User Controlled LightPaths

Carol Meertens, Tijmen van den Brink

February 2, 2007

Contents

List of Figures

List of Tables

Abstract

UCLP is a service oriented system offering the user(-process) the ability to set up and tear down lightpaths at will. A testbed was configured using three Nortel OME6500 SDH switches and a UCLP installation: version 2 of the 'Ottawa implementation'. We investigated the design and development status of UCLP and compared it to DRAC. UCLP is built upon existing and open SOAP technology. In its design the concept of 'network' is being kept very generic, so the UCLP system will well be able to adopt to future insights. The implementation investigated isn't 'production ready'.

1 Introduction

This is the report of our one month internship at SARA. We have been installing an implementation of 'User Controlled LightPaths', UCLP. We assembled a working UCLP testbed and evaluated the UCLP system.

Today creating connections in the field of optical networks can be quite a burden and costly. There is a lot of administrative labour associated with creating a lightpath. And once that is done there often isn't a 100% guarantee the lightpath will be there at the right time. In some environments this is not acceptable. Think about grid and optical network research: there is a real need for lightpaths both dynamically configurable ánd reliable.

UCLP is an attempt to offer the user a way to setup and tear down lightpaths without any of the above mentioned the administrative burden and risc. The design of UCLP is based on the broad concept of 'Service Orientation Architecture'. Among other things this means the 'user' does not need to be a human being interacting through a GUI. The user can (and will very often) be an application, for example a distributed system interfacing with UCLP through UCLP's API. It will be able to setup and tear down lightpaths at will.

2 Objectives

The primary objective of this project was to build a testbed consisting of some optical hardware and the UCLP system. SARA was interested in the design and in the development status of UCLP. Three Nortel OME 6500's were available in an existing SURFnet testbed.

Other objectives were to compare its functionality with DRAC's and to evaluate interoperability aspects between the two. Some examples of interesting questions to answer:

- Is it difficult install UCLP; what is the quality of the software?
- What are the UCLP design choices made?
- What kind of user roles are defined?
- How does an end user create a lightpath?
- How does UCLP handle scheduling?

3 Design Principles

This section covers the design principles of UCLPv2. First the concept of a Service Oriented Architecture is explained, followed by the explanation of each layer as shown in Figure 3.1.

3.1 Service Oriented Architecture

The Service Oriented Architecture (SOA) is relatively new in the IT world. However the foundations of SOA date from ten years ago. To understand the design principles of UCLP it's important to understand the concept of a SOA and why it's so usefull. This will be explained in this section.

The evolution of programming shows us that in the beginning monolithic software was written. This soon led to software getting too complex. The next step was to modularize the software so they could also reuse their code. SOA adds the possibility to remotely use services offered by systems using their own technology.

A SOA is an architecture in which resources (e.g. network elements, interfaces, lightpaths) are being exposed as services. Such a service's interface must be platform-independent and should be dynamically discoverable. Another requirement is that it should be possible to remotely invoke this service. In a lot of cases these services are web services because of the protocols used (HTTP, XML, UDDI, WSDL and SOAP). By offering these seperately accessed services that could be part of a larger application a few problems are eliminated. First of all a service can be reused by several applications. Secondly they use the same instance of the service and thus a bug in one service will cause only that service to be fixed as opposed to fixing a bug in a module that has been used in the code of several different applications.

UCLP's SOA is depicted in Figure 3.1.The lower layer exposes web services that manage and control network elements. For the Higher Level Services or Applications to control the network elements the second layer provides an abstract interface. This way the underlying technology is of no importance for the Higher Level Services or Applications which makes integration into applications less complex and more robust.

The next subsections (subsection 3.2, 3.3 and 3.4) will discuss these layers in more detail.

