Statement of Work

User Controlled Lightpaths Project

Prepared for: CANARIE Inc. and CISCO Canada Inc.

Date: May 6, 2003

Submitted by:
Prof. Raouf Boutaba (rboutaba@bbcr.uwaterloo.ca)

School of Computer Science
University of Waterloo
200 University Avenue West
Waterloo, ON N2L 3G1
Canada

1 Cover Sheet

Customer Controlled Lighpaths Project

Prepared for: CANARIE Inc. and CISCO Canada Inc.

Submitted by: Prof. Raouf Boutaba
(rboutaba@bbcr.uwaterloo.ca)

School of Computer Science
University of Waterloo
200 University Avenue West
Waterloo, ON N2L 3G1
Canada

Other Participants: Mohamed Dadi
Acila Derbal
Youssef Iraqi
Adel Ghlamallah
Wojciech Golab
Boris Jabes
Basem Shihada
Iban Touchet

Start date: 02/06/03
End date: 10/30/04

2 Project Summary

The objective of the project is to provide a software system that enables user control of lightpaths through interaction with CA*net 4 lightpath cross-connect devices. The system allows users to create end-to-end lightpaths across multiple management domains as well as to partition a single lightpath into sub-lightpaths that can be advertised and leased to other users. The system offers a Web-based interface for interaction with users as well as a Grid services interface for interaction with Grid applications.

3 Project Overview

The objective of the project is to provide a software system that enables user control of lightpaths through interaction with CA*net 4 lightpath cross-connect devices. The original proposal responds to a call for proposals by CANARIE Inc. and Cisco Canada Inc., under a shared-cost, CA*net 4 Directed Research Program.

The software allows users to create end-to-end lightpaths across multiple management domains as well as to partition a single lightpath into sub-lightpaths that can be advertised and leased to other users. Domain administrators and customers interact with the system using a Web-based interface. Domain administrators are able to advertise existing lightpaths within a management domain. Customers are able to lease advertised lightpaths and re-advertise portions of such for lease. Functionality related to lightpath manipulation, including advertisement, concatenation, partitioning, and termination, is implemented in the form of Grid services. These comprise a separate layer that can be directly accessed by Grid applications. Actual changes to lightpath-related data objects and to the configuration of CA*net 4 network hardware devices are effected via a set of resource agents. Resource agents provide a virtualization of hardware resources, allowing customers to exercise control over the subset of resources dedicated to their own lightpaths. The collection of resource agents maintains lightpath-related data in a distributed object space.

The project contributes to the goals of CA*net 4 by realizing the concept of a "customer-empowered network." Our software places dynamic allocation of network resources in the hands of users by allowing users to concatenate and partition lightpaths. By providing a standard Grid interface, our software also facilitates the deployment of high bandwidth Grid applications between research institutions across Canada. Thus, the project promotes the widespread adoption of a next-generation user-controlled optical Internet and reinforces Canada's leadership role in this endeavour, satisfying the overall purpose of the CA*net 4 Directed Research Program.

The project further contributes to specific program objectives as follows. As described earlier, our software allows users to control a subset of resources on CA*net 4 cross-connect devices. Moreover our user access layer provides a means to offer this service to a set of authorized users only. By allowing users to advertise portions of lightpaths for lease, our software makes it possible to delegate control of the corresponding subset of hardware resources to other

authorized users. The software provides a Web-based human interface as well as a Grid services interface compliant with the OGSA standard that allows Grid applications such as GridFTP to directly query lightpath availability and create bandwidth-guaranteed paths across the network. Standards-based technologies such as JavaSpaces and ebXML are used to store lightpath-related data and to advertise services for querying these data to Grid applications. Industry standard protocols such as TL1 are used by resource agents to interact with network hardware devices.

The result of the project, namely the software system, will be evaluated against two objectives: correctness with respect to the specification and satisfaction of program goals. Evaluation against the first objective will be performed by the project team members, following the testing plan outlined in the Detail Project Plan section. Evaluation against the second objective will be performed by CANARIE during the process of reviewing progress reports and deliverables. In addition, CANARIE staff providing the design team with technical expertise will be in a position to evaluate the team members' growing understanding of the CA*net 4 environment and comment on the degree to which the team's vision coincides with program goals on a technical level.

4 Participants and Their Contributions

4.1 Organisations Involved

This project will be realized by the University of Waterloo, which is renowned for its innovation and commitment to advancing knowledge. For the 11th year in a row, Maclean's national reputation survey has recognized Waterloo as the best overall and most innovative among 47 universities across Canada[1].

Waterloo is known worldwide for its commitment to both curiosity-driven and applied research, promoting not only the discovery of new knowledge but also the application of that knowledge in novel ways. The faculty, staff, and students of the university attracted more than $115.5 million in research funding from public and private sources in 2001-02. This amount represents a 38 percent increase from $84 million in 2000–01.

Waterloo is also a leader in partnering with the private sector. Currently, eleven of the university's NSERC Research Chairs are industry-sponsored. This relationship promotes the dissemination of new knowledge and technological advances for the benefit of society. At the same time, the university assists researchers in commercializing the results of their work through its Technology Transfer and Licensing Office. Many high-tech and knowledge-based companies trace their roots to the University of Waterloo, including Waterloo Maple, Open Text, Dalsa, Certicom, Intelligent Mechatronic Systems, Sirific Wireless, Ignis Innovation, and Research in Motion.

[1] http://www.adm.uwaterloo.ca/infoipa/macleans.html

The University of Waterloo project team will rely on CANARIE staff to provide technical advice concerning the CA*net 4 environment, network hardware in particular, and to share their vision of customer empowered networks.

4.2 Participants

4.2.1 Raouf Boutaba

Position: Head of the Network Management Group, University of Waterloo.
Associate Professor in the School of Computer Science, University of Waterloo

Email address: rboutaba@bbcr.uwaterloo.ca

Role in the project: Project Leader

Responsibilities:

- organize human resources and assign roles
- oversee all project activities
- monitor project progress
- delegate management duties to Chief Research Assistant

Relevant qualifications:

- leadership and management skills demonstrated during extensive research experience

Relevant experience:

- led the development of the Virtual Network Resource Management System for customer management of networks
- led numerous research and development projects both in the industry and academia
- ten years of experience as Associate/Assistant Professor at a university
- five years of experience as a research scientist in the industry

4.2.2 Mohamed Dadi

Position: Software Designer and Developer

Email address: mddadi@yahoo.com

Role in the project: Database Administrator

Responsibilities:

- install and fine-tune database
- create database schema

Relevant qualifications:

- relational database modeling

- database installation and tuning in context of web based applications
- SQL specialist
- ORACLE, MySQL, SQL Server
- Oracle 9ias, Weblogic

Relevant experience:
- 7 years experience in database field as consultant for various clients, including National Defense, Sureté du Québec

4.2.3 Acila Derbal

Position: Freelance Graphic Designer

Email address: aciloo@yahoo.com

Role in the project: Web Interface Designer

Responsibilities:
- design and develop Web pages in user access layer

Relevant qualifications:
- thorough knowledge of Adobe Photoshop, Adobe Illustrator and Macromedia Fireworks for Photo editing and interface design
- Macromedia Flash for Web Animation
- site building with Macromedia Dreamweaver
- HTML, CSS

Relevant experience:
- worked as part of a designing team for ERFA Canada producing an online catalogue; tools used: Dreamweaver, Photoshop, HTML and CSS
- designing online ads for various clients of wedodesign.net; tools used: Flash and Fireworks
- college work experience as a Dreamweaver instructor's assistant

4.2.4 Youssef Iraqi

Position: Postdoctoral Fellow, University of Waterloo. Ph.D. University of Montreal.

Email address: iraqi@bbcr.uwaterloo.ca

Role in the project: Chief Research Assistant

Responsibilities:
- organize team meetings
- administer project Web site
- assist Project Leader in management duties

Relevant qualifications:

- Java, C++, C, Pascal, Assemblers 6800, 6809, 68000, DBaseIV, Informix 4GL, SQL, Fortran, Prolog, Lisp, HTML, SDL, SDT, LOTOS, Matlab and various other software packages

Relevant experience:

- participated in the development of a management architecture for self-configurable networks
- investigated the use of the active networks paradigm in the Internet
- investigated the use of intelligent agents for network management
- developed a dynamic bandwidth allocation algorithm for wireless networks
- Student Member of the Institute of Electrical and Electronics Engineers (IEEE)
- Vice-Chair of the IEEE Kitchener-Waterloo Communications, Information Theory and Vehicular Technology joint chapter since 2001
- Chief Teaching Assistant, Winter 2002, University Waterloo
- Chief Teaching Assistant, Fall 2001, University Waterloo
- Chief Teaching Assistant, 1995-1999, University Montreal

4.2.5 Boris Jabes

Email address: bsjabes@uwaterloo.ca

Role in the project: Co-op Student

Responsibilities:

- assist in programming, testing, and documentation of all three layers of the software system

4.2.6 Adel Ghlamallah

Position: Consultant and Senior Programmer/Analyst – Nomade Technologies

Email address: oka_adel@yahoo.com

Role in the project: Architect

Responsibilities:

- responsible for overall system architecture
- design and develop different software layers
- coordinate the work of Research Assistants and Co-op Students
- configuration control

Relevant qualifications:

- experience in architecture and design of web-based and distributed systems
- Web-related technologies: JSP/Servlet, JavaScript, HTTP, HTML...

- Apache and Iplanet Web servers, Tomcat and Jrun Web-Containers, as well as WebLogic and Jboss Server-Applications.
- EJB, JMS, JNDI and Web-Services
- UP and XP methodologies
- Java, C and Perl

Relevant experience:
- architecture of Web-based complex systems (including network services system, and phone call management system)
- lead two technical teams of 4 to 6 developers at BCE-Emergis and OKA-Info
- more than 5 years of design and development of object-oriented distributed systems

4.2.7 Wojciech (`voi-tek) Golab

Position: Master of Math Candidate, School of Computer Science, University of Waterloo

Email address: wgolab@uwaterloo.ca

Role in the project: Research Assistant

Responsibilities:
- responsible for resource management layer
- design resource allocation algorithms

Relevant qualifications:
- Java, C, C++, JavaScript
- HTML forms, HTTP, CGI, Java servlets, JSP
- Apache Tomcat, Apache HTTP Server
- X.509 certificates, OpenSSL
- multithreaded client/server socket programming in Java and Microsoft Visual C++
- Matlab, Maple
- algorithmic aspects of optical network provisioning

Relevant experience:
- full-time summer and subsequent part-time work in Java servlet development at the University of Toronto at Scarborough, Department of Statistics
- full-time summer co-op positions in Web development at CIBC Mortgages Inc. and cryptographic software development at JAWZ Inc.
- full-time summer position in database development at the University of Toronto at Scarborough, Computer Science Co-op Department

4.2.8 Basem Shihada

Position: Ph. D. Candidate, School of Computer Science, University of Waterloo

Email address: bshihada@bbcr.uwaterloo.ca

Role in the project: Research Assistant

Responsibilities:
- responsible for user access layer and service provisioning layer
- interconnect different technologies
- design and develop Grid interfaces and applications

Relevant qualifications:
- EJB, WSDL, XML, Java servlets, SOAP, SOAP/HTTP, ebXML, Open Grid Service Architecture (OGSA)
- Novell networks (installation and configuration), Windows NT and Linux based nodes
- Virtual Private Networks (IPSec) and Active networks
- Network programming with TCP/IP, UDP sockets, and multithreaded task operations
- ISDN, PPP, X.25, ATM
- BGP, IGRP, RIP, OSPF, SNMP
- C/C++, JAVA, PASCAL, PLAN, Visual BASIC, Assembly, PROLOG and HTML

Relevant experience:
- Parallel Computing: implemented Parallel Sorting with Regular Sampling (Shi & Schaeffer), with Message Passing Interface on LAM using a maximum of 4 processes under one sequential processor
- Process Object Oriented Design: designed and implemented a simulation of computer networks performing an e-mail application using multithreaded task computing
- Methodology of Software Evaluation: surveyed and provided a critical software evaluation to four programming development environments: Borland C++ 5.5 vs. Microsoft Visual C++ 6.0 and JAVA Workshop 2.0 vs. Visual Cafe

4.2.9 Iban Touchet

Position: Exchange Student, University of Waterloo

Email address: itouchet@engmail.uwaterloo.ca

Role in the project: Co-op Student

Responsibilities:
- interface with network hardware, Cisco ONS 15454 in particular
- provide an API for lightpath configuration

Relevant qualifications:
- C, JAVA
- LAN Technology (switching, STP, trunking), TCP/IP, routing protocols
- Cisco IOS, Cisco CLI

Relevant experience:

- RouterSim / Boson CISCO network simulator
- simulator of the 68000 CPU written in C
- complete kernel of a multi-tasking operating system (PIII architecture) written in C

4.3 Summary of Human Resources

The success of the project depends on the contributions of each member of the team. Raouf Boutaba is an experienced team leader. He was the Director of the Telecommunications and Distributed Systems Division in the Computer Science Research Institute of Montreal from 1995 to late 1997, and is currently the head of the Network Management Group at the University of Waterloo. Youssef Iraqi has demonstrated leadership ability and organization skills while holding the position of Chief Teaching Assistant from 1995 to present. Adel Ghlamallah has extensive experience in Java programming and the software development process. The remaining team members have diverse but complementary profiles: Mohamed Dadi is an industry professional knowledgeable in database design, Acila Derbal is an experienced graphics designer, Wojciech Golab has experience with interactive Web technologies and has studied optical network provisioning algorithms, Basem Shihada is familiar with Java and Grid technologies, and Iban Touchet is studying Cisco product configuration. Together, members of the project team are able to provide the leadership, organization, and technical competence necessary to carry the project through to completion.

5 Technology Architecture and Implementation Plan

5.1 Software Architecture

Please refer to Appendix: Software Architecture and Appendix: Draft XML Schema and WSDL Descriptors.

5.2 Policy Components and Support for Extensions

Policy decisions are made throughout the system in order to control access to lightpath resources. In user-to-system interactions, policy decisions are made on the basis of the user ID. For example, only a domain administrator is permitted to create root-LPOs, and only the owner of an LPO is allowed to advertise it for lease. Users can also police the use of their own lightpaths by means of the programmable controller module of a resource agent. For example, a customer may wish to operate customized routing within the virtual network consisting of his own lightpaths. This can be achieved by uploading appropriate scripts to the resource agents corresponding to the relevant CA*net 4 lightpath cross-connect devices.

Depending on the CANARIE requirements, policy components can be implemented in various ways. In the proposed implementation, policy components are realized in the form of Java classes in the user access layer and scripts in the resource management layer. A moderate level of extensibility can be achieved by allowing the scripts to be upgraded dynamically using

the Web-based interface. As an extension to the proposed software system, a more sophisticated policy-based-management framework can be realized by implementing the standardized COPS protocol along with the COPS-PR client. For example, the Meta-Policy Information Base Platform, an open source Java package, can be incorporated. This package enables flexible specification and efficient distribution of policies using the meta-policy concept[2].

5.3 Integration with Grid Architecture Standards

The proposed software system integrates with Grid standards by implementing the OGSA/OGSI Grid services framework within the service provisioning layer. This allows Grid applications to interface with the system directly to query lightpath availability and create lightpaths for large data transfers.

Note that because Grid technologies have not yet matured, integration of our system with Grid standards is dependent on the availability of a stable OGSA framework.

5.4 Support for Heterogeneous Network Equipment

The resource management layer provides an abstraction of network hardware to the upper layers. The switch interface component of the resource agent serves as the link between our software and CA*net 4 hardware devices. At one end, the switch interface exposes a hardware-independent API to the other components of the resource agent. At the other end, it communicates with hardware devices through a protocol layer that implements industry standard protocols such as TL1 and SNMP. The need for compatibility with heterogeneous network equipment also calls for a middle layer consisting of a set of hardware adapter modules that translate API calls into appropriate protocol messages. This middle layer resolves functional differences between devices of different types (e.g. optical cross-connect vs. SONET switch) as well as vendor-related differences between devices of like types.

Loss of hardware compatibility resulting from changes to CA*net 4 network hardware can be remedied through a combination of adding hardware adapter modules in the middle layer of the switch interface and incorporating additional protocol implementation packages in the protocol layer.

5.5 Future Upgrades and Extensions

Any errors in the code or documentation written by the project team that emerge during the maintenance phase will be corrected. Updates will be made available through the project Web site. This process will occur on an ad-hoc basis throughout the maintenance phase.

Major upgrades and extensions may be undertaken under a separate project agreement with CANARIE. The remainder of this section outlines possible directions for such work.

[2] A. Polyrakis and R. Boutaba. The Meta-Policy Information Base. *IEEE Network*, 16(2):40-48, March/April 2002.

The system makes extensive use of open source software packages. Some of these, OGSA, for example, are early releases subject to significant changes in the near future. As newer, more stable releases of these components appear, and the corresponding technologies mature, upgrades to the system may be desired to improve reliability and facilitating gradual evolution.

Software upgrades may also be necessitated by evolution of CA*net 4 network hardware. As discussed under Support for Heterogeneous Network Equipment, the switch interface component of the resource agent module may need to be updated if compatibility problems arise with new hardware devices.

Extensions to the system are possible along several directions. As discussed under Policy Components and Support for Extensions, the standardized COPS and COPS-PR protocols can be implemented for flexible policy-based management. In order to achieve greater flexibility and scalability, the user access layer and service provisioning layer can also be decentralized. For example, each management domain can execute its own instance of these layers. This entails significant increase in development effort due to the need to resolve resource contention issues during simultaneous creation of overlapping end-to-end lightpaths, and the need to manage trust relationships between domains. Another possible extension involves adding functionality in the service provisioning layer to enable the creation of survivable lightpaths. For example, path protection can be incorporated as an option during the process of end-to-end lightpath creation, and restoration mechanisms can be applied to unprotected lightpaths.

