MonitorES como una Herramienta de acercamiento a las Redes de Computadoras

Flores S., Printista A.M., Molina S., Pérsico A., Apolloni R. *
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950
5700 - San Luis
Argentina
e-mail: {sflores, mprinti, persicoa, smolina, rubenga}@unsl.edu.ar

Resumen

En este trabajo presentamos las conclusiones de la primera etapa planificada en un proyecto educativo de nuestra Facultad. Como objetivos generales nos propusimos desarrollar una herramienta educativa que estuviese dirigida a los alumnos avanzados de nuestra carrera, a través de la currícula corriente. La idea es ofrecerles un ámbito de discusión y desarrollo disciplinar en este área, no fuertemente abordada en nuestro Departamento. De esta manera también contribuir a la formación de tesistas, becarios y pasantes interesados en esta temática.

Concretamente, en este trabajo presentamos el diseño de *MonitorES*, un *Monitor* de *E*stado de *S*ervidores en una red de computadoras, trabajando en un ambiente *TCP/IP*. La motivación principal del desarrollo de esta primera etapa estuvo basada en el siguiente concepto: *para diseñar eficientemente aplicaciones distribuidas, primero hay que familiarizarse y conocer la performance de la red sobre la que se implementará la aplicación*

Una consecuencia inmediata del proyecto fue la formación de recursos humanos en el área altamente crítica de redes, produciendo un acercamiento amigable de usuarios no expertos en la temática.

Palabras Claves:
Monitor, Performance, Métrica, TCP/IP, Cliente-Servidor.

*Grupo soportado por la UNSL (Proyecto Educativo P-3-0387/02 y Proyecto de Inv. P-338403) y la ANPCYT (Agencia Nacional para Promoción de la Ciencia y la Tecnología)

1 Motivaciones y Objetivos

Muchos años dedicados a la docencia y a la investigación, al diálogo y al trabajo con docentes y alumnos provenientes de distintas formaciones, han delimitado un deterioro de intereses comunes con respecto a la enseñanza y a los modos posibles de intervención para participar activamente en el proceso de las transformaciones educativas que en distintos niveles se vienen produciendo. Los problemas que se fueron planteando en el campo de la tecnología, conducen a la instrumentación de abordajes metodológicos alternativos.

La práctica docente debe organizar la escena en la que se vinculan los sujetos a través del conocimiento, en este contexto es que se realiza la inserción del uso de la herramienta con fines didácticos.

La mayoría de los integrantes de esta propuesta somos docentes de cátedras de la carrera Licenciatura en Ciencias de la Computación. En nuestras prácticas se utiliza software de alto nivel para acceder a los servicios de una red. Sin embargo se percibe que, el interés de los alumnos por conocer más profundamente el proceso real de comunicación entre computadoras (aspecto que es ocultado por el software de alto nivel), ha crecido considerablemente en los últimos años.

Esto se debe a que los alumnos en la ultima década, perciben como natural y corriente sentarse a desarrollar y ejecutar sus aplicaciones sobre una red más que sobre el viejo modelo de única computadora.

Lo expresado en los párrafos anteriores, fueron entre otras, motivaciones que llevaron, en la actualización del plan de estudio de la Licenciatura, a reformular la materia Sistemas Operativos por Sistemas Operativos y Redes. Acompañando este proceso de cambio y actualización, un grupo de docentes, insistimos en la necesidad de generar un ámbito de discusión, intercambio y experimentación en el tema, a fin de formar y perfeccionar a los docentes involucrados y habilitarlos para que la transferencia de los conocimientos, relacionados a este área de emergente tecnología, sea consistente.

Como objetivos generales se estableció desarrollar una herramienta educativa que estuviese dirigida a los alumnos avanzados de nuestra carrera, a través de la currícula corriente. Podría directamente actuar de apoyo en las cátedras "Sistemas Operativos y Redes", "Arquitectura II", "Sistemas Distribuidos y Paralelismo" y "Redes de Computadoras"; pero también enque contribuyese a la formación de tesistas, becarios y pasantes interesados en ésta temática. La idea es ofrecerles un ámbito de discusión y desarrollo disciplinar en este área, no fuertemente abordada en nuestro Departamento.