3.2 Resource Management

As stated earlier in this document the lower layer exposes web services, called Network Element Web Services (NE-WS), that control the resources of physical network elements. The way these devices are managed depends on the technology they use. As of today only one technology has been implemented, by the developers of UCLPv2[1], as NE-WS, which is the Cross Connect WS (XC-WS). This WS controls physical devices and is able to create and delete "Cross Connections". As depicted in Figure 3.1 a lot more technologies are possible and could be implemented. A few of these technologies are [1]:

[1]New implementations as the MPLS-WS and an Ethernet-WS are known to be implemented by HEAnet which is Ireland's National Education & Research network

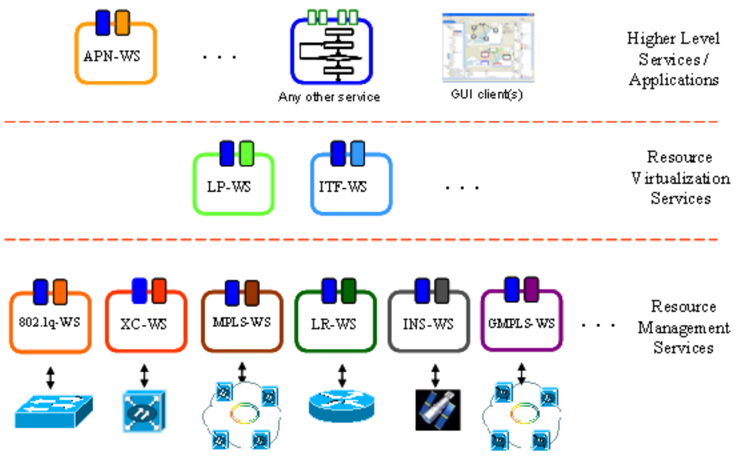


Figure 1: UCLPv2 Service Oriented Architecture. [1]

- XC-WS: Manages devices that are able to create Cross Connects on SONET/SDH based equipment, wavelength switching based equipment and fibre cross connects. Currently the XC-WS is the only member of the NE-WS family that has been implemented.
- 802.1q-WS: Manages devices that use Virtual Local Area Network (VLAN) technology to multiplex connections belonging to different users.
- LR-WS: Manages logical routers. Some routers are able to partition a single router into multiple logical devices that perform independent routing tasks (each logical instance has its own routing table and protocols). These logical devices are called logical routers.
- MPLS-WS: Allows UCLP to control MultiProtocol Label Switching (MPLS) based devices.
- INS-WS: Presents a simple interface to control instruments such as sensors, data sinks/sources and storage devices.
- GMPLS-WS: Allows UCLPv2 to trigger switched Generalized MPLS (GMPLS) connections using the O-UNI (Optical User to Network Interface).

To implement services the Web Service Description Language (WSDL) is used. It's an XML-based language that describes services with it's functions [2]. In case of the XC-WS two port types are created. One to manage the network element (XC ConfigPort) and one to manage the resources (XC Operational-Port). Relatively they contain functions to create, modify and delete network elements and to use, release, bond, partition, (un)allocate, query subLease and return resources.

In order to remotely use these functions XML-based messages are sent invoking a particular function. The protocol used to send XML-based messages over HTTP is the Simple Object Access Protocol (SOAP)[3]. The UCLP backend uses Apache Axis as it's SOAP Engine. A client can connect to a web service and query or invoke the available functions that are listed in a WSDL file.

At the moment the configuration of the physical devices relies on TL1. In the future Command Line Interface (CLI) and NetConf[2] will be added. To establish the connections between the NE-WS and the physical devices UCLPv2 supports TCP, UDP and SSL.

3.3 Virtualization layer

As stated in subsection 3.1 the Resource Virtualization Services layer provides an virtualized interface to the Higher Level Services or Applications in order to control the network elements. At this moment there are two types of services, being:

- Lightpath Web Services
- Interface Web Services

To provide a standard interface to the top layer some virtualization of the creation of a lightpath, that is technology dependent, has to be made. This is done by the Lighpath Web Service (LP-WS). Just as the XC-WS this service provides operations to manipulate the layer below it. These operations are [1]:

- Create: Creates a new instance of a LP-WS.
- Delete: Destroys the LP-WS instance. If it has a finite lifetime, the LP-WS instance will destroy itself upon lifetime expiration.
- Query: Provides data about the LP-WS properties (bandwidth, endpoints, expiration date, etc.) and status (in use, available, faulted).
- Use: Causes the LP-WS to call the NE-WS, which in turn will configure the network element in a certain way depending on the network element technology (create a cross-connection, create a VLAN or create an LSP).
- Release: Reverts the use operation, causes the LP-WS to call the network element web services, which in turn clean the network element configurations.
- Partition: If it is physically possible, the LP-WS is divided in N LP-WSs with smaller bandwidth. The addition of all the bandwidths must match the original LP-WS bandwidth.
- Bond: If it is physically possible, a group of LP-WS with the same network endpoints is combined into a single LP-WS whose bandwidth is the addition of all the original LP-WSs bandwidth.
- Lease: Changes the ownership of the LP-WS.

The same applies to the interface of a network element. An interface is a single port on a network element and to provide it's services to the top layer in a standardized manner an abstraction is made. This abstraction, called an Interface Web Service (ITF-WS) causes the underlying technology to be

[2]NetConf is a protocol to configure network elements.

hidden from the top layer just like the LP-WS. The operations provided by this service are the same as the ones of the LP-WS except the `partition` and `bond` operations which do not apply to an interface.

These two services are the building blocks of the system. They can be used by users to create their own virtualized network but also by applications that can interface with these services. The biggest advantage in this layered and abstracted approach is that technology at the lower level is hidden from the top level, which allows for easy integration into applications and it increases robustness.

To orchestrate interactions between ITF-WS's and LP-WS's an XML-based language is used called "Business Process Execution Language" (BPEL). Using BPEL, workflows can be created like the LP-WS which is the orchestration of two web services that both control a network enabled endpoint. Note that these workflows can be modified to add more functionality for instance by invoking another web service.

3.4 Higher Level Services and Applications

The previous subsection discussed the abstracted interfaces the LP-WS and ITF-WS provide. Services and applications that reside on the top layer are now able to manipulate the underlying physical network without having to know what technologies are used. This great advantage can be used to build new services (e.g. VPN services, reservation services, bandwith on demand) on top of the existing services.

At the moment only one higher level service has been actually implemented. The Articulated Private Network (APN) service allows users or applications to combine the provided resources, LP-WS's and ITF-WS's, and create their own virtual network. Such a virtual network is called an APN scenario. One or more of these scenarios can be deployed to the UCLPv2 backend. After the scenarios are deployed one particular scenario can be set by an application or user and the underlying hardware will be configured accordingly. The following operations are offered by the APN-WS [1]:

- Init(userID): Initializes the APN (a new process is created in the BPEL engine). Some validation actions are performed to ensure the correctness of the device configurations.

- SetConfig(scenarioID, userID, usageTime): Performs all the device configurations specified in the scenario named scenarioID. Checks that the user userID has the valid access rights to set up the scenario. When the usageTime is over, the scenario is automatically unset.

- UnsetConfig(scenarioID, userID): Clears all the device configurations specified in the scenario name scenarioID. Checks that the user userID has the valid access rights to teardown the APN.

- QueryStatus(userID): Returns the status of the APN (i.e. provides information about the scenario being executed).

- Stop(userID): Destroys the process instance of the APN in the BPEL engine.

4 Security

This section covers how security is imlemented by using user roles in the UCLPv2 architecture. Besides these user roles, as UCLPv2 is still under development there are quite some security issues that need to be solved (e.g. storing passwords in clear text, datacommunication over an unsecured protocol). This report does no security audit on UCLPv2, but will discuss the security as it is implemented at this moment.

4.1 User roles

The UCLPv2 security architecture relies on user roles. This architecture can be explained by the layered model depicted in Figure 3.1 where the three different users can access and or modify services in one or more layers.

- An organization owns physical resources that are managed by one or more Physical Network Administrators (PN-Admin). The PN-Admin is authorized to create, use and modify all of the services within these layers. Besides using the resources himself he can also lease them to others.
- A Virtual Network Administrator (VN-Admin) operates at the top two layers. This means he will not be able to modify the physical network. Instead he is offered LP-WS's and ITF-WS's that he can use himself or lease to others. A VN-Admin is able to create APN's for himself and third parties and can add resources to a specific APN.
- The End User can only access the services a VN-Admin offered him.