5.6 Technical Support and Maintenance

One Research Assistant will be assigned on a part-time basis to the task of providing technical support and software maintenance for a period of one year after the completed software system is submitted to CANARIE. This individual will be specified in the software documentation and will be reachable through a University of Waterloo email address.

As discussed under Future Upgrades and Extensions, maintenance duties will be limited to correcting code and documentation written by the project team.

6 Detailed Project Plan

This section defines the dates, milestones, and deliverables that will drive the project based on the phases and iterations required to build the proposed software system.

6.1 Assumptions and Constraints

A working release of the software is intended to be available by the end of October. The plans and the milestones presented in this document are based on the following assumptions:

- access to a test environment with the necessary equipment (switches) and the underlying optical network during all the development phases

- availability of Cisco switches and CANARIE network experts
- availability of stable open source components related to Grid interfaces, OGSA framework in particular

The principal risks are related to Grid technologies and the capability of network hardware to support external (software) configuration. Grid technologies and the related OGSA framework are still in the early stages of evolution and it is difficult to predict whether current software releases will prove sufficiently stable. An alternative would be to use other interface technologies instead (e.g. Web Services, RMI, or CORBA). The other issue concerns the configuration of network hardware through external software. It is not clear how this will be implemented in the system and supported by hardware devices.

6.2 Project Deliverables

The following deliverables will be produced during the project:

- Vision: a high-level description of the system, produced during the Inception phase. It defines the project scope and contains the critical features the software must provide to the customer.
- Project Plan: an action plan for developing the required system, produced during the Inception phase and refined in subsequent phases. It includes schedules, project plans, commitments, and resources. It describes the process for designing, implementing, documenting, and testing the software system. The estimates and plans in this document will continually change during the project.
- Use Case Model: describes the proposed functionality of the system based on detailed use cases. This document is created in the Inception phase and evolves during the Elaboration and Construction phases.
- Supplementary Specification: captures the system requirements that are not captured in the use cases of the Use Case Model, for example the performance and reliability of the system.
- Software Architecture Document: provides a source of technical information throughout the project. At the end of Elaboration, it may contain detailed descriptions for the architecturally significant use cases, as well as identify key mechanisms and design elements.
- Design Model: describes how the software is structured into packages and classes using different UML diagrams. Produced during Elaboration and Construction phases.
- Iteration Plan: details the use cases, scenarios, and other work items corresponding to an iteration. This document is produced one iteration in advance.
- Test Plan: provides an overview of the test effort throughout the life of the project. This document is created during the Elaboration phase and updated during Construction and Transition phases.

- Deployment Plan: describes the set of tasks necessary to install, test, and effectively transition the product to the target environment. This document will be produced during Construction phase and refined during the Transition phase.
- Configuration Management Plan: describes configuration and control management tasks to be performed during the project. This document will be started in Inception phase and refined during subsequent phases.
- User Manual: will be produced at the end of the project.
- Demonstrations (see Demonstrations section)

6.3 Project Organization

6.3.1 Organizational Structure

The organizational chart of the project team is presented in

Figure 1.

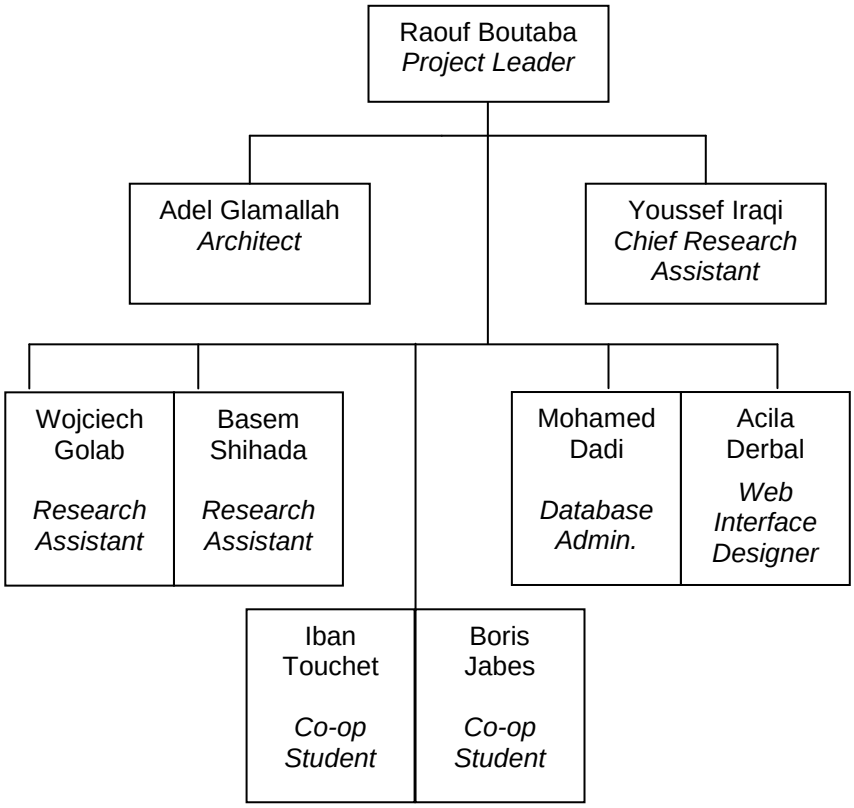

Figure 1: Team Organization.

6.3.2 Interaction with CANARIE

The Project Leader will provide Status Assessment, as scheduled in the plan, to CANARIE. The project team will also interact with CANARIE staff to obtain technical advice and to solicit feedback regarding deliverables and other relevant artefacts.

6.3.3 Roles and Responsibilities

The team members involved in the project, their roles, and their responsibilities are identified in Table 1.

Name	Role	Responsibilities
Raouf Boutaba	Project Leader	▪ organize human resources and assign roles ▪ oversee all project activities ▪ monitor project progress ▪ delegate management duties to Chief Research Assistant
Youssef Iraqi	Chief Research Assistant	▪ organize team meetings ▪ administer project Web site ▪ assist Project Leader in management duties
Adel Ghlamallah	Architect	▪ responsible for overall system architecture ▪ design and develop different software layers ▪ coordinate the work of Research Assistants and Co-op Students ▪ configuration control of the project
Basem Shihada	Research Assistant 1	▪ responsible for user access layer and service provisioning layer ▪ interconnect different technologies ▪ design and develop Grid interfaces and applications
Wojciech Golab	Research Assistant 2	▪ responsible for resource management layer ▪ design resource allocation algorithms
Canarie staff Darcy Quesnel	CA*net 4 Expert Technical liaison	▪ provide support during various activities
Mohamed Dadi	Database Administrator	▪ install and fine-tune database ▪ create database schema
Acila Derbal	Web Interface Designer	▪ design and develop Web pages in user access layer
Iban Touchet	Co-op Student 1	▪ implement modules to access and configure CA*net 4 lightpath cross-connect devices
Boris Jabes	Co-op Student 2	▪ assist in programming, testing, and documentation of all three layers

Table 1: Roles and Responsibilities.

6.3.4 Team Management

Members of the project team will be located in the Kitcehener/Waterloo area (Raouf Boutaba, Youssef Iraqi, Boris Jabes, Wojciech Golab, Basem Shihada, Iban Touchet), and in Montreal (Mohamed Dadi, Acila Derbal, Adel Ghlamallah). We propose the following solutions to address the challenges associated with a distributed team:

- *Communication*
  Team members can use many different forms of communication to bridge the physical distance, including telephone conferences and e-mail. In addition, meetings will be held regularly at the University of Waterloo to foster inter-personal relationships.

- *Availability*
  A weekly schedule of the availability of each team member will be prepared and made easily accessible to the entire team. Design and testing sessions will be scheduled based on this availability.

- *Management*
  Weekly reports from all the team members will be sent to the Project Leader. Regular feedback will be given to team members in order to steer the project and promote open communication.

- *Configuration Management*
  A Web-based CVS will be used as a repository to share documents and to control remote access to source code.

In addition to periodic meetings of the entire project team, participants at the University of Waterloo will also meet 1-2 times per week.

Disputes within the project team will be resolved by senior team members. Disputes regarding software architecture and other technical details will be decided by the Architect, subject to final approval by the Project Leader. Disputes regarding the assignment of roles and responsibilities will be settled by the Project Leader. In the event that a participant withdraws or is withdrawn from the project, duties will be temporarily reassigned to other team members and a replacement will be sought from a large resource pool consisting of current University of Waterloo students in the School of Computer Science, past students of Professor Raouf Boutaba, and other skilled contacts developed by team members during their academic and professional careers.

6.4 Development Process

The development cycle of the proposed software system will be conducted in phases, where each phase consists of one or more iterations. This approach, based on the Unified Process (UP), is highly dynamic and adaptive. It will give us the ability to accommodate changes to requirements, technologies, and goals.

6.4.1 Phase Plan

The proposed phases and relative timeline of the development cycle are shown in Table 2.

Phase	Number of Iterations	Iteration length (weeks)
Inception	1	2
Elaboration	2	5
Construction	3	4
Transition	1	3
Maintenance	many	52

Table 2: Development cycle: phases and iterations.

The iterations are described in more detail in under Iterations and Use Cases.

The related phases and major milestones are described in Table 3.

Phase	Description	Milestone
Inception	The Inception Phase will develop the product requirements and establish the scope for the system. The major use cases will be developed as well as a high level Software Development Plan.	**Business Case Review** at the end of the phase, when the project is well scoped and funded.
Elaboration	The Elaboration Phase will analyze the requirements and will develop the architectural system. At the completion of the Elaboration Phase, most use cases (70%) will have been completed for analysis and design. The high risk use cases will be considered first. The architectural system will test the feasibility and performance of the architecture that is required.	**Architectural Prototype** at the end of the phase. The prototype will include major architectural components and will set a requirements baseline.
Construction	During the Construction Phase, remaining use cases will be analyzed and designed. The pre-release version of the software will be developed and test activities will be completed.	**Pre-release Version** at the end of the phase, including full functionality.
Transition	The Transition Phase will prepare the pre-release version of the software for deployment, ensuring smooth installation. User training will occur here.	**Release Version** at the end of the phase.
Maintenance	The Maintenance Phase will take place after the release version of the system is deployed. It will be used to identify and fix remaining defects in code and documentation..	**Maintenance Version(s)** of the system.

Table 3: Project phases and major milestones.

6.4.2 Iterations and Use Cases

Each phase in the development cycle consists of iterations in which a subset of the system is designed, specified, developed, or tested. In general, these iterations

- reduce technical risk
- provide early versions of a partially working system
- allow maximum flexibility in planning features for each release

Each iteration is based on one or more use cases. A use case describes a feature of the system and is defined as a sequence of actions performed by a system that yields a result of value to the user. Example use cases are briefly described in Table 4.

Use Case Name	UC1 : Logon
Brief Description	Authenticate user based on username/password, authorize access.
Use Case Name	UC2 : Advertise LPO
Brief Description	Advertise the availability of an LP.
Use Case Name	UC3 : Terminate LPO
Brief Description	Find the specified LPO and terminate it.
Use Case Name	UC4 : Search LPO
Brief Description	Retrieve and return a list of LPO(s) that match specified attributes.
Use Case Name	UC5 : Establish End-to-End LP
Brief Description	Find and concatenate appropriate component LPs to establish an end-to-end LP between specified endpoints.
Use Case Name	UC6 : Reconfigure LPO
Brief Description	Modify a specified LPO's attributes. For example, increase or reduce the reserved bandwidth.
Use Case Name	UC7 : Access LPO
Brief Description	Allow the owner to gain access to and directly use the specified LPO.

Table 4: Use Cases.

A detailed plan for each iteration will be produced one iteration in advance (see Project Schedule section).

The maintenance cycle will be considered a series of repeated iterations, each iteration involving the following activities:

- evaluate severity of remaining defects and impact of corrections, prioritize defects
- identifies a subset of defects to be addressed in this iteration
- plan specific tests to validate corrections to defects
- make corrections and execute tests, repeat until all tests are passed
- update the design documents if necessary

- release updated software and documentation

6.4.3 Demonstrations

The following demonstrations will be provided during the development cycle:

- at the end of the elaboration phase, an Architectural Prototype will be presented with critical features
- at the end of the construction phase, a Pre-release Version of the software will be provided with a full set of features
- at the end of the transition phase, a Release Version of the software will be made available

The planned content of the system to be presented for demonstrations is expected to change as the project progresses. This may be due to a number of factors. In particular benefit, effort, and risk will be considered in prioritising product requirements.

6.5 Project Schedule

High-level schedule showing project phases, iterations, and milestones is shown in Table 5 and Table 6.

Milestones	Start Date	End Date
All Milestones	02/06/03	10/30/03
Business Case Review (Inception Phase)	02/06/03	05/13/03
Architectural Prototype Version 1 (Iteration 1 of Elaboration Phase)	05/14/03	06/18/03
Architectural Prototype Version 2 (Iteration 2 of Elaboration Phase)	06/19/03	07/22/03
Pre-release Version 1 (Iteration 1 of Construction Phase)	07/23/03	08/20/03
Pre-release Version 2 (Iteration 2 of Construction Phase)	08/21/03	09/17/03
Pre-release Version 3 (Iteration 3 of Construction Phase)	09/18/03	10/14/03
Release (Beta) Version (Transition Phase)	10/15/03	10/29/03
(Maintenance Phase)	10/30/03	10/30/04

Table 5: Schedule of Milestones

Task ID	Deliverables / Milestones / Task Names	Budget[3]	Effort / Completion Date
	Inception Phase		**52 d**
	Vision		**28 d**
	Project Scope		2 d

[3] Budget amounts are computed using average per diem.

	Requirements		20 d
	Define project requirement		
	High level use cases		
	Non-functional requirements		
	Define system constraints		
	Define Resources		1 d
	Environment		2 d
	Risk List		3 d
	Configuration Management		**12 d**
	Iteration plan for next iteration		**1 d**
	Management		**3 d**
	Meetings		**8 d**
	Inception Phase complete		*05/13/03*
	Elaboration Phase		**232 d**
	Iteration I – Develop Architectural Prototype Version 1		**121 d**
	Configure development environment (development and test tools)		**11 d**
	Install and configure Database		5 d
	Install and configure OGSA		5 d
	Install and configure Tomcat		1 d
	Requirements (Use Case Model)		**5 d**
	Detail selected use cases		5 d
	Web User Interface		**15 d**
	User Interface Modeling		5 d
	User Interface Prototype		10 d
	Design Model		**19 d**
	Design sequence diagrams for selected use cases		3 d
	Design class diagrams for selected use cases		3 d
	Design database		5 d
	Design Grid service interface		4 d
	Design modules to access switches		4 d
	Implementation		**44 d**
	Implement selected use cases		24 d
	Implement Grid interfaces		8 d
	Create Web interface		5 d
	Update database schema		3 d
	Fix defects		4 d
	Test		**8 d**
	Plan tests		1 d
	Implement tests		7 d
	Management		**4 d**
	Iteration plan for next iteration		1 d
	Staff the project		3 d
	Meetings		**15 d**
	Iteration I complete		*06/18/03*
	Iteration II – Develop Architectural Prototype		**111 d**

	Version 2		
	Environment		**6 d**
	Fine-tune and configure development environment		3 d
	Fine-tune and configure		3 d
	Refine Requirements (Use case Model)		**5 d**
	Detail use cases		5 d
	Web User Interface		**10 d**
	Refine user interface prototype		10 d
	Design Model		**18 d**
	Revisit architectural analysis and design		2 d
	Design sequence diagrams for selected use cases		3 d
	Design class diagrams for selected use cases		3 d
	Design database		4 d
	Design Grid service interface		3 d
	Design modules to access switches		3 d
	Implementation		**45 d**
	Implement selected use cases		24 d
	Implement Grid interfaces		8 d
	Update database schema		4 d
	Create Web interface		5 d
	Fix defects		4 d
	Test		**8 d**
	Plan tests		1 d
	Implement tests		7 d
	Management		**4 d**
	Iteration plan for next iteration		1 d
	Staff the project		3 d
	Meetings		**15 d**
	Elaboration Phase complete		*07/22/03*
	Construction Phase		**320 d**
	Iteration I – Develop Pre-Release Version 1		**113 d**
	Requirements (Use case Model)		**5 d**
	Refine and detail use cases		5 d
	Web User Interface		**5 d**
	Build Web pages		5 d
	Design Model		**19 d**
	Design sequence diagrams for selected use cases		5 d
	Design class diagrams for selected use cases		5 d
	Design database		3 d
	Design Grid Applications (FTP…)		3 d
	Design resource agent		3 d
	Implementation		**59 d**
	Implement selected use cases		28 d
	Implement resource agent components		10 d
	Update database schema		3 d
	Create Web Interface		5 d
	Implement Grid interfaces		8 d

	Fix defects		5 d
	Test (Unit and Functional Tests)		**6 d**
	Plan tests		1 d
	Implement tests		5 d
	Management		**4 d**
	Iteration plan for next iteration		1 d
	Staff the project		3 d
	Meetings		**15 d**
	Iteration I complete		*08/20/03*
	Iteration II – Develop Pre-Release Version 2		**104 d**
	Refine Requirements (Use case Model)		**3 d**
	Refine and detail use cases		3 d
	Design Model		**17 d**
	Design sequence diagrams for selected use cases		4 d
	Design class diagrams for selected use cases		4 d
	Design database		3 d
	Design resource agent		3 d
	Design Grid applications integration		3 d
	Implementation		**59 d**
	Implement selected use cases		28 d
	Implement Grid applications		8 d
	Implement resource agent components		10 d
	Update database schema		3 d
	Create Web interface		5 d
	Fix defects		5 d
	Test (Unit and Functional Tests)		**6 d**
	Plan tests		1 d
	Implement tests		5 d
	Management		**4 d**
	Iteration plan for next iteration		1 d
	Staff the project		3 d
	Meetings		**15 d**
	Iteration II complete		*09/17/03*
	Iteration III – Develop Pre-Release Version 3		**103 d**
	Refine Requirements (Use case model)		**3 d**
	Refine and detail use cases		3 d
	Design Model		**17 d**
	Design Sequence diagrams for selected use cases		4 d
	Design Class diagrams for selected use cases		4 d
	Design resource agent		3 d
	Design Database		3 d
	Design Grid Applications Integration		3 d
	Implementation		**59 d**
	Implement selected use cases		28 d
	Implement Grid Applications		8 d
	Implement Resource Agents Components		10 d
	Update database schema		3 d
	Create Web Interface		5 d
	Fix Defects		5 d

	Test (Unit and Functional Tests)		**6 d**
	Plan Test		1 d
	Implement Tests		5 d
	Management		**3 d**
	Staff the project		3 d
	Meetings		**15 d**
	Construction Phase complete		*10/14/03*
	Transition Phase - Develop Release Version		**90 d**
	Tests and bug Fix		40 d
	Test and Fix Performances issues		15 d
	Prepare Deployment		10 d
	Install and configure database		8 d
	Management		**11 d**
	Staff the project		11 d
	Meetings		6 d
	Transition Phase complete		*10/29/03*
	Maintenance Phase		**?**
	Develop an iteration plan		?
	Plan Tests		?
	Correct Defects		?
	Prepare Deployment		?
	Management		?
	Staff the project		
	Meetings		?
	Maintenance Phase complete		*10/30/04*

Table 6: Development Plan.