Como consecuencia resulta la idea de construir la herramienta estableciendo los siguientes objetivos específicos:

- Lograr una herramienta educativa, de estructura modular de manera tal que sea sencilla su manipulación.
- Lograr el desarrollo de una Interfase de Programas de Aplicación (API) que facilite el desarrollo de aplicaciones distribuidas.
- Lograr que la misma herramienta realice la colección de datos que faciliten la evaluación de la performance en aplicaciones distribuidas.

Estos objetivos permitieron diferentes niveles de acceso a la herramienta en orden creciente de complejidad. En primer lugar, lograr un acercamiento amigable que permitiera el tratamiento experimental de los aspectos relacionados a las redes, conocer su comportamiento, topología y performance. En segundo lugar, permitir que cada usuario construya sus propias solicitudes de acceso a los servicios de la red, a través de ciertos programas genéricos (esqueletos) que lo abstraerán de detalles de estructuras y de comunicaciones entre las computadoras. Finalmente los usuarios podrán re-programar, por las condiciones en que estarán construidos los programas genéricos, cualquier subconjunto de las capas que conforman la herramienta.

2 Introducción

En la tecnología de computadoras el enfoque tradicional de las últimas décadas fue evolucionar hacia multiprocesadores o supercomputadoras [5, 7]. Actualmente, en los países desarrollados existe una tendencia que plantea ejecutar las aplicaciones distribuyéndolas entre varios procesadores de diferente poder computacional. Las argumentaciones a favor de esta tendencia se basan en consideraciones económicas y de eficiencia. Desde el punto de vista de los usuarios, se ha difundido el uso de sistemas en red con el objetivo de compartir archivos, recursos, información (internet), el uso del correo electrónico, etc. Sin embargo las posibilidades que brinda la conexión en red de un conjunto de computadoras son mucho más amplias que la difundida entre usuarios corrientes. Aún resta generalizar cómo realizar las ejecuciones remotas de programas sobre una red, el acceso a bases de datos remotas, cómo generar ambientes computacionales distribuidos, cómo desarrollar computaciones paralelas y cómo realizar desarrollos sobre redes que involucren inteligencia computacional.

Hay un número de problemas fundamentales asociados al uso generalizado de las redes, especialmente cuando el ambiente distribuido soportado por la red está en continuo crecimiento, tanto en tamaño como en complejidad. Ante esta situación, un administrador de sistemas necesita herramientas que lo ayuden a analizar la situación actual y entonces poder predecir el comportamiento futuro de los recursos y asesorar a desarrolladores de software acerca del rendimiento de la red subyacente.

La Figura 1 muestra la visión de un usuario, a partir de su máquina *front-end*. En ella se observan la disponibilidad de dos herramientas, cada una con distintos objetivos y desarrolladas en distintas etapas del proyecto.

La primera herramienta, *MonitorES*, ofrece la posibilidad de monitorizar la condición en que se encuentran los servidores, en cuanto a disponibilidad y performance, a través de distintas métricas tales como memoria disponible, utilización de CPU, balance de carga, etc. En un futuro se espera que asista en el análisis y experimentación de todos los aspectos relacionados a la conexión, comunicación y performance de la red.

La segunda herramienta, consistirá de una librería flexible de esqueletos algorítmicos que permita fácilmente el diseño de aplicaciones distribuidas del tipo TCP/IP. Cada esqueleto se corresponde con un paradigma clásico de computación distribuida. La palabra paradigma es a menudo usada con una connotación general como "metodologías de alto nivel que se reconocen en muchos de nuestros algoritmos efectivos". En este trabajo se usará el término en un sentido más preciso: *un paradigma de programación es una clase de algoritmos que resuelven problemas diferentes pero tienen la misma estructura de control*. Estos programas a menudo son llamados *esqueletos algorítmicos o programas genéricos*[2]. A partir de estos esqueletos

se pueden derivar *programas modelos* [1] que ilustren cómo el paradigma resuelve problemas específicos. Un esqueleto incluye tipos de datos no especificados y procedimientos que varían de una aplicación a otra. Un programa modelo luego se obtiene reemplazando estos tipos de datos y procedimientos con los tipos de datos correspondientes y con los procedimientos que constituyen el programa secuencial que resuelve el problema específico.

