Infraestructura para laboratorios de acceso remoto

Francisco López Luro, Leandro Bertogna, Laura Sánchez, Jorge Rodríguez, Rodolfo Del Castillo

Departamento de Ciencias de la Computación, Universidad Nacional del Comahue, Neuquén, Argentina

Resumen

Las tecnologías de Internet permiten el uso de sistemas de software distribuido para el acceso en forma remota a laboratorios físicos y virtuales, para llevar a cabo actividades de investigación experimental a distancia. A partir de los marcos teóricos que definen los estructurantes de los laboratorios, los implementadores de laboratorios de acceso remoto deben diseñar y desarrollar arquitecturas de servicios que permitan un acceso flexible y controlado. Este trabajo define una infraestructura para la implementación de aplicaciones para el acceso remoto a laboratorios físicos y virtuales y para la gestión de los mismos. Se presentan conceptos que permiten extender la modalidad tradicional de conexión con un dispositivo, para llevar adelante actividades prácticas de laboratorio colaborativas y concurrentes.

Palabras clave: Laboratorio remoto, framework, aprendizaje y trabajo colaborativo, software libre.

1. Introducción

Las tecnologías de Internet y el aumento de la velocidad de los medios de comunicación digital permiten el uso de sistemas de software distribuido para el acceso en forma remota a laboratorios virtuales o físicos, para llevar a cabo actividades de investigación experimental a distancia.

Un laboratorio de acceso remoto (LAR) es "un espacio de trabajo electrónico para la colaboración y experimentación en investigación u otras actividades creativas, para la generación y distribución de los resultados de investigación utilizando tecnologías de información distribuidas" [1].

Mediante los laboratorios de acceso remoto, recursos virtuales y físicos, escasos y dispersos, pueden ponerse a disposición de mayor cantidad de investigadores y estudiantes.

A partir de los marcos teóricos que definen los estructurantes de los laboratorios, los implementadores de LAR deben diseñar y desarrollar arquitecturas de servicios que permitan el acceso a sus recursos de manera flexible y controlada. Las tecnologías de comunicación para el acceso remoto a los laboratorios deben ser variadas e intercambiables, pero siempre respetando la semántica de la información que se transmite desde y hacia los recursos. De esta manera se logra un nivel de independencia entre la información que generan o consumen los recursos y el medio utilizado para ello, que permite agregar nuevos recursos a los LAR sin modificar sus arquitecturas de servicios actuales, sino extendiendo y adaptándose a las mejores tecnologías de comunicación.

Por su parte, las aplicaciones que acceden a los recursos de los LAR deben poder adaptarse a nuevos servicios y a los cambios tecnológicos de servicios existentes. Esto se logra construyendo aplicaciones modulares y extensibles, sin limitar el diseño a las tecnologías de comunicación o a los métodos de acceso utilizados para conectarse a los recursos.

Un Framework Orientado a Objetos (FOO) es un diseño reusable de una aplicación o subsistema, representado por un conjunto de clases abstractas, y la forma en que estas clases colaboran [2, 3]. Este diseño representa una aplicación semi-completa para una clase específica de software, que puede ser especializada para producir aplicaciones a medida.

Este trabajo describe una infraestructura para la implementación de aplicaciones para el acceso remoto a laboratorios y para la gestión de los mismos. Para la construcción de aplicaciones de usuario, se diseñó un FOO basado en Java [4] que permite la construcción de aplicaciones altamente adaptables a los cambios tecnológicos y desacopladas de las tecnologías de comunicación utilizadas para el acceso, obteniendo herramientas reusables para el acceso a distintos laboratorios, sin importar las tecnologías por las cuales se acceda, además de facilitar la adición y el acceso a nuevos servicios.

Para los servidores de laboratorios, se diseñó una arquitectura de servicios que provee a múltiples usuarios distantes entre sí, el acceso concurrente, colaborativo y gestionado dinámicamente, a recursos físicos y virtuales provistos por una o más organizaciones.

En la sección siguiente se presenta la arquitectura general. En la sección 3 se presentan los conceptos estructurantes y la definición de un recurso. En la sección 4 se presenta el Framework para la creación de aplicaciones Cliente para acceder a recursos remotos. La sección 5 describe las mejoras implementadas en la arquitectura del Servidor de Recursos. La sección 6 presenta un prototipo funcional de la infraestructura.

2. Arquitectura general

Los aportes que se encuentran en este trabajo están insertos en la arquitectura de la figura 1.

Los componentes de dicha arquitectura comprenden:

El "Servidor de labs", que funciona como un portal web para los clientes (alumnos, docentes y administradores), permitiendo que estos inicien una sesión segura, accedan a contenidos teóricos varios, planifiquen actividades prácticas e inicien su participación en un LAR. El "Servidor de Recursos (SR)", que provee acceso a los recursos físicos y virtuales en una o más organizaciones, es decir, está conectado físicamente a los recursos. El "Servidor de Aplicación de Políticas" es el encargado de hacer efectivas las diferentes políticas sobre los SR para permitir que cada cliente luego de ser autenticado pueda acceder a los distintos recursos del laboratorio con los permisos que corresponda. El Servidor LDAP actúa como base de datos de autenticación para alumnos, docentes y administradores para acceder al portal y por lo tanto a los laboratorios.

