Performance Computational Grid Environments*. Parallel Computing Journal, Vol. 28 (5), May 2002.

[9] T. V. Lakshman, and U. Madhow: *The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss*. IEEE/ACM Trans. on Networking 5, 3 (1997).

[10] D. Clark, M. Lambert, and L. Zhang: *NETBLT: A high throughput transport protocol*, SIGCOMM '87, (Stowe, VT), pp. 353--359.

[11] D. Cheriton: *VMTP: Versatile Message Transaction Protocol Specification,* RFC1045, April 1993.

[12] Ramakrishnan, K.K., Floyd, S., and Black, D: The Addition of Explicit Congestion Notification (ECN) to IP, RFC 3168, September 2001.

[13] T. Hacker, B. Athey, and B. Noble: *The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network*, IPDPS 2002.

[14] HighSpeed TCP, http://www.icir.org/floyd/hstcp.html, retrieved on 04/03/2003.

[15] Scalable TCP, http://www-lce.eng.cam.ac.uk/~ctk21/scalable/, retrieved on 04.03.2003.

[16] Tsunami, http://www.anml.iu.edu/anmlresearch.html, retrieved on 04/07/2003.

[17] M. Mazzucco, A. Ananthanarayan, R. L. Grossman, J. Levera, and G. B. Rao, Merging Multiple Data Streams on Common Keys over High Performance Networks, Proceedings of SC 02.