

Experiences in implementing an experimental wide-area GMPLS network

Xiangfei Zhu, Xuan Zheng, and Malathi Veeraraghavan
University of Virginia

Abstract

In this article, we describe our experiences in implementing an experimental wide-area GMPLS network called CHEETAH (Circuit-Switched End-to-End Transport Architecture). The key concept is to add a complementary end-to-end circuit based service with dynamic call-by-call bandwidth sharing to the connectionless service already available to end hosts via the Internet. The current CHEETAH experimental network consists of off-the-shelf GMPLS-capable SONET switches (with Ethernet interfaces) deployed at three locations, Research Triangle Park, North Carolina, Atlanta, Georgia, and Oak Ridge, Tennessee. We describe our solutions to various problems relating to control-plane design, IP addressing and control-plane security. We designed and implemented a CHEETAH software package to run on Linux end hosts connected to the CHEETAH network. Among other functions, this software package includes an RSVP-TE module to enable end users and applications to dynamically initiate requests for dedicated end-to-end circuits and receive/respond to requests for circuits. We present measurements for typical end-to-end circuit setup delays across this network. For example, end-to-end circuit setup delay from a Linux end host in NC to a host in Atlanta is 166ms.

I. INTRODUCTION

There is a growing interest in experimenting with optical networking technologies to meet the communication needs of applications in various science projects in the fields of high energy and nuclear physics, astrophysics, molecular dynamics, earth science and genomics. To support their research in these various fields, scientists not only require high-speed wide-area connectivity for

rapid movement of massive data sets, but also predictable (rate-/delay-guaranteed) connectivity for remote visualization and remote control of computations and instruments.

There is a recognition in the networking research community that both these requirements (high-speed and predictable service) can be met with Generalized Multi-Protocol Label Switched (GMPLS) networks. GMPLS networks include Time Division Multiplexed (TDM) Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) networks, Wavelength Division Multiplexed (WDM) networks, and Space Division Multiplexed (SDM) networks. Using OC192 (10Gb/s) SONET interfaces or 10GbE (10Gb/s Ethernet) interfaces, several GMPLS testbeds [1]–[7] have been created to meet the high-speed needs of scientific applications. Since SONET/SDH, WDM, and SDM technologies are circuit-switched, these networks also meet the predictable service requirements of scientific applications.

Given that the geographically distributed scientists and resources (computation, storage, visualization, and instruments) are fairly large in numbers, it becomes cost prohibitive to build dedicated networks for each scientific project. Instead, the high-capacity links of these networks need be shared dynamically. A set of control-plane protocols has been defined for GMPLS networks to enable such dynamic sharing of bandwidth on a call-by-call basis [8]. One of these GMPLS control-plane protocols, called signaling protocol, is used to request and release circuits on an as-needed basis. A commonly implemented GMPLS signaling protocol is Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE) [9].

In this paper, we describe an experimental **wide-area GMPLS network** called **CHEETAH** that was deployed to meet the above-described needs of scientific applications [10] [11]. CHEETAH stands for “Circuit-switched High-speed End-to-End Transport ArcHitecture.” The CHEETAH network consists of “Ethernet-SONET circuit-based gateways.” An “Ethernet-SONET circuit-based gateway” is a network node that has Ethernet interface cards and SONET interface cards, and can be programmed to crossconnect any port on an Ethernet interface card to an equivalent-rate time-division multiplexed SONET signal on any port of the SONET interface cards. Technologies to transparently carry Ethernet frames within SONET frames have been defined and implemented in Ethernet-SONET circuit-based gateways. Given that most end host

network interface cards (NICs) are Ethernet based, these circuit-based gateways are critical to realizing our goal of connecting distant end hosts with wide-area dedicated circuits. An end-to-end circuit consists of Ethernet segments at the edges mapped to SONET circuits (carrying the Ethernet signals transparently) through metro-/wide-area segments. We chose this combination of Ethernet and SONET for our experimental testbed to ease technology transfer to production networks in which Ethernet dominates LANs and SONET dominates MANs/WANs.

By using GbE and 10GbE interfaces in hosts and gateways, and SONET based circuit-switched service in the WAN, CHEETAH meets both the high-speed and predictable service requirements of scientific applications. To meet the dynamic bandwidth-sharing requirement, CHEETAH is built with switches and gateways that implement GMPLS control-plane protocols.

We describe **several interesting problems** we encountered in the design and implementation of the CHEETAH network, and our solutions to these problems. Our design philosophy is to select solutions that will allow for the scalability of the CHEETAH network and its interconnection to other GMPLS networks. Our long-term goal is to create a global-scale connection-oriented inter-network in which bandwidth sharing is based on dynamic call-by-call reservations to complement the unreserved bandwidth-sharing service available in today's Internet. These goals lead us to recommend the use of IPv6 static public IP addresses for most of the interfaces in a GMPLS network, the use of Domain Name System (DNS) to perform wide-area IP-to-MAC address resolution, the use of dynamic updates to IP and Address Resolution Protocol (ARP) tables at end hosts after the setup/release of wide-area Ethernet-SONET-Ethernet circuits, the use of the Internet as the control-plane network, and the use of IPsec to secure the control plane.

To bring the benefits of CHEETAH service to scientists without requiring their deliberate participation in the invocation of this service, we implement a software package called **CHEETAH software** for Linux end hosts, and provide a **CHEETAH-API (Application Programming Interface)** for application programmers. For example, by integrating this API into an FTP server, such as `vsftpd` [12], a scientist using `vsftpd` would enjoy the high-speed and predictable CHEETAH service while being unaware that the `vsftpd` program has dynamically invoked the setup of a dedicated end-to-end CHEETAH circuit prior to executing data transfers. The

end-host CHEETAH software package consists of many components, primary among which is an RSVP-TE module. This module initiates the setup and teardown of end-to-end circuits across the CHEETAH network in response to requests from applications.

Finally, we describe **signaling experiments** that we conducted across the CHEETAH network and provide measurements of end-to-end circuit setup delay for different types of circuits. A typical number for a wide-area SONET circuit that traverses two switches is 166ms.

The rest of the paper is organized as follows. Section II reviews GMPLS protocols, and summarizes the CHEETAH concept and related work. Section III describes the wide-area CHEETAH experimental network, including details on IP addressing and control-plane security. We describe the end-host CHEETAH software package in Section IV. Section V describes the signaling performance experiments conducted on the CHEETAH network to obtain call setup delay measurements. We conclude the paper in Section VI.

II. BACKGROUND AND RELATED WORK

A. GMPLS control-plane protocols

There are three major components in GMPLS control-plane protocols: RSVP-TE signaling protocol, Open Shortest Path First - Traffic Engineering (OSPF-TE) routing protocol [13], and Link Management Protocol (LMP) [14]. These three protocols are designed to be implemented in a control processor associated with each network switch to provide corresponding functionality for its switch. Each of these protocols provides an increasing degree of automation, and correspondingly a decreasing dependence upon manual network administration.