Se puede encontrar una visión más detallada en [5, 6], donde se presentan aspectos y mejoras del diseño de aplicaciones Cliente que acceden al laboratorio, y agregados de funcionalidad a los servicios del LAR para dar soporte a actividades de e-learning colaborativas .

En particular, este trabajo se enfoca en los Clientes de laboratorios y en el concepto y la operación de lo que llamamos Distribuidor y Manejador de Conexiones (DMC) y Concentrador de Conexiones (CDC). Estos últimos, proveen a los Servidores de laboratorios la funcionalidad necesaria para poder gestionar distintos laboratorios al mismo tiempo, permitir el acceso a grupos de alumnos por laboratorio y conectarse para trabajar de manera concurrente y colaborativa, permitiendo opcionalmente, la presencia de un facilitador que altere la dinámica de trabajo durante la práctica en forma remota.

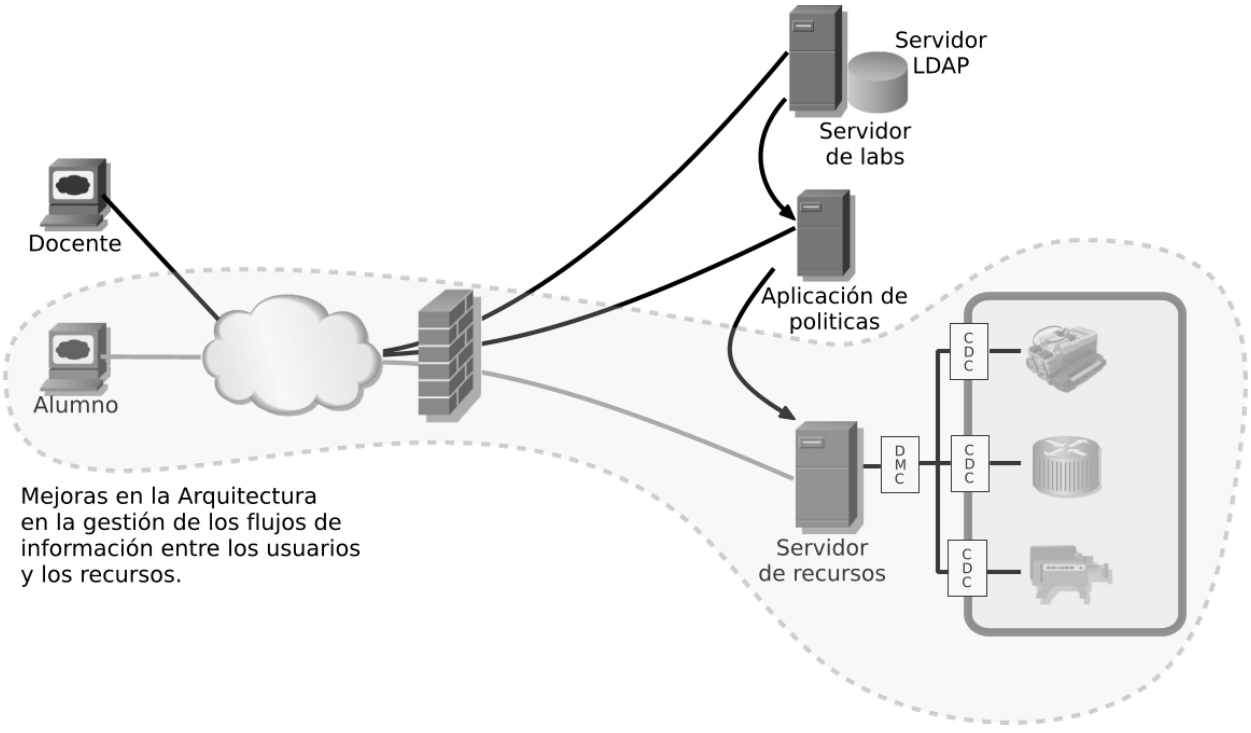


Figura 1: Arquitectura general

3. Recursos remotos físicos y virtuales

Utilizaremos el término recurso en su sentido más amplio, como una entidad real o virtual que puede producir información de cualquier naturaleza en función de un requerimiento de entrada y que tiene un estado asociado. Estos recursos pueden ser estáticos o dinámicos y su acceso puede realizarse en forma sincrónica o asincrónica.

Ejemplos de estos recursos pueden ser páginas web, sensores físicos de cualquier naturaleza, dispositivos que generan flujos multimedia, un generador de números aleatorios de alta precisión, una máquina virtual de pc, etc.

Existen muchos recursos hoy en la Internet que son gestionados y accedidos de manera eficiente. Muchas de las tecnologías y herramientas de groupware para el acceso a dichos recursos han demostrado su efectividad y no necesitan ser reemplazadas en el corto plazo. El protocolo de aplicación HTTP por ejemplo, implementado por los navegadores web y los servidores web, es utilizado masivamente para permitir el acceso a contenido Hipertextual (HTML) de forma simple y efectiva.

