DESIGN AND DEPLOYMENT OF A GMPLS CONTROL PLANE IN IP OPTICAL NETWORKS

By

ZHAOMING LI

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

THE CITY UNIVERSITY OF NEW YORK

2007

UMI Number: 3283629

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Prof. Ibrahim Habib

_____ _____

Date Chair of Examining Committee

Dean Mumtaz Kassir

_____ _____

Date Executive Officer

Prof. Leonid Roytman (CUNY, CCNY EE Dept.)

Prof. Truong-Thao Nguyen (CUNY, CCNY EE Dept.)

Prof. Mohamed M. Zahran (CUNY, CCNY EE Dept.)

Prof. Nirwan Ansari (NJIT, ECE Dept.)

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

2007

Abstract

DESIGN AND DEPLOYMENT OF A GMPLS CONTROL PLANE IN IP OPTICAL NETWORKS

By

Zhaoming Li

Adviser: Professor Ibrahim Habib

The primary goal of this thesis is to design, implement and deploy a Generalized Multiprotocol Label Switching (GMPLS) software engine in the experimental optical network called Circuit-switched High-speed End-to-End Transport Architecture (CHEETAH) network. This GMPLS software engine, called CHEETAH Virtual Label Switch Router (CVLSR), equips non-GMPLS devices such as Ethernet switches, routers, and other cross-connects with the GMPLS capabilities such that they can participate in the dynamic provisioning of end-to-end bandwidth-guaranteed connections. An important premise of dynamic provisioning is to allow the dynamic sharing of the large bandwidth pipes (e.g., 10 Gbps and above) of backbone networks by many users, thus cutting down the expensive costs of communications at these capacities. The CVLSR is also equipped with a novel dynamic network clustering capability to enable a broad range of applications such as remote visualization, grid computing and e-Learning. The CVLSR has been deployed and tested though experimental CHEETAH optical network across nation-wide high performance networks such as Hybrid Optical/Packet Infrastructure

(HOPI) and Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON) networks. We believe that this and similar efforts will result in a significant progress towards enabling the vision of dynamic provisioning of the end-to-end connections in the multi-region connection-oriented networks.

Next, a link bundling mechanism is presented to improve the efficient utilization of the optical circuits by sharing such circuits among multiple users or multiple applications. The concept of link bundling for dynamic allocation of the optical circuit has been demonstrated through both experimentations and simulations. We demonstrate that link bundling can handle the TCP/IP traffic without TCP connection abort when the buffer size of router interface is appropriately chosen. The packet loss introduced by link unbundling can be recovered by TCP retransmissions.

Finally, a linear programming based scheduling model is proposed to make optimized reservations for the book-ahead services subject to quality of service constraints such as minimizing connection blocking rate while maximizing network utilizations. Our algorithm is simulated based on a single routing area and it could be used for intra-area as well as inter-area and inter-domain scheduling.

Acknowledgments

First, I would like to express my deepest gratitude to my adviser Professor Ibrahim Habib for his continuous support, encouragement and invaluable academic advice. I have been greatly influenced by his methodology in the field of scientific research.

I also would like to thank the faculties and staffs of the department of electrical engineering at the City College of New York for their academic and financial support. Special thanks go to the Examining Committee for their help, efforts, and support with the thesis.

Special thanks are due to the people we have been working together. Thank Qiang Song, my colleague in ANRL lab, for everything from close work to his friendship. Thank Xuan Zheng and Xiangfei Zhu, from University of Virginia, as well as Xi Yang and Chris Tracy, from HOPI/DRAGON networks, for their collaboration and efforts.

This thesis has partly been supported by NSF ANI-0312376, NSF ANI-0335190 grants. Their support is gratefully acknowledged.

Finally, my deepest thanks are to my family, particularly my parents and my husband, for everything. They are always there for me and give me the love and support all the time. Without them, this dissertation would not have been possible.

To My Family

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Some grid computing applications (e.g., e-science applications) are characterized by their massive data storage and processing requirements. Typical data sets generated by these applications may reach terabytes per day. Scientists located all over the globe who work on these applications require a collaborative environment in which they can simultaneously conduct experiments on data stored in geographically dispersed locations. Processing of this data requires the aggregate capabilities of many computers and data analysis tools that are joined together in a grid. The nodes (e.g., processors) in this grid form a hardware/software processing infrastructure that are located in different organizations or even spread over the world. These nodes are dynamic in the sense that they can be added to or removed from the grid depending upon the specific demands of the grid applications. In addition, the processing capabilities may vary depending on the work-loads. The nodes therefore form a "dynamic pool of resources" that can be requested to perform specific computations for a certain period by any application. Dynamic allocation of resources is clearly an important theme not just in allocating grid computing resources, but also in allocating networking resources that are required to inter-connect nodes/grids together.

Depending upon the type of Grid application, requests for resources (whether grid computing or networking) could be immediate or scheduled for a future time (i.e., book-ahead). Further diversity in applications includes duration of connections, i.e., short holding times (in the seconds to minutes range) or long holding times (in the minutes to hours range); and bandwidth granularity, i.e., fine bandwidth pipes (range of tens to hundreds of Mbps) or large bandwidth pipes (ranging from 1 to 10 Gbps). Clearly, the dynamic demands of the Grid applications necessitate that the underlying networks be capable of setting-up and releasing end-to-end connections that cross traditional administrative networks' demarcations. Additionally, it is important to devise resources' scheduling mechanisms that can cost-effectively allocate networking resources to applications while maintaining efficient utilization of these resources. At the outset, the above scenarios imply that dynamic provisioning of end-to-end deterministic bandwidth pipes is an essential capability to enabling grid computing applications.

Examples of such emerging e-science applications that may require grid computing capabilities include high-energy particle physics experiments such as the Large Hadron Collider (LHC) at CERN [1] which produce datasets measured in tens of petabyte per year. Requisite data is then transferred for processing and analyzing from the Collider at research centers distributed all over the globe. This requires end-to-end connections that can deliver 10 Gbps stable throughput between these centers. Requests for these connections are usually scheduled for a future time. In some cases, multiple connections in different networks across the globe may be required to setup simultaneously. Another example of eScience applications is Terascale Supernova Initiative (TSI) [2]. Here, terabytes of data are generated from parallel simulations run on a supercomputer or grids

of high performance computers. The data is then transferred to visualization machines which are usually connected in a cluster mode. Scientists use these simulations to monitor the behavior of physical phenomena arising in the space. Finally, National Aeronautics and Space Administration (NASA) is currently developing several weather and climate prediction applications to predict hurricanes and tsunamis. In these applications, several hundreds terabytes of observational data is collected from locations across the globe and analyzed at different NASA sites. Another similar application to NASA's weather prediction is eVLBI [3] in which large data sets (several hundreds of Gbytes) are collected from telescopes around the world and processed at a central location. There are many other applications outside the eScience community that require connection-oriented paths with predictable performance measures. Examples include health industry applications, as well as commercial ones, such as Video-on-Demand (VoD) and IP-TV among others.

1.2 Motivations

It is clear from the above that emerging new applications have specific challenging networking requirements that can be summarized as followings:

1) Most important is cost-effectiveness. Demands for large or small capacity pipes will not be substantial unless it is economically attractive to users. Offering users quality guarantees at affordable prices implies that network providers maximize the productivity of the network assets without sacrificing quality. This requires new modes of operations, in which end-to-end deterministic bandwidth paths are dynamically provisioned. Large capacity pipes have been traditionally statically provisioned which has been a very costly solution. Statistical sharing of these paths among many users will significantly cut the

cost down. Obviously, there will be a chance of blocking. However, the network resources (interfaces, ports, wave lengths, and time-slots) can be appropriately sized to maintain an acceptable rate of blocking. This circuit switching sharing mode is similar to what we have today in telephone networks. Connection-oriented technologies such as Multiprotocol Label Switching (MPLS) and Generalized MPLS (GMPLS) control plane are the right enabling tools to provide users with on-demand, dynamic provisioning capabilities.

2) Users now require high degree of flexibility for their communication needs. The applications require variable amounts of bandwidths pipes for varying durations. Except for requests scheduled for a future time, there is a need for "application-driven" provisioning of communications paths. This requires the end-users' devices equipped with capabilities for signaling end-to-end connections without consulting a centralized network management system. The GMPLS distributed control plane is a good fit for this requirement.

3) Inter-domain and inter-networking dynamic provisioning is increasingly important. Supercomputers, instrumentation tools, and other storage facilities form grids that cross networks demarcations. Furthermore, these networks are heterogeneous in the sense that networks' elements have different switching and multiplexing capabilities such as wavelength capable, time-slot capable, packet capable, or layer 2 switching capable. Hence, provisioning of heterogeneous paths across multiple networks is an important requirement. Furthermore, devices that are not equipped with GMPLS control plane capabilities (e.g., Ethernet switches) should be equipped with such capabilities.

4) There is a need to schedule (i.e., book-ahead) short-term and long-term, as well as low capacity and large-capacity connections. This requirement will ensure that the networking resources are efficiently utilized in response to the applications' demands. Scheduling is needed not just to allocate and reserve resources, but also to trigger dynamic reconfiguration of the network resources (e.g., time slots at varying granularities) to meet these dynamic demands.

5) Dynamic provisioning of large bandwidth pipes may also require distributed fault management mechanisms, for reliability purposes, capable of performing rapid restoration even across multiple networks.

The above requirements draw a fundamental new picture of networking that is quite different from what is happening in the telecommunications industry today. Users interact with networks in a client-server model. Users provide the carrier with requirements for "static networking demands". The carrier, then, uses a centralized network management (CNM) system to provision static paths to fulfill the customer request. It is suitable for book-ahead large bandwidth pipes (e.g., several Gbps and above) where significant network resources are to be committed, thus prior authorization and network capacity planning are needed from a centralized agent. However, it is not scalable for Bandwidth-on-Demand (BoD) services where requests for services are frequent and require near real time provisioning. Clearly, the distributed control plane is more suited for BoD services because of its scalability. Furthermore, the future high performance inter-networking might consist of diverse data plane technologies, diverse services at different layers, and diverse administrative barriers. A hybrid approach where both planes co-exist together is more likely to be the right way.

Despite the recent developments in the standards and research communities, our experience with the deployment issues of GMPLS-based control plane in either production or experimental networks has been rather limited. To overcome this shortcoming, federal agencies such as the Department of Energy (DoE) and National Science Foundation (NSF) have recently funded several new experimental high performance optical networks such as UltraScience Net (USN) [4], Circuit-switched High-speed End-to-End Transport ArcHitecture (CHEETAH) [5] and Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON) [6]. Furthermore, the internet2 community together with several research and industrial partners has started a new network called HOPI (Hybrid Optical Packet Initiative) [7].

1.3 Thesis Aims

The aims of this thesis are summarized as the followings:

1) Designing and implementing a GMPLS software engine for non-GMPLS devices such as Ethernet switches, routers, and other cross-connects and allow them to participate in the dynamic provisioning of end-to-end bandwidth-guaranteed connections. This dynamic provisioning allows the dynamic sharing of the large bandwidth pipes (e.g., 10 Gbps and above) of backbone networks by many users, thus cutting down the expensive costs of communications at these capacities. This GMPLS engine should be equipped with a novel dynamic network clustering capability to enable a broad range of applications such as remote visualization, grid computing and e-Learning.

2) Deploying and testing this GMPLS engine through the experimental CHEETAH optical network across nation-wide high performance networks such as HOPI/DRAGON network. Its interoperability with other commercial or non-commercial GMPLS control

planes (e.g., Sycamore SN16000 GMPLS module and DRAGON GMPLS control plane) should be demonstrated.

3) Presenting a link bundling mechanism to improve the efficient utilization of the optical circuits by sharing such circuits among multiple users or multiple applications, e.g., the CHEETAH circuits can be used for other applications (e.g., TCP traffic) or other users while there is no CHEETAH connection requested. The link bundling algorithm should be able to handle the TCP/IP traffic without TCP connection abort if one or several component links are unbundled from the bundled link.

4) Proposing a linear programming based scheduling module to make optimized reservations for the book-ahead requests subject to quality of service constraints such as minimizing connection blocking rate while maximizing network utilizations. The performance through this algorithm, i.e., reduction of blocking rate and improvement of network utilization, should be demonstrated via simulation. This optimized scheduler could be used for intra-area as well as inter-area and inter-domain reservation, and thus more connection requests can be admitted.

1.4 Thesis Structure

The rest of the thesis is organized as followings:

Chapter 2 presents a brief overview of GMPLS control plane and different architecture solutions for dynamic provisioning namely: centralized, distributed, and hybrid. We then describe several current experimental end-to-end dynamically provisioned IP optical networks, i.e., USN, DRAGON and CHEETAH networks.

Chapter 3 first introduces the objectives of CHEETAH Virtual Label Switch Router (CVLSR) implementation, i.e., allowing non-GMPLS devices to participate in the

dynamic provisioning of end-to-end bandwidth-guaranteed connections. It then describes the software architecture of CVLSR implementation. The next of this chapter elaborates the design and implementation of each of the CVLSR modules, i.e, parameter configuration, routing, and signaling modules. The dynamic network clustering capability used to enable a broad range of grid computing applications (e.g., remote visualization, grid computing and e-Learning), and the interoperability of the CVLSR with other GMPLS engines (commercial Sycamore SN16000 GMPLS module, DRAGON GMPLS control plane) are also described. Finally, the security mechanism for CHEETAH network is discussed.

Chapter 4 begins with the plan and implementation of CHEETAH network data plane connectivity. The experimental CVLSR deployment and several test scenarios in CHEETAH network across HOPI/DRAGON network are illustrated. The interoperability of the CVLSR with commercial GMPLS SONET-based cross-connect switch (Sycamore SN16000) and DRAGON Virtual Label Switch Router (DVLSR) has been demonstrated. This chapter ends by discussing the delay of Label Switched Path (LSP) establishing and throughput of established end-to-end circuits.

Chapter 5 begins with the introduction of link bundling mechanism. We then demonstrate, through experimentations and simulations, how to use the concept of link bundling for dynamic allocation of the optical circuit such that the bandwidth could be efficiently shared. We demonstrate that link bundling can handle the TCP/IP traffic without TCP connection abort when the buffer size of router interface is appropriately chosen. The packet loss introduced by link unbundling can be recovered by TCP retransmissions. Finally, the router configuration delay is discussed.

In Chapter 6, we first discuss the requirements and challenges of book-ahead scheduling in today's IP optical networks. Linear programming, a very popular algorithm used for multi-commodity traffic flow problem, is then briefly introduced. We propose an optimization model based on the linear programming to schedule the book-ahead connection requests subject to quality of service constraints such as minimizing connection blocking rate while maximizing network utilizations. This chapter ends by simulation results that show the significant improvement of the proposed approach on the desired performance, i.e., reduction of blocking rate and improvement of utilization.

The thesis ends with Chapter 7, which presents a summary of the thesis, the major contribution and future work.

Chapter 2

GMPLS Control Plane in IP Optical Networks

GMPLS is an enabling technology to provide the dynamically provisioned deterministic paths in today's IP optical network. This chapter, first, gives an overview of GMPLS control plane; then summarizes different architecture solutions for dynamic provisioning namely: centralized, distributed, and hybrid. Next, we briefly describe the current activities undergoing in recent experimental high performance optical networks such as USN, DRAGON, and CHEETAH. We compare the control and management approaches adopted in each of these networks and analyze their capabilities vis-à-vis the functional requirements of grid computing applications.

2.1 Overview of GMPLS Control Plane

The GMPLS [8] control plane has three main capabilities: distributed routing, signaling for near real time connection setup, and auto-discovery mechanisms of both network services and topology. It is the enhancement of MPLS [9], that is a packet-forwarding technology to provide traffic engineering and bandwidth guarantee for a specified path. Extended from MPLS, GMPLS supports multiple types of switching, i.e., TDM, lambda, and fiber switching, as well as packet switching. Therefore it is able to provide a unified control plane across layer 3, 2, and 1 devices and provision dynamic end-to-end deterministic paths across different administrative domains.

The GMPLS control plane is based on well-known GMPLS routing and signaling protocols extended and/or modified from MPLS Traffic Engineering (MPLS-TE) [10]. Additionally, a new signaling protocol, Link Management Protocol (LMP) has been introduced.

GMPLS routing protocols are used for the auto-discovery of network topology, advertisement of resource information, and route selection. GMPLS routing extensions [11] are based on two traffic engineering routing protocols, i.e., Open Shortest Path First - Traffic Engineering (OSPF-TE) [12][13] and Intermediate System to Intermediate System - Traffic Engineering (ISIS-TE) [14][15], where GMPLS OSPF-TE is predominant in IP optical networks.

GMPLS signaling protocols are used for the establishment of TE-LSPs. GMPLS signaling extensions [16] are based on two signaling protocols defined for MPLS-TE, i.e., Resource Reservation Protocol - Traffic Extension (RSVP-TE) [17][18] and Constraint-based Label Distribution Protocol (CR-LDP) [19][20], where RSVP-TE is widely deployed in today's optical networks.

LMP [21] is used to manage TE links between adjacent nodes in data plane. It provides the functionalities such as control channel management, link connectivity verification, link property correlation, as well as fault management (i.e., fault localization and notification).

The provisioning of dynamic end-to-end path is accomplished by the establishment of the TE-LSP. The routing protocol computes a TE path, while the signaling protocol performs the signaling and cross-connection of this TE path. The TE link, i.e., the physical resource and its TE properties (e.g., bandwidth, switching capabilities, and

protection type, etc.) between two GMPLS nodes, is advertised as the Link State Advertisement (LSA) by the routing protocol.

The GMPLS control plane supports a peer model, an overlay model, and an augmented model. In a peer model, the complete resource and topology information, the same routing and signaling protocols are shared by all network devices. In an overlay model, the network consists of a set of Autonomous Systems (ASs) that are managed by different administrative authorities. Each of them desires to hide the inner network information from others. The inter-operation can be done via User-Network Interface (UNI). The augmented model is a hybrid model of full peer and overlay models.

2.2 Architecture Solutions for Dynamic Provisioning

End-to-end dynamically provisioned optical network, which is current used for e-Science applications, is being adopted by scientists for grid computing applications to meet the terabyte to petabyte data transfer requirement [22][23][24]. In terms of network architecture, dynamic provisioning can be realized in centralized management, distributed control, and hybrid approaches.

In the centralized management approach (Fig. 2.1), the element management layer is responsible for activating (i.e., cross-connecting) the actual network element based upon requests from the upper network management layer. It relays to the upper layer the information of each network element such as configuration, connectivity, alarms, and other measurements for performance and capacity management. The element management layer also gathers the network topology and links' TE information and sends them to the network management layer where they are used for functions like network planning and provisioning of services. It is also used to update important databases such

as the network inventory database where the network assets are maintained and cross-correlated with the services they support. In many cases, this process is not fully automated, thus leading to errors in determining which network resources are available for provisioning new services and which ones are already committed.

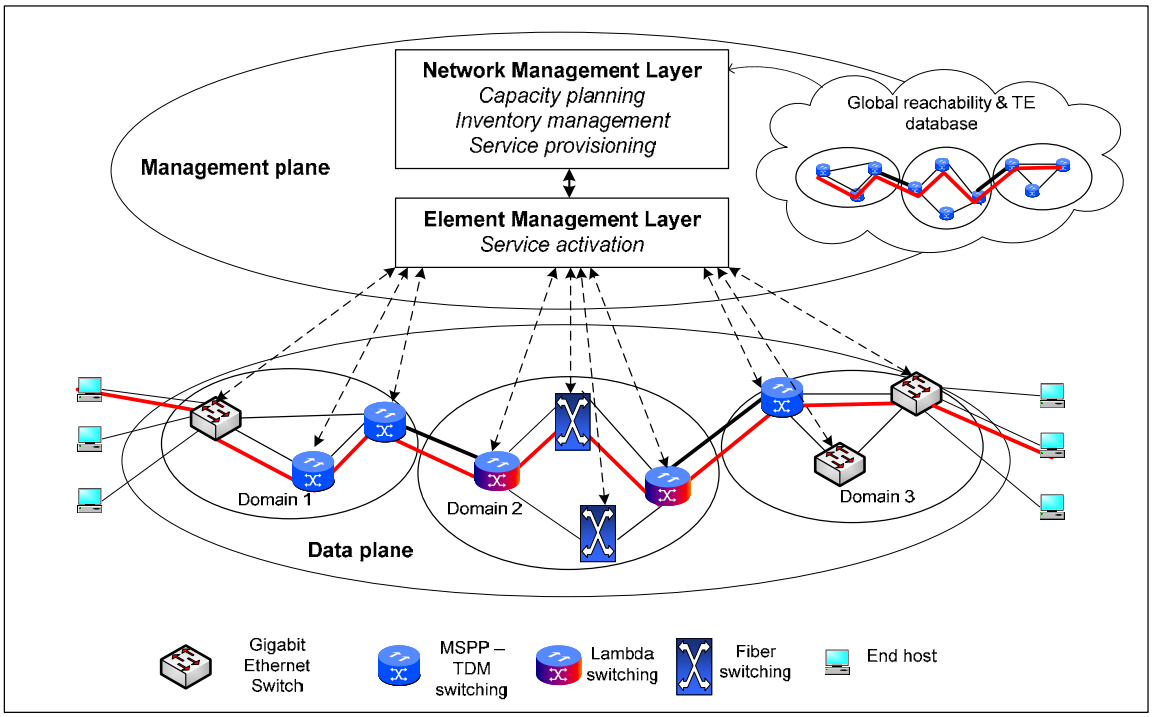

Fig. 2.1 Centralized dynamic provisioning approach

The network management layer is responsible for the service activation and provisioning. Upon receiving a connection request from end-host, the network management layer calculates the end-to-end path, generates the activation signal, and then sends it to each network element through the element management layer. Since the centralized approach has a global view of the whole network resources, it allows a more optimal path computation, which leads to more efficient bandwidth utilization compared to the distributed approach.

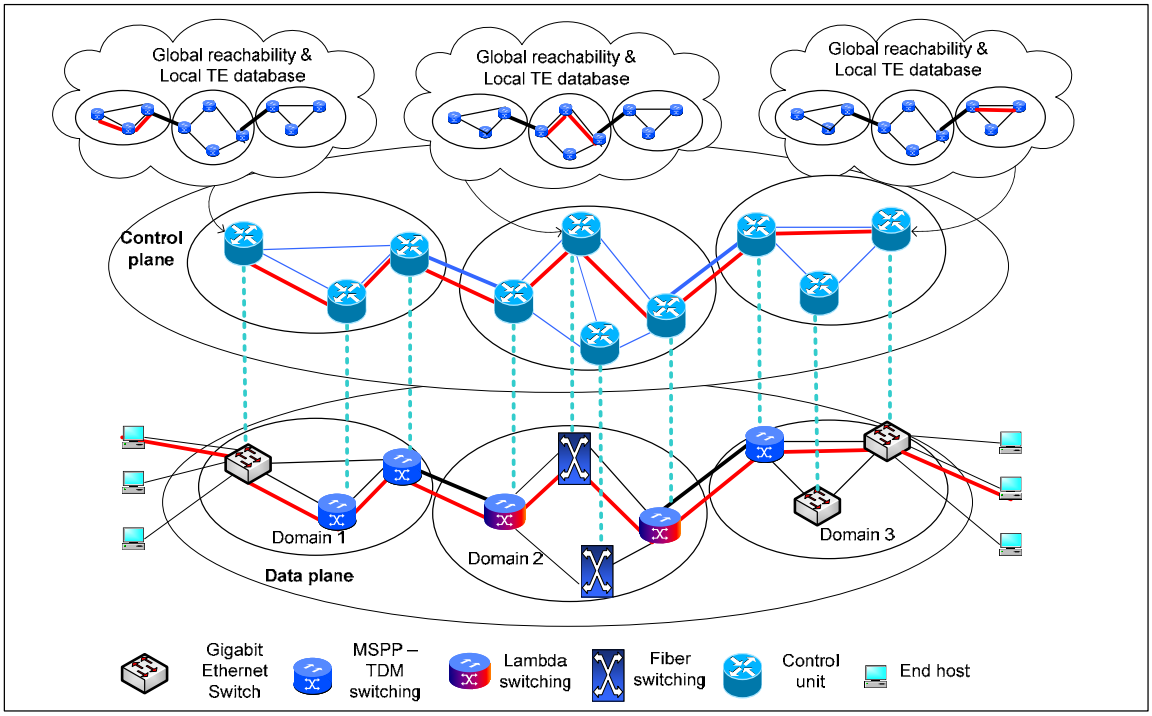

Fig. 2.2 Distributed dynamic provisioning approach

The distributed GMPLS control plane (shown in Fig. 2.2) is based upon three main pinnacles: routing, signaling and auto-discovery. Each network element is controlled by GMPLS software containing OSPF-TE or IS-IS-TE module for dynamic routing, RSVP-TE or LDP-TE module for signaling, and LMP module for auto-discovery of services and topology between neighbors. Each GMPLS within every switch has a topological view (reach-ability and TE information) of the network within its specific routing domain, where a domain is defined as a collection of network elements with the same address space or path computation responsibility such as an Interior Gateway Protocol (IGP) area. Within a single domain, path computation can be done via source routing and a strict explicit route can be computed to the routing-area boundary. For inter-domain routing, since only reach-ability information is available, loose routing or hop-by-hop routing is used to find the next hop between different domains.

Path computation across different domains can be done in the domain border switches, or in a centralized agent responsible for a specific domain, or even multiple domains. This centralized agent, called Path Computation Element (PCE) [25][26], gathers the TE information and disseminates it to the other domains' path computation modules. The inter-domain end-to-end path then can be computed by this PCE. However, this hybrid approach may not scale due to the amount of TE information exchanges. As the network grows, and support for services such as BoD is required, connection requests may arrive faster than the updating of link state databases. The decisions made by a PCE (crossing multiple domains) will be outdated, thus leading to computation of paths that are no longer available. This scenario is similar to telephone networks today where a path is signaled on an end-to-end using distributed routing, without consulting a centralized agent about bandwidth availability in different domains. However, scenarios where some level of abstracted information about other domains transport and/or switching capabilities may need to be exchanged across different domains. Examples of such scenarios include signaling heterogeneous paths with different switching capabilities. Hence, path computation across different domains will be required in order to improve the efficiency of bandwidth utilization.

To summarize, there is a tradeoff between scalability, and optimization of the network resources. Scalable solutions require distributed path computation approaches that lend themselves to a hop-by-hop approach. In IP optical networks, small routing areas with few network elements and node degrees, in addition to techniques like link bundling, are crucial to avoid excessive floods of OSPF-TE information and excessive synchronization delays of OSPF databases. On the other hand, computation of end-to-end

strict paths achieves better utilization of the network assets but requires extreme caution with respect to scalability. Finally, inter-domain signaling can be achieved using RSVP-TE and utilizing new mechanisms such as nesting, stitching, or contiguous or hybrid mixture of them, for details see [27] and the references therein.

2.3 Examples of Experimental Optical Networks

Recently, some experimental optical networks have been deployed to study and demonstrate the efficacy of GMPLS in enabling e-Science applications (i.e., large data transfers) among others. Several of such examples, USN, DRAGON, and CHEETAH, are introduced in this section.

2.3.1 USN

USN is under construction by Oak Ridge National Laboratory (ORNL). It is commissioned by the U.S. Department of Energy (DOE) to facilitate the development of technologies specifically targeting the large-scale science applications carried out at national laboratories and collaborating institutions.

User sites can be connected to USN through its edge switches, and can utilize the provisioned dedicated channels during the allocated time slots. Its design requires several new components including a Virtual Private Network (VPN) infrastructure, a bandwidth and channel scheduler and a dynamic signaling daemon. The control-plane employs a centralized scheduler to compute the channel allocations, and a signaling daemon to generate configuration signals to the switches (as shown in Fig. 2.3). The USN control plane facilitates a number of USN functions such as (a) monitoring, configuration and recovery of its core switches, Multiservice Provisioning Platforms (MSPPs), and hosts;

(b) providing user access to USN hosts, and user/application access for requesting channel setup and obtaining state information about hosts and channels; and (c) signaling for on-demand setting-up and tearing down of the dedicated channels. Various allocations and cross-connection information is stored on a central server located at ORNL. Clearly, the USN control plane is currently a "centralized one"; however, a distributed GMPLS control plane will be adopted in the near future.

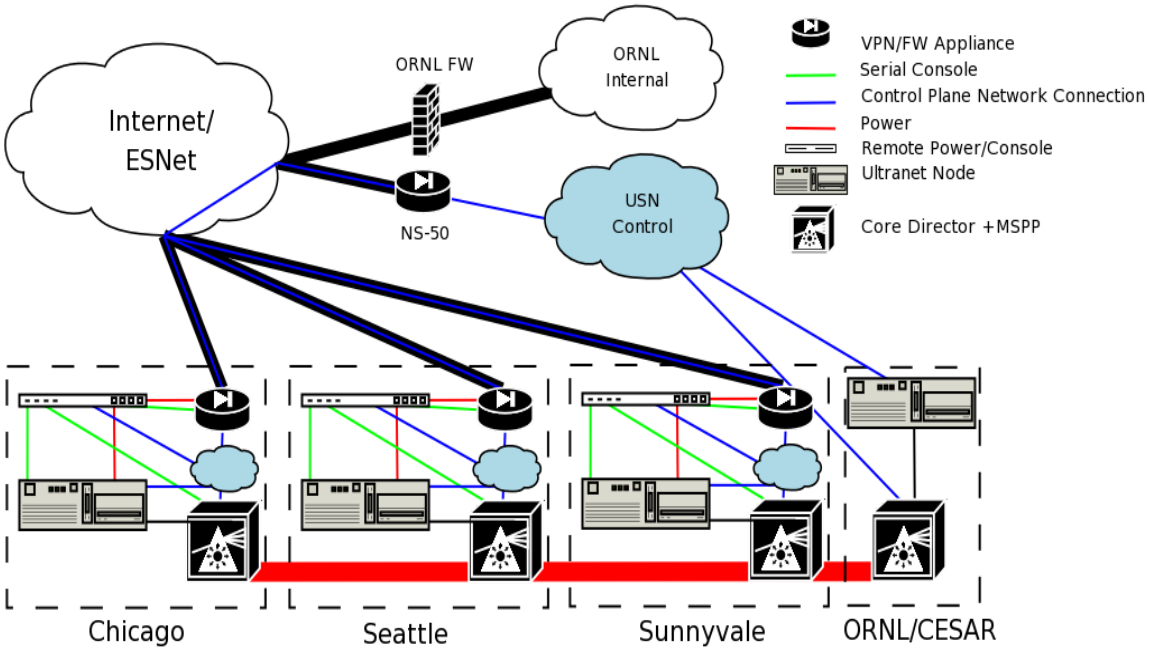

Fig. 2.3 USN network centralized control plane

2.3.2 DRAGON

DRAGON is collaborative project between University of Maryland (UMD) Mid-Atlantic Crossroads (MAX), University of Southern California Information Sciences Institute East (USC/ISIE), and George Mason University (GMU). The DRAGON project is developing technology and deploying network infrastructure which allows advanced e-science applications to dynamically acquire dedicated and deterministic network resources to link computational clusters, storage arrays, visualization facilities, remote

sensors, and other instruments into globally distributed application specific topologies. A reference implementation has being constructed in the Washington D.C. area as shown in Fig. 2.4.

