DOE/MICS Final Project Report

September 2004 - September 2006

Project Title: Highly Scalable, UDP-Based Network Transport Protocols for Lambda Grids and 10 GE Routed Networks

PI: Robert Grossman

Co-PI: Stephen Eick

Institution: National Center for Data Mining; University of Illinois at Chicago

Number: Graduate Students: 1

Number of PhD Students: 1

Number of PostDocs: 1

Project Website: www.ncdm.uic.edu

UDT is maintained as an open source Source Forge Project at <http://sourceforge.net/projects/udt>.

Summary of Report

In work prior to this grant, NCDM developed a high performance data transport protocol called SABUL. During this grant, we refined SABUL's functionality, and then extended both the capabilities and functionality and incorporated them into a new protocol called UDP-based Data transport Protocol, or UDT. We also began preliminary work on Composable UDT, a version of UDT that allows the user to choose among different congestion control algorithms and implement the algorithm of his choice at the time he compiles the code. Specifically, we:

- Investigated the theoretical foundations of protocols similar to SABUL and UDT.
- Performed design and development work of UDT, a protocol that uses UDP in both the data and control channels.
- Began design and development work of Composable UDT, a protocol that supports the use of different congestion control algorithms by simply including the appropriate library when compiling the code.
- Performed experimental studies using UDT and Composable UDT using real world applications such as the Sloan Digital Sky Survey (SDSS) astronomical data sets.
- Released several versions of UDT and Composable, the most recent being v3.1.

Detailed Report

Of the many challenges facing scientists and researchers today, one of the most significant is how to transport large amounts of data, which may be distributed, so that it can be mined, analyzed, and visualized at locations remote to where it was collected. As the number of next generation data intensive applications increases, and as the speeds of networks available to researchers increases, the bottleneck in this process is proving to be the performance of current network transport protocols.

Over the past few years, there have been three approaches being used to investigate ways to improve high performance data transport: 1) modifying standard TCP, 2) creating new protocols that are layered over UDP and/or TCP, and 3) developing entirely new network transport layer protocols. NCDM used this grant to investigate the second approach. In work prior to the grant, we developed a new high performance network protocol, called SABUL, which employed UDP in the data channel but continued to use TCP in the control channel. In our research, we demonstrated that this approach was very effective at

transporting data over 1 Gb/s networks and were able to move data consistently at 950 Mb/s and higher using single and aggregated SABUL flows.

Broadly speaking, the objective of this grant was to expand our work with SABUL by refining and extending its capabilities and functionality and developing a new application layer high performance network protocol that could be deployed as an application layer library and that would work effectively on networks with speeds of 10 Gb/s and higher.

More specifically, we had four objectives, which will be discussed below:

1. The first objective was to develop a highly scalable and reliable UDP-based network transport protocols that could be deployed at the application layer.
2. The second objective was to develop a framework for composable network protocols that includes traditional TCP, improved TCP, and UDP-based network transport protocols.
3. The third objective was to develop data analysis and data visualization tools for the performance analysis of highly scalable network protocols containing a mixture of high volume data flows (what we call *teraflows*) and commodity data flows.
4. The fourth objective was to perform experimental studies in order to understand the trade-offs between different protocols and different deployment strategies.

UDT

Development of UDT began as an extension of SABUL. Like SABUL, we designed UDT as an application level protocol so that it could be implemented and installed by researchers with no changes to existing network infrastructures.

Unlike SABUL, however, UDT uses UDP in both the data and control channels. Because SABUL used TCP to transfer control packets, it proved inefficient when data was transferred over networks with high bandwidth delay products (BDP). High BDP networks, however, are proving to be the norm as scientists build more and more data intensive applications that rely on data located at distributed locations around the world.

To improve the protocol's efficiency, therefore, UDT uses UDP to transfer both the data and the control packets. While this increases the complexity of the protocol by eliminating the inherent stability and reliability of TCP, experimental studies have shown that the overall increase in transport performance has more than compensated for this.

In addition to efficiency, we designed UDT to ensure that it was both *fair* and *friendly*. By this we mean that we developed UDT so that it was a) *fair*, in the sense that multiple teraflows, each with a different round-trip time, are able to share a single route without adversely affecting each other, and b) *friendly*, in the sense that UDT flows can share a single route with other TCP flows without adversely affecting those TCP flows.