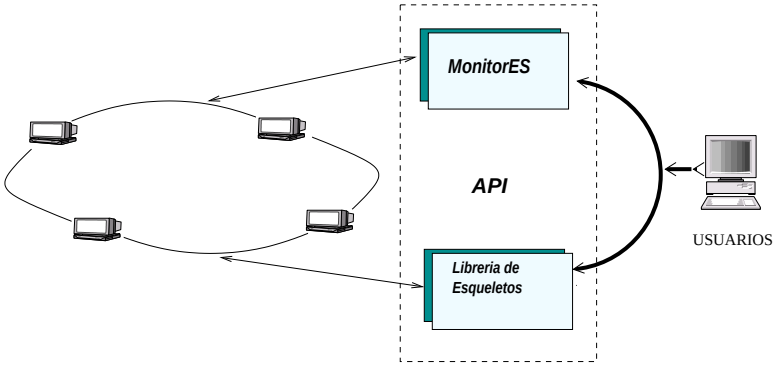

Figura 1: API: Proyecto Educativo

3 *MonitorES*

La monitorización de una colección de *hosts* es necesaria para la administración, tanto de las redes de comunicación como para los sistemas distribuidos que se ejecutan en ella.

Como se muestra en la Figura 2 la información recolectada es utilizada en la toma de decisiones de administración y en la ejecución de acciones apropiadas de control sobre el sistema.

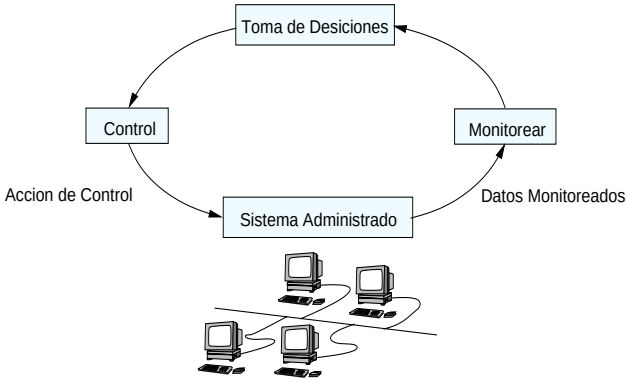

Figura 2: Modelo de Administración de un Sistema

En este trabajo nosotros discutimos principalmente la monitorización de un conjunto de *hosts*, en particular el uso de un monitor como una herramienta de acercamiento y de administración. Las acciones de control que pueden cambiar el estado de una red de servidores son dejadas fuera de discusión. Sin embargo, mostramos cómo el monitor responde dinámicamente

al impacto que pueden tener estas acciones de control sobre la red, tanto en arquitetura como en cambios de estado.

Las acciones de monitorización son ejecutadas sobre un recurso o grupos de recursos relacionados. Un recurso es definido como cualquier componente de hardware (o eventualmente de software) cuyo comportamiento puede ser controlado por un sistema de administración. Este recurso tiene una actividad propia, intrínseca a su definición y cuyos detalles internos son de vital importancia para los propósitos de la monitorización.

La información de monitorización describe el estado y los eventos asociados con un recurso o un grupo de recursos que están bajo estudio. Tal información puede ser representada por reportes de estado individuales o por una secuencia de reportes.

El objetivo de este trabajo es definir un ambiente de control de servidores capaz de brindar suficiente información sobre el estado de los diferentes *hosts* que se quieren monitorear. La información describirá la performance de los nodos a través de distintas métricas tales como memoria disponible, utilización de CPU, balance de carga, etc. Esta información será utilizada por el monitor para construir una base de datos que almacenará vistas históricas de la actividad de la red. Los reportes finales serán requeridos bajo demanda de usuarios apropiados o administradores autorizados. A través de una interfase Web, la información recolectada podrá ser manipulada, formateada o filtrada para ser reportada de modo que reúna los requerimientos específicos de la aplicación.