All users can add users of their own organization that have the same or less privileges. This is done using the User Management Web Service (UM-WS) that is build on OpenLDAP[3]. Each user must be authenticated and their user role must be known before they can access the system. WS's use these credentials to validate the use of the WS by this particular user.

5 Installation and Usage of UCLP

5.1 The Testbed

We assembled a testbed using three Nortel OME 6500's that were available in an existing SURFnet6 testbed. The OME's are interconnected using the OC192 cards as a triangle; see Figure 5.1. One of the OME's is used to connect the host on which we installed UCLP system.

We have been using UCLP version 2; the 'Ottawa' implementation[4].

[3]LDAP stands for Lightweight Directory Access Protocol and is used to modify or query directory services.

[4]Communications Research Centre Canada (CRC), University of Ottawa, i2CAT Foundation and Inocybe Technologies. The software and documentation is downloaded from http://uclp.ca

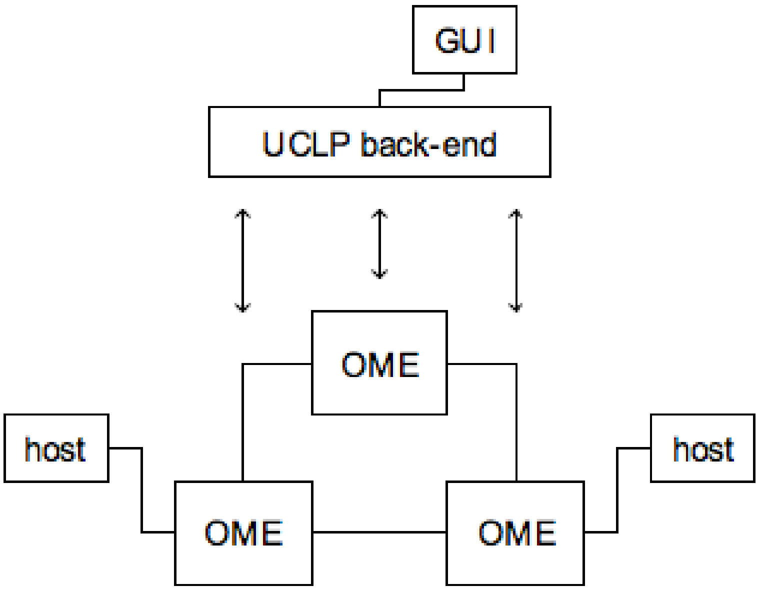


Figure 2: UCLP testbed

5.2 Creating a Lightpath

To give an impression of the 'look and feel' of the UCLP GUI we will describe shortly how a lightpath in UCLP is created. For more detail the user manual that comes with UCLP is a good source of information.

When starting the GUI one first needs to log in. After logging in the user adds the information about his network (network elements and links) to UCLP. He does that by using the 'Physical Network Editor', see Figure 5.2. When adding a network element to the Physical Network Editor one gets the opportunity to enter the required credentials, i.e. a login for a TL1 interface and/or the path to an SSL certificate.

Each link and each network element interface can be used to create a 'resource' web service. Resources are assembled in a resource list and can be added to or removed from an APN (Articulated Private Network). For each APN multiple scenario's can be created. Figure 5.2 shows a scenario resembling a lightpath.

The user activates one scenario at the time. On activation of a scenario the UCLP back-end system configures the network elements. In our testbed that