Este trabajo complementa dichas tecnologías y aplicaciones instauradas en la comunidad de usuarios de Internet, permitiendo el acceso a recursos de variada naturaleza, a través de interfaces bien definidas, para ser utilizados en actividades de investigación y educación a distancia. Ejemplos de estos recursos pueden ser "dispositivos de hardware especial", "interpretes de comandos de texto", "microscopios electrónicos", "robots", etc. Estos recursos a veces escasos o no disponibles en todos los laboratorios generan flujos de información, los consumen como datos de entrada, o ambas cosas.

Se define entonces para este trabajo el concepto de recurso como una entidad que puede ser Productora (Productor) de información, Consumidora (Consumidor) de información, o ambas. Además, un recurso dispone de cero o más operaciones que modifican su estado y funcionamiento, que deben poder ser ejecutadas sobre este a distancia y en un ambiente controlado y seguro.

Las posibilidades de tener acceso a un recurso en particular dependen de la existencia de una interfaz de entrada/salida bien documentada y del ancho de banda necesario para transmitir los datos desde el recurso hacia el usuario y viceversa, que puede variar desde un simple flujo de caracteres hasta audio y video de alta calidad.

El agrupamiento lógico de los recursos dentro de un "laboratorio de acceso remoto" facilita la administración y disposición para las distintas actividades que quiera llevar adelante una institución.

4. Framework para la construcción de clientes de LAR

Las aplicaciones que permiten el acceso a recursos dentro de LARs, sigue claramente un modelo Cliente-Servidor [7], en el cual el proveedor de los servicios, quien crea y dispone los recursos reales o virtuales en sus laboratorios, implementa dichos servicios mediante alguna tecnología de sistemas distribuidos que dé soporte al acceso remoto a sus recursos.

Ejemplos de estas tecnologías de comunicación pueden ser RMI, CORBA, Remote Procedure Calls (RPC), Web Services, el protocolo HTTP, o un socket TCP entre dos pares. Cada una de ellas, tiene sus ventajas y desventajas en el momento de implementar un servicio, publicarlo y permitir el acceso.

Por ejemplo, para el acceso a un intérprete de texto los Web Services no proveen el grado de interacción suficiente para generar la telepresencia deseada en un recurso con tal interfaz

Ante la gran diversidad de recursos y tecnologías de comunicación existentes, se plantea la necesidad de construir aplicaciones Cliente que tengan previsto desde su diseño, la posibilidad de intercambiar fácilmente la tecnología de comunicación utilizada para acceder a los recursos.

4.1. Diseño del Framework

En el contexto del diseño de software, un framework puede ser visto como la materialización concreta de familias de patrones de diseño [8] para resolver problemas de un dominio de aplicación en particular, mientras que por otro lado, los patrones pueden ser vistos como elementos abstractos microarquitecturales de los frameworks que documentan y motivan la semántica del framework de una manera efectiva.

El lenguaje utilizado para la implementación del Framework es Java [9]. Esta decisión está basada fundamentalmente en que Java es un lenguaje orientado a objetos, multiplataforma y permite la carga de clases de manera dinámica, facilitando el desacoplamiento de la lógica de la aplicación, la interfaz de usuario y la tecnología de comunicación para acceder a un recurso específico. El uso de la tecnología de Applets Java simplifica las tareas de distribución y actualización del software [10].

La figura 2 muestra el diagrama de clases en UML del framework diseñado

.