3.1 Arquitectura del Monitor

La arquitectura del monitor está basada en un modelo Cliente-Servidor [5, 7] como se muestra en la Figura 3. Se puede observar la presencia de nodos (X, Y, Z) los cuales actúan como clientes, denominados `mclient` y un nodo que se comporta como servidor, denominado `mserver`. Entre los procesos clientes y el servidor se implementa la funcionalidad necesaria para que *MonitorES* realice la recolección, la clasificación y el almacenamiento de toda la información de los múltiples nodos de la red. Cada `mclient` estará encargado de examinar su *host* local asociado y de proporcionar información referente al procesador, la memoria y otras métricas. Las flechas que conecta cada proceso `mclient` con el proceso `mserver` están representando el pasaje de mensajes, transmitiendo la información de estado requerida hacia el servidor. El proceso `mserver`, por su parte, estará encargado de recolectar los datos que le llegan desde múltiples fuentes de la red, ordenarlos y almacenarlos en bases de datos Round Robin, *RRD* [8]. El envío de paquetes al `mserver` se realizará a intervalos regulares de tiempo, vía TCP/IP [1, 3]. Un proceso `manager` será el encargado de chequear dos archivos de configuración: el de métricas y el de los *hosts* a monitorear. En caso de detectar un cambio de configuración, este proceso deberá comunicarse con las máquinas clientes de *MonitorES*. Una `interfase front-end`, provee una vista de la información a través de un servidor Web.

Hay dos modos de operar con *MonitorES*: modo *usuario* y modo *super-usuario*. El modo usuario permite a un usuario común visualizar la performance de los *host* que componen la red, relativa a las métricas evaluadas. El modo *super-usuario* permite a un administrador de red inicializar y modificar el pool de *hosts* a ser monitoreados, como así también activar las métricas del conjunto disponible.

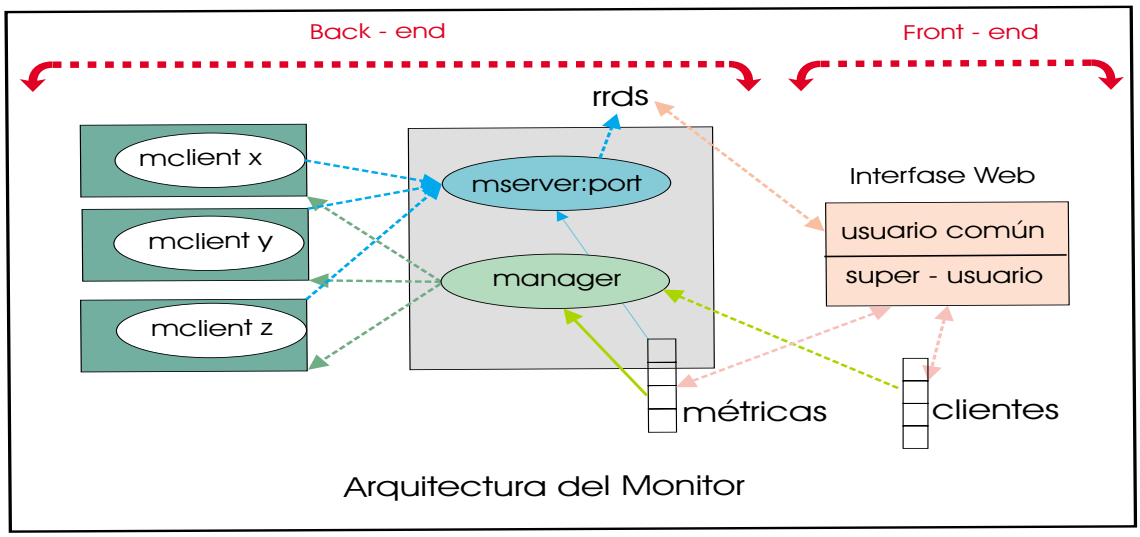

Figura 3: Arquitectura de *MonitorES*

3.1.1 Módulo Recolector

En esta sección se presentan detalles del funcionamiento de un módulo fundamental de *MonitorES*.

Conocer el estado de un conjunto de *hosts*, constituye una herramienta fundamental en la buena administración de una red. Identificar el uso de la memoria, la CPU, el disco y otras variables le permiten al administrador de sistemas racionalizar el uso de los recursos.