The Development Plan presented in Table 6 will be revised prior to the start of each iteration phase. Question marks denote the fact that the effort required to complete tasks in the maintenance phase is difficult to estimate at this time.

7 Project Budget

8 Intellectual Property

The intellectual property expected to arise from this project comprises the software system (source code) and the design documents. University of Waterloo researchers shall retain the Intellectual Property generated during this project. It is usually the goal of University of Waterloo researchers to disseminate the research results through publications in conferences/journals and open source software.

9 Web Site Information

The project Web site URL is http://bbcr.uwaterloo.ca/~canarie/. The site is protected using HTTP authentication. Credentials will be provided once content appears. A publicly accessible Web site will also be made available.

10 Appendix: Resumes of Key Project Personnel

11 Appendix: Software Architecture

11.1 Introduction

This section provides detailed information on the architecture of the proposed software system. We begin this section by defining relevant terms and explaining the design philosophy. In particular, the notion of LP spawning is introduced and discussed in detail. Then an overview of the overall architecture is presented. Next we explain in detail how each component inside the architecture functions and we specify what protocols are used for inter-component communications. Technologies chosen to implement each component are also justified. Some scenarios of typical LP operations are described at the implementation level.

11.2 Terminologies and Definitions

This section provides definitions of terms that will be used throughout this document.

Terms	**Definitions**
Management Domain	The set of optical network resources under the control of a single entity such as a regional network provider, an enterprise, a university, a hospital, or a government department.
LightPath (LP)	A unidirectional point (source) to point (destination) connection with effective guaranteed bandwidth [1]. Examples of a LP include: - Analog wavelength on a CWDM or DWDM system - STS channel on a SONET or SDH circuit - ATM CBR circuit - DiffServ "gold" service on a packet based network - Gigabit Ethernet over dedicated fiber strand An end-to-end LP is the concatenation of multiple inter-domain LPs.
LightPath Object (LPO)	An LPO is an abstract representation of a LP. An inter-domain or Root LPO represents an inter-domain LP while an end-to-end or Compound LPO represents an end-to-end LP. LPOs can be concatenated to form Compound LPOs. An LPO may also be partitioned into smaller (bandwidth-wise) Partitioned LPOs.
Customer	A customer is a user of the proposed software system and accesses it through a Web interface.
Domain Administrator	A domain administrator of domain A is an entity that has authority to control and manage the inter-domain LPs within domain A. It accesses the software system through a Web interface.
Grid Application	Grid application queries and accesses Grid resources through the standard OGSA interface. One example is GridFTP for file transfers.

Grid Service	A Grid service is one type of Web services that conforms to the OGSA standard. That is, it allows Grid applications to discover and access it using standard OGSA methods. We use both terms interchangeably throughout this document.
System	This refers to the proposed software system to be implemented.

Table 7: Terms and definitions table.

11.3 Key Concepts

In this section, we describe and explain the key concepts behind the design of the proposed software system. We begin by defining the abstraction process of LPs as LPOs and discuss the set of operations that can be applied on them. Then the ownership issue of an LPO is described, followed by a detailed discussion of the key LP operations including LPO spawning, LPO termination and end-to-end LPO establishment.

11.3.1 Abstraction of LPs

As defined earlier, an LPO is an abstract representation of an established LP in the network. There are two types of LPs: an inter-domain LP and an end-to-end LP. In order to allow operations on the LPs, inter-domain LPs are represented as Root LPOs and end-to-end LPs are represented as C-LPOs (Compound LPOs). Several operations can be applied to LPOs. The process of partitioning a subset of an LPO's resources to form another LPO is referred to as LPO spawning. The newly constructed LPO from the previous operation is called a P-LPO (Partitioned LPO). The process of concatenating several LPOs together to form another LPO is referred to as LPO composing. These two operations are illustrated in Figure 2.

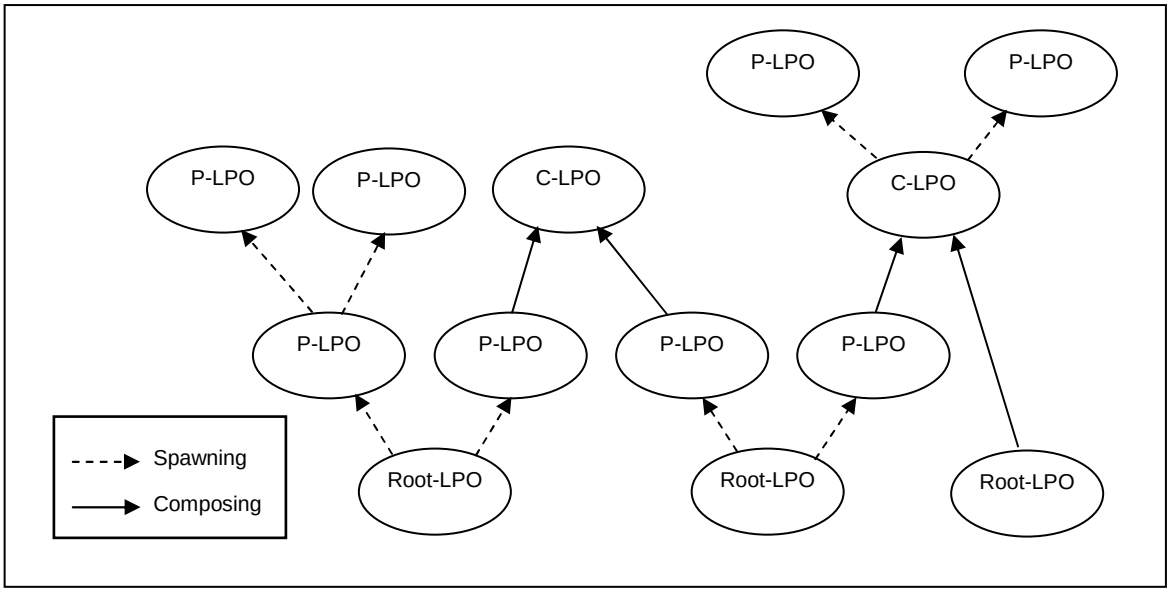

Figure 2: Spawning and composing of LPOs.

11.3.2 LPO Ownership

Each LPO has an owner stack for keeping track of its current and past owners. The top of an owner stack is the current owner or simply the LPO owner. An LPO will always have exactly one (current) owner. Only the owner of an LPO may apply control operations to the LPO. When an LPO is assigned to (or used by) a customer, the customer becomes the new owner of the LPO and a new node representing this customer will be pushed onto the owner stack of this LPO. For instance, at time T1 in Figure 3, LPO_1 is advertised by domain administrator A. Its owner stack will contain a node labeled with 'DA-A' (Domain Administrator A). At time T2, when user A forms a LP using LPO_1 and LPO_2, a new node with label 'User-A' is pushed onto the owner stack of these two LPOs. The owner stack of LPO_3 simply contains a node labeled with 'User-A', as illustrated in the same diagram.

One of the benefits of storing an owner stack structure is for terminating an LP (i.e. LPO termination). Refer back to Figure 3. Again, at time T3, some more LPOs have been created. Suppose we would like to terminate LPO_7. Recall that this operation can only be triggered by the LPO_7 owner, namely User-C. When a terminate LPO operation is issued on LPO_7, its owner stack is popped, the associated resource is released, and this operation is implicitly applied on all underlying LPOs that LPO_7 uses. The result after LPO_7 is terminated is shown in the diagram at time T4.

It is important to note that if a single variable is used to store the LPO owner, the only way to terminate LPO_1 to LPO_6 is via User-C. But if User-C is a Grid application instance and it has since finished execution after terminating LPO_7, then there will be no way to terminate LPO_1 to LPO_6. However with the owner stack, ownership is automatically transferred to the previous owner after LPO termination occurred.

It should be noted that each LPO contains an LPO list structure to store the list of LPOs used to form this LPO. This structure will be discussed in a later section. Referring back to Figure 3, at time T4, LPO_5's LPO list structure contains LPO_1 and LPO_2. Therefore it is able to determine which LPOs it is using and thus implicitly apply LPO operations on them.

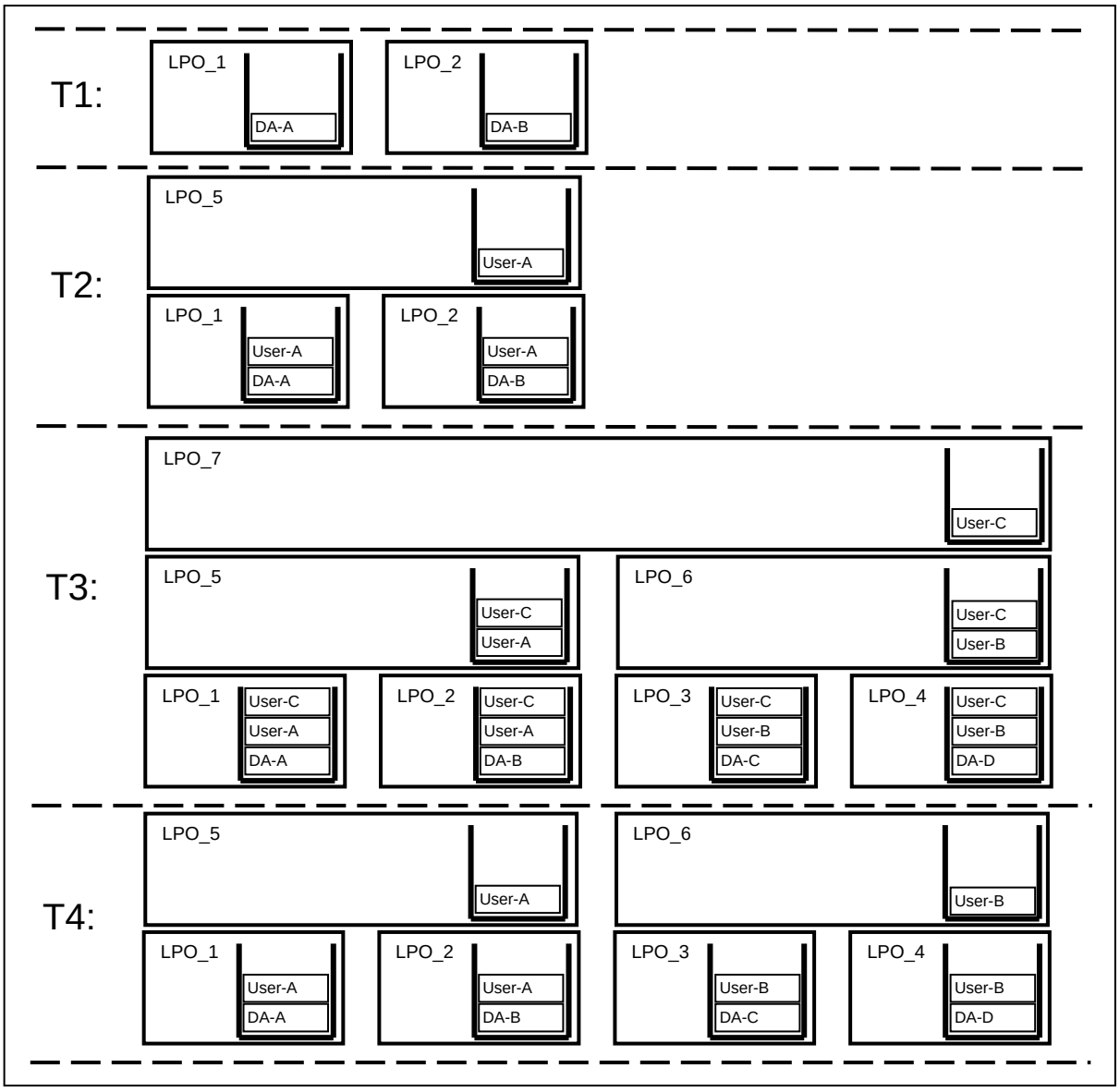

Figure 3: The owner stack of each LPO.

11.3.3 LPO Spawning

One of the key operations of this software is the ability for LPO owners (current owners) to partition and lease some portions of their LPOs to other users. The process of dividing a LP into two or more partitions is referred to as LPO spawning. It is important to note that LPO spawning is a recursive operation. That is, the owner of a LP X may grant access to a portion of its LP (called Y) to user B, who in turn leases a portion of LP Y to another user. LPO spawning may arise in two different situations. In the first case, the LP to be partitioned is an inter-domain LP. In the second and more complicated case, the LP to be spawned is an end-to-end LP which consists of two or more inter-domain LP concatenated together. Figure 4 illustrates the spawning of an inter-domain LP to three levels. Initially at time T1, 10Mbps of bandwidth is reserved for LPO_1. Then the owner of LPO_1 decides to lease a portion of it to another user, resulting in a new LPO, LPO_2, which has 5Mbps of reserved bandwidth. The

bandwidth for LPO_1 is reduced to 5Mbps as shown in T2. Later on, the owner of LPO_2 chooses to lease a portion of LPO_2 to other users. As a result, at T3, LPO_3 is created having 2Mbps of bandwidth and the remaining bandwidth available in LPO_2 is reduced to 3Mbps.

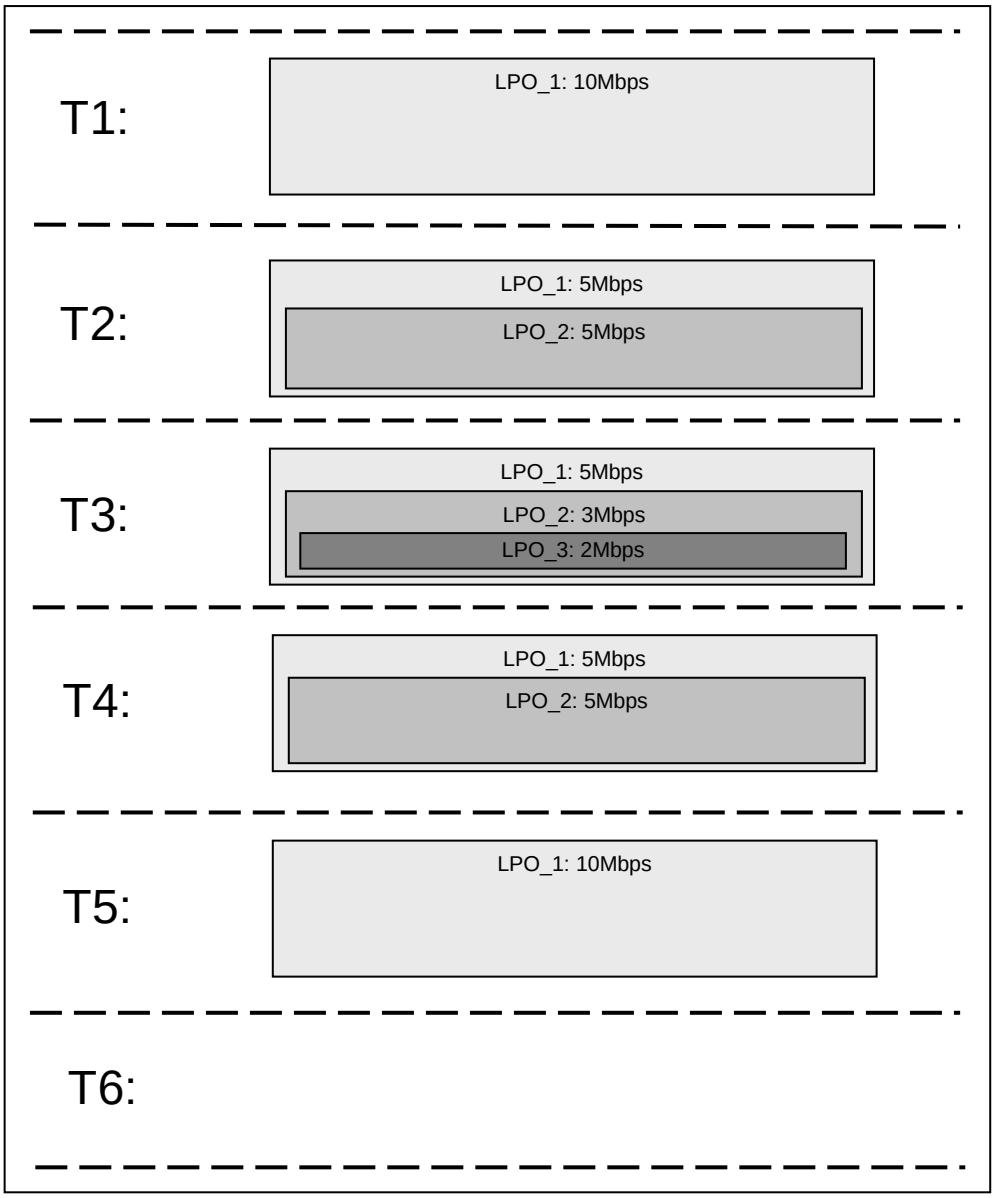

Figure 4: Spawning of an inter-domain LPO to multiple partitions.