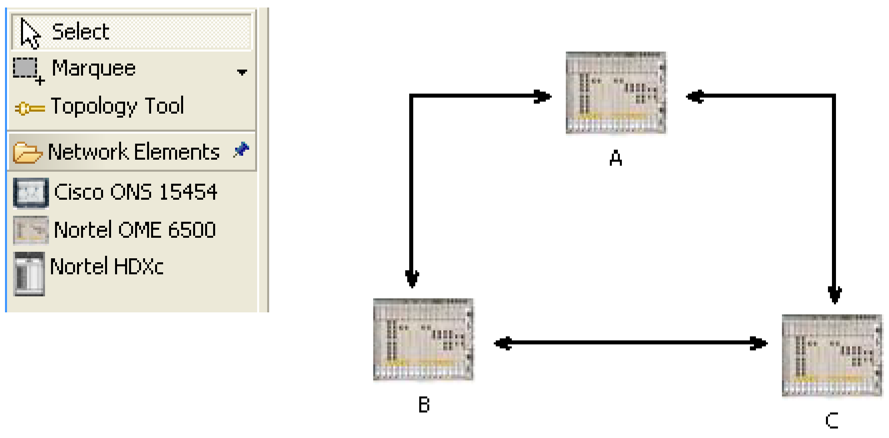


Figure 3: Physical Network Editor: letting UCLP know about NE's and links

means a telnet session is being set up between UCLP and the OME's and cross connects are being set using TL1 commands. By setting the cross connects a lightpath is created.

5.3 Issues

We have been installing UCLP version 2.0 on both Windows and Linux. It had been tested on Windows and the GUI currently is available as a Windows executable only. However: all the back-end software can be installed on Linux without problems (see Appendix). Halfway our research we were given a pre-release of UCLP version 2.1. The upgrade was seamless on both OS's.

At the same time we got version 2.1 our network administrator removed some wrong settings in the OME's in our testbed. We didn't try to deliberately re-misconfigure the OME's to test version 2.1 for this. Therefore sometimes we couldn't tell whether a problem has been solved by the upgrade or because of the removal of wrong OME settings.

This is an (incomplete) list of issues we encountered. It should give a rough idea about the development status UCLP currently is in:

- Version 2.0 sometimes hang when creating LP-WS's. No meaningfull error message other then a socket timeout after about 5 minutes. This problem disappeared after the OME cleanup and installation of version 2.1. Version 2.0 seemed to hang if there were zero cross connects in one of the OME's.
- In Version 2.0 A Java nullpointer exception was thrown when running an APN. After a lot of searching we discovered this should be related to the fact that UCLP does not support VCAT[5]. This isn't documented very clearly and the error message didn't help very much to find the problem. Error messages in version 2.1 appear to be more elaborate.

[5]Virtual conCATenation

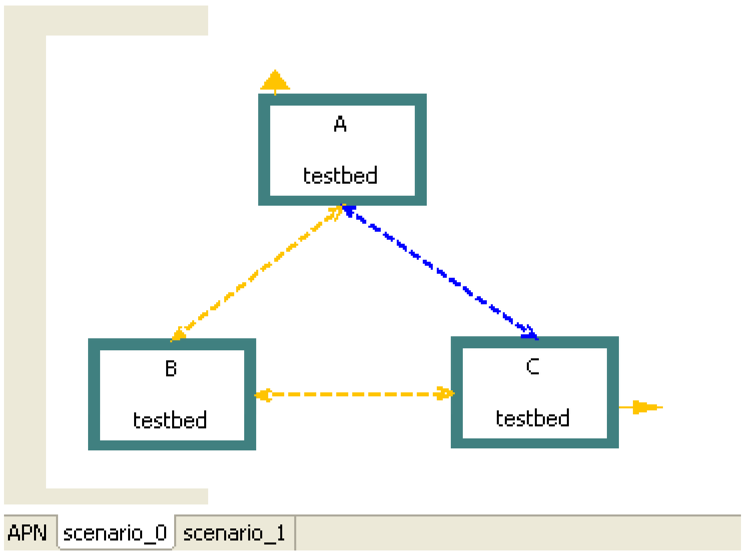
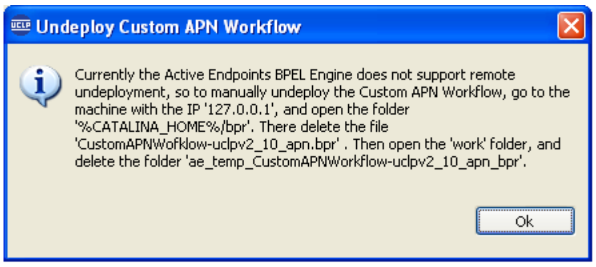


Figure 4: Lightpath in a ‘scenario’