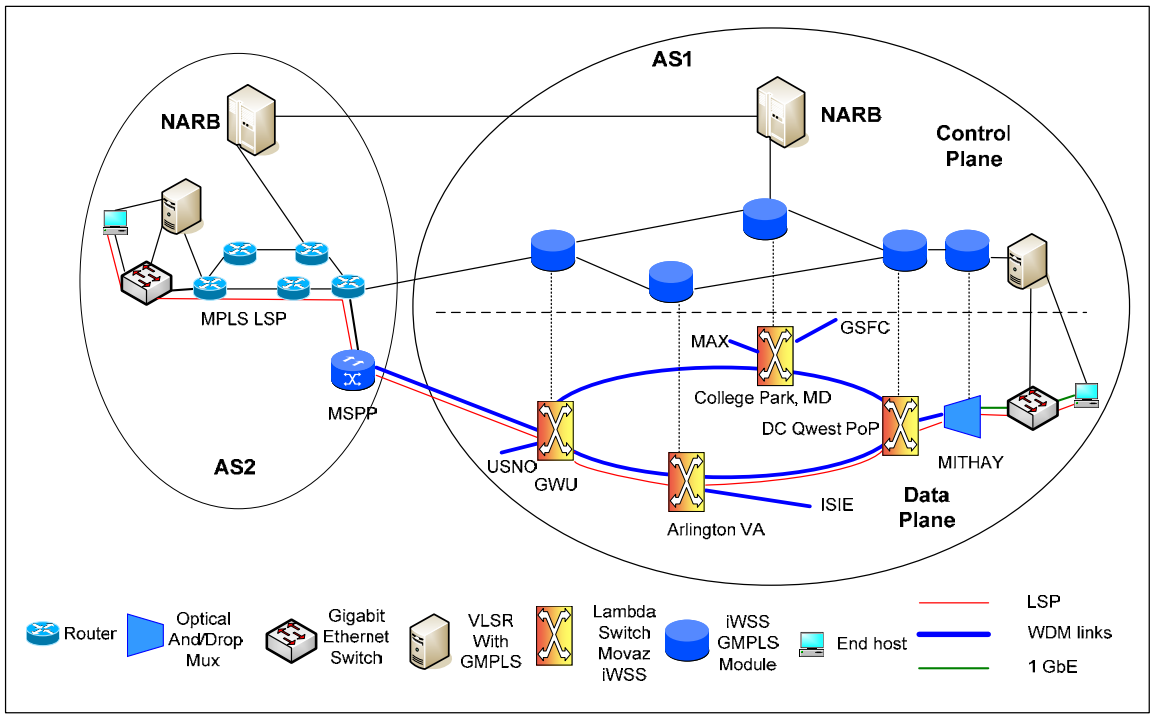

Fig. 2.4 DRAGON network architecture in Washington D.C. area

DRAGON is an example of deploying a hybrid approach consisting of both the control plane and a semi-centralized agent, called Network Aware Resource Broker (NARB). The control plane is based on GMPLS which provides the basis for the dynamic provisioning of network resources. The DRAGON architecture is a hybrid between the peer-to-peer and overlay model. Inter-domain traffic engineering routing is accomplished by utilizing a modified version of OSPF-TE which incorporates domain abstraction and hierarchical routing techniques. Domain abstraction provides mechanisms for an administrative domain to advertise to the outside world a highly simplified view of its

topology. This allows domains to hide their real topologies as well as minimize the amount of external updates required.

An important objective for the DRAGON architecture is to be able to provision across heterogeneous network technologies and vendors' equipment. For vendor equipment which is not GMPLS capable, the concept of the Virtual Label Switch Router (VLSR) is adopted. The VLSR is a GMPLS engine that allows non-GMPLS devices to participate in the routing and signaling of end-to-end paths. The NARB is responsible for multi-domain, multi layer network resource provisioning and management. Routing and path computation on an inter-domain basis across topologies that include a heterogeneous mix of network technologies and vendors' equipment is beyond what is defined in standards and also beyond the capability of current vendors' equipment. The NARB is used to enable routing, path computation and signaling in such environments. There is generally one NARB per administrative domain. It acts as a protocol listener to the intra-domain routing protocols. It also takes care of domain abstraction, inter-domain routing, and inter-domain path computation. Allowing the NARB to hold the inter-domain link state topology alleviates the need to flood that information into the local domain.

2.3.3 CHEETAH

CHEETAH project is a comprehensive effort to develop the infrastructure and networking technologies to support a broad class of e-Science projects and specifically TSI. CHEETAH team consists of optical networking researchers, transport-layer and middleware researchers, and radio-astronomy scientists from University of Virginia (UVa), ORNL, North Carolina State University (NCSU), and City University of New York (CUNY).

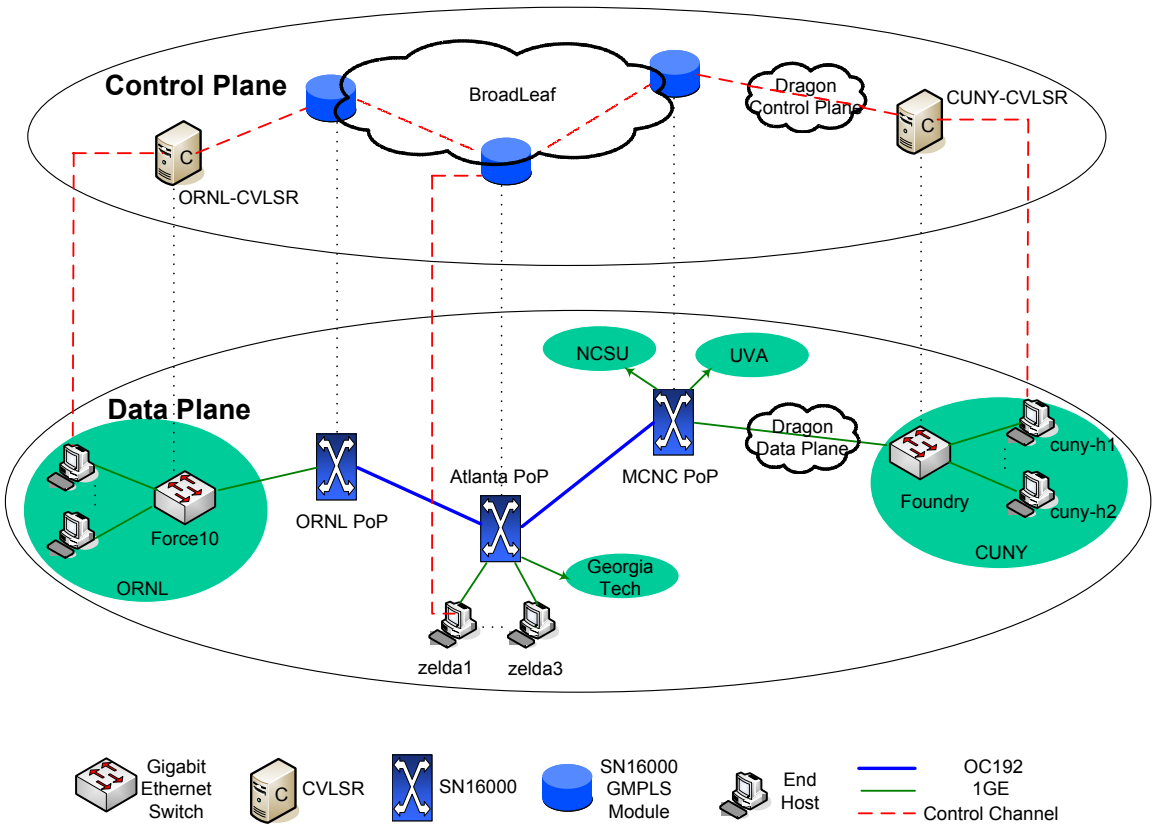

Fig. 2.5 CHEETAH network topology

Fig. 2.5 shows the wide-area CHEETAH network topology. Its data plane consists of OC-192 SONET and 10 Gbps Ethernet connections interconnecting several sites like ORNL with NCSU where eScience applications run. Three CHEETAH Point of Presences (PoPs) equipped with Sycamore SN16000 SONET cross-connect switches (SONET-XC) form the backbone of CHEETAH network and are deployed at MCNC in Raleigh, NC, Southern Crossroads (SOX)/Southern Light Rail (SLR) in Atlanta, GA, and ORNL in Oak Ridge, TN. Other connections include several 1 Gbps Ethernet links to other sites as outlined in the figure, extending CHEETAH network to ORNL, NCSU, CUNY, UVa, and Georgia Tech, etc.

CHEETAH uses a distributed GMPLS control plane to enable application-initiated, on-demand dynamic provisioning of end-to-end circuits, where both GMPLS peer and

overlay models are supported. Additionally, high performance transport protocol and application middleware, Fixed Rate Transport Protocol (FRTP), is used to provide guaranteed and stable throughput for data transfer, visualization, steering and control applications. Dynamic routing protocol OSPF-TE and signaling protocol RSVP-TE with GMPLS extensions are enabled in each control unit.

In CHEETAH network, some end hosts (for example, zelda1 and zelda3), physically located at the CHEETAH PoP sites, are connected to the SN16000 switches with direct cables/fibers. However, since enterprises are geographically distant from the CHEETAH PoPs, it is not feasible to connect enterprise end hosts to the SN16000 switches via direct physical connections. Therefore, we connect such end hosts to SN16000 switches through Gigabit Ethernet switches by provisioned Virtual Local Area Network (VLAN). These Gigabit Ethernet Switches should be GMPLS enabled so that they can participate in the routing and signaling process to provide dynamic provisioning. This GMPLS engine for Ethernet switch is called CHEETAH VLSR (CVLSR), which will be described in detail in the next chapter.

The data plane connections in CHEETAH network are Ethernet over SONET (EoS) dynamically provisioned at sub-1Gbps and above granularities. The visualization and steering application requires the setup of a connection from ORNL supercomputer to a cluster of end hosts at NCSU that share the bandwidth of the connection. To enable this application, tagged VLAN capabilities have been used. The bandwidth allocated to the cluster is controlled by the CLVSR.

Chapter 3

CVLSR Design and Implementation

The CHEETAH Virtual Label Switching Router (CVLSR) is an implementation of the GMPLS software suite that allows non-GMPLS devices (e.g., Ethernet switches, routers and other cross-connects) to participate in the dynamic provisioning of end-to-end bandwidth-guaranteed connections. This chapter first introduces the objectives of CVLSR implementation. It then describes the software architecture of CVLSR implementation. The next of this chapter elaborates the design and implementation of each of the CVLSR modules, i.e., parameter configuration, routing, and signaling modules. The CVLSR dynamic network clustering capability used to enable a broad range of grid computing applications (e.g., remote visualization, grid computing and e-Learning), and the interoperability with other GMPLS engines (e.g., commercial Sycamore SN16000 GMPLS module and DRAGON VLSR) and UVa RSVPD client are also described. Finally, the security mechanism for CHEETAH network is discussed.

3.1 Implementation Objectives

In CHEETAH network, a distributed GMPLS control plane approach is used to enable application-initiated, on-demand dynamic provisioning of end-to-end circuits at various granularities (e.g., 10 Mbps to 10 Gbps). An important premise is the dynamic sharing of the large bandwidth pipes of backbone networks by many users, thus cutting

down the expensive costs of communications at these capacities. In order to realize dynamic provisioning of such end-to-end circuits (e.g., Ethernet-EoS-Ethernet, or others), each Ethernet switch is equipped with a GMPLS-enabled engine, called CVLSR, to enable non-GMPLS Ethernet switch to participate in the dynamic setup and release of bandwidth guaranteed connections. Therefore, the users could share the network resources dynamically and makes it possible to extend the connections across different administrative domains to the end-users. It offers a scalable and cost-efficient solution for adding users to the network, whereas adding users directly to PoP nodes will quickly exhaust the capacity of the PoP ports, i.e., the GbE ports of the Sycamore switches in CHEETAH network.

The objectives of CVLSR implementation could be summarized as the followings:

1) Designing, developing and implementing a stable GMPLS software engine, i.e., CVLSR, to equip non-GMPLS devices, such as Ethernet switches, with capabilities to dynamically provision guaranteed bandwidth pipes. The dynamic provisioning can be realized via utilizing dynamic setup and control of VLANs within the switch. The CVLSR should be capable of supporting both Fiber Switch Capable (FSC) and Layer-2 Switch Capable (L2SC) concerned in current CHEETAH network. It should be able to set rate limiting and bandwidth guarantees to the ports and VLANs of Ethernet switch in order to provision sub-1G Ethernet VLAN connections.

2) Grid computing application requires any host in any enterprise cloud should be able to setup a connection to one or more hosts in another enterprise. Optionally, there may be a need that the host in one enterprise sets up multiple connections (sum of which is less than 1GbE) with several hosts in other enterprises simultaneously. The CVLSR

should be equipped with a dynamic network clustering capability to dynamically setup and release a single connection with multiple participating end-hosts on each end of the connection. It is a possible solution to enable a broad range of dynamic clusters applications such as remote visualization, grid computing and e-learning at sub-1Gbps or above levels.

3) In CHEETAH network, Sycamore SN16000 backbone network and HOPI/DRAGON network belong to different ASs. The CVLSR should be able to inter-operate with their GMPLS control units, i.e., SN16000 GMPLS module and DRAGON VLSR. Additionally, the CHEETAH end host is not equipped with routing module in order to reduce the flooding of routing information. The end host initiates an LSP request through UVa RSVPD client, which sends Path message directly to the CVLSR. The CVLSR should be capable of interoperating with this client.

4) The CVLSR is developed based upon the DVLSR [28], which is implemented for dynamic provisioning in HOPI/DRAGON network. The DVLSR, in turn, is based upon the implementation of an open source Open Shortest Path First (OSPF) engine (GNU Zebra) [29] and Resource ReSerVation Protocol (RSVP) engine (KOM RSVP) [30][31]. Due to the distinguishing features of CHEETAH network, CVLSR also should be capable to support some functions that were not supported by DVLSR, for example, hop-by-hop routing, Connection Admission Control (CAC) function, assignment of IP addresses to point-to-point control channels, support of unnumbered data plane interfaces, and control channel separation, etc.

5) The routing and signaling messages are transmitted over the internet in CHEETAH network. Securing control plane access to the network elements is important

to protect them from internet attacks, such as Denial-of-Service (DoS) attacks, prevent hackers from accessing the control processors, prevent un-authenticated users from sending RSVP Path messages to request the CHEETAH circuits, and prevent malicious users from tying up large amounts of bandwidth, etc.

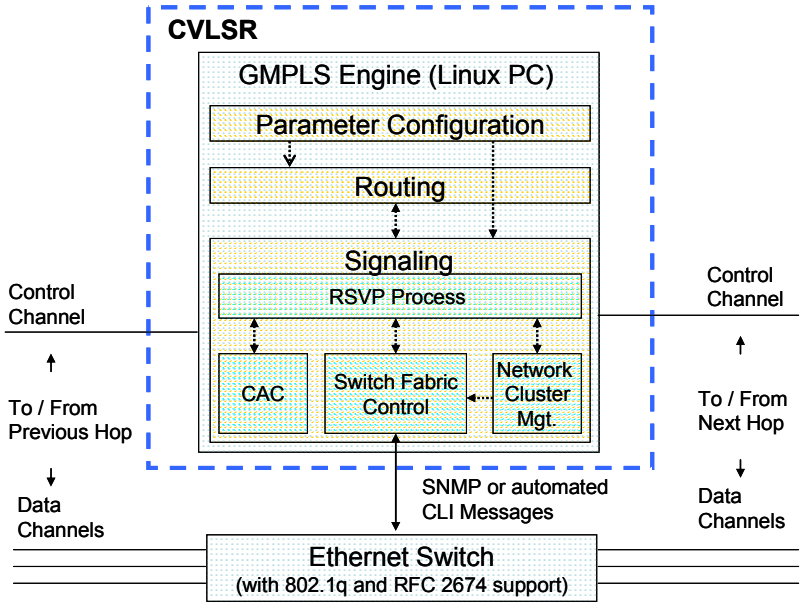

Fig. 3.1 CVLSR software architecture

3.2 Implementation Architecture

The CVLSR is an implementation of the essential components of the GMPLS control plane, such as routing, signaling, and auto-discovery. As shown in Fig. 3.1, it consists of three basic modules: parameter configuration, routing, and signaling. The system configurations, including link state information, port and VLAN resources, and other parameters, are managed by the parameter configuration module. GMPLS OSPF-TE is adopted as the routing protocol to be responsible for the routing and path computation, while GMPLS RSVP-TE is adopted as the signaling protocol to be responsible for signaling processing and switch fabric cross connection. The auto-

discovery is accomplished by parameter configuration and by means of OSPF-TE. The signaling module is composed of the following four sub-modules: RSVP process, CAC, switch fabric control, and network clustering management function.

The parameter configuration module reads the configuration files and builds the data structure for routing and signaling modules, for example, the routing table, link state database, and port mapping, etc. The routing and signaling modules exchange the resource information, such as path computation, bandwidth update and among others. The RSVP process sub-module of signaling module parses and processes the incoming RSVP messages. While the CAC sub-module is used for the admission control of the LSP request, the switch fabric control sub-module performs the cross connection to the Ethernet switch when end-to-end reservation is successfully admitted. If the reservation is for cluster connection, the cluster network establishment will be accomplished by the clustering sub-module.

3.3 Parameter Configuration Module

The Parameter configuration module is a user interface to configure the CVLSR network parameters. It loads these parameters from configuration files to the routing and signaling modules so that the following data structures such as routing table, link state database, control/data interfaces mapping, data interface ID/port number mapping, local/remote interfaces mapping, and cluster member ID/port number mapping, can be established accordingly.

The following parameters should be manually configured for CVLSR:

1) Router address, area ID, and network address.

2) Control channel interfaces.

3) OSPF-TE information, such as the local and remote control channel IP addresses, local and remote data channel IDs and TE information including the port numbers, static VLAN IDs, interface switching capabilities, and maximum reserve-able bandwidths, etc.

4) CHEETAH related information, such as egress port number, tagged port numbers, cluster port numbers and associated IP addresses, etc.

3.4 Routing Module

The major function of routing module is computing the path from source to destination at requested bandwidth and switching capability. It supports the following IETF RFCs: OSPF Version 2 (RFC 2328) [12], The OSPF Opaque LSA Option specification (RFC 2370) [32],Traffic Engineering (TE) Extensions to OSPF Version 2 (RFC 3630) [13], Routing Extensions in Support of GMPLS (RFC 4202) [11], OSPF Extensions in Support of GMPLS (RFC 4203) [33], and GMPLS Signaling Functional Description (RFC 3471) [[17]], etc.

The implementation of routing module is based on the DVLSR routing module. It has been modified to meet some specialized CHEETAH requirements. The modification of CVLSR routing module can be summarized as the followings:

1) Routing method

The CVLSR supports both GMPLS peer and overlay model. The peer model is used when the CVLSR is able to obtain all routing and TE information of the core network and enterprise networks, where source routing is used for path computation. The overlay model is used whenever it is desired to hide the details of the backbone topology from the enterprise networks. Since the CVLSR is not able to obtain the complete resource and

topology information in the overlay model, hop-by-hop routing is adopted for path computation.

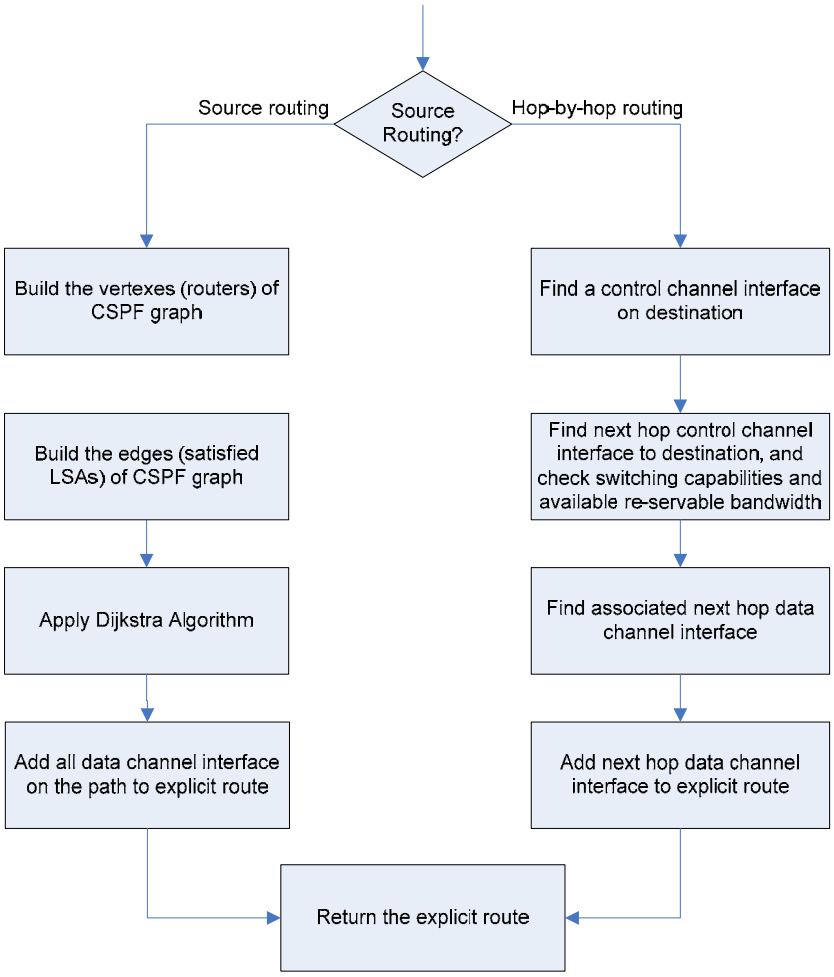

Fig. 3.2 Flow chart of routing discovery

Fig. 3.2 illustrates the flow chart of the path computation in the CVLSR. In source routing, an end-to-end path is computed via Constrained Shortest Path First (CSPF). A weighted directed CSPF graph is built based on the TE link state database first. All router nodes are added as the vertexes of the graph, whereas only the links with satisfied switching capabilities and available bandwidth are added as the edges of graph. The explicit route to destination is then computed by Dijkstra algorithm. In hop-by-hop routing, the CVLSR could not obtain the next hop from routing table directly, since the

destination might cross multiple areas such as HOIP/DRAGON and Sycamore backbone cloud. It should find the control channel interface of the destination and then looks for the next hop interface based on the destination interface, where the available bandwidth and switching capabilities to the next hop are checked. The explicit route containing only the next hop address is returned upon the successful computing. In CVLSR, the explicit route is represented by a series of data channel interfaces.

2) Data exchange between routing and signaling module

The routing module also processes certain information requests from the signaling module and returns the results. For examples, it returns the next hop or explicit route upon the path request; it returns the control interface ID according to the data interface ID and vice versa; it returns the remote control/data channel interface ID according to the local control/data channel interface ID and vice versa. All above functions are implemented based on the routing table and link state database built by itself. Additionally, the routing module updates and advertises the bandwidth information of the corresponding TE LSAs in link state database upon receiving the RSVP bandwidth reserve/release notifications from the signaling module.

3) Point-to-point channel configuration

Generally, a Generic Routing Encapsulation (GRE) or IP over IP (IPIP) tunnel is required for the transmission of control messages in GMPLS control plane. In DVLSR, each pair of control or data channel interfaces must be within 30-bit mask network address. The routing module computes the peer interface address for a given interface based on the default 30 bit mask. For example, if the local IPIP tunnel interface's IP address is 10.0.100.1, the remote IPIP tunnel's interface IP address must be configured to

10.100.0.2. It caused the interoperability problem of the CVLSR with SN16000 as well as the scalability problem of address assignment of interfaces. SN16000 GMPLS control module requires a single local tunnel interface's IP address for all IPIP tunnels with other GMPLS engines. This 30-bit mask limitation results in only one CLVSR is able to setup an IPIP tunnel with a SN16000 switch, whereas two CVLSRs are not able to be connected to the same SN16000 switch simultaneously. The CVLSR removes this limitation and supports the real point-to-point control or data channel [12] by adding the remote interface into the existing OSPF data structure.

4) Unnumbered data plane interface support

The CVLSR supports both numbered and unnumbered data channel interfaces, whereas unnumbered interfaces were not supported by DVLSR. Since unnumbered interfaces are locally managed, it has the benefits of reducing management overhead and saving the IP addresses resources. Unnumbered addressing is adopted in CHEETAH network.

5) Control channel separation

Finally, it supports the control channel separation, which is defined in RFC 3471 [17]. A single control channel should be able to associate with multiple data channels. It is used to define the parallel TE-links between the CVLSR and SN16000 SONET-XC, and the TE-links between the CVLSR and multiple CHEETAH end hosts attached to the Ethernet switch.

3.5 Signaling Module

The CVLSR signaling module could be signaled for reservation requested by either an end host, or a GMPLS-enabled engine, such as another CVLSR, SONET cross-

connect, MSPP among others. Fig. 3.3 illustrates the diagram of RSVP messages functional flow.

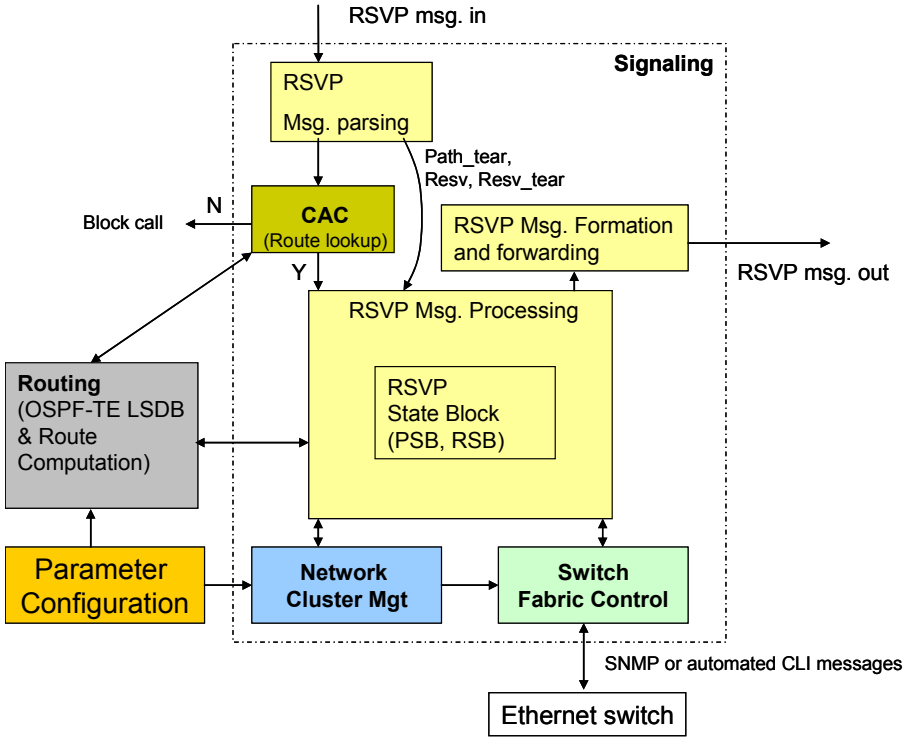

Fig. 3.3 RSVP message functional flow diagram

The RSVP messages will be parsed and processed by the RSVP process sub-module. Two main cases arise here: 1) Upon receiving the Path message, the routing module will be called by CAC sub-module to check whether the request could be admitted or not; based on the admission, the new Path message will be constructed and sent to the downstream node. 2) Upon receiving the Resv message, the switch fabric control sub-module will perform cross connection by sending Simple Network Management Protocol (SNMP) or automatic Command Line Interface (CLI) messages to the Ethernet switch to dynamically set up or tear down VLANs and perform port/VLAN rate-limiting and priority queuing as well. The network cluster management sub-module will handle the cluster connection LSP request.

3.5.1 RSVP Process

The RSVP process sub-module supports the following set of IETF RFCs: RSVP - Version 1 Functional Specification (RFC 2205) [17], RSVP - Version 1 Message Processing Rules (RFC 2209) [34], The Use of RSVP with IETF Integrated Services (RFC 2210) [35], GMPLS Signaling RSVP-TE Extensions (RFC 3473) [36], RSVP Refresh Overhead Reduction Extensions (RFC 2961) [37], RSVP-TE: Extensions to RSVP for LSP Tunnels (RFC 3209) [18], GMPLS Signaling Functional Description (3471) [17], RFC 3477 - Signaling Unnumbered Links in RSVP-TE (RFC 3477) [38], and Procedures for Modifying the RSVP (RFC 3936) [39], etc.

The functionalities of RSVP process sub-module include: (a) sending, receiving, parsing, processing and constructing RSVP messages; (b) building and updating RSVP state blocks; (c) informing the routing module to update its database when bandwidth is reserved/released; and (d) communicating with routing module to obtain the control/data interfaces, switch IP address, VLAN ID if statically configured, and port numbers being cross-connected. Additionally, the Path processing function need modification to avoid re-processing of Path message from DVLSR code, since the path computation is only needed to be done once after receiving the first Path message.

In CVLSR, FSC signaling is implemented for full 1 Gbps connection, whereas L2SC LSP signaling is implemented for sub-1Gbps connection. Since Sycamore SN16000 SONET-XC does not support sub-1Gbps provisioning, a LSP hierarchy [40] solution is used to aggregate multiple sub-1Gbps LSPs by creating a hierarchy LSP. A forwarding adjacency LSP (FA-LSP) is formed out of that LSP and advertised as a TE link so that this FA-LSP can be used for other LSP path computation.