In the case where an LPO represents an end-to-end LP as a concatenation of inter-domain LPOs, the spawning process involves partitioning the inter-domain LPOs first and then merging the partitions together to form the newly spawned end-to-end LPO. Figure 5 illustrates such situation. Initially at time T1, two inter-domain LPOs exist and each has 10Mbps of reserved bandwidth. To setup an end-to-end LP, two new LPOs, LPO_3 and LPO_4, are spawned from LPO_1 and LPO_2 respectively. The end-to-end LP LPO_5 is

formed by using these two new LPOs. Note that each LPO has an LPO list structure that stores the list of LPOs used to form an end-to-end LP. In this example, LPO_5's LPOList will contain LPO_3 and LPO_4 at time T2. To spawn another LPO from LPO_5, each LPO inside the LPOList is partitioned first to produce two new LPOs, LPO_6 and LPO_7. Then, these new LPOs are concatenated to form a new LPO, LPO_8, as illustrated in Figure 5 at time T3.

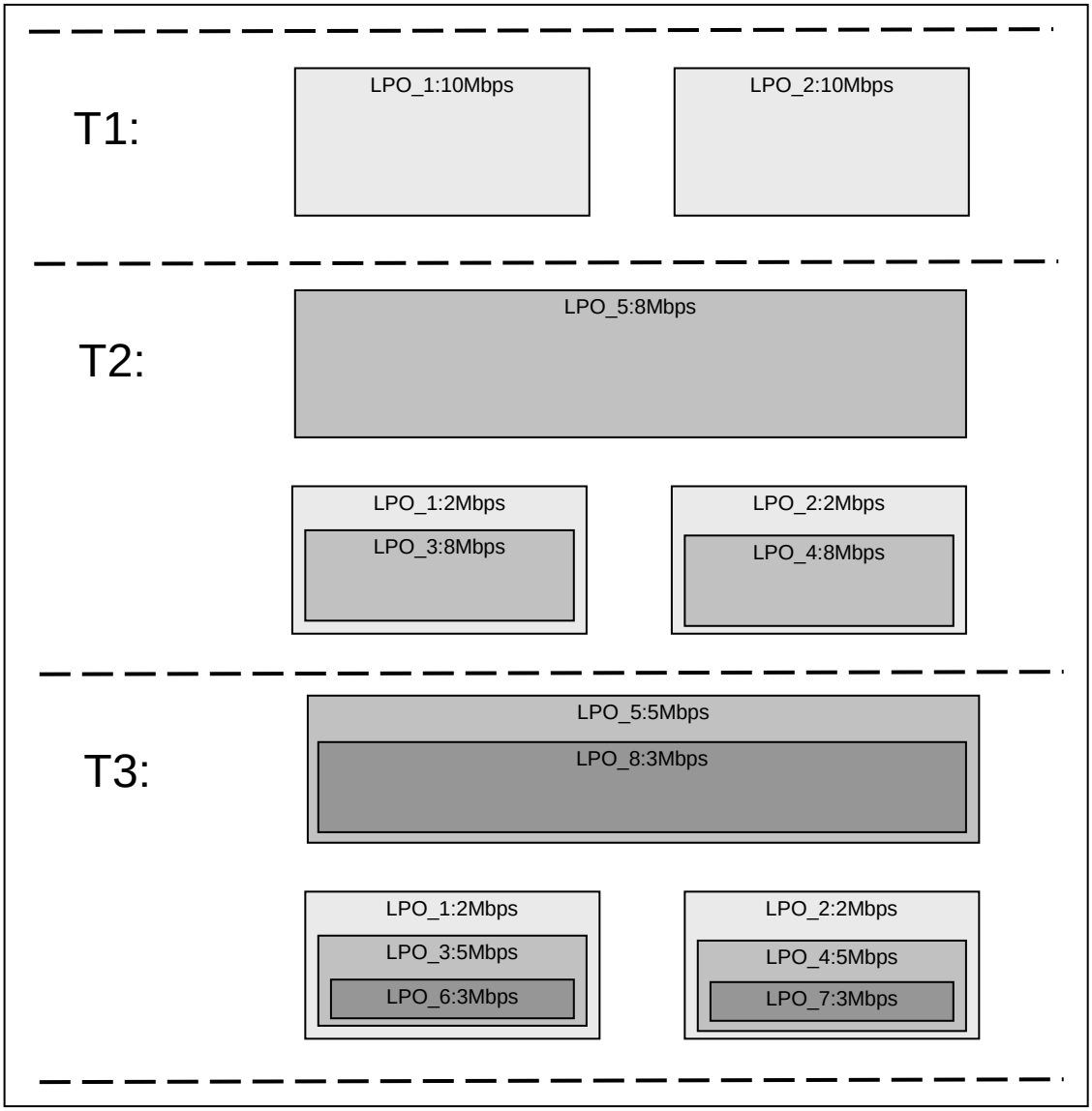

Figure 5: Setting up of an end-to-end LPO and spawning of an end-to-end LPO.

Some boundary cases for the LPO spawning operation include spawning an LPO with zero bandwidth and spawning an LPO whose bandwidth equals that of the parent LPO. Our system does not allow the first case but permits the second case. The reason is that a LPO with zero reserved bandwidth is of no practical use. But spawning an LPO with the same reserved bandwidth as that of its parent LPO is equivalent to leasing the whole LP to another user. This technique can be used to achieve LPO swapping functionality. A real life analogy to this

situation is when a landlord leases the whole house to a single tenant rather than leasing individual rooms inside the house to different tenants.

An LPO spawning operation usually involves several intermediate steps and any one of these steps may not complete successfully. This may result in a system with inconsistent state and it is an issue that will be addressed in Section 11.6.

Not all LPOs can be used for spawning. If an LPO has zero reserved bandwidth or its isSpawnable field (listed in Table 8) is set to false, then it cannot be used for spawning. Attempts to spawn from such LPOs will result in an error returned back to the user.

11.3.4 End-to-end LPO establishment

An LPO that represents an end-to-end LP (i.e. an LP that crosses multiple domains) is called an end-to-end LPO. It is established by concatenating multiple LPOs, each of which can be either an inter-domain LPO or an end-to-end LPO.

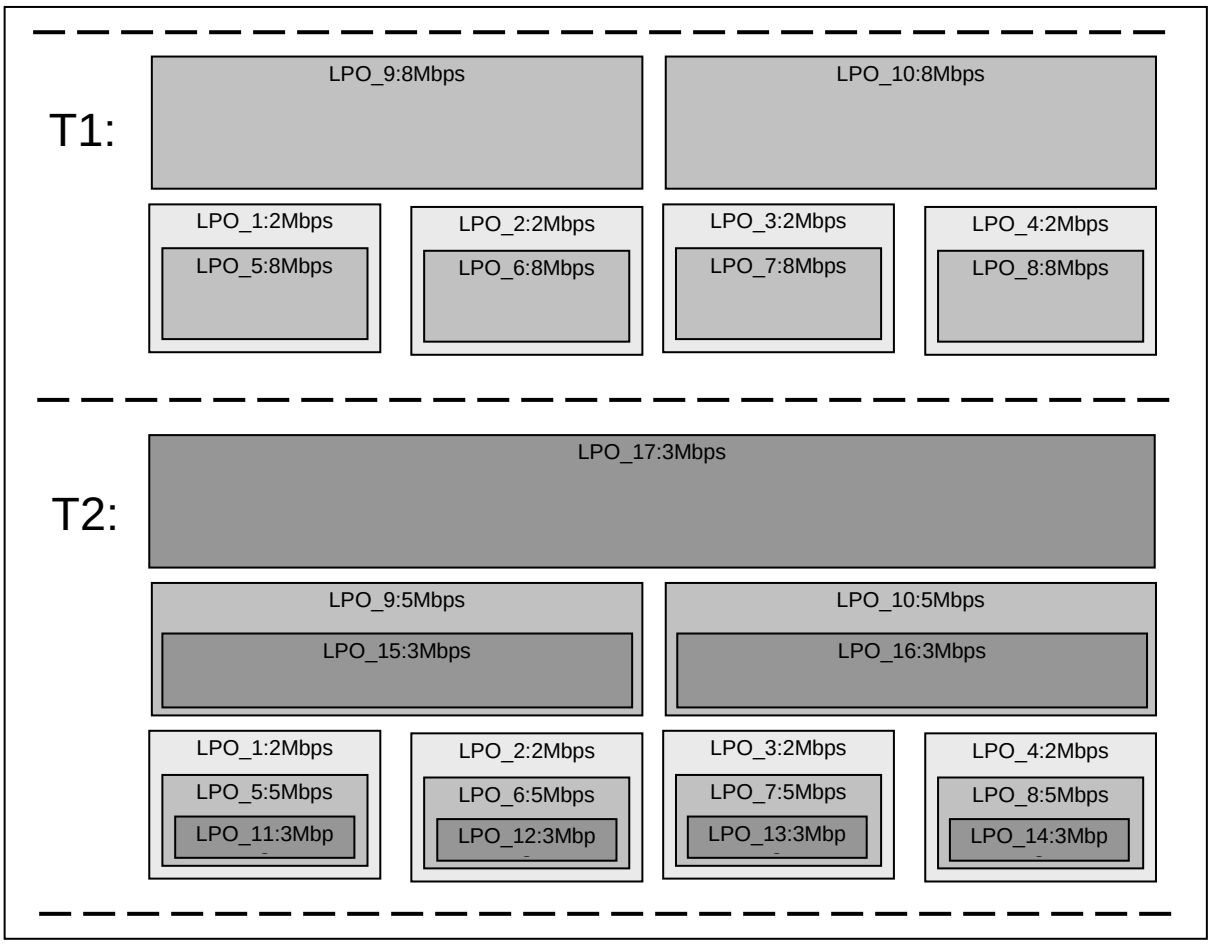

Figure 6: End-to-end LPO establishment.

An example for end-to-end LPO setup by concatenating a set of end-to-end LPOs is illustrated in Figure 6. In the diagram, two end-to-end LPs have already been created at time T1 and they

are represented by LPO_9 and LPO_10 respectively. In the diagram, LPO_1, LPO_2, LPO_3 and LPO_4 denote inter-domain LPs among four different domains. Suppose a user wishes to create an end-to-end LP across the four domains with reserved bandwidth of 3Mbps. The LP is created as follows: four LPOs each with bandwidth of 3Mbps are spawned from LPO_5, LPO_6, LPO_7 and LPO_8. Then LPO_15 and LPO_16 are spawned from the end-to-end LPs denoted by LPO_9 and LPO_10. LPO_15 contains LPO_11 and LPO_12, while LPO_16 contains LPO_13 and LPO_14. Finally at time T2, LPO_17 is created and it contains LPO_15 and LPO_16. It is important to realize that the setup of end-to-end LP always starts at the lowest level (inter-domain LPO level) and moves upwards.

11.3.5 LPO Termination

The LPO termination process entails the teardown of the LP corresponding to the LPO. Under normal operating conditions, this operation can only be carried out by the current owner of the LPO. There are some special situations where an LPO may be terminated by a past owner, they will be addressed in Section 11.6.

Similar to LPO spawning, we will discuss LPO termination under two different circumstances: inter-domain LPO termination and end-to-end LPO termination. In the first case, the LPO to be terminated represents an inter-domain LP or a partition of an inter-domain LP. Consider the situation depicted in Figure 4. Suppose the current owner of LPO_3 wishes to terminate the associated LP. The resources occupied by LPO_3 will be freed and merged back to its parent LPO, which is LPO_2. LPO_3 will also be removed as indicated in the diagram at time T4. Similarly, to remove LPO_2, the reserved bandwidth of LPO_2 is first merged back to LPO_1 as denoted in the diagram at time T5. At time T6, LPO_1 is terminated and since it is the top level LP, no resource merging occurs.

The process of end-to-end LPO termination is a little more complicated. First, each LPO inside the LPO list is terminated and its associated bandwidth is merged back to its parent LPO. Referring back to Figure 5, to terminate LPO_8, LPO_6 and LPO_7 are terminated first. Bandwidth previously occupied by LPO_6 and LPO_7 is then given back to the parent LPOs, namely LPO_3 and LPO_4. Next, LPO_8 is removed. Terminating an LPO usually involves several steps. Any one of the steps may fail for different reasons. There needs to be a way to ensure that the system state is consistent even if one of the steps failed and this issue is addressed in Section 11.6.

Not all LPOs can be terminated. An LPO can only be terminated if it has no child LPOs associated with it (i.e. its childList field is equal to null). The system verifies this property and attempts to terminate an LPO with a non-null childList will result in an error returned back to the user. In addition, a status field value (see next section) of ‘available’ must be asserted prior to LPO termination.

11.3.6 LPO Status

An LPO has a status field that indicates its availability. When an LPO’s status field is set to be ‘available’, that means a user can use its associated resource directly, or it can be concatenated with other available LPOs to form an end-to-end LP. When an LPO is being used, its status is set to be ‘reserved’, and no one can use it other than its owner. A third

possible LPO status value is ‘inactive’, which means that the LPO is unavailable and may or may not be reserved by some entity. This status value is primarily used for fault recovery mechanisms discussed in Section 11.6.

11.4 Data Structures

This section describes the data structures that will be used in the software system. One of the key data structure design principles used is to decouple LPO attributes from the LPO class. This allows classes, such as LPOtemplate, that deal with just the LPO attributes to implement just the LPO attributes interface, rather than extending the LPO class. The LPO method interface contains a set of function calls for LPO manipulation.

Data Structure	interface LPOattribute
Description	This contains all of the basic LPO attributes.
Attribute	String LPOid
	An ID that uniquely identifies this LPO. It is assigned by the LPO space when this LPO is inserted into the space.
Attribute	Int status
	The current status of the associated LP. It is an Int type that takes on values from {0=available, 1=reserved, 2=inactive} • available: this LPO is available for use • reserved: this LPO is being used by a customer. • inactive: this LPO is unavailable for any use and can only be queried
Attribute	Int prevStatus
	Whenever the ‘status’ field is modified, the current ‘status’ value is stored in this attribute. This allows the previous status to be restored after a failure that temporarily causes the status to be set to ‘inactive’.
Attribute	Stack ownerStack
	Stores the current and past owner(s) of this LPO. The top of this stack is the current owner. This stack will never be empty (i.e. there will be always an owner for a LPO).
Attribute	String parentLPOid
	If this LPO is a partition of another LPO X, then this field will be assigned with X’s LPOid field. Otherwise this field is set to null.
Attribute	String sourceSWTID
	A unique ID (such as the IP address) that identifies the upstream switch of the associated LP.
Attribute	String destSWTID
	A unique ID (such as the IP address) that identifies the downstream switch of the associated LP.
Attribute	Boolean isSpawnable
	True if and only if the associated LP can be further spawned from. One situation where this occurs is when the reservedBandwidth of the LPO is zero. Another scenario is when an LP can no longer be

	further subdivided into smaller bandwidth channels due to bandwidth granularity constraints.
Attribute	Long bandwidth
	The amount of bandwidth in Kbps reserved for the associated LP.
Attribute	Date expireDate
	The time when this LPO will expire. When an LPO expires, its terminateLP() method is invoked. This field is used to store the lease time of an LP.
Attribute	List childList
	A list of child LPOs that were spawned from this LPO.
Attribute	List LPOlist
	If this is an end-to-end LPO, then this field contains a list of LPOs used to form the associated LP. If this is an inter-domain LPO, then this field is set to null.
Attribute	String loopbackLPO
	The LPO ID of the loopback LP for the associated LP, if one exists.
Data Structure	class LPOclass implements LPOattribute, net.jini.core.entry.Entry
Description	The LPOclass implements a set of attributes and methods for LPO object manipulation purposes. The LPOclass implements the Entry interface, which is required in order for LPOs to be inserted and used with a JavaSpace. Each instance of the LPOclass is an LPO object. Each LPO object (or just LPO) is assigned a unique ID when inserted into the LPO space.
Method	LPOclass spawnLPO(long bw)
	Spawn and return a new LPO with reserved bandwidth equal to 'bw' from this LPO. If this LPO is an inter-domain LPO, then the source and destination switches of this LPO are contacted to have a new LP setup with bandwidth equals to 'bw'. If this is an end-to-end LPO, then spawnLPO() is invoked on each LPO in LPOlist. Detailed information on LPO spawning can be found in Section 11.3.3. If the operation is unsuccessful, null is returned.
Method	boolean terminateLP()
	Return false if this LPO is the parent of some existing LPO. Otherwise release the reserved bandwidth back to the parent LPO and then remove this LPO from the system. If this is an end-to-end LPO, then invoke this method on each LPO in the LPOlist. If this LPO has no parent, then set the status of this LPO to 'available' and insert it back into the LPO Space. Return true if the operation is successful; return false otherwise.
Method	void pushOwnerStack(String ownerID)
	Push an owner Id onto the top of the owner stack.
Method	String popOwnerStack()
	Pop and return the top owner Id of the owner stack.