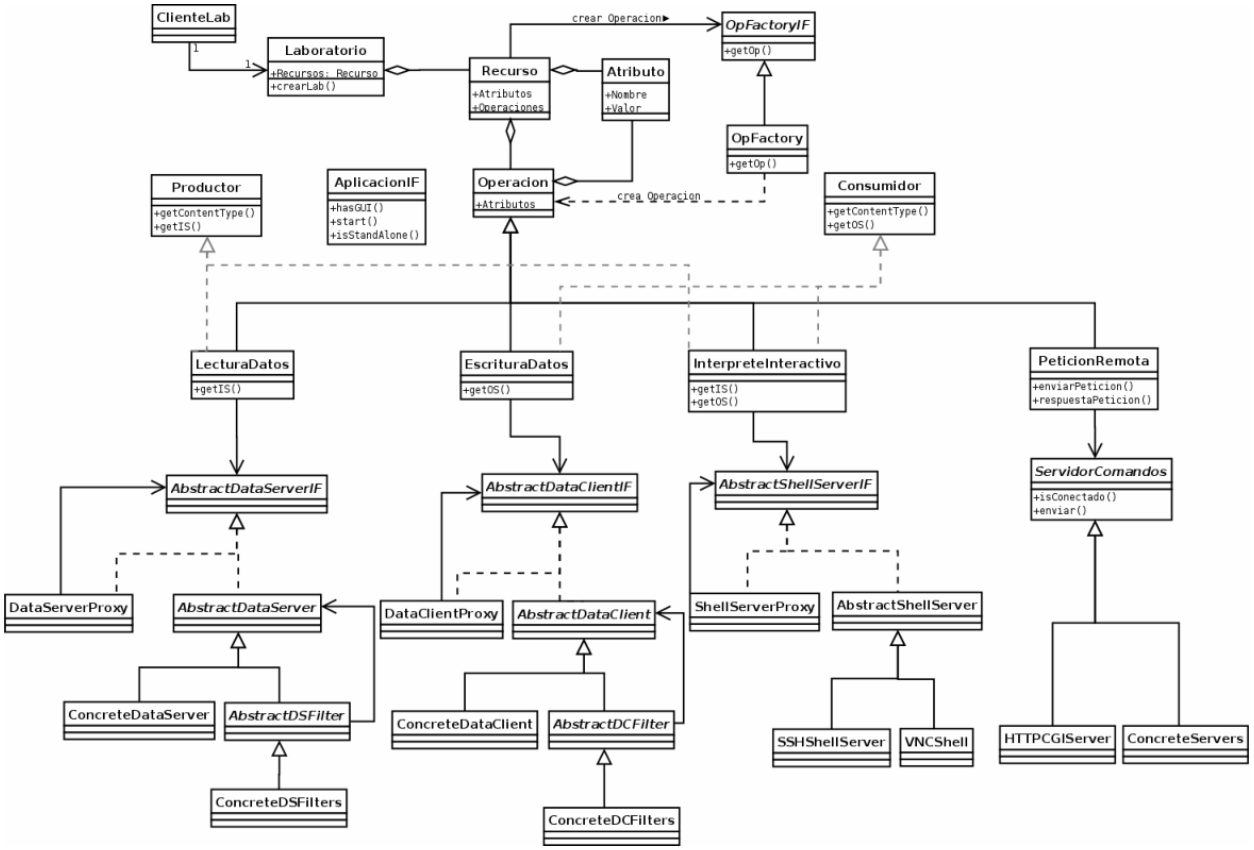


Figura 2: Diagrama de clases

4.2. Descripción general del diagrama de clases del Framework

El usuario del framework, representado en el diagrama por la clase *Cliente*, es la aplicación final que provee la lógica de aplicación y la interfaz de usuario (UI). Esta clase *Cliente* usará los servicios que proveen las clases *Laboratorio, Recurso y Operacion*, y las interfaces *Productor*, *Consumidor* para poder acceder a los recursos remotos de manera transparente.

La clase *Laboratorio* es una colección de Recursos, y es el punto de entrada al framework desde el punto de vista de una aplicación final. Desde la clase *Laboratorio*, la clase *Cliente* puede acceder a los distintos recursos que componen un LAR. Cada clase *Recurso* consiste de una colección de *Atributos* y de *Operaciones*, donde los atributos son simplemente pares (nombre,valor) que permiten enumerar cualquier tipo y cantidad de atributos para un recurso, mientras que las operaciones pueden ser una de las cuatro subclases de la figura 2, *LecturaDatos, EscrituraDatos, InterpreteInteractivo* o *PeticionRemota.*

Existen tres roles, representados como interfaces Java en el diagrama de clases, que caracterizan el comportamiento de las distintas operaciones. La interfaz *Productor* será implementada por aquellas operaciones que necesiten leer datos desde un *Recurso*, análogamente la interfaz *Consumidor* será implementada por aquellas operaciones que envíen datos a un *Recurso*.

Asociadas a cada subclase de *Operacion*, están las interfaces de implementación. Dichas interfaces desacoplan totalmente las implementaciones concretas con las cuales se accede al recurso remoto. De esta manera se pueden reemplazar las implementaciones respetando las interfaces sin la necesidad de modificar la aplicación final e incluso en tiempo de ejecución de la misma.

Para poder acceder a un determinado LAR se necesita conocer la información sobre los recursos que este ofrece y las tecnologías necesarias para acceder a dichos recursos. Por ejemplo, un laboratorio puede ofrecer un recurso "sensor" que representa lecturas de un proceso en un laboratorio real y permite el acceso a dichas lecturas mediante un simple socket TCP. El Cliente del framework accederá a la información que genere dicho sensor mediante una operación de lectura de datos sin

conocer en concreto que la lectura esta siendo realizada con una determinada tecnología (clase *LecturaDatos*).

Según sea necesario, se irán agregando implementaciones concretas, que luego podrán ser utilizadas por cualquier diseñador de una aplicación Cliente que tenga acceso a las clases compiladas o el código fuente. Técnicamente, estas implementaciones se denominan hot-spots o puntos de extensión en el Framework.

Se debe notar que las implementaciones que permiten comunicarse mediante algún tipo de tecnología no son parte del framework, son Clases independientes que implementan las interfaces correctas del framework para poder ser usadas dentro del mismo. Estas pueden ser cargadas en forma dinámica durante la ejecución del framework, lo que agrega flexibilidad y capacidad de adaptación de la aplicación en tiempo de ejecución.

Como se mencionó, es necesario conocer con qué tecnología se implementó el servicio para acceder a la información que genera un recurso determinado. Cada tecnología necesita distintos parámetros para su correcta configuración, el enfoque que se adoptó es el de disponer de una descripción genérica y flexible, en formato XML, del recurso y sus operaciones, permitiendo abarcar las descripciones de tecnologías actuales así como de tecnologías nuevas o aún desconocidas.

