On demand network and application provisioning through web services

Liu, Sandy; Liang, Yong; Xu, Bo; Zhang, L.; Spencer, Bruce; Brooks, Martin

Publisher's version / Version de l'éditeur:

The Proceedings of the IEEE International Conference on Web Services (ICWS 2007), 2007

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
https://nrc-publications.canada.ca/eng/view/object/?id=33f70a47-4a96-4afe-ab8d-ad7f74025b10
https://publications-cnrc.canada.ca/fra/voir/objet/?id=33f70a47-4a96-4afe-ab8d-ad7f74025b10

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

On Demand Network and Application Provisioning Through Web Services *

Liu, S., Liang, Y., Xu, B., Zhang, L., Spencer, B., Brooks, M.
July 2007

* published in The Proceedings of the IEEE International Conference on Web Services (ICWS 2007). Salt Lake City, Utah, USA. July 9-13, 2007. NRC 49343.

Canada

On Demand Network and Application Provisioning Through Web Services

Sandy Liu, Yong Liang, Bo Xu, Libo Zhang, Bruce Spencer, Martin Brooks
Institute for Information Technology, National Research Council Canada
{*FirstName.LastName*}*@nrc.gc.ca*

Abstract

While most service-oriented solutions are developed for system integration or data integration, we propose a Web Services-based solution, Eucalyptus, for provisioning applications and networks on demand. Eucalyptus is built on the power of the User-Controlled Lightpath Provisioning (UCLP) tool, which pioneers a user-centric and service-oriented approach for creating and managing a private end-to-end optical network. As a first user of UCLP, Eucalyptus aims to construct an participatory environment for geographically distributed teams of architects and industrial designers with the with the support of configurable broadband switched networks, as well as the traditional routed IP networks.

The contribution of Eucalyptus is to provide a proof-of-concept example on how Web Service and Service-oriented Architecture (SoA) can effectively provide on-demand provisioning for heterogenous resources in hybrid networks. These resources can be provisioned, launched, monitored, terminated, and reserved through Web Services. Eucalyptus is network and platform neutral. It offers a single point of entry for users to access resources ranging from video conference applications, rendering clusters, to the underlying networks. Each resource is configured through a resource-specific Web Service. Eucalyptus also includes a set of generic management Web Services to coordinate sessions, to manage resources and users, and to compose workflows, such that the network and the application are properly configured for the users engaging in a participatory design session.

1 Introduction

"Imagine a large scale urban redevelopment project involving a number of collaborative stakeholders such as architects, urban designers and planners, landscape architects, artists, lighting designers, engineers, heritage conservationists, stone masons, developers, financiers, city officials, and the general public. Due to the scale of the project these diverse parties are located around the world". They have to acquire digital content on and off site, create, manipulate, and deploy across networks in immersive environments, output to print, or visualize to displays"... It is a huge challenge to bring these geographically distributed expertise into effective participatory design sessions, in which a wide variety high-end resources, such as visualization clusters, rendering farms, and media repositories, are easily accessible. This scenario pinpoints the most sought- after elements in new media industry today: real-time, interactive 3D visualization, collaborative tools, and broadband technologies [6].

Eucalyptus moves a step forward in this direction by assembling a set a of heterogenous resources into a simple-to-use dashboard, allowing architects and industrial designers to access these resources without knowing the logistic complexities and the configuration details. For example, a user can launch an Isabel (a multipoint videoconference for PCs, http://www.agora-2000.com/) video-conference session from the Eucalyptus dashboard to talk to another team member, in the meantime, s/he can also engage in a co-design session using a variety of tools with a few clicks in the Eucalyptus dashboard. The provisioning of this set of devices, tools, and networks, which we collectively refer to as *resources*, is transparent to the end user.

Eucalyptus builds on the power of User-Controlled Lightpath Provisioning (UCLP) [11], a service-oriented solution allowing users to create and manage application-specific private end-to-end optical networks. Although network elements are typically considered as fixed physical entities, UCLP makes network resources as application-level Web Services, effectively releases the control of these network resources to the hands of the application users, making real time on demand network provisioning possible. Therefore, reconfiguring the network to accommodate a high-definition video-conference session is no longer impossible.