Data Structure	class LPOtemplate implements LPOattribute
Description	An LPO template is a placeholder for LPO attributes. It contains all of the attributes of an LPO. It is useful for passing LPO attributes into function calls as well as for containing information on how to get access to an LPO (i.e. by a Grid application).
Data Structure	class LPOSpace implements net.jini.space.JavaSpace
Description	This class represents a storage space for LPO and LPOref objects. It is inherited from JavaSpace class, which enables persistent storage of objects and provides a set of functions to add, remove and find objects inside the space. This class has methods that automatically remove expired objects from the space. An LPO space may contain other LPO spaces. This is an important property of this class because it enables a logically centralized but physically distributed LPO space to be implemented.
Attribute	String LPOSpaceId
	A string that uniquely identifies this LPO Space.
Attribute	Date startTime
	Since when is this space running.
Attribute	Date curTime
	The current time of the LPO space.
Attribute	ExpireLPOThread expireLPOCleaner
	This is a background thread that periodically invokes the removeExpireLPO method.
Attribute	CheckLPOThread checkLPOworker
	This is a background thread that periodically checks this LPO space to ensure that the source and destination switches of a pre-defined set of LPOs are up.
Attribute	String parentLPOSpaceId
	The identification of the parent LPO space of this LPO space. If this LPO space has no parent, then this field is set to null.
Attribute	Set LPOSpaceSet
	The set of LPO spaces contained by this LPO space. A call to read an LPO from this space will also invoke the read method of the LPO spaces in this set.
Method	List removeExpiredLPO()
	Find and terminate all expired LPOs (by invoking the terminateLP() method on them). This method is invoked by the expireLPOCleaner periodically. Return the list of removed LPOs.
Method	String addLPO(LPOclass lpo)
	Add the specified LPO ‘lpo’ to this LPO space and generate and return a unique string identifying the specified LPO.
Method	boolean addLPOSpace(LPOSpace lpoSp)

	Add the specified LPO space to this LPO space. 'lpoSp' must not already have a parent LPO space. Return true if the operation is successful; return false otherwise.
Data Structure	class LPOThread implements Runnable
Description	Defines the superclass of a set of useful background thread objects that periodically check the LPO space for various purposes.
Attribute	long timePeriod
	How often this thread is executed, measured in seconds.
Data Structure	class ExpireLPOThread extends LPOThread
Description	This is a thread for finding and removing expired LPOs by calling the removeExpiredLPO() method every 'timePeriod' seconds.
Data Structure	class CheckLPOThread extends LPOThread
Description	This is a thread for ensuring that the source and destination switch of a pre-defined set of LPOs in the LPO space are up. One way to use this thread is to check all 'available' LPOs by providing the appropriate set definition. The definition is passed into the constructor of this thread as an LPO template. If an LPO is an end-to-end LPO, then all LPOs in the 'LPOlist' will also be examined. If a switch cannot be contacted, the corresponding LPO is set to 'inactive'. In addition to checking the pre-defined set of LPOs, this thread will also automatically examine LPOs with status equal to 'inactive' to see if the associated switch is up or not. If it is up, then the LPO will be assigned to its previous status.

Table 8: LPO attributes, data structures, and methods.

11.5 Detailed System Architecture

The detailed system architecture for the proposed software system is depicted in Figure 7. The system is divided into three main layers: the user access layer, the service provisioning layer and the resource management layer. The user access layer is concerned with handling and parsing requests from customers and domain administrators. The service provisioning layer is about providing Grid services to users as well as applications. The resource management layer deals with LPO maintenance and low-level communication with switches of the underlying optical network. In subsequent sections, we will detail the purpose and functionalities of each component at each layer.

The user access layer and service provisioning layer are logically centralized, whereas the resource management layer is both physically and logically distributed.

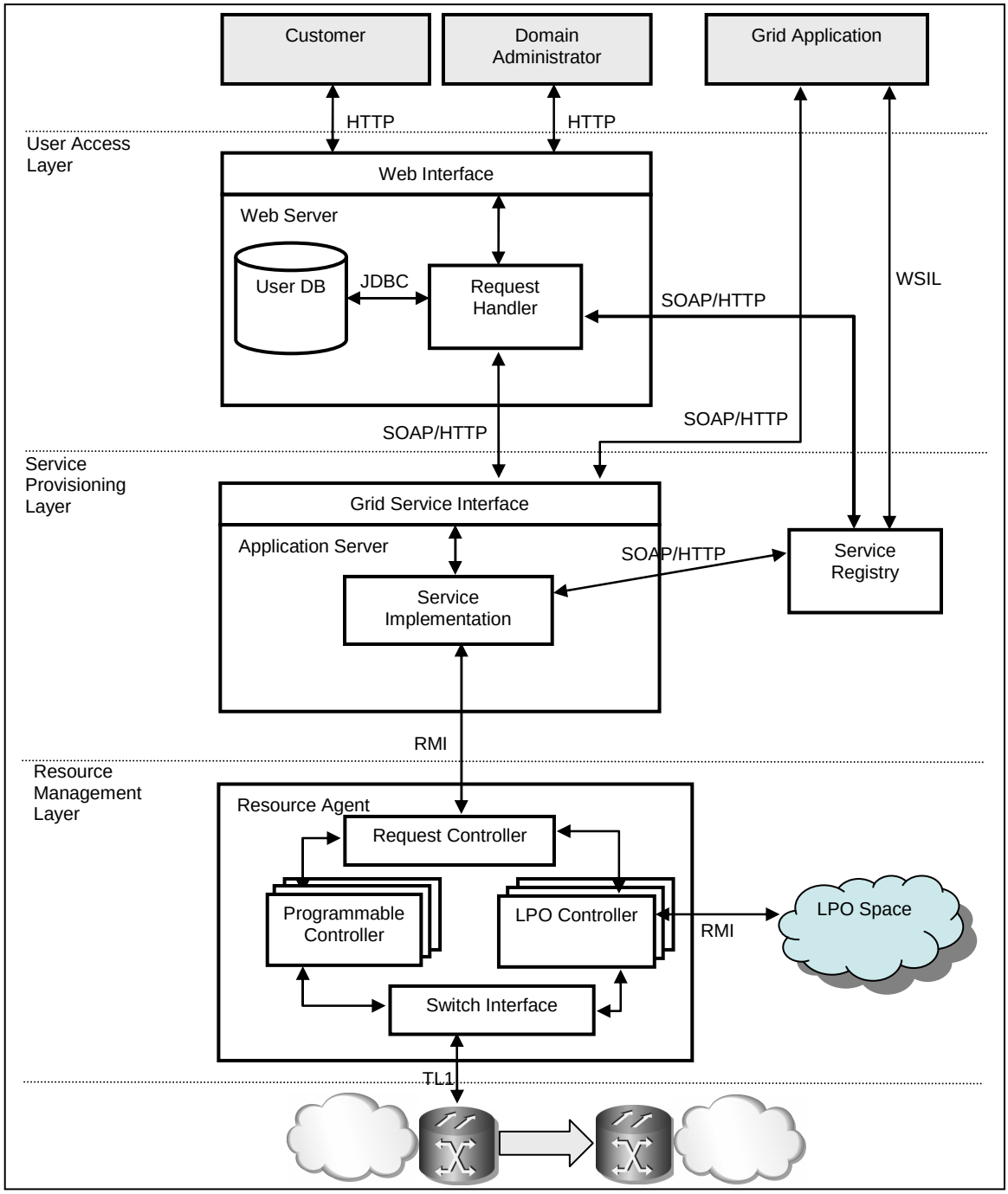

Figure 7: Detailed system architecture.

11.5.1 User Access Layer: Web Server

The Web server receives and handles customer or domain administrator requests through a Web interface. The Web interface allows domain administrators to add, remove, query and modify LPOs. Any LPO advertised by the domain administrator must represent an established inter-domain LP. The Web interface permits customers to create end-to-end LPs across

multiple domains or partition an existing LP and re-advertise the partitions. When a user request is received by the Web server, the request handler first ensures that the request is from an authorized party and then parses this request. If the request is an LPO operation, then the request handler will attempt to find an appropriate service for the requested operation by querying the service registry. After a suitable service is located, the request handler accesses the service by creating and exchanging SOAP/HTTP messages with the application server.

The Web server is connected to a database that contains the set of accounts of users who may access the system. User accounts may be added, removed, and modified by the Web server administrator (a different entity from a domain administrator). However, the Web server administrator does not have access to the LPO services, and therefore it is not shown in Figure 7. We plan to use mySQL as the database server for user accounts since it is free for non-commercial use and it is a widely used open-source database management system.

The Java Web Services Developer Pack (Java WSDP) provides a set of tools for building and deploying XML applications, Web services, and Web applications [4]. Java WSDP is from Sun and is available at no cost. One of the tools from the Java WSDP toolset is the Apache Tomcat container, which is a suitable server solution for the Web server implementation of this project. Tomcat is a commercial-quality server that is used for the official reference implementation for Java Servlet and JavaServer Pages technologies. The request handler will be implemented as a set of Java Servlets. A Grid service may not complete instantaneously and a mechanism is needed for returning results to the user at a later time. Tomcat integrates well with Apache HTTP server, which is the de facto Web server standard in the market today. Another tool provided by Java WSDP is the Java API for XML Messaging (JAXM). It is a package for building SOAP conforming messages, which is needed by the request handler for sending and receiving SOAP messages.

11.5.2 Service Provisioning Layer: Service Registry

The service registry maintains a list of available Grid services. The request handler inside the Web server and a Grid application may query this registry directly via SOAP/HTTP and WSIL, respectively. Each Grid service is described in the standard WSDL format. The service registry is essentially an XML registry and there are two complementary standard specifications for it including the ebXML registry and repository standard and the UDDI specification. For this project, we will use xbxmlrr for the service registry because it is a stable and free reference implementation for ebXML [6].

11.5.3 Service Provisioning Layer: Application server

The application server implements all LPO-related services and uses an OGSA conforming interface so that Grid applications can access it directly. The Grid service interface inherits from portType interfaces, which are the standard interfaces for Grid applications. The portType interfaces that the Grid service interface inherits from are listed in Table 9.

PortType Name	**Description**
GridService	Encapsulates the root behavior of the component model.
HandleResolver	Mapping from a Grid service handle to a Grid service reference.

NotificationSource	Allows clients to subscribe to notification messages.
NotificationSubscription	Defines the relationship between a single NotificationSource and NotificationSink pair.
NotificationSink	Defines a single operation for delivering a notification message to the service instance that implements the operation.
Factory	Standard operation for creation of Grid service instances.
Registration	Allows clients to register and unregister registry contents.

Table 9: PortType interfaces (source [3]).

The application server contains the implementation of a set of Grid services. When the application server starts, it publishes this set of services onto the service registry via SOAP/HTTP. The list of services to be implemented inside the application server is described in Table 10. Both the request handler of the Web server and the application server need to construct and exchange SOAP messages for accessing and communicating with the registry for various reasons. For implementation purpose, we can use the Java API for XML registries (JAXR) for SOAP message composition. JAXR is another tool provided by the Java WSDP and it can access registries that follow either UDDI or ebXML specification. For the actual service implementation, Java is the programming language of choice. JBoss is chosen for use as the application server. JBoss is written in 100% pure Java and it is a widely used and popular application server solution. The application server communicates directly with resource agents using RMI.

Service Name	**Input Parameters**	**Return Value**	**Description**
LPO advertisement	An LPO template	LPO ID	Advertises the availability of an LP by adding the corresponding LPO to the LPO space. Return a unique ID that identifies the newly created LPO.
LPO termination	LPO ID	Boolean	Finds the specified LPO in the LPO space and invokes terminateLP() on it. This terminateLP() method is also invoked on the loopback LPO of the specified LPO, if one exists. Return true if operation is successful; return false otherwise.
LPO query	An LPO template	A list of LPOs that match the specified search criteria	Retrieve and return a list of LPO(s) that match the attributes specified by the input LPO template.

ETE LP establishment	sourceSWTID, destSWTID, bandwidth, isLoopback	LPO ID	Setup and establish an end-to-end LP. This is accomplished by finding a set of useful LPOs from the LPO space, creating a new LPO to contain this set of LPOs and adding this new LPO to the LPO space. The 'isLoopback' argument indicates whether or not to create a loopback LP for this LP. If yes then an LPO representing a loopback LP is created and stored in the 'loopbackLPO' field of the new LPO. The loopback LPO should have reserved bandwidth equal to 'bandwidth'. If 'isLoopback' is set to true and a loopback LP cannot be found, the whole ETE LP setup operation is considered unsuccessful. Note that a special case for this service is to setup an LP between two neighbouring domains. In this case if an LPO that meets the bandwidth requirement exists, its ID will be returned. Otherwise, a new LPO ID will be returned.
LPO reconfiguration	LPO ID, a LPO template	Boolean	Modify the specified LPO's attributes. For example, increase or reduce the reserved bandwidth for the associated LP.
LPO spawning	LPO ID, a LPO template	An LPO ID	Create a list of new LPOs each using a partition of the specified LPO's resources. Add these new LPOs to the LPO space, and modify the specified LPO's resource attributes accordingly. A special case of this operation is to spawn a child LPO that has the same bandwidth requirement as the parent LPO. In this case, the parent LPO's bandwidth reservation will be reduced to zero and the child LPO will be instantiated. If the LPO indicated by 'LPO ID'

			contains a loopback LPO, then a partition of the loopback LPO will also be spawned in the same manner as described in the previous paragraph.
LPO concatenation	A list of LPO ID	LPO ID	Given a set of LPOs that have the same owner, concatenate them together to form an end-to-end LPO. Return the ID of the newly created LPO.
LPO access	LPO ID	An LPO ref.	This allows the caller to gain access and directly use the specified LPO. Return a handle (in the form of LPOref) to the caller so that it can use the LPO directly. This service will mostly be used by a Grid application to directly use the LP associated by a LPO.

Table 10: List of Grid services to be implemented.

Figure 6 illustrates a simple scenario where there exists only one route from one domain to another. In reality there could be multiple paths between the same pairs of non-neighboring domains. The application server maintains the topology of the whole optical network by keeping track of the inter-domain LPOs being advertised. An LPO is advertised by inserting the LPO with status set to 'available' into one of the LPO spaces corresponding to the relevant switch (note: this is explained further in Section 11.5.5). The application server receives LPO advertisement notification messages from the resource agent and modifies its view of the overall network topology accordingly. For instance, if the LPO advertisements listed in Table 11 are made, then the overall network topology view stored inside the application server is depicted in Figure 8. The internal view of the overall network topology is used by services such as end-to-end LP establishment. Using this structure, a route discovery mechanism determines a feasible set of LPOs for constructing an end-to-end LPO. It is possible that no such LPO set can be found, and in this case an error is returned back to the user.

Advertised by	**Advertisement**	**Bandwidth**	**Associated LPO**
Domain Administrator of domain A	Domain A to Domain B	OC-48	LPO_1
Domain Administrator of domain B	Domain B to Domain C	OC-48	LPO_2
Domain Administrator of domain B	Domain B to Domain D	OC-48	LPO_3
Domain Administrator of domain C	Domain C to Domain A	OC-48	LPO_4

Table 11: Example advertisement messages from domain administrator.

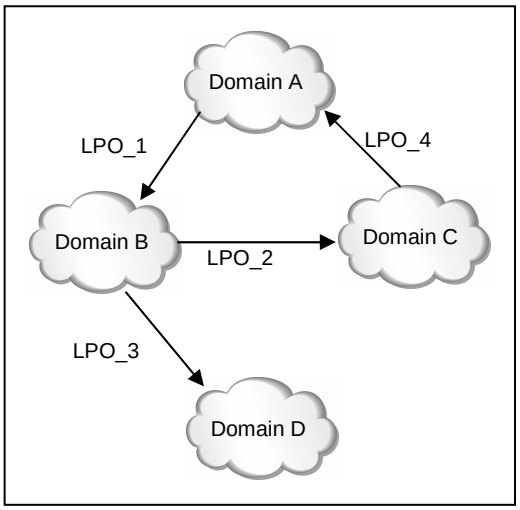

Figure 8: Internal view of the overall network topology.

11.5.4 Resource Management Layer: Resource Agent

The resource agent manages and controls all the LPs originating from a switch. There is one-to-one mapping between a resource agent and a switch. When a domain administrator advertises the availability of an inter-domain LP, a resource agent instance is created for the upstream switch (if one such resource agent does not already exist). Note that although the resource agent is associated with the upstream switch, under some situations, it will also contact the downstream switch. One such situation is for constructing an LP with loopback. A resource agent maintains an LPO space, which contains LPOs representing LPs (and possibly partitions of them) with the source switch being the switch associated with this resource agent. A sample situation is illustrated in Figure 9, where a resource agent manages switch X and maintains an LPO space containing LPOs corresponding to all the inter-domain LPs that originate from X. Note that when an LPO is added to the switch, a notification message that contains the LPO information is sent to the application server. The application server will then update its internal view of the overall network according to the message content.