- Sometimes, in both versions, when trying to create a connection one gets the error message ‘TTL less than required usage time’. After cleaning up the database and start over defining WS’s and APNs a connection can be created without this message. We did not find out what caused this.
- UCLP does not set a name in the Nortels for the cross connects it creates. (in Nortel’s Site Manager the cross connect’s name column is empty). For interational reasons it would be nice to have some indication that the cross connect was created by UCLP, an organization or user name or the endpoints of the associated lightpath.
- Creating a physical network in version 2.0 involved a lot of pointing, clicking and repetitive work. The introduction of profiles in version 2.1 greatly improves on this.
- Scheduling is not yet implemented.
- UCLP did not support the DWDM cards in the Nortel 6500. However the developers sent us a simple patch that solved this.
- Once when setting and unsetting a scenario in version 2.1 a link disappeared in the Physical Network Editor. We had to remove the APNs and WS’s before being able to recreate the link.
- Once in version 2.1 we removed a resource list in order to recreate it with the same name. Removing appeared to be successful. When trying to recreate it, we got a message it already existed. We were forced to use another name.
- Undeploying an APN in version 2.1 gives the following message:



```
Currently the Active Endpoints BPEL Engine does not support remote
undeployment, so to manually undeploy the Custom APN Workflow, go to the
machine with the IP '127.0.0.1', and open the folder
'%CATALINA_HOME%/bpr'. There delete the file
'CustomAPNWofklow-uclpv2_10_apn.bpr' . Then open the 'work' folder, and
delete the folder 'ae_temp_CustomAPNWorkflow-uclpv2_10_apn_bpr'.
```

Figure 5: Undeploying an APN

- The color property of links in the logical network editor cannot be changed. The GUI suggests it can be done, but the chosen color isn't applied. This was a real problem: our background image and the default color of a LP-WS were the same. Therefore links weren't visible.
- Sometimes when deploying an APN we got 'no target service to operate for'. Developers sent us a updated pre-release for this, but that didn't solve the problem.

The pre-release of version 2.1, seems to be more stable then version 2.0. Also the GUI shows some important improvements, i.e.:

- version 2.1 shows existing cross connects in white
- version 2.1 adds the concept of scenario's to define different APN setups
- version 2.1 seems to be working faster i.e. when creating ITF-WS's in PN editor

UCLP is built upon open technology: Tomcat, Axis, JSP, ... At the time of this writing source code and API for version 2 aren't available. UCLP is expected to be well extensible once the technical documentation is online. Version 2.1 and the accompanying technical documentation will be released in the first half 2007.

6 UCLP and DRAC

6.1 DRAC

Like UCLP DRAC (Dynamic Resource Allocation Control) is a service oriented system built using Java and JSP. End users using DRAC create their lightpath by selecting its two endpoints; i.e. by selecting two gigabit ethernet ports as endpoints for a connection. Requesting a lightpath is done by scheduling it using a web client or web services. The user always uses a start time and an end time when scheduling a lightpath. If the schedule is successful, DRAC will establish the lightpath during the given time interval.

The user explicitly needs to be authorized for the lightpath's endpoints. DRAC internally decides on the route between the endpoints. This decision is based on the values of a few optimization parameters.

6.2 Similarities

Both UCLP and DRAC are designed around the concept of (web-)service orientation. Both systems use Java technologies and existing standard toolsets to achieve this. This can make them portable and extendable. Neither of them is in production use today yet.

Both are meant to be used by end users, offering them a way to control lightpaths without the need to contact a service provider by mail or phone every time.

6.3 Differences

The main differences between DRAC and UCLP spring from their respective design objectives. DRAC offers a way to schedule single lightpaths. Bandwidth is guaranteed, but the topology of the network isn't. UCLP offers the user – or some user process – a dedicated APN. He can activate and de-activate multiple scenario's at will. This way he can change back and forth network topologies. Disabled resources will keep being available to him exclusively.

