

User Controlled Lightpath Management System based on a Service Oriented Architecture

Biswajit Nandy, Don Bennett
Solana Networks
{bndandy, dbennett}@solananetworks.com

Imran Ahmad, Shikharesh Majumdar
SCE, Carleton University
{iahmad, majumdar}@sce.carleton.ca

Bill St.Arnaud
CANARIE
bill.st.arnaud@canarie.ca

Abstract — This paper describes a User Controlled Lightpath provisioning and configuration management system. The system allows users to obtain network resources from multiple domains and combine these resources forming *Articulated Private Networks*. This unique ability to acquire lightpaths, interfaces, and tunnels from a variety of suppliers, and offer value added services to end customers, is the trend for next generation networking. The system is designed and implemented based on a Service Oriented Architecture and Web Services Resource Framework (WSRF) specifications.

I. INTRODUCTION

In recent years there have been several initiatives to build next generation networks that address the following needs:

- An increasing number of system integrators, content suppliers, and service providers are building their own private networks that integrate wavelengths, SONET channels, and MPLS tunnels from different suppliers. They want to manage this heterogeneous mix of resources as a single network, and offer services to their customers. The ability to build a virtual network from multiple physical network resources will provide a cost effective way to build such a network.
- Leading network researchers [2] are demanding "parallel virtual" networks, rather than dedicated networks, in order to experiment with new protocols and services. Representing switches, routers, and links as web services allows researchers to easily reconfigure their network, supporting their experimentation. More importantly, it also allows a host to be connected simultaneously to both a traditional IP network and a more advanced network running innovative services. This will allow for much easier migration to a future incarnation of the Internet.
- Many data intensive applications, including high-energy physics experiments and medical applications, require the flexibility to be able to establish connectivity when needed. By providing a means by which users are able to control and manage resources in the network, UCLPv2 offers this flexibility.

UCLPv2 [1] provides, to users, the ability to build and manage complex optical networks, known as *Articulated Private Networks* (APN), across multiple administrative domains. These networks comprise many varieties of network devices and communication links including lightpaths, switches,

routers, instruments, and sensors. These components are exposed as, and managed through, web services. Users should be able to build complex virtual networks by harvesting and manipulating these resources.

Physical entities, such as network devices and connections, are represented as web services defined by Web Services Description Language (WSDL) documents. These services provide flexible management and control plane functionality upon which one can base complex applications and architectures. UCLPv2 is one effort that enables customers to control resource management of optical networks and to create virtual networks across multiple domains. CANARIE currently sponsors the development of three UCLPv2 systems for the CA*NET4 network.

The remainder of this paper describes one implementation of UCLPv2. Section 2 captures related work in the field. Section 3 discusses various UCLPv2 concepts. Section 4 describes the domain server architecture and various novel features of the system. Section 5 describes domain server functionalities. Section 6 briefly outlines the client architecture.

II. RELATED WORK

This section discusses a representative set of background work related to the current UCLPv2 development efforts.

As discussed in [3], early versions of UCLP lacked a dynamic topology discovery process, making it hard to extend UCLP to other networks. The work presented in [3] overcomes these limitations by introducing a neighbor discovery procedure and by extending the range of equipment that the system supports. [3] also introduced the concept of "auto-routing". Users can either manually configure lightpaths using a provided GUI, or they can use auto-routing to automatically configure lightpaths using an intelligent algorithm.

[4] discusses a customizable resource management function for grids deployed over user controlled optical networks. It describes a modular approach towards resource management by introducing the concept of three distinct service layers: User Access, Service Provisioning, and Resource Management Layers. The Service Provisioning Layer is managed with a grid application. Customers and administrators utilize the User Access Layer to configure and use end-to-end UCLP resources. The lowest level resources are managed with the Resource Management Layer. [7] expands on this layered

architecture and introduces the concept of a centralized registry to discover resources. It also provides better support for multi-domain environments.

Giga Spaces uses a service-oriented architecture that focuses on providing user controlled provisioning of optical networks [6]. Another paper, [5], discusses a web-services based three-layer architecture for user ownership and control of lightpaths in an optical network.

Current UCLPv2 system is designed and implemented based on Service Oriented Architecture with interactive GUI as user control panel. Also, this version introduces the concept of Articulated Private Network.