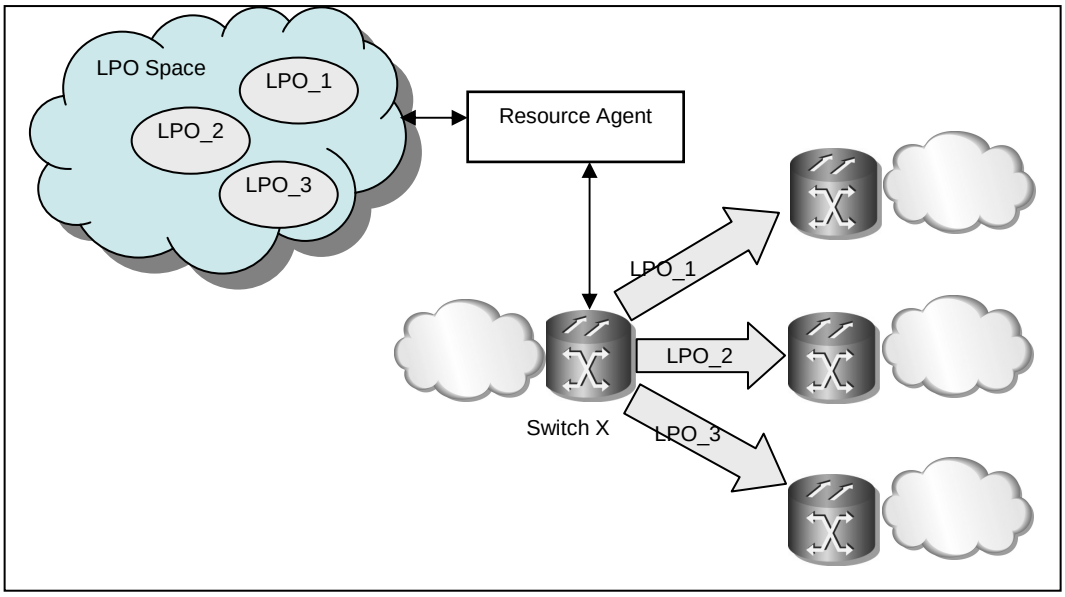

Figure 9: A resource agent instance managing three inter-domain LPOs.

The resource agent is composed of four building blocks: request controller, LPO controller, programmable controller, and switch interface. The request controller receives and handles RMI calls from the service implementation of the application server. The LPO controller is associated with a customer and it manages all the LPOs assigned to that customer. LPO-related operations such as LPO spawning and LPO termination are done through the LPO controller. For instance, to spawn an LPO, the request handler will create an LPO controller instance for the customer (if one does not already exist for this customer) and grab the LPO from the LPO space. Then this LPO controller instance will invoke the spawnLPO() method on the LPO and this operation will configure the switch through the switch interface component. Next, the LPO controller will insert both the newly spawned LPO and the parent LPO to the LPO space.

Another building block of the resource agent is the programmable controller, which is also instantiated on a per-customer basis. It provides an environment for customers to install and execute scripts. When a single customer owns a partition of LPs between a set of neighboring domains, then the set of LP partitions effectively forms a virtual networking environment for that customer. With the programmable controller, the customer may operate customized routing and/or signalling protocols in his/her virtual environment, for example to control peering with other customers.

The fourth component of the resource agent is the switch interface. It provides a facility to construct, send and receive commands to and from the involved switches. On the resource agent side, the switch interface offers a hardware-independent API to the programmable controller and the LPO controller. A middle layer encapsulates hardware compatibility logic in the form of a set of hardware adapter modules. On the hardware side, the switch interface contains a protocol layer supporting industry standard protocols such as TL1 or SNMP. AdventNet's Java TL1 API can be used here.

As mentioned earlier, the LPO class contains a set of operations for self-manipulation. In order for the changes made to an LPO to be reflected on the physical hardware, commands need to be issued to the associated switch. This is achieved by binding the LPO instance onto the switch interface so that TL1 or SNMP commands can be sent through the switch interface.

11.5.5 Resource Management Layer: LPO Space

The LPO space is an entity that provides persistent object storage and object manipulation functions. It is always associated with a resource agent and as a result it maintains the set of available, reserved and inactive LPOs local to an upstream switch. It will be implemented using the JavaSpaces API, which already provides a set of useful functions such as object lookup based on a set of attribute values, as well as addition and removal of objects. An LPO may optionally have a lifetime. A background thread will periodically check the LPO space and terminate expired LPOs. Note that Jini's lease management function can be used instead and it will be an implementation choice. In order to modify or manipulate an LPO in the space, it has to be logically taken out from the space. This operation hides the LPO from other users and implicitly disallows multiple users to modify the same object simultaneously.

11.6 Fault Recovery

Fault recovery refers to recovering the system from unexpected or exceptional circumstances. Many different types of errors may occur for various reasons such as hardware failure and equipment misconfiguration. Some failures are recoverable and some are not. We can only deal with errors that are recoverable, such as the ones that will be mentioned in the following paragraph.

One exceptional condition that may often occur is that the system cannot contact some network switches for a period of time. Under such circumstance, all the 'available' LPOs linked to the unreachable switches should be temporarily made inactive from the LPO space until the switches are reachable again. This is because if an LPO is not reachable by the system, that means we cannot configure it or use it. Therefore such LPOs should be disabled so that end-to-end LP setup process will not consider them. The solution to this situation is to initiate a background thread on the system LPO space that periodically checks 'available' LPOs to ensure that their source and destination switches are accessible. When a switch is not reachable, all LPOs related to this switch are temporarily set to an 'inactive' state. Note again that this checking is performed only on LPOs with status equal to 'available'. The reason is that if a switch along an end-to-end LP is unreachable to the system, it does not mean that the LP is unavailable to the user. But if a switch associated with an 'available' LPO is unreachable, then the system can not configure it when someone needs to use it. Hence there is no point in keeping this LPO available in the LPO space.

Similar to most mission-critical distributed applications, we need to ensure that the system is in a consistent state after each operation. For example, when setting up an end-to-end LP, one of the steps is to issue appropriate TL1 commands to the set of involved switches. Acknowledgement messages are returned back from the switches to the system to indicate the success or failure of the requested action. If the system receives failure messages from one of the switches, the whole LP setup is considered as unsuccessful. In this case, any actions

issued to the other switches need to be roll back. In our system, any operation that requires communication with remote entities is logged step by step. The log denotes at what time a particular action is taken. For an end-to-end LP setup operation, one section of the log will contain the set of TL1 commands issued, the target switches of the commands and the time at which the commands were sent. The log also contains information on when an acknowledgement message was received and from which switch. Thus, when one switch fails to send back an acknowledgement message, this log will be examined to perform a rollback operation.

11.7 Sample Scenarios

11.7.1 LP Advertisement by a Domain Administrator

The administrator of domain A wishes to advertise an available LP from domain A to domain B. This operation is illustrated in Figure 10.

1. A domain administrator (a human operator) of domain A attempts to login to the system. The system checks the entered user name and password pair by verifying it against the user DB.
2. The domain administrator is now logged to the system. He/she enters the information for the LP to be advertised and selects the LPO advertisement option on the Web page.
3. The request handler on the Web server receives this request; contacts the service registry for this LP operation.
4. The service registry returns the location and the method for accessing the LPO advertisement service.
5. The request handler contacts the application server through the Grid service interface and passes the LP information to the application server. It should be understood that inside the service implementation, the LPO service factory creates a service session to handle the LPO advertisement request. These are not shown in the diagram because the request handler will not issue keep-alive message to maintain the session for later use. Each time a service is accessed by the request handler, a new service session is created. The next scenario describes this stage in more detail.
6. The service implementation (i.e. the corresponding service session) passes the LPO advertisement request to the appropriate resource agent.
7. The resource agent creates an LPO controller for this user (if one does not already exist) and the controller creates a new LPO according to the specified attributes and inserts it to the LPO space.

8. The result of the operation is propagated from the resource agent to the application server, and then back to the user via the Web server.

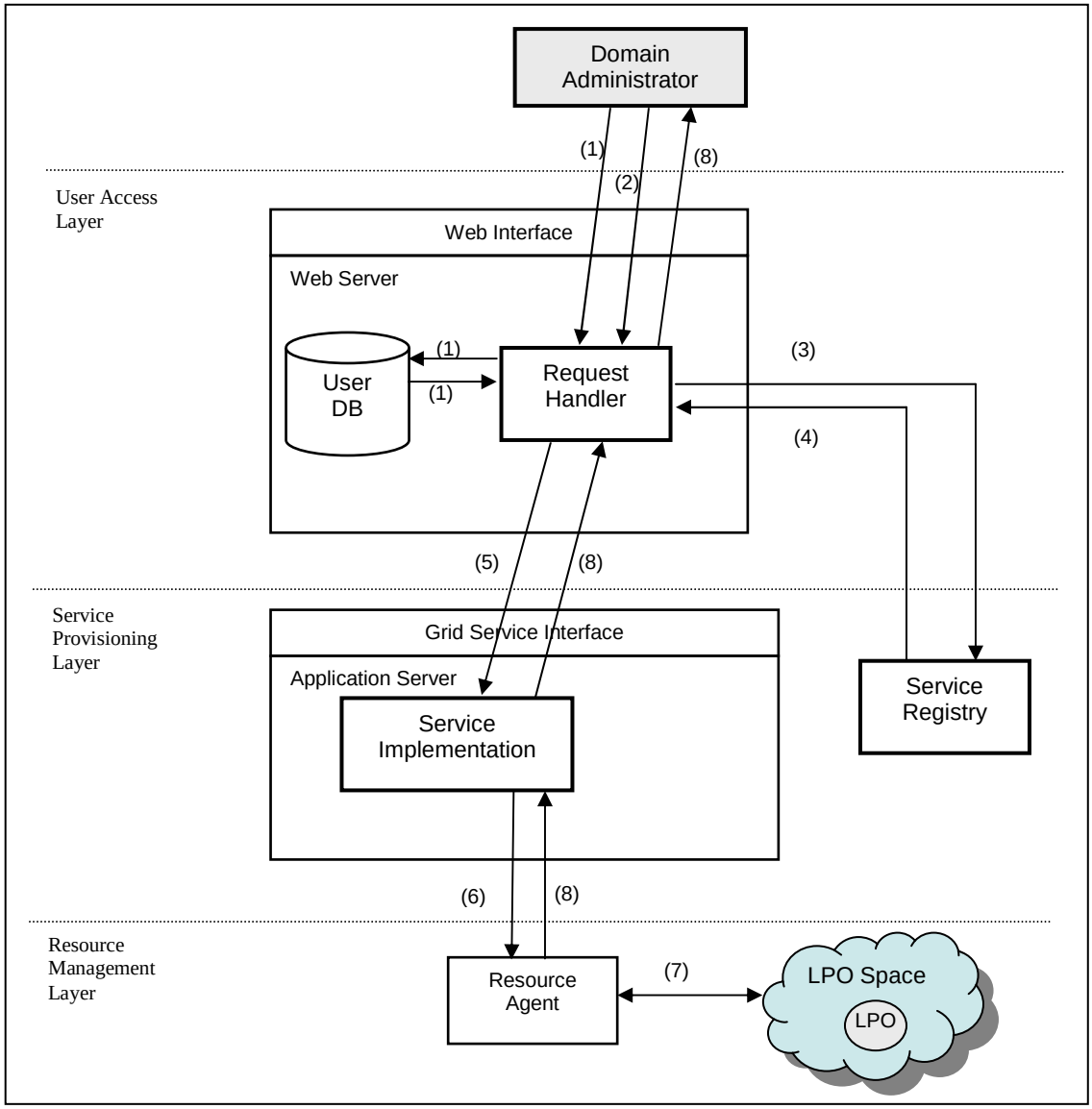

Figure 10: LPO Advertisement by a domain administrator.

11.7.2 Setup end-to-end LP for a Grid Application

GridFTP, a Grid application, needs to setup an end-to-end LP for data transfer. The steps involved for setting up a LP for this application are shown in Figure 11.

1. GridFTP contacts the service registry to find the ETE LP establishment service.
2. The service registry returns a handle of the ETE LP establishment service via WSIL. The handle contains information on where and how to access the service.

3. GridFTP contacts the application server to use the ETE LP establishment service via SOAP/HTTP.
4. The LPO service factory creates a service session instance for providing ETE LP establishment service for GridFTP.
5. A handle of this service session is returned to the caller.
6. GridFTP passes the required parameters including source switch ID, destination switch ID and bandwidth requirement to the service session via SOAP/HTTP. GridFTP also periodically sends keep-alive messages to ensure that the service session exists and can be used as long as the GridFTP wants.
7. The service session contacts the route discovery mechanism to find an appropriate set of LPOs that can be used to setup a LP with the requested attribute (such as desired bandwidth). If one exists, go to the next step. Otherwise, return an error message to the GridFTP and skip the rest of the steps.
8. The service session contacts the relevant resource agents via RMI calls and passes the needed information (such as the ID of the LPO to use) for setting up a LP.
9. The resource agent grabs the requested LPO from the LPO space.
10. The resource agent configures its associated switch by issuing TL1 commands to it. It also configures the downstream switch by contacting the resource agent associated with that switch. In Figure 11, suppose that the GridFTP source resides in a host inside domain A and the destination host (the receiver of GridFTP transfer) is inside domain C. The switch configuration part includes several tasks that happens simultaneously as follows:
    a. A route is assumed to have been setup within domain A so that GridFTP can direct its traffic to switch E.
    b. An LP is setup and switch E is configured (by LPO service session) to forward GridFTP traffic along a pre-established LP to switch F.
    c. An intra domain path is setup within domain B so that traffic can be transmitted from switch F to switch G.
    d. An LP is setup and switch G is configured to forward GridFTP traffic along it.
    e. An intra domain path is setup inside domain C so that traffic can be directed from switch H to the destination host inside domain C.

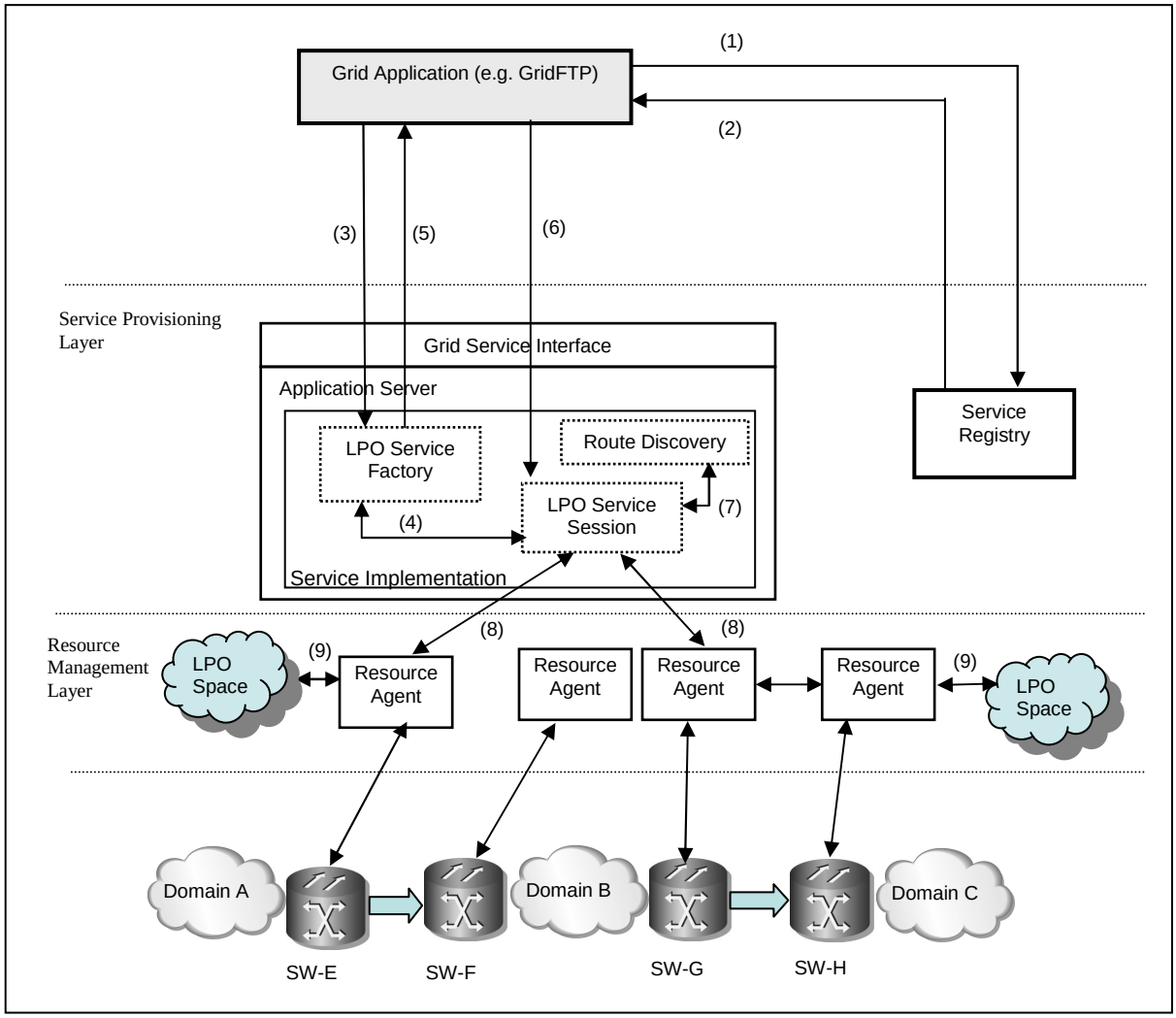

Figure 11: GridFTP operation.

11.7.3 Setup end-to-end LP by a customer through the Web interface

The setup of an end-to-end LP by a customer is very similar to that by a Grid application, which is described in Section 11.7.2. The main difference between the two is the addition of the Web interface layer in the former case. The scenario is illustrated in Figure 12 and the steps involved are as follows:

1. A customer (human user) enters his user name and password on the system login Web page. The Web server receives the user name and password pair and verifies it against the user DB.
2. The customer is authenticated and is now logged into the system. He/she chooses the end-to-end LP setup option and enters the required information including the source and destination of the LP, and how much reserved bandwidth is needed for the LP.