La descripción genérica de un recurso en XML se corresponde con las clases *Recurso*, *Operacion* y *Atributo* del diagrama y tiene la siguiente forma:

```
<recurso>
   <atributos>
      <atributo> <nombre>...</nombre> <valor>...</valor> </atributo>
   <atributos>
   <operaciones>
       <operacion>
          <atributos>
             <atributo> <nombre>...</nombre> <valor>...</valor> </atributo>
             </atributos>
       </operacion>
   </operaciones>
</recurso>
```

El Cliente del framework debe proveer una configuración en XML que describa el laboratorio, para que las clases del framework se puedan instanciar correctamente. Toda la información necesaria para describir completamente cada recurso y construir sus operaciones debe estar en este XML. Por ejemplo, si la comunicación con un recurso determinado se realizará usando un socket TCP, se debe conocer la dirección IP y el puerto en el que se puede establecer la conexión.

Básicamente, solo la implementación concreta posee la información de qué argumentos necesita para ser inicializada correctamente, entonces el Framework durante su construcción, consulta a dichas clases por los atributos necesarios, los extrae del XML y se los pasa a través de un método de inicialización común. Con esta técnica se elimina la necesidad del uso de introspección para la creación de clases desconocidas al momento de compilación. Si faltara alguno de los atributos en el XML se produce un error fatal en el Framework dado que la implementación no podrá ser inicializada correctamente.

La interfaz *AplicacionIF* tiene un propósito especial en el diseño, y es el de poder reusar una aplicación final hecha en Java, como puede ser una terminal SSH o un Cliente VNC de escritorio remoto. Mediante el uso de esta interfaz, el programador puede consultar a una determinada operación de un recurso si esta provee una aplicación final (con GUI propia) independiente y lanzarla en caso afirmativo con los argumentos que necesite. Este enfoque permite adaptar fácilmente aplicaciones existentes a los LARs y reusarlas de manera transparente dentro del framework.

4.3. Patrones de diseño

A continuación se describen los patrones presentes en el diagrama de clases:

- Interface [11]: El patrón interface permite mantener una clase que hace uso de los datos y los servicios de instancias de otras clases independiente de dichas clases mediante el acceso a estas a través de una interfaz [12]. El lenguaje Java dispone de un tipo de dato específico para este patrón, denominado justamente *Interface* y hace exactamente lo que define el patrón de mismo nombre. Mediante la aplicación de este patrón, se logra desacoplar el diseño genérico del framework de las implementaciones concretas que acceden a los recursos reales. Las clases *AbstractDataServerIF*, *AbstractDataClientIF* y *AbstractShellServer* entre otras, realizan este patrón.
- Filter [13]: El patrón filter permite que objetos que realizan diferentes transformaciones y cálculos sobre flujos de datos sean conectados en forma dinámica para componer transformaciones mas complejas. El único requerimiento para estos objetos transformadores es que implementen una interfaz común. El concepto es similar a las tuberías de Unix que procesan flujos de bytes o caracteres en forma separada como filtros. Las clases concretas de filtros pueden ser compuestas en forma dinámica sin necesidad de modificar la aplicación Cliente, con lo cual la aplicación puede adaptarse a situaciones nuevas como por ejemplo que los datos de la red vengan en un formato comprimido. Las clases centrales de este patrón en el diagrama son *AbstractDSFilter* y *AbstractDCFilter*.
- Virtual Proxy [13]: Si la instanciación de un objeto es costosa, y quizás el objeto no sea utilizado, puede ser provechoso retrasar dicha instanciación hasta el momento preciso en que se requiera de los servicios del mismo. Al disponer de una interfaz igual a la del

objeto en cuestión, que simplemente tenga una referencia al objeto y delegue las operaciones requeridas por los clientes en el instante que sucedan, se puede agregar un nivel de indirección en los requerimientos. De esta manera, los clientes no pueden saber en que momento es construido el objeto real que implementa el servicio. A esta técnica también se la denomina "Lazy Instantiation" (instanciación retrasada).Si la aplicación basada en el framework se debe descargar de la red, como es el caso del LAR, la descarga será muy rápida puesto que solo se transmite el esqueleto del framework y la aplicación cliente, dejando las implementaciones concretas para descargar cuando se realicen las solicitudes de servicio por el usuario final. Las clases *DataServerProxy, DataClientProxy, ShellServerProxy* implementan en parte este patrón.

5. Acceso controlado y concurrente a los recursos del laboratorio

5.1. Distribuidor y Manejador de conexiones (DMC)

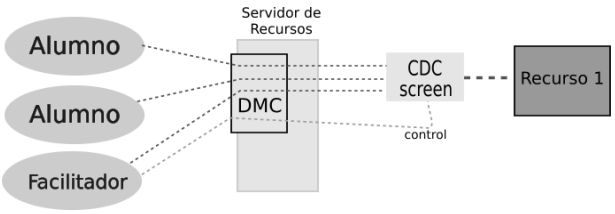
Desde el punto de vista de la institución o empresa, sus recursos deben poder ser accedidos de manera coordinada y controlada, para cumplir con esta restricción, definimos el concepto de "Distribuidor y Manejador de conexiones" (DMC), que es un componente de software que se ejecuta como un servicio de red en el SR y su principal objetivo es controlar el acceso de los clientes a los distintos recursos disponibles. Pueden existir una o más instancias de DMCs en cada SR según sea necesario acceder a distintas clases de recursos.