The main purpose of the LMP module is to automatically establish and manage the control channels between adjacent nodes, to discover and verify data-plane connectivity, and to correlate data-plane link properties. The purpose of the OSPF-TE routing protocol software module located at a switch is to enable the switch to send topology, reachability and loading conditions of its interfaces to other switches, and receive corresponding information from other switches. The purpose of the RSVP-TE signaling engine at a switch is to manage the bandwidth of all interfaces on the switch, and to program the data-plane switch hardware enabling it to forward

demultiplexed incoming user bits or packets as and when they arrive. A typical RSVP-TE engine implemented in switches executes five steps when it receives a connection setup request (i.e., an RSVP-TE *Path* message): signaling message parsing, route determination, connection admission control, data-plane configuration (i.e., switch fabric configuration), and signaling message construction.

B. CHEETAH Concept

The CHEETAH architecture, first proposed in [10], is illustrated in Fig. 1. End hosts are equipped with two Ethernet NICs. The primary NICs (NIC1) in the end hosts are connected to the public Internet through the usual LAN Ethernet switches and/or IP routers, while the secondary NICs (NIC2) are connected to Ethernet ports on Ethernet-SONET circuit gateways. Ethernet-SONET circuit gateways, in turn, are connected to wide-area SONET circuit-switched networks, in which both circuit gateways and pure SONET switches are equipped with GMPLS protocols to support call-by-call dynamic bandwidth sharing. Dynamic bandwidth sharing through a distributed mechanism, such as the one offered by GMPLS control-plane protocols, is critical to achieving scalability, an important factor for any network as noted in [15]. End-to-end CHEETAH circuits (as shown with dashed lines in Fig. 1) are set up dynamically between end hosts with RSVP-TE signaling messages being processed at each intermediate gateway/switch in a hop-by-hop manner (*Path* messages 1-5 in the forward direction and *Resv* messages 6-10 in the reverse direction).

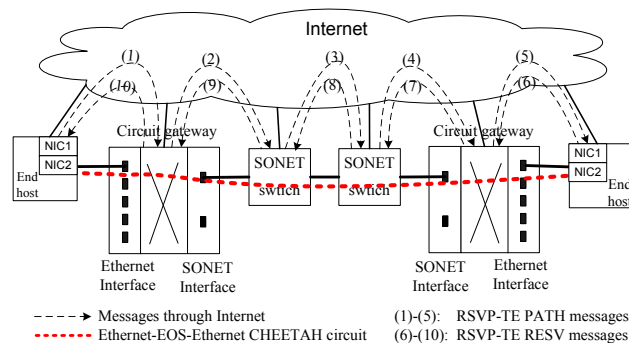


Fig. 1: CHEETAH concept

CHEETAH is designed as an “**add-on**” service to existing Internet connectivity, leveraging the services of the latter. This design brings the following benefits: (i) The connection to the Internet allows a CHEETAH end host to communicate with other non-CHEETAH hosts on the Internet while it is in a communication session with another CHEETAH end host through a dedicated circuit; (ii) Signaling messages can be transmitted through the primary NICs and the Internet eliminating the need for a dedicated signaling channel; (iii) For applications such as file transfers that can use both the Internet and the CHEETAH paths, end-host software can selectively choose to request CHEETAH circuits only when the Internet path is estimated to provide a lower service quality than the CHEETAH circuit, and further “fall back” to the Internet path if the CHEETAH circuit setup attempt fails due to an unavailability of circuit resources on the CHEETAH network.

The trigger for CHEETAH circuit setup is provided by end host applications. If source code is available for an application, we propose modifying it to make use of CHEETAH circuits as and when appropriate. If source code is unavailable, users are provided a program called a `circuitrequestor` to first set up a dedicated host-to-host circuit before initiating the application. As triggers for circuit setup come from applications running on end hosts, automatic large flow detection for the purpose of initiating circuit setup as was proposed for ATM switched virtual circuits in [16] and for circuits in [17] is not required in CHEETAH.

C. Related work

Other optical connection-oriented testbeds being deployed in support of eScience applications include CANARIE’s CA*net 4 [1], OMNInet [2], SURFnet [3], UKLight [4], DOE’s UltraScience net [5], NSF DRAGON [6] and NSF UltraLight [7] networks. These networks differ in their choice of technology, between SONET/SDH, WDM and SDM, and in their control-plane solutions. For example, UltraScience net uses SONET switches, while the DRAGON network uses WDM switches. Similarly, on the control plane, CA*net 4 implements User Controlled LightPath (UCLP) software [18], while DRAGON is based on GMPLS control-plane protocols.

In spite of these differences, the goal of most of the above-mentioned testbeds is to enable the dynamic creation of long-held high-speed circuits that are reserved in advance for specific

scientific projects. These circuits are often used to interconnect routers or Ethernet switches in user/application-specific network topologies. The CHEETAH project differs from these testbeds primarily in its goal to create large-scale networks, and hence emphasizes the bandwidth sharing aspect of GMPLS networks. With end-host CHEETAH software and the CHEETAH-API integrated into end-host applications, the goal is to encourage bandwidth-sharing for short-duration calls that are triggered automatically by application software such as `vsftpd` and web servers. Circuits extend between end hosts rather than IP routers or Ethernet switches, and are established immediately upon request rather than in advance.

III. CHEETAH WIDE-AREA NETWORK

This section describes the wide-area CHEETAH experimental network, and several interesting problems that we encountered relative to the design of the control-plane network, IP addressing, address resolution and control-plane security. We describe our solutions to these various problems.

A. Network switches and hosts

We selected the Sycamore SN16000 Intelligent Optical Switch for the primary network nodes in the CHEETAH network. Our reasons for doing so were **two-fold**. **First**, the SN16000 switch implements a standardized data-plane solution for Ethernet-SONET mapping called Generic Framing Procedure (GFP) [19]. In addition, it implements Virtual Concatenation, which is also a standardized mechanism, to select the SONET rate to be as closely matched to the Ethernet rate as possible [20]. For example, a 1Gb/s Ethernet signal is mapped to a 21-OC-1 virtually concatenated SONET signal instead of an OC-48 (2.5Gb/s) signal, which would have been the appropriate level to use in the original SONET hierarchy.

Second, the Sycamore SN16000 switch has a robust implementation of GMPLS control-plane protocols, RSVP-TE and OSPF-TE, which includes the major GMPLS control-plane-related IETF RFCs, RFC 3473, 3477, 3945, 3946 for RSVP-TE, and RFC 3630 for OSPF-TE. While these specifications specify procedures for homogeneous circuits (e.g., pure-SONET,

where the incoming and outgoing ports used on the circuit are both SONET), they do not support hybrid circuits, i.e., Ethernet-SONET circuits. To establish such circuits, most equipment require administrators to use network management software. Standardization is underway to extend RSVP-TE to handle such hybrid circuits. Meanwhile, Sycamore provided us a version of their control-plane software, which implements a proprietary solution for RSVP-TE control of such hybrid circuits. The built-in RSVP-TE software at the SN16000 switches makes these switches easy to use. We simply configure parameters at each switch to specify the data and control-plane addresses of the remote ends. Our end-host RSVP-TE software inter-operates with the built-in RSVP-TE engines of these switches.

The **end hosts** on the CHEETAH network are general-purpose Linux PCs equipped with two Ethernet NICs, the primary NIC (which can be copper or optical) for connectivity to the Internet, and the secondary NIC, an optical 1GbE or 10GbE card, to connect to an SN16000 switch.