One of the limitations of SABUL was that it was too good at its job and SABUL could starve other flows, such as TCP, within the same network because it is too aggressive. We incorporated a combination of both window control and rate control mechanisms to control the data flows. Rate control, which uses an AIMD control algorithm, is the primary mechanism used to control the packet-sending period. For this, we generalized a class of control algorithms called "AIMD with decreasing increases," and showed that such algorithms are fair and can be efficient. Window control supports this by specifying a dynamic threshold that limits the number of unacknowledged packets.

Fairness and friendliness were improved by implementing a bandwidth estimation technique that uses packet pairs to estimate routes of current bandwidth. UDT sends out a pair of packets back-to-back and at regular intervals and then estimates the current bandwidth by measuring the time interval separating their arrival at the receiving end. This information is then fed into the AIMD rate control algorithm to control the number and frequency of packets released by the sender.

Investigations into Composable UDT

With this success, we then turned our attention to building an open protocol framework that was flexible enough to support traditional TCP, enhanced versions of TCP, and UDT, and which enabled common congestion control and flow control mechanisms to be used by multiple protocols. Our objective was to use this framework to enable new congestion and flow control mechanisms to be used by multiple protocols as they are developed. With this, we began work on Composable UDT.

In preliminary investigations we surveyed all known control events and began the process of designing a framework that would provide the flexibility to handle these various events. Once we understood this, we built the Composable UDT framework so that it supports the use of different congestion control algorithms.

The framework enables alternate congestion control algorithms to be easily developed and implemented. Implementation is as simple as including the appropriate library (or libraries) when compiling the code, plus, the user is able to change the algorithm at runtime. Numerous algorithms have been implemented and tested to date, including but not limited to: UDT-based reliable transport protocols (e.g., RBUDP), TCP and its variants (e.g., TCP NewReno, Vegas, FAST, Scalable, HighSpeed, and BiC), and group-based transport protocols (e.g., GTP and Congestion Manager).

Other improvements to Composable UDT include:

- It now supports protocols that do not necessarily require lossless transport;
- New functionality now supports messaging service with partial data reliability;
- We have enhanced support for firewall trespassing, Ipv6, and the Windows' platform;
- It includes a new API that is similar to WinSock 2, and which is more complete and effective than previous releases;
- We developed several distributed data intensive applications to test UDT's performance in real world applications.

The advantages of the Composable UDT framework include:

- 1) Different congestion control and bandwidth discovery approaches can be easily modified and then, as easily, compared experimentally;
- 2) UDT can be tuned for various environments;
- 3) Application specific protocols using UDP can be developed relatively easily; and
- 4) A framework that provides a testbed where new UDP-based protocols can be rapidly prototyped and tested, based on the UDT core.

With Composable UDT, we have been able to consistently and reliably transport data over wide area networks at speeds approaching transfers between two machines sitting next to each other on the same rack. In the examples that follow, LLPR is the "Long distance-Local Performance Ratio," which is

defined as the ratio of the actual performance over the wide area network to the performance when the same data is transferred between two local machines with same configuration.

Transferred From/To	Average Transfer Speed	LLPR
StarLight to UIC	850 Mb/s	0.99
NASA GSFC to SC06	800 Mb/s	0.83
SC06 to UIC	850 Mb/s	0.99
Chicago to Greenbelt, MD	615Mb/s	0.98
Chicago to Pasadena, CA	550Mb/s	0.83
Tokyo to Caltech/Pasadena, CA	550Mb/s	0.83

During the course of this grant, NCDM released several versions of UDT and Composable UDT, with the most recent being v3.2.

Data Analysis and Visualization Tools

We also developed an application that allows us to visualize and comparatively analyze tests of UDT's fairness and friendliness. Two screen shots from one of these tests are included below. As explained below, they summarize the performance of 10 UDT teraflows and 200 commodity TCP flows. The visualization shows that UDT is both fair to other teraflows and friendly to commodity TCP flows.

There are four windows in Figure 1. Each window represents an experiment that compares the transfer performance between a pair of machines running 50 commodity TCP flows and 1 to 4 UDT teraflows. The small boxes at the top of each window represent TCP throughput. UDT performance is represented by the large box in the middle of each window.

With no flows, all of the boxes are blue, and the size of the box represents the maximum throughput of a flow. For these experiments, we set the TCP congestion windows at 256KB, which means that the theoretical maximum throughput of a TCP flow is about 11 Mb/s (i.e., 256KB/190ms). The maximum throughput of a UDT flow is 1 Gb/s divided by the number of UDT flows between the pair of machines.