Como se mencionó anteriormente, los recursos están englobados lógicamente en una organización por laboratorio, por lo tanto, un usuario que quiera acceder a un recurso en particular debe poder hacerlo también a su respectivo laboratorio. Se optó por un modelo de control de acceso a los recursos basado en roles (RBAC) para gestionar el laboratorio, implementado a través de un servidor LDAP. A la hora de acceder al laboratorio, cada usuario tendrá uno o más roles asignados, creados al momento de la construcción del laboratorio específico, que le permitirán realizar determinadas acciones sobre cada uno de los recursos disponibles en el laboratorio.

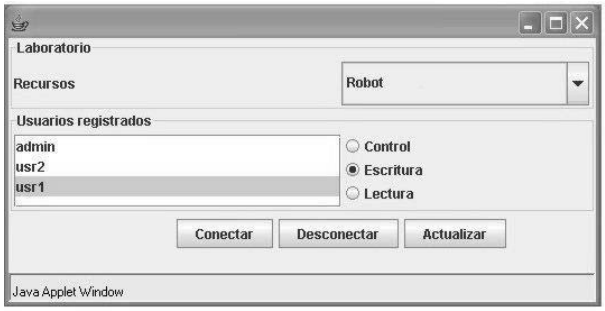
Cuando un usuario se autentica en el sistema, a través de la interfaz web en el Servidor de Laboratorios, y escoge el laboratorio práctico que va a realizar, el Servidor de Aplicación de Políticas, con la información del usuario y el laboratorio correspondiente, configura los DMC en los Servidores de Recursos afectados para que el usuario pueda acceder. Luego que se han configurado los DMC, queda disponible un archivo XML con las descripciones de los recursos, sus atributos y operaciones, que será usado por la aplicación Cliente del usuario final para configurarse dinámicamente y acceder al laboratorio.

5.2. Concentrador de Conexiones (CDC)

Para poder llevar adelante las prácticas remotas satisfaciendo las características de un entorno colaborativo, se necesita extender el esquema de conexión básico con un recurso, para permitir compartir de manera controlada la información que entra y sale de estos, concentrando todas las conexiones entrantes hacia el recurso y replicando la salida del recurso hacia las distintas conexiones (clientes). Este componente de software lo denominamos Concentrador de Conexiones (CDC), cada clase o tipo de recurso tendrá su propio CDC y su complejidad variará según la naturaleza del recurso y el control que se quiera ejercer sobre los flujos de datos. Un esquema conceptual de la función de los componentes DMC y CDC se puede ver en la figura 3 se muestran conceptualmente la función del DMC y el CDC relacionados a un recurso y un conjunto de usuarios.



Esquema de DMC y CDC en un Servidor de Recursos



Front-End para gestión de CDC de caracteres

Figura 3

Por ejemplo, en la clase de recursos orientados a caracteres como consolas de texto, donde todo lo que entra y sale de un dispositivo son caracteres, se optó por implementar un CDC utilizando la herramienta de software"GNU Screen" [14]. Este concentrador permite conectar fácilmente las entradas y salidas estándar de

varios procesos a uno en particular, que en nuestro caso es un proceso asociado al recurso en cuestión. Además, se puede controlar el flujo de caracteres de cada una de las conexiones entrantes, limitando una conexión por ejemplo a "solo lectura". Entonces el Cliente conectado con "solo lectura" es un simple observador de lo que está entrando al recurso. Esto es útil por ejemplo, para que durante una práctica remota un solo usuario a la vez pueda enviar información hacia el recurso, mientras el resto observa en el mismo instante los cambios.

Debido a que los CDC pueden ser bastante complejos de administrar en tiempo real, se han desarrollado varias herramientas para hacer más amigable su configuración y uso, para aislar al facilitador o docente de la complejidad del concentrador en sí mismo y solamente presentar una interfaz conceptual de las operaciones sobre cada uno de los roles (alumnos) y recursos asociados.

En particular, para el concentrador de conexiones orientadas a caracteres, se ha desarrollado una herramienta denominada "screenController" en lenguaje Python que se comunica con el programa "screen" para modificar en tiempo de ejecución el estado de las conexiones con un recurso determinado. Además se desarrolló un Applet Java (front-end) para interactuar visualmente con el "screenController" (parte derecha de la figura [fig:Concentrador-de-conexiones]) y permitir al docente controlar y modificar remotamente el estado del CDC desde el cliente de LARs.

6. Experiencia de laboratorio de robots

Durante la demostración del prototipo de LAR de programación de robots realizada en TEyET'06 desde la Universidad Nacional del Comahue, en el marco del proyecto "Software para Aprendizaje y Trabajo Colaborativos" [6], un grupo de alumnos participó a distancia y de manera colaborativa en una clase de programación a distancia.

El laboratorio consiste de una etapa teórica y una práctica. El docente realiza una presentación introductoria mediante transparencias a través de la web para explicar el funcionamiento básico del robot y la forma de trabajo. A continuación, los alumnos concretan la práctica de programación propuesta usando un cliente SSH para conectarse a una terminal remota en el SR.

