Project Title: Enabling Supernova Computations by Integrated Transport and Provisioning Methods Optimized for Dedicated Channels
PI: Malathi Veeraraghavan
Collaborators: ORNL, Dr. Nageswara S. Rao
Institution: University of Virginia
Website: http://cheetah.cs.virginia.edu/DOE
Number of Graduate Students: 1 (funded on this project) + 3 (funded on related NSF grant)
Number of Ph.D. Students: 1 (funded on this project)
Number of PostDoc Fellows: 0 (funded on this project) + 1 (funded on related NSF grant)
Date of report: Sept. 25, 2005

Abstract of the project goals:

This project consists of three tracks: (i) Track 1: transport protocol for high-throughput file transfers on dedicated circuits, (ii) Track 2: peering of the CHEETAH network and UltraScience Net, and (iii) Track 3: designing a connection-oriented internet. The goal of the NSF-sponsored CHEETAH (Circuit-switched High-speed End-to-End Transport Architecture) project is to provide end-host applications access to end-to-end connection-oriented (circuit and virtual-circuit) services, while preserving the connectionless services already available to them through the Internet/Internet2/ESnet. End-to-end circuits on the CHEETAH network consist of Ethernet segments from hosts mapped to wide-area SONET circuits. The goal of this DOE-sponsored project is to develop a transport protocol for file transfers on high-speed dedicated end-to-end circuits, to interconnect the CHEETAH network to the UltraScience Net, and to extend the concepts of creating Ethernet-SONET-Ethernet circuits to more heterogeneous connections, where segments could be MPLS, VLAN based virtual circuits or SONET or WDM circuits.

Description of the major research activities:

For Track 1, we just completed a paper titled “A transport protocol for dedicated circuits” in which we describe a protocol that we call *Circuit-TCP (C-TCP)* [1]. After much experimentation with user-space implementation of transport protocols using UDP sockets, we decided to turn instead to kernel-based TCP implementations. To implement C-TCP in Linux we used the Web100 instrumented TCP stack. The Web100 instrumented stack provides an interface for user space programs to access many of TCP’s internal state variables. The interface also allows some fields (control parameters), in the internal data structure that Linux maintains for each TCP socket, to be set from the user space. We **added 2 control parameters** to the Web100 stack, **modified TCP sender code** to ignore the congestion window *cwnd*, and instead maintain a minimum of a set sending window size (set equal to the bandwidth-delay product) and the receiver’s flow control window, *rwnd*, of unacknowledged data in the network throughout the transfer. Linux uses a slow start like scheme to update *rwnd* too. This makes *rwnd* a bottleneck during the initial part of the transfer and defeats the purpose of the changes made at the sender. Therefore, we **modified the TCP receiver code** to advertise the maximum possible *rwnd* when the socket is being used over a CHEETAH circuit. We tested C-TCP across the CHEETAH network using an end-to-end 1Gbps circuit on a 13-ms round-trip-delay path. Data transfers on the order of a few KB to 100MB will be served much faster with C-TCP than with TCP on a dedicated circuit because of TCP’s Slow Start mechanism. For larger data

transfer sizes, as long as the TCP send and receive buffers are properly sized for the bandwidth-delay product of the path, the utility offered by C-TCP over the dedicated circuit instead of TCP will decrease. We also show with iperf that a sustained data transfer rate is better maintained with C-TCP than TCP. Finally, for disk-to-disk transfers, we show how the disk receive rate can be determined with a disk-write program and then used to set the circuit rate. With C-TCP, as the sender maintains a constant sending rate equal to the disk-write rate, as long as there is no multitasking on the sender or receiver, which will cause the circuit to be under-utilized, delay can be reduced to propagation delay plus transmission delay.

For Track 2, i.e., peering of the CHEETAH network and UltraScience Net, we completed the data-plane connectivity: we connect a GbE port on a CHEETAH Sycamore SN16000 switch at ORNL to a GbE port on the ORNL UltraScience Net Force10 Ethernet switch. On the control-plane, we completed the implementation of distributed GMPLS based signaling [2]. To address the question of how to interwork the distributed immediate-request GMPLS control-plane of CHEETAH with the Phase 1 centralized book-ahead (pre-reserved) control-plane solution of UltraScience Net, we studied the question of which applications require circuits to be pre-reserved versus which should be handled in immediate-request mode. We completed a paper with this analysis [3].

For Track 3, i.e., the design of a connection-oriented internet to complement the existing connectionless Internet, we completed a paper that describes how GMPLS protocols can be used to set up and release (dynamically using distributed control) heterogeneous connections whenever needed [4]. In other words, if an end-to-end path between two hosts traverses two or more of these types of networks: Ethernet VLAN based network, MPLS network, SONET network, WDM network. Inter-area intra-domain and inter-domain scenarios are considered.

Recently completed papers:

[1] A. P. Mudambi, X. Zheng, and M. Veeraraghavan, “A transport protocol for dedicated circuits, submitted to IEEE ICC 2006.

[2] X. Zhu, X. Zheng, M. Veeraraghavan, Z. Li, Q. Song, I. Habib N. S. V. Rao, “Implementation of a GMPLS-based Network with End Host Initiated Signaling,” submitted to IEEE ICC 2006.

[3] M. Veeraraghavan, X. Fang, X. Zheng, “On the suitability of applications for GMPLS networks,” submitted to IEEE ICC 2006.

[4] M. Veeraraghavan, X. Zheng, Z. Huang, “On the use of GMPLS networks to support Grid Computing,” submitted to IEEE Communications Magazine.

The impact to specific DOE science applications:

John Blondin, NCSU, of the Terascale Supernova Initiative (TSI) project, uses the CHEETAH network to move files from ORNL to NCSU.