Understanding the Performance of TCP and UDP-based Data Transfer Protocols using EMULAB

Sunae Shin Kaustubh Dhondge Baek-Young Choi
University of Missouri – Kansas City
5100 Rockhill Rd. Kansas City, MO, USA
Email: {sshin,kaustubh.dhondge,choiby}@umkc.edu

***Abstract*—In this paper, we present a hands-on course project that explores the performance of data transfer protocols using a GENI resource. TCP is one of the key topics in networking courses, and understanding its behavior as well as limitations, from real experiments, offers an invaluable and deep learning experience. A protocol's performance is directly impacted by network parameters such as network bandwidth, delay and loss. However, it is difficult to control and even vary those parameters, if it is not evaluated with simulations. GENI facilities conveniently provide a virtual laboratory that enables us to control the network settings with real network systems. Through this educational project, students had an opportunity to control important network parameters, and measure and compare TCP's performance with a UDP-based data transfer protocol, UDT, using EMULAB. Students were enthusiastic to witness the protocols' performances, and the limitations of TCP under a high bandwidth delay product network in the presence of packet loss, and to recognize the importance of protocol design and system issues for the future Internet.**

I. Introduction

Understanding TCP behavior is indispensable in a networking course, as TCP is the dominant transport layer protocol in today's Internet, and performs vital network functionalities such as connection-oriented reliable transfer and congestion control.

The best way of learning a protocol behavior is to carry out experiments with real traffic in real network environments. A protocol's performance is directly impacted by network parameters such as network bandwidth, delay and loss. However, it is difficult to control and even vary those parameters, if it is not evaluated with simulations. Fortunately, GENI facilities, such as EMULAB, conveniently provide a virtual laboratory that enables us to control the network settings with real network systems.

In this paper, we present a hands-on educational project development using EMULAB [1]. It is to provide students an opportunity to explore the performance of TCP and to compare it with the one of a UDP-based data transfer protocol, UDT [2], under various network conditions. As UDT takes very distinctive approaches from TCP, it provides students a profound learning opportunity to examine diverse protocol design spectrums. EMULAB is a large public network emulation testbed that allows a controllable environment of network topology, delay and line-speed up to 1 Gbps running at multiple sites around the world, ranging from testbeds with a few nodes up to hundreds of nodes. Using EMULAB, we have varied various network settings including bandwidth, delay, and loss, and measured TCP and UDT's throughput performances. The showcase result has demonstrated that TCP throughput significantly suffers in high bandwidth networks especially with long delays, and the impact of loss was relatively less critical than delay on the TCP throughput. On the other hand, UDT could quickly reach and stay at almost the full network capacity in high bandwidth and delay network settings, but was very sensitive to packet loss. Both of the protocols have permitted fairness among the flows. The real experiments have also exposed a possible system issue that is apart from network or protocol issues, but could affect the performance of data transfer.

The rest of the paper is organized as follows. Section II provides background of the evaluated protocols in the course project. The experiment environments are described in Section III. The evaluation results are discussed in Section IV. Section V concludes the paper.

II. Background

In this section, we describe the background of the evaluated protocols – TCP and UDT.

The Transmission Control Protocol (TCP) is the *de facto* transport protocol of today's Internet for reliable data transfer. It provides connection-oriented reliable data transfer using cumulative ACK based Automatic Repeat reQuest (ARQ). It also performs flow control, and congestion control. Flow control is to avoid the case of a sender's overflowing a receiver's buffer when the receiver is slow in processing its received data to pass onto its application layer. Congestion control is for a sender to find an appropriate sending rate and also to reduce it in response to network congestion. A TCP sender recognizes network congestion based on three duplicate ACKs or timeout events. Congestion control is largely responsible for the achieved rate of throughput. TCP uses a window-based rate control, and the congestion control has different phases such as slow-start and AIMD (Additive Increase and Multiplicative Decrease). In the slow-start phase, it begins with a small value of congestion window size, and the window increases by 1 MSS per acknowledgement until it reaches a threshold. This results in doubling the window size per every RTT. After the congestion threshold, it enters conservative AIMD phase until it experiences network congestion, where the windows size increases by a fraction of an MSS per acknowledgement in order to cause the increase of the window size by only 1 MSS over RTT. When TCP recognizes the network congestion by three duplicate ACKs or timeout events, it drastically reduces the windows size either to 1 MSS or half of the previous window size.