El objetivo de *MonitorES* es chequear los *hosts* de una red a intervalos regulares o bajo demanda explícita, y tomar acciones predefinidas (no correctivas) cuando un recurso no responde. Puede alertar de forma visual y sonora, en viarun e-mail a una casilla de correo, a un localizador o teléfono móvil, ejecutar programas externos, etc. Todas estas acciones permiten al administrador de sistema solucionar un problema antes del deterioro masivo de la performance de la red o de los sistemas.

`mserver`

Es un proceso servidor que monitorea un conjunto de *hosts*. Este proceso espera por la solicitud de comunicación de algún proceso `mclient`, en un puerto conocido. Los paquetes que recibe concurrentemente de los nodos, los guarda en bases de datos Round Robin (*RRD*) para su posterior procesamiento.

El proceso `mserver` almacena la información de estado de los *hosts* en bases de datos *Round Robin*. Una base de datos *Round Robin* es una base de datos circular que permite almacenar y representar datos en intervalos regulares de tiempo. Una característica importante de una base de datos Round Robin es que al ser circular, su tamaño no crece en el tiempo, es decir, siempre contiene la misma cantidad de datos, ya que cuando completa la extensión de la base de datos, simplemente sobreescribe a partir de los datos más antiguos. Las bases de datos circulares se crean en un archivo con extensión *.rrd* y *MonitorES* utiliza la herramienta *RRDtool* para su procesamiento.

Estas facilidades alertarán al administrador de sistema, sobre posibles ca ídasen la performance de los *hosts*, con el objeto de prevenir situaciones no deseadas.

`mclient`

Es un proceso cliente que se encuentra recolectando información del estado de su nodo local. El proceso `manager` es quién establece el tipo de métrica que cada `mclient` debe recopilar. Esta información se envía, en forma de paquetes, al `mserver` que está esperando en un puerto conocido.

El modelo de comunicación elegido para este monitor es socket con conexión usando TCP/IP, consecuentemente es una comunicación sincrónica. Dentro de este esquema, el proceso `mserver` da a conocer un puerto por el cual los procesos `mclient` solicitan la comunicación. Una vez establecida esta comunicación, cada `mclient` comienza a transmitir la información solicitada, y seguirá transmitiendo hasta que se detecte un cambio de configuración.

3.2 Un Sitio Web como Interfase de *MonitorES*

La `interfase front-end` de $MonitorES$ es un sitio Web, por lo tanto está disponible a cualquier usuario de Internet, en especial al administrador del sistema, que independientemente del lugar en donde se encuentre puede realizar los controles necesarios, mejorando su administración en situaciones críticas, tomando decisiones atiempo.

La `interfase front-end` de $MonitorES$ es básicamente una herramienta de configuración y la expresión gráficade toda la información recolectada por $MonitorES$ (los valores de las métricas de un *host* particular).

Este sitio Web permite dos modos de acceso a sus páginas, modo*usuario* (páginas delibre acceso) y modo *super-usuario* (páginas con acceso restringido).

Páginas de libre acceso son páginas correspondientes al modo de operación *usuario*, con acceso habilitado a todo visitante Web, permiten la configuración y visualización del gráfico correspondiente a los valores de alguna métrica de un *host* paticular. La configuración se basa en la selección de parámetros, tales como: métrica a graficar, *host* para cada métrica, color de línea, ancho, tipo de línea, etc.

Páginas con acceso restringido son páginas correspondientes al modo de operación *super-usuario*, destinadas al uso del administrador del sistema, permiten consultar y crear información de configuración para el monitor.

La Figura 4 muestra la página a través de la cual, cualquier usuario puede seleccionar las propiedades de una visualización personalizada de las métricas de uno o varios *hosts*.

La Figura 5 muestra la página de configuración de `Métricas-Hosts`, accesible sólo en modo *super-usuario*. Esta página manipula el grupo de métricas, el grupo de *hosts* disponibles y permite especificar el conjunto de métricas asociadas a un determinado *host*.

Restringir el acceso de un usuario distinto del administrador del sistema a ciertas páginas, forma parte de nuestro mecanismo de protección, para resguardar la información de las estructuras básicas de configuración del monitor.

