UDT: An Application Level Transport Protocol for Grid Computing

Yunhong Gu and Robert L. Grossman[1]
Laboratory for Advanced Computing / National Center for Data Mining
University of Illinois at Chicago
700 SEO, M/C 249, 851 S Morgan St, Chicago, IL 60607, USA

1. Introduction

As network bandwidth and delays increase, TCP becomes inefficient [1, 5, 8, 9]. These problems are due to slow loss-recovery, a RTT bias inherent in its AIMD congestion-control algorithm, and the bursting data flow caused by its window control. Data-intensive applications over high bandwidth delay product (BDP) networks, such as computational grids, need new transport protocols to support them.

To meet this requirement, we have developed an application-level protocol built above UDP, called UDP-based Data Transfer protocol, or UDT. UDT has a congestion control mechanism that maintains efficiency, fairness and stability, and its application-level nature enables it to be deployed at the lowest cost, without any changes in the network infrastructure or operating systems. In experimental studies, UDT has been shown to be effective at moving data over 950 Mb/s between nodes in Chicago and Amsterdam with 1 Gb/s NICs where the RTT is 110 ms.

Related work includes: TCP enhancements such as FAST TCP, HighSpeed TCP and Scalable TCP, UDP bulk-based data transport protocols such as Tsunami and RBUDP, and the open-looped approach of XCP [10].

2. The UDT Protocol

UDT brings reliability and congestion control to UDP. It uses packet-based sequencing (i.e., the sequence number is increased by 1 in MTU size for each packet, including all headers). Selective positive acknowledgement (ACK) is sent at every constant interval, whereas negative acknowledgement (NAK) is generated as soon as packet loss is detected. Congestion control combines rate-based and window-based control mechanisms to ensure efficiency and fairness, as well as TCP friendliness and delay independence.

Rate control tunes the inter-packet time at every constant interval, which is called SYN. The value of SYN is 0.01 seconds, an empirical value reflecting a tradeoff among efficiency, fairness and stability. For every SYN time, when the packet loss rate during the last SYN time is less than a threshold, the maximum possible link Bit Error Rate (BER), the number of packets that will be sent in the next SYN time is increased by:

$$inc = \max(10^{\lceil \log_{10}(B-C)\times MTU \times 8 \rceil} \times \beta / MTU, 1/MTU)$$

Where B is the estimated bandwidth and C is the current sending rate, both in number of packets per second. β is a constant value of 0.0000015. MTU is the maximum transmission unit in bytes, which is the same as the UDT packet size. The inter-packet time is then recalculated using the total estimated number of sent packets during the next SYN time. The estimated bandwidth B is probed by sampling UDT data packet pairs [2, 3].

The inter-packet time is increased by 1/8 (or equivalently, the sending rate is decreased by 1/9) when the sender receives a NAK packet whose largest lost sequence number is greater than the largest sent sequence number when the last decrease occurs, or the number of NAKs since the last rate decrease has exceeded an increasing threshold. No data is sent out during the next SYN time after a rate decrease.

UDT uses a flow-control window to limit the amount of unacknowledged packets. The UDT receiver calculates the packet arrival rate (AS) when it is time to feed back the ACK by using a median filter on the recent packet arrival intervals it recorded. It then attaches AS within the ACK packet. On the sender's side, if an ACK is received and the AS value is greater than 0, the size of the flow window, W, is updated as:

$$W = W * 0.875 + (RTT + SYN) * AS * 0.125$$

[1] Robert L. Grossman is also with Open Data Partners.

Rate control is used to obtain fast bandwidth discovery, fast loss-recovery, and intra-protocol fairness. Flow-control helps to reduce packet loss and oscillation and it helps to avoid congestion collapse [7]. They both contribute to TCP friendliness.

During congestion, loss reports from the receiver can be dropped or delayed. When the sender continues to send new packets, it worsens the congestion. Flow control prevents this from happening. By reducing packet loss, flow control also helps to improve efficiency and fairness. Because a low loss-rate reduces the frequency of sending-rate decreases, it makes UDT less aggressive.