While UCLP offers a solution for network provisioning, resources connecting to the network also need to be coordinated and configured. As most architects are not IT experts, the motivation of Eucalyptus is to offer an end-to-end solution with a single point of entry for provisioning all the

resources required in a participatory design session. Eucalyptus, therefore, is a UCLP-enabled Participatory Design Studio (UCLP-PDS, a.k.a. Eucalyptus).

In the subsequent section, we briefly introduce some underlying concepts and technologies we based on. We then provide an overview of the system design, our demonstration experiments and follow this by the results, conclusion, on-going and future work.

2 Background

With the exponential growth of the Internet and the increased cost of routing, the layer 3 (IP network) sometimes cannot provide the required bandwidth and stability required by certain applications. Many e-science projects involve the usage of remote sensors and instruments generating very large volumes of data that need to be delivered and processed in far away facilities. Similar situation for architects and industrial designers, who need to share high quality multimedia files in real-time. This calls for high transport capacity networks.

A hybrid network provides a practical solution. It consists of both the traditional routed IP access (layer 3) to the Internet and circuit switched point-to-point connections (layer 2). These connections are often referred to as *lightpaths*. More specifically, a *lightpath* is an abstraction of a connection between two or more switches in an optical network, and typically connects two points on the network at speeds up to 10 gigabits per second. These lightpath connections offer a guaranteed Quality of Service (QoS) regarding bandwidth and latency.

To fulfill the demand for network bandwidth and QoS, advanced network organizations such as CANARIE (a non-profit organization who provides a national optical Internet research and education network in Canada. *http://www.canarie.ca/canet4/index.html*) have been investigating ways to provide application-oriented and user-controlled networks services. A resulting product is called the User-Controlled Lightpath Provisioning (UCLP) [5] tool. UCLP is a Web Services based solution for provisioning lightpaths. UCLP can be thought of as a configuration and partition manager that exposes each lightpath in a physical network and each network element associated with a lightpath as an "object" or "service" that can be put under the control of different network users to create their own logical IP network topologies [8]. The network users can then reconfigure and partition the lightpaths. This privately articulated end-to-end network is therefore called Articulated Private Network (APN). Within each APN, a number of network scenarios (i.e. logical topologies) can be specified to support different applications and usage scenarios. The APN Web Service can then be generated as a BPEL (Business Process Excution Language) [3] workflow linking together various network elements across multi-domain networks. When an APN BPEL workflow is deployed and published, the applications can set up the suitable network topology by invoking the APN Web Service.

However, these high-speed connections are not pervasive, often only a limited set of end-points are connected through lightpaths. To leverage the benefit of a hybrid network, we configure gateway computers that have access to both routed IP networks and the switched lightpath networks. We then deploy the set of management Web Services on these gateway computers. As all available resources are published and maintained through these management services, consequently users of Eucalyptus can conveniently look up resources via either a layer 2 or layer 3 connection, and provision the resources (including the underlying networks) through corresponding Web Services.

3 Overall System Design

All the core functions in Eucalyptus are provided by Web Services, either as a single service or a combination of services.

We divide the services into two groups: task-oriented services and management services. As the name implies, task-oriented services offer the capability to conduct a task, such as launching a video conference session. We use a set of generic management services to provide support and management for the task-oriented services. For instance, the Resource Management Web Service (WS) is responsible for managing all the resources that are made available through Eucalyptus.

Figure 1 illustrates the overall system design in Eucalyptus.

In Eucalyptus, we regard the network as a type of resources. As illustrated in Figure 1, we consider the APN setup Web Service as a task-oriented service. This service allows the Eucalyptus administrative user to set up different APNs, each with different configuration scenarios through the Web Services provided by UCLP. The Eucalyptus end-user can later invoke an APN setup service or switching from one scenario to another within the same APN.