B. Connectivity

1) *Data-plane*: Fig. 2 shows the data-plane topology of the wide-area CHEETAH experimental network. Three CHEETAH Point of Presences (PoPs) equipped with SN16000 switches are located at MCNC in Research Triangle Park (RTP), NC, Southern Crossroads (SOX)/Southern

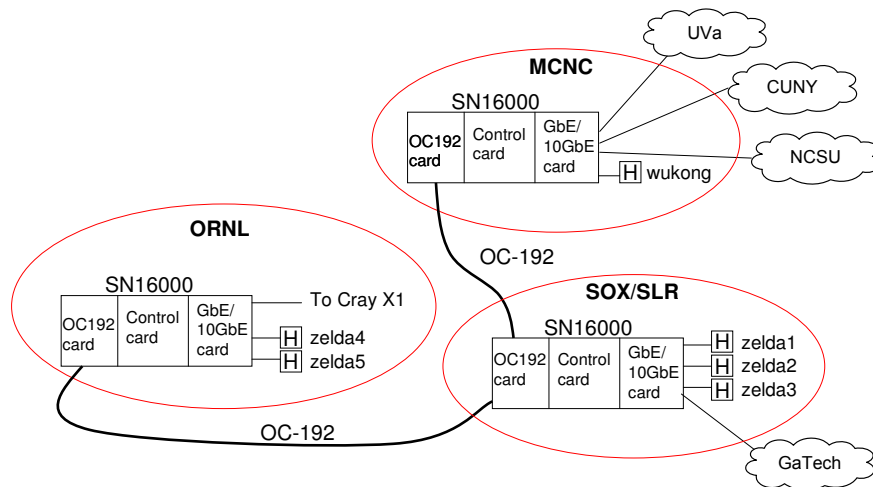


Fig. 2: Wide-area CHEETAH testbed topology (data-plane)

Light Rail (SLR) in Atlanta, GA, and Oak Ridge National Laboratory (ORNL) in Oak Ridge, TN. The three SN16000 switches are interconnected by long-distance OC-192 SONET circuits.

Each of the three SN16000 switches is equipped with GbE interface cards (each with ten ports), which are used to connect enterprise networks and/or end hosts. Fig. 2 shows that currently, four university campuses, University of Virginia (UVa), City University of New York (CUNY), North Carolina State University (NCSU), Georgia Institute of Technology (GaTech) are enterprises connected to the CHEETAH PoPs. The access links are created by provisioning Virtual Local Area Networks (VLANs) or MultiProtocol Label Switch (MPLS) connections across regional networks and the Hybrid Optical and Packet Infrastructure (HOPI) network [21]. A few end hosts, *wukong* in MCNC, *zelda1*, *zelda2*, *zelda3* in SOX/SLR, and *zelda4* and *zelda5* in ORNL, are physically located at the three CHEETAH PoPs and directly connected to the SN16000 switches for experimental research.

2) *Control-plane*: The SN16000 switch supports two types of interfaces for transmitting control-plane messages: (i) in-band Data Communication Channels (DCC) embedded within SONET data-plane interfaces, and (ii) out-of-band Ethernet control ports in the control cards of the SN16000s as shown in Fig. 2. We choose the out-of-band signaling option in the CHEETAH network for switch-to-switch control-plane communication simply because we needed the ability to capture control-plane messages for experimental reasons. The Ethernet control ports of the switches are connected to the Internet.

Control-plane communication between end hosts and switches are also carried on the Internet through the primary NICs on the end hosts.

C. Addressing

Given the shortage of IPv4 addresses, enterprises often use private and/or dynamic IP addresses with techniques such as Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP). In this section, we consider the question of whether private and/or dynamic IP addresses can be assigned to the data-plane and control-plane interfaces in GMPLS networks.

1) *Data-plane addressing*: End hosts in a GMPLS network are comparable to servers on the Internet in that they can be “called” by other clients. A “calling” end host needs to be able to find the IP address of the “called” end host, which implies that the address needs to be *static*. Further, the address needs to be globally unique, which means it has to be *public* if the GMPLS network is to be scalable with subnetworks managed by multiple autonomous-system administrators. As scalability is an important goal of the CHEETAH project, we assign the secondary NICs (data-plane NICs) static public IP addresses. When a process running on a CHEETAH end host, H_1 , wants to initiate a circuit setup to another CHEETAH end host, H_2 , it will look up the domain name of the secondary NIC of host H_2 , find its static public IP address and use this as the destination address in the *Session* object of the RSVP-TE *Path* message.

In GMPLS networks, the same addressing scheme needs to be used at both ends of a link. For example, both ends need to be “numbered,” or both ends need to be “unnumbered.” A numbered interface is assigned its own IP address. An unnumbered interface is assigned an interface identifier (ID), which is used in conjunction with the single IP address assigned to the node on which the interface is located [22]. As there is no necessity to address data-plane interfaces of switch-to-switch links, we use the unnumbered scheme for these links in the CHEETAH network (e.g., see the link between the SOX/SLR and MCNC SN16000s in Fig. 3). However, since the data-plane interfaces of end hosts need to be assigned IP addresses, the corresponding switch interfaces are also numbered (e.g., see the link between the wukong and the MCNC SN16000 in Fig. 3). For reasons that will become apparent in the next section, switch interfaces connected to end hosts need static public IP address assignments.

2) *Control-plane addressing*: As we described in section III-B.2, end hosts send their control-plane messages via their primary NICs. We assign *static public* IP addresses to the control ports of all the end hosts and switches in CHEETAH, as illustrated in Fig. 3. The reason that these addresses need to be *static* is that Traffic-Engineered (TE) links need to be configured at the GMPLS switches to provide information needed by GMPLS signaling engines. Link configuration includes specification of the data-plane addresses (local and remote) and the remote control-plane address. For example, we show the configuration information for two TE-links at

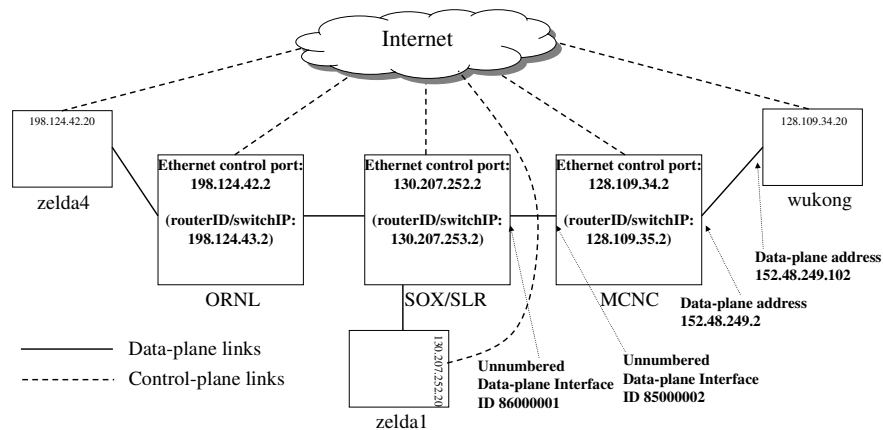


Fig. 3: IP addressing in the CHEETAH network

TABLE I: TE-link configuration at the MCNC Sycamore SN16000 switch

Link	Local data-plane IP address / interface ID	Remote end data-plane IP address / interface ID	Remote end control-plane address	Capacity
Link to SOX/SLR SN16000	85000002	86000001	130.207.253.2	OC-192
Link to end host wukong	152.48.249.2	152.48.249.102	128.109.34.20	1Gb/s

the MCNC SN16000 switch in Table I. The first link is between switches and hence uses unnumbered addressing, i.e., interface IDs, for its data-plane interfaces as described in the previous section. The second link is from the switch to an end host and is hence assigned an IP address for both ends of the data-plane interface. When a call setup request (an RSVP-TE *Path* message) arrives at a switch, it needs the control-plane address of the next-hop switch or control-plane address of the destination host toward which it should route the setup procedure. Given the TE-link configuration process is static, in that it is executed once at initialization time, we need to make the control-plane IP addresses and data-plane numbered IP addresses static. If these addresses are obtained dynamically using DHCP, a dynamic TE-link database update mechanism is needed to inform the neighboring switch or end host. As these procedures are undefined, we use static addresses.