Figura 4: *MonitorES* - Modo Usuario.html

Figura 5: *MonitorES* - Modo_Super_Usuario.html

La comunicación de la `interfase Web` con el proceso `manager`, es decir la conexión del `front-end` con el `back-end`, se realiza mediante un archivo de configuración, y las bases de datos Round Robin administradas por el `mserver`.

El archivo de configuración contiene: información de los *hosts* que se están monitoreando, el tipo de métricas que se están evaluando para los *hosts* en cuestión y los requerimientos pendientes del administrador del sistema. Los requerimientos pendientes consisten en informarle al proceso `manager` las modificaciones de las métricas y el grupo de *hosts* que se pretenden

ev aluara partir de ese momento.

La `interfase` del monitor genera los gráficos de las distintas métricas a partir de los valores almacenados en sus correspodientes base de datos Round Robin. De esta manera, los usuarios podrán evaluar el funcionamiento del sistema eficientemente basándose en valores recientes.

3.3 Un Caso de Prueba

Las *RRD* son de gran utilidad para representar y monitorizar datos que puedan ser medidos, como por ejemplo la memoria disponible. A continuación se muestra un ejemplo de uso.

En particular, *MonitorES*, en su versión Demo, realiza una monitorización de tres servidores de uso intensivo de la Universidad.

Las Figuras 6 , 7 8 muestran los datos coleccionados en tres *hosts* con el objetivo de monitorizar, a medida que transcurre el tiempo, su memoria disponible. En cada figura, el eje de coordenadas x representa el tiempo y el eje y representa la cantidad en Megabytes de la memoria disponible en el *host* respectivo.

server **srv1**: con 384 Megabytes de memoria y con 7000 cuentas de usuarios, brinda los siguientes servicios: POP (Post Office Protocol), SMTP (Simple Mail Transfer Protocol), DHCP (Dynamic Host Configuration Protocol) y DNS (Domain Name System).

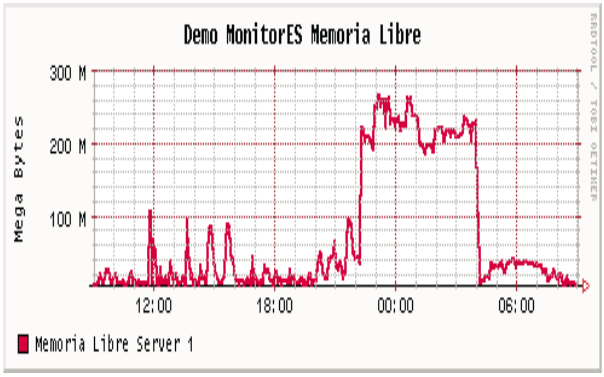

Figura 6: *MonitorES* - Memoria Disponible: **srv1**

server **srv2**: con 128 Megabytes de memoria y con 600 cuentas de usuarios, brinda los siguientes servicios: SMTP (Simple Mail T ransferProtocol, POP (Post Office Protocol) y servidor web APACHE.

server **srv3**: con 64 Megabytes de memoria y 500 cuentas de usuarios, brinda los siguientes servicios: POP (Post Office Protocol), PPP (Point-to-Point Protocol) y servidor web APACHE.

Finalmente, en la Figura 9, se muestra el resultado de otra solicitud a *MonitorES*, a través de la `interfase front-end`. El resultado es el desarrollo de una gráfica comparativa de la disponibilidad de memoria de los tres servidores, utilizando la información recolectada de las últimas 24 horas de operación de cada servidor.