III. UCLPv2 CONCEPTS

UCLPv2 is a lightpath management system where authorized users can obtain network resources from a pool of available resources and manage them according to their requirements. For example, Domain A publishes all resources that are available for use by other domains. Domain B can obtain required resources in the form of a “resource list” that contains lightpaths and connection points. After obtaining the resources, Domain B may wish to concatenate a portion of resources obtained from A with its own lightpaths to form the required network. Resources obtained from Domain A, along with resources from Domain B, can be offered as a lease to a third party, Domain C. Domain C then harvests these resources and manages as necessary. Resources are returned to their original domain after their leases expire. Users within a domain are capable of manipulating lightpaths to form networks of lightpaths. The ability to partition lightpaths is an important feature that allows one to manage resources in a domain. A “resource based” authentication and authorization system provides a very flexible mechanism for protecting access to secure network resources.

There are four basic objects that are used in the currently implemented UCLPv2 system:

- *Network Element (NE) Objects* represent an abstraction of physical network elements. Through these objects, one can manipulate the resources of network elements independent of the protocol used to communicate with them. Examples of network elements include routers, switches, scientific instruments, and other network devices.
- A *Connection Point Object (CPO)* represents an interface that forms one end of a light path. For example, a CPO can represent a GigE port, a set of STS channels, an OC192 port, or a port on a router or an instrument.
- A *Light Path Object (LPO)* represents a path between two points, represented as Connection Points. LPOs are building blocks that can be concatenated to form

composite LPOs, or partitioned into LPOs with smaller bandwidth.

- An *APN Resource List* is a set of LPOs and CPOs that are managed as a single entity. They can be published for consumption by third parties, or harvested from third parties for use in the local network. The harvester of a resource list can connect LPOs at any of the listed CPOs, or connect CPOs to form a larger light path.

IV. DOMAIN SERVER ARCHITECTURE

An optical network controlled by UCLPv2 may span multiple domains, each of which is managed by a domain server. The UCLPv2 domain server comprises several functional blocks that interoperate to provide the required resource management services. The functional building blocks, and their high level relationships, are illustrated in Figure 1 and summarized below.

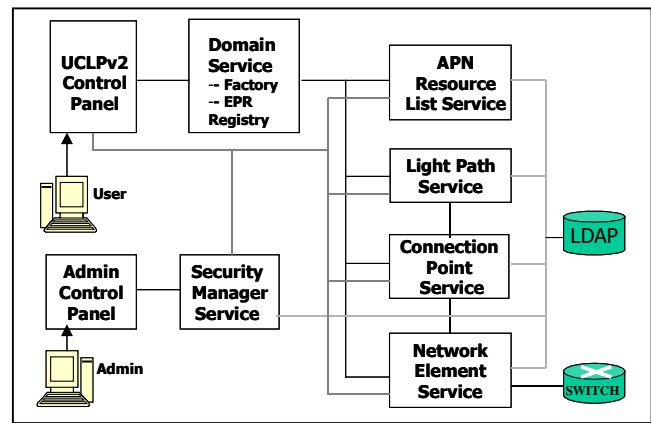


Figure 1: Domain Server Architecture

The functional building blocks, shown in Figure 1, are as follows:

- *Domain Service* – This service comprises a resource factory that is responsible for allocating WS-Resources for NEs, LPOs, and APNs. It also maintains a registry of qualified end point references to objects within a domain and allows users to obtain user specific views of managed resources.
- *Connection Point Service* – A stateful service that represents a connection point. There may be several different implementations of the CPO service to handle the various types of possible network element interfaces.
- *Network Element Service* – A stateful service representing various types of elements within a network. This service encapsulates the communication protocol required to communicate with the network devices they represent. These services are primarily used to create and manage CPOs that represent interfaces on the device.
- *Light Path Service* – A stateful service that represents a logical connection between two CPOs and provides operations for managing the path.
- *APN Resource List Service* – A stateful service representing an APN Resource List that contains a list of

lightpaths and connection points, and provides an interface with which to manage these resources.

- *Security Manager Service* – A service that is used to manage user accounts and provide user authentication and authorization functionality.
- *UCLPv2 Control Panel* – This is the graphical interface with which one can create and manipulate resources that are managed by the UCLPv2 system.
- *Admin Control Panel* – This is a second graphical interface, used to create and manage users within the UCLPv2 system.