3.5.2 Connection Admission Control

Upon receiving a Path message, CAC determines whether to accept or reject the request based on the availability of the bandwidth and the switching capabilities of the TE-links along the TE path.

In the hop-by-hop routing, CAC sends a query to the routing module to find the next hop node and check the TE information of the link between the Ethernet switch and next hop node. If the unreserved bandwidth and link switching capability are satisfied, the Path request is granted and the connection is admitted, i.e. the CVLSR will process this Path message, construct and send a new Path message to the next hop. Otherwise, the request is rejected. In source routing, CAC requests the OSPF routing module to obtain the ERO through CSPF calculation with the bandwidth and switching capabilities constraints. If an ERO is successfully returned, the request is granted. Otherwise, the request is rejected.

3.5.3 Switch Fabric Control

The switch fabric control sub-module provides the following capabilities: setting up or releasing cross-connection, finding empty or static configured VLAN ID, assuring rate-limiting and bandwidth guarantee. During the processing of Path message, the switch fabric control sub-module determines the appropriate VLAN ID and allocates the input and output ports to be cross-connected. If the VLAN ID is not statically configured, we need to check the usage status of VLAN IDs and find an empty one for cross connection. During the reservation, the RSVP process invokes the switch fabric control to perform cross-connection. The switch fabric control sends the SNMP messages to the Management Information Base (MIB) of Ethernet switch to place the input and output

ports into the tagged or untagged VLAN. For general file transfer application, only one input port and one output port are moved to the specified VLAN. For cluster application, switch fabric configuration module moves not only the port connected to the source or destination end host, but also the other ports connected to all other cluster members into the VLAN.

For multiple CHEETAH application through one switch, the CVLSR must guarantee that, each user can actually obtain the bandwidth it requests, and can not use the extra bandwidth exceeding its request. Following the VLAN setup, the switch fabric control sends SNMP control messages to the switch MIB to put the VLAN in a higher priority queue in order to guarantee the bandwidth of the VLAN as well as limit the rate of input port or VLAN so that the VLAN traffic would not exceed the specified bandwidth. The FSC connection is accomplished via full 1Gbps port mapping, whereas L2SC connection is accomplished via port mapping and VLAN rate limiting to identify sub 1Gbps rate.

3.5.4 Network Cluster Management

A dynamic network cluster is defined as a group of end-hosts in one enterprise network interconnecting with one or more end-hosts in a different enterprise network by an Ethernet point-to-point data path. As shown in Fig. 3.4, A1 and A2 of Enterprise A interconnect with B1 and B2 of Enterprise B via Cluster 1. A network cluster connection is thus a point-to-point one whereby each point is an Ethernet switch or similar switching device.

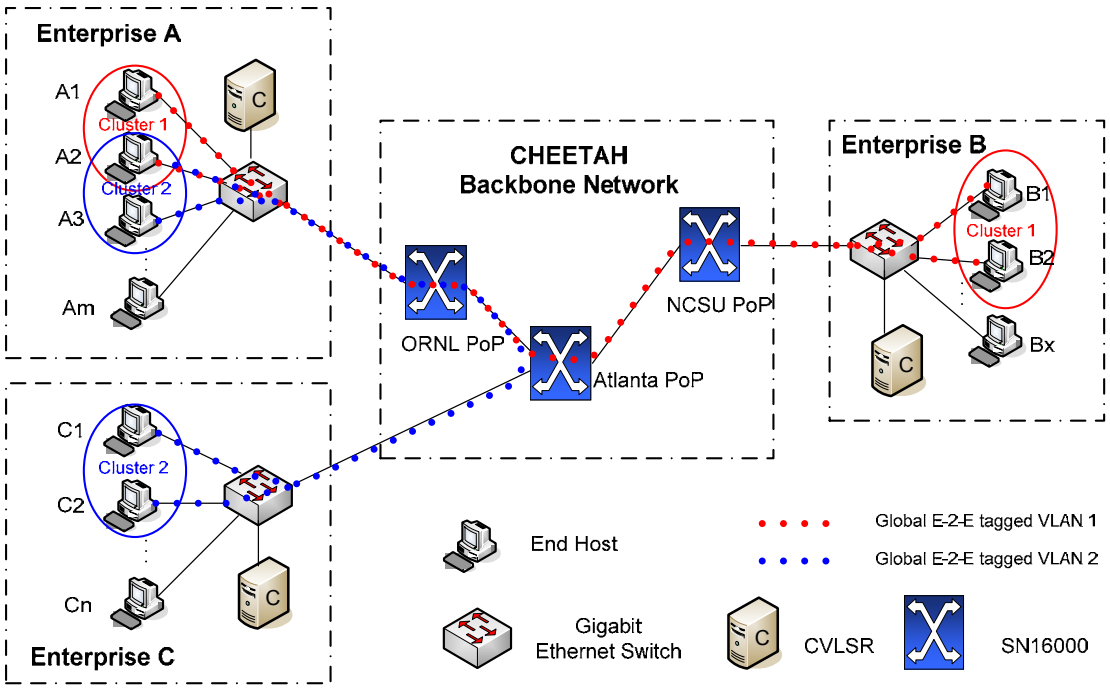

Fig. 3.4 Dynamic network cluster illustration

Any end-host that is authorized to use the clustering function, could initiate a new cluster connection without any prior manual configuration. Once a cluster connection is established, a new host could join the cluster while an existing member could leave. The clustering function could be used in the applications that require several end-hosts to share the same connection. This scenario arises in grid-computing, e-learning, and video-on-demand applications among others. Provisioning connections (at the application level) for such applications could be problematic unless the network provides functionality such as the network cluster. The setup and release of the cluster connection would then relieve the application from the burden of performing a similar function at the application layer.

The setup or release of the cluster is implemented by adding or removing the cluster participants into or from a certain VLAN through the CVLSR. The cluster participants in the same location would equally share the guaranteed bandwidth. Therefore, the cluster membership, i.e., the hosts' IP addresses of both local and remote clusters, should be

passed across the control plane. A new RSVP object with unknown class, called CLUSTER, is defined for this purpose.

```
class CLUSTER_Object {
        uint8 action;
        uint8 labelType;
        SortableList<uint32, uint32>  clusterList;
}
```

The host who initiates the cluster setup determines the initial members of the cluster and constructs the CLUSTER object with both local and remote cluster participants' IP addresses, held in *clusterList*. Then it sends the Path message containing the CLUSTER object to the downstream node. Upon receiving the Path message containing CLUSTER object, the CVLSR will save this object for future cross connection, whereas the other nodes would just forward this unknown object within the downstream Path messages without any modification. Upon receiving the Resv message, the CVLSR checks the cluster members contained in CLUSTER object to search the attached port numbers. The cluster is formed via placing these ports in the same VLAN through SNMP. The network cluster management module then calculates the bandwidth for each cluster member and informs the switch fabric control module to perform rate-limiting for these ports accordingly.

A single end-host could participate in multiple clusters simultaneously. For example, end-host A2 could be a member of Cluster 1 while it is a member of Cluster 2 (shown in Fig. 3.4). The globally tagged End-to-End (E2E) VLANs are used to differentiate each cluster. The CVLSR should be able to find a global VLAN ID that is available to all

nodes along the connection. Two new RSVP objects with unknown class (VLAN_ID_SET and VLAN_ID) are defined for this purpose. The VLAN_ID_SET object is constructed in the Path message by the initiating host. It stores the working VLAN IDs of the local CVLSR that could not be assigned to the new request. When a downstream CVLSR receives the Path message, it adds its own working VLAN IDs into the VLAN_ID_SET object, and then places this new VLAN_ID_SET object in the new Path message to the downstream node. The destination host should choose a VLAN ID, which does not belong to the VLAN_ID_Set, as the global VLAN ID. The VLAN_ID object is constructed with this chosen global VLAN ID, and sent to the upstream node within Resv message. The CVLSRs should use the global VLAN ID specified in the VLAN_ID object within Resv message for the VLAN setup.

Once a dynamic cluster network is established, a new host can join the cluster, while an existing can leave the cluster. Only the sender or receiver of the cluster LSP is capable to add or remove a cluster member. The notification of the change of the cluster members is transmitted to the remote site via trigger Path message [37]. The implementation includes the following steps:

1) The sender or receiver starts a request to add or remove a cluster member into or from an existing network cluster. The new member's IP address will be added to the CLUSTER object, whereas the leaving member's IP address will be removed from the CLUSTER object. The *action* field in the CLUSTER object is defined as the following:

action = 0: setting up the network cluster or tearing down the network cluster.

action = 1: adding new cluster members to the existing network cluster.

action = 2: removing cluster members from the existing network cluster.

2) The host, which starts the request, increases the Minimum path latency field of ADSPEC object by 1, and sends the new Path with updated ADSPEC and CLUSTER objects to the downstream node.

3) The local CVLSR, if there is one, would check its cluster member list to see if the coming/leaving cluster member is locally located. If it is, the CVLSR would appropriately add or remove corresponding port into or from the VLAN, and adjust rate-limiting for each clustering port.

4) The update of ADSPEC object results in the trigger of refresh Path message immediately [17]. Therefore, the change of the cluster can be propagated end-to-end without delay.

5) The remote CVLSR, if there is one, would check its cluster member list to see if the coming/leaving cluster member is locally located. If it is, the CVLSR would appropriately add or remove corresponding port into or from the VLAN, and adjust rate-limiting for each clustering port.

3.6 Interoperability

The CVLSR supports both GMPLS peer and overlay model. The peer model is used when the CVLSR is able to obtain all routing and TE information of the core and enterprise networks, whereas the overlay model is used whenever it is desired to hide the details of the backbone topology from the enterprise networks. Fig. 3.5 shows the topology of the CHEETAH GMPLS peer model. The complete information of physical TE links are provisioned on all control plane interfaces, including the CVLSR. All network devices (CVLSR and SN16000) share a common signaling and routing instance (OSPF-TE and RSVP-TE).

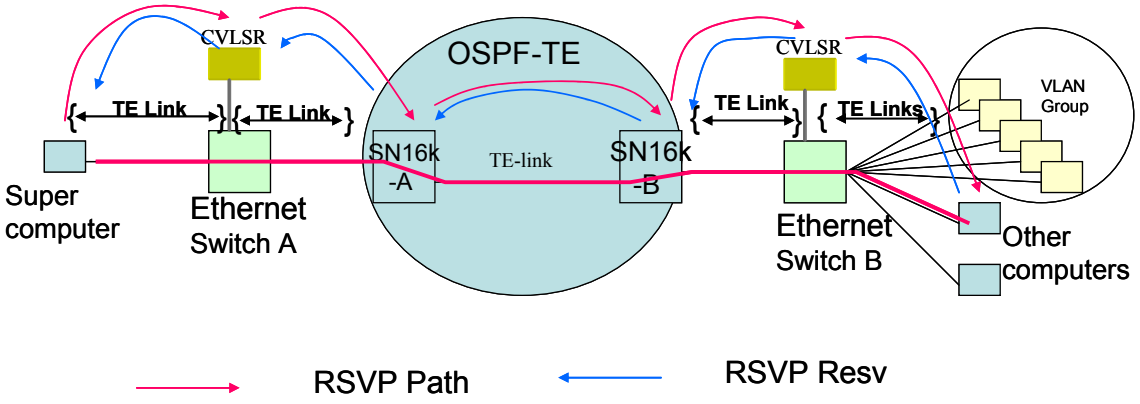

Fig. 3.5 CHEETAH GMPLS peer model

However, the SN16000 switch only supports GMPLS overlay model in current CHEETAH network. Additionally, the DRAGON network is deployed and managed by HOPI/DRAGON administrations. The CVLSR could not obtain the topology and resource information from either of them. The interconnecting can be realized by Boder Gateway Protocol (BGP) or Domain to Domain Routing Protocol (DDRP) or other protocols. In CHEETAH implementation, the CVLSR interoperability is implemented via UNI interfaces provided by the boundary nodes, i.e., SN16000 GMPLS module from CHEETAH backbone network, and DVLSR from DRAGON Network.

Furthermore, the CVLSR should be able to interoperate with the UVa RSVPD client installed on the CHEETAH end hosts.

3.6.1 Interoperability with Sycamore SN16000

In current CHEETAH network, the SN16000 switch only supports GMPLS overlay model. The SN16000 switches use a proprietary GMPLS based routing and signaling protocol, called BroadLeaf (shown in Fig. 3.6), to setup connections within the Sycamore networks I-NNI (Internal Network-Network Interface). The UNI is defined between the Sycamore cloud and CVLSR. The backbone network's topology and resources

information are restricted to identify OSPF adjacencies at the UNI reference points. Virtual TE-links are defined and advertised to identify logical adjacency between SN16000 switches within the I-NNI. The CVLSR is implemented as a UNI client to interoperate with the adjacent SN16000 switch. Without the internal topology and resources information of the Sycamore cloud, the CVLSR adopts hop-by-hop routing to find the next hop gateway.

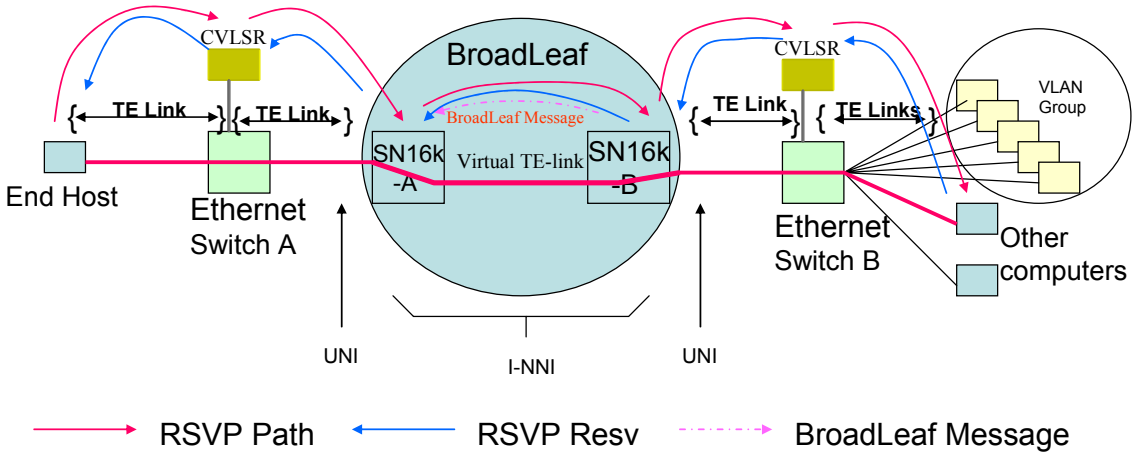

Fig. 3.6 CHEETAH GMPLS overlay model with broadleaf

In CHEETAH network, the area ID of SN16000 back bone network is set to 0.0.0.0, whereas the one for all the CVLSRs is set to 0.0.0.1. The LSAs originally advertised from the network elements across Sycamore cloud are advertised as AS external LSAs by SN16000 SONET-XCs. In order to interoperate with SN16000 SONET_XC, the following issues have been addressed: (a) Hop-by-hop routing for GMPLS overlay model with Broadleaf, (b) DVLSR's 30-bit mask limitation for point-to-point link configuration, (c) Unnumbered data channel interfaces support. They have been discussed in details in Section 3.4.

3.6.2 Interoperability with DRAGON VLSR

The CUNY network is connected to the CHEETAH backbone network via HOPI/DRAGON network, as shown in Fig. 2.5. In order to provide dynamic provisioning capability between CUNY hosts and similar hosts in CHEETAH and DRAGON networks across HOPI/DRAGON network, the CVLSR should be able to interoperate with DVLSR.

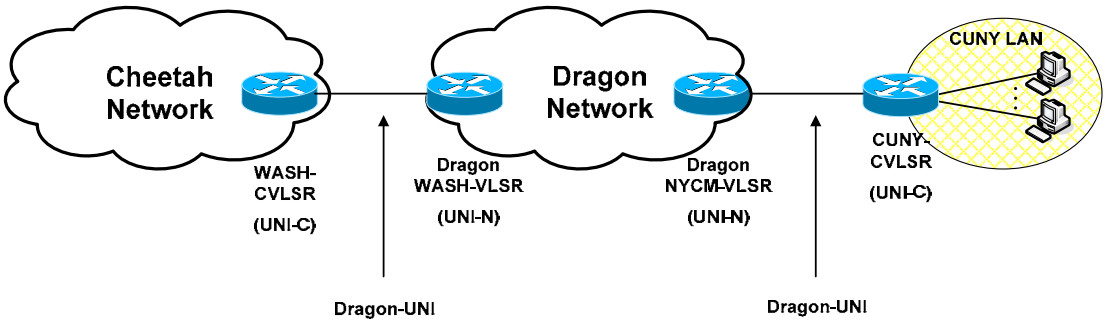

Fig. 3.7 Interoperability of CVLSR with DRAGON-UNI

Two solutions have been developed: one uses UNI, whereas the second utilizes VLAN in VLAN. Fig. 3.7 shows a solution to support dynamic provisioning to inter-connect CHEETAH with DRAGON via DRAGON UNI. In order to protect the internal topology and resources information of DRAGON network from being exposed to the outside, the DRAGON VLSR provides a UNI feature to interoperate with other networks. CUNY-CVLSR and WASH-CVLSR in the figure are deployed as UNI clients to the DRAGON network. Different from the Sycamore UNI, the DRAGON UNI is implemented by adding a DRAGON_UNI object into RSVP Path message. The DRAGON_UNI object specifies the IP addresses, control and data interfaces on both the ingress and egress UNI interfaces. The CVLSR, which works as a DRAGON UNI client, constructs a new Path with DRAGON_UNI object to downstream DRAGON UNI

network node, and processes the Path containing DRAGON_UNI object from upstream DRAGON UNI network node.

The major concern in the development is the session management. In CVLSR, SESSION object from Path message is used as the index of an LSP request, where SESSION object maintains the same destination IP address all the way along the path. However, the SESSION and SENDER_TEMPLATE objects have to be changed to be compatible with DRAGON UNI when CVLSR interoperates with DRAGON VLSR UNI, where the upstream user node sends a Path message with the source address in the SENDER_TEMPLATE object being the upstream network node's loopback IP, and the destination address in the SESSION object being the downstream network node's loopback IP. Therefore, the CVLSR manages the LSPs that coming from or going to DRAGON-UNI via downstream network node's loopback IP. It should be able to convert and process the RSVP messages accordingly as shown in Fig. 3.8. In addition, since the switching capability of CHEETAH circuit could be defined as both S_FSC and S_L2SC whereas only S_L2SC is defined in DRAGON UNI, the CVLSR should be able to convert the switching capability accordingly.

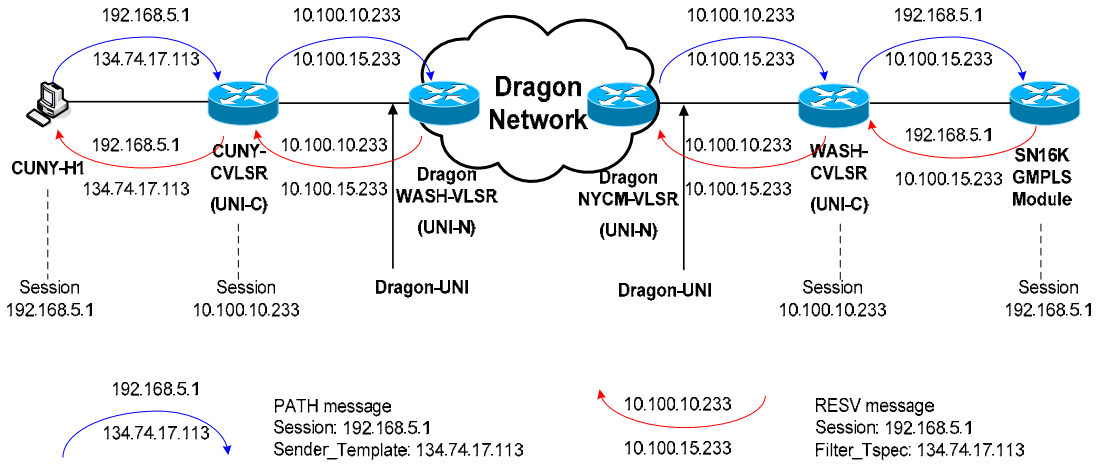

Fig. 3.8 DRAGON UNI SESSION management in CVLSR

A new object CHEETAH_DRAGON is defined for the above purpose:

Class CHEETAH_DRAGON_Object{

NetAddress srcAddress;

NetAddress destAddress;

uint8 switchingType;

}

The DRAGON UNI information, including upstream and downstream DRAGON UNI interfaces, and network address used to identify whether or not they are going to cross HOPI/DRAGON network, is defined by parameter configuration module.

The modification of RSVP processing varies depending on the type of RSVP messages interacting with DRAGON UNI. Upon receiving a RSVP message, the CVLSR would first check whether this LSP is across DRAGON network or not. If the Path is from DRAGON UNI network node, the CVLSR is considered as downstream CVLSR. Otherwise; if the receiver is the CHEETAH end host across DRAGON network, the CVLSR is considered as upstream CVLSR. For all other cases, the CVLSR works in the same way as before.

1) Path message

The upstream CVLSR should create a new SESSION object, whose destination is set to the downstream DRAGON UNI network node. The newly created SESSION object would be the index of this LSP request on this CVLSR in the future. It also should save the actual source and destination IP address, and construct a new Path message, containing DRAGON_UNI object obtained from configuration, CHEETAH_DRAGON object with actual source and destination, SESSION object with destination address being

the downstream DRAGON UNI network node's loopback IP, SENDER_TEMPLATE object with source address being the upstream DRAGON UNI network node's loopback IP, and LABEL_REQUEST Object with switching capability being S_L2SC.

The downstream CVLSR could obtain the actual source and destination IP address from CHEETAH_DRAGON object, and construct a new Path message, containing SESSION object with actual destination address, SENDER_TEMPLATE object with actual source address, and LABEL_REQUEST Object with original switching capability.

2) Resv & ResvTear messages

The upstream CVLSR would construct a new Resv/ResvTear message, containing SESSION object with destination address being actual receiver IP, and FILTER_SPEC object with source address being actual sender IP.

The downstream CVLSR would convert its FILTER_SPEC to the downstream DRAGON UNI network node, and use the new FILTER_SPEC to find matching PSB. It constructs a new Resv/ResvTear message, containing SESSION object with destination address being the downstream DRAGON UNI network node's loopback IP, and FILTER_SPEC object with source address being the upstream DRAGON UNI network node's loopback IP address.

3) PathErr message

The upstream CVLSR would construct a new PathErr message, containing SESSION object with actual destination address, and SENDER_TEMPLATE object with actual source address; LABEL_REQUEST Object with original switching capability.

The downstream CVLSR would convert its SENDER_TEMPLATE to upstream DRAGON UNI network node, and use the new SENDER_TEMPLATE to find matching

PSB. It would construct a new PathErr message, containing SESSION object with destination address being the downstream DRAGON UNI network node's loopback IP, SENDER_TEMPLATE object with source address being the upstream DRAGON UNI network node's loopback IP, and LABEL_REQUEST Object with switching capability being S_L2SC.

4) ResvErr message

The upstream CVLSR would convert its FILTER_SPEC to upstream DRAGON UNI network node, and use the new FILTER_SPEC to find matching PSB. It would construct a new ResvErr message, containing SESSION object with destination address being the downstream DRAGON UNI network node's loopback IP, SENDER_TEMPLATE object with source address being the upstream DRAGON UNI network node's loopback IP, and FILTER_SPEC object with source address being the upstream DRAGON UNI network node's loopback IP.

The downstream CVLSR would construct a new ResvErr message, containing SESSION object with destination address being actual receiver, and SENDER_TEMPLATE object with source address being actual sender.

5) PathTear message

The upstream CVLSR would convert its SENDER_TEMPLATE to upstream DRAGON UNI network node, and use the new SENDER_TEMPLATE to find matching PSB. It would construct a new PathTear message, containing SESSION object with destination address being the downstream DRAGON UNI network node's loopback IP, and SENDER_TEMPLATE object with source address being the upstream DRAGON UNI network node's loopback IP.

The downstream CVLSR would construct a new PathTear message, containing SESSION object with destination address being actual receiver, and SENDER_TEMPLATE object with source address being actual sender.

6) ResvConf message

The upstream CVLSR would find the session using the SESSION object with destination being downstream DRAGON UNI network node. It sends a new ResvConf message with new SESSION object to the upstream DRAGON UNI.

The downstream CVLSR would find the session using the same SESSION object in the received message. It sends a new ResvConf message with new SESSION object with actual receiver to the next hop.

In VLAN in VLAN solution, one IEEE 802.1Q VLAN tag is encapsulated within another 802.1Q Ethernet frame through IEEE 802.1Q-in-Q VLAN tagging, thus producing a double-tagged Ethernet frame. Via this solution, the HOPI/DRAGON network can use a single VLAN to support multiple CHEETAH VLANs so that all these VLANs may share this HOPI/DRAGON GbE connection.

3.6.3 Interoperability with UVa RSVPD Client

The CHEETAH end host is not equipped with OSPF routing module. The TE information between the CVLSR and end host, such as the end host IP address, maximum reservable bandwidth, data interface ID, and port number attached to the Ethernet switch, is manually configured and managed by the corresponding CVLSR.

The end host initiates an LSP request through UVa RSVPD client, which sends Path message directly to the CVLSR. The problem arises when the CVLSR receives an LSP request ended with a receiver CHEETAH end host attached to the Ethernet switch

controlled by another CVLSR. Since the CHEETAH end host is not configured as the router node, the routing module needs to set the destination to be the remote CVLSR that is neighbored to the receiver end host in order to compute the path. After the path to remote CVLSR is computed, additional work has to be done for the source routing, i.e., adding the receiver end host data channel interfaces to the explicit route to form a full ERO.

Another problem is the GRE or IPIP tunnel required for the transmission of control messages in GMPLS control plane. In order to reduce the system management cost, neither GRE nor IPIP tunnel is configured between the end host and CVLSR. The CVLSR is modified to adapt to both cases: with and without tunnel configurations. For the case without tunnel configuration, the CVLSR uses the same control interface to communicate with all neighboring end hosts, i.e., the physical broadcast interface of CVLSR is used for the signaling messages exchanged with the end hosts. The algorithm used to identify the end hosts is based on the control channel separation defined in RFC 3471 [17].

3.7 Security Design

Security is an important issue in CHEETAH network. There are four main requirements concerning network security:

1) Authentication: proving the user is who he/she claims he/she is. Only authorized users can be allowed to use network resources.

2) Confidentiality/Privacy: ensuring the message is not disclosed. Only intended receivers can understand the message.

3) Integrity: assuring the received message is not altered during transmission.

4) Non-repudiation: proving the message is really sent by the sender. The sender can not deny having sent a given message.

In CHEETAH network, the control messages are transmitted over the internet. The routing and signaling messages should be secured through a hop-by-hop secure strategy. The CHEETAH security mechanism is implemented via a joint of Juniper Netscreen-5XT (NS5XT) firewall [41] and Openswan software [42] to ensure the privacy and integrity of these messages. Additionally, in order to assure that only CHEETAH users have privileges to use CHEETAH circuit, the AAA services are applied for user administration and accounting.

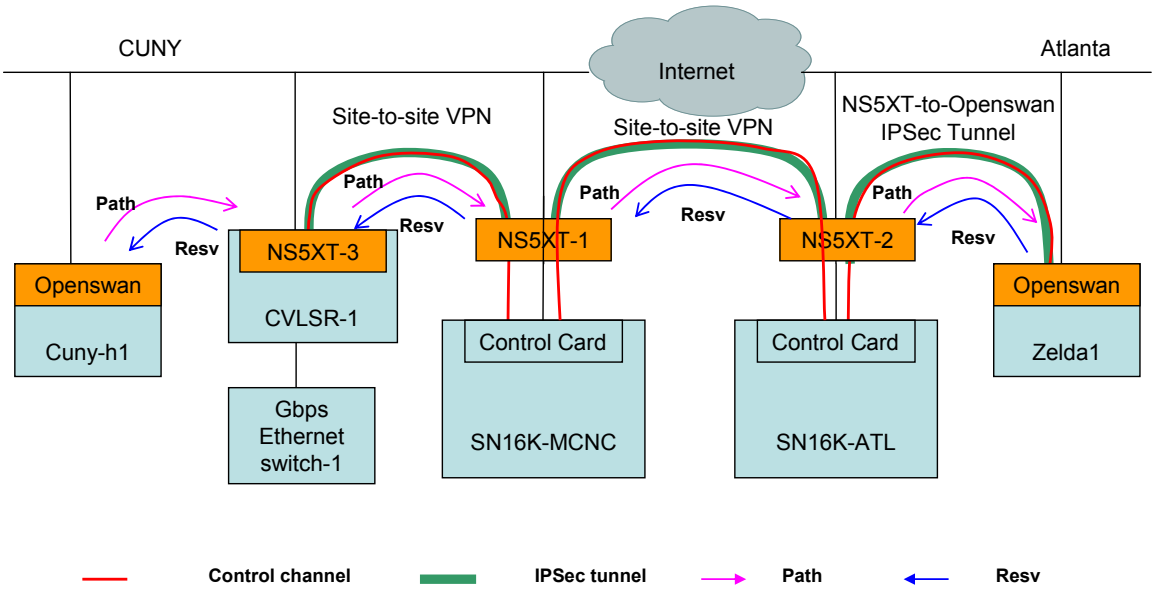

Fig. 3.9 CHEETAH security mechanism

3.7.1 Security in Control Plane

Fig. 3.9 illustrates the security mechanism applied in CHEETAH network. Along the path of RSVP Path/Resv signaling messages, IP Security (IPSec) tunnels are established to prevent unwanted traffic reaching the routing and signaling nodes. A Juniper

Netscreen-5XT (NS5XT) firewall is placed in front of the Ethernet control port of each SN16000 switch, or the CVLSR in CUNY. The Opensware software is installed on the end host that is remotely connected to the SN16000 switch or CVLSR.