Unfortunately, it is known that TCP does not utilize available network bandwidth well, especially with high bandwidth and delay product settings. Furthermore, the throughput of TCP is significantly impacted by packet loss ([3], [4]). The following formula ([5], [6]) for a simple TCP throughput modeling indicates that its throughput is inversely proportional to both RTT and the square root of loss rate.

$$Throughput_{TCP} \approx \frac{\sqrt{1.5} \cdot MSS}{RTT \cdot \sqrt{lossrate}} \qquad (1)$$

There have been many solutions proposed to improve TCP's congestion control mechanism. For example, Scalable TCP [7], BIC-TCP [8], H-TCP [9], FAST TCP [10] and CUBIC-TCP [11] propose to change the sender's adaptation rate in response to congestion.

On the other hand, DCTCP [12] and XCP [13] require routers' feedback in TCP congestion control, and PSocket [14] suggests using multiple TCP flows to achieve a higher throughput.

A very different approach for reliable data transfer is taken from UDT [2] that is a UDP-based application-level protocol designed for wide area high-speed networks (i.e., high bandwidth and delay product environments). UDT uses UDP at the transport layer in order to transfer bulk data, and provides its own reliability and congestion control mechanisms on top of UDP. It uses time-based selective acknowledgement that generates an acknowledgement at a fixed interval if there are new continuously received data packets. This results in a lower ratio of acknowledgement in a higher speed network, and almost an acknowledgement for each data packet in a low speed network.

UDT uses a mixture of ACK (acknowledgement), ACK2 (a sender's acknowledgement responding to a receiver's acknowledgement), and NAK (negative acknowledgement). ACK2 is generated more sparsely than ACK, and used to reduce repeated ACKs as well as to calculate RTT. As for the congestion control, UDT decreases its sending rate x by a constant factor upon any negative feedback, and increases the rate by $\alpha(x)$, for every rate control interval if there is only positive feedback. $\alpha(x)$ is non-increasing and approaches 0 as x increases using the following formula.

$$\alpha(x) = 10^{\lceil log(L-C(x)) \rceil - \tau} \times \frac{1500}{S} \cdot \frac{1}{SYN} \qquad (2)$$

In the formula, L is the link capacity (bps), $C(x)$ is the current sending rate, S is the packet size (the factor $\frac{1500}{S}$ is to balance the impact of the packet size), and τ is a protocol parameter set as 9. The rationale is that $\alpha(x)$ should be large around its initial value for efficiency, and should decrease quickly to reduce oscillations.

Differences	TCP	UDT
Layer	transport layer	application layer
Feedback	Cum. ACK	NAK, Periodic ACK, ACK2
Congestion control	Window-based	Rate-based

TABLE I
TCP VS. UDT APPROACHES

We choose TCP and UDT protocols for this performance evaluation project, since they take drastically different schemes for data transfer. Table I summarizes the differences between TCP and UDT for data transfer.

III. Evaluation Environments

The evaluations are performed using EMULAB [15]. EMULAB is a public GENI network testbed that allows PC nodes with full root access and a controllable environment of topology, delay and line-speed up to 1 Gbps. EMULAB runs at more than two dozen sites around the world, ranging from testbeds with a few nodes up to hundreds of nodes. Not only for the controllability but also due to its scalable resource, it is well-suited for class students to use at any time.

The network environment settings that we have varied for this project experiments of TCP and UDT data transfer are synopsized in Figure 1. Figure 2 illustrates the network configuration on EMULAB that uses up to three concurrent flows.

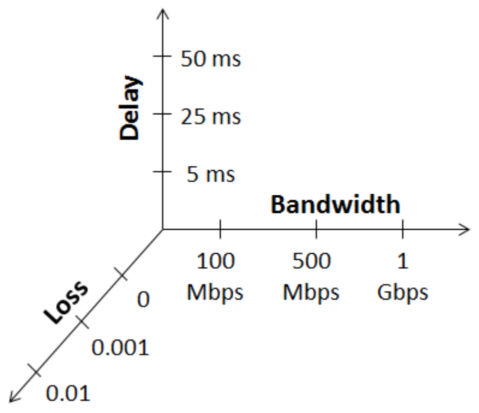


Fig. 1. Varied network environments

IV. Experiment Results

In this section, we discuss the experimental results obtained from the aforementioned settings. For a concise presentation, we show a few representative results out of our extensive experiments.

