MonitorES
Un Monitor de Estado de Servidores

Apolloni R., Flores S., Molina S., Pérsico A., Printista M. *

Departamento de Informática

Universidad Nacional de San Luis

Ejército de los Andes 950

5700 - San Luis

Argentina

e-mail: {rubenga, sflores, smolina, mprinti}@unsl.edu.ar

Resumen

La performance de las redes de computadoras se presenta como un tópico atractivo, cuyo análisis es una tarea de gran interés en la actualidad. En este trabajo se presenta el diseño de *MonitorES*, un *Monitor* de *E*stado de *S*ervidores en una red de computadoras, trabajando en un ambiente *TCP/IP*. La función del monitor es obtener información que describa 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 caso de la detección de alguna anormalidad en la red, *MonitorES* alertará a un administrador de sistemas o de red que tomará las acciones necesarias antes de que el problema resulte inmanejable. Finalmente, en este trabajo también se presenta el ambiente de ejecución de *MonitorES* a través de una interfase Web.

1 Introducción

En la tecnología de computadoras el enfoque tradicional de las últimas décadas fue evolucionar hacia multiprocesadores o supercomputadoras [5, 7]. Sin embargo, en los países desarrollados existe una tendencia contrapuesta al enfoque tradicional 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.

*Grupo soportado por la UNSL y la ANPCYT (Agencia Nacional para Promoción de la Ciencia y la Tecnología)

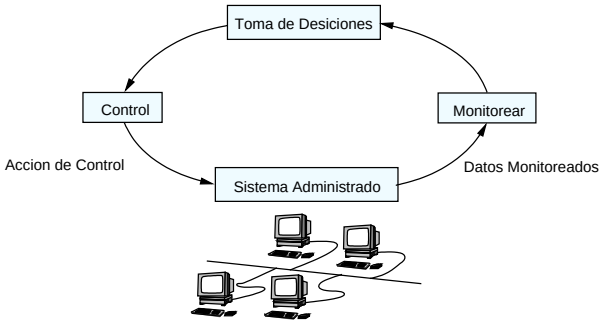

Figura 1: Modelo de Administración de un Sistema

Hay un número de problemas fundamentales asociados al uso generalizado de las redes, especialmente cuando el ambiente distribuido soportado por la red está en contínuo 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.

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 al Figura 1 la información recolectada es utilizada en la toma de decisiones de administración y en la ejecución de acciones apropiadas de control sobre el 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 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.

Dos enfoques son comúnmente utilizados para chequear el comportamiento de un recurso: en términos de su estado y de sus eventos. El estado de un recurso es una medida de su comportamiento en un punto discreto de tiempo y es representado por el valor corriente de un conjunto de variables de estado, denominadas métricas. Un evento es una entidad atómica que refleja un cambio en el estado de un recurso.

Para *MonitorES* se escogió una técnica de Monitorización Manejada por Tiempo (*Time-Driven Monitoring*). Esta técnica está basada en adquirir información de estado periódica con el objetivo de recolectar vistas instantáneas del comportamiento de un recurso. Hay una relación estrecha entre los intervalos de muestreo y la cantidad de información generada.

El estado de un objeto tiene una duración en el tiempo, como son la longitud de cola ready o la cantidad de memoria disponible y estos son ejemplos del tipo de monitorizacion manejada por tiempo.

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.

La Organización de Estándares Internacionales (ISO) ha definido una serie de estándares para la administración de sistemas de comunicación para la inteconexión de Sistemas Abiertos (OSI)[2, 6]. Los aspectos metodológicos del presente trabajo se basan en las definiciones

realizadas por OSI respecto a la administración de un recurso.

En la administración de una red existe una jerarquía de protocolos los cuales son conjuntos de reglas que describen una secuencia de acciones que inicializan y controlan la transmisión de datos a lo largo de los medios físicos. Una de las principales familias de protocolos es la de TCP/IP [1, 3]. Específicamente, para el desarrollo de este trabajo se ha seleccionado como Modelo de Programación, el *Modelo Socket*, a través del cual, se podrá acceder completamente a la funcionalidad de TCP, UDP y a la de otros protocolos [1, 2, 3, 6].

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.

2 Arquitectura del Monitor

La arquitectura del monitor está basada en un modelo Cliente-Servidor [5, 7] como se muestra en la Figura 2. 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.

3 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 racionalizar el uso de los recursos.