In CHEETAH network, the NS5XT is configured as the VPN gateways so that it can establish the secure IPSec tunnels with the remote sites. The control port on the switch is connected by direct cable to the NS5XT and is thus intrinsically secure. The control messages between SN16000 switches (or CVLSRs) are secured by site-to-site VPN tunnels. The route-based VPN is configured on the NS5XT, where a specific VPN tunnel is referenced by the route pointing to the specific tunnel interface bound to the VPN tunnel.

Openswan is an open source implementation of IPSec (IP Security) for Linux operating system. In addition to setting up the IPSec tunnels between Linux systems, it is known to be able to interoperate with other existing IPSec and Internet Key Exchange (IKE) systems to provide IPSec tunnel between Linux host and other IPSec and IKE systems. Via the interoperability capability of Openswan with NS5XT IPSec, a gateway-to-gateway IPSec tunnel between Linux system and NS5XT can be established to secure the control messages between SN16000 switch and Linux end hosts.

3.7.2 Security in Data Plane

Since CHEETAH signaling establishes an on-demand provisioning of end-to-end high-speed circuits, this dedicated link is considered to be secure. Further per-user protection on the data-plane is left up to the applications that use CHEETAH circuits. For example, if secure data transfers are required, user can use Secure File Transfer Protocol (SFTP) or GridFTP.

3.7.3 CHEETAH User Authentication and Accounting

In CHEETAH network, only CHEETAH users have privileges to use CHEETAH circuits, therefore we need an authentication protocol to authenticate and authorize the CHEETAH users. Remote Authentication Dial-in User Service (RADIUS), one of the most commonly used Authentication Authorization and Accounting (AAA) services, is chosen for CHEETAH user authentication and accounting.

We use the open source software FreeRADIUS [43] for the RADIUS protocol implantation. FreeRADIUS is within the top 5 RADIUS servers world-wide, in terms of the number of people who use it daily for authentication. It can support systems with millions of users. FreeRADIUS also contains dictionary files that support for PostgreSQL database [44], that is used for the user authentication database and accounting database in our implementation. PostgreSQL is an open source software object-relational database management system. It has most necessary features of the commercial database like Oracle, and is more suitable than MySQL for read-intensive databases such as dynamic websites for enterprise users.

Currently, the centralized authentication is applied in CHEETAH network. Before a CHEETAH end host initiates a connection request of CHEETAH circuits, it needs to query the authentication server for authorization for itself and the peer end host. The problem of this centralized solution is that, when the number of CHEETAH users goes large such as tens of thousands, the authentication and accounting as well as the transmission of query messages would cause a big burden to the network. Multiple regional AAA servers plus Lightweight Directory Access Protocol (LDAP) could be our future solution.

Chapter 4

CVLSR Deployment and Test Results

The main goal of our efforts in this chapter is to demonstrate through experimental test-beds the concept of fast file transfers on an end-to-end basis using circuit switching technology. The chapter begins with the plan and implementation of CHEETAH network data plane connectivity. The CVLSR deployment and several test scenarios in CHEETAH network across HOPI/DRAGON network are illustrated. We have successfully deployed the CVLSR in experimental CHEETAH optical network across HOPI/DRAGON network. The interoperability of the CVLSR with commercial GMPLS SONET-XC (Sycamore SN16000 switch), DRAGON VLSR, and UVa RSVPD client has been demonstrated. The chapter ends by discussing the test results and throughput measurements.

4.1 Data plane Connectivity

As shown in Fig. 4.1, CHEETAH data plane backbone network consists of three CHEETAH PoP equipped with Sycamore SN16000 SONET-XCs. These three SN16000 switches are interconnected by long distance OC-192 SONET circuits. The 10 Gbps Ethernet connections link several enterprise networks such as the supercomputer at ORNL and the Centaur physics laboratory at NCSU. Other connections include several 1 Gbps Ethernet links to CUNY through Internet2 HOPI/DRAGON network, to University

of Virginia (UVa) and other enterprise networks as illustrated in the Figure. The end hosts, in each enterprise network, gain the access to CHEETAH network via Gigabit Ethernet switch.

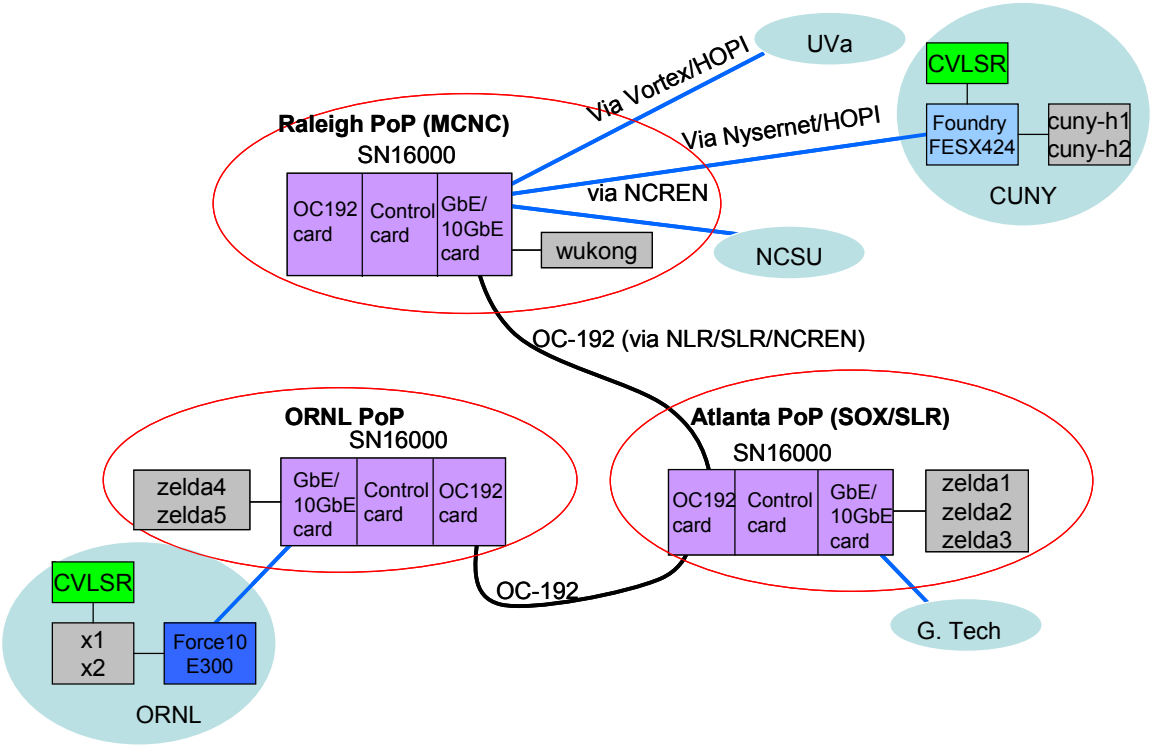

Fig. 4.1 CHEETAH Network overview

Each SN16000 switch is equipped with a GbE interface card containing 10 GbE ports. Each end host is equipped with two Ethernet Network Interface Cards (NICs). One card (10/100 Mbps) is used for Internet services including control messages transmission while the other one (GbE) is used for data plane forwarding. The CHEETAH client host can be attached to either the SN16000 GbE port or the Ethernet switch via the secondary NIC. The figure shows several of the end hosts. The end hosts, wukong in MCNC, zelda1, zelda2, zelda3 in SOX/SLR, and zelda4 and zelda5 in ORNL, are physically located at the three PoP sites and directly connected to the SN16000 switches. Whereas the end hosts, cuny-h1 and cuny-h2 in CUNY, and x1 and x2 in ORNL, are attached to the Ethernet switches. The introduction of CVLSR allows these end hosts to dynamically

share the backbone network resources. It offers a scalable and cost-efficient solution for adding users to the network, whereas adding users directly to PoP nodes will quickly exhaust the capacity of the PoP GbE ports.

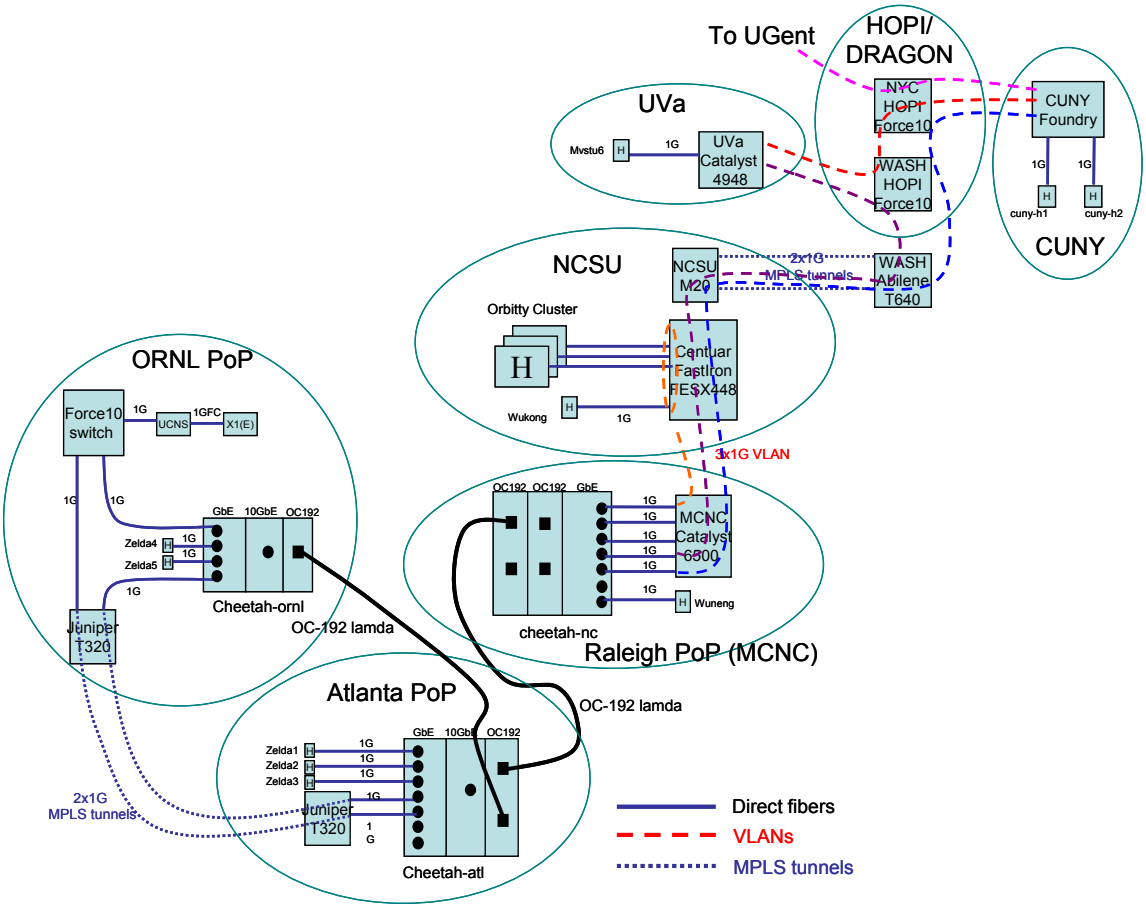

Fig. 4.2 CHEETAH network data plane

Since the above enterprises/campuses are geographically distant from the CHEETAH PoPs, it is not feasible to connect the end hosts to the SN16000 switches via direct physical connections. Therefore, the CHEETAH data plane with Virtual Local Area Networks (VLANs) and/or MPLS connections were set up to connect the NCSU, UVa and CUNY networks to the CHEETAH backbone network. Fig. 4.2 illustrates the complete cheetah network data plane topology. Several layer 2 tunnels have been brought

up for the connections between CUNY and CHEETAH, between CUNY and UVa, between UVa and CHEETAH, and between NCSU and CHEETAH.

The CUNY connection to CHEETAH network was implemented via a VLAN path, which initiated at CUNY Foundry switch, and terminated at MCNC Catalyst 6500 before SN16000 in MCNC, across NYSERNet, HOPI, Abilene and NCREN networks. A 1 Gbps Ethernet connection from CCNY to the HOPI force10 switch at 32 Avenue of Americas (AoA) collocation facilities in New York was completed by utilizing existing NYSERnet infrastructure (as shown in Fig. 4.3). To that effect, we have purchased, installed, and provisioned 4x1 GbE cards in the Cisco MSPPs located in the NYSERnet path that is owned by City College.

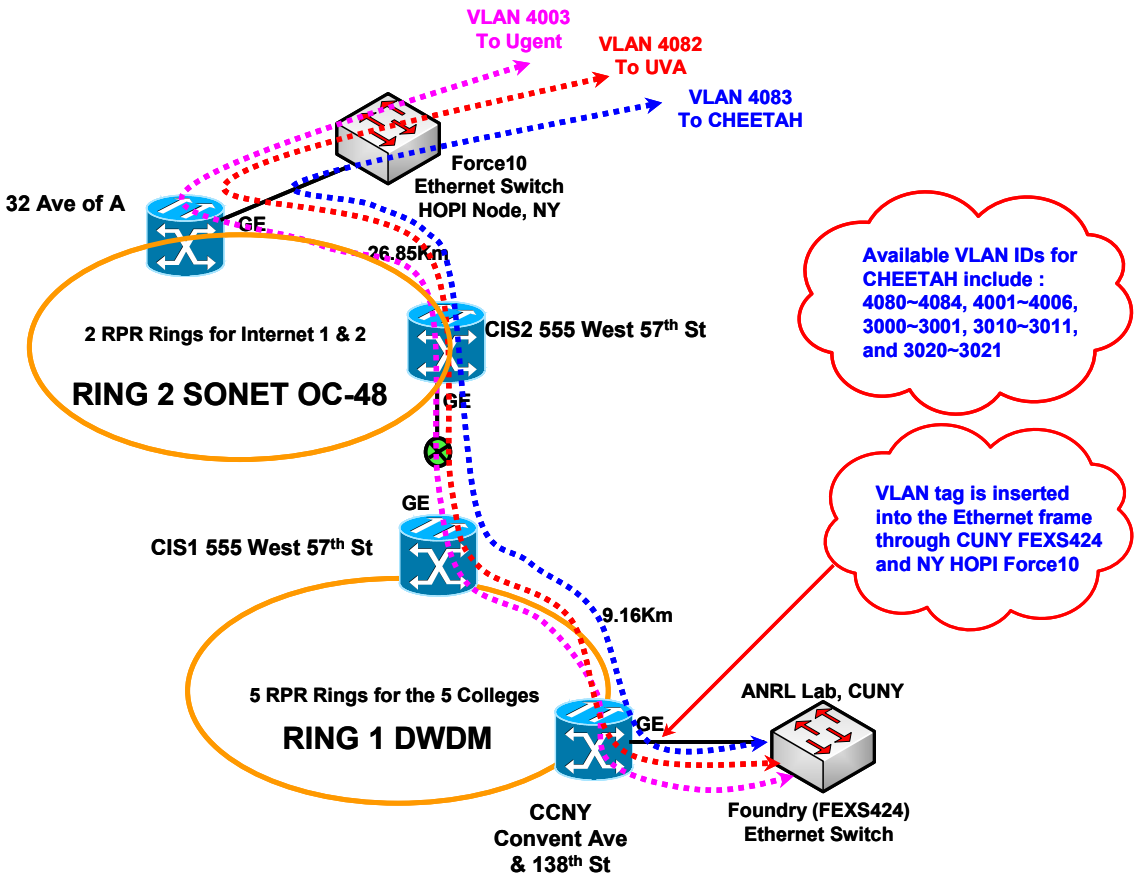

Fig. 4.3 City College connectivity to HOPI via NYSERnet

Three VLANs were created to establish the layer 2 tunnels connected to CHEETAH backbone network (VLAN 4083), UVa (VLAN 4082), and University of Ghent (UGent), Belgium (VLAN 4003). These three VLAN tunnels were statically configured.

The connection from UGent to CHEETAH network through HOPI, MANLAN, GEANT, and Belgian NREN (Belnet) was also implemented (Fig. 4.4). This connection can be used for dynamic provisioning between UGent users and CHEETAH users. This connection will serve us in the future to demonstrate the efficacy of our solution to support new applications such as grid computing and video teleconferencing on a global scale.

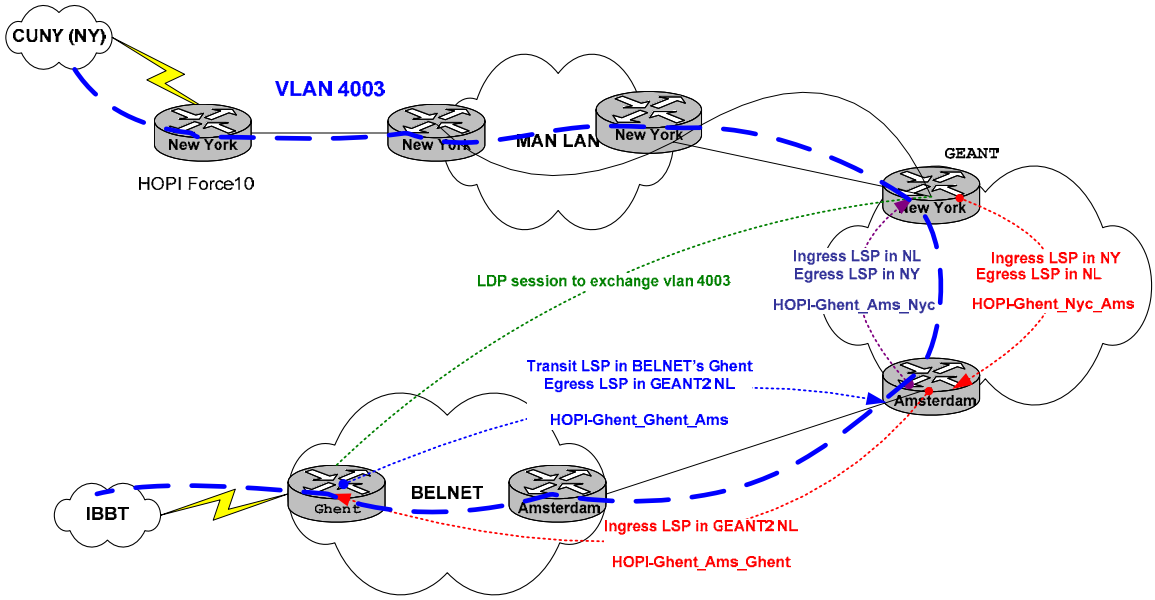

Fig. 4.4 Connectivity between CUNY and UGent

4.2 CVLSR Deployment and Test Scenarios

The CHEETAH deployment was planned based on the network topology shown in Fig. 2.5. As in the figure, two CVLSRs are deployed in the experimental CHEETAH network. One CVLSR is deployed as the GMPLS engine to the Foundry FEXS424 Ethernet switch located in CUNY so that it could dynamically provision guaranteed

bandwidth to CUNY end hosts. Similarly, the other CVLSR is deployed in ORNL, and it could do so to the ORNL end hosts. In this section, we first discuss how to set up CHEETAH control plane. Several specific testing scenarios were planned and carried out for the demonstration based on the CHEETAH network topology, i.e., local CVLSR deployment, experimental CVLSR deployment in CHEETAH network, and interconnectivity of CVLSR with HOPI/DRAGON test scenarios.

4.2.1 CVLSR Control Plane Setup

Deploying the CVLSR in the CHEETAH control plane consists of the following steps:

1) Plan the control plane and configure the control channels. The control channels are used for OSPF-TE and RSVP-TE control messages, such as OSPF Hello messages, OSPF link state advertisements, and RSVP signaling messages, etc. These out-of-band control channels are created by configuring GRE or IPIP tunnels between adjacent control nodes. The CVLSR supports control channel separation in contract to the previous one-to-one control/data channel association. The control channel separation has the benefits of reducing management overhead and saving the IP address resources.

2) Plan the data plane and configure the TE-link information, including data interface IDs, available bandwidths and switching capabilities, etc. Each pair of data interfaces between the CVLSR and the host or SN16000 represents one TE-link. Both numbered and unnumbered data channel interfaces are supported in CHEETAH network. Similar to the control plane separation, the unnumbered interface configuration has the benefits of reducing management overhead and saving the IP addresses resources.

3) Configure other parameters, such as DRAGON UNI interface configuration, mapping of end hosts and port numbers attached to the Ethernet switch, and cluster information, etc.

NS5XT firewall and Openswan software are installed to establish the IPSec tunnels for control channels. Such IPSec tunnels are brought up between SN16000 switches in MCNC and Atlanta, between SN16000 switches in Atlanta and ORNL, between SN16000 switch in MCNC and CUNY-CVLSR, between SN16000 switch in Atlanta and Zelda end hosts. There are no IPSec tunnels configured between CUNY-CVLSR and CUNY end hosts because they are physically located in the same room.

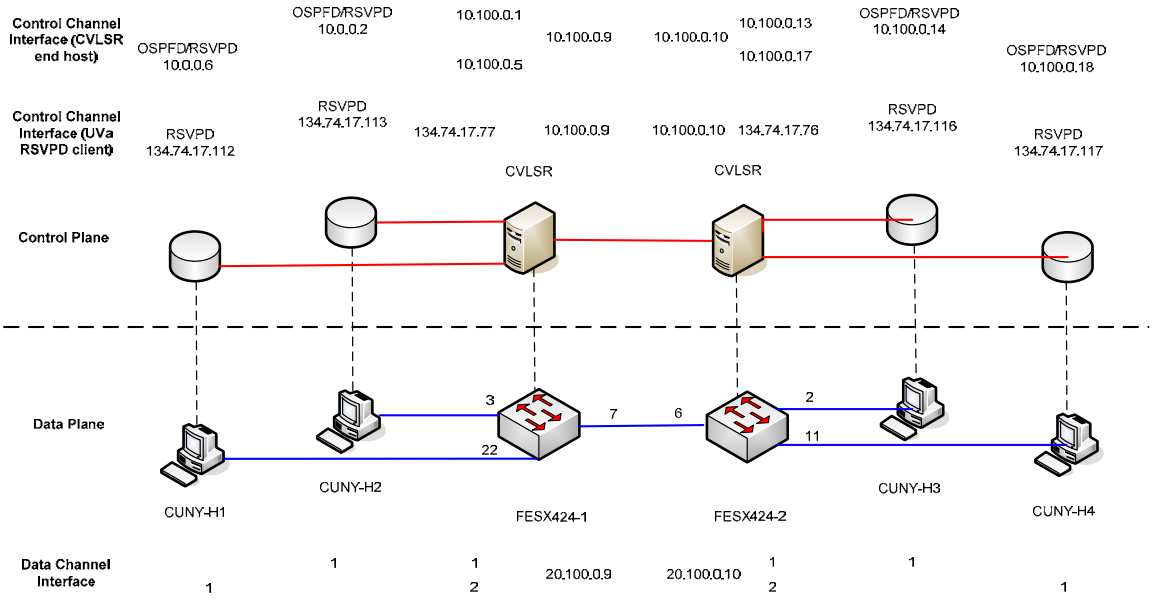

Fig. 4.5 Configuration of local CVLSR deployment testbed

4.2.2 Local CVLSR Deployment Test Scenario

Fig. 4.5 shows the topology where only two CVLSRs are locally deployed to provide dynamic provisioning for the end hosts attached to the local Ethernet switches. Since we only have one Ethernet switch in the lab, it is configured as logical two switches. The

control channel interfaces are configured depending on which software is running on the end hosts, UVa RSVPD client, or CVLSR software suite.

In this scenario, the CVLSR was confirmed to be capable of enabling the Ethernet switch to participate in the dynamic provisioning:

1) The CVLSR is capable of interoperating with the UVa RSVPD client installed on the CHEETAH end hosts where requests for bandwidth are generated via the UVa RSVPD client. The control messages between the CVLSRs are transmitted through the specified point-to-point control channel interfaces. On the other hand, the ones between the CVLSRs and end hosts are transmitted through the broadcast Ethernet interfaces, where no control channel interface needs to be configured. Only full 1 Gbps bandwidth is supported by the UVa RSVPD client currently. For example, the LSP request destined to CUNY-H4 is initiated on CUNY-H1 by command: *bwrequestor 134.74.17.117 1000*.

2) In order to support sub-1 Gbps bandwidth request and network clustering function, the end host should be equipped with CVLSR software suite. All control messages are transmitted through control channel interfaces, which are configured on both CVLSRs and end hosts. The LSP request is generated by the DRAGON interface defined by the routing module. Both 1 Gbps FSC and sub-1 Gbps L2SC LSPs are supported. The following is an example of 500 Mbps L2SC LSP request generated from CUNY-H1 and destined to CUNY-H4:

cuny-cvlsr> edit lsp test500M

cuny-cvlsr(edit-lsp-test500M)# set source ip-address 134.74.17.112 lsp-id 5001 destination ip-address 134.74.17.117 tunnel-id 5000

cuny-cvlsr(edit-lsp-test500M)# set bandwidth eth500M swcap fsc encoding ethernet gpid ethernet

cuny-cvlsr(edit-lsp-test500M)# set direction bi upstream-label 65536

cuny-cvlsr(edit-lsp-test500M)# exit

cuny-cvlsr> commit lsp test500M

3) Both numbered and unnumbered data channel interfaces are supported. Since unnumbered interface IDs are locally managed, the adoption of unnumbered data plane interfaces greatly reduces the addressing management cost and improves the scalability of CHEETAH system.

4) It has been demonstrated the switch fabric control function of CVLSR: adding or removing ports into or from the VLAN; dynamically assign an empty VLAN ID if no static VLAN ID is configured; setting higher queuing priority to the CHEETAH traffic over other traffic to guarantee the reserved bandwidth; and setting rate limiting to provision sub-1Gbps Ethernet VLAN connections, etc.

5) Dynamic network cluster function has also been demonstrated in the test. Any end-host that is authorized to use the clustering function, could initiate a new cluster connection:

cuny-cvlsr> edit lsp cluster

cuny-cvlsr(edit-lsp-cluster)# set source ip-address 134.74.17.112 lsp-id 5001 destination ip-address 134.74.17.117 tunnel-id 5000

cuny-cvlsr(edit-lsp-cluster)# set direction bi upstream-label 65536

cuny-cvlsr(edit-lsp-cluster)# set bandwidth gige swcap fsc encoding ethernet gpid ethernet

cuny-cvlsr(edit-lsp-cluster)# set cluster-set 134.74.17.112

cuny-cvlsr(edit-lsp-cluster)# set cluster-set 134.74.17.113

cuny-cvlsr(edit-lsp-cluster)# exit

cuny-cvlsr> commit lsp cluster

Once a cluster connection is established, a new host could join the cluster while an existing member could leave:

cuny-cvlsr> add lsp cluster 134.74.17.116

cuny-cvlsr> remove lsp cluster 134.74.17.116

The fabric control functions on Foundry switch were validated accordingly.

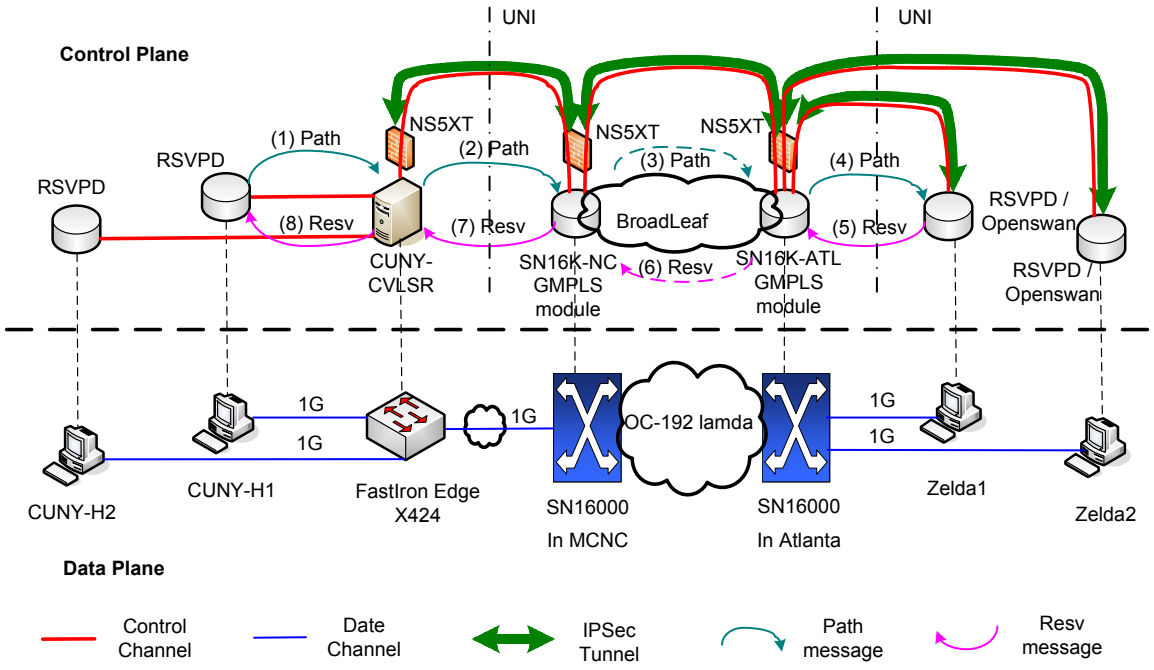

Fig. 4.6 Experimental CVLSR deployment in CHEETAH network testbed

4.2.3 Experimental CVLSR Deployment in CHEETAH Network Test Scenario

Fig. 4.6 shows one of the experimental CVLSR deployment scenarios. The data plane is interconnected via SN16000 in MCNC, SN16000 in Atlanta, and Gigabit

Ethernet switch in CUNY. The control plane consists of the SN16000 GMPLS modules, CVLSR, and RSVPD running on the CHEETAH end hosts.

In the control plane, since current version of SN16000 switch does not support peer model, the GMPLS overlay model is adopted in CHEETAH network. Sycamore uses Broadleaf for routing and signaling protocol in the Sycamore backbone cloud. The backbone network's topology information and links' bandwidth information are not advertised, thus hop-by-hop routing is adopted by the CVLSR to find the path to the destination end host.

Out-of-band control channels are setup via IPIP tunnels over the IPSec tunnels, which are established along the connectivity of control channels to secure the control messages (i.e., OSPF and RSVP messages). Two different security schemes are applied along the path of the control messages in this scenario: 1) Two IPSec tunnels are established between SN16000 switches in MCNC and Atlanta, between SN16000 in MCNC and CUNY-CVLSR via NS5XT firewalls, where all NS5XTs are configured as the VPN gateways; 2) The IPSec tunnel is established between the NS5XT firewall and each end host in Atlanta, where both the NS5XT and end host are configured as the IPSec gateways. The gateway on the Linux host is configured through Openswan software.