We first observe the average throughput achieved by TCP and UDT for a variety of conditions of i) No additionally introduced delay or loss, ii) No delay and 0.01 loss, and iii) 50 ms delay and no loss, under different network bandwidths of 100 Mbps, 500 Mbps, and 1 Gbps. The results from TCP and UDT are shown in Figure 3 (a) and (b), respectively. While TCP remains at substantially lower throughput, UDT displays close to the maximum average throughput across the network

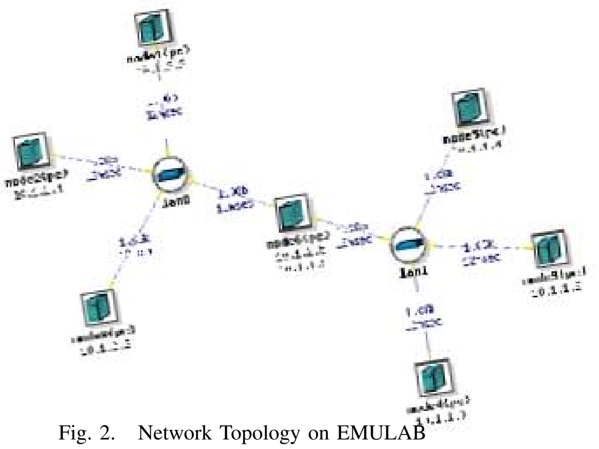


Fig. 2. Network Topology on EMULAB

bandwidths when there is no added delay and loss. Also, the large delay substantially impacts TCP, while the impact on UDT is negligible. Interestingly, Figure 3 (a) shows that on the TCP throughput experience, the impact of delay is greater than the loss, while UDT is affected by loss more significantly than delay since throughput drops more with change of loss than delay as shown in figure 3 (b).

Next, we take a closer look at the throughput using the time series that are depicted in Figures 6 (a) and (b). For these experiments, we measure the throughput at the granularity of 1 second to get a time series of throughput. We observe from Figure 6 (b), that the UDT protocol rapidly utilizes almost all of the available bandwidth and remains at the rate in absence of any loss. However, when a loss factor of 0.01 is introduced, the performance degrades to a large extent. The two points at which the throughput drops can be observed for the 50 ms added delay case, we have encountered 2 and 3 NAKs, respectively, that caused the drop in performance. Even if we do not have injected loss, we may encounter NAKs due to increased delay and timeout value. The small number of 2 and 3 NAKs can decrease throughput notably and this confirms that UDT is sensitive to loss observed in Figure 3 (b).

We now examine the behavior of multiple concurrent TCP and UDT flows, in order to investigate fairness. Figures 4(a) and (b) show the performance of TCP and UDT flows in the presence of cross traffic flows. The experimental setup involved the topology as in Figure 2 with 1 Gbps links and 25 ms delay in them. For both Figure 4 (a) and (b), the Flow 1 starts at time t = 0 sec and goes on for 150 sec. The cross traffic Flow 2 starts at time t = 30 sec and goes on for 90 sec. The last cross flow - Flow 3 starts at time t = 60 sec and goes on for 30 sec. For the TCP based protocol, as shown in Figure 4 (a), since TCP performs poorly and is unable to tap into the available high bandwidth anyway, there is no noticeable effect of cross flows. This leads to the increased total throughput of all the flows. On the other hand, an individual UDT flow well exploits the high bandwidth, and thus, it can be observed in Figure 4 (b) that the occurrence of concurrent flows decreases the throughput of the existing flows in such a manner that their combined bandwidth usage is pretty much equivalent to a single flow.

Finally, we report a system issue on data transfer performance that we have encountered during the experiments. We have found an interesting performance behavior when sending a large file that was different from sending dummy data consecutively. The performance of a large file transfer is shown in Figure 5. For a file transfer, the file has to be loaded into the memory and then written to a socket. Hence, the peak is the buffered file data being sent at high speed by UDT. The trough is the representation if the system is unable to keep up with the high speed protocol resulting in it going idle until the data is available to be sent. This phenomenon is not observed in TCP except with a very large file, because the sending rate itself does not reach high enough speeds. The observation reminds us that the network itself may not be the sole source of limited throughput, and care needs to be taken when transmitting large files.

V. Concluding Remarks

We have presented a course project development that allows students to explore the performance of TCP and to compare it with one from a UDP-based data transfer protocol (UDT) under various network conditions. The protocols' performances were measured under various network settings of bandwidth, delay, and loss, using EMULAB testbed. The showcase result has demonstrated that TCP throughput significantly suffers in high bandwidth networks especially with long delays, and the impact of loss was relatively less critical than the delay on the TCP throughput. On the other hand, UDT could quickly reach and stay at almost the full network capacity in high bandwidth and delay network settings, but was very sensitive to packet loss. Both of the protocols have permitted fairness among the flows. The real experiments have also exposed a possible system issue that is apart from network or protocol issues, but could affect the performance of data transfer. The controllable virtual laboratory environments of GENI have offered students an invaluable and deep learning experience.