Rate control decides the throughput only when there are a small number of concurrent sources. As RTT and the number of concurrent flows increase, flow control limits the throughput. In such situations, flow control helps to reduce the oscillations that would be higher due to the high rate increase per RTT.

The flow window size starts from 2 as a slow start, and is updated to a number of acknowledged packets when the sender receives an ACK. Slow start ends when either the sender receives a NAK or it reaches the maximum window size, after which the congestion-control algorithm begins working. The inter-packet time is 0 during the slow start phase. It is set to the packet arrival interval when slow start ends.

3. Simulation and Implementation

We have developed a UDT NS-2 simulation module and an open source C++ library that has been posted to several platforms (both available online) [4]. Simulations and experiments on real networks were done to examine efficiency, fairness, and stability features. Meanwhile, the UDT library was used in several practical, data intensive applications within grid-computing environments.

The impact of bandwidth and RTTon UDT is small, since the increase parameter is related to an end-to-end link capacity and because the rate-control interval is independent of RTT. UDT efficiently utilizes available bandwidth even with rapid background changes. This fast adaptation is especially useful for applications involving large amounts of disk IO, whose speed oscillates due to file locating and to disk scheduling.

UDT can reach almost 950 Mbps on a 1 Gbps link over a 110 ms path, from Chicago to Amsterdam. During IGrid 2002, 3 parallel UDT connections on the same link reached 2.8 Gbps [6].

For concurrent UDT flows that share a single bottleneck, max-min fairness can be reached in a short time because, according to the rate control algorithm, slower UDT flows have a higher increase parameter. They also have the same decrease factor. This fairness is still obtained, even if the RTT varies for different flows. In multiple bottleneck topologies, the flow on the narrowest link is guaranteed to obtain at least half of its fair share.

In competition with bulk TCP (Reno) flows, TCP obtains more bandwidth than UDT, in moderate network BDP scenarios. However, its bandwidth share decreases as the BDP increases. Because TCP cannot efficiently utilize the bandwidth in high BDP links, UDT can acquire more bandwidth without impacting other flows. On the other hand, UDT has little impact on TCP flow for traditional web-traffic.

In summary, we expect UDT to be used in the environments where a small number of sources share high bandwidth. We simulated 400 concurrent bulk flows in 1 Gbps link with 1-second RTT link. Traditional, well-behaved TCP flows, such as web traffic, will still perform well in the presence of a UDT flow. However, we have not yet tested large numbers of concurrent UDT flows.

4. Reference

[1] T. V. Lakshman, U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss". IEEE/ACM Trans. on Networking, July 1997, pp. 336-350.
[2] S. Keshav. "A Control-Theoretic Approach to Flow Control". Proceedings of ACM SIGCOMM'91, Zurich, Switzerland, September 1991, pp. 3-15.
[3] M. Allman, V. Paxson. "On Estimating End-to-End Network Path Properties". ACM SIGCOMM, September 1999.
[4] UDT source code. <http://sourceforge.net/projects/dataspace>.
[5] V. Firoiu, J. Padhye, J. Kurose, D. Towsley. "Modeling TCP Throughput: a Simple Model and its Empirical Validation". ACM SIGCOMM, September 1998.
[6] R. L. Grossman, Y. Gu, D. Hanley, X. Hong, J. Levera, D. Lillethun, J. Mambretti, M. Mazzucco, J. Weinberger. "Experimental Studies Using Photonic Data Services at IGrid 2002". FGCS, 2003.

[7] K. Fall S. Floyd. "Promoting the Use of End-to-End Congestion Control in the Internet". IEEE/ACM Transactions on Networking, August 1999.
[8] W. Feng, P. Tinnakornsrisuphap, "The Failure of TCP in High-Performance Computational Grids". Supercomputing 2000.
[9] D. Katabi, M. Hardley, and C. Rohrs, Internet Congestion Control for Future High Bandwidth-Delay Product Environments, In ACM SIGCOMM 2002.
[10] E. He, R. Kettimut, S. Hegde, Y. Gu, M. Welzl, and W. E. Allcock, Survey of Transport Protocols Other Than Standard TCP, GGF White paper Draft, 2003.