INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

MONITOREO DEL COMPORTAMIENTO DE SERVIDORES DE APLICACIONES

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
CON ESPECIALIDAD EN
COMPUTACIÓN DE ALTO RENDIMIENTO

P R E S E N T A
HÉCTOR JULIÁN SELLEY ROJAS

DIRECTOR DE TESIS
Dr. Felipe Rolando Menchaca García

AGRADECIMIENTOS

A mi asesor el Dr. Felipe Rolando Menchaca García por todo el apoyo y conocimientos que amablemente compartió con mi persona, así como por todo el valioso tiempo que dedicó para el desarrollo de esta tesis.

Al Instituto Politécnico Nacional y al Centro de Investigación en Computación por haberme otorgado el honor de pertenecer a estas grandes Instituciones. Al CONACYT por haberme financiado con la beca la realización de mis estudios y desarrollo de esta tesis.

A mis padres, ya que si su apoyo y amor incondicional hubiese sido imposible la realización de este trabajo. Los quiero mucho.

A mi madre, que sin su cariño, atención y extremas precauciones en mi persona, no habría llegado hasta este momento. Te quiero mucho.

A mis hermanos: Armando, Karenin (a quién considero como una segunda hermana), César y Roxana, por haber representado un ejemplo ilustre en mi vida, gracias por su apoyo, preocupaciones y cariño incondicional. Los quiero a todos.

A mis amigos de toda la vida: Daniel, Adrián, Víctor, el "S", Paco, Armando y Pedro, que con su apoyo moral (y económico en muchas ocasiones), compañía y ejemplo pude reunir las energías necesarias para poder continuar en esta labor. Muchas gracias por permitirme ser parte de sus vidas.

A mi amiga Selene por permitirme formar parte de su vida, y por ser una persona fundamental de la mía. Por ti he aprendido mucho de la vida y sobre todo de mi mismo. Te quiero mucho.

A mis amigos y compañeros de la maestría: Adrián, Obdulia, Adriana, Edgar, Diego, Alex y todos los demás, gracias por apoyarme en el desarrollo de esta tesis y por las innumerables experiencias que vivimos juntos. A mi amiga Reyna por compartir más que un cubículo, música y café, sin tu apoyo y tolerancia en los momentos difíciles y de desesperación me hubiera sido imposible; gracias por todas tus notitas y comentarios que me llenan de ánimo.
Gracias a todos ustedes por su tiempo, cariño y amistad, los quiero a todos.

Y finalmente gracias a Dios por permitirme vivir hasta este día (lo veía muy lejos pero llegué), gracias por darme salud, una gran familia y muchos amigos; todas estas son la mejor de las virtudes. Y muchas gracias a todos los demás que ya no pude mencionar.

Atte. Héctor Selley

ÍNDICE

MONITOREO DEL COMPORTAMIENTO DE SERVIDORES DE APLICACIONES

RESUMEN

Los servicios que se ofrecen en Internet son importantes debido a que tienen un gran impacto social en la actualidad, ya que forman parte del estilo de vida de muchas personas. Por otro lado los proveedores de servicios ofrecen una amplia gama de servicios electrónicos, en los cuales invierten una gran cantidad de dinero para tener una infraestructura capaz de servir a muchos usuarios. Para brindar servicios de alta calidad se requieren servidores de aplicaciones con especificaciones tales que permitan un buen desempeño. Por lo tanto, la calidad de servicio está en función del desempeño de estos servidores, por lo que la medición de este cobra una importancia relevante.

En el presente trabajo se presenta una metodología para la medición del desempeño y una herramienta que permita medir este factor en un servidor de aplicaciones. Para diseñar ésta metodología se realizó un análisis para seleccionar un conjunto de métricas que permita medir apropiadamente el desempeño del servidor empleando mínimos recursos de cómputo. Se considera que este conjunto de métricas es mínimo y adecuado para caracterizar el desempeño del servidor.

Las principales contribuciones de este trabajo son la metodología para caracterizar el desempeño del servidor, una herramienta para medir el conjunto selecto de métricas y una metodología de análisis para supervisar el funcionamiento del servidor y solucionar algún posible problema que afecte al desempeño.