Having understood the reason for requiring control-plane addresses to be *static*, we next examine whether they can be private instead of public. If the GMPLS network under design is a small-scale single-domain network, then control-plane ports can be assigned private addresses. However, given the CHEETAH project goals of creating multiple-domain scalable GMPLS networks, addresses in the control plane will need to be assigned by different administrators. Hence for global uniqueness, these addresses need to be public. The only exception is the control-plane interfaces of internal switches within a single provider’s network, which can be assigned static private IP addresses. However, we need static public IP addresses for the control ports of all border switches [23].

One final point to note in Fig. 3 is that besides needing to assign IP addresses to Ethernet control port(s) in SN16000 switches, a RouterID/SwitchIP address allocation is required for each switch. The RouterID in a GMPLS switch, such as the SN16000, is similar to the RouterID used in IP routers for routing protocol exchanges. The SwitchIP is a similar address used in RSVP-TE signaling messages. Just as the RouterID should be a public static “routable” address in the Internet to allow routing protocol messages to be routed via any of the router’s interfaces, so does the SwitchIP. As recommended by Sycamore, we use a single address for both the RouterID and SwitchIP in the CHEETAH network.

Given the shortage of public IPv4 addresses, we recommend the use of IPv6 to create scalable GMPLS networks. However, due to limitations of the current software implementation in the SN16000, we use IPv4 static public addresses in the CHEETAH network.

3) *Handling the effects of our addressing solution:* Our addressing solution requires dynamic updates to the IP routing table and ARP table at the end hosts as circuits are established and released. We explain the need for this action below.

IP routing table update: We explain the need for IP routing table updates at the end hosts with an example. Say a CHEETAH circuit is established between an end host at UVa to an end host in the NCSU campus through the CHEETAH network (see Fig. 2). Using our recommendation of assigning static public IP addresses (see Section III-C.1), we assign the UVa end host data-plane

interface an address from UVa's 128.143 Class-B address space allocation, and similarly, we assign the NCSU end host data-plane interface an address from NCSU's 152.1 Class-B address space allocation. Once the circuit is set up, IP datagrams generated by an application running at the UVa end host destined for the NCSU end host must be routed directly on to this Ethernet-SONET-Ethernet circuit. But the default configuration of the IP routing table at the UVa end host will indicate that packets destined to the NCSU host interface address be directed to a default gateway (an IP router) since the NCSU interface address is not in the same subnet as UVa's interface address.

The routing problem can be solved by adding a host-specific entry to the IP routing table at the two end hosts after the circuit is established. At the UVa end host, the entry would indicate that the NCSU end host interface is directly reachable on the secondary NIC and vice versa. When the circuit is released, the entries at both end hosts should be deleted.

ARP table update: Continuing with our UVa to NCSU example, having updated the IP routing table at the UVa host to indicate that the next-hop IP router to use for the NCSU host interface address is the interface itself, we must now contend with needing the MAC address of the NCSU host interface. The typical solution is to use ARP. However, using ARP, an address resolution mechanism designed for local-area networks, across a wide-area network, has two drawbacks. First, it incurs a round-trip propagation delay, adding to the overhead of circuit setup delay. In the CHEETAH project, one of our goals is to reduce call setup delay to enable the use of circuits for short-duration calls for increased resource sharing. Second, ARP, being a broadcast mechanism, could lead to broadcast storms if there are Ethernet switches on the end-to-end circuits. Ideally each of these switches should be programmed with a port-mapped untagged or tagged VLAN corresponding to the circuit, but in case of configuration errors, a broadcast storm could be triggered across the wide area.

If ARP is to be avoided as a solution to this wide-area address resolution problem, what then is a more suitable technique? Here, we turn to the other ubiquitous address resolution mechanism used in the Internet today, i.e., the DNS. This system is designed for large-scale networks and

seems ideal to provide support for global-scale IP address to MAC address translations that will be needed if GMPLS networks create wide-area Ethernet-SONET/WDM-Ethernet circuits. We propose using the TXT resource record type of DNS to store MAC addresses corresponding to domain names along with the A-type resource record, which stores IP addresses corresponding to domain names.

As a domain name lookup is invariably required to find the IP address of the remote host (which could incur a wide-area round-trip delay for infrequently accessed hosts), obtaining the MAC address simultaneously would save the additional round-trip delay incurred if ARP was used after the circuit is established. Thus, we propose using the TXT resource records to store MAC addresses of CHEETAH end hosts in the existing DNS hierarchy.

Once the MAC address is obtained, the signaling software at the end host configures the ARP table adding an explicit entry mapping the IP address of the remote end of the CHEETAH circuit to the obtained MAC address after the circuit is established. When user data starts flowing, an ARP table lookup will directly provide the destination MAC address allowing for a quick Ethernet header/trailer encapsulation of the IP datagram being sent to the distant host.

D. Control-plane design

In Section III-B.2, we stated that the Internet is used to transport all control-plane messages in CHEETAH. This is accomplished by connecting the primary NICs of end hosts (which are effectively the control-plane ports of end hosts) and the Ethernet ports of the SN16000 control cards to the Internet. In this section, we describe our solutions to two problems that arise in the control-plane design of GMPLS networks.

First, with the use of an out-of-band control-plane network, i.e., the Internet, two GMPLS switches that are data-plane neighbors may not necessarily be control-plane neighbors. In other words, two switches interconnected by a data-plane interface may be interconnected on a path consisting of one or more IP routers on the Internet. This causes problems for the automatic neighbor discovery process in OSPF-TE.

Second, given the security problems faced on the Internet, there is significant risk in using

the Internet as the control-plane network of a GMPLS network. Both RSVP-TE and OSPF-TE control-plane protocols are used to carry out critical functions. A malicious user could cause significant harm by sending RSVP-TE messages to tie up bandwidth. Similarly a malicious user could send OSPF-TE messages that create routing loops or other such problems. To prevent such attacks, we need a solution to secure the control-plane ports of a GMPLS network.

We describe our solutions to these two problems below.