Each end-host has two interface cards: a primary 1 Gbps for data plane connectivity, and a secondary one for TCP/IP connectivity as well as for transport of control messages. The end host is equipped with UVa RSVPD client for the signaling. The establishment of LSP request can be described by an example of the LSP initiated by CUNY-H1 and destined to Zelda1. The CUNY end host CUNY-H1 generates and sends a RSVP Path message without ERO to CUNY-CVLSR. The CUNY-CVLSR processes this message

and constructs new Path message to the UNI interface of SN16000 in MCNC. The SN16000 I-NNI then uses Broadleaf to setup the connection within Sycamore cloud, and sends a new Path to the peer end host, i.e., Zelda 1. Zelda1 returns a Resv message back to SN16000 in Atlanta upon the successful Path labels reservation. Once CUNY-CVLSR receives the Resv message from SN16000 in MCNC, it performances the cross connection by placing the port that CUNY-H1 attached into the VLAN 4083. In order to ensure the Quality of Service (QoS) of this connection, the higher queuing priority and rate limiting are performed. The CHEETAH traffic is transmitted with higher privilege than other (TCP/IP) traffic through the Ethernet switch, while it would not be able to use the bandwidth exceeding what it requires. The dedicated end-to-end path is established at this moment.

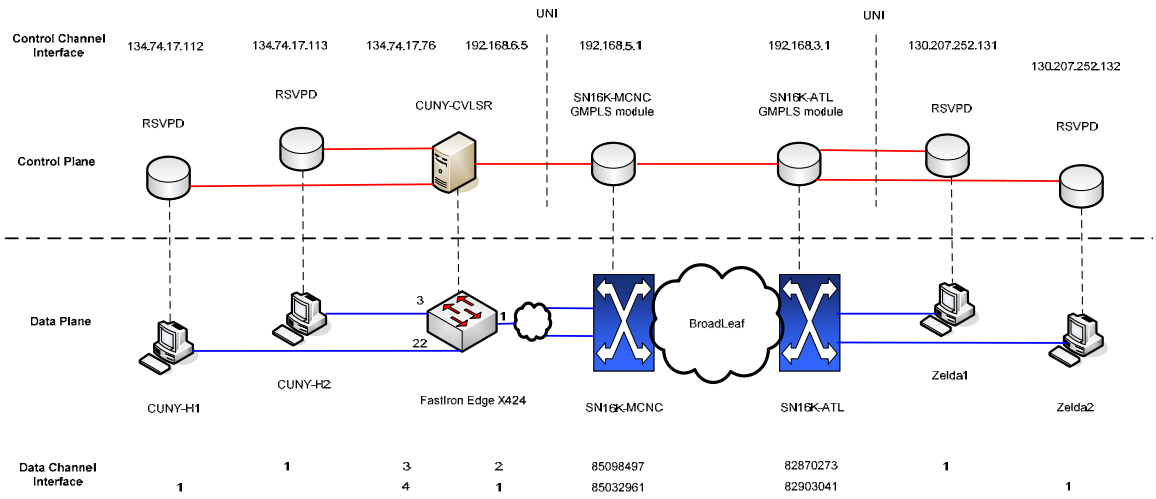

Fig. 4.7 Configuration of experimental CVLSR deployment testbed

Fig. 4.7 illustrates the configuration of experimental CVLSR deployment in CHEETAH network testbed shown in Fig. 4.6. The CVLSR interoperate with SN16000 SONET-XC via UNI. The CHEETAH end hosts are equipped with UVa RSVPD client only. Two TE links are configured between CUNY-CVLSR and SN16000 in MCNC.

In this test scenario, the following issues have been tested to demonstrate the interoperation of the CVLSR engine with the GMPLS module of Sycamore SONET-XC switch:

1) Supporting hop-by-hop routing. Since Sycamore SN16000 backbone network uses Broadleaf for routing and signaling, the CVLSR could not obtain the ERO to the destination. The next hop is computed via hop-by-hop routing, where its bandwidth availability is checked for CAC.

2) Removing 30-bit mask limitation. The DVLSR code limits the point-to-point link to be within 30-bit mask, whereas the local IPIP tunnel interface's IP address is fixed in SN16000 switch. This 30-bit mask limitation results in only one CLVSR is able to setup an IPIP tunnel with a SN16000 switch, whereas two CVLSRs are not able to connect to the same SN16000 switches simultaneously.

3) Supporting control channel separation: the SN16000 switch adopts the unnumbered data channel interface in CHEETAH network. In order to support multiple TE links between the CVLSR and SN16000 switch, one control channel is associated with multiple data channels.

4) Identified two interoperability problems for parallel LSP requests processing. (a) RSVP path resolving problem: SN16000 always picked one specific port even though it was occupied and other ports were available. (b) RSVP refresh problem: the MESSAGE-ID of one of the LSPs in Srefresh message was set to 0, which resulted in the tearing down of this LSP. The second problem was fixed in the latest release of SN16000 code, whereas the first problem is currently being studied by Sycamore.

In addition to the UVa RSVPD client, the end host can also be equipped with the CVLSR software suite. It is used for the network clustering and sub-1G LSP requests which are not supported by current version of UVa RSVPD client.

4.2.4 CVLSR Deployment for Interconnectivity with HOPI/DRAGON Test Scenario

As shown in Fig. 4.2, the dynamic provisioned connections from CUNY end hosts to the other CHEETAH end hosts pass through the HOPI/DRAGON network. The GMPLS engine adjacent to the CUNY-CVLSR was the SN16000 GMPLS module in MCNC. The CUNY-CVLSR sends the Path messages initiated from CUNY end hosts to SN16000 in MCNC, and vice versa to the ones initiated from the other CHEETAH end hosts (shown in Fig. 4.6). The HOPI/DRAGON portion was statically configured. Clearly, the HOPI/DRAGON did not participate in the dynamic provisioning.

We designed and implemented an interconnectivity solution to deploy dynamic provisioning of end-to-end connections between CUNY, and other nodes that are reachable to CHEETAH or DRAGON networks via the HOPI network. The dynamic provisioning solution is achieved via the interoperability of the CVLSR and DVLSR via two possible solutions: one is based upon a UNI technology, whereas the second uses the VLAN in VLAN technique.

In the UNI solution (shown in Fig. 4.8), the CUNY-CVLSR acts as the GMPLS engine of CUNY Ethernet switch interacting with DVLSR that is used as GMPLS engine of HOPI Force10 Ethernet switch located in New York. WASH-CVLSR is only a logical CVLSR, and it is physically located in CUNY lab. The purpose of this deployment is for the interoperating with the DVLSR in Washington, D.C., and SN16000 in MCNC, because DVLSR is not capable of interoperating with SN16000 GMPLS module.

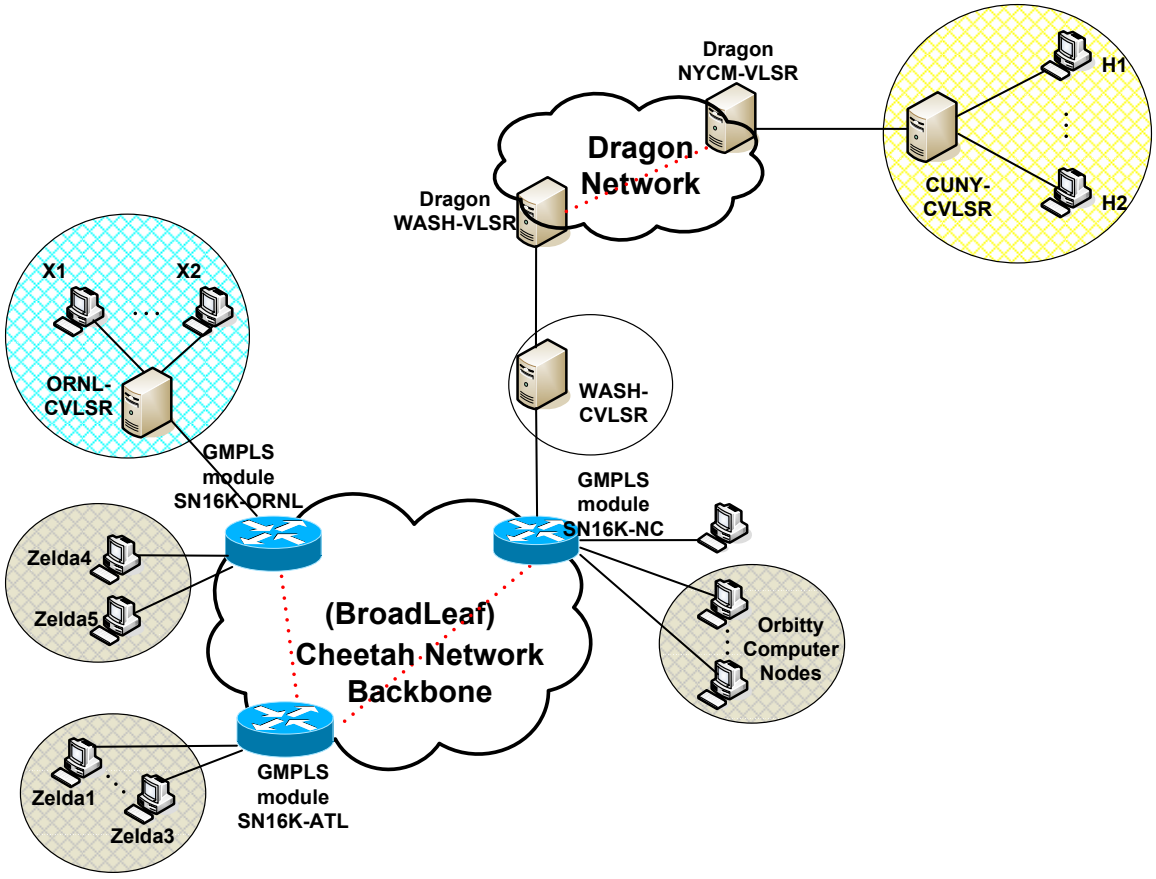

Fig. 4.8 CVLSR Deployment for inter-connectivity with HOPI/DRAGON testbed

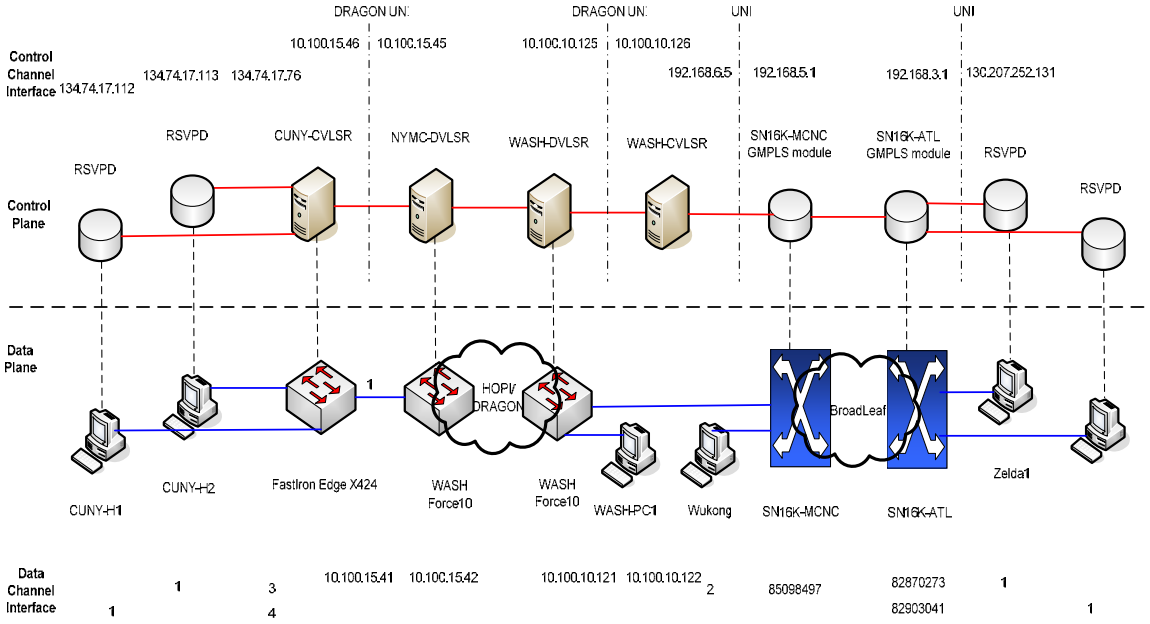

Fig. 4.9 Configuration of inter-connectivity with HOPI/DRAGON testbed

Instead of sending the Path message to SN16000 in MCNC as shown in scenario Fig. 4.6, CUNY-CVLSR sends the Path message to NYCM-DVLSR. NYCM-DVLSR will find the route via NARB based on the hybrid dynamic provisioning architecture that DRAGON uses (as shown in Fig. 2.4). The interoperability of CVLSR with DVLSR is realized through DRAGON UNI solution. Via this UNI solution, the CUNY end hosts are able to dynamically establish the end-to-end connections through HOPI network to the other CHEETAH end hosts located in NCSU, Atlanta, and ORNL. Meanwhile, the CHEETAH end hosts can dynamically setup LSP with the DRAGON end hosts via interconnecting of CVLSR and DVLSR. Fig. 4.9 illustrates the configuration of CVLSR Deployment for inter-connectivity with HOPI/DRAGON testbed shown in Fig. 4.9.

In this test scenario, the following issues have been demonstrated:

1) The CHEETAH end host is able to request dynamic provisioned LSP destined to the DRAGON end hosts. For example, the CUNY-H1 may initiate an LSP destined to WASH-PC1 that is attached to the HOPI Force10 Ethernet switch located in Washington, D.C. WASH-PC1 here is configured as VLSR-to-VLSR provisioning, which utilizes the DRAGON UNI feature and configures the end host and DVLSR as the UNI client and network sides respectively. In DRAGON VLSR-to-VLSR provisioning, the edge switch ports is attached to the LSP using the DRAGON Local ID feature, including port number and VLAN tag. The DRAGON end host does not participate neither routing nor signaling at all. The CHEETAH end host here must be equipped with the entire CVLSR software suite. The LSP request is initiated by:

cuny-cvlsr> edit lsp todragon

cuny-cvlsr(edit-lsp-todragon)# set uni client ingress implicit egress implicit

cuny-cvlsr(edit-lsp-todragon)# set source ip-address 134.74.17.112 group 1000 destination ip-address 10.100.20.233 group 2000

cuny-cvlsr(edit-lsp-todragon)# set direction bi upstream-label 65536

cuny-cvlsr(edit-lsp-todragon)# set bandwidth gige swcap fsc encoding ethernet gpid ethernet

cuny-cvlsr(edit-lsp-todragon)# set vtag 3000

cuny-cvlsr(edit-lsp-todragon)# exit

cuny-cvlsr> commit lsp todragon

Where *group 2000* is the Local ID pre-assigned to WASH-PC1, and *vtag 3000* indicates the VLAN tag used for this LSP.

2) The CHEETAH end host is able to request the LSP destined to the end host on the other side of CHEETAH network across dynamic provisioned HOPI/DRAGON networks. Since CUNY-CVLSR and WASH-CVLSR are configured as the DRAGON UNI clients on the both sides, they would take care of the inter-connectivity with HOPI network. The LSP requests can be initiated and terminated on the CHEETAH end hosts via UVa RSVPD clients. For example, the single 1 Gbps FSC LSP request destined to Zelda1 could be initiated on CUNY-H1 by command: *bwrequestor 130.207.252.131 1000*, where *130.207.252.131* is the control plane IP address of Zelda1.

3) The CHEETAH end host is not only able to request the LSP destined to the other CHEETAH end host across dynamic provisioned HOPI/DRAGON networks, but also the one on the same side of CHEETAH network. For example, CUNY-H1 can request an LSP destined to the end host located in MCNC such as Wukong shown in Fig. 4.1 and Fig. 4.9, or the end host located in UVa.

4) Dynamic network cluster function has been tested in this test scenario. The dynamic cluster network has been established based on the request. When a new host joins the established cluster or an existing one leaves the cluster, the notification Path message with the changed cluster members is transmitted to the remote site across HOPI/DRAGON and CHEETAH backbone networks immediately. The switch fabric control, including adding/removing port and adjusting rate limiting, is performed accordingly.

The VLAN in VLAN solution was demonstrated in the lab. However, it's not adopted because the HOPI network does not support VLAN in VLAN in its new architecture.

4.3 Performance Analysis

The CVLSR is implemented on a Linux PC with 2.0 GHz CPU and 512M RAM. The main features of the CVLSR, including general GMPLS functionalities, dynamic network cluster management, interoperability with the Sycamore SN16000 switch, DRAGON VLSR and RSVPD client, etc., have been tested through experiments.

Table 4-1 End-to-End circuit setup delay measured for the LSP request initiated from CUNY-H1 and destined to Zelda1 in Atlanta

Type of Delay	Average	Deviation
End-to-end circuit setup delay (s)	1.862	0.212
CVLSR Path processing delay (ms)	29.3	8.2
CVLSR Resv processing delay (ms)	91.5	1.0
VLAN cross-connection via SNMP delay (ms)	3.7	0.5
Priority queuing and rate-limiting via SSH delay (ms)	86.3	1.0
SN16000 cross-connection delay (s)	1.553	0.239

Table 4-1 shows the end-to-end circuit setup delay measured for the LSP request initiated from CUNY-H1 and destined to Zelda1 in Atlanta (as shown in Fig. 4.6). The

major statistical data includes the measurement of end-to-end circuit setup delay, delay introduced by the CVLSR, CVLSR Path and Resv processing delay, and throughput in the data plane.

The average end-to-end path setup delay measured from the above LSP setup was about 1.862 seconds. The average measured delay, from the time instance at which CUNY-CVLSR sent the Path message to SN16000 in MCNC, to the time instance at which CUNY-CVLSR received the Resv message from SN16000 in MCNC, was about 1.698 s. In other words, the delay introduced by both CUNY-CVLSR and CUNY-H1 was only 164 milliseconds. The end-to-end path setup delay is mainly caused by the cross-connection delay (about 1.553 seconds) between SN16000 in MACN and Atlanta. According to the measurements, the average CVLSR Path processing delay was about 29.3 milliseconds, while the average CVLSR Resv processing delay was about 91.5 milliseconds. In the CVLSR implementation, the Resv processing delay contains the time used to process the Resv message, cross-connect the ports to VLAN via SNMP, set priority queuing and perform rate liming via Secure Shell (SSH). Since Foundry Gigabit Ethernet switch does not support rate limiting via SNMP, this function is implemented via CLI through SSH. Automatic CLI interaction is realized via EXPECT [45]. EXPECT is a program designed specially for interactive applications, such as Telnet and SSH, etc. These interactive sessions can be executed automatically according to the pre-defined EXPECT script files, without manual interaction. The drawback of this solution is the longer delay (about 86.3 milliseconds) introduced by the SSH session, whereas the VLAN configuration via SNMP only took about 3.7 milliseconds. If rate limiting could be performed via SNMP, this delay would be significantly decreased.

Since there is no CVLSR deployed in Atlanta, each host must be connected to one of the SN16000 GbE ports directly. It would result in the quick exhaustion of the capacity of GbE ports. On the other hand, all CUNY hosts could share the 1 GbE link to SN16000 in MCNC via CUNY-CVLSR. Obviously, the involvement of the CVLSR offers a scalable and cost-efficient solution for adding users to the optical network.

The dedicated end-to-end data plane connection dynamically provisioned by CUNY-CVLSR, SN16000 in MCNC and Atlanta was demonstrated via ping test and throughput measurement from CUNY-H1 to Zelda1 in Atlanta. The TCP/UDP throughputs measured from CUNY to Atlanta were 564 Mpbs/560 Mbps with 0% packet loss. On the reverse direction, the TCP/UDP throughputs measured were 535 Mpbs/560 Mbps with 0% packet loss. They were consistent with the maximum circuit size (STS-12c, about 622 Mbps) that NYSERNet MSPPs (ONS 15454) support. The measurement was done by using NUTTCP, which is a network performance measurement tool.

The TCP buffer sizes have been tuned appropriately for throughput measurement. In theory, it should be set to the product of the available bandwidth of the network and the Round Trip Time (RTT) of data going over the network. For the dynamically provisioned connection between CUNY-H1 to Zelda1 in Atlanta, the RTT obtained from ping test is around 23.1 ms. Thus the max buffer size should be set as least:

$$23.1 * 10^{-3} * 622 * 10^{6} = 1.44\ Mbps$$

The connectivity from CUNY to UVa was also verified via VLAN 4082. The TCP/UDP throughputs measured from CUNY to UVa were 511 Mpbs/560 Mbps with 0% packet loss. On the reverse direction, the TCP/UDP throughputs measured were 334 Mpbs/476 Mbps with 0% packet loss. The lower throughput on the connection from Uva

to CUNY was caused by the losses of packets. The Losses could be caused by a number of reasons, for example, dirty fibers, bad connections, or maybe a driver issue. It needs further investigation to locate the problem.

Chapter 5

Using Link Bundling for Efficient Utilization of Dynamically Provisioned Optical Circuits

This chapter begins with the introduction of link bundling mechanism. We define two modes for operating the circuit: default and high speed transfer modes. In the default mode the circuit is used for TCP traffic until a request for the large file transfer is generated. In high speed transfer mode, the end-router releases a certain amount of bandwidth for the large file application; the remaining bandwidth could be used for other applications (e.g., TCP traffic) or other users. We then demonstrate, through experimentations and simulations, how to use the concept of link bundling for dynamic allocation of the optical circuit such that the bandwidth could be efficiently shared. We demonstrate that link bundling can handle the TCP/IP traffic without TCP connection abort when the buffer size of router interface is appropriately chosen. The packet loss introduced by link unbundling can be recovered by TCP retransmissions. Finally, the router configuration delay is discussed.

5.1 Introduction of Link Bundling Mechanism

Large-scale e-Science applications generate huge amount of data that can reach terabyte or even more. In many cases, the data is delay-sensitive and is shared by scientists in different locations. Due to the bandwidth limitations, and intrinsic TCP flow

control mechanism, the current Internet cannot be used to provide a guaranteed high-bandwidth pipe to timely transfer this huge amount of data [46][47]. CHEETAH is one of the promising methods to enable such capability, but only if it is cost-effective.

CHEETAH is a comprehensive effort to support the dynamic provisioning of high-speed optical circuits on the end-to-end basis. It is envisioned that applications such as large file (gigabyte to terabyte) transfers, remote visualization and e-learning can benefit from the high throughput offered by these circuits. Crucial to this concept is the efficient utilization of the optical circuits, due to their costly prices. Statistical sharing of such circuits among multiple users or multiple applications is one way to overcome the prohibitive costs.

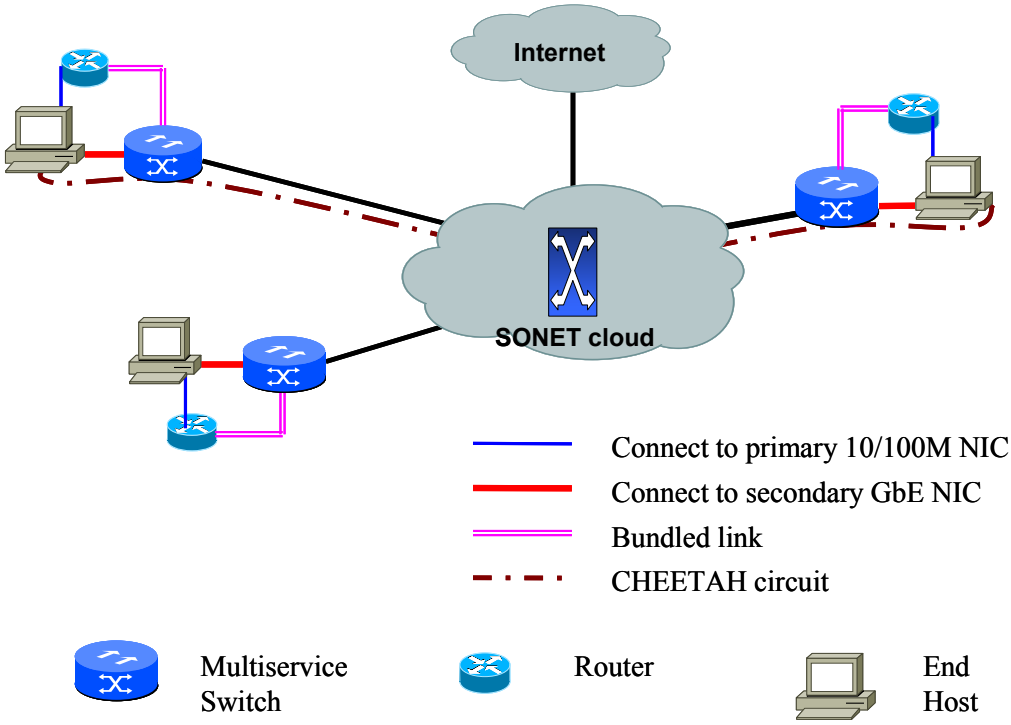

Fig. 5.1 CHEETAH link bundling architecture

Fig. 5.1 illustrates the link bundling architecture in CHEETAH network. Each end host is equipped with two Ethernet NICs. One card (10/100 Mbps) is used for Internet services while the other one (GbE) is used for large file transfers. The detailed equipment

connection is shown in Fig. 5.2. The multi-service switch interfaces the Ethernet signal from the host to the SONET network; alternatively, this switch could be replaced by a 10 GbE switch. In the latter case, the router could be replaced by an Ethernet switch representing a local area network. In the figure, the host is shown directly connected to the multi-service switch, however, the host could be connected through a GbE to a multi-service switch, or connected directly to a 10 GbE without the presence of the multi-service switch altogether.

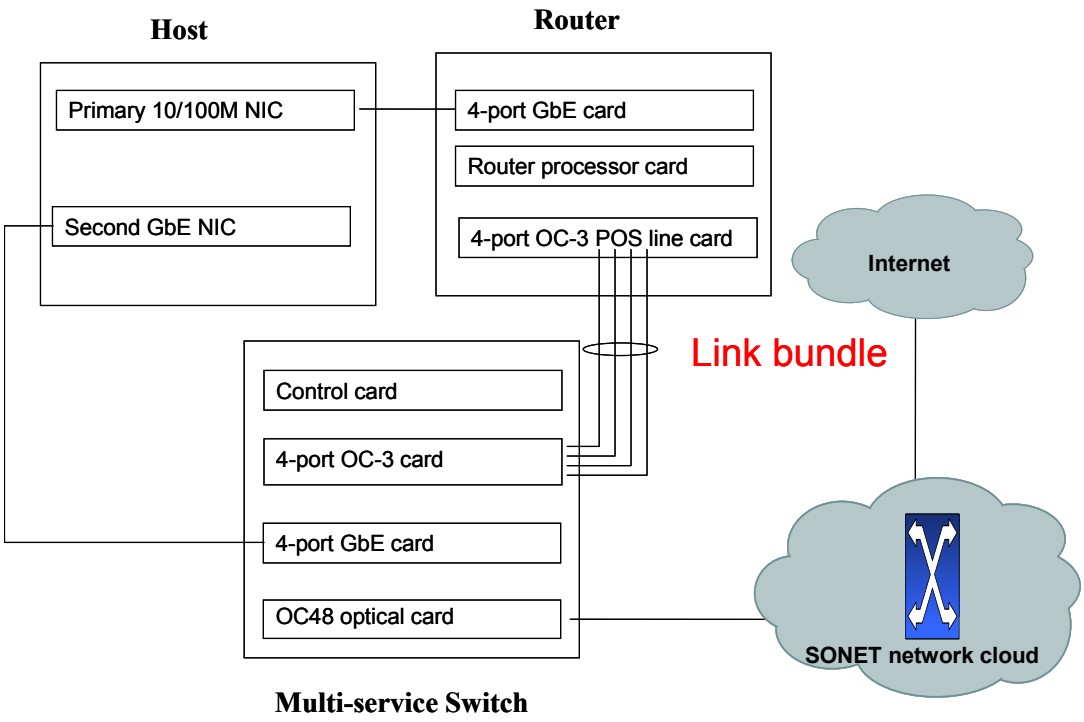

Fig. 5.2 Detailed equipment connection

These different configurations depend upon specific connectivity to the service provider network, and cost-effectiveness of the networking solution design. In the default mode, i.e. there is no large file to be transferred, the end host is connected to the optical circuit through a high speed router to the Internet. In the high speed transfer mode, the host signals the multi-service switch to setup a dedicated end-to-end optical circuit for

large file transfer. After the file is transferred, this optical circuit is then released to the high-speed router for TCP/IP traffic.

In order to maintain the TCP/IP traffic all the time, a mechanism called link bundling is used to allocate the bandwidth in the mode transfer from default to high-speed transfer. The router only releases a certain amount of bandwidth for the large file transfer, and the remaining bandwidth is used for TCP/IP traffic.

5.2 Link Bundling and Load balancing

The link bundling groups multiple point-to-point component links together into one logical link (shown in Fig. 5.3) to provide higher bandwidth (a bigger pipe), as well as load balancing between the two routers. A virtual interface is assigned to the bundled link. The component links can be dynamically added and deleted from the virtual interface. The virtual interface is treated as a single interface on which one can configure an IP address and other software features used by the link bundle. The packets sent to the link bundle are forwarded to one of the links in the bundle. The load balancing should be supported on all links in the bundle.

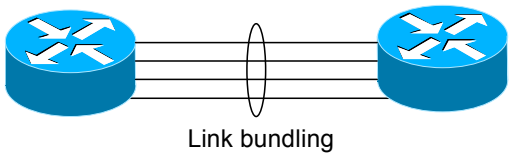

Fig. 5.3 Link bundling between routers

The load balancing function is a forwarding mechanism to distribute traffic over multiple links based on the layer 3 routing information in the router. There are two types of load balancing schemes: per-destination load balancing and per-packet load balancing. When a traffic stream arrives at the router, the per-packet load balancing allows the

traffic to be evenly distributed among multiple equal cost links. The per-packet scheme makes routing decision based on the round-robin techniques, regardless of the individual source-destination hosts. The per-destination load balancing allows the router to distribute packets over one of the links in the bundle to achieve load sharing. The scheme is realized through a hash calculating based on the source-destination address and user sessions.