La tesis está organizada en seis capítulos que abarcan tres aspectos del diseño.
En el capítulo 1 se presentan algunos conceptos teóricos y definiciones para el estudio de las métricas en las diversas capas de la infraestructura del servidor, ya que son fundamentales para la metodología. En el capítulo 2 se incluyen algunas investigaciones previas realizadas por algunas organizaciones. En el capítulo 3 se muestra la arquitectura y diseño de las etapas que conforman la aplicación, así como algunos detalles de operación. Mediante la implementación de una herramienta de software, en el capítulo 4 se establece que será posible realizar un monitoreo permanente, mostrar los resultados que describen el estado actual mediante gráficas de comportamiento y evaluar con los criterios de análisis un índice de comportamiento que indicará cuan bueno es el servicio que ofrece el servidor de aplicaciones. Las pruebas efectuadas y los resultados obtenidos en el capítulo 5 sustentan que el diseño y su implementación son efectivos para efectuar la medición del desempeño de un servidor de aplicaciones de acuerdo a los objetivos planteados. Finalmente en el capítulo 6 se presentan las conclusiones y comentarios finales referentes al término de la presente tesis. Así mismo se presenta una propuesta de trabajos a futuro que pueden ser realizados a partir de este.

APPLICATIONS SERVER PERFORMANCE MONITORING

ABSTRACT

Services provided through Internet have significant relevance for society; nowadays they're a lifestyle of many people. In the other hand providers offer a wide variety of electronic services, in wich they spend lots of money in order to build infraestructure to serve more and more users. In order to provide high quality services are required applications servers with high specifications to get good performance. Then quality of service (QoS) is determinated by server performance, that's why measuring application server performance has become so important.

This work presents a methodology for measuring server performance and also a piece of software capable of determinate performance of an application Server. To design this methodology we worked on analysis to select a metric set wich could be adequate to measure server performance without to spend a lot of resources of computing infrastructure. We consider that metric set selected is minimal but it's enough to characterize Server performance properly.

Thus principal contributions of this work are methodology to characterize Server performance, software to measure metric set selected and analysis methodology to supervise Server operation and fix problems related with performance.

This thesis is organized in six chapters wich cover these three aspects of the design.

Chapter 1 presents some theorical concepts and definitions for studying metrics on diverse layers of the infrastructure to build the Server, because the're fundamental to design methodology. On chapter 2 we include an analysis of some previous researches developed for some kind of organizations. In chpater 3 is shown the architecture and design of every stage of the current application. The're also shown some operacional details.

Through implementation of this application, chapter 4 establishes that is possible to create a permanent monitorying, showing current server status through some performance graphics and evaluate by some analysis criteria a performance index. This index is going to specify what kind of performance is happening in server.

Chapter 5 shows some tests and their results that actually confirm veracity of all generated data by this application. Tests confirm that desig and implementation stages let reach proposed goals. Finally chapter 6 presents conclusions and some final comments about developing this work. It's also presented some suggestions for future works based on this thesis.

CAPÍTULO 1

Introducción

1.1 PREÁMBULO

Los servidores de aplicaciones han cobrado gran popularidad e importancia en los últimos 10 años. Estos productos son importantes debido a que en el trabajo cotidiano se utilizan como herramientas importantes de intercambio y consulta de información. Como la palabra misma lo indica, un servidor es una computadora (y el software que aloja) que provee un servicio a otros equipos que se denominan clientes.

La necesidad de compartir información entre computadoras surgió con la inherente cualidad de comunicación que posee la computadora, lo que inicialmente llevó a la creación de las redes, cuyo objetivo principal era el de compartir recursos entre todas las computadoras que se encuentran dentro de la red. A partir de las redes se crearon varios métodos para compartir dichos recursos, entre ellos están los servidores Web, correo electrónico y comercio electrónico entre otros.

Un servidor atiende peticiones de servicio emitidas por un cliente, que puede ser una computadora o bien un dispositivo móvil, mediante un programa en ejecución permanente que escucha y responde dichas peticiones.

Un servidor de aplicaciones es un equipo de cómputo con un software que proporciona servicios de aplicación a las computadoras cliente. Además generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

Los servidores de aplicación típicamente incluyen también *middleware* (o software de conectividad) que les permite intercomunicarse con varios servicios, para confiabilidad, seguridad, no-repudiación, etc. Los servidores de aplicación también brindan a los desarrolladores una Interfaz para Programación de Aplicaciones (API), de tal manera que no tengan que preocuparse por el sistema operativo o por la gran cantidad de interfaces requeridas en una aplicación Web moderna.