The domain server is characterized by a number of novel features that are briefly discussed below.

A. Scalability

Each domain server maintains and manages the state of resources owned by that domain. Once a resource is harvested from another domain, its state is maintained and managed locally. For example, if Domain B harvests a lightpath from Domain A, Domain B manages all operations on this lightpath, and all associated state. This approach promotes scalability since all state associated with resources is managed within the domain of the current owner. A domain may contain a large number of resources from different domains but cross-domain state management is not required.

B. Security

Proper mechanisms for authentication and authorization are pivotal for a distributed resource management system like UCLPv2. Authentication mechanisms ensure that users of the system are who they claim to be. Once authenticated, users are able to access a set of resource based on their authorization with said resources.

Encryption is used to prevent eavesdropping and message integrity when communicating with UCLPv2 services. All communication between a client and services takes place over SSL. Each UCLP user is issued a digital certificate, signed by a trusted Certification Authority. This certificate, a username, and a password are required to authenticate a particular user. At server side, user credentials are stored in a LDAP repository. Each user is authorized to access a subset of UCLP resources. Users may have guest, privileged, or ownership status on various resources. This status determines the operations available to different users. To gain access to resources, users may contact either their domain administrator, or the owner of the resources in question.

C. Notifications

The Observer pattern [8] is a powerful way to respond to situations that arise without the tight coupling of notification producer and consumer. This pattern is used throughout the UCLPv2 data model. The web services, which model resources within a management domain, use notification to broadcast state changes. An example of this is notification upon expiration of a lease on an object. For example, consider the expiration of a lease on a LPO. Once the lease expires, the LPO web service notifies the two end point CPO services of

the situation. The CPOs, upon receiving this notification, notifies their respective NE services to inform them that they're no longer in use. The NEs reclaim the CPOs and make their resources available for future use.

D. Error Handling & Fault Management

Two types of error handling are provided: synchronous and asynchronous. In case of synchronous errors, an error message is returned to the user as soon as the error is detected. Examples of synchronous error handling include logical errors as well as authentication and security related errors. In case of asynchronous error handling, a notification can be generated much later than the fault that triggered it. A user receives a notification only when she/he attempts to use a faulty component. Examples of such faults include physical faults such as a connection point failure, a switch failure, and a fiber cut. Since a lower level fault, such as a connection point failure, can give rise to a lightpath failure the system uses an algorithm to propagate fault notifications upwards to the appropriate higher-level objects. The algorithm is "distributed" in the sense that it can propagate notification for faults detected in one domain to another domain for objects that span multiple domains.

E. Interoperability

Web-services based implementations promote interoperability through the use of standard service descriptions based on XML. For example, although the client and server in different domains can be implemented on top of different platforms, they can still communicate with one another and share resources. As long as implementations are based on the same set of service descriptions, they should be interoperable. Alternatively, by developing a simple, standard, representation of core resources, proxy services can be employed to automatically translate operations from one implementation to another.

V. DOMAIN SERVER FUNCTIONALITIES

Important Domain Server functionalities, such as lightpath concatenation and partitioning, and resource list publishing and harvesting, are described in detail below:

A. LPO Concatenation

Lightpath concatenation is the process whereby two or more distinct lightpaths are joined, end-to-end, forming a new, composite lightpath. This new lightpath can be manipulated in the same manner as the sub-paths that comprise it.

One initiates this operation by invoking the *Concatenate* operation of the Domain Service, passing a list of lightpath references as one of the arguments. For example, consider the scenario depicted in Figure 2.

Here we see two lightpath objects, A and B, each of which contains two connection point objects. Each of the connection point objects reference the network element object that created it, and therefore represent a resource on the switch associated with their network element. Notice that CP2 and CP3 both

reference NE2, indicating that CP2 and CP3 represent logical interfaces on the same switch.