When the per-destination load balancing is enabled, all packets for a certain source-destination pair will go through the same link, though there are multiple links available. In other words, the per-destination load balancing can ensure that packets for a certain source-destination pair could arrive in order.

5.3 Link Bundling Simulation

OPNET 10.0 is used for the simulations of link bundling. OPNET has a failure/recovery module which can send failure/recovery interrupt to the routers. We use this module to send the link unbundling instruction messages. As soon as the router receives the link unbundling request, it stops forwarding packets to the unbundled interface. In the mean time, the forwarding table is reset so that the packets can be forwarded to the remaining interface. Since only several values need to be changed in the forwarding table according to the link bundling instruction, the delay for resetting the forward table is minimal and it is ignored in our simulation. After the link unbundling, the packets in the unbundled interface queue will not be transferred. To obtain the queue size of each router interface, the IP QoS parameters are set for each interface. TCP sender considers these packets the lost packets and recovers them by retransmitting these

packets. We use the TCP module provided by OPNET to simulate TCP file transfer. Table 5-1 shows the value of TCP parameters used in simulations.

Table 5-1 TCP parameters used in link bundling simulation

Maximum Segment Size (Bytes)	Auto Assigned
Receive Buffer (bytes)	65535
Receive Buffer Adjustment	None
Delayed ACK Mechanism	segment/clock based
Maximum ACK Delay (sec)	0.2
Slow-start Initial Count (MSS)	1
Fast Retransmit	Disabled
Duplicate ACK Threshold	3
Fast Recovery	Disabled
Window Scaling	Disabled
Selective ACK (SACK)	Disabled
ECN Capability	Disabled
Segment Sent Threshold	Byte Boundary
Active Connection Threshold	Unlimited
Nagle Algorithm	Disabled
Kara担Algorithm	Enabled
Time Stamp	Disabled
Initial Sequence Number	Auto Compute
Retransmission threshold	Attempt based
Max. Connection Attempt	3
Max. Data Attempt	6
Initial RTO (sec)	1
Minimum RTO (sec)	0.5
Maximum RTO (sec)	64
RTT Gain	0.125
Deviation Gain	0.25
RTT Deviation Coefficient	4
Time Granularity (sec)	0.5
Persistence Timeout	1

The network is setup as shown in Fig. 5.4. The clients and servers are connected to Gigabit-Ethernet switches to form their own LANs. The two LANS are interconnected by high-speed routers A and B. Between router A and router B there are 4 parallel component OC-1 (or OC-3) links that can be bundled together to form a bundled link. The two routers are placed 2000 kilometers apart, which means the round trip propagation delay of the TCP message is about 13 milliseconds. The single TCP throughput is limited by the long TCP round trip delay, the fat pipe (4-OC1 links), and

the fixed TCP buffer size (in our simulations it is 64 Kbytes, and windows scaling is disabled). The link utilization is increased by adding more TCP flows.

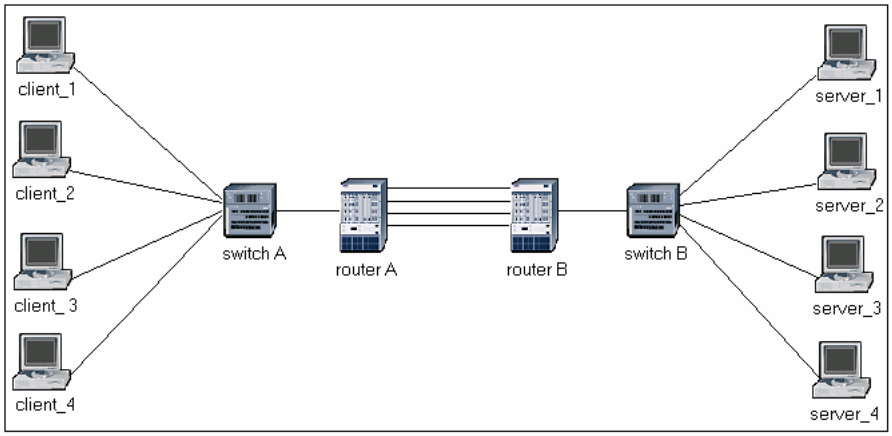

Fig. 5.4 Link bundling simulation network

Two scenarios were tested in our simulation. In the first scenario, called default mode, the four component links are all used as a single bundled link for the TCP file transfer using FTP protocol. In the other scenario, called link-unbundled mode, 1 or 3 components links are unbundled from the bundled link in the middle of the file transfer, leaving only 3 or 1 component link for TCP file transfer. We need a large file size (50 Mbyte) in our file transfer simulation so that the TCP flow can be stabilized when the link unbundling occurs. What we want to observe from the simulations are: 1) whether there is packet loss during the mode transition; 2) whether there is TCP connection abort; 3) What would be the impact of link unbundling on the TCP file transfer in terms of required router interface buffer size, traffic throughput and file download time.

Fig. 5.5 shows the link utilization of one of the four component links (OC-1) for one file transfer. All the other component links have the same link utilization in the default mode. The average link utilization of each component link is about 15.43% of OC1, and the throughput is 7.67 Mbits/sec. The throughput of the whole bundled link is 7.87x4 =

31.48 Mbits/sec. In the link unbundling scenario, 3 of the 4 component links are unbundled from the bundled link in the middle of the file transfer. The dashed curve shows the utilization of the remaining link. The average link utilization after the link unbundling is about 62% of OC1, and the throughput is 31.62 Mbits/sec. Since only one file is transferred, the remaining link's utilization is not fully loaded even after the link unbundling. Thus, neither TCP connection abort nor the TCP retransmission after link unbundling is shown in the simulation results. That is, there is no packet in the unbundled interface queue waiting for transmission before the link unbundling occurs. When the number of file transfers increases, i.e., the link load increases, the impact of link unbundling increases.

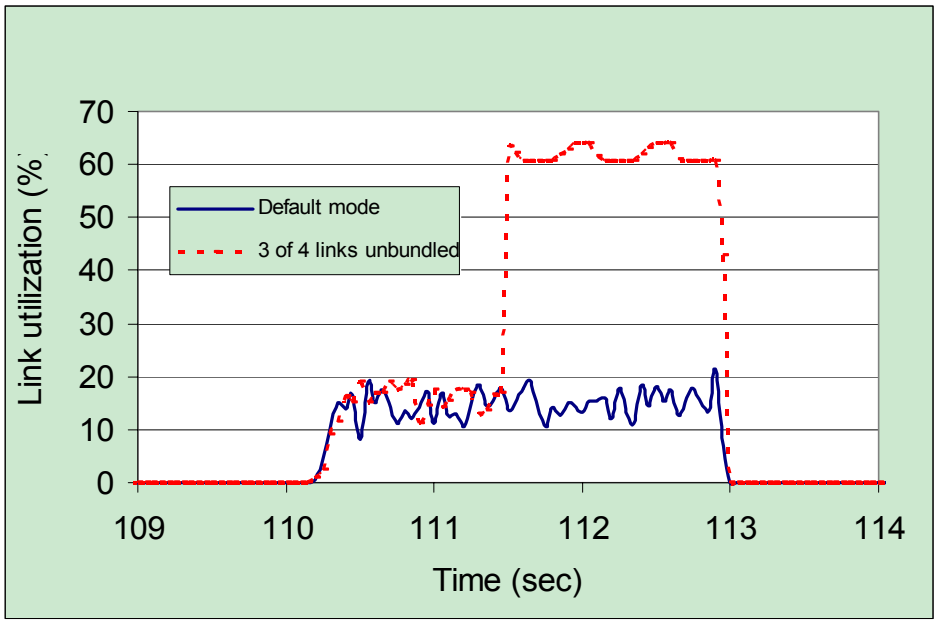

Fig. 5.5 Link utilization of one component

Fig. 5.6 sketches the remaining component link utilization before and after the link unbundling. In the 1-link unbundled (OC-1) case, for one file transfer, the remaining link utilization before and after the link bundling is 22% and 30% respectively. The remaining

link utilization (after link unbundling) reaches 100% if the number of file-transfer is more than 5. For 1-link unbundled (OC-3) case, this value is 8.

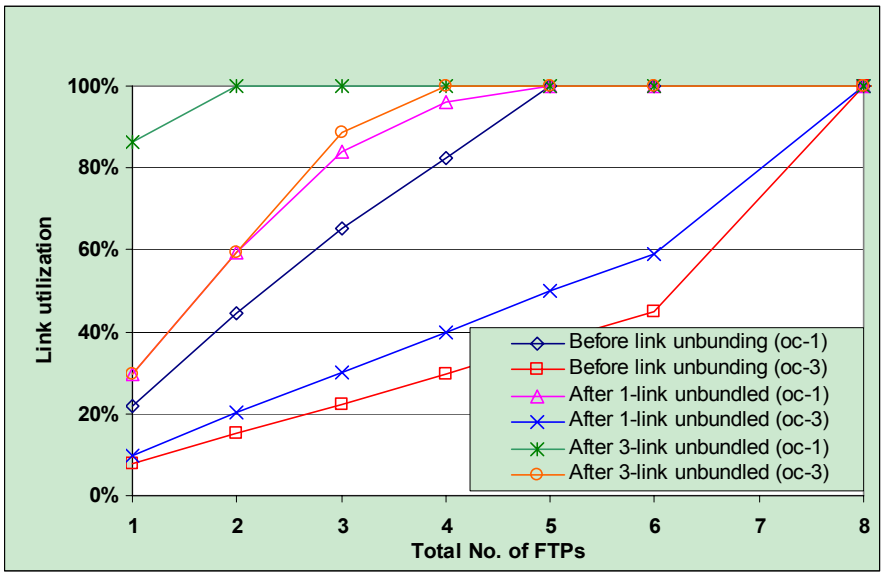

Fig. 5.6 Link utilization vs. number of file transfers

To verify that the packets retransmitted are actually the packets halted in the unbundled interface queue, the size of the router interface buffer is set large enough so that no TCP retransmissions are induced by the buffer overflow. By doing this, we also identify the required buffer size of the router interface in order to avoid buffer overflow after the link unbundling operation. As shown in Fig. 5.7, the router interface queue size increases significantly in the 3-link unbundled case, compared to the default and 1-link unbundled case. Take the 3-link unbundled curves for example; the maximum interface queue size at 100 file transfers is 6.5, 5.4, 3.4 Mbytes for OC-1, OC-3 and OC-12 cases respectively. Thus, the interface buffer size must be carefully chosen to avoid unnecessary TCP retransmissions caused by the interface buffer overflow when link unbundling occurs.

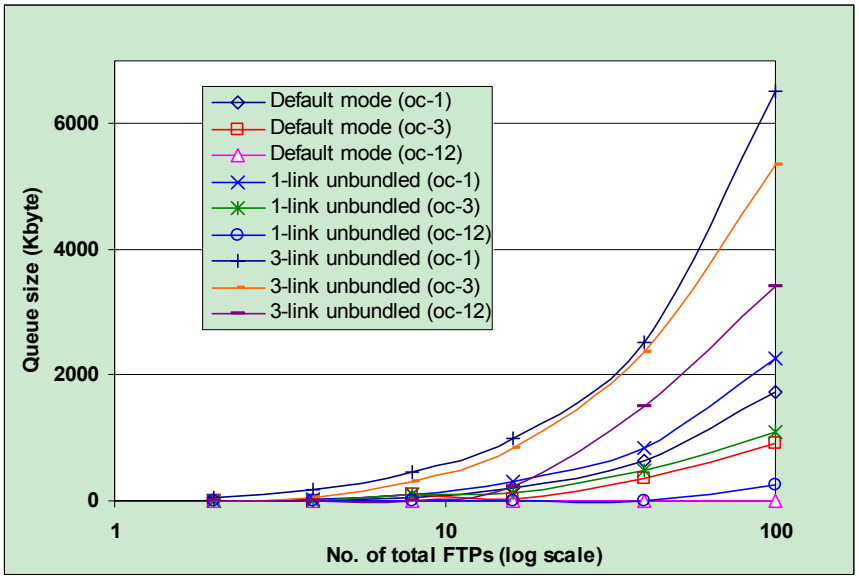

Fig. 5.7 Maximum queue size after link unbundling

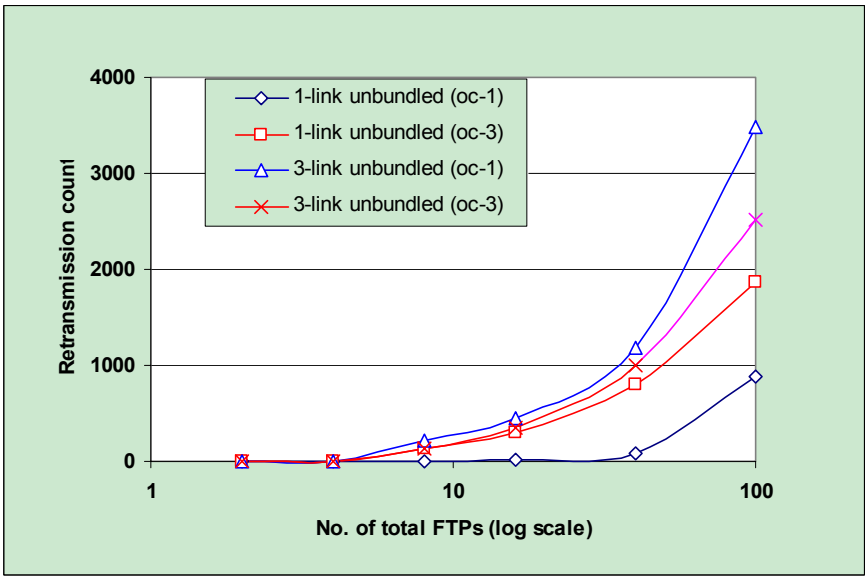

Fig. 5.8 Retransmission count vs. number of file transfers

Fig. 5.8 shows the total TCP retransmission count as the number of file transfer increases. The retransmissions are caused by the packets that stalled in the router's interfaces after the link unbundling. Fig. 5.9 depicts the size of the stalled packets in the interface queue. These packets are considered lost and will be recovered by TCP retransmissions. We observe there are less 10 or no TCP retransmissions if the

component link utilization (before link unbundling) is less than 100%. At such a link utilization level, the packets forwarded to the interface are almost sent out immediately without being queued in the interface buffer. After the component link utilization (before link unbundling) reaches 100%, the number of packet queued in the interface raises as the number of file-transfer increases, hence resulting in more TCP retransmissions after the link unbundling.

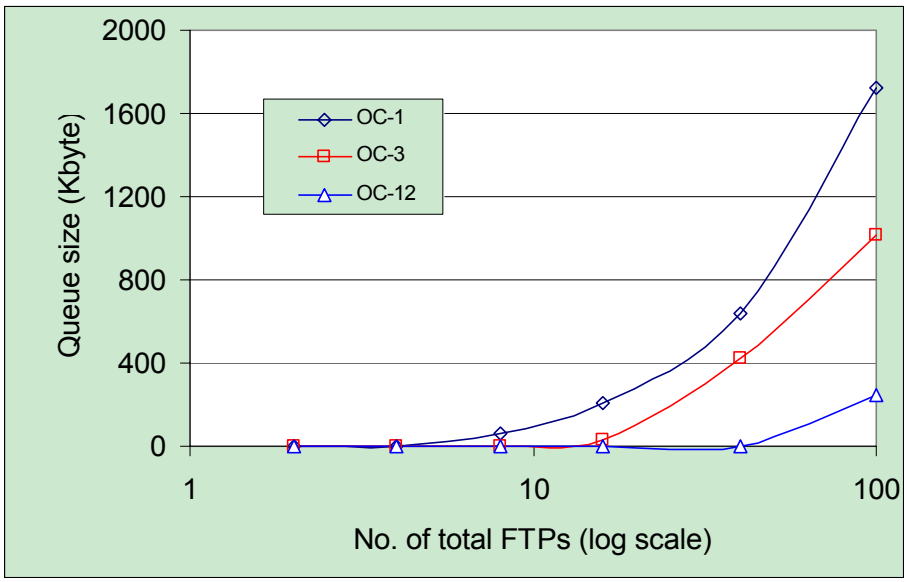

Fig. 5.9 Queue size in unbundled interface

The congestion window (CWND) varies due to the TCP packet loss, is shown in Fig. 5.10. It is the case for 16-file-transfer in 3-link (OC-1) unbundled scenario. The link-unbundling happens at 113 second, which generated 408 TCP packet retransmissions. The curve (with packet loss) shows the window size of TCP flow affected by the packet loss. The TCP performs slow-start due to the packet loss after the link unbundling. The curve (without packet loss) shows the window size of TCP flow without packet loss during the file transfer. After the link unbundling, its window size still increases, but more slowly compared to the default mode due to the narrowed bandwidth.

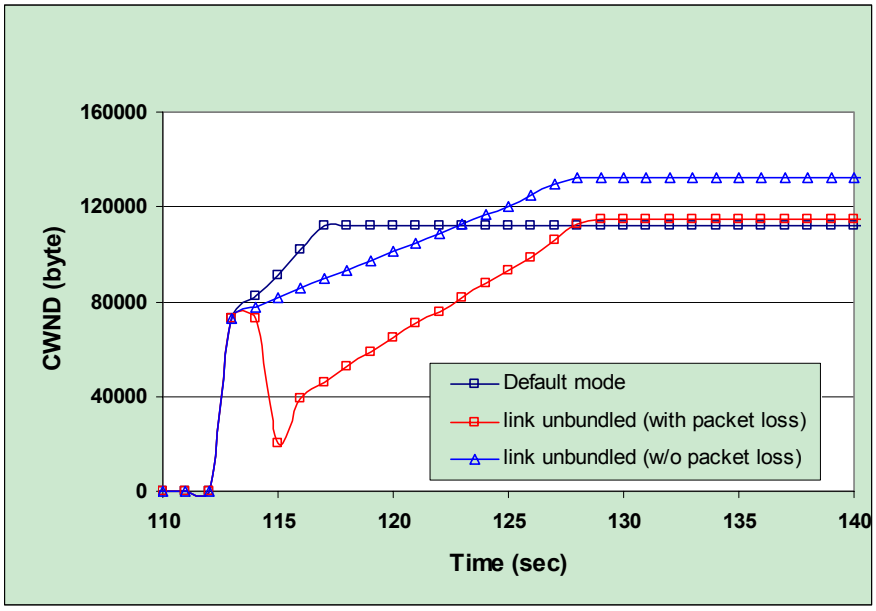

Fig. 5.10 Congestion window size during mode transition

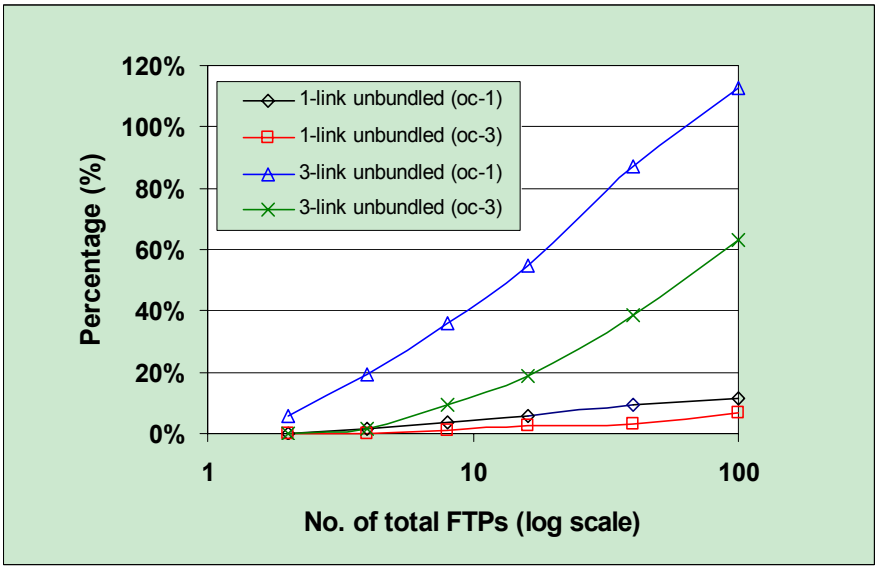

Fig. 5.11 Percentage of file transfer time increased vs. number of file transfers

The percentage of the increase in the file transfer time after link unbundling vs. the number of file transfers is shown in Fig. 5.11. If the remaining link's utilization (after link unbundling) does not reach 100%, for example, the number of file transfer is less than 5 in 1-link (OC-1) unbundled scenario (please see Fig. 5.6) or the number of file-

transfers is less than 2 in 3-link (OC-1) unbundled scenario, the increase of file transfer time is not visible. For example, there is no noticeable difference for 1 file-transfer time between the default mode and the 3-link unbundling mode, so is for the 3 file-transfer time between the default mode and the 1-link unbundling mode.

Fig. 5.12 shows the relationship between the total TCP goodput and the number of file transfers. The total TCP goodput increases as the number of file transfers increases. As the number of file transfers is more than 20 in the OC-1 case, the total TCP goodput reaches its plateau, which suggests all the links has been saturated. For the OC-3 cases, more TCP flows are needed to saturate the links.

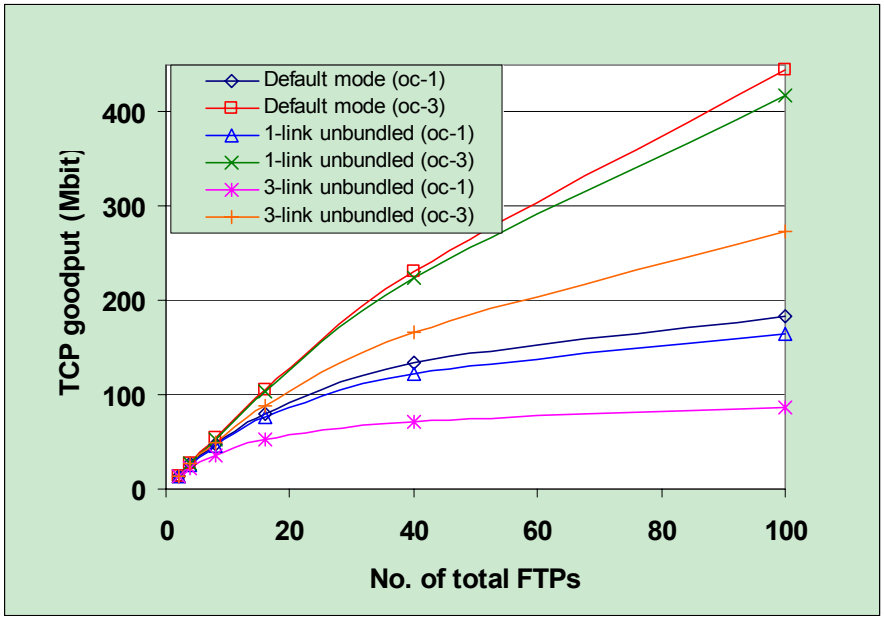

Fig. 5.12 Total TCP goodput vs. number of file transfers

5.4 Link Bundling Experiment

To verify our simulations, link bundling experiments are carried out with two Cisco 12008 Gigabit switch routers in an office LAN environment. Each router is equipped with one 4-port gigabit Ethernet card and one 4-port OC-3 packet-over-SONET line card.

The two routers are connected by optical fibers. We bundled two OC-3 component links into a bundled link. Fig. 5.13 shows the experimental topology. Since the 4-port gigabit Ethernet line cards we're using only support per-destination based load balancing, in order to balance the load to the two component links, we create multiple client-server pairs between the two routers.

Two scenarios which are similar to the ones in the simulation are tested. For the default mode, the two component links in the bundled link are both used for the file transfer. For the link-unbundling mode, one of the component links is unbundled from the bundled link in the middle of the file transfer.

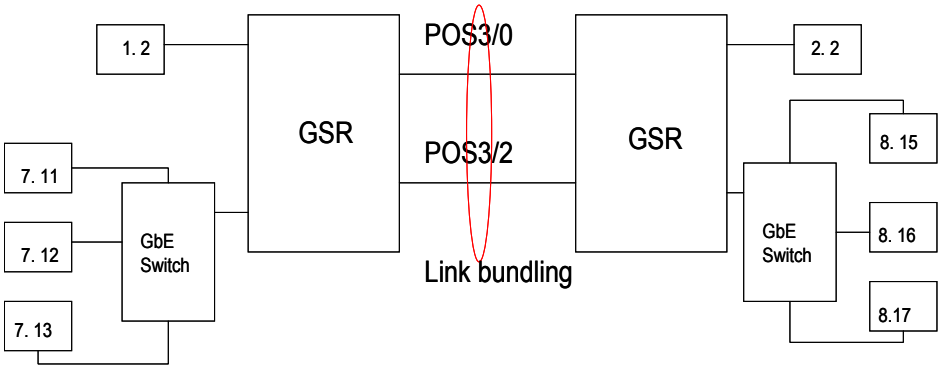

Fig. 5.13 Link bundling experimental topology

In the experiments, iperf is used to initiate the TCP file transfer, tcpdump is used to capture packets during the file transmission, and ethereal is used to analyze the packets captured by tcpdump. Comparison of the results between the experiments and simulations is sketched in Fig. 5.14 and Fig. 5.15. Fig. 5.14 shows the retransmission traffic generated by the link unbundling, while Fig. 5.15 shows the percentage of increasing time for the file transmission introduced by the link unbundling.

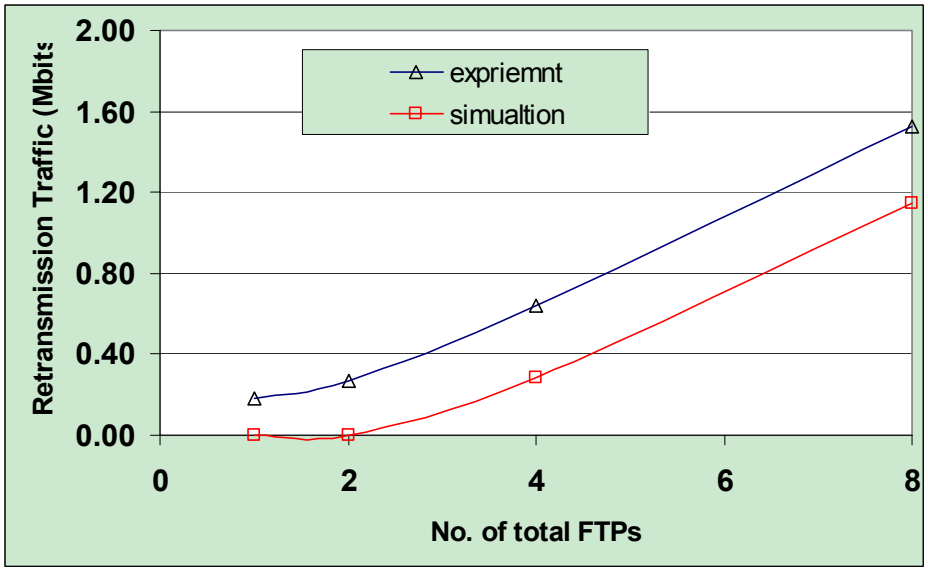

Fig. 5.14 TCP Retransmission introduced by link unbundling

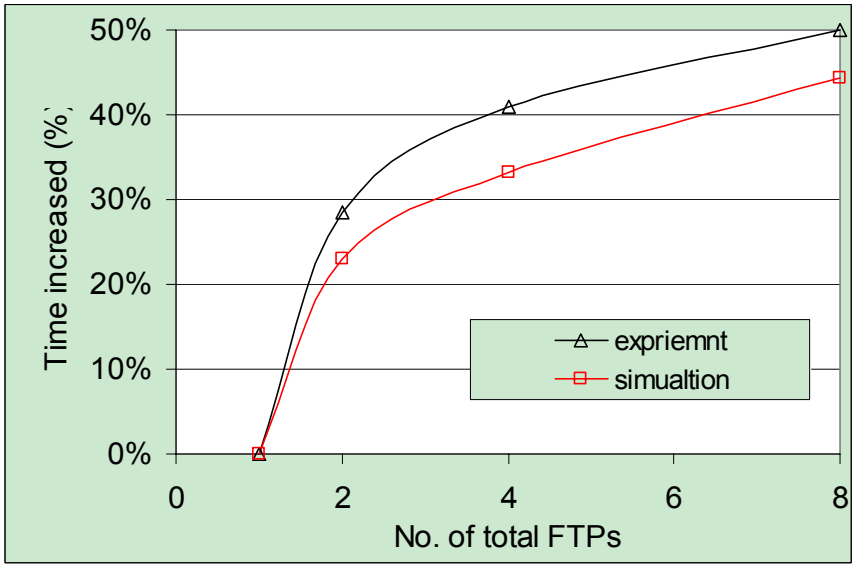

Fig. 5.15 Percentage of increasing time introduced by link unbundling

Similar to the results of the simulations, no TCP connection abort is observed in our experiments. The difference is caused by the link utilization in the simulations and experiments. The link utilization in the experiments can reach 100% even with one file transfer situation. It is different from the situation in the simulation, where the link utilization increases with the number of file transfer. In the simulation, the maximum

TCP receiver buffer is set to 64k bytes. The TCP buffer size required for fully link bandwidth utilization equals to the multiplication of the bandwidth and the delay. In the simulations, one file transfer can not utilize the whole link bandwidth since the delay introduced by the long distance is so long that the required TCP buffer size exceeds the 64k bytes. Whereas in the office LAN environment, due to the small delay, the required TCP buffer size is below the maximum 64k bytes. The link can be fully utilized by one file transfer.

5.5 Router Remote Control and Delay

In order to use the link bundling function for dynamic bandwidth allocation, the routers must be dynamically configurable from local or remote hosts. Currently, Telnet/SSH are commonly used to access remote routers. If public network is used to transmit configuration messages, Telnet is not recommended, since it can not set up a secured session and all messages including username and password are transmitted in plain text. SSH is a protocol for secure remote login which all the messages are encrypted, thus it is used to connect to the remote routers for link bundling and unbundling. Since SSH is a CLI based interactive session, EXPECT is adopted to implement automated remote configuration in our experiment.

The router configuration delays are measured in both the local control mode and remote control mode. In local control mode, the host which triggers the unbundling is located in the same local area network as the router. In this case, the delay is mostly caused by router processing delay. The delay is measured as 13.7 milliseconds. In remote control mode, the host is located in a remote network. In addition to router processing delay, the total delay may vary depending on the message propagation and

transmission delay. For the experiment in remote control mode, we use the host in our campus to control the router located in UVa site. The distance between the two sites is about 900 kilometers. The measured delay is 22.3 milliseconds. If SNMP can be used for the link bundling/unbundling management, this delay would be greatly reduced.