We group the computers that make up our solution into three categories according to each role:

PDSF refers to **PDS Framework** computers. These computers have the Web Services platform installed and are generally used for exposing resources in Eucalyptus. A Web Services platform typically includes an HTTP server (*e.g. Apache*), a SOAP engine (*e.g. Axis*), and a Servlet container (*e.g. Tomcat*) that hosts the Web Services.

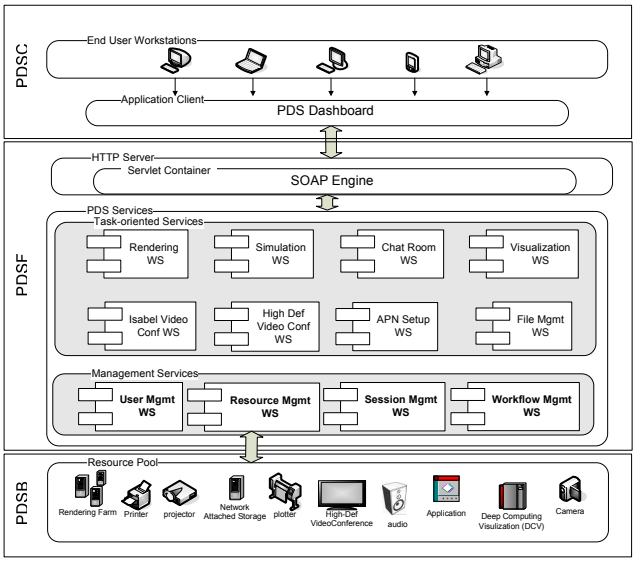

Figure 1. Eucalyptus System Overview

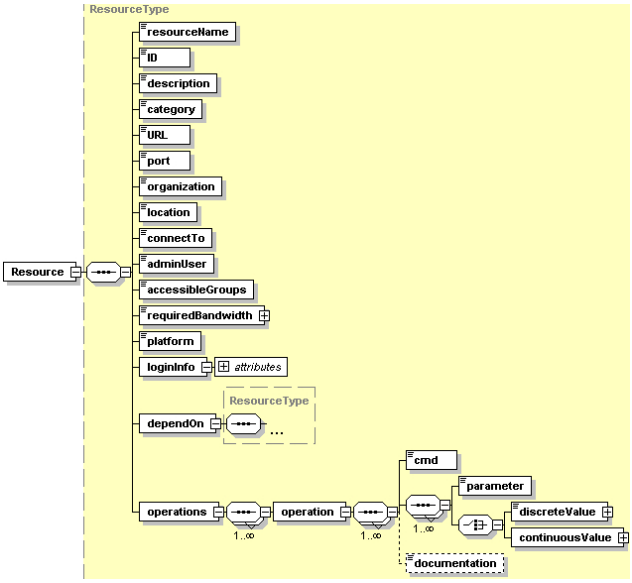

Figure 2. Resource Schema

PDSC refers to **PDS Client** computers. These computers have the *PDS Dashboard* installed and are typically used by end users. The *PDS Dashboard* will be explained in the following section. In some cases, a PDSC computer can make certain local resources (*e.g. VNC*) available to the people in the session, but these resources are not exposed as Web Services on the local machines; instead, the access parameters are posted via the resource management service. For example, a PDSC can host a VNC server, and publish its access information (e.g. IP address) through the resource management service. Generally there is one person using each PDSC computer.

PDSB refers to **PDS Backend** computers or devices such as the Deep Computing Visualization Server (an IBM BladeServer) and the Rendering Farm, which is a network of high performance computers devoted to rendering. They are accessed via the PDSF computers and do not communicate directly with PDSC computers.

Note that one computer can play multiple roles. For example, a computer can run the Eucalyptus dashboard (PDSC) while hosting some Web Services (PDSF) for a few resources.

3.1 Resource Wrapping and Task-oriented Services

As stated, we consider resources to be any software applications, devices that can be controlled through computer programs, network elements, and the network itself. To make all resources accessible through Web Services, we developed a generic approach for resource wrapping. Note that we are only concerned about provisioning of the resource, not the actual data communication among resources. The basic tasks include launching, shutting down, or checking status of resources. For example, to launch a multi-point video-conference application, we start the conference application with the proper parameters and configure the underlying network to make sure it can support the bandwidth requirement. The actual communication among different conference machines is handled by the native application. In order to generate the Web Service wrappers efficiently, we define each non-network resource with an XML description file. Figure 2 shows the schema for defining resources.