DRAC supports VCAT, automatic pathfinding and scheduling. It is possible to implement these capabilities in UCLP but this is something not yet done. These are not among UCLP's main objectives. Using DRAC is a means of making optimised use of bandwith, which is a financial benefit. UCLP holds the concept of multiple layers of services which can be delegated and subdelegated. DRAC doesn't. Any APN administrator can make part of his resources available to another user. In UCLP the use of 'Super Lightpaths' provide a way to hide internal network topology towards endusers.

Table 1 summarizes some important differences between UCLP and DRAC.

	DRAC	UCLP
Provides:	short lived lightpaths	long term APNs
Pathfinding:	by algorithm	by user(-process)
Scheduling:	both start end end date	end date planned
Bandwidth usage:	optimising	fixed
VCAT support:	yes	no
Strength:	scheduling	scenario's, abstraction
Delegation of resources:	no	yes
Client:	browser	windows executable
Licence:	proprietary	GPL

Table 1: Differences between DRAC and UCLP

7 Summary and Future work

User Controlled Lightpath is a concept that arose to fulfill the need to dynamically create lightpaths. The creation of these lightpaths is done by users or applications. It is based on a service oriented architecture that enables the abstracted physical resources to be exposed as (web) services. Users and applications can use these web services to create their own dedicated network without the knowledge of the underlying technology.

Securing UCLPv2 is done primarily by the authentication and authorization of users. Improvements will have to be made to provide a safe UCLP environment.

Installing UCLPv2 on both Linux and Windows was without problems. However we did encounter some problems using UCLPv2.0. Most of these problems were fixed in the pre-release of UCLPv2.1[6]

Using the UCLP software with the SURFnet6 testbed turned out to be quite easy. The only problem encountered was a card that initially was not supported by the UCLP software. Thanks to the developers of i2CAT who included support for this card in a short time frame.

The open standards used in UCLPv2 offer great extensibility; however source code and API aren't available yet.

Another application that offers the user great flexibility in creating their own lightpaths is Nortel's Dynamic Resource Allocation Controller (DRAC). These two applications are very similar exept for the fact that UCLP is offering resources in a more abstracted way which reduces the complexity integrating UCLP in applications.

Future work would hopefully include:

- the release of the source code and the API.
- a pathfinding module, so that an user can create a lightpath between two endpoints without having to worry about the resources in between.
- VCAT support

8 Acknowledgements

First of all we want to thank Ronald van der Pol and Andree Toonk by guiding us this project. In such a short time frame we couldn't have managed it without them. Secondly a big thanks goes to SURFnet who provided the testbed. Also we would like to thank Eduard Grassa (i2CAT foundation) and Scott Campbell (CRC) for fixing bugs and adding hardware support on the fly, answering our questions and providing us with the pre-release UCLPv2.1. Furthermore we thank Bram Peeters and Victor Reijs to share their view on both DRAC and UCLP with us. It turned out to be very useful. And thanks to Paola Grosso and Paul Wielinga for their feedback on our pre-presentation. Last but not least we thank SARA who gave us the opportunity to do this project.

References

[1] Albert Lopez Sergi Figuerola Michel Savoie Eduard Grasa, Joaquim Recio. Uclpv2: A network virtualization framework built on web services. 2007.

[2] Greg Meredith Sanjiva Weerawarana Erik Christensen, Francisco Curbera. Web services description language, 2001. [Online; accessed 27-January-2007].

[6]Thanks to the UCLP developers of i2CAT who actually planned to release the software in the first half of 2007 we were able to test this new release.

[3] M. Nottingham M. Baker. Simple object access protocol, 2004. [Online; accessed 27-January-2007].

9 Appendix: Installing UCLPv2.0 on Debian GNU/Linux

This is a log of a successfull installation of UCLPv2.0 on Debian. I'm starting out with a clean Debian Sarge installation. This page is to be used alongside the installation instructions found at http://www.uclp.ca/uclpv2/documents/help/uclpv2.0.2/ d.d. 9 jan 2007. 10 jan. 2007, cmeertens os3 nl