En el SR, podrán programar el robot y ejecutar dicho programa para ver los resultados concretos en un robot real a través de una cámara web vía HTTP. Para lograr una experiencia colaborativa que permita a todos los integrantes poder hacer un aporte en el resultado final, y haciendo uso del CDC desarrollado para consolas de textos, los alumnos se conectan a la misma consola de texto, por la cual todos ven una única imagen del estado del recurso, y el instructor define de manera dinámica quien puede escribir en dicha consola en un instante determinado. Durante el ejercicio de programación el/los alumnos reciben la asistencia del docente y de sus compañeros para resolver el problema planteado, permitiendo que cualquiera pueda continuar la tarea del compañero o detectar algún error tempranamente.

Se construyeron dos DMCs y un CDC para implementar el prototipo mencionado. Para la gestión y el acceso de consolas de texto, por las cuales los alumnos pueden programar el robot, se desarrolló un DMC en Python, que se ejecuta inmediatamente después de que los usuarios acceden al SR a través del protocolo SSH. Este DMC controla que el usuario que está accediendo tenga los permisos correctos en el laboratorio que quiere utilizar y que el laboratorio efectivamente esté ejecutándose. Si estas condiciones se cumplen, el DMC "conecta" al usuario al CDC correspondiente al recurso.

En la figura 4 se presenta una instancia completa de la arquitectura para el prototipo mencionado, a la izquierda de la nube se encuentran los clientes, basados en el Framework. Para los alumnos simplemente se necesitan terminales SSH para realizar la tarea de programación, mientras que el docente dispone también del front-end del ScreenController para poder controlar la interacción de los alumnos durante la práctica remota.

Del lado del servidor, a la derecha de la nube, los clientes se conectan todos al CDC asociado al recurso, y luego, cuando los alumnos finalizan la tarea asignada pueden "subir" el programa creado al Robot Físico a través de una torre infrarroja Lego que esta conectada al SR y tiene al Robot dentro de su alcance.

El otro DMC desarrollado, similar al sistema WebCast que ofrece OpenOffice [15] para exportar transparencias, sirve para que un docente pueda presentar contenidos teóricos a los participantes del laboratorio, como si fuera una presentación clásica de transparencias en clase. Este DMC se construyó usando el Servidor HTTP Apache [16], y el recurso en concreto es un programa CGI. Básicamente, en el programa CGI se mantiene el estado de la página que está viendo el "instructor", y los clientes (alumnos) mediante una consulta HTTP obtienen la URL de la siguiente transparencia. En el Cliente, se ejecuta un script que consulta de manera iterativa la dirección URL asociada al programa CGI para actualizar a la transparencia que el instructor desee. En este caso, el recurso es un simple programa que contiene un estado, que puede ser consultado por los clientes (Productor de información hacia el/los clientes) y puede ser modificado por el docente o instructor (Consumidor de información desde un cliente). Esta herramienta para la presentación de contenido teórico sincronizado, y de bajo consumo de recursos de red, junto con el uso de alguna herramienta de audioconferencia como Skype [17] resulta en una solución aceptable para el dictado de clases a distancia

para 3 o 4 grupos de alumnos ubicados en distintas instituciones.

En el caso de que el uso de audioconferencia sea prohibitivo por el ancho de banda necesario, los alumnos disponen de una aplicación de Chat en Java, que ejecuta también el en SR.