As TCP and UDT packets begin to flow, green boxes begin to fill in the center of the original blue boxes. The size of the green boxes represents the real-time throughput obtained by a flow during the experiment, and as the throughput of the flows increases, the size of the green boxes will increase.

Figure 2 shows that the total throughput of 50 TCP flows is 170.09 Mb/s and the total throughput of 4 UDT flows is 762.032 Mb/s. We can see from this that all 4 UDT flows have approximately the same throughput and therefore UDT is fair to other UDT flows. We also see that UDT is fast, with a throughput of 760 Mb/s. We see the same results in any of the other windows.

The visualization also shows that UDT is friendly to commodity TCP flows. While UDT achieved throughputs of over 760 Mb/s it did not aggressively take over the network and choke off all TCP flows. The 50 TCP flows, even with 1, 2, 3, or 4 UDT flows, achieved a total throughput of between 170 Mb/s and 223 Mb/s.

Figure 1: Visualization of the performance of 10 UDT teraflows and 200 commodity TCP flows

Figure 2: Enlarged view of bottom-left window in Figure 1

Experimental Studies

We investigated the similarity characteristics of Composable UDT, i.e., we looked to see if the congestion control protocols have similar performance characteristics as their native implementations. Because TCP is the most widely used and the most typical congestion control protocol, we compared our Composable UDT-based TCP NewReno protocol (CTCP) to the native Linux TCP NewReno implementation.

In our experiments, we started different numbers of concurrent CTCP and TCP flows in each run, and compared their aggregate throughput, intra-protocol fairness index, and stability index. CTCP and TCP showed similar performances in each of these three characteristics.

Other studies have also shown that, by using Composable UDT, less than 10% of the number of lines of codes is required as compared to the native implementations of the same algorithms.

We also examined Composable UDT's CPU usage. Because it is an application level implementation and its congestion control framework introduces further CPU overheads, we wanted to investigate the quantitative details of these overheads in order to do further optimizations. Our studies showed that at the sender side, Composable UDT could require up to 2 times more CPU time as a native implementation (depending on the algorithm used), whereas as at the receiver side, Composable UDT requires no more than 30% additional CPU time.

Publications

1. Yunhong Gu, Xinwei Hong and Robert Grossman, An Analysis of AIMD Algorithms with Decreasing Increases, Proceedings of GridNets 2004, IEEE Press, 2004.
2. Parthasarathy Krishnaswamy, Stephen G. Eick, Robert L Grossman, Visual Browsing of Remote and Distributed Data, Proceedings of InfoVis 2004, IEEE Press, 2004.
3. Robert L. Grossman, Yunhong Gu, Dave Hanley, Xinwei Hong, Dave Lillethun, Jorge Levera, Joe Mambretti, Marco Mazzucco, and Jeremy Weinberger, Photonic Data Services: Integrating Path, Network and Data Services to Support Next Generation Data Mining Applications, Data Mining: Next Generation Challenges and Future Directions, H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, editors, AAAI Press, 2004.