Los servidores de aplicación también brindan soporte a una gran variedad de estándares, tales como HTML, XML, IIOP, JDBC, SSL, etc., que les permiten su

funcionamiento en ambientes Web y la conexión a una gran variedad de fuentes de datos, sistemas y dispositivos.

Un ejemplo común del uso de servidores de aplicación (y de sus componentes) son los portales Web, que permiten a las empresas la gestión y divulgación de su información, y un punto único de entrada a los usuarios internos y externos. Teniendo como base un servidor de aplicación, dichos portales permiten tener acceso a información y servicios (como servicios Web) de manera segura y transparente, desde cualquier dispositivo.

1.2 ANTECEDENTES

En este apartado se tratará un poco acerca de lo que son las redes de computadoras, Internet y de los servidores de aplicaciones; esto se hace con la intención de proporcionar los fundamentos necesarios para posteriormente utilizar indistintamente de la terminología que implica el Internet y los servidores que brindan servicios a través de el.

Es necesario describir los conceptos básicos de redes debido a que el Internet en si es una red de computadoras. Sin embargo, no se hará un análisis profundo de la teoría de redes, ya que para la comprensión de este trabajo no se requiere un conocimiento amplio en esta área.

1.2.1 REDES DE COMPUTADORAS

Una red de computadoras no es otra cosa que computadoras conectadas entre sí compartiendo recursos (software o hardware).

Las redes surgieron a mediados de los 80's bajo la necesidad de compartir información entre computadoras. Por ejemplo: en una oficina se tienen dos computadoras, una que contiene la nómina y otra la contabilidad. El contador necesitará en algún momento información de la nómina, o bien la persona encargada de la nómina necesitará detalles de la contabilidad. Existe entonces una necesidad de compartir la información y poder acceder a ella sin tener que necesariamente estar frente a la computadora en cuestión.

En la actualidad las computadoras se encuentran listas para el trabajo en red y naturalmente para Internet, y cuentan en algunos casos con varias tarjetas de red. Esto les permite poder trabajar de forma versátil, permitiendo que se conecten en varias redes a la vez.

Los motivos más frecuentes por los que una persona, compañía, institución o escuela pueden desear tener una red de computadoras en sus instalaciones, puede ser algunos de los siguientes:

- Compartir archivos o programas
- Compartir impresoras o escáneres
- Compartir una conexión a Internet
- Establecer políticas de uso y seguridad
- Correo Electrónico
- Mensajería Instantánea Local o Foránea
- Integración operacional del equipo
- Implementación de grupos de trabajo

1.2.2 MEDIOS DE COMUNICACIÓN

Existen diversos medios de comunicación mediante los cuales se puede realizar la interconexión de las computadoras en una red, y estos medios tienen diferentes características. Para la elección de alguno de estos en la implementación de una red en particular se deben considerar los siguientes factores:

- Satisfacer el ancho de banda requerido en la red
- Cobertura de la red
- Ajuste al presupuesto
- Reducir costos de instalación y mantenimiento
- Reducir efectos de interferencia ambiental y climática
- Minimizar las posibilidades de falla
- Permitir la posibilidad de crecimiento de la red

Los medios más comunes y comercialmente factibles son los siguientes:

- Cable Coaxial
- Cable Par Trenzado
- Fibra Óptica
- Ondas Electromagnéticas

1.2.3 INTERNET

En la actualidad el Internet es una herramienta que se utiliza todos los días para diversos propósitos. Algunos de los servicios accesibles mediante Internet son correo electrónico, transferencias bancarias, reserva y compra de boletos de avión, consulta de información, comunicación instantánea entre personas, compra y venta

de libros y muchos más. Toda esta gama de servicios está sustentada en la infraestructura de Internet y dependen naturalmente de ella.

Si se quiere una definición precisa de Internet, esta puede variar dependiendo el enfoque que se le de a la misma. Es decir, la descripción dada define a Internet mediante el uso o servicio que presta, por otro lado, si la definición que se busca es un tanto más teórica, se puede decir que el Internet es un conjunto de computadoras conectadas en red que comparten recursos; de allí la idea de nombrarla como la *red mundial*.