References

[1] "EMULAB, a Network Emulation Testbed." [Online]. Available: http://www.emulab.net/

[2] Y. Gu and R. L. Grossman, "Udt: Udp-based data transfer for high-speed wide area networks," *Computer Networks*, vol. 51, no. 7, pp. 1777–1799, 2007.

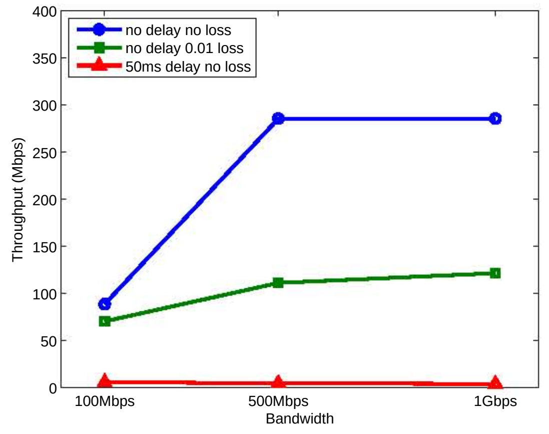
[3] W. Feng and P. Tinnakornsrisuphap, "The failure of tcp in high-performance computational grids," in *ACM/IEEE Supercomputing*, November 2000, p. 37.

[4] T. Faber, A. Falk, J. Bannister, A. Chien, R. Grossman, and J. Leigh, "Transport protocols for high performance: Whither tcp?" *Communications of the Association for Computing Machinery*, vol. 47, no. 11, November 2003.

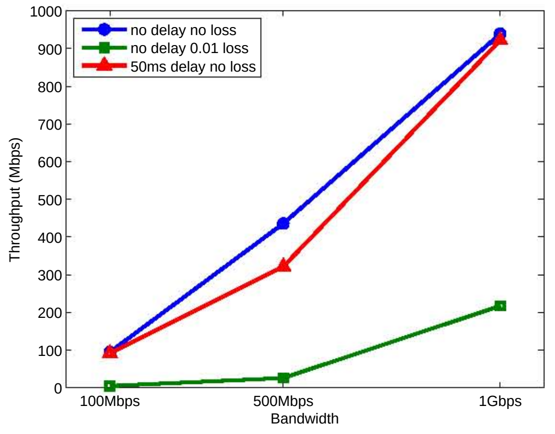
[5] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: a simple model and its empirical validation," in *SIGCOMM*. New York, NY, USA: ACM, 1998, pp. 303–314.

[6] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 67–82, July 1997.

[7] T. Kelly, "Scalable tcp: Improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83–91, 2002.

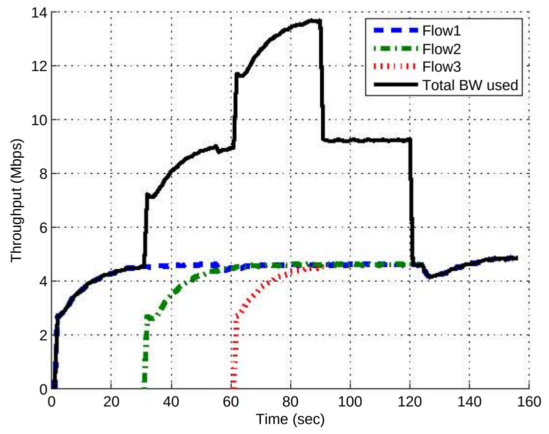


(a) TCP

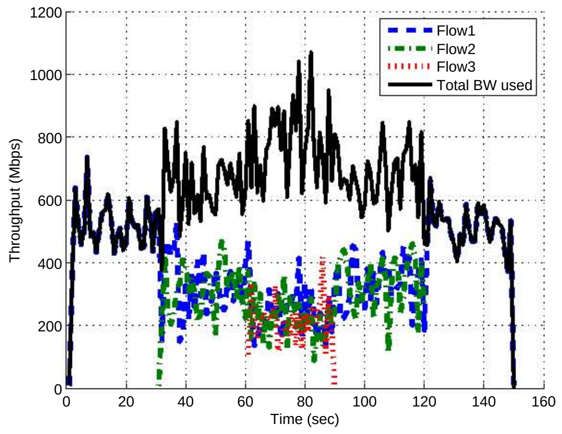


(b) UDT

Fig. 3. Average throughput under varied network bandwidths (no cross traffic)



(a) TCP



(b) UDT

Fig. 4. Fairness of multiple flows (network bandwidth = 1Gbps)

[8] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (bic) for fast long-distance networks," in *INFOCOM*, 2004.