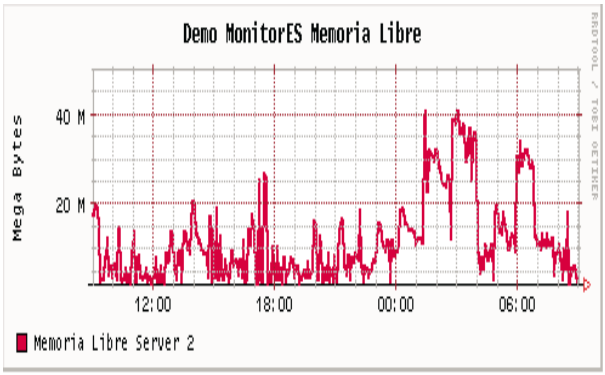

Figura 7: *MonitorES* - Memoria Disponible: **srv2**

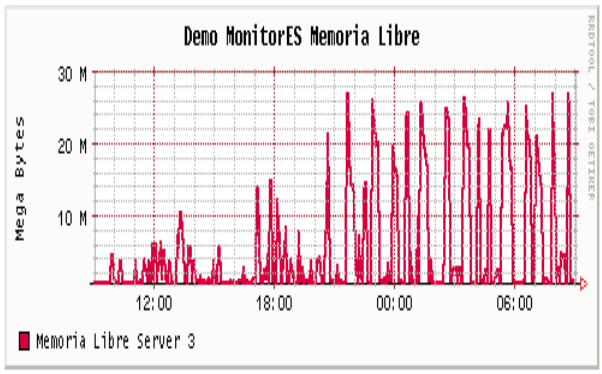

Figura 8: *MonitorES* - Memoria Disponible: **srv3**

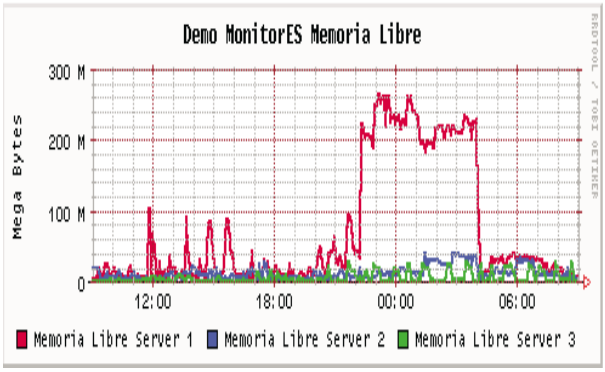

Figura 9: *MonitorES* - Cuadro Comparativo de Memoria Disponible de los tres servidores

3.4 Aporte de *MonitorES*

Generar un sistema de monitorización es una herramienta importante para la administración de una red.

Nosotros hemos definido un modelo de monitorización en términos de un conjunto de servicios. Las principales funciones descriptas son: generación de información de monitorización (a través de un `sistema recolector de información` el cual incluye el relevamiento del estado de ciertos recursos); el almacenamiento de la información (en base de datos, *RRD*), procesamiento,

combinación y filtrado de información relevante; y además *MonitorES*, ante los requerimientos de los usuarios vía Web, realiza la presentación gráfica de la información de monitorización resultante.

Uno de los aspectos atractivos de *MonitorES* es que no es un monitor que interfiere fuertemente en la performance o que altera la secuencia de actividades que ocurren en una red. *MonitorES* es un sistema independiente que se utiliza para recolectar información de estado de ciertos recursos.

Una función importante provista por *MonitorES* es el envío automático de `alertas` que ayudan al administrador de sistemas a tomar decisiones a tiempo, evitando que pequeños problemas se tranformen en grandes problemas, debilitando así la performance total de la red.

Aunque *MonitorES* esté en estado de desarrollo, lo discutido en esta publicación confirma la utilidad de la herramienta para la administración de estado de los servidores de una red.

4 Trabajo Futuro: la construcción final de la *API*

Para reducir su diseño la mayoría de las redes están organizadas como una serie de capas o niveles y cada una es construida sobre la capa predecesora. La función de cada capa es proveer ciertos servicios a la capa mas alta, de manera que capas mas altas no traten con los detalles de cómo los servicios de capas subyacentes son implementados realmente. En cada nivel se deberá establecer un conjunto de reglas o protocolo que especifique su funcionalidad. Esto es necesario para establecer comunicación (virtual) entre procesos que corren en distintas máquinas pero a un mismo nivel de la jerarquía de capas. Es importante para el proyecto, en estado de diseño, tener claro dos aspectos:

- Definir claramente las interfases entre las capas.
- Especificar el conjunto de funciones específicas que cada capa debe implementar.