Fundamentalmente el Internet es una red a gran escala, es una red de redes, es decir un conjunto de redes independientes que están interconectadas a nivel mundial mediante ruteadores principalmente, formando así un enorme conjunto de computadoras y recursos compartidos. Estos equipos pueden brindar los mismos tipos de servicios que podrían brindar para una red pequeña, solo que el alcance que tendrán a través de una red de esta magnitud será ciertamente mucho mayor, por lo tanto el servicio será global.

1.2.4 ARQUITECTURAS LÓGICAS DE REDES

En redes existen varias arquitecturas lógicas mediante las cuales la comunicación puede ser posible entre computadoras, la comunicación es el medio mediante el cuál se puede compartir información y servicios. Las arquitecturas lógicas principales son las siguientes:

- Cliente – Servidor
- Punto a Punto
- Maestro – Esclavo

1.2.4.1 MODELO CLIENTE – SERVIDOR

En este modelo, la comunicación generalmente adopta la forma de un mensaje de solicitud del cliente al servidor pidiendo que se efectúe alguna tarea específica. De esta forma el servidor recibe la petición, realiza la tarea y devuelve el resultado de ésta. Regularmente, muchos clientes utilizan un número pequeño de servidores, aunque esto depende del tipo de servicios que requiera el cliente para las tareas que realiza.

En este modelo un servidor atiende peticiones de los clientes, y además el servidor mismo puede ser cliente de algún otro servicio a la vez. En la figura 1.1 se muestra un esquema del modelo cliente – servidor que aclara esta situación.

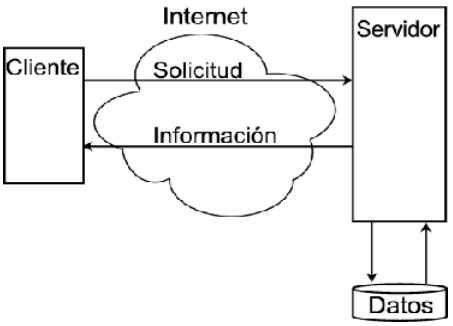


Figura 1.1 Modelo Cliente – Servidor

El cliente envía con la petición los datos que requiere que sean procesados o analizados, el servidor los recibe y efectúa los procedimientos que estén definidos en él, los cuales dependen naturalmente de la naturaleza del servicio que presta en la red. Posterior a esto, regresa el resultado de este procedimiento, los datos procesados los cuales son recibidos por el cliente y así termina la tarea que se encontraba realizando.

Resulta claro que al servidor le toma una cierta cantidad de tiempo y recursos el realizar esta operación, por lo que la capacidad de atención de tareas simultaneas es limitada, y este límite esta establecido por el sistema operativo, el software y hardware del servidor.

1.2.4.2 MODELO PUNTO A PUNTO

Las redes implementadas bajo este modelo consisten en muchas conexiones entre pares individuales de máquinas. Para ir del origen al destino, un paquete puede tener que visitar primero una o más máquinas intermedias. Algunas veces son posibles múltiples rutas de diferentes longitudes, por lo que los algoritmos de enrutamiento desempeñan un papel importante en este tipo de redes.

En este modelo se considera a todos los nodos como iguales, por lo que no existen clientes ni servidores, solo nodos que reciben y transmiten paquetes entre sí. Sin embargo, se puede decir que en este modelo todos los nodos hacer labores de servidor y clientes a la vez, por lo que el procesamiento de datos no se centraliza en un solo punto, por lo que la rapidez del servicio ya no depende de los recursos con los que cuente el servidor.

En la figura 1.2 se muestra un esquema que ejemplifica el modelo punto a punto.