3. The request handler checks the service registry for the requested operation via SOAP/HTTP.
4. The service registry returns a handle of the end-to-end LP establishment service via SOAP/HTTP. The handle includes where the service is located and how to access it.
5. The request handler contacts the application to access the desired service via SOAP/HTTP.
6. As in the last example, the LPO service factory creates an end-to-end LP establishment service session.
7. A handle of this service session is returned to the caller.
8. The service session decodes the SOAP message to extract the service parameters.
9. The service session attempts to find a feasible end-to-end path using the route discovery mechanism. If unsuccessful, return an error back to the user and skip the rest of the steps.
10. A feasible path has been computed. The service session will contact the relevant resource agents and pass on the message parameters.
11. The resource agent receives the path setup request, goes into the LPO space to grab the appropriate LPO.
12. The resource agent configures its associated switch by issuing TL1 commands to them. It also configures the downstream switch by contacting the resource agent associated with that switch.
13. The operation result is propagated back to the application, then to the user in the form of a Web page via the Web server.

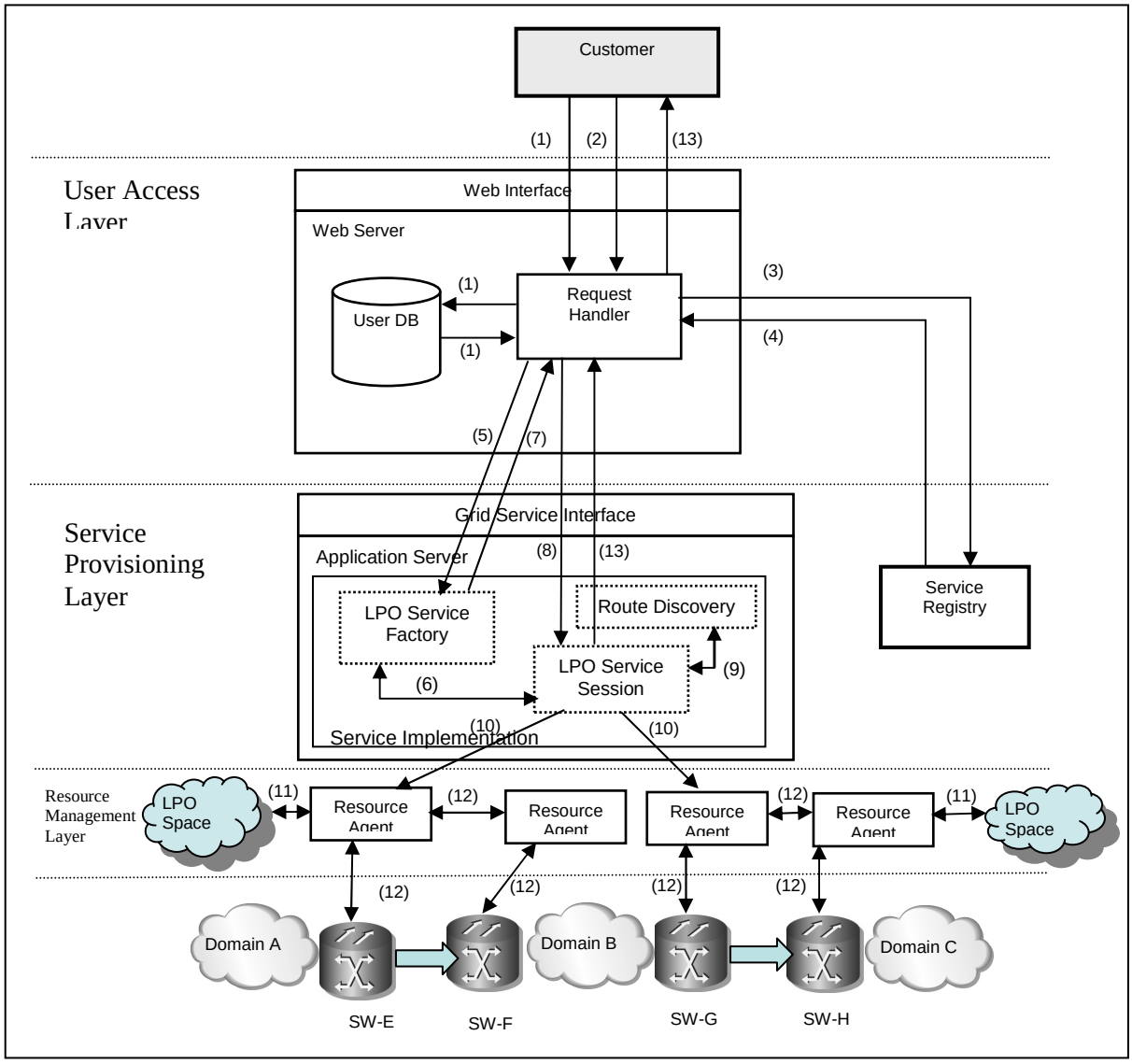

Figure 12: End-to-end LP setup through Web interface.

11.8 References

[1]	User Controlled LightPaths Definition Document Version 6.0, Nov. 22, 2002.
[2]	R. Boutaba, W. Ng and A. Leon-Garcia. "Web-based Customer Management of VPNs". Journal of Network and Systems Management, Vol. 9, No. 1, 2001.
[3]	S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Grahma, C. Kesselman and P. Vanderbilt. Grid Service Specification. Draft-ggf-ogsi-Gridservice-04, work-in-progress, Oct. 2002.
[4]	Java Web Services Developer Pack (Java DWSP), http://java.sun.com/Webservices/downloads/Webservicespack.html
[5]	XMLSpy, http://www.xmlspy.com
[6]	S. V. Kartalopoulos. "Understaing SONET/SDH and ATM". IEEE Press, pp.43.
[7]	ebXML, http://www.ebxml.org/

12 Appendix: Draft XML Schema and WSDL Descriptors

12.1 Introduction

We provide sample WSDL descriptors for some of the services described in the previous section including LPO advertisement, LPO spawning and ETE LP establishment. The service descriptors are created using XMLSpy [5]. XMLSpy enables users to create and edit any XML documents (such as WSDL documents, SOAP envelopes, and XML schemas) via a human friendly interface and it also validates XML documents against W3C standard conformance. The sample WSDL documents are generated by this tool and can be found in subsequent subsections. Figure 13 shows a sample WSDL descriptor inside XMLSpy.

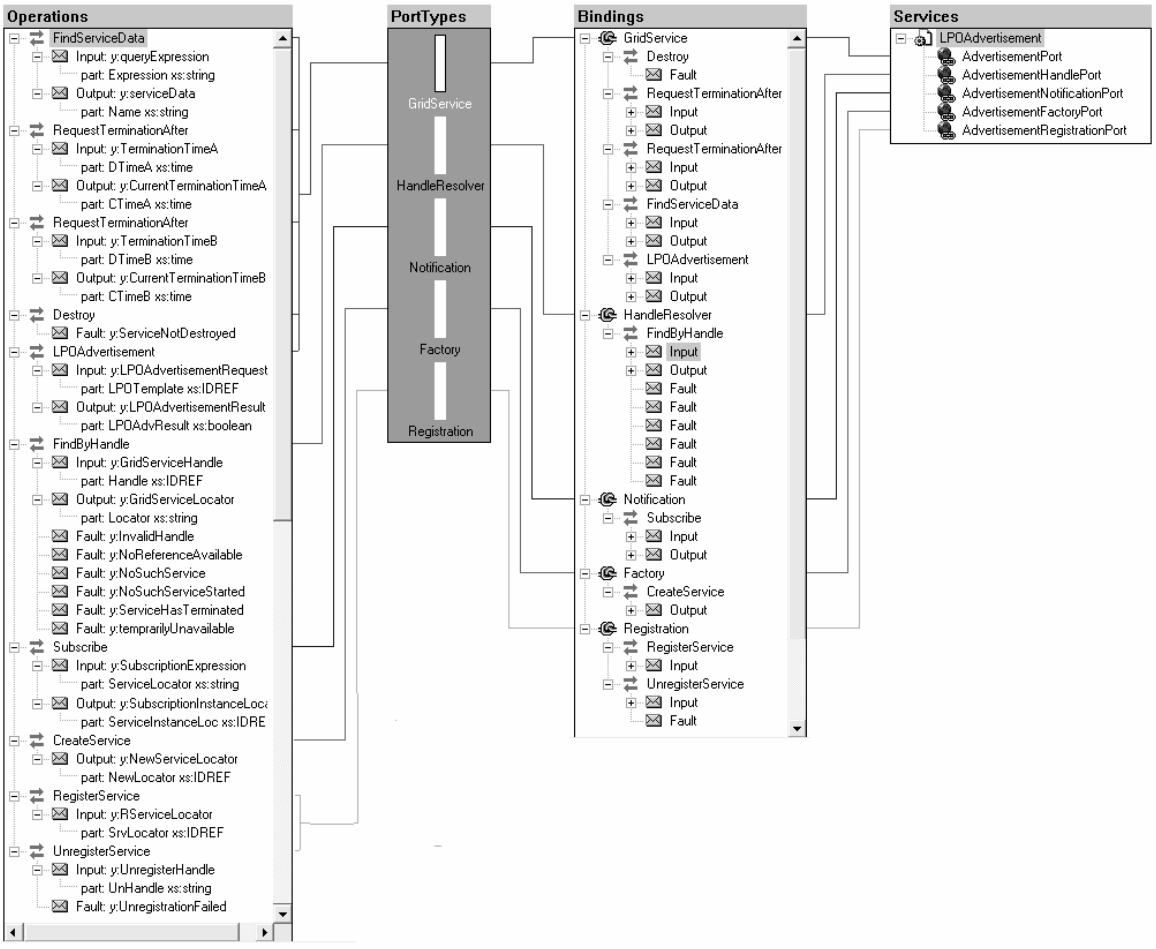

Figure 13: LPO advertisement service screen capture from XMLSpy.

12.2 Sample XML Schema

A sample XML schema that defines the types in our system.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by basem (u waterloo) --
>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
     <xs:element name="UofWaterloo">
          <xs:annotation>
               <xs:documentation> the root element is university of waterloo
</xs:documentation>
          </xs:annotation>
          <xs:complexType>
               <xs:sequence>
                    <xs:element name="LPOTemplate" type="LPOTemplateType"/>
                    <xs:element ref="User" maxOccurs="unbounded"/>
               </xs:sequence>
          </xs:complexType>
     </xs:element>
     <xs:complexType name="LPOTemplateType">
          <xs:sequence>
               <xs:element name="LPOID" type="xs:string"/>
               <xs:element name="Status" type="xs:int"/>
               <xs:element name="prevStatus" type="xs:int"/>
               <xs:element name="ownerStack" type="OwnerStackType"/>
               <xs:element name="parentLPOid" type="xs:string"/>
               <xs:element name="sourceSWTID" type="xs:string"/>
               <xs:element name="destSWTID" type="xs:string"/>
               <xs:element name="isSpawnable" type="xs:boolean"/>
               <xs:element name="bandwidth" type="xs:long"/>
               <xs:element name="expireDate" type="xs:date"/>
               <xs:element name="childList" type="List"/>
               <xs:element name="LPOList" type="List"/>
               <xs:element name="LoopbackLPO" type="xs:string"/>
          </xs:sequence>
     </xs:complexType>
     <xs:complexType name="OwnerStackType">
          <xs:sequence>
               <xs:element name="LPO_ownerID" type="xs:string"/>
          </xs:sequence>
     </xs:complexType>
     <xs:complexType name="List"/>
     <xs:element name="User">
          <xs:complexType>
               <xs:complexContent>
                    <xs:extension base="UserType">
                         <xs:sequence>
                              <xs:element name="Name" type="xs:string"/>
                              <xs:element name="Authentication" type="xs:string"/>
                         </xs:sequence>
                    </xs:extension>
               </xs:complexContent>
          </xs:complexType>
     </xs:element>
     <xs:complexType name="UserType">
          <xs:choice>
               <xs:element name="DomainAdmin" type="xs:string"/>
               <xs:element name="Client" type="xs:string"/>
          </xs:choice>
     </xs:complexType>
```

```
</xs:schema>
```

12.3 Sample DTD

The following is a sample DTD file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by basem (u
waterloo) -->
<!--DTD generated by XMLSPY v5 rel. 2 U (http://www.xmlspy.com)-->
<!-- the root element is university of waterloo -->
<!ELEMENT UofWaterloo (LPOTemplate, User+)>
<!ELEMENT User ((DomainAdmin | Client), (Name, Authentication))>
<!ELEMENT LPOTemplate (LPOID, Status, prevStatus, ownerStack, parentLPOid,
sourceSWTID, destSWTID, isSpawnable, bandwidth, expireDate, childList,
LPOList, LoopbackLPO)>
<!ELEMENT DomainAdmin (#PCDATA)>
<!ELEMENT Client (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Authentication (#PCDATA)>
<!ELEMENT LPOID (#PCDATA)>
<!ELEMENT Status (#PCDATA)>
<!ELEMENT prevStatus (#PCDATA)>
<!ELEMENT ownerStack (LPO_ownerID)>
<!ELEMENT parentLPOid (#PCDATA)>
<!ELEMENT sourceSWTID (#PCDATA)>
<!ELEMENT destSWTID (#PCDATA)>
<!ELEMENT isSpawnable (#PCDATA)>
<!ELEMENT bandwidth (#PCDATA)>
<!ELEMENT expireDate (#PCDATA)>
<!ELEMENT childList EMPTY>
<!ELEMENT LPOList EMPTY>
<!ELEMENT LoopbackLPO (#PCDATA)>
<!ELEMENT LPO_ownerID (#PCDATA)>
```

12.4 Sample WSDL Documents

The first WSDL document describes the LPO advertisement service. The second one describes the LPO spawning service. The third one describes the ETE LP establishment service.

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:y="http://bbcr.uwaterloo.ca/LPOservices/"
targetNamespace="http://bbcr.uwaterloo.ca/LPOservices/">
     <types>
          <xs:schema/>
     </types>
     <message name="messageName"/>
     <message name="queryExpression">
```

```
     <part name="Expression" type="xs:string"/>
</message>
<message name="serviceData">
     <part name="Name" type="xs:string"/>
</message>
<message name="TerminationTimeA">
     <part name="DTimeA" type="xs:time"/>
</message>
<message name="CurrentTerminationTimeA">
     <part name="CTimeA" type="xs:time"/>
</message>
<message name="TerminationTimeB">
     <part name="DTimeB" type="xs:time"/>
</message>
<message name="CurrentTerminationTimeB">
     <part name="CTimeB" type="xs:time"/>
</message>
<message name="ServiceNotDestroyed"/>
<message name="GridServiceHandle">
     <part name="Handle" type="xs:IDREF"/>
</message>
<message name="GridServiceLocator">
     <part name="Locator" type="xs:string"/>
</message>
<message name="InvalidHandle"/>
<message name="NoReferenceAvailable"/>
<message name="NoSuchService"/>
<message name="NoSuchServiceStarted"/>
<message name="ServiceHasTerminated"/>
<message name="temprarilyUnavailable"/>
<message name="SubscriptionExpression">
     <part name="ServiceLocator" type="xs:string"/>
</message>
<message name="SubscriptionInstanceLocator">
     <part name="ServiceInstanceLoc" type="xs:IDREF"/>
</message>
<message name="NewMessage"/>
<message name="ServiceParameters">
     <part name="LPOTemplate" type="xs:IDREF"/>
</message>
<message name="NewServiceLocator">
     <part name="NewLocator" type="xs:IDREF"/>
</message>
<message name="RServiceLocator">
     <part name="SrvLocator" type="xs:IDREF"/>
</message>
<message name="UnregisterHandle">
     <part name="UnHandle" type="xs:string"/>
</message>
<message name="UnregistrationFailed"/>
<message name="LPOAdvertisementRequest">
     <part name="LPOTemplate" type="xs:IDREF"/>
</message>
<message name="LPOAdvertisementResult">
     <part name="LPOAdvResult" type="xs:boolean"/>
</message>
<portType name="GridService">
     <operation name="FindServiceData">
          <input message="y:queryExpression"/>
          <output message="y:serviceData"/>
     </operation>
     <operation name="RequestTerminationAfter">
          <input message="y:TerminationTimeA"/>
```

```
                <output message="y:CurrentTerminationTimeA"/>
            </operation>
            <operation name="RequestTerminationAfter">
                <input message="y:TerminationTimeB"/>
                <output message="y:CurrentTerminationTimeB"/>
            </operation>
            <operation name="Destroy">
                <fault name="FaultName" message="y:ServiceNotDestroyed"/>
            </operation>
            <operation name="LPOAdvertisement">
                <input message="y:LPOAdvertisementRequest"/>
                <output message="y:LPOAdvertisementResult"/>
            </operation>
        </portType>
        <portType name="HandleResolver">
            <operation name="FindByHandle">
                <input message="y:GridServiceHandle"/>
                <output message="y:GridServiceLocator"/>
                <fault name="InvalidH" message="y:InvalidHandle"/>
                <fault name="noReference" message="y:NoReferenceAvailable"/>
                <fault name="noService" message="y:NoSuchService"/>
                <fault name="serviceNotStarted" message="y:NoSuchServiceStarted"/>
                <fault name="serviceTerminated" message="y:ServiceHasTerminated"/>
                <fault name="TempUnavailable" message="y:temprarilyUnavailable"/>
            </operation>
        </portType>
        <portType name="Notification">
            <operation name="Subscribe">
                <input message="y:SubscriptionExpression"/>
                <output message="y:SubscriptionInstanceLocator"/>
            </operation>
        </portType>
        <portType name="Factory">
            <operation name="CreateService">
                <output message="y:NewServiceLocator"/>
            </operation>
        </portType>
        <portType name="Registration">
            <operation name="RegisterService">
                <input message="y:RServiceLocator"/>
            </operation>
            <operation name="UnregisterService">
                <input message="y:UnregisterHandle"/>
                <fault name="UnFail" message="y:UnregistrationFailed"/>
            </operation>
        </portType>
        <binding name="GridService" type="y:GridService">
            <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="Destroy">
                <fault name="FaultName"/>
            </operation>
            <operation name="RequestTerminationAfter">
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
            <operation name="RequestTerminationAfter">
                <input>
                    <soap:body use="literal"/>
```