1) *Enabling OSPF-TE automatic neighbor discovery:* Automatic neighbor discovery occurs by the OSPF-TE implementation at a switch broadcasting a *Hello* message to all its neighbors. If two GMPLS switches are data-plane neighbors, we need to ensure that they are control-plane neighbors as well. We do this by provisioning IP-in-IP tunnels between neighboring GMPLS switches through the Ethernet control ports. A tunnel is viewed as an interface by the switch software. All control-plane messages sent to the broadcast IP address, such as the OSPF-TE *Hello* message, will be routed to all tunnels and interfaces on which OSPF-TE is enabled.

The outer datagram header of the OSPF *Hello* message sent on the IP-in-IP tunnel will carry the SN16000 Ethernet control port IP addresses as source and destination (since these are used in the tunnel-creation commands), and the inner datagram header will carry the sending switch's RouterID and the broadcast IP address as source and destination, respectively. The packets will be routed through the Internet using the outer datagram header and upon reaching the destination, it will be delivered to the OSPF-TE software at the neighboring switch.

When two neighboring SN16000s have exchanged *Hello* messages, the OSPF-TE software in each switch learns the neighbor's RouterID, and automatically adds an IP routing table entry mapping this address to the tunnel interface. Subsequent OSPF-TE messages, such as Link State Updates, which are addressed to the RouterID of a neighbor, will be routed through the IP-in-IP tunnel interface. As noted in Section III-C.2, the SwitchIP address used by RSVP-TE messages is the same as the RouterID. Thus, both OSPF-TE and RSVP-TE messages exchanged between two switches will carry two IP headers when traversing the Internet, the inner one carrying the RouterID/SwitchIP addresses as source and destination (except for the OSPF Hello message, whose inner header will carry the broadcast IP address as destination), and the outer one carrying

the Ethernet control-port IP addresses of the switches as source and destination.

As in connectionless IP networks, we do not expect end hosts to run OSPF-TE in the CHEETAH network. Therefore, no IP-in-IP tunnels are required between end hosts and switches for routing protocol messages.

2) *Security*: Since RSVP-TE and OSPF-TE run directly over IP, we choose IPsec [24] to secure the control plane rather than Secure Shell (SSH) [25], Secure Sockets Layer (SSL) [26], or Transport Layer Security (TLS) [27].

Open-source IPsec software such as Linux Openswan [28], or Cisco/Juniper IPsec clients can be used at the end hosts. Since the SN16000 switch software does not support IPsec, we use an external security device, the Juniper NS-5XT [29], through which tunnel-mode IPsec is configured. Openswan, which runs on the end hosts, interoperates with IPsec software in the Juniper NS-5XT. As shown in Fig. 4, the Ethernet control ports of the three SN16000s are connected to the trusted (secured) interface on an NS-5XT device each.

We establish IPsec tunnels between the NS-5XTs at the two switches configuring them to add an outer IP header with an Encapsulating Security Payload (ESP) header to IP datagrams received from its trusted interface (the link to the switch). The Security Policy Database (SPD) is configured to allow only those packets whose destination and source IP addresses match the

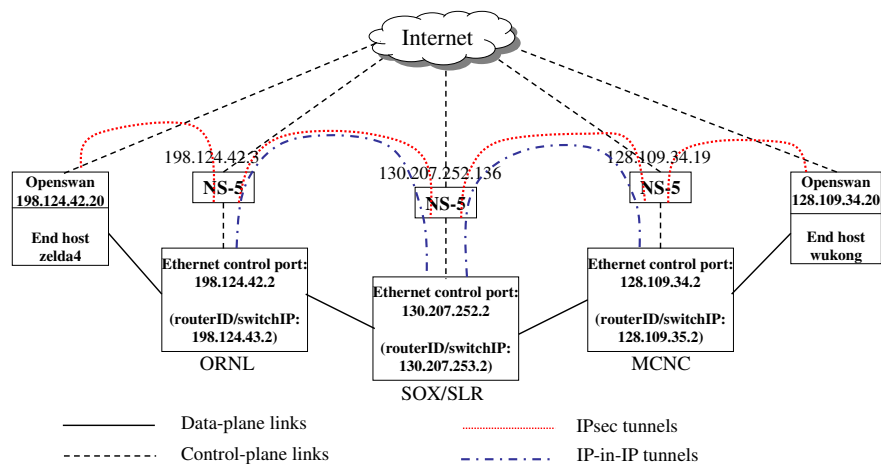


Fig. 4: Creating IPsec tunnels on the CHEETAH network control plane

Ethernet control-port addresses and RouterID/Switch IP of the two adjacent switches through this IPsec tunnel.

When an OSPF-TE or RSVP-TE message is sent on an IP-in-IP tunnel set up between adjacent switches (as described in the last subsection), the payload is already encapsulated in two IP headers. When the NS-5XT receives these IP-in-IP datagrams, it determines which security association and routing entry, if any, the packet matches (by consulting the security policy database and the routing table), and then applies the security services indicated in the SA database for that association. For these IP packets, NS-5XT encrypts the whole IP packet and adds an outer IP header whose destination and source addresses are the NS-5XT untrusted (Internet) interfaces' IP addresses. Thus, there are two tunnels in the CHEETAH control-plane network between any two SN16000s, an inner IP-to-IP tunnel described in the last subsection, and an outer IPsec tunnel. Similarly we configure IPsec tunnels between Linux end hosts running Openswan and the SN16000 switches.

With the ESP header, IPsec authentication capability is provided in addition to encrypting messages. Each switch authenticates the host from which it receives RSVP-TE messages, and its neighboring switches from which it receives RSVP-TE and OSPF-TE messages. As this authentication is host-based, if an intruder is able to gain access to a CHEETAH end host, then this host can be made to issue RSVP-TE messages indiscriminately to tie up network bandwidth. To solve this problem, we plan to use the RSVP-TE Integrity object [30], which will be implemented in the next release of the SN16000 software, to support user-level authentication.

Given our proposed use of static public IP addresses for scalability reasons, it becomes important to configure firewalls to prevent unwanted access to the control interfaces of the switches and end hosts. The Juniper NS-5XTs have firewall capability, which we use in the CHEETAH network. At the end host, we use the Linux `iptables` firewall software.

Finally, Denial of Service (DoS) attacks are possible to the RSVP-TE software at the switches. The NS-5XT device provides some DoS protection but it is not extensive. In summary, through the use of Openswan clients on CHEETAH end hosts and Juniper NS-5XT devices, we are able to provide a minimal level of security for the control plane.

IV. END-HOST CHEETAH SOFTWARE

The end-host CHEETAH software architecture is shown in Fig. 5. We provide a brief overview of CHEETAH software at end hosts, and then focus on the RSVP-TE module. Readers are referred to [31] for more details about the other modules.

The first component of the CHEETAH software shown in the end hosts in Fig. 5 is the **Optical Connectivity Service (OCS) client**. The purpose of this module is as follows. As described in Section II-B, CHEETAH is designed as an add-on service to the existing Internet connectivity. This allows a CHEETAH end host to both communicate via the Internet with end hosts that are not connected to the CHEETAH network, as well as communicate with CHEETAH end hosts on dedicated circuits. To allow an end host to determine whether or not the correspondent end host with which it wants to communicate is on the CHEETAH network, we propose using DNS. Again, exploiting the TXT resource record capability of DNS, we propose adding a text string such as “on-cheetah-network” in this resource record. Thus a DNS query for the TXT resource record of a domain name will yield an answer indicating whether the remote host is on the cheetah network. The calling end host performs this lookup via an Optical Connectivity Service (OCS) client module.