Chapter 6

Optimized Scheduling for Book-ahead Services

The CHEETAH network is designed to support a broad class of e-Science projects and specifically TSI. These applications, such as grid applications, e-science, and BoD applications, could benefit from the ability to schedule the connection across multiple areas or ASs. In this chapter, we first discuss the requirements and challenges of connection requests scheduling in today's IP optical networks. Linear programming, a very popular algorithm used for multi-commodity traffic flow problem, is then briefly introduced. We present an optimization model based on the linear programming to schedule the book-ahead connection requests subject to QoS constraints such as minimizing connection blocking rate while maximizing network utilizations. This chapter ends by discussing the simulation results that show the significant improvement of the proposed approach on the desired performance, i.e., reduction of blocking rate and improvement of utilization.

6.1 Challenging Requirements

Recent advances in optical networking technologies have made it possible to realize the vision of enabling of BoD services directly to end-users. These applications can be classified into two main classes: 1) some require high capacity pipes for short or long periods of time (e.g., large file transfers for remote processing of large datasets); whereas

2) others require low bandwidth pipes also for short or long durations (e.g., interactive visualization, video-on-demand, music downloads). Some of these applications (e.g., VoD, IP-TV, or other web-based downloads such as music, or internet gaming) are examples of Immediate Request (IR) applications that require near real-time setup, whereas others such as e-learning, multimedia conferencing and grid computing belong to a class of Book-Ahead (BA) applications that normally require book-ahead scheduling.

The significance of GMPLS control plane with respect to its impact on different types of grid computing applications is summarized in Table 6-1. For small to medium bandwidth pipes requiring immediate scheduling, i.e., BoD services, there is a high value for using the distributed routing and near real-time signaling mechanisms [48][49]. It is worth mentioning that it is more efficient from network utilization point to dynamically provision longer holding times connections rather than shorter ones [50]. Clearly, for book-ahead applications the value of distributed routing and near real-time signaling is not as high. On the other hand, book-ahead applications may not benefit from the near real time connection setup capability of the control plane.

Table 6-1 Value of GMPLS control plane for different types of application

	Type of application		
Control plane capabilities and functionalities	Small to medium bandwidth request		Large bandwidth request
	Immediate scheduling	Book-ahead scheduling	Book-ahead scheduling
Distributed routing	HV	LV	LV
Near-real-time connection setup	HV	LV	LV
Auto-discovery	HV	HV	HV
Rapid restoration functionality	LV	HV	HV

HV: higher value; LV: lower value

The challenging requirements of the advance scheduling for the BA services can be summarized as the followings:

1) Path feasibility. Enabling technologies of connection-oriented paradigms, such as MPLS and GMPLS, are the right tool to provide the users on-demand dynamic provisioning paths. Obviously, there will be a chance of blocking. However, the network resources (interfaces, ports, wavelengths, and time-slots) can be appropriately sized to maintain an acceptable rate of blocking.

2) Schedule evaluation. The applications outlined above, indeed, require variable amounts of bandwidths pipes at different times, for varying durations, some require a "virtual private network" setting such as in interconnecting multiple computing and storage facilities, clustering for visualization, or multicasting in VoD and other interactive services. The network should be equipped with an advance scheduler to make the reservations for the connections at certain future times, so that the higher network utilization and lower blocking rate can be achieved.

3) Path and schedule reconciliation. Many of above applications are often dynamic requiring not only varying amounts of bandwidths but also varying degrees of connectivity among the nodes. Hence, scheduling is needed not just to allocate and reserve resources, but also to trigger dynamic reconfiguration of the network resources (e.g., time-slots, wavelengths at varying granularities) in order to direct the network resources towards efficient utilization in response to the applications' demands. This is a realization of the principle of statistical sharing of resources which is imperative for maximizing the productivity of the networks' assets.

4) Architecture of schedulers. In terms of network architecture, the schedulers can be realized in different architecture: (a) single centralized scheduler, (b) fully distributed on each switch, or (c) scheduler per domain. A single centralized scheduler presents scalability problems and integration problems amongst different administrative domains. Potential problem with fully distribution is the excessive floods of OSPF-TE information and other updating information, and excessive synchronization delays of OSPF databases. For example, the reservation information has to be distributed to individual switches/routers. The occurrences of reservations along with the modification or cancellation of previously made reservations would involve all the routers/switches in that scheduled path. It could be an excessive burden for a large scale networks.

6.2 Scheduling Problems

The emergence of the optical control plane and the ability to control other infrastructure layers such as Ethernet in CHEETAH network has given us the ability to drastically reduce lead times and duration commitments. Taking example of airline reservation system, we can show up at the airline counter at the day of flight and request a seat. However, people usually make the reservations ahead of the traveling date to get better prices and assure the seats. In the case of high-speed networks, even though we now have a control plane that permits rapid connection provisioning, current networks do not generally have this simple ability.

In addition, unlike the airline reservation system, if we know the connection demands ahead of time, we can optimize how traffic is placed across the network so that more traffic/users can be serviced. For example, let's consider the five nodes with 1-wavelength links network shown in Fig. 6.1. The cost associated with each link is

denoted in the figure. Suppose we receive advance reservations for 1-wavelength connections for some future time interval in the following order: 1) node 2 to 3, 2) node 1 to 4, 3) node 3 to 4, and 4) node 4 to 5.

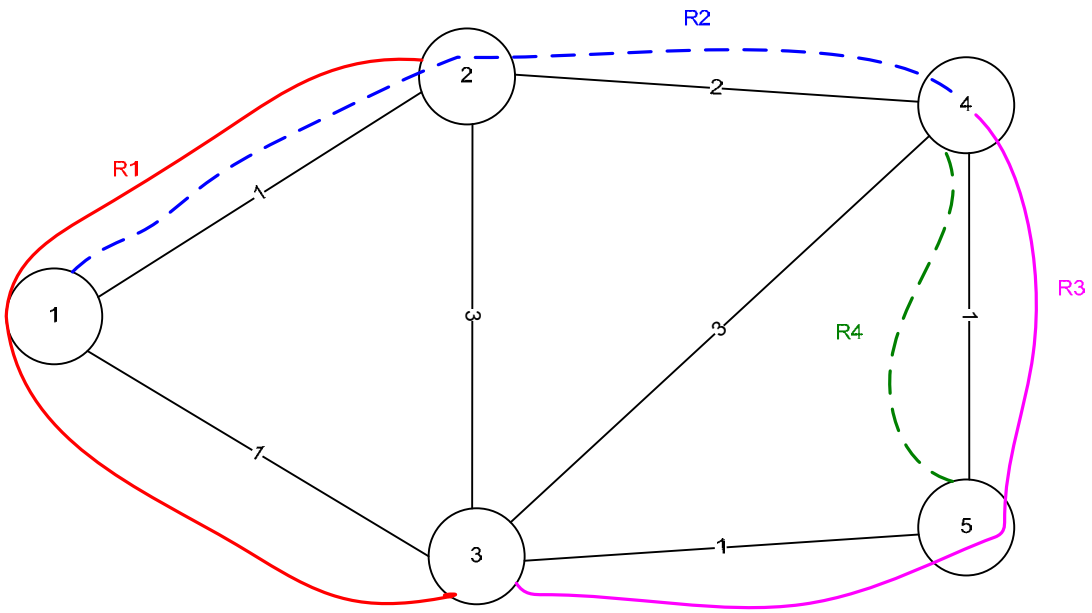

Fig. 6.1 Example network with four connection requests of which only two are admitted

Using the popular CSPF algorithm, that is the most common type of algorithm utilized in MPLS and GMPLS, each bandwidth request is serviced individually and optimally at the time it comes in. By optimality in this case, we find the shortest path between source and destination nodes meeting the bandwidth requirements for each connection individually. The result with such a procedure, shown in Fig. 6.1, is that only two out of four of the connection requests can be admitted.

On the other hand, since the requests are for some bandwidths to be used in the future, we may make optimized reservations for all four connections as a group, and we get the results shown in Fig. 6.2 where all four of the connection requests can be simultaneously admitted.

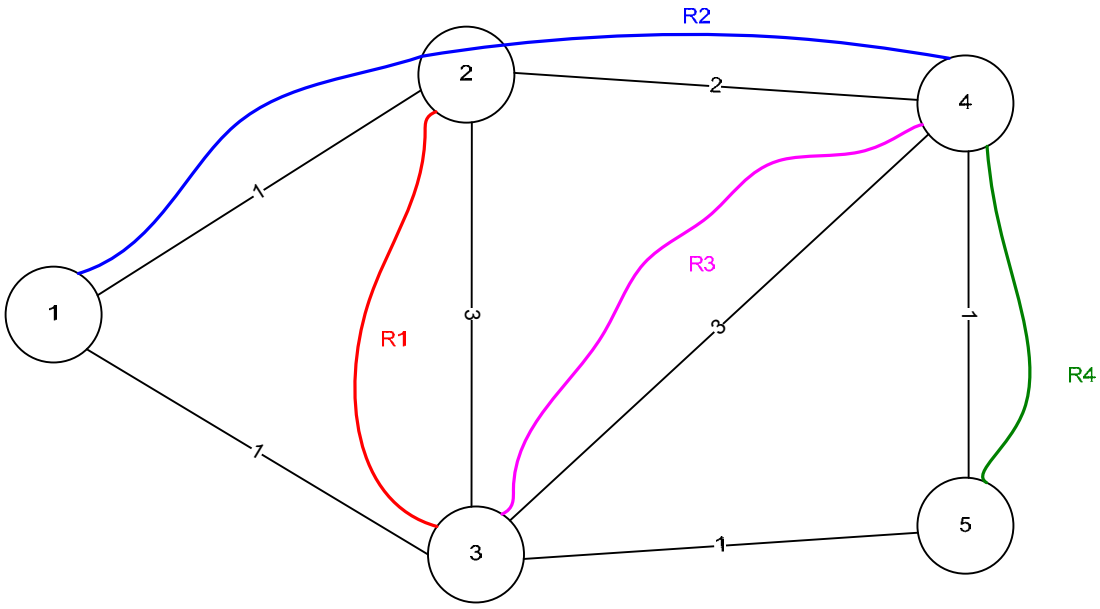

Fig. 6.2 Example network of Fig. 6.1 where all four connection requests are admitted

The problem isn't that the CSPF is not good. In the case of advance reservations, we can apply more sophisticated optimization algorithms since we have a better view of the future demands on the network and hence can perform a more global rather than local optimization of network resources.

The types of reservations vary depending on the type of applications. The first type of reservations that we consider is that for a fixed bandwidth beginning at a prescribed starting time and stopping time (both times being in the future). Such a scenario includes e-learning, the broadcast of a real time high definition sporting event, or a variety of other types of communications with specific start and end times. The second type of reservations that we consider is more oriented towards bulk data transfer. In this case there is a fixed amount of data to be transferred and the transfer cannot start until after a prescribed time and must finish by a prescribed time. Examples of this type of reservation include: the transfer of experimental data from one site to another for processing, distribution of non-real time high definition video, off hours database replication, etc.

Other reservation types may also be viable in addition to above variations on starting time and stopping time, e.g., minimum acceptable bandwidth when a heavy traffic is expecting in the starting time. In this case, if the connection request with requested bandwidth could not be admitted, the user can accept the connection with lower bandwidth (defined as minimum acceptable bandwidth).

In summary, a connection request can be defined by the following parameters: (a) connection ID, (b) source node, (c) destination node, (d) bandwidth requested, (e) starting time, (f) holding time, (g) maximum acceptable starting time from prescribed starting time, and (h) minimum acceptable bandwidth, etc.

In order to achieve minimum connection blocking rate and maximum network utilization, we propose an optimized scheduling algorithm that makes advance reservations for all present connection requests together as a group. The algorithm is based on the Integer Linear Programming (ILP) algorithm.

6.3 Introduction of ILP

The Linear Programming (LP), also known as linear optimization, is a mathematical programming method used to find the extreme (minimum or maximum) of a linear object function, subject to some linear equality and/or inequality constraints. A standard LP problem can be expressed as the following form:

Minimize (or Maximize)	cx
Subject to	$Ax = b$
Where	$x \geq 0$

Here is an example of the LP problems:

$$\max imize \quad f = c_1 x_1 + c_2 x_2 + ... + c_n x_n$$

subject to linear constraints :

$$a_{11} x_1 + a_{12} x_2 + ... + a_{1n} x_n \le b_1$$
$$a_{21} x_1 + a_{22} x_2 + ... + a_{2n} x_n \le b_2$$
$$.....$$
$$a_{m1} x_1 + a_{m2} x_2 + ... + a_{mn} x_n \le b_m$$

and bounds of variables :

$$l_1 \le x_1 \le u_1$$
$$l_2 \le x_2 \le u_2$$
$$.....$$
$$l_n \le x_n \le u_n$$

Two families of algorithms, simplex method and interior-point method, are widely used to solve the LP problems today. Both methods visit a progressively improving trial of solution till an extreme of the object function is found. The simplex method visits the points along the edges of the boundary while the interior-point method visits the points within the interior of the feasible region [51].

If all variables are restricted to be integer, the problem is called Integer Linear Programming (ILP) problem. If only some of variables are restricted to be integer, the problem is called Mixed integer Linear Programming (MIP) problem. Branch and bound, branch and cut, and branch and price are some commonly used advanced algorithms to solve the ILP/MIP problems.

Linear programming method has been widely used to find the optimization solutions for network flow problems and multi-commodity flow problems such as traffic grooming [52]and light-path reservation problems [53] in today's IP optical networks.

6.4 Optimized Scheduling Model

The network is formulated as a directed graph G = (V, E), where V is the set of nodes and E is the set of edges (links). Each node denotes one network node, while each edge denotes one unidirectional fiber from one node to the paired node. The optimized scheduling problem is therefore formulated as an integer multi-commodity network flow problem, which can be solved by the ILP algorithm. The optimization model is described by the parameters, variables, constraints and optimization objective.

1) Parameters

• *V*: set of nodes, *1*...*N*, where *N* is the number of nodes in the network. *s* and *d* are two positive integers used to denote the source and destination nodes, $s \in V, d \in V$.

• *E*: set of links, *1*...*L*, where *L* is the number of links in the network. *l* is a positive integer used to identity each link, $l \in E$.

• *C*: cost of the links, where C_l denotes the cost of link *l*.

• *M*: capacity of the links, where M_l denotes the capacity of link *l*. The capacity of the unit could be the bandwidth of one wavelength, 10 Gbps, or 1 Gbps, depending upon the granularity of the network.

• *S*: set of slots, *1*...*S*, where *S* is the number of slots used to define the length of scheduling period. *t* is the index number of each slot, $t \in S$.

• *P*: set of all possible routes in the network. For any nodes pair, i.e., from source node *s* to destination *d*, the traffic could be transmitted through several different paths. If $P_{s,d}^{k,l}$ takes value of 1, it means that the link *l* is on the k^{th} path of route from *s* to *d*. The k^{th} path of route from *s* to *d* carries the links $\sum_{l} P_{s,d}^{k,l}$.

• *R*: set of bandwidth requests. The request is uniquely identified by $R_{i,s,d}$, where *i* is the connection ID, *s* is the source node, and *d* is the destination node.

• *H*: set of all possible reserved slots of all requests. $H_{i,s,d}^{j,t}$ indicates the usage of the time slot *t* for the connection *i* from node *s* to *d*, where *j* could be all acceptable time shifts. The number of used slots for each shifting would be $\sum_t H_{i,s,d}^{j,t}$, i.e., the holding time slots of this request.

• *BW*: set of non-negative integer. The requested bandwidth of the connection *i* from node *s* to *d* is represented by $BW_{i,s,d}$.

• *FR*: set of non-negative integer. FR_l^t stores the previously reserved bandwidth on the link *l* at time slot *t*. This part of bandwidth should not participate in the ongoing optimization.

2) Variables

• *I*: binary variable. If $I_{i,s,d}^{k,j,t}$ takes value of 1, it indicates that the time slot *t* is reserved for the request *i* from node *s* to *d*, where the k^{th} path is chosen for this request. The start time is shifted at *j* time slots if *j* has none-zero value.

• *F*: non-negative integer variable. F_l^t represents the total bandwidth on the link *l* at time slot *t*.

• *WF*: non-negative integer variable. WF_l^t represents the weighted total bandwidth on the link *l* at time slot *t*. Generally, the users prefer to have the requests scheduled to start at the exact time instance. Only when the asked time slot is not satisfied, the start time is trying to be shifted. A weight is given to the call with shifted start time.

3) Constraints

• Indicator constraints: during the entire transmission time, a specific bandwidth request always takes the same path. Only one path is admitted to a single request.

(a) $\sum_{k}\sum_{j} I_{i,s,d}^{k,j,t} = 1$, where $\forall t \in S, (i,s,d) \in R$

(b) $\sum_{t} I_{i,s,d}^{k,j,t} = S * I_{i,s,d}^{k,j,1}$, where j could be any acceptable time shift, and k could be any possible path, $(i,s,d) \in R$.

• Flow and capacity constraints: the bandwidth, including both being reserved bandwidth and reserved bandwidth, on a specific link should not exceed the capacity of this link.

(a) $F_l^t = FR_l^t + \sum_{i,s,d}\sum_{k}\sum_{j} BW_{i,s,d} * I_{i,s,d}^{k,j,t} * P_{s,d}^{k,l} * H_{i,s,d}^{j,t}$, where $\forall t \in S, \forall l \in E$

(b) $F_l^t \le M_l$, where $\forall t \in S, \forall l \in L$

(c) $WF_l^t = FR_l^t + \sum_{i,s,d}\sum_{k}\sum_{j} BW_{i,s,d} * I_{i,s,d}^{k,j,t} * P_{s,d}^{k,l} * H_{i,s,d}^{j,t} * (s+1)$, where, the flow weight is given by *(s+1)*, i.e., more shift and bigger (heavier) weight.

4) Objective

The objective of the optimization is to minimize the total cost while all requests are admitted.

$$\min \sum_{l}\sum_{t} WF_l^t * C_l$$

If no optimal solution was found, the network can not admit all requests. Therefore, we decrease the number of requests per optimization, called optimization window here, till the single request that could not be admitted is screened, i.e., this request would be

blocked. The algorithm of decrease of the optimization window varies. Since the bandwidth requests are generated randomly, the half decrease method is chosen in our simulation. Apparently, the number of times of optimization at each optimization circle increases when the blocking happens.

6.5 Simulation and Results

The ILP optimization problem is solved using GNU linear programming kit [54], where two steps of optimization are involved. The optimal basic solution is obtained by the simplex-based solver in the first step. The branch-and-bound-based MIP algorithm is then used to find the integer optimal solution.

The limitation of our optimization is the complexity of the simplex and branch-and-bound algorithm. Theoretically, the complexity of simplex algorithm is exponential time of the problem size at its worst case. Here the problem size is determined by the size of the network, the number of paths for each pair of nodes, the number of time slots per scheduling period, the acceptable number of time shifts, and the number of requests per optimization circle. The change of any of them would greatly affect the efficiency of the solver. In our simulation, a 5 nodes network is chosen (shown in the Fig. 6.1), where there are 20 time slots per scheduling period, and each request is allowed to have at most 2 slots shifted from the requested start time. The number of requests per optimization circle is exponentially randomly generated. The simplex optimization is observed as an efficient solver in most cases of our simulation. In contrast to the simplex algorithm, the MIP algorithm is clarified as Nondeterministic Polynomial-time hard (NP-hard). The branch-and-bound method that is used by GNU solver is one of the advanced MIP solvers. Its efficiency critically depends on the nature of the problem and the

effectiveness of the algorithm used. According to our observation, some optimizations might take hours, even days to find the "ideal" solution. Practically, we set a searching iterations limit to stop this kind of long time optimization. For example, the solver stops searching when 100,000 iterations have been performed, no matter the optimal solution is found or not. Clearly, the results from this optimization are good but not perfect.

Two types of scheduling models are implemented in our simulation. In the first model, called pre-reserve algorithm, the system actually reserves all time slots scheduled to the requests that are admitted, no matter they are scheduled to start at current or future time slots. The second model, called reconciliation algorithm, only reserves the time slots for admitted requests that start at current time slot. Other future requests are admitted but without reserving specific time slots. They would participate in the next optimization circle together with new connection requests.

The simulation results are compared with a First Available Shortest Path (FASP) algorithm. Since traditional CSPF does not support path computation on the time domain, the FASP algorithm is developed to obtain a sub-optimal solution on a call-by-call basis. The pseudo code of FASP is shown in and described as the following steps:

Step 1: Dijkstra's shortest path (DSP) algorithm is applied to obtain the shortest path which contain link E1, E2, …Ep.

Step 2: Starting from the first link in the calculated path and first slot, determining if there is a series of continuous slots (total length is equal to the call holding time) and whose available capacities can satisfy the request.

Step 3: If at certain time slot *ti*, the link does not satisfy the request, go to step 2 and check again, but this time starting from *t(i+1)* slot. The search is looped until the

continuous slots if find, record the starting slot *tj*, then go to Step2 staring the search for the next link from the *tj* slot.

Step 4: If during the searching at certain link, the end of the book-ahead slot (total-slot length – call holding length) is reached without finding the consecutive slots, this link is pruned from the network graph and go back to step 1 to calculate the next shortest path.

Step 5: If the last link in the path is reached and the common continuous slots are found in all the links along the path, the search in done and the reservation is made. Otherwise, the call is blocked.

```
1:      t_start = 0;
2:      Path_Vector (E1,E2,...,Ep) = SPF_Path_Compute (Source, Destination, G(V,E))
3:              if (Path_Vector == NULL)
4:                      return NULL; //the path cannot be found, the call is blocked
5:      for (Ei = E1; Ei<=Ep; Ei++){
6:              Ei_status = 0;//Set status of link Ei "not satisfied"
7:              for (ti= t_start; ti<T; ti++){
8:                      for (tj=ti;tj < (ti+H); tj++){
9:      //In link Ei, no continues time slots for holding time H available
10:     //starting from ti,try staring from next t(i+1) slot
11:                             if (available bandwidth of Ei < requested bandwidth){
12:                                     t_start =tj;
13:                                     break;
14:                             }
15:                     }
16:     //Find continuous time slots for holding time H in link Ei, check next
17:     //link staring from slot tj
18:                     t_start = tj;
19:                     Ei_status = 1;//Set status of link Ei "satisfied"
20:                     break;
21:             }
22:     //Link Ei does not satisfy the request, remove it from E, re-compute the path
23:             if (Ei_status == 0){
24:             remove Ei from E;
25:             goto 2;
26:             }
27:     }
28:     //Every link of the path pasts the bandwidth test, the call is admitted.
29:     return Path_Vector (E1,E2,...,Ep);
```

Fig. 6.3 Pseudo code of FASP algorithm

Several scenarios with different link capacities under different traffic loads have been simulated and discussed. In our simulation, the traffic load is measured at Erlang, which is defined as the ratio of average request holding time to average request inter-arrival time.

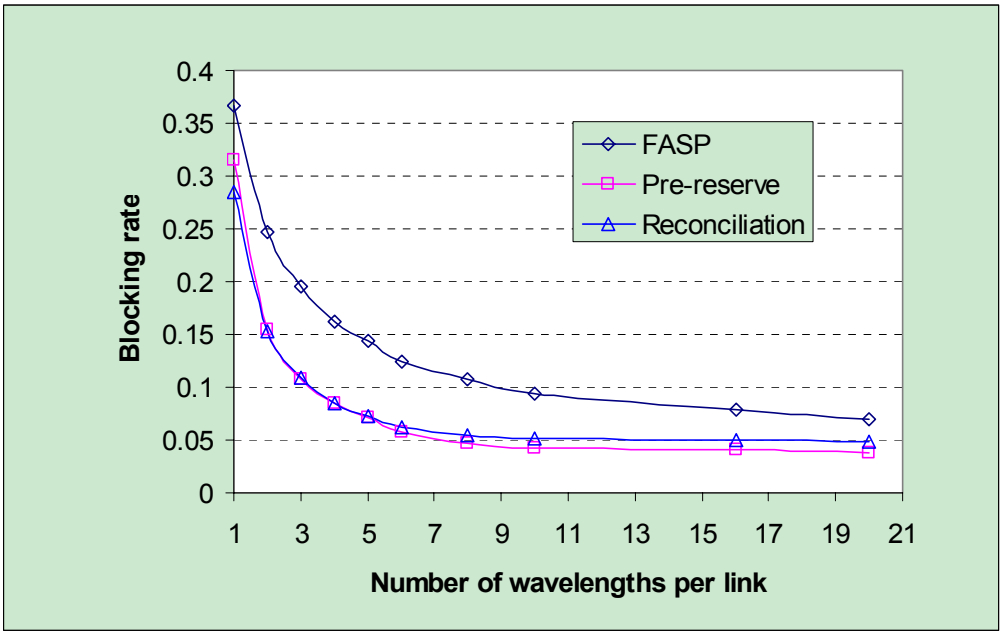

Fig. 6.4 Blocking rate with different link capacity under 1 Erlang traffic load

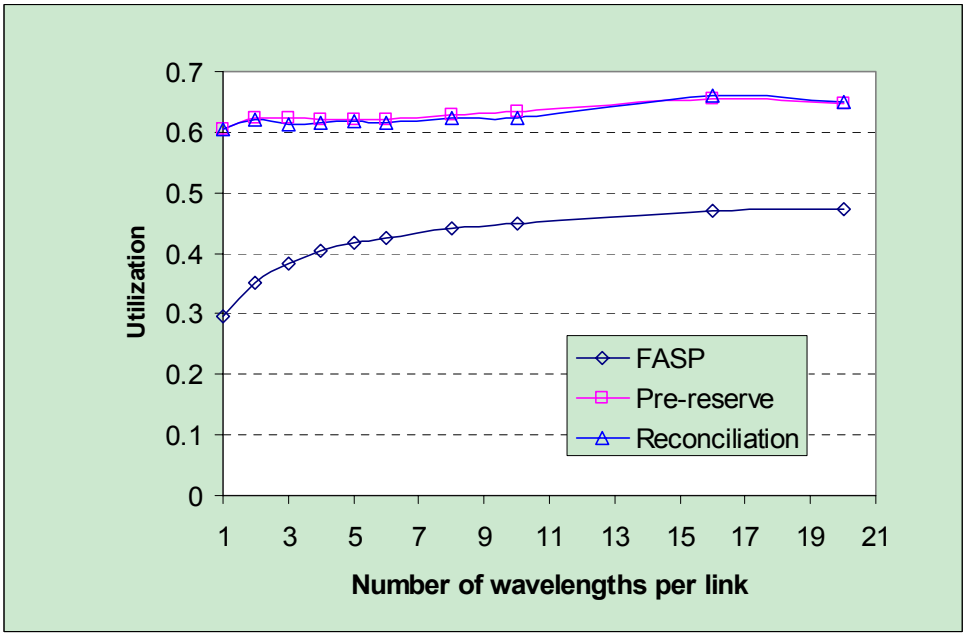

Fig. 6.5 Utilization with different link capacity under 1 Erlang traffic load

1) 1 Erlang network traffic load with different link capacities

Fig. 6.4 and Fig. 6.5 show the variation of blocking rate and utilization from the simulations based on 1 Erlang network traffic load, with the link capacities being 1, 2, 3,

4, 5, 6, 8, 10, 16, and 20 units (wavelengths), respectively. The destination is randomly selected among other nodes, with the requested bandwidth being 1 unit link capacity. Compared with FASP algorithm, both optimized scheduling algorithms show the significant performance on reducing blocking rate and improving network utilization. When the link capacity reaches a certain wavelengths, i.e., 16 wavelengths in our simulation, the blocking rate and network utilization appear stable trend.

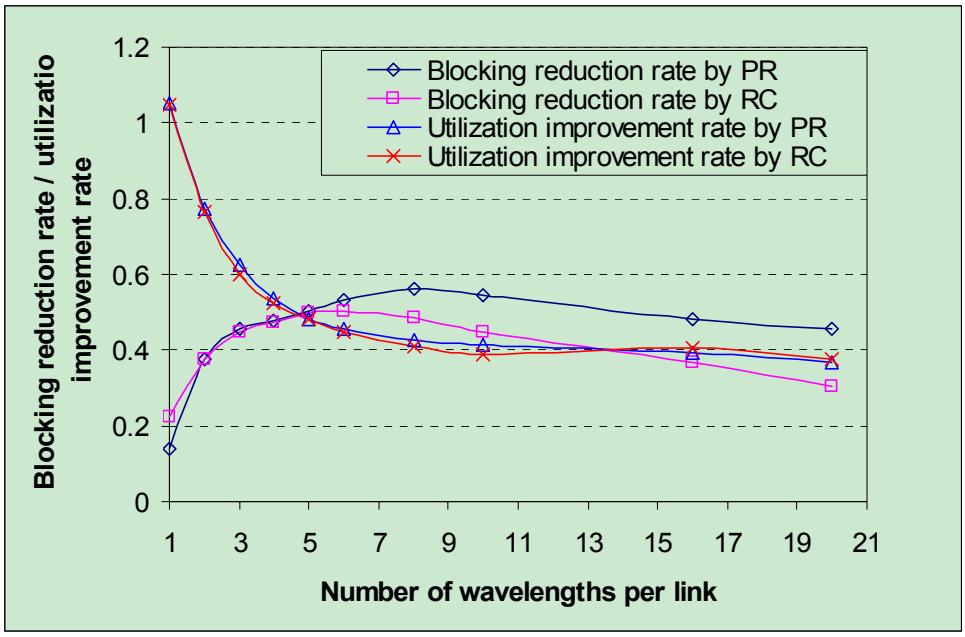

Fig. 6.6 Reduction rate of blocking rate and improvement rate of utilization with different link capacity under 1 Erlang traffic load

The reductions of blocking rate and improvements of utilization by advance scheduling algorithms compared with FASP algorithms are shown in Fig. 6.6, where PR represents pre-reserve algorithm and RC represents reconciliation algorithm. The reduction rate of blocking rate is defined as the ratio of the reduction of blocking rate to the blocking rate of FASP algorithm, while the improvement rate of utilization is defined as the ratio of the increase of utilization to the utilization of FASP algorithm.