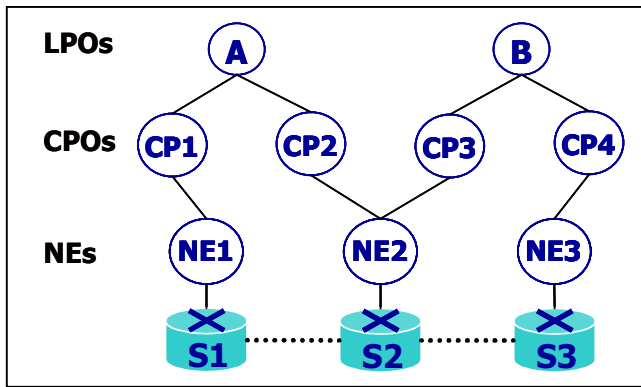


Figure 2: Concatenation of two LPOs

The *Concatenate* algorithm loops over the entire list of lightpaths, provided as an argument, attempting to connect each lightpath to the next one in the list. In order to connect two paths together, one must determine which ends of the paths terminate on the same network element. Once this pair of points has been determined, their *Connect* operation is invoked with the other connection point as its argument. In the example, this translates to the calls *CP2.Connect(CP1)* and *CP1.Connect(CP2)*. Each of these *Connect* calls results in a call to the *CreateCrossConnect* operation of their associated network element, passing both connection points as arguments. The first *CreateCrossConnect* call sets up the connection request and the second call commits the resources. Once all sub-paths have been connected end-to-end, a new lightpath object is created to represent the new path. A reference to this new path is returned by *Concatenate*.

B. LPO Partitioning

Partitioning a lightpath essentially divides its bandwidth among a number of child paths, each of which terminates on the same network elements as the parent. One initiates this process by invoking the *Partition* operation, providing the bandwidth requirements of each child as one of the arguments.

Partition is a recursive operation. If one wishes to partition a composite lightpath (as described above), it must be broken down into its components. This results in the disconnecting of any cross connections made at switches along the path. As illustrated in Figure 3, *Partition* is recursively invoked on all sub-paths of the lightpath object. The *Partition* operation returns a list of references to the newly created child lightpaths of the lightpath object on which the operation was invoked. Composite lightpaths must be reformed once all the sub-paths have been partitioned. The previously described *Concatenate* operation is used for this.

If a lightpath object contains no sub-paths, *Partition* is invoked on the two connection point objects that represent the ends of the lightpath. As above, the connection point *Partition*

operation returns a list of child connection point references. Once a lightpath has partitioned its endpoints, child lightpaths are created from pairs of the child connection points. These new lightpath objects are returned from the lightpath *Partition* operation.

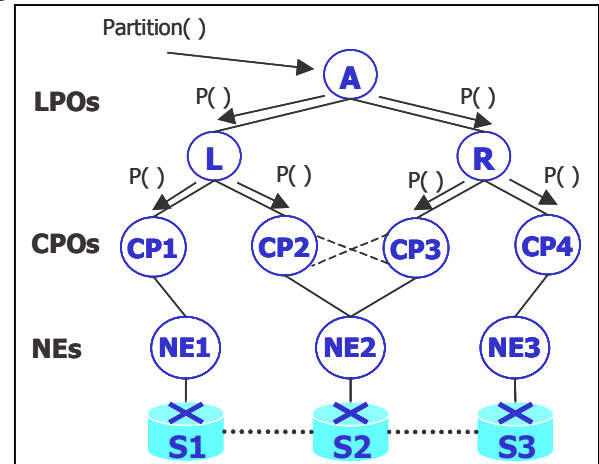


Figure 3: Partitioning of a composite LPO

The connection point *Partition* operation is relatively simple. Once the requested bandwidth configuration is verified against that which is possible on the associated switch, new connection point objects are created to represent the partitioned interfaces. A list of these new objects is returned to the caller.

C. APN Resource List Publishing

An APN Resource List contains a list of lightpath and connection point objects. By “publishing” this list, it is made available for harvesting by others. The harvester of this list is given access to all of the objects contained within, and can therefore manipulate associated resources. Typically, an administrator within one domain publishes these lists so that users in other domains can lease the advertised resources. One who wishes to lease resources from a particular domain must be authenticated with the domain and authorized for the resources.

The act of publishing a resource list for use by a particular individual or group involves modifying the authorization privileges for all objects contained within the list. The leaser of a resource list must have privileged access to the list itself and all contained objects. Resource lists should not be modified once they have been published. Once a resource list has been published, the leaser must obtain a copy of an endpoint reference to the published resource. For example, these endpoint references could be posted on and obtained from a web page, distributed via emails, or written on a piece of paper and given to the recipient. Alternative representations of resource lists, including an XML document describing the list’s contents, could also be used to promote interoperability.