Each resource is assigned by a Resource ID and is described by a set of non-functional descriptions, including name, category (e.g. communication tools, visualization tools, etc.), physical location, URL, port, the admin user, the access restrictions (which user group has access to this resource), what platform is this resource is running on (i.e. Windows, Linux), the login information for accessing the machine, which router or switch it is connected to, the bandwidth requirement, and any resources it depends upon. In addition, the XML file also specifies the operations supported by this resource; each operation is described by a command and the corresponding parameters. With this information, we can use the resource wrapping utility to quickly generate the corresponding Web Service for each resource.

As a result, for adding a new instance of an existing type of resource, all we need is to deploy the Web Service of the same type to the machine that hosts that resource.

We categorize task-oriented services into different types according the functionalities of the associated resources: communication tools, visualizing tools, management tools and others, as shown in Figure 3. For example, the Isabel video-conference application is a kind of communication tool. This categorization allows us to retrieve available resources by type.

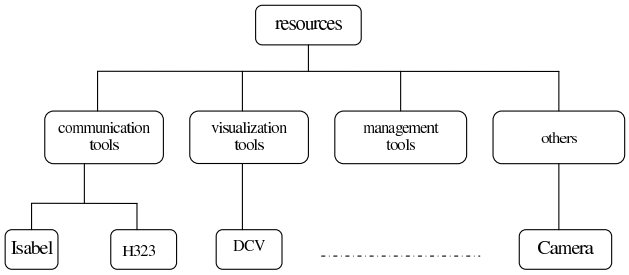

Figure 3. Resource Category Definition

The resources in each category have similar functions, which means every resource in the same category has similar operations to be exposed to the user. For example, in communication tools, all resources should have following functions: `startResource()`, `stopResource()`, `getStatus()`. Different resources have different input and output parameters in their functions. To make the generic Web Service interface extensible to all of the more specific resource type, we declare all the input parameters and the return type as `String`. We will use parameter information described in XML resource description file to parse the input and output strings properly.

3.2 Management Services

By wrapping the resources with our Resource Wrapper, we have the resources accessible for provisioning through Web Services. However, we also need a set of management Web Services to manage and coordinate the usage of the resource, such that we can use the Web Services as a control panel to all the available resources. This section will introduce the set of generic management Web Services. These management services form the basic building blocks of our infrastructure.

3.2.1 Resource Management Web Service

The Resource Management WS provides services to add new resources and modify the properties of existing resources such as changing the IP address of a resource. The Resource Management WS also acts as a registry, where one can look up available resources by its properties, and get the current status of different resources.

3.2.2 User Management Web Service

The User Management Web Service provides services to define user profiles, add new users, modify and delete existing users. It also keeps track of the contact information, login status, as well as the user groups.

Typically each resource specifies the access restriction. When a resource is being required, the Resource Management WS will contact the User Management Web Service to verify if the user has the proper permission to use that resource.

3.2.3 Session Management Web Service

A session is a collection of users and resources. A session starts when a user engages in using some resources. A user can potentially be involved in multiple sessions. For example, a user can participate in a session using a video-conference application with another two users while using a visualization resource for displaying some designs.

To reserve a session, a thread-safe check is made that each included resource is available to all included people. Then the session can be reserved. In more detail, a resource is available to a person if at least the following conditions and perhaps others are met:

1. The resource can be provided to that person via their own computer, or via another computer in the same room (location check is required);
2. The resource is permitted to be used by this user;
3. The resource is not otherwise allocated to that user through some other session, unless that resource can be part of two different sessions. Text messaging systems can be part of multiple sessions, as one can text messaging in more than one conversation at the same time, but two different video-conference sessions may not be hosted by one computer.

Note that the sessions themselves are not delivered through Web Services, but they are controlled through Web Services. Thus, our system does not deliver the collaborative application data such as video or audio streams. A user, however, can specify, invoke and manage these sessions.