For 1 wavelength per link, the reconciliation algorithm shows some advantage over the pre-reserve one. While both algorithms achieve about 105% utilization improvement, the reconciliation one reduces the blocking rate by about 22% whereas the pre-reserve one reduces it by about 14%. When the link capacity increases, the blocking rate reduces by about 30~55% on the average while the network utilization increases by about 40% on the average. For example, for 5 wavelengths per link, the reduction rate of blocking rate is up to around 50% while the improvement rate of utilization is up to around 48%. When the link capacity is higher than 5wavelengths, the pre-reserve algorithm shows more advantage over the reconciliation one. For the heavier traffic load, the previous connection requests accumulated in the optimization queue will be a big burden to reconciliation optimization. It takes more iterations to find the optimized solution, and it might be terminated before it finds a better solution due to the limit of the searching iterations we set in the simulations. We may increase the limit of iteration, but it would need much longer time to finish an optimization circle. On the contrary, the pre-reserve optimization actually reserves all time slots scheduled to the requests that are admitted. Therefore only new requests will participate in the new scheduling.

Fig. 6.7 illustrates the average optimization time in seconds for both scheduling algorithms. The optimization time is measured on a Linux PC with 2.0 GHz CPU and 1024M RAM. Please be noticed that the average optimization time is plotted on the logarithmic scale. Since reconciliation algorithm only reserve specific time slots for the connections requested to start at current time slots, it would re-schedule all future connection requests even though they are admitted before. Apparently, it would take much longer time to find optimizing solution. The optimization time of reconciliation

optimization is several or even tens times of what pre-reserve optimization takes. When the link capacity is 1 wavelength, the reconciliation algorithm takes about 4 times of that the pre-reserve one takes (about 164 seconds for reconciliation and 40 seconds for pre-reserve algorithm). Increasing the link capacity, both algorithms take longer time for optimization. When the link capacity reaches 8 wavelengths per link, the reconciliation algorithm takes about 10 times of that the pre-reserve one takes (about 1668 seconds for reconciliation and 176 seconds for pre-reserve algorithm). When the link capacity keeps going heavier, the average optimization of reconciliation algorithm is almost unbearable (hours). Obviously, pre-reserve optimization is the better choice for the 1 Erlang network traffic scenarios. Since the optimization time will increase significantly along the size of network increases, this model is good for a small network with several nodes.

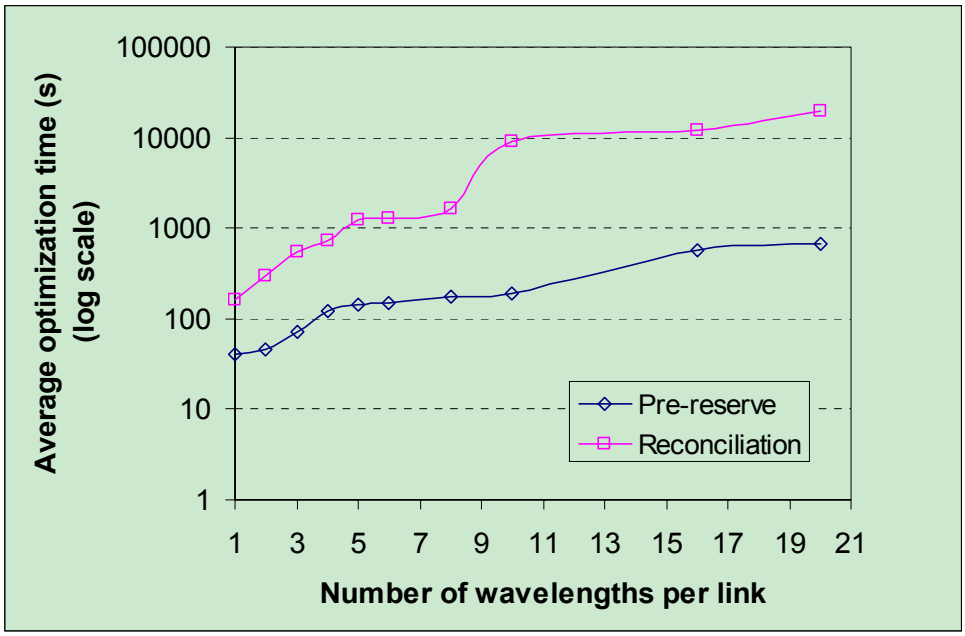

Fig. 6.7 Average optimization time of pre-reserve vs. reconciliation algorithms

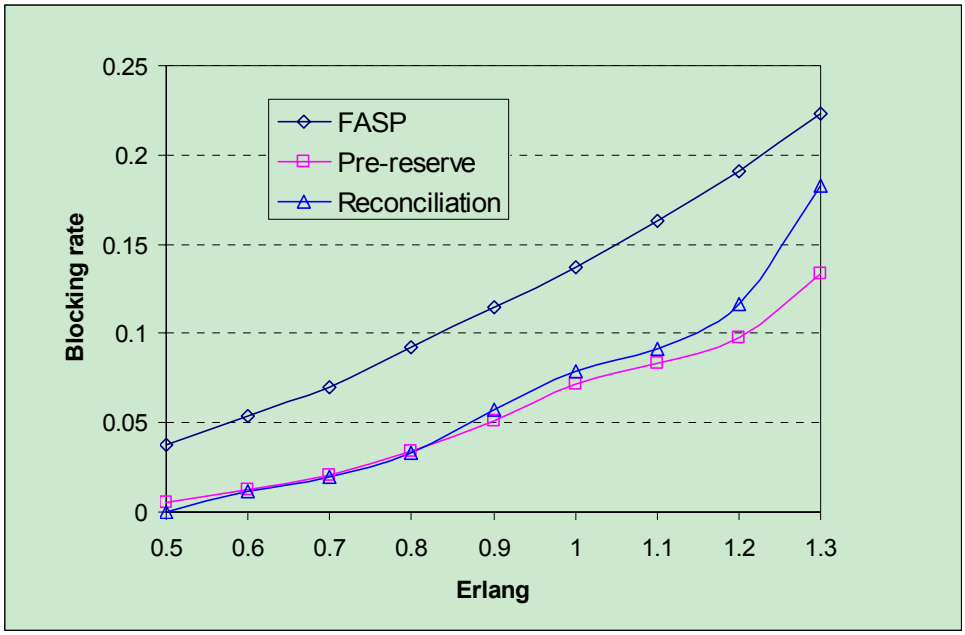

Fig. 6.8 Blocking rate with 5 wavelengths per link under different network traffic loads

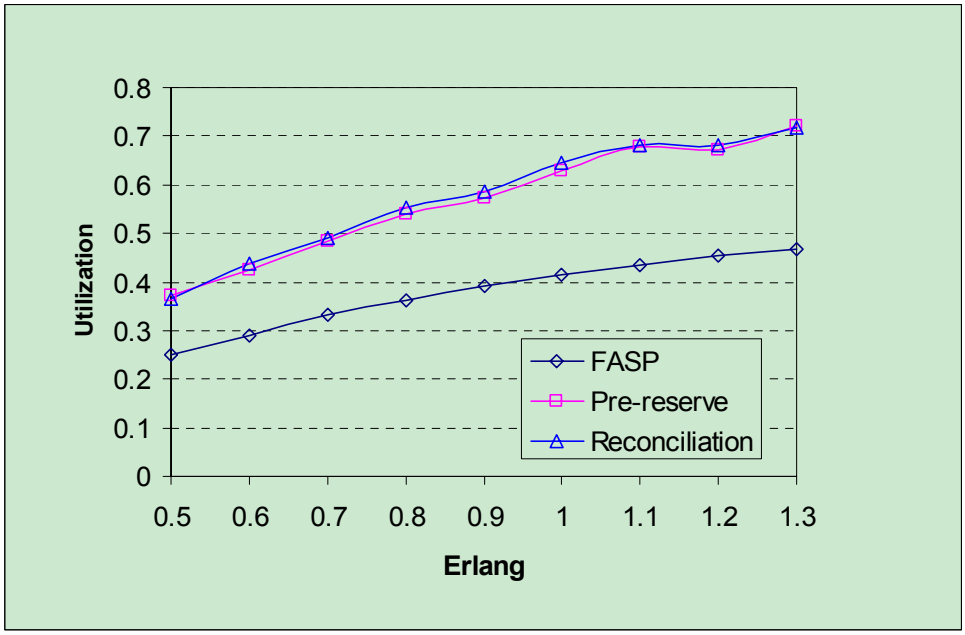

Fig. 6.9 Utilization with 5 wavelengths per link under different network traffic loads

2) 5 wavelengths with different Erlang network traffic loads

Fig. 6.8 and Fig. 6.9 show the variation of blocking rate and utilization rate from the simulations based on 5 wavelengths capacity per link, with the network traffic loads

being 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, and 1.3 Erlang, respectively. Compared with traditional FASP algorithm, both optimized scheduling algorithms significantly reduce the blocking rate and improve the utilization.

The reduction rate of blocking rate and the improvement rate of utilization by optimized scheduling compared with FASP algorithms are shown in Fig. 6.10. For the light traffic, reconciliation optimization shows a big advantage over pre-reserve one. For example, under 0.5 Erlang traffic load, the blocking rate reduces to 0% for reconciliation optimization while it's 5.2% for pre-reserve one. The reduction rate of blocking rate is up to about 100% for reconciliation optimization over 86% for pre-reserve one. When the traffic load goes up, the performance improvement of optimization decreases, but it is still magnificent. For example, under 0.8 Erlang traffic load, the reduction rate of blocking rate is up to about 64% while the improvement rate of utilization is up to about 53%. Meanwhile, the performance difference of two optimization algorithms is merging. However, when the traffic load is getting heavier, the pre-reserve optimization shows more advantage over reconciliation one on reducing the blocking rate. For example, under 1.3 Erlang traffic load, the blocking rate reduces to 18.2% for reconciliation optimization whereas 13.3% for pre-reserve one. The reduction rate of blocking rate is about 18% for reconciliation optimization whereas 40% for pre-reserve one. It's caused by the same reason as the scenarios with high link capacity under 1 Erlang network traffic load. For reconciliation optimization, the accumulation of previous connection requests needs more time and iterations to find the optimized solution, and it might be terminated before it finds one due to the limit of searching iterations we set in the simulation.

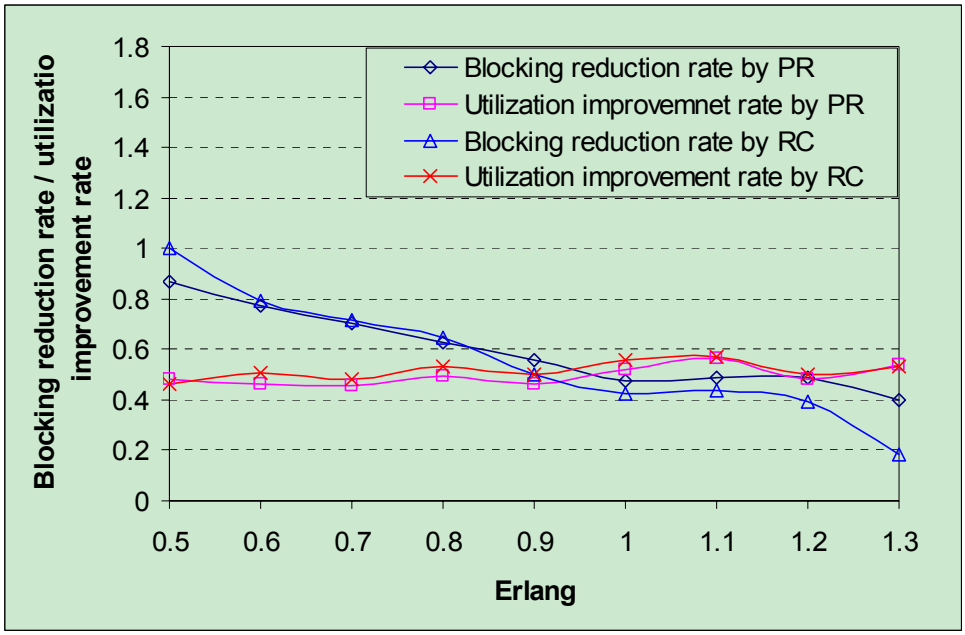

Fig. 6.10 Reduction rate of blocking rate and improvement of utilization for 5 wavelengths per link under different network traffic loads

In summary, both optimized scheduling algorithms present the significant performance on minimizing connection blocking rate while maximizing network utilizations. They perform perfectly for the small size network with not so heavy traffic load. Especially, the reconciliation algorithm is great for light traffic loading network. However, because of the complexity limitation, the proposed scheduler is not suitable to the overloaded network with many network elements.

There is a tradeoff between scalability and optimization of the network resources. Scalable solutions require distributed scheduling approaches that have the potential problems of excessive floods of control messages. On the other hand, the global optimized scheduling could achieve lower blocking rate and better utilization but requires entire network assets and long optimization time.

Because the optimization efficiency will decrease significantly along with increasing network size, the proposed scheduler is good for the small networks with several network

nodes and raw granularities. The medium and large scale networks should be logically divided to several small sub-networks. One scheduler is used for each sub-network. It could be used for intra-area as well as inter-area and inter-domain scheduling. The scheduling software could be installed on the PCEs, which could be a centralized agent responsible for a specific area, domain or even multiple domains, or domain border switches across different domains.

Chapter 7

Conclusions and Future Directions

In this chapter, the conclusions of the thesis are presented. This chapter begins with an overview of the arguments of the thesis. Following this, the main contributions of the thesis are presented. Finally, the thesis concludes by providing some indicators for future work.

7.1 Summary of Thesis

Chapter 2 presents a brief overview of GMPLS control plane and different architecture solutions for dynamic provisioning namely: centralized, distributed, and hybrid. We then describe several current experimental end-to-end dynamically provisioned IP optical networks, i.e., USN, DRAGON and CHEETAH networks.

Chapter 3 first introduces the objectives and software architecture of CVLSR implementation. It then elaborates the design and implementation of each of the CVLSR modules, i.e, parameter configuration, routing, and signaling modules. The dynamic network clustering capability and the interoperability of the CVLSR with other GMPLS engines are also described. Finally, the security mechanism for CHEETAH network is discussed.

Chapter 4 begins with the plan and implementation of CHEETAH network data plane connectivity. The CVLSR deployment and several test scenarios in CHEETAH

network across HOPI/DRAGON network are described. We demonstrate the interoperability of the CVLSR with commercial Sycamore SN16000 SONET-XC, DRAGON VLSR, and UVa RSVPD client. This chapter ends by discussing the delay of LSP establishing and throughput of established end-to-end circuits.

Chapter 5 begins with the introduction of link bundling mechanism. We then demonstrate, through experimentations and simulations, how to use the concept of link bundling for dynamic allocation of the optical circuit so that such circuits could be efficiently shared. We demonstrate that link bundling can handle the TCP/IP traffic without TCP connection abort when the buffer size of router interface is appropriately chosen. Finally, the router configuration delay is discussed.

In Chapter 6, we first discuss the requirements and challenges of scheduled connection engineering in today's IP optical networks. We present an advance optimization model based on linear programming to schedule the book-ahead connection requests subject to quality of service constraints such as minimizing connection blocking rate while maximizing network utilizations. The simulation results show us the significant improvement of the proposed approach on the desired performance, i.e., reduction of blocking rate and improvement of utilization.

7.2 Conclusions

In summary, this thesis presents the design, implementation and deployment of a GMPLS engine (CVLSR) for non-GMPLS devices such as Ethernet switch to allow them to participate in the dynamic provisioning of end-to-end bandwidth-guaranteed connections in CHEETAH network. A link bundling mechanism is then proposed to improve the efficient utilization of the optical circuits, i.e., the CHEETAH circuits can be

used for other applications (e.g., TCP traffic) or other users while there is no CHEETAH connection requested. An advance optimization model based on linear programming is finally propounded to schedule the connection requests so that the higher network utilization and lower blocking rate can be achieved.

1) Design, implementation and deployment of CVLSR

A stable CVLSR software suite has been designed, implemented and tested. It has been deployed and tested in CHEETAH network. The main features demonstrated through experiments include:

• It provides non-GMPLS devices, such as Ethernet switches, with capabilities to dynamically provision guaranteed bandwidth pipes via utilizing dynamic setup and control of VLANs within the switch.

• It allows users to share the network resources dynamically and makes it possible to extend the connections across the local area network to the end-users.

• It offers a scalable and cost-efficient solution for adding users to the network, whereas adding users directly to PoP nodes will quickly exhaust the capacity of the ports.

• It has a dynamic network clustering capability which is a possible solution to enable a broad range of dynamic clusters applications such as remote visualization, grid computing and e-learning at sub-1Gbps levels.

• It is used to enable dynamic provisioning of connections from CUNY end hosts to the other CHEETAH end hosts pass through the HOPI/DRAGON network.

• It is used to enable dynamic provisioning of connections from CUNY end hosts to the HOPI/DRAGON end hosts.

• It could be used a means for simplifying the implementation of signaling heterogeneous Paths by relieving the end-hosts from extra routing and signaling processing burden.

We have successfully demonstrated the inter-operability of the CVLSR with commercially available GMPLS enabled SONET-based cross-connect switches, i.e., Sycamore SN16000. End-to-end path setup delays revealed that the CVLSR introduced about 123 ms delay. The measured Path processing delay on CVLSR is about 29 ms, while Resv processing delay is about 91 ms, which is dominated by rate-limiting via SSH (Rate limiting via SNMP would significantly improve this delay).

Additionally, We have completed a dedicated 1GbE Ethernet connection from CCNY lab to HOPI network switch at 32 AoA, New York and from that location we connect with CHEETAH network as well as DRAGON network and could reach the sites on Internet2. We have demonstrated the connectivity through ping test and throughput measurement.

The concept of VLSR can be extended to other switch or cross-connect equipment that currently does not fully support GMPLS. For example, the VSLR can be added to the Cisco MSPP 15454 so that it can act as a signaled node in the GMPLS network. In this case, the switch fabric control sub-module controls the time slots (wavelength or fiber) instead of Ethernet port.

2) Link bundling

The concept of link bundling for dynamic allocation of the optical circuit has been demonstrated, through experimentations and simulations, so that such circuits could be efficiently shared. The following issues have been demonstrated:

• Link bundling can handle the TCP/IP traffic without TCP connection abort. The packet loss introduced by link unbundling can be recovered by TCP retransmissions.

• The required buffer size of the router interface in order to minimize the TCP retransmissions after the link unbundling operation has been identified.

• Link unbundling operation delay has been measured. In LAN environment, the minimum link unbundling delay is 13.65 ms, which equals 13.3 ms (router processing time) plus 0.35 ms (one-way propagation delay). In WAN environment (from CUNY to UVa), the minimum delay is 22.3 ms, which equals 13.3 ms (router processing time) plus 9 ms (one-way propagation delay).

3) Optimized scheduling for book-ahead services

An optimization model based on the linear programming is presented to schedule the connection in order to minimize connection blocking rate while maximize network utilizations. Two types of scheduling model, pre-reserve algorithm and reconciliation algorithm, are proposed and simulated.

• Performance evaluation results show that the proposed approaches present significant gains in terms of reducing the call blocking and improving network utilization.

• In the scenarios with different link capacities and 1 Erlang network traffic load, the optimized scheduling algorithms greatly reduce the blocking rate varying from about 14% to 56%, while improve the network utilization varying from 37% to 105% along with increase of the link capacity.

• In the scenarios with 5 wavelengths per link and different Erlang network traffic loads, the optimized scheduling algorithms greatly reduce the blocking rate varying from

about 87% to 18% while improve the network utilization varying from 46% to 57% along with increase of the traffic load.

• For the light traffic and/or low link capacity, the reconciliation optimization shows a big advantage over pre-reserve one. The performance difference of two optimization algorithms will merge when the traffic load and/or link capacity increase. However, if the traffic load is getting overloaded and/or the link capacity is getting higher, the pre-reserve optimization show a big advantage over reconciliation one.

The proposed scheduler is good for the small networks with several network nodes and raw granularities. The medium and large scale networks should be logically divided to several small sub-networks. One scheduler is used for each sub-network. It could be used for intra-area as well as inter-area and inter-domain scheduling. The scheduling software could be installed on the PCEs, which could be a centralized agent responsible for a specific area, domain or even multiple domains, or domain border switches across different domains.

7.3 Future Work

Possible future research work includes:

1) Extending the CVLSR engine to support the establishment of heterogeneous paths in multi-layer, multi-region, and multi-domain networks. For example, current Sycamore SN16000 SONET-XC does not support sub-1Gbps granularity provision. In order to support multiple end-to-end L2SC sub-1Gbps LSPs from CUNY end hosts to the end hosts in Atlanta, the CVLSR should be capable of heterogeneous functionality so that these connections could share the 1 Gbps pipe provisioned for CUNY within Sycamore cloud. Furthermore, the future high performance inter-networking consists of diverse data

plane technologies, diverse services at different layers, and diverse administrative barriers. It requires a heterogeneous inter-networking framework to support the fast dynamic end-to-end provisioning across multi-layer, multi-region, and multi-domain networks.

2) More sophisticated optimization scheduler. The traffic engineering of today's network is becoming more and more sophisticated. While new reservation requests arrive, the previously reserved requests might be cancelled or changed by the users. Additionally, the network administration might allow preempting or overbooking the network resources in order to gain more benefit. In addition, how much flexibility a user might have with respect to time, bandwidth or other potentially requirements. Furthermore, the proposed optimized scheduler is a time consuming approach, which is crucial to today's high speed network. We need to investigate alternative implementation possibilities, and derive a heuristic algorithm that meets both optimization and time expectations.

3) Inter-domain scheduling. A number of services such as grid applications, e-science, high bandwidth video distribution could benefit from the ability to schedule connections across multiple domains such as IGP areas, or BGP autonomous systems. Our optimized scheduling is based on a single routing area network. We need to investigate alternative implementation possibilities, derive and simulate a baseline of architecture for inter-domain connections scheduling. Further issues include how to deploy, and how to coordinate the schedulers between inter-domain and local connections with respect to resources and reservations, etc.

References

[1] http://public.web.cern.ch/Public/Welcome.html, accessed on Apr. 25, 2007.

[2] http://www.evlbi.org/, accessed on Apr. 25, 2007.

[3] http://www.phy.ornl.gov/tsi/, accessed on Apr. 25, 2007.

[4] http://www.csm.ornl.gov/ultranet, accessed on Apr. 25, 2007.

[5] http://cheetah.cs.virginia.edu, accessed on Apr. 25, 2007.

[6] http://dragon.maxgigapop.net, accessed on Apr. 25, 2007.

[7] http://networks.internet2.edu/hopi, accessed on Apr. 25, 2007.

[8] E. Mannie, "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, Oct. 2004.

[9] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, Jan., 2001.

[10] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, Sep., 1999.

[11] K. Kompella, Ed., and Y. Rekhter, Ed., “Routing Extensions in Support ofGeneralized Multi-Protocol Label Switching (GMPLS)”, RFC 4202, Oct., 2005.

[12] J. Moy, “OSPF Version 2”, RFC 2328, Apr., 1998.

[13] D. Katz, K. Kompella, and D. Yeung, “Traffic Engineering (TE) Extensions to OSPF Version 2”, RFC 3630, Sep., 2003.

[14] R. Callon, “Use of OSI IS-IS for routing in TCP/IP and dual environments”, RFC 1195, Dec., 1990.

[15] H. Smit and T. Li, “Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE)”, RFC 3784, Jun., 2004.

[16] L. Berger, Ed., “Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description”, RFC 3471, Jan., 2003.

[17] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin, “Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification”, RFC 2205, Sep. 1997.

[18] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, Dec., 2001.

[19] B. Jamoussi, Ed., L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty, and A. Malis, "Constraint-Based LSP Setup using LDP", RFC 3212, Jan., 2002.

[20] P. Ashwood-Smith, Ed., and L. Berger, Ed., “Generalized Multi-Protocol Label Switching (GMPLS) Signaling Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions”, RFC 3472, Jan., 2003.

[21] J. Lang, Ed., “Link Management Protocol (LMP)”, RFC 4204, Oct., 2005.

[22] Gigi Karmous-Edwards, “Photonics for E-science,” 13th Mardi Gras Conference, Frontiers of Grid Applications and Technologies, Feb., 3-5, 2005.

[23] Wesam Alanqar, Admela Jukan, “Extending End-to-End Optical Service Provisioning and Restoration in Carrier Networks: Opportunities, Issues, and Challenges,” IEEE Communications Magazine, Jan. 2004.

[24] Dimitra Simeonidou and Reza Nejabati (Editors), “Optical Network Infrastructure for Grid, Category: Informational,” Aug, 2004.

[25] A. Farrel, J. Vasseur, and J. Ash, “A Path Computation Element (PCE) –Based Architecture”, RFC 4655, Aug. 2006.

[26] Mallik Tatipamula, Zafar Ali, “A PCE-based MPLS Traffic Engineering LSP Path Computation Apprach”, IP+ Optical Network 2005.

[27] K. Shiomoto, and D. Papadimitriou, etc., “Requirements for GMPLS-based multi-region networks (MRN)”, IETF Internet draft, work in progress, Feb. 2005.

[28] T. Lehman, J. Sobieski, and B. Jabbari, “DRAGON: A Framework for Service Provisioning in Heterogenous Grid Networks,” IEEE Commun. Mag. Vol. 44, Issue 3, 84-90 (Mar. 2006).

[29] “GNU Zebra,” http://www.zebra.org/what.html, accessed on Apr. 25, 2007.

[30] M. Karsten, J. Schmitt, and R. Steinmetz, "Implementation and evaluation of the KOM RSVP engine," in Proceedings of IEEE Twentieth Annual Joint Conference on Computer and Communications Societies (INFOCOM 2001), Vol. 3, pp. 1290 – 1299.

[31] "KOM – Multimedia Communications Lab: KOM RSVP Engine," http://www.kom.tu-darmstadt.de/en/downloads/software/kom-rsvp-engine/, accessed on Apr. 25, 2007.

[32] R. Coltun, "The OSPF Opaque LSA Option", RFC 2370, Jul., 1998.

[33] K. Kompella, Ed., and Y. Rekhter, Ed., "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4203, Oct., 2005.

[34] R. Braden and L. Zhang, " Resource ReSerVation Protocol (RSVP) -- Version 1 Message Processing Rules", RFC 2209, Sep., 1997.

[35]J. Wroclawski, "The Use of RSVP with IETF Integrated Services", RFC 2210, Sep. 1997.

[36] L. Berger, Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, Jan. 2003.

[37] L. Berger, D. Gan, G. Swallow, P.Pan, F. Tommasi, and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, Apr., 2001.

[38] K. Kompella and Y. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)", RFC 3477, Jan. 2003.

[39] K. Kompella and J. Lang, " Procedures for Modifying the Resource reSerVation Protocol (RSVP)", RFC 3936, Oct., 2004.

[40] K. Kompella and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)," IETF RFC 4206, Otc. 2005.

[41] http://www.juniper.net/products/integrated/ns_5series.html, accessed on Apr. 25, 2007.

[42] http://www.openswan.org/, accessed on Apr. 25, 2007.

[43] http://www.freeradius.org/, accessed on May 4, 2007.

[44] http://www.postgresql.org/, accessed on May 4, 2007.

[45] http://expect.nist.gov, the Expect home page, accessed on Jul. 6, 2006.

[46] W. Feng and P. Tinnakornsrisuphapá, "The Failure of TCP in High-Performance Computational Grids," Proc. of SC2000: High-Performance Network and Computing Conference, Dallas, TX, Nov. 2000.

[47] S. Floyd, "HighSpeed TCP and Quick-Start for Fast Long-Distance Networks," Workshop on Protocols for Fast Long-Distance Networks, CERN, Geneva, Switzerand, Feb. 3-4, 2003.

[48] R. Skoog, A. Von Lehmen, and G. Clapp, etc., "Metro Network Design Methodologies That Build a Next-Generation Network Infrastructure Based on This Generation's Services and Demands," Lightwave Technology, Journal of Volume 22, Issue 11, Nov. 2004 Page(s):2680 – 2692.

[49] J.W. Gannett, G. Clapp, R.A. Skoog, and A. Von Lehmen, "Performance of IP over Optical Networks with Dynamic Bandwidth Allocation," Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC Volume 5, 6-11 March 2005 Page(s):3 pp. Vol. 6.

[50] Veeraraghavan, M., Fang, X., Zheng, Z., "On the suitability of applications for GMPLS networks", submitted to ICC 2006.

[51] http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html, accessed on May 14, 2007.

[52] J. Fang and A. K. Somani, "IP Traffic Grooming over WDM Optical Networks," 9th Optical Network Design and Modelling, Milano, February 7-9, 2005.

[53] Kuri J., Puech N., Gagnaire M., Dotaro E., and Douville R., "A Routing and Wavelength Assignment of Scheduled Lightpath Demands," IEEE Journal on Selected Areas in Communications, Vol.21, No. 8, pp.1231-1240, Oct. 2003.

[54] http://www.gnu.org/, accessed on May 14, 2007.

Publications

1. Z. Li, Q. Song, and I. Habib, "CHEETAH Virtual Label Switching Router for Dynamic Provisioning in Connection Oriented Networks," submitted to Journal of Optical Networking.

2. Q. Song, Z. Li, and I. Habib, "Cheetah Virtual Label Switch Router Design and Deployment in GMPLS Optical Networks," Global Telecommunications Conference, 2006, GLOBECOM '06. IEEE, Vol. 4, 28 Nov.-2 Dec. 2006. Page(s):2067 – 2071.

3. I. Habib, Q. Song, Z. Li, and N.S.V., Rao, "Deployment of the GMPLS control plane for grid applications in experimental high-performance networks," Communications Magazine, IEEE, Vol. 44, Issue 3, March 2006 Page(s):65-73.

4. 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," Communications, 2006 IEEE International Conference on, Vol. 6, 11-15 June 2006 Page(s):2710-2716.

5. Q. Song, Z. Li, I. Habib, and W. Alanqar, "Value-added proposition of the GMPLS control plane in IP optical networks," Journal of Optical Networking. Vol. 4, Issue 12, Page(s): 838-855 (2005).

6. Q. Song, Z. Li, and I. Habib, "Using link bundling for efficient utilization of dynamically provisioned optical circuits," Optical Network Design and Modeling, 2005. Conference on Feb. 7-9, 2005. Page(s):205 - 211