D. APN Harvesting

Before one can make use of resources offered by another domain, one must harvest a resource list published from the source domain. As mentioned previously, before one is able to harvest a resource list, one must have privileged access to it and all contained resources.

The act of harvesting a resource list involves making local proxy objects to represent resources in the source domain. For example, a local lightpath proxy web service would instantiate a resource to represent a harvested light path. Users within the local domain can operate on this proxy light path in the same way as a local lightpath. The proxy can be connected to other light paths and partitioned as if it were a local resource. Proxies are made of the two Connection Points that represent the ends of the path, except that they maintain references to the original Network Elements. This is necessary because Connection Points require the services of Network Elements to make or break cross-connects between light paths.

An APN resource list can only be harvested once from its source domain. Essentially, when one harvests a resource list, one assumes responsibility for the resources contained within. The harvester is responsible for making use of contained resources and releasing the lease on the resource list when no longer needed.

VI. CLIENT ARCHITECTURE

The Client part of the UCLPv2 implementation framework is designed as a control panel application. This Control Panel application is used to allocate and manage resources within managerial domains. It provides a graphical interface to the web services defined in the previous sections. The Control Panel application comprises three major components:

- Graphical User Interface – handles user input and data presentation.
- Data Model – provides a local model of the resources present in a network.
- Web Service Interaction – Invokes web services, upon user request, to allocate and manage resources within a network, and handles asynchronous change notification sent from the services.

VII. SUMMARY

In this paper, we have described the design of a user controlled lightpath management system. The management system enables a set of basic functionalities like concatenation and partitioning so that users can build and manage complex optical networks, across multiple administrative domains. The system is designed based on Service Oriented Architecture.

The system is currently being implemented based on WSRF specification in Globus Toolkit Version 4 [9]. Java Axis is used as the SOAP engine which marshal and de-marshal SOAP request. Jakarta Tomcat is used as the application server, which provides a 'living space' for applications that

must be accessed by different clients. The SOAP engine runs as an application inside the application server.

Once the developed UCLPv2 system is deployed on live network, various performance characterization of WSRF based system will be undertaken. Also our development experience of a SOA based management system will be shared with the research community.

ACKNOWLEDGMENT

We would like to thank Rupinder Makkar and Bo Zhao of Solana Networks for various discussion and development work for UCLPv2. Gegs Jones has significant contribution as project manager of this project. We would also like to thank Herve Guy and Jun Jian of Canarie for many discussions at the early stage of this project.

REFERENCES

- [1] Bill St. Arnaud, "UCLP Roadmap for creating User Controlled and Architected Networks using Service Oriented Architecture", CANARIE., January 2006.
- [2] T. Anderson, L. Peterson, S. Shenker, J. Turner, "Overcoming the Internet Impasse through Virtualization", IEEE Computer, April 2005 (Vol. 38, No. 4) pp. 34-41
- [3] Recio, J.; Grasa, E.; Figuerola, S.; Junyent, G., "Evolution of the user controlled lightpath provisioning system"; Transparent Optical Networks, 2005, Proceedings of 2005 7th International Conference Volume 1, 3-7 July 2005 Page(s):263 - 266 Vol. 1
- [4] R. Boutaba, W. Golab, Y. Iraqi, B. St-Arnaud, "Grid-Controlled Lightpaths for High Performance Grid Applications", Journal of Grid Computing, Vol.1, No.4, December 2003, pp. 387-394
- [5] R. Boutaba, W. Golab, Y. Iraqi, B. St-Arnaud, "Lightpaths on Demand: A Web Services-Based Management System", IEEE Communications Magazine, Vol. 42, No. 7, July 2004.
- [6] W. Hong, "Giga Spaces", Department of Physics, Carleton University, Ottawa, 2006, available from: http://www.gigaspace.com/academic/Carleton_University.html
- [7] Jing Wu; Hanxi Zhang; Campbell, S.; Savoie, M.; Bochmann, G.V.; St Arnaud, B. "A grid oriented lightpath provisioning system", Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004. IEEE 29 Nov.-3 Dec. 2004 Page(s):395 – 399.
- [8] E. Gamma et-al, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Professional
- [9] Globus: WSRF – The WS Resource Framework, <http://www.globus.org/wsr/>