3.2.4 Workflow Management Web Service

A workflow clearly defines the sequence of activities that must be performed in order to accomplish a certain task, and for each activity, it defines the preconditions required.

Figure 4. The Eucalyptus Dashboard

In the context of Eucalyptus, activities are performed by Web Services. The Web Service workflow can be defined by a workflow language such as WS-BPEL (previously called BPEL4WS [3]). Users can orchestrate a set of Web Services together to perform certain tasks. The workflow defined has to comply with the dependency relations associated with every resource participating in the workflow. We employ ActiveBPEL [1] as the runtime engine for workflows.

3.3 Customizable Integrated Service Client

To provide a single entry point to provision the resources, we present a dashboard to provide a graphical user interface for users to effectively use resources and provision participatory design sessions. One of the goals of Eucalyptus is to provide easy access remotely to some complex high-end resources that are typically not available in most labs. Thus the dashboard hides the complexities of configuring the resources required, providing access to those resources through a few clicks of buttons.

In Eucalyptus, we decided to develop the integrated client as a desktop application as opposed to a web application for several reasons: 1) Web applications have limited functionality on the client computer. There is no easy way to access the local file systems. 2) The implementations of the HTML, CSS, DOM and some other tools are browser specific and they often act inconsistently in different browsers. 3) It is less convenient to maintain an accurate reflection of status of all the resources with a web application since it does not have its own thread.

To maintain a desktop application over many computers is normally not an easy task. However, with the help of Java Web Start [10], the deployment and maintenance of Java desktop applications become easier. The advantages of Java Web Start include automatic application update, desktop integration, platform independence, Java runtime environment management, and security.

The dashboard interface is carefully designed to be unobtrusive and user-friendly. Inspired by DragThing [13], it is implemented as a floating dock, similar to the Mac OS X system dock. The dashboard (sometimes also referred to as the FloatingDock) only appears at the bottom of the desktop, and it can be anchored to any other edge of the desktop.

Each resource has its own button on the dashboard as shown in Figure 4. The user can also define his/her own often-used resources on the dashboard and can add them as new buttons. Existing executable application can be dragged and snapped into the dashboard, as a shortcut to launch that application.

3.4 Steps to Provision a Session

There are a few steps involved in setting up a session. The Web Service for each specific resource can only be invoked by the Resource Management WS. If someone requests an Isabel session, the request will be first handled by the Session Management WS. The Session Management WS will contact the Resource Management WS and the User Management WS prior to the invocation of the Isabel video-conference WS to ensure that only authenticated and authorized user have access to it. The example shown in Figure 5 demonstrates how to start an Isabel video-conference session consisting of several resources. This is a four-site video-conference where computer A is the conference host, while computers B, C and D work as clients to join the conference.

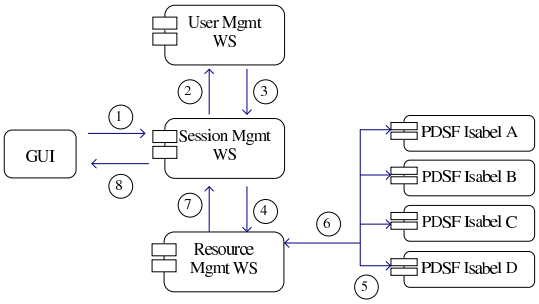

Figure 5. Steps to Launch an Isabel Session

(1) The user selects the participants' sites and provide the input parameters through the Dashboard GUI.

(2)(3) The Session Management WS asks User Management WS to get the user's access permission information.

(4)(5) The Session Management WS then makes the request to the Resource Management WS with the access permission just obtained from the User Management WS. If the user has the proper permission to access the resources, i.e. the Isabel computers, and they are available, then the corresponding Isabel Web Services will be invoked in parallel. We use multiple threads to perform the invocation to enhance the overall performance.

(6) Each Isabel computer returns its status indicating whether the resource is successfully reserved.

(7)(8) The session status is returned to the user.