The second component of end-host CHEETAH software shown in Fig. 5 is the **routing decision** module. Recall the choice of two paths available to CHEETAH end hosts as described in Section II-B. Using measurements collected about the state of the two networks, this module

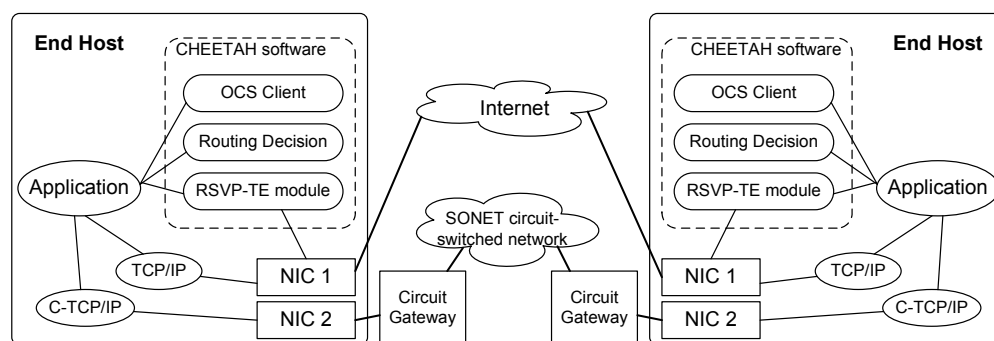


Fig. 5: End-host CHEETAH software architecture

answers queries on which network to use for specific data transfers.

The third component of end-host CHEETAH software shown in Fig. 5 is the **RSVP-TE** module. This module is used to format and send RSVP-TE messages requesting the setup or release of circuits. It also manages the bandwidth of the host-to-switch link (the secondary NIC) and configures the IP and ARP tables as described in Section III-C.3.

In addition to the CHEETAH software module, Fig. 5 shows that CHEETAH end hosts run a module labeled **Circuit-TCP (C-TCP)/IP** [32]. C-TCP is actually a modified version of the TCP code in the kernel with changes made to match the nature of dedicated end-to-end circuits. Since CHEETAH is built as an add-on service to basic Internet communication, our modifications are triggered only for TCP connections established on CHEETAH circuits through NIC2. For connections passing through the primary NIC, NIC1, standard TCP code will be executed. For details on the C-TCP modifications, see [32].

Given the signaling focus of this paper, we will now describe more details about the RSVP-TE module.

A. Functional description of the RSVP-TE module

Recall from Section II.A that switches perform the following five steps upon receiving a *Path* message: message parsing, route lookup, connection admission control, data-plane configuration, and message construction. Similar functions are needed in the **end-host RSVP-TE module**. The **message parsing and construction** steps are the same on end hosts as in the switches. Let us focus on the remaining three steps.

We treat **route lookup** at an end host in the same manner as is currently in practice with IP packet forwarding, in that end hosts are simply programmed with a default gateway address and all IP packets, destined to nodes outside the local network, are sent to this gateway. Similarly, the RSVP-TE module at the end host sends all *Path* messages to its circuit gateway. The circuit gateway then either performs source route computation to reach the specified destination and adds the computed source route in an Explicit Route Object (ERO) to the *Path* message, or simply determines the next-hop switch toward which to direct the call setup. This mode of

operation, in which the end client does not include an ERO, is called the GMPLS overlay model [33], in contrast to the GMPLS peer model where the end client is a router that computes the ERO itself.

Connection admission control (CAC) on end hosts is required because multiple applications running on an end host can independently ask the RSVP-TE module of the end-host CHEETAH software to initiate requests for circuits to remote end hosts. The CAC sub-module of the RSVP-TE module checks to see whether or not the secondary NIC is already tied up in a circuit. In the current implementation, the secondary NIC can be part of only one circuit. In later implementations, we plan to support VLAN multiplexing, which will allow for multiple circuits (to the same or different destinations) to be supported on the secondary NIC. The CAC sub-module will then need to be upgraded to ensure that the total bandwidth reserved across the multiple circuits does not exceed the NIC2 bandwidth.

The **data-plane configuration** needed at an end host is comparable to that needed at a switch. In the latter, it consists of configuring the switch fabric to properly forward user data from an input interface to a corresponding output interface. An equivalent packet forwarding function at end hosts is to move outgoing data from applications to the NIC and in the reverse direction from the NIC to applications. As described in Section III-C.3, this requires configuring the IP routing table and ARP table.

B. RSVP-TE module design and implementation

The end-host RSVP-TE module we developed for Linux hosts is based on RSVP-TE KOM [34]/DRAGON VLSR [6] code. We reused the RSVP-TE message parsing and construction functions and added software for CAC and data-plane configuration. As shown in Fig. 6, the software is divided into three parts: `rsvpd`, `sig_proc` (signaling procedures), and `bwlib`. `Rsvpd` is a daemon that sends and receives RSVP-TE messages, and maintains RSVP-TE sessions. `Sig_proc` is another daemon, which (i) listens for circuit setup requests from local-host applications, performs connection admission control for the host's secondary NIC, and initiates RSVP-TE signaling if the request is admitted, (ii) listens for circuit setup requests

passed to it by `rsvpd` for incoming calls and processes them. `Sig_proc` has a configuration file, which defines the control-plane IP address of the end host, the control-port IP address of the edge switch, and the data-plane link information. `Bwlib` is a library that is integrated into applications to enable the application to communicate with `sig_proc` to request the setup and release of circuits. For example, in our end-host RSVP-TE software package, we include an example application integrated with `bwlib` called `circuitrequestor`, which allows users to request circuits manually. This tool is useful when application source code is unavailable for integration with `bwlib`.

`Rsvpd` and `sig_proc` are configured to automatically start up when the end host system is booted. When started, `sig_proc` loads configuration information from the configuration file, binds itself to a specific port, and listens for circuit-related requests from local applications through `bwlib`. It also registers a default session with `rsvpd` to handle *Path* messages for calls initiated by remote end hosts.

On the sender side, when an application needs a circuit, `bwlib` sends a request, with destination domain name and desired bandwidth, to `sig_proc`. `Sig_proc` then performs CAC and if successful, it forks a child process to handle this request. The newly-forked process initiates circuit setup by constructing a *Path* message and asking the `rsvpd` daemon to send it to the edge switch. When the sender host receives the *Resv* message confirming successful circuit setup, `sig_proc` updates the IP routing table and ARP table to add corresponding entries for the receiver.

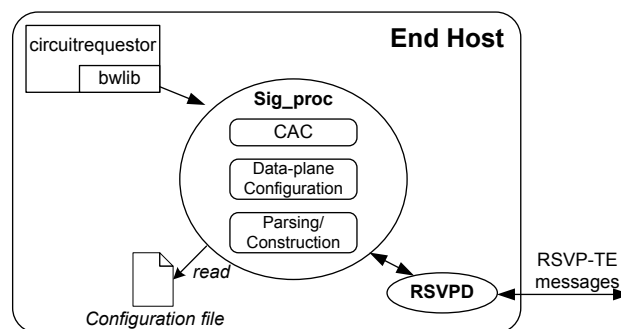


Fig. 6: RSVP-TE module architecture

On the receiver side, when `sig_proc` receives a *Path* message from `rsvpd`, it forks a child process to handle the request. The newly-forked process creates a new session in `rsvpd` for this request and performs the following steps: (i) updates the CAC table to indicate that the data-plane NIC is now reserved, (ii) sends back a *Resv* message to the edge switch, (iii) updates the IP routing table and ARP table by adding entries for the far-end host, and (iv) handles follow-up messages for this session.