```
                </input>
                <output>
                     <soap:body use="literal"/>
                </output>
           </operation>
           <operation name="FindServiceData">
                <input>
                     <soap:body use="literal"/>
                </input>
                <output>
                     <soap:body use="literal"/>
                </output>
           </operation>
           <operation name="LPOAdvertisement">
                <soap:operation soapAction="urn:#LPOAdvertisement"/>
                <input>
                     <soap:body use="literal"/>
                </input>
                <output>
                     <soap:body use="literal"/>
                </output>
                <soap:operation soapAction="urn:#LPOAdvertisement"/>
           </operation>
     </binding>
     <binding name="HandleResolver" type="y:HandleResolver">
           <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
           <operation name="FindByHandle">
                <soap:operation soapAction="urn:#FindByHandle"/>
                <input>
                     <soap:body use="literal"/>
                </input>
                <soap:operation soapAction="urn:#FindByHandle"/>
                <output>
                     <soap:body use="literal"/>
                </output>
                <fault name="InvalidH"/>
                <fault name="noReference"/>
                <fault name="noService"/>
                <fault name="serviceNotStarted"/>
                <fault name="serviceTerminated"/>
                <fault name="TempUnavailable"/>
           </operation>
     </binding>
     <binding name="Notification" type="y:Notification">
           <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
           <operation name="Subscribe">
                <soap:operation soapAction="urn:#Subscribe"/>
                <input>
                     <soap:body use="literal"/>
                </input>
                <soap:operation soapAction="urn:#Subscribe"/>
                <output>
                     <soap:body use="literal"/>
                </output>
           </operation>
     </binding>
     <binding name="Factory" type="y:Factory">
           <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
           <operation name="CreateService">
                <soap:operation soapAction="urn:#CreateService"/>
```

```
                <output>
                     <soap:body use="literal"/>
                </output>
           </operation>
     </binding>
     <binding name="Registration" type="y:Registration">
           <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
           <operation name="RegisterService">
                <soap:operation soapAction="urn:#RegisterService"/>
                <input>
                     <soap:body use="literal"/>
                </input>
                <soap:operation soapAction="urn:#RegisterService"/>
           </operation>
           <operation name="UnregisterService">
                <soap:operation soapAction="urn:#UnregisterService"/>
                <input>
                     <soap:body use="literal"/>
                </input>
                <soap:operation soapAction="urn:#UnregisterService"/>
                <fault name="UnFail"/>
           </operation>
     </binding>
     <service name="LPOAdvertisement">
           <port name="AdvertisementPort" binding="y:GridService">
                <soap:address location="No Target Adress"/>
           </port>
           <port name="AdvertisementHandlePort" binding="y:HandleResolver">
                <soap:address location="No Target Adress"/>
           </port>
           <port name="AdvertisementNotificationPort" binding="y:Notification">
                <soap:address location="No Target Adress"/>
           </port>
           <port name="AdvertisementFactoryPort" binding="y:Factory">
                <soap:address location="No Target Adress"/>
           </port>
           <port name="AdvertisementRegistrationPort" binding="y:Registration">
                <soap:address location="No Target Adress"/>
           </port>
     </service>
</definitions>
```

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:y="http://bbcr.uwaterloo.ca/LPOservices/"
targetNamespace="http://bbcr.uwaterloo.ca/LPOservices/">
     <types>
          <xs:schema/>
     </types>
     <message name="messageName"/>
     <message name="queryExpression">
          <part name="Expression" type="xs:string"/>
     </message>
     <message name="serviceData">
          <part name="Name" type="xs:string"/>
     </message>
     <message name="TerminationTimeA">
```

```
     <part name="DTimeA" type="xs:time"/>
</message>
<message name="CurrentTerminationTimeA">
     <part name="CTimeA" type="xs:time"/>
</message>
<message name="TerminationTimeB">
     <part name="DTimeB" type="xs:time"/>
</message>
<message name="CurrentTerminationTimeB">
     <part name="CTimeB" type="xs:time"/>
</message>
<message name="ServiceNotDestroyed"/>
<message name="GridServiceHandle">
     <part name="Handle" type="xs:IDREF"/>
</message>
<message name="GridServiceLocator">
     <part name="Locator" type="xs:string"/>
</message>
<message name="InvalidHandle"/>
<message name="NoReferenceAvailable"/>
<message name="NoSuchService"/>
<message name="NoSuchServiceStarted"/>
<message name="ServiceHasTerminated"/>
<message name="temprarilyUnavailable"/>
<message name="SubscriptionExpression">
     <part name="ServiceLocator" type="xs:string"/>
</message>
<message name="SubscriptionInstanceLocator">
     <part name="ServiceInstanceLoc" type="xs:IDREF"/>
</message>
<message name="NewMessage"/>
<message name="ServiceParameters"/>
<message name="NewServiceLocator">
     <part name="NewLocator" type="xs:IDREF"/>
</message>
<message name="RServiceLocator">
     <part name="SrvLocator" type="xs:IDREF"/>
</message>
<message name="UnregisterHandle">
     <part name="UnHandle" type="xs:string"/>
</message>
<message name="UnregistrationFailed"/>
<message name="LPOSpawningRequest">
     <part name="LPOTemplate" type="xs:IDREF"/>
     <part name="LPOID" type="xs:string"/>
</message>
<message name="LPOSpawningResult">
     <part name="LPOID" type="xs:string"/>
</message>
<portType name="GridService">
     <operation name="FindServiceData">
          <input message="y:queryExpression"/>
          <output message="y:serviceData"/>
     </operation>
     <operation name="RequestTerminationAfter">
          <input message="y:TerminationTimeA"/>
          <output message="y:CurrentTerminationTimeA"/>
     </operation>
     <operation name="RequestTerminationAfter">
          <input message="y:TerminationTimeB"/>
          <output message="y:CurrentTerminationTimeB"/>
     </operation>
     <operation name="Destroy">
```

```
                <fault name="FaultName" message="y:ServiceNotDestroyed"/>
            </operation>
            <operation name="LPOSpawning">
                <input message="y:LPOSpawningRequest"/>
                <output message="y:LPOSpawningResult"/>
            </operation>
        </portType>
        <portType name="HandleResolver">
            <operation name="FindByHandle">
                <input message="y:GridServiceHandle"/>
                <output message="y:GridServiceLocator"/>
                <fault name="InvalidH" message="y:InvalidHandle"/>
                <fault name="noReference" message="y:NoReferenceAvailable"/>
                <fault name="noService" message="y:NoSuchService"/>
                <fault name="serviceNotStarted" message="y:NoSuchServiceStarted"/>
                <fault name="serviceTerminated" message="y:ServiceHasTerminated"/>
                <fault name="TempUnavailable" message="y:temprarilyUnavailable"/>
            </operation>
        </portType>
        <portType name="Notification">
            <operation name="Subscribe">
                <input message="y:SubscriptionExpression"/>
                <output message="y:SubscriptionInstanceLocator"/>
            </operation>
        </portType>
        <portType name="Factory">
            <operation name="CreateService">
                <output message="y:NewServiceLocator"/>
            </operation>
        </portType>
        <portType name="Registration">
            <operation name="RegisterService">
                <input message="y:RServiceLocator"/>
            </operation>
            <operation name="UnregisterService">
                <input message="y:UnregisterHandle"/>
                <fault name="UnFail" message="y:UnregistrationFailed"/>
            </operation>
        </portType>
        <binding name="GridService" type="y:GridService">
            <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="Destroy">
                <fault name="FaultName"/>
            </operation>
            <operation name="RequestTerminationAfter">
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
            <operation name="RequestTerminationAfter">
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
            <operation name="FindServiceData">
                <input>
```

```
                    <soap:body use="literal"/>
               </input>
               <output>
                    <soap:body use="literal"/>
               </output>
          </operation>
          <operation name="LPOSpawning">
               <soap:operation soapAction="urn:#LPOSpawning"/>
               <input>
                    <soap:body use="literal"/>
               </input>
               <output>
                    <soap:body use="literal"/>
               </output>
               <soap:operation soapAction="urn:#LPOSpawning"/>
          </operation>
     </binding>
     <binding name="HandleResolver" type="y:HandleResolver">
          <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
          <operation name="FindByHandle">
               <soap:operation soapAction="urn:#FindByHandle"/>
               <input>
                    <soap:body use="literal"/>
               </input>
               <soap:operation soapAction="urn:#FindByHandle"/>
               <output>
                    <soap:body use="literal"/>
               </output>
               <fault name="InvalidH"/>
               <fault name="noReference"/>
               <fault name="noService"/>
               <fault name="serviceNotStarted"/>
               <fault name="serviceTerminated"/>
               <fault name="TempUnavailable"/>
          </operation>
     </binding>
     <binding name="Notification" type="y:Notification">
          <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
          <operation name="Subscribe">
               <soap:operation soapAction="urn:#Subscribe"/>
               <input>
                    <soap:body use="literal"/>
               </input>
               <soap:operation soapAction="urn:#Subscribe"/>
               <output>
                    <soap:body use="literal"/>
               </output>
          </operation>
     </binding>
     <binding name="Factory" type="y:Factory">
          <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
          <operation name="CreateService">
               <soap:operation soapAction="urn:#CreateService"/>
               <output>
                    <soap:body use="literal"/>
               </output>
          </operation>
     </binding>
     <binding name="Registration" type="y:Registration">
```

```
          <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
          <operation name="RegisterService">
               <soap:operation soapAction="urn:#RegisterService"/>
               <input>
                    <soap:body use="literal"/>
               </input>
               <soap:operation soapAction="urn:#RegisterService"/>
          </operation>
          <operation name="UnregisterService">
               <soap:operation soapAction="urn:#UnregisterService"/>
               <input>
                    <soap:body use="literal"/>
               </input>
               <soap:operation soapAction="urn:#UnregisterService"/>
               <fault name="UnFail"/>
          </operation>
     </binding>
     <service name="LPOspawning">
          <port name="SpawningPort" binding="y:GridService">
               <soap:address location="No Target Adress"/>
          </port>
          <port name="SpawningHandlePort" binding="y:HandleResolver">
               <soap:address location="No Target Adress"/>
          </port>
          <port name="SpawningtNotificationPort" binding="y:Notification">
               <soap:address location="No Target Adress"/>
          </port>
          <port name="SpawningFactoryPort" binding="y:Factory">
               <soap:address location="No Target Adress"/>
          </port>
          <port name="SpawningRegistrationPort" binding="y:Registration">
               <soap:address location="No Target Adress"/>
          </port>
     </service>
</definitions>
```

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:y="http://bbcr.uwaterloo.ca/LPOservices/"
targetNamespace="http://bbcr.uwaterloo.ca/LPOservices/">
     <types>
          <xs:schema/>
     </types>
     <message name="messageName"/>
     <message name="queryExpression">
          <part name="Expression" type="xs:string"/>
     </message>
     <message name="serviceData">
          <part name="Name" type="xs:string"/>
     </message>
     <message name="TerminationTimeA">
          <part name="DTimeA" type="xs:time"/>
     </message>
     <message name="CurrentTerminationTimeA">
          <part name="CTimeA" type="xs:time"/>
     </message>
     <message name="TerminationTimeB">
```

```
     <part name="DTimeB" type="xs:time"/>
</message>
<message name="CurrentTerminationTimeB">
     <part name="CTimeB" type="xs:time"/>
</message>
<message name="ServiceNotDestroyed"/>
<message name="GridServiceHandle">
     <part name="Handle" type="xs:IDREF"/>
</message>
<message name="GridServiceLocator">
     <part name="Locator" type="xs:string"/>
</message>
<message name="InvalidHandle"/>
<message name="NoReferenceAvailable"/>
<message name="NoSuchService"/>
<message name="NoSuchServiceStarted"/>
<message name="ServiceHasTerminated"/>
<message name="temprarilyUnavailable"/>
<message name="SubscriptionExpression">
     <part name="ServiceLocator" type="xs:string"/>
</message>
<message name="SubscriptionInstanceLocator">
     <part name="ServiceInstanceLoc" type="xs:IDREF"/>
</message>
<message name="NewMessage"/>
<message name="ServiceParameters"/>
<message name="NewServiceLocator">
     <part name="NewLocator" type="xs:IDREF"/>
</message>
<message name="RServiceLocator">
     <part name="SrvLocator" type="xs:IDREF"/>
</message>
<message name="UnregisterHandle">
     <part name="UnHandle" type="xs:string"/>
</message>
<message name="UnregistrationFailed"/>
<message name="ETELpEstablishmentRequest">
     <part name="sourceSWTID" type="xs:string"/>
     <part name="destSWTID" type="xs:string"/>
     <part name="bandwidth" type="xs:long"/>
     <part name="isLoopback" type="xs:boolean"/>
</message>
<message name="ETELpEstablishmentResponse">
     <part name="LPOSpnResult" type="xs:string"/>
</message>
<portType name="GridService">
     <operation name="FindServiceData">
          <input message="y:queryExpression"/>
          <output message="y:serviceData"/>
     </operation>
     <operation name="RequestTerminationAfter">
          <input message="y:TerminationTimeA"/>
          <output message="y:CurrentTerminationTimeA"/>
     </operation>
     <operation name="RequestTerminationAfter">
          <input message="y:TerminationTimeB"/>
          <output message="y:CurrentTerminationTimeB"/>
     </operation>
     <operation name="Destroy">
          <fault name="FaultName" message="y:ServiceNotDestroyed"/>
     </operation>
     <operation name="ETELpEstablishment">
          <input message="y:ETELpEstablishmentRequest"/>
```

```
                <output message="y:ETELpEstablishmentResponse"/>
            </operation>
        </portType>
        <portType name="HandleResolver">
            <operation name="FindByHandle">
                <input message="y:GridServiceHandle"/>
                <output message="y:GridServiceLocator"/>
                <fault name="InvalidH" message="y:InvalidHandle"/>
                <fault name="noReference" message="y:NoReferenceAvailable"/>
                <fault name="noService" message="y:NoSuchService"/>
                <fault name="serviceNotStarted" message="y:NoSuchServiceStarted"/>
                <fault name="serviceTerminated" message="y:ServiceHasTerminated"/>
                <fault name="TempUnavailable" message="y:temprarilyUnavailable"/>
            </operation>
        </portType>
        <portType name="Notification">
            <operation name="Subscribe">
                <input message="y:SubscriptionExpression"/>
                <output message="y:SubscriptionInstanceLocator"/>
            </operation>
        </portType>
        <portType name="Factory">
            <operation name="CreateService">
                <output message="y:NewServiceLocator"/>
            </operation>
        </portType>
        <portType name="Registration">
            <operation name="RegisterService">
                <input message="y:RServiceLocator"/>
            </operation>
            <operation name="UnregisterService">
                <input message="y:UnregisterHandle"/>
                <fault name="UnFail" message="y:UnregistrationFailed"/>
            </operation>
        </portType>
        <binding name="GridService" type="y:GridService">
            <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="Destroy">
                <fault name="FaultName"/>
            </operation>
            <operation name="RequestTerminationAfter">
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
            <operation name="RequestTerminationAfter">
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
            <operation name="FindServiceData">
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
```

```
                </output>
            </operation>
            <operation name="ETELpEstablishment">
                <soap:operation soapAction="urn:#LPOEstablishment"/>
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
                <soap:operation soapAction="urn:#LPOEstablishment"/>
            </operation>
        </binding>
        <binding name="HandleResolver" type="y:HandleResolver">
            <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="FindByHandle">
                <soap:operation soapAction="urn:#FindByHandle"/>
                <input>
                    <soap:body use="literal"/>
                </input>
                <soap:operation soapAction="urn:#FindByHandle"/>
                <output>
                    <soap:body use="literal"/>
                </output>
                <fault name="InvalidH"/>
                <fault name="noReference"/>
                <fault name="noService"/>
                <fault name="serviceNotStarted"/>
                <fault name="serviceTerminated"/>
                <fault name="TempUnavailable"/>
            </operation>
        </binding>
        <binding name="Notification" type="y:Notification">
            <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="Subscribe">
                <soap:operation soapAction="urn:#Subscribe"/>
                <input>
                    <soap:body use="literal"/>
                </input>
                <soap:operation soapAction="urn:#Subscribe"/>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
        </binding>
        <binding name="Factory" type="y:Factory">
            <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="CreateService">
                <soap:operation soapAction="urn:#CreateService"/>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
        </binding>
        <binding name="Registration" type="y:Registration">
            <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="RegisterService">
                <soap:operation soapAction="urn:#RegisterService"/>
                <input>
```

```
                    <soap:body use="literal"/>
               </input>
               <soap:operation soapAction="urn:#RegisterService"/>
          </operation>
          <operation name="UnregisterService">
               <soap:operation soapAction="urn:#UnregisterService"/>
               <input>
                    <soap:body use="literal"/>
               </input>
               <soap:operation soapAction="urn:#UnregisterService"/>
               <fault name="UnFail"/>
          </operation>
     </binding>
     <service name="LPOspawning">
          <port name="ETELpEstablishmentPort" binding="y:GridService">
               <soap:address location="No Target Adress"/>
          </port>
          <port name="ETELpEstablishmentHandlePort" binding="y:HandleResolver">
               <soap:address location="No Target Adress"/>
          </port>
          <port name="ETELpEstablishmentNotificationPort" binding="y:Notification">
               <soap:address location="No Target Adress"/>
          </port>
          <port name="ETELpEstablishmentFactoryPort" binding="y:Factory">
               <soap:address location="No Target Adress"/>
          </port>
          <port name="ETELpEstablishmentRegistrationPort" binding="y:Registration">
               <soap:address location="No Target Adress"/>
          </port>
     </service>
</definitions>
```