Si estos dos requerimientos son bien definidos se minimiza la información que debe fluir entre las capas, y también se facilita el reemplazo de una capa con otra implementación totalmente diferente que ofrezca el mismo conjunto de funciones a sus capas vecinas pero con un diseño distinto (como ya fue mencionado es uno de nuestros principales objetivos!).

Tanto el conjunto de capas como el de protocolos conforman la arquitectura de la red. (No forman parte de la arquitectura los detalles de implementación ni las especificación de las interfases).

4.1 Actividades globales

1. Realizar una especificación de la arquitectura que contenga suficiente información para permitir que distintos implementadores escriban un programa para cada capa, de modo que el programa obedezca correctamente al protocolo apropiado. Es muy importante resaltar la abstracción de los procesos que administran las capas. Sin esta técnica será muy difícil o imposible particionar el diseño completo en partes mas pequeñas y mas manejables.

2. Desarrollar un esqueleto de la arquitectura el cual contenga solo las interfases en tre las capas. Es decir, en este punto ya se ha decidido cuáles ser´anlos servicios que nuestra red esta dispuesta a ofrecer. En tal direcci´on se deberá preparar al prototipo para que permita transferir información en sentido vertical en nuestra API, recepcionando y entregando información entre las capas. En este punto es de esperar la reformulaci´on en algún sentido de la arquitectura propuesta inicialmente, y a que pueden surgir características no contempladas inicialmente o simplemente pueden surgir necesidades adicionales que no aparecieron en el momento de diseño.

3. Implementar las capas de la red diseñada. El conjunto de capas es la que le proveerá la funcionalidad y la característica particular a la red. Como ya se dijo cada etapa realizará una funci´on perfectamente definida y el ´exito de esta etapa dependerá de la claridad con que la red hay a sido definida en el paso 1.

4. Ensamblar el prototipo. La intenci´on final de este proyecto, como ya se mencionó, es la construcción de un API, una interfaz de alto nivel entre la red y los programas de usuarios. Las funciones concernientes a c´omo transmitir bits sobre un canal de comunicación, lo cual es implementado a nivel de una capa física, queda totalmente fuera del alcance de este proyecto. Nosotros asumimos la existencia de facilidades de comunicación física.

5 Conclusiones

Las ideas expuestas en este trabajo, se encuadran dentro de un proyecto educativo del Departamento de Informática de la Facultad de Ciencias Físico, Matemáticas y Naturales de la UNSL.

El objetivo general, contempla el estudio, diseño e implementación de una Interfase de Programas de Aplicación (API) que facilite el desarrollo de aplicaciones distribuidas.

Esta herramienta ha constituido un ámbito de discusi´on permanente para los docentes que participaron en el proyecto y para todos aquellos que se encuentran dedicados a la enseñanza que de una u otra manera está influenciada por una plataforma de red.

La relevancia de esta herramienta es que provee un medio para concretizar conceptos abstractos, como es la performance de un conjunto de servidores ofreciendo servicios.

Considerando que la enseñanza se imparte a alumnos de carreras de grado, es adecuado captar la atenci´on de los mismos con resultados concretos sobre conceptos teóricos. Se la ha considerado como una manera de lograr el aprendizaje significativo, incorporando conceptos de redes, dentro de la estructura cognitiva del alumno.

Referencias

[1] Comer D. E. - Internetworking with TCP/IP Vol I Principles, Protocols, and Architecture - Second Edition - Prentice Hall - 1991.

[2] Comer D. E. - Computer Networks and Internet - Second Edition - Prentice Hall - 1999.

[3] Comer D. E. and Stevens D.L.- Internetworking with TCP/IP - Prentice Hall.

[4] Stallings W. - Data and Computer Communications 6a Edición - Prentice Hall.

[5] Tanenbaum A. S. - Modern Operating Systems - Prentice Hall.

[6] Tanenbaum A. S. - Computer Networks - Third Edition - Prentice Hall - 1996.
[7] Wilkinson B. And Allen M. - Parallel Programming Techniques and Applications using Networked Workstations and Parallel Computers - Prentice Hall -1999.
[8] Tutorial RRDtool, http://people.ee.ethz.ch/ oetiker/webtools/rrdtool