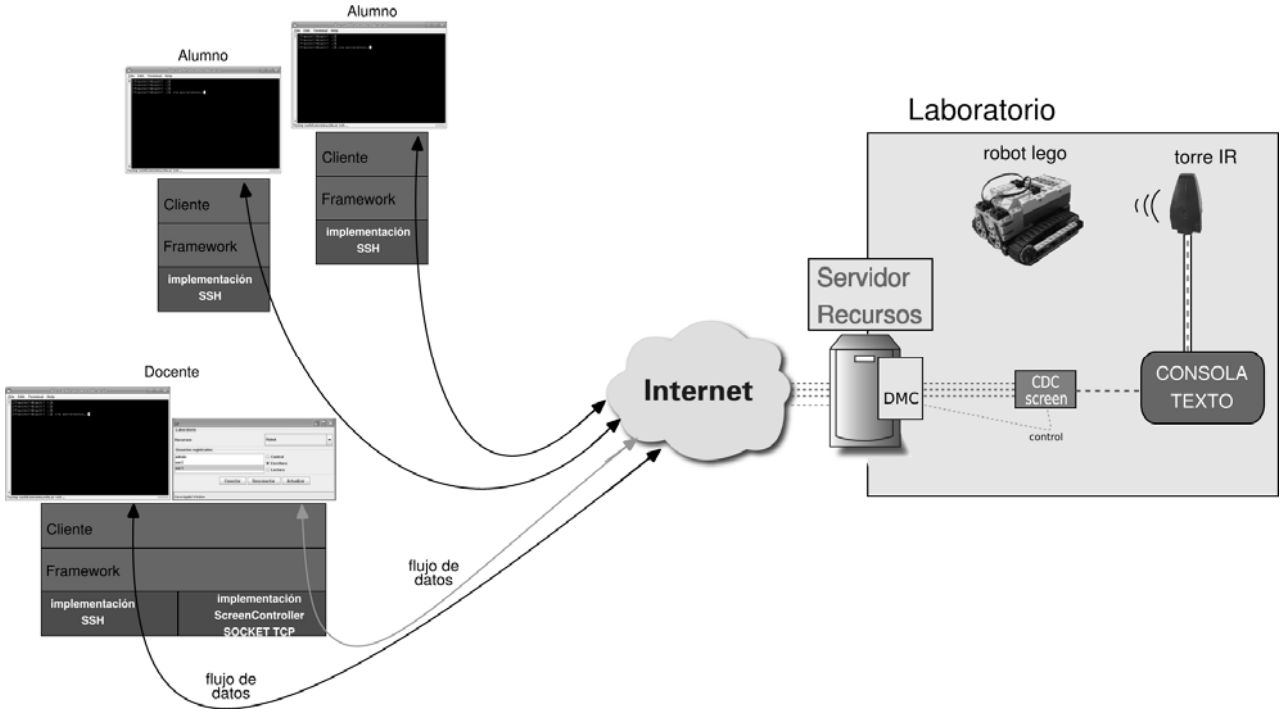


Figura 4 - Instancia de la arquitectura para LAR de Programación

Conclusiones y trabajo futuro

El uso de un FOO para la construcción de aplicaciones que acceden a laboratorios físicos y virtuales de manera flexible e independiente, junto con la definición e implementación de entidades de control y mejora del acceso tradicional a los recursos en los servidores, permiten que la instauración de los laboratorios de acceso remoto para actividades de investigación sea cada vez más fácil de implementar. El desarrollo de nuevos prototipos de LARs como el expuesto, que logren un nivel de madurez en el software permitirá generar paquetes de software instalables, que además de estar disponibles para las instituciones también puedan ser utilizados por los usuarios finales de manera completamente local, simulando en su computador los recursos y evitando la necesidad de disponer de una conexión a Internet o de la presencia de otros alumnos para realizar una práctica. Mediante este uso, se complementa el concepto de LARs, con laboratorios virtuales y/o físicos pero utilizados localmente por el usuario final.

Ejemplos de esta última modalidad de trabajo son la distribución de linux ADIOS [18] y el sistema VNUML [19]. ADIOS viene en un formato de livecd, y hace uso de máquinas virtuales UML y switches emulados por software para construir una topología de red para que un usuario pueda hacer prácticas de configuración de red en Linux. Por su parte, VNUML, utiliza también UMLs como emulación de computadores, pero además utiliza el simulador Dynamips [20] para simular completamente el hardware de los routers Cisco, permitiendo realizar complejos escenarios de networking con dispositivos de un lider mundial en estas tecnologías.

Agradecimientos

Los autores agradecen a los organizadores de TEyET´06, por brindar el espacio para la demostración del prototipo; a CDF y Asociados, por permitirnos el uso de sus instalaciones y a la

Universidad Politécnica de Madrid por el suministro de los robots.

Referencias

[1] Report of the Expert Meeting on Virtual Laboratories - UNESCO, 2000, Paris.

[2] Ralph E. Johnson. Frameworks = (Components + Patterns). How Frameworks compare to other object-oriented reuse techniques. Communications of the ACM. October 1997/ Vol. 40.,No 10.

[3] Ralph E. Johnson. Components, Frameworks, Patterns. Department of Computer Science, University of Illinois. 1997

[4] Lopez Luro Francisco. "Framework basado en Java para el acceso Laboratorios Remotos Físicos y Virtuales". Departamento de Ciencias de la Computación, Facultad de Economía y Administración. Universidad Nacional del Comahue, Diciembre 2006.

[5] M. Bertogna, E. Grosclaude, R. del Castillo, F. Lopez Luro, C. Zanellato. "Arquitectura para Laboratorios Remotos Físicos y Virtuales". CACIC 2005. pp 317-328.

[6] Grosclaude Eduardo, Bertogna Leandro, Lopez Luro Francisco, Zanellato Claudio, Sánchez Laura, Rodríguez Jorge, Del Castillo Rodolfo. "Experiencia con Laboratorio Remoto Colaborativo". Exposición y Demostración. La Plata 2006

[7] Arquitectura Cliente-Servidor. http://en.wikipedia.org/wiki/Client-server.

[8] Patrones de Diseño, http://en.wikipedia.org/wiki/Design_pattern

[9] Java, http://java.sun.com/

[10] Java Applet, http://java.sun.com/applets/

[11] Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995

[12] Mark Grand. Patterns in Java: a catalog of reusable design patterns illustrated with UML, volume 1. Wiley Computer Publishing, 1998. ISBN 0471258393

[13] Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[14] GNU Screen, www.gnu.org/software/screen

[15] OpenOffice.org, http://www.openoffice.org/

[16] The Apache HTTP Server Project, http://httpd.apache.org/

[17] Skype Internet Phone, http://www.skype.com

[18] ADIOS Linux. http://os.cqu.edu.au/adios/index.html

[19] VNUML. http://www.dit.upm.es/vnumlwiki/index.php/Main_Page

[20] Dynamips. http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator

Dirección de Contacto del Autor/es:
Buenos Aires 1400 – Neuquén - Argentina
{flopez,mlbertog,lsanchez,jrodrig,rolo}@uncoma.edu.ar

{flopezluro,mlbertog,lmgsanchez,jrodrig05,rdc541}@gmail.com