[9] D. Leith and R. Shorten, "H-tcp: Tcp for high-speed and long-distance networks," 2004.

[10] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "Fast tcp: motivation, architecture, algorithms, performance," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 1246–1259, December 2006. [Online]. Available: http://dx.doi.org/10.1109/TNET.2006.886335

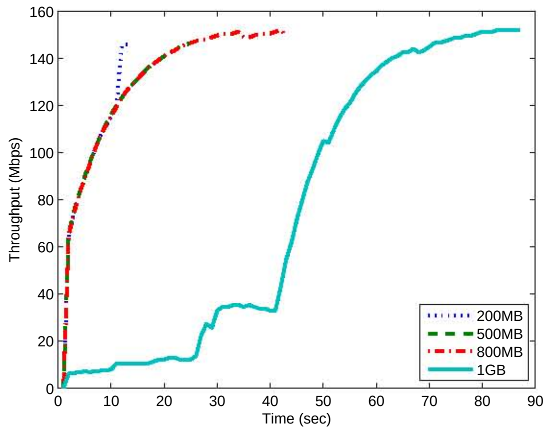
[11] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 64–74, July 2008. [Online]. Available: http://doi.acm.org/10.1145/1400097.1400105

[12] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 63–74.

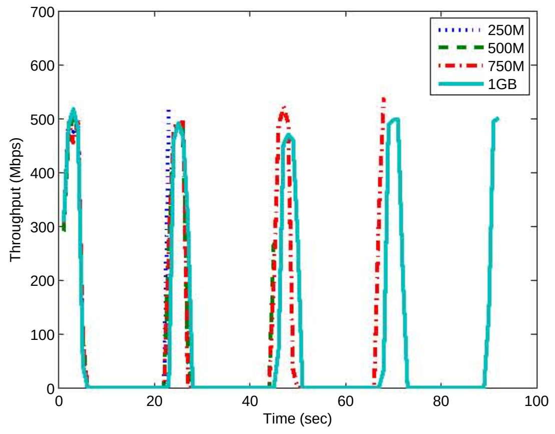
[13] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *SIGCOMM*. New York, NY, USA: ACM, 2002, pp. 89–102.

[14] S. B. Grossman, 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," in *Supercomputing*, 2000.

[15] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," Boston, MA, Dec. 2002, pp. 255–270.

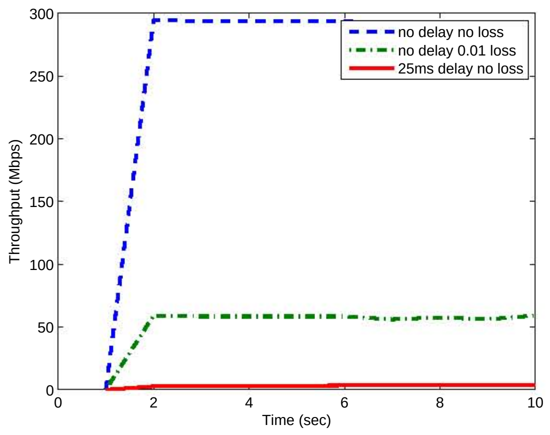


(a) TCP

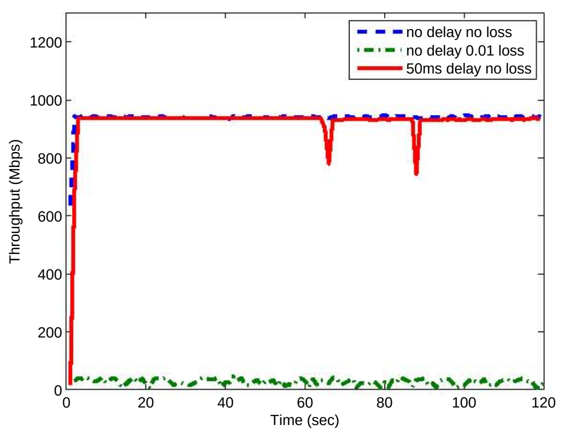


(b) UDT

Fig. 5. Impact of a System Issue of Network Performance (Large File Copy onto Memory on a Sender Throttles Sending Rate)



(a) TCP



(b) UDT

Fig. 6. Time series of throughput (network bandwidth = 1Gbps)