*

Install Java

downloaded jdk-1_5_0_10-linux-i586.bin from http://java.sun.com/javase/downloads/index_jdk5.jspl (click 'Get the JDK download' somewhere at the top of the page – this is the linux self-extracting file, non-RPM, d.d. 9 jan 2007).*The jre would have been sufficient. I recommend the latter as it is smaller.*

installed it as a normal user from within /home/uclp/install-workdir/; got an helloworld working

export JAVA_HOME=/home/uclp/install_workdir/jdk1.5.0_10/

*

Install Apache Tomcat

downloaded http://apache.proserve.nl/tomcat/tomcat-5/v5.5.20/bin/ apache-tomcat-5.5.20.tar.gz and unpacked it

export CATALINA_HOME=/home/uclp/install_workdir/apache-tomcat-5.5.20/

*

Install Apache Axis

There is axis C++ and axis Java, took Java: http://archive.apache.org/dist/ws/axis/1_3/axis-bin-1_3.tar.gz

*

Install Axis in Tomcat

cp -Rp axis-1_3/webapps/axis/ apache-tomcat-5.5.20/webapps/

needed activation.jar

*

Install the UCLPv2 Web Services

downloaded http://www.uclp.ca/files/uclpv2/uclpv2.0.2_WS_2006_08_08.zip

install to /home/uclp/install_workdir/UCLP

it doesn't ask for start menu items

*

Create KeyStore for NE-WS

did not perform this step

*

BPEL Engine

downloaded http://www.activebpel.org/download/files/2.0/final/activebpel-2.0-bin.zip

ran install.sh

*

Install MySQL

downloaded static linux mysql 4.1 via http://dev.mysql.com/downloads/mysql/4.1.html and unpacked

following INSTALL-BINARY: scripts/mysql_install_db -user=uclp

configured a mysql root password

*

Create the database for the Active Endpoints 2.0 BPEL engine

mysql -u root -p <ActiveBPEL-MySQL.sql

*

Setup a Tomcat datasource

downloaded http://apache.essentkabel.com/tomcat/tomcat-5/v5.5.20/bin/ apache-tomcat-5.5.20-admin.tar.gz

tomcat startup.sh does not need explicit backgrounding; mysql_safe does.

Login using ‘admin’ with an empty password didn’t work. Searched the web for it and found this thread on a java.sun.com forum[7]. I edited ~/install_workdir/apache-tomcat-5.5.20/conf/tomcat-users.xml to give the ‘tomcat’ user extra rights:

```
<tomcat-users>
<role rolename="tomcat"/>
<role rolename="role1"/>
<role rolename="manager"/>
<role rolename="admin"/>
<user username="tomcat" password="tomcat" roles="tomcat,admin,manager"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
<user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
```

Note that the ‘tomcat’ user has a non-empty password. Login now succeeded.

The rest of this step succeeded without any modifications to the original installation instructions.

*

Configure BPEL Engine for Persistance

OK without modifications.

[7]See URL http://forum.java.sun.com/thread.jspa?threadID=596489

*

Configure BPEL Engine

I got a java nullpointer exception when updating. Restarted tomcat and succeeded (unstable?)

*

Installing UCLP BPEL services

OK without modifications.

*

Install a LDAP server and a LDAP browser

Openldap does not provide precompiled packages. Debian does, but we are trying to avoid all system wide installations. Got openldap 2.3.32 sources from openldap.org, configured for installation in ~/install_workdir/ldap/ and compiled+installed (configere –aprefix=...; make depend; make; make test; make install).

Did not copy slapd.conf from uclp, instead edited slapd.conf from install. Command slapadd gave an error (on format of email in init.ldiff?), but it did create database entries, so I decided to continue. *The decision to not copy slapd.conf was an error. You SHOULD copy it.*

libexec/slapd -d 1 -h ldap://127.0.0.1:9009/

*

GUI

Installed WinXP-pro and downloaded the GUI and its patch. Installed the GUI. Not the patch (not planning to use -testmode).

Installed JRE.

Login didn't succeed: admin not in LDAP DB. Created a separate admin.ldif.

Login didn't succeed: passwd is ok, was thrown back to login screen. Wrong LDAP version? *no – see below*

Created uclpv2.ini. Connect succeeded. Adding a user gives 'bad attribute type'.

Reconfiguring LDAP using uclp's slapd.conf makes it work.