3.5 Resource Status Monitoring

In order to correctly provision a resource, we need to properly detect its status. However, Web Services are typically stateless and passive. They only respond to service request initiated from the requestor, but not vice versa. One way to get the status of a particular resource is by creating Web Service that can return the status and having the client to poll such a service periodically. This approach introduces traffic to the service host and depending on the polling interval, it may not reflect the latest status of that resource. This is a common problem for Web Service projects.

In the current prototype, we introduce a lightweight TCP daemon, namely PDSDaemon, to detect changes and relate the changes back to clients who are interested in such changes. PDSDaemon is a TCP listener based on the MVC (Model-View-Controller) architecture. It listens to the client's request, forwards the request to management service and then sends the result back to all registered clients.

When a user logs in to Eucalyptus, the Floatingdock will send a connection request to the PDSDaemon. The PDSDaemon then creates a thread for the client who is requesting a connection. The resulting connection between the Floatingdock and the PDSDaemon will stay alive until the user logs out of the system. For each connected Floatingdock client, there is a separate thread that connects back to the daemon.

When the client (Floatingdock) established a connection with the PDSDaemon, at any time, it can make a request for using a resource. Figure 6 shows the interactions among the client, the daemon, the resource service, and the management services in a sequential diagram:

1. A Floatingdock client sends a message to the daemon, where the message includes the following information: the resourceID, the actionID (which operation is intended? e.g. start, stop, checkstatus...), and the content (a MessageObject).
2. The Daemon thread parses the message according to the resourceID and the actionID.

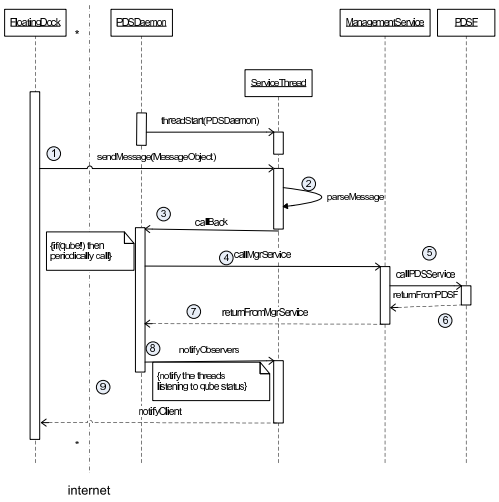

Figure 6. Status Monitoring in Eucalyptus

3. The Daemon thread forwards the message to the PDSDaemon.
4. The PDSDaemon then calls the Management WS.
5. The Management WS calls the WS for that specific resource (a PDSF service).
6. The PDSF service returns the result to the Management service.
7. The Management service then forwards the result to the PDSDaemon.
8. The PDSDaemon then notifies all the active service threads with the new updates.
9. Each service thread then notifies its corresponding client.

4 Experiments and Evaluations

We emphasize a spiral approach for developing Eucalyptus. Many design, development, testing, deployment (in actual labs) and feedback (from authentic users) cycles has carried out. Each cycle we revisit the previous prototype and revise it according to the user feedback. This proved to be an efficient approach. Now the school of architecture in Pennsylvania State University, and the Carleton University (Canada) are using Eucalyptus in their collaborative project. This involves about 30 students from both schools forming teams of four in co-designing an aviation museum.

We also introduce the system to audience from different backgrounds through various demonstration sessions. Figure 7 shows the set up of one of our demos.