V. CALL SETUP DELAY MEASUREMENTS

Given that dynamic bandwidth sharing in connection-oriented switches is controlled by the signaling engine, the request handling performance of the signaling engine is critical to the scaling of networks. The faster the response times of signaling engines, the lower the cost to an application to release and re-acquire bandwidth as and when needed, and the higher the link utilization. Therefore we conducted experiments to measure circuit setup delays.

A. Experimental setup

Measurements are taken for the circuit setup procedure between host `zelda1` and host `wukong` (see Fig. 2). Both `zelda1` and `wukong` are Dell PowerEdge servers running the Red Hat Linux operating system. As seen from Fig. 2, `zelda1` is connected to the SOX/SLR SN16000 (`sn16k-atl`) and `wukong` is connected to the MCNC SN16000 (`sn16k-nc`). The two switches are interconnected by a wide-area OC-192 circuit. The RSVP-TE module on `zelda1` initiates the call. For this circuit, RSVP-TE *Path* message processing is required at the `sn16k-atl` and `sn16k-nc` switches, and at `wukong`, as illustrated in Fig. 1. The primary NICs at the hosts on which signaling messages are carried and the SN16000 control ports are 100Mb/s Ethernet.

Measurements are obtained for three types of circuits: OC-1 SONET circuit, OC-3 SONET circuit, and 1Gb/s Ethernet-over-SONET (EoS) circuit. The experiment was repeated ten times for each circuit type, and the average values are reported. The variance was very small since the Internet path is lightly loaded.

Performance data collected includes end-to-end **circuit setup delay** and **per-switch signaling message processing delay**. **Circuit setup delay** is defined as the time interval from the instant at

which an end host initiates a circuit setup to the instant at which it receives a circuit setup success confirmation from the network. In the experiment, we recorded the time difference between the instant that function `bwrequest` (the function in the `bwlib` library that is invoked by applications to request circuits) is called and the time that the function returns, which indicates that the circuit was set up successfully.

There are two types of **per-switch signaling message processing delays**. The first is “per-switch *Path* message processing delay,” which is defined as the time interval from the instant at which an SN16000 receives a *Path* message from the previous hop to the instant at which it sends out a *Path* message to the next hop. Similarly we define “per-switch *Resv* message processing delay” as the time interval from the instant at which an SN16000 switch receives a *Resv* message to the instant at which it sends out a *Resv* message. To obtain per-switch signaling processing delays, we captured RSVP-TE messages using Ethereal [35] and recorded the time stamps of incoming and outgoing messages.

Besides per-switch signaling message processing delays, the end-to-end circuit setup delay consists of other two components: processing delays incurred at the two end hosts, and propagation plus emission delay of signaling messages. We also measured these delays.

B. Experimental results and analysis

Table II shows measured data for different circuit setup procedures between host `zelda1` and `wukong`. The first column shows the circuit type. The second column shows the end-to-end circuit setup delay. The third and the fourth columns show the per-switch signaling message processing delays observed at the `sn16k-nc` for the *Path* and *Resv* messages, respectively. Finally, we show the round-trip signaling message propagation plus emission delays.

We can see that end-to-end circuit setup delays for an OC-1 circuit and an OC-3 circuit are both around 166ms. The reason that these two delay values are similar is that the procedures for setting up these two pure-SONET circuits are the same.

On the other hand, the end-to-end circuit setup delay for a 1Gb/s EoS circuit is much higher (around 5s). This is because 21 OC-1 circuits are established one after another between the two

TABLE II: Signaling delays incurred in setting up a circuit between `zelda1` and `wukong` across the CHEETAH network

Circuit type	End-to-end circuit setup delay (s)	Processing delay for <i>Path</i> messages(s) at the <code>sn16k-nc</code>	Processing delay for <i>Resv</i> messages(s) at the <code>sn16k-nc</code>
OC-1	0.166103	0.091119	0.008689
OC-3	0.165450	0.090852	0.008650
1Gb/s EoS	4.982071	4.907113	0.008697
Round-trip signaling message propagation plus emission delay between <code>sn16k-atl</code> and <code>sn16k-nc</code> : 0.025s			

switches before the Ethernet segments are mapped to this 21-OC1 circuit. It is purely an artifact of the current release of the SN16000 software whereby virtually concatenated signals, while supported in the data plane are not yet supported in the control-plane. We expect to receive a software upgrade that would reduce EoS circuit setup delays to the same order of magnitude as pure-SONET circuits.

The *Path* message processing delay measured at the `sn16k-nc` for the 1Gb/s EoS circuit can be validated by summing 42 per-switch *Path* message processing delays (one at each switch for each OC-1), 21 times round-trip propagation plus emission delays, and 42 per-switch *Resv* message processing delays (one at each switch for each OC-1). An approximation of *Path* message processing delay for a 1Gb/s EoS circuit is $42 \times (0.091 + 0.0087) + 21 \times 0.025 = 4.71$ s. Roughly speaking, this explains the 4.9s *Path* message processing delay seen at the `sn16k-nc` for the 1Gb/s EoS circuit.

We use these measured end-to-end call setup delays in our analytical/simulation models of file transfers on the CHEETAH network. For example, in [10], where we originally proposed the CHEETAH concept, we used analytical models to determine the conditions under which a dedicated circuit offered a delay advantage over a TCP/IP path for a file transfer. In that study, we used a low call setup processing delay of 4μ s based on our hardware-accelerated implementation of RSVP-TE [36]. This led us to draw conclusions that for TCP connections across a WAN, e.g., connections with a round-trip time (RTT) of 50ms, if the bottleneck link rate is the same as

the circuit rate, then from a delay comparison perspective, it is always appropriate to attempt a circuit setup because if resources are available and setup succeeds, the file transfer will incur a much shorter delay. However, from a utilization perspective, file transfer delay should be much larger than call setup delay. If we assume a factor of 10, then for a 100Mbps circuit across a 50ms-RTT path, we concluded that a circuit setup attempt is warranted for files over 6MB if we assume our hardware-accelerated signaling implementation. On the other hand, with our newly measured value of 5 seconds for call setup delay across a network of off-the-shelf switches, this minimum file size increases to 600MB. We are thus using these measurements for creating a more usable routing-decision module, and are completing a journal paper submission on this topic.

VI. CONCLUSIONS

The CHEETAH project aims at creating a scalable high-speed dynamically controlled circuit-switched network by crossconnecting Ethernet links from end hosts to wide-area SONET circuits. The current wide-area CHEETAH network deployment consists of three Sycamore SN16000 switches interconnected by wide-area OC192 circuits, and Linux end hosts connected to the SN16000s via GbE and 10GbE links. End hosts are equipped with second NICs for this connectivity, thus leaving intact their primary NICs for Internet access. We developed an end-host CHEETAH software package for Linux and a CHEETAH API for usage by application programmers. Applications running on end hosts, such as file-transfer applications, can be upgraded to include the CHEETAH API for a transparent invocation of the CHEETAH service, when appropriate. The end-host CHEETAH software package includes an RSVP-TE module to initiate the setup and release of end-to-end circuits on behalf of end-host applications. Wide-area signaling experiments conducted on the CHEETAH network show that the end-to-end circuit setup delay across a 25ms round-trip propagation-delay path that traverses two SONET switches is around 166ms.