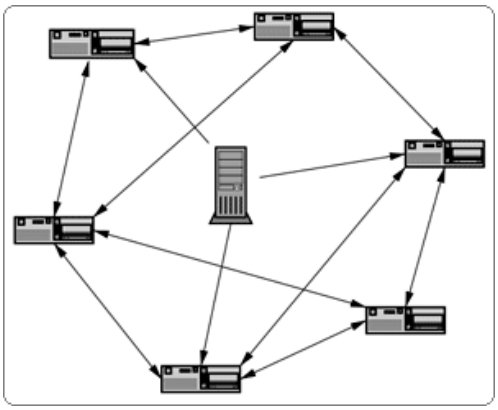


Figura 1.2 Modelo Punto a Punto

En la práctica este tipo de redes resulta muy convenientes, convencionalmente si un cliente posee un archivo que contiene información que muchos otros clientes requieren, necesitarían conectarse todos a la vez al servidor, y este al recibir y atender a todas ellas podría saturar sus recursos con excesivas tareas. En contraste, en este tipo de redes todos los clientes comparten entre si la información que tienen, intercambiando fragmentos de ella. Por esta razón es que en este modelo no existe un cliente en el que todo se encuentre centralizado, el tráfico ocasionado por el intercambio de la información se reparte en toda la red.

1.2.4.3 MODELO MAESTRO – ESCLAVO

En este modelo de comunicación un dispositivo tiene un control unidireccional sobre uno o más dispositivos. Una vez que se ha establecido la relación de comunicación, el control siempre fluye desde el maestro hacia los esclavos.
En algunos casos, cuando varios esclavos no tienen un maestro, uno de ellos realiza esas funciones temporalmente y cede el control a otro esclavo.

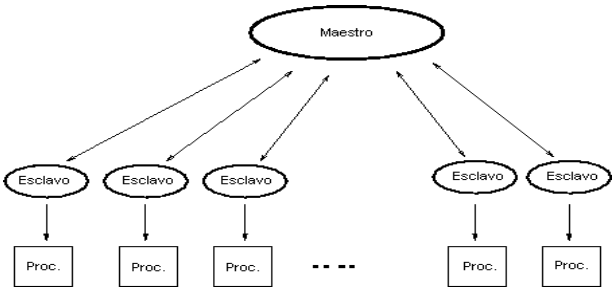


Figura 1.3 Modelo Maestro – Esclavo

En la figura 1.3 se muestra un ejemplo del modelo maestro – esclavo, en el que en la parte superior se encuentra el maestro, debajo sus esclavos con el proceso que realizarán cada uno de ellos.

Este tipo de redes se utilizan en equipos con recursos muy limitados, pero debido a que la adquisición de un equipo con muchos recursos es cada vez más viable, este tipo de red esta en desuso.

1.3 EL PROBLEMA DE MEDIR EL COMPORTAMIENTO

El comportamiento o desempeño en términos computacionales se define como el tiempo de respuesta experimentado por los usuarios al solicitar y recibir algún servicio. La cantidad de trabajo procesado y el tiempo que le toma hacerlo, determina la efectividad del sistema y en forma secundaria la productividad de los usuarios del sistema.

Por lo tanto la medición del comportamiento en un sistema ha cobrado mayor relevancia a partir de que la calidad con la que se brinda un servicio es un factor clave, mediante el cuál una empresa puede ganar o perder clientes, ya que es natural y comprensible pensar que las personas desean recibir el mejor desempeño de un sistema o servicio, y más aún si es que están pagando alguna cantidad de dinero por este, por lo tanto el proveedor se encontrará comprometido en otorgar un servicio de muy buena calidad, con la que el cliente se encontrará satisfecho.

Tradicionalmente cuando una compañía vende algún producto nuevo, menciona entre sus características innatas un mejor desempeño que sus predecesores, no obstante que el vendedor lo asegure no garantiza que así lo sea. De esta forma resulta útil saber cuan bueno es el desempeño que experimenta el producto y verificar que este sea tan bueno como lo aseguran las características nominales.

Por otro lado, un fabricante necesita medir el desempeño de su producto en la fase de desarrollo. Este punto resulta fundamental este punto debido a que en esta fase, los desarrolladores tienen la posibilidad de modificar su producto y ajustar todos los parámetros que sean necesarios para alcanzar el mejor desempeño posible.

Las compañías en la actualidad invierten grandes cantidades de dinero en una infraestructura para sus sistemas computacionales, tanto en sus paquetes de aplicaciones empresariales así como en sus redes computacionales. Debido a esto para ellos el desempeño representa algo muy importante, ya que este afecta directamente a la calidad del servicio que puede prestar a