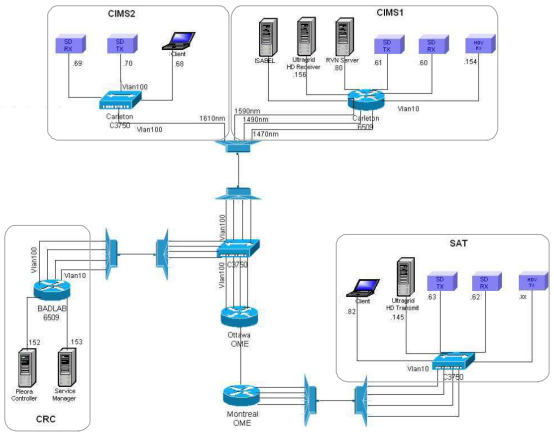

Figure 7. Detailed Network Configuration

Eucalyptus runs on the mixed layer 3 and 4Gb lightpath environment connecting Carleton Immersive Media Studio (depicted as CIMS1 in Figure 7), Society for Arts and Technology (SAT) in Montreal, and Communication Research Centre Canada (CRC). We deployed the management Web Services on a server (called Service Manager) at CRC, which is configured to be on both layer 2 and layer 3 networks. The first demo scenario shows the architects located at CIMS and SAT use the Eucalyptus interface to control visual communication and 3D modeling tools in their participatory design session. They use the IBM's Deep Visualization Computing (DCV) [4] cluster technology to provide hands-on direct collaboration using Maya. With DCV, high-end graphical images can be created in a visualization mode called Remote Visualization Networking(RVN), which distributes graphical images to remote (collaborative) clients called endstations. In our experiment, both architects were connected to the RVN server (running on Linux) at CIMS, while one architect works at CIMS and the other one sits at SAT. By clicking the corresponding buttons in Eucalyptus, it starts the RVN server with the proper configuration and launches the RVN client and the DCV enabled VNC in the meantime. With Eucalyptus, the architects do not need to know each single configuration parameters and deal with the Linux server or the system administrator before using this kind of tools.

While working on the design by sharing the desktop, the designers may want to see and hear each other for more discussions. Eucalyptus provides the access to some video-conference tools and they can be configured and launched automatically by Eucalyptus according to user's preference. For instance, one can choose the uncompressed Standard Definition (SD) video (480i) - DVD-quality streaming at 300Mb using Pleora technology. When graphically precise work details at a designer's desk need to be shared, Pleora HDV (720p) close-up view gives startling clarity. When larger-scale views must be shared at full resolution, for example when viewing street-level activity, UltraGrid [12] uncompressed HD (1080i) provides the highest resolution video. Eucalyptus not only do provide easy access to these conference tools, but also provide the corresponding network setup as a workflow for the underlying network connections.

In the second scenario, we created a 6000 mile loop from CIMS (back to CIMS2) to evaluate the system's latency in long haul connections and its capability in a global scale. This is ideal to test any system latency that may have caused by application, network, or layers of Web Services calls. In this scenario, users were able to manipulating a fully rendered massive 3D model of a Montreal neighborhood using a joystick in an open source immersive environment called OpenSceneGraph [2], over the RVN connection. Again Eucalyptus activates the APN connection scenario, as well as launching the set applications (e.g. OpenSceneGraph, RVN, VNC) accordingly.

Notably, Eucalyptus was able to dynamically switch between network scenarios defined by the Articulated Private Networks (APN) based on the application needs.

Collaborating designers also utilize the Eucalyptus interface to move data between tools. For example, a camera path through the Maya 3D model of Montreal's Blvd. St. Laurent district may be transformed to input to a rendering farm, for conversion to an HD-quality synthetic video. Moreover, the CD++ discrete event simulator [14] can receive Maya 3D model data in order to analyze emergency management plans in urban environments. All of these tools can be properly set up and run from the Eucalyptus dashboard.

5 Results and Conclusion

In conclusion, Web Service's component-based, web-oriented, standard-based, language, platform, and domain independent nature makes it an appropriate solution for many system and data integration projects. We consider a Service-oriented Architecture (SoA) implemented by Web Services are also a desired approach for provisioning resources spanning from networks to devices.

This approach allows Eucalyptus to quickly build a toolbox that consists of tools developed by different vendors with different execution environments with a uniform interface. It is flexible and extensible: new resources can be simply added by creating and publishing new services or they can be removed by removing them from the resource registry.