To create scalable multi-domain GMPLS networks, we recommend the use of an Internet-complementary network architecture, the use of IPv6 static public IP addresses for GMPLS

interfaces, the use of the Internet as the control-plane network, the use of IPsec and RSVP-TE Integrity object to secure the control-plane, the use of dynamic updates of IP routing and ARP tables at end hosts as circuits are established and released, and the use of the DNS TXT record to store MAC addresses and an indicator of GMPLS-network connectivity for end hosts.

Our planned next steps are to solve some of the inter-domain routing issues, and then inter-connect the CHEETAH network to the HOPI network and UltraScience Net.

ACKNOWLEDGMENT

This work was carried out under the sponsorship of NSF ITR-0312376, NSF ANI-0335190, NSF ANI-0087487, and DOE DE-FG02-04ER25640 grants. We thank colleagues from CUNY, ORNL, NCSU, MCNC, NLR, GaTech, SLR, the NSF Dragon project, the Internet2 HOPI project and Sycamore Networks, for their help in creating the CHEETAH experimental network.

REFERENCES

- [1] CANARIE's CA*net 4. [Online]. Available: <http://www.canarie.ca/canet4/index.html>
- [2] OMNInet. [Online]. Available: <http://www.icaire.org/omninet/>
- [3] SURFnet. [Online]. Available: <http://www.surfnet.nl/info/en/home.jsp>
- [4] UKLight. [Online]. Available: <http://www.uklight.ac.uk/>
- [5] N. S. V. Rao, W. R. Wing, S. M. Carter, and Q. Wu, "Ultrascale net: Network testbed for large-scale science applications," *IEEE Commun. Mag.*, vol. 43, no. 11, pp. 12–17, Nov. 2005.
- [6] Dynamic resource allocation via GMPLS optical networks (DRAGON). [Online]. Available: <http://dragon.east.isi.edu/>
- [7] UltraLight: An ultrascale information system for data intensive research. [Online]. Available: <http://ultralight.caltech.edu/>
- [8] E. Mannie, "Generalized multi-protocol label switch (GMPLS) architecture," RFC 3945, Oct. 2004.
- [9] L. Berger, "Generalized multi-protocol label switching (GMPLS) signaling - resource reservation protocol-traffic engineering (RSVP-TE) extensions," RFC 3473, Jan. 2003.
- [10] M. Veeraraghavan, X. Zheng, H. Lee, M. Gardner, and W. Feng, "CHEETAH: Circuit-switched high-speed end-to-end transport Architecture," in *Proc. of Opticomm 2003*, Dallas, TX, Oct. 2003.
- [11] X. Zhu, X. Zheng, M. Veeraraghavan, Z. Li, Q. Song, I. Habib, and N. S. V. Rao, "Implementation of a GMPLS-based network with end host initiated signaling," in *Proc. of IEEE ICC 2006*, Istanbul, Turkey, June 2006.
- [12] vsftpd. [Online]. Available: <http://vsftpd.beasts.org/>
- [13] D. Katz, K. Kompella, and D. Yeung, "Traffic engineering (TE) extensions to OSPF version 2," RFC 3630, Sept. 2003.
- [14] J. Lang, "Link management protocol (LMP)," RFC 4204, Oct. 2005.
- [15] B. Metcalfe, "Metcalfe's Law: A network becomes more valuable as it reaches more users," in *Inforworld*, Oct. 1995.

- [16] P. Newman, T. Lyon, and G. Minshall, "Flow labelled IP: A connectionless approach to ATM," in *Proc. of IEEE Infocom*, June 1996, pp. 1251–1260.
- [17] P. Molinero-Fernandez and N. McKeown, "TCP Switching: Exposing circuits to IP," *IEEE Micro Magazine*, vol. 22, no. 1, pp. 82–89, Jan/Feb 2002.
- [18] User-controlled lightpaths (UCLP). [Online]. Available: <http://www.canarie.ca/canet4/uclp/>
- [19] *Generic Framing Procedure (GFP)*, ITU-T Std. G.7041/Y.1303, Dec. 2001.
- [20] *Synchronous Optical Network (SONET) - Payload Mappings*, American National Standards Institute Std. T1.105.02, 1998.
- [21] The hybrid optical and packet infrastructure project (HOPI). [Online]. Available: <http://networks.internet2.edu/hopi/>
- [22] K. Kompella and Y. Rekhter, "Signalling unnumbered links in resource reservation protocol - traffic engineering (RSVP-TE)," RFC 3477, Jan. 2003.
- [23] M. Veeraraghavan, X. Zheng, and X. Zhu. Addressing and secure control-plane network design in GMPLS networks. [Online]. Available: <http://cheetah.cs.virginia.edu/documents/dcn/dcn-design.pdf>
- [24] S. Kent and R. Atkinson, "Security architecture for the internet protocol," RFC 2401, Nov. 1998.
- [25] T. Ylonen and C. Lonvick, "The secure shell (SSH) protocol architecture," RFC 4251, Jan. 2006.
- [26] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL protocol version 3.0. [Online]. Available: <http://wp.netscape.com/eng/ssl3/draft302.txt>
- [27] T. Dierks and C. Allen, "The TLS protocol version 1.0," RFC 2246, Jan. 1999.
- [28] Openswan. [Online]. Available: <http://www.openswan.org/>
- [29] Netscreen-5xt. [Online]. Available: http://www.juniper.net/products/integrated/ns_5series.html
- [30] F. Baker, B. Lindell, and M. Talwar, "RSVP cryptographic authentication," RFC 2747, Jan. 2000.
- [31] X. Zheng and M. Veeraraghavan. CHEETAH end-host software design document. [Online]. Available: <http://cheetah.cs.virginia.edu/software/software-document.pdf>
- [32] A. P. Mudambi, X. Zheng, and M. Veeraraghavan, "A transport protocol for dedicated end-to-end circuits," in *Proc. of IEEE ICC 2006*, Istanbul, Turkey, June 2006.
- [33] G. Swallow, J. Drake, H. Ishimatsu, and Y. Rekhter, "Generalize multiprotocol label switching (mpls) user-network interface (uni): Resource reservation protocol-traffic engineering (rsvp-te) support for the overlay model," RFC 4208, Oct. 2005.
- [34] R. Steinmetz. KOM RSVP engine. [Online]. Available: <http://www.kom.tu-darmstadt.de/en/downloads/software/kom-rsvp-engine/>
- [35] Ethereal. [Online]. Available: <http://www.ethereal.com/>
- [36] H. Wang, M. Veeraraghavan, R. Karri, and T. Li, "Design of a high-performance RSVP-TE signaling hardware accelerator," *IEEE Journal on Selected Areas in Communication*, vol. 23, pp. 1588 – 1595, Aug. 2005.