4. Yunhong Gu and Robert L. Grossman, SABUL: A Transport Protocol for Grid Computing, *Journal of Grid Computing*, Volume 1, pages 377-386, 2004
5. H. Sivakumar, R. L. Grossman, M. Mazzucco, Y. Pan, Q. Zhang, Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks, *Journal of Supercomputing*, Volume 34, Number 3, pages 231-242, 2005.
6. Yunhong Gu and Robert L. Grossman, Optimizing UDP-Based Protocol Implementations, *Proceedings of the Third International Workshop on Protocols for Fast Long-Distance Networks PFLDnet 2005*, 2005.
7. Robert L. Grossman, Yunhong Gu, Xinwei Hong, Antony Antony, Johan Blom, Freek Dijkstra, and Cees de Laat, Teraflows over Gigabit WANs with UDT, *Journal of Future Computer Systems*, Elsevier Press, Vol. 21, Issue 4, pp. 501-513, 2005.
8. L. Wilkinson, A. Anand and R. Grossman, Graph-theoretic scagnostics, *Proceedings of the IEEE Information Visualization 2005 (INFOVIS'05)*, pages 157-164.
9. Rajmonda Sulo, Stephen Eick, Robert Grossman, DaVis: A tool for Visualizing Data Quality, *Proceedings of the IEEE Information Visualization 2005 (INFOVIS'05)*.
10. Yunhong Gu and Robert Grossman, Supporting Configurable Congestion Control in Data Transport Services, *ACM/IEEE SC 2005 Conference (SC'05)*, Seattle, WA, Nov. 12 – 18, 2005
11. Robert L. Grossman, Michal Sabala, Javid Alimohideen, Anushka Aanand, John Chaves, John Dillenburg, Steve Eick, Jason Leigh, Peter Nelson, Mike Papka, Doug Rorem, Rick Stevens, Steve Vejckik, Leland Wilkinson, and Pei Zhang, Real Time Change Detection and Alerts from Highway Traffic Data, *ACM/IEEE International Conference for High Performance Computing and Communications (SC '05)*.
12. L. Wilkinson, A. Anand and R. Grossman. High-dimensional Visual Analytics: Interactive Exploration Guided by Pairwise Views of Point Distribution, *IEEE Transactions on Visualization and Computer Graphics*, 2006.
13. Robert L. Grossman, Yunhong Gu, David Handley, and Michal Sabala Joe Mambretti, Alex Szalay and Ani Thakar, Kazumi Kumazoe and Oie Yuji, Minsun Lee, Yoonjoo Kwon, and Woojin Seok, Data Mining Middleware for Wide Area High Performance Networks, *Journal of Future Generation Computer Systems (FGCS)*, Volume 22, Number 8, pages 940-948, 2006
14. Yunhong Gu, Robert L. Grossman, Alex Szalay and Ani Thakar Distributing the Sloan Digital Sky Survey Using UDT and Sector, *Proceedings of e-Science 2006*.
15. Rajmonda Sulo, Anushka Anand, Leland Wilkinson, Robert Grossman, and Stephen Eick, Topographically-Based Real-Time Traffic Anomaly Detection in a Metropolitan Highway System, submitted for publication.

Presentations:

NCDM has given numerous presentations of UDT and demonstrations of data transfer using the UDT transfer protocol. These include:

1. First International Workshop on Networks for Grid Applications (Gridnets 2004) in October, 2004
2. Supercomputing 2004 Conference in November 2004. Pittsburgh, PA.

- High Performance Data Transport Using UDT (SC04)
UDT Fair, Fast & Friendly Demo (SARA → SC04 Booth)
 - 16 Gb/s Memory-to-Memory Data Transfer using UDT using UDT
(SC04 Booth → Tokyo; SC04 Booth → SARA; StarLight → SC04 Booth; SARA → SC04 Booth)
 - Stanford Linear Accelerator Center (SLAC) Testing of UDT Using High Energy Physics Data
 - 1.66 Gb/s Disk-to-Disk transfer of Sloan Digital Sky Survey (SDSS) Data Using UDT
(SC04 Booth → Tokyo; SC04 Booth → SARA)
 - Transporting and Integrating Astronomical Data From The Sloan Digital Sky Survey
3. Third International Workshop on Protocols for Fast Long-Distance Networks in February 2005, Lyon, France
 4. Conference on Knowledge Discovery and Data Mining 2005 (KDD 2005) in August 2005. Chicago, IL
 5. IGrid 2005 Conference in September 2005. San Diego, CA.
 - Data transfer from San Diego to Daejeon, S. Korea: 1,207 Mb/s
 - Data transfer from San Diego to Chicago: 653 Mb/s
 6. ON*VECTOR Photonics Workshop in February 2005. San Diego, California
 7. Supercomputing 2005 Conference in November 2005. Seattle, WA.
 - Data transfer at 25 Gb/s memory-memory on the Teraflow Testbed, and 18 Gb/s streaming data mining throughput.
 8. Supercomputing 2006 Conference in November 2006. Tampa, FL.
 - Bandwidth Challenge data transfer from UIC to booth at SC06:
Average speed of 8 Gb/s over 6 flows. Peak speed of 9.1 Gb/s.

Interactions with Application Communities

- In collaboration with the Sloan Digital Sky Survey (SDSS) Project, UDT has been used to transfer Sloan Digital Sky Survey (SDSS) astronomical data to astronomers in San Diego, CA, USA; Urbana-Champaign, IL, USA; Daejeon, S. Korea, Beijing, China; Tokyo, Japan; Edinburgh, Scotland; Amsterdam, The Netherlands; Garching, Germany; Budapest, Hungary; and Moscow, Russia.
- We developed prototypes of data mining applications to analyze real time data streams such as highway traffic, web traffic, and astronomy instrumental data at line speeds.