Although Web Services can be used in both data integration and provisioning, there are a few differences. Provisioning Web Services mostly interact with the control flow of the applications, while Web Services for data integration

mostly interact with the data flow, where data is wrapped in XML. Wrapping application data in XML is not appropriate in the broadband context, where the data streams are typically in the 1-1000 Mb range. In addition, the overhead of marking up data into XML-based SOAP message often hinders the number of Web Services in a service composition. On the other hand, provisioning parameters normally are very light-weight comparing to application data. Thus a provisioning Web Service workflow can comprise a number of nesting Web Services without affecting the efficiency of the system. Monitoring the status of applications and their underlying network is important to provide intelligent provisioning services such as comparing actual and desired states of an application, while this is not as critical in the data integration.

The Eucalyptus prototype is useful in assisting architects to do collaborative design in distributed labs [6] [7]. Although the current prototype is applied for architectural design, using Web Services in provisioning is indeed applicable to many industries. The value of Eucalyptus is also being recognized by users, tool vendors, system integrators and venture capitalists. CANARIE has already acted to establish Canadian leadership in the new application space of which Eucalyptus is the first example. St. Arnaud [9] believes that the agile infrastructure we created will have real potential and significant impact in many other fields and disciplines. As a result of our demonstrations, the Department of Public Safety in Province of New Brunswick (Canada) is interested in having us to apply our approach for responding to critical events; a few big companies are also interested in adopting the Eucalyptus platform for their industrial environments. In summary, this service-oriented approach can serve as a basic building block for agile low-cost enterprize system without investing in expensive enterprise solutions. The service-oriented approach adopted by Eucalyptus makes provisioning, running, and monitoring heterogeneous networks and network-enabled resources relatively easy and intuitive.

References

[1] ActiveBPEL, LLC. ActiveBPEL, the Open Source BPEL Engine. `http://www.activebpel.org`.

[2] O. Community. Openscenegraph.

[3] T. A. et al. Business Process Execution Language for Web Services version 1.1. "ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf", 2003.

[4] IBM. Deep computing visulization networking. `http://www-03.ibm.com/servers/deepcomputing/visualization/`.

[5] J. Wu et al. User-managed End-to-end Lightpath Provisioning over CA*Net4. In *Proceeding of the National Fiber Optic Engineers Conference (NFOEC), Orlando, FL, USA*, pages 275–282, September 2003.

[6] M. Jemtrud, M. Brooks, B. Ho, S. Liu, P. Nguyen, J. Spence, and B. Spencer. Eucalyptus: User controlled lightpath enabled participatory design studio. In *ACADIA(The Association for Computer-Aided Design in Architecture)International Conference 2006*, 10 2006.

[7] M. Jemtrud, P. Nguyen, B. Spencer, M. Brooks, S. Liu, Y. Liang, B. Xu, and L. Zhang. Eucalyptus: Intelligent Infrastructure Enabled Participatory Design Studio. In *WSC '06: Proceedings of the 37th conference on Winter simulation*, pages 2047–2054. Winter Simulation Conference, 2006.

[8] B. St.Arnaud. CA*net4 research program update - UCLP roadmap: Web Services workflow for connecting research instruments and sensors to networks. `http://www.canarie.ca`, December 2004.

[9] B. St.Arnaud. Cyber-infrastructure and grids for Architecture Collaborative Design. `http://lists.canarie.ca/pipermail/news/2006/000362.html`, 12 2006.

[10] Sun MicroSystems, Inc. Java Web Start Overview, White Paper. `http://java.sun.com/developer/technicalArticles/WebServices/JWS_2/JWS_White_Paper.pdf`, May 2005.

[11] The UCLP Development Team. User Controlled Lightpaths. `http://www.uclp.ca`, 2006.

[12] the UltraGrid Project team. UltraGrid: A High Definition Collaboratory. `http://ultragrid.east.isi.edu/`.

[13] J. Thomson. DragThing. `http://www.dragthing.com`.

[14] G. A. Wainer. CD++, A tool for and DEVS and Cell-DEVS Modelling and Simulation. `http://www.sce.carleton.ca/faculty/wainer/wbgraf/manuals/CD++.pdf`.

Acknowledgement

This project is funded by CANARIE's Intelligent Infrastructure Program. Our partners include The Carleton Immersive Media Studio (CIMS) at Carleton University, Communication Research Centre Canada (CRC), IBM, Pleora Technologies Inc., and AutoDesk.