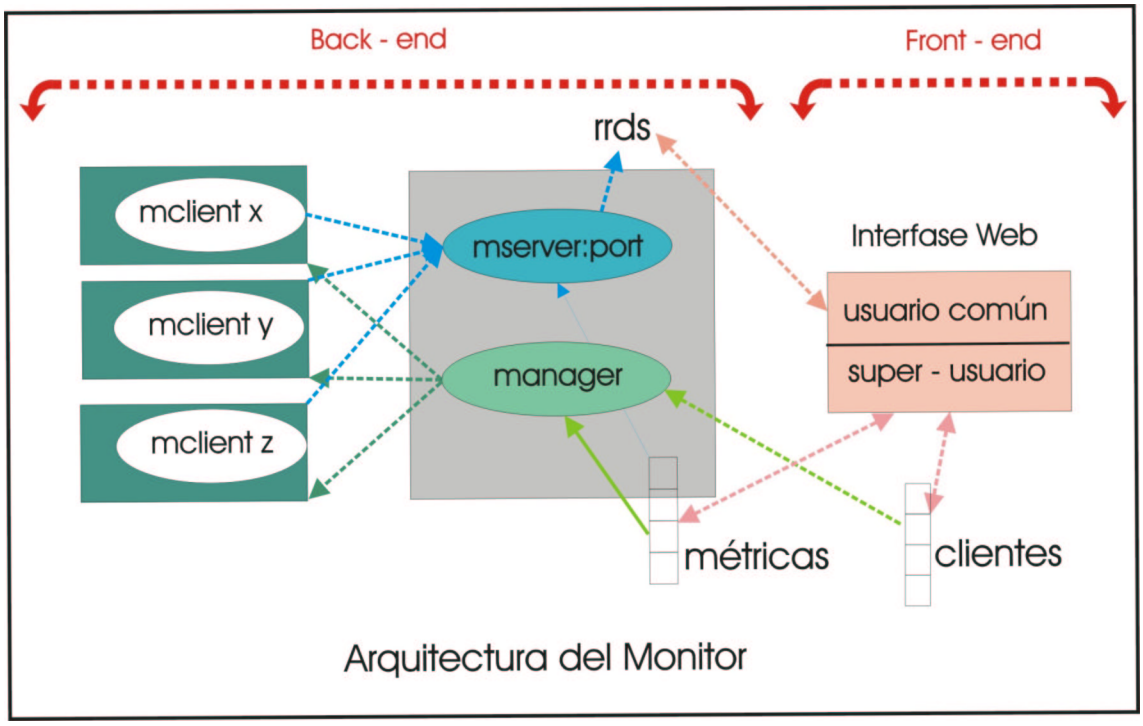

Figura 2: Arquitectura de MonitorES

El objetivo de *MonitorES* es chequear los *hosts* de una red a intervalos regulares o bajo demanda explícita, y tomar acciones predefinidas cuando un recurso no responde. Puede alertar de forma visual y sonora, enviar un e-mail a una casilla de correo, a un localizador o teléfono móvil, ejecutar programas externos, reiniciar computadores locales o remotas, conectarse con la red, etc. Todas estas acciones permiten 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.

Como se muestra en la Figura 3 fue necesario determinar un umbral (threshold) para cada una de las métricas, de modo que de acuerdo a este umbral se establecerá la precisión del monitor para evaluar la performance de un nodo.

Estas facilidades alertarán al administrador de sistema, sobre posibles caídas en la performance de los *hosts*, con el objeto de prevenir situaciones no deseadas. Un pseudocódigo del procedimiento ejecutado por *MonitorES* para enviar un mail electrónico al administrador de sistema se muestra en la Figura 4.

`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.

```
// Proceso mserver
  //Inicia el proceso mserver
  sockfd=new(PORT);

  while ( 1 ) {
    // Acepta conexiones de los clientes
    newsockfd = acceptClients(sockfd);

    if ( fork() == 0  )  {
      // Recibe los paquetes de clientes
      packet = recv_metric(newsockfd)

      // Genera alerta
      if ( metric(packet) > THRESHOLD )
        alert(packet);

      // Colecciona m'etricas en RRD
      rrd(packet);
      close(newsockfd);
    }
    else
      close(newsockfd);
  }
```

Figura 3: *Proceso mserver*

Dentro de este esquema, el proceso `mserver` da a conocer un puerto por el cual los procesos `mclients` 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.

3.1 Round Robin Data

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, 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.

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 a tiempo.

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

```
// Proceso Alerta
#define MAILCMD "mail -s \"MonitorRed: ALERTA Urgente\""

alertar_admin(string mensaje){
  string mailcmd= NULL;
  string admin;
  FILE *mailproc;

  // Busca administrador de turno, en la base de datos de administradores
  admin= find_admin_db(mails.db);
  // Armado del comando mail mail -s <"subject"> <"direccion de correo electronico">
  strcat (mailcmd, MAILCMD);
  strcat (mailcmd, " ");
  strcat (mailcmd, admin);

  // Invoca al programa mail creando un pipe
  if ((mailproc= popen(mailcmd, "w"))==NULL){
    printf("Error: No se pudo abrir el pipe.");
    exit(1);
  }

  // Envia el texto del mensaje
  fprintf(mailproc, "%s", mensaje);
  fflush(mailproc);

  // Cierra el pipe y finaliza
  pclose(mailproc);
}
```

Figura 4: *Proceso de Alerta: Envio de un mail*

```
// Proceso mclient espera conexiones del proceso manager
// para ejecutar la(s) m'etrica(s)o cambia la configuraci'on de la(s)
// m'etrica(s) a evaluarse(s)
  manager_sockfd=config(MANAGER_PORT);

  while (1) {
    // Configuraci'on de una m'etrica
    m_sockfd=acceptmetric(manager_sockfd);

    if ( fork() == 0 ) {
      // Recibe configuracion
      strcpy(metric,recv_tcp(m_sockfd));

      while ( TIME_INTERVAL ) {
        // Ejecuta regularmente
        packet = decode(metric);

        // Env'ia metrica al mserver
        enviarMetric(s,port);

        close(m_sockfd);
      }
    }
    else
      close_tcp(m_sockfd);
  }

//  Funci'on decode
packet decode( metric ) {
  switch (metric){
    case '0': /* memoria */
              packet(ip,metric,value,timestamp) = run(script_mem);

    case '1': /*  disco  */
              packet(ip,metric,value,timestamp) = run(script_disk);

    case '2': /*  cpu    */
              packet(ip,metric,value,timestamp) = run(script_cpu);
  }
}
```

Figura 5: *Proceso mcliente*

Este sitio Web permite dos modos de acceso a sus páginas, modo *usuario* (páginas de libre 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 6 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*.

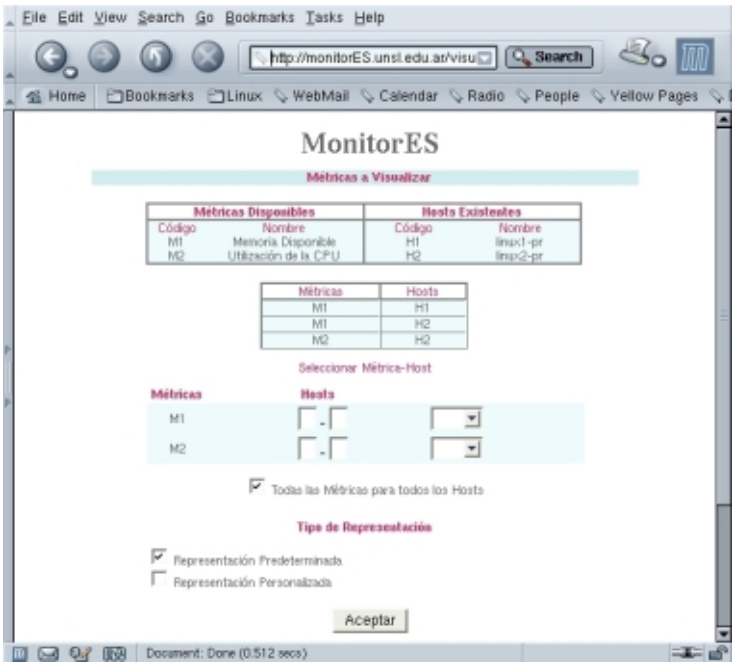

Figura 6: *MonitorES* - Modo Usuario.html

La Figura 7 muestra la página de configuración de `M'etricas-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.

La comunicación de la `interfase Web` con el proceso `manager`, es decir la conexión del `front-end` con el `back-end`, se realiza mediante archivos de configuraciones, 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 evaluar a partir de ese momento.

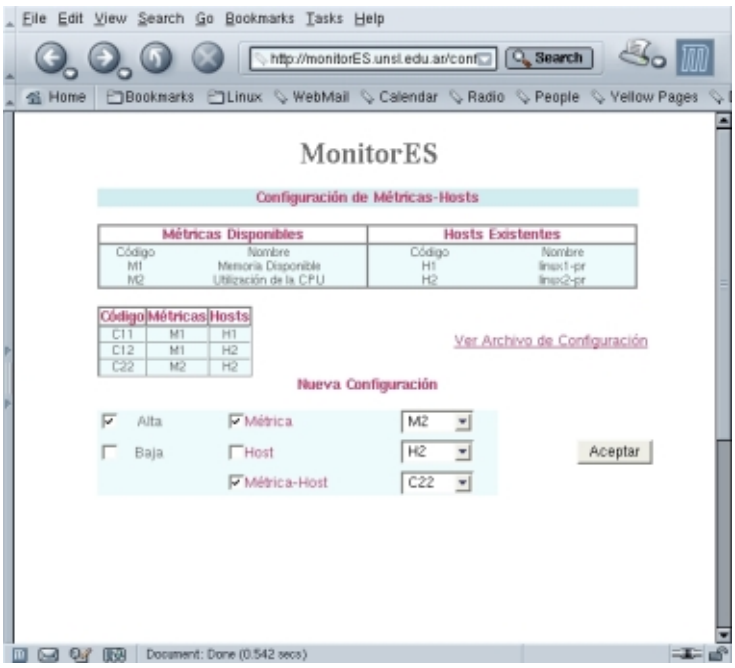

Figura 7: *MonitorES* - Modo_Super_Usuario.html

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.

4 Un Caso de Prueba

Las *RRD* son de gran utilidad para representar y monitorear 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, realizó una monitorización de tres servidores de uso intensivo de la Universidad.

Las Figuras 8 , 9 10 muestran los datos coleccionados en tres *hosts* con el objetivo de monitorizar en 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).

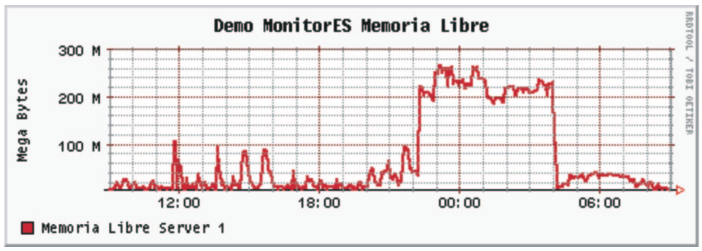

Figura 8: *MonitorES* - Memoria Disponible: **srv1**

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

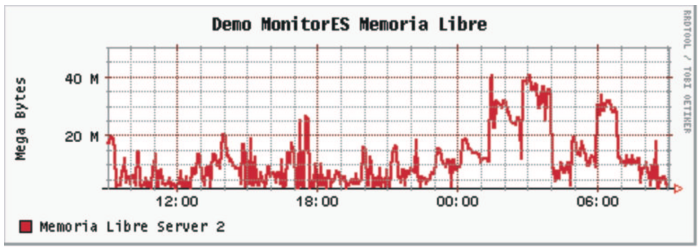

Figura 9: *MonitorES* - Memoria Disponible: **srv2**

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.

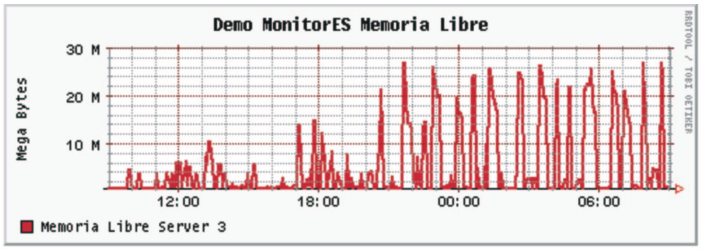

Figura 10: *MonitorES* - Memoria Disponible: **srv3**

En la Figura 11, 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.

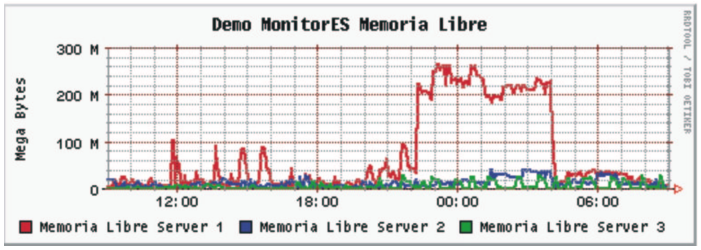

Figura 11: *MonitorES* - Cuadro Comparativo de Memoria Disponible de los tres servidores

Finalmente, la Figura 12 muestra la respuesta de *MonitorES* a la solicitud de monitorización del estado de la CPU del server **srv1**. La figura muestra los porcentajes de utilización del usuario y sistema.

5 Conclusiones

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

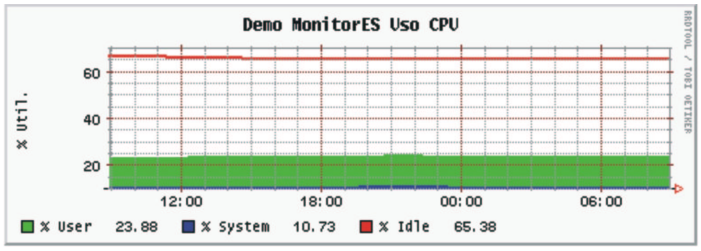

Figura 12: *MonitorES* - Estado de CPU: **srv1**

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 informaci'on` 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*, antes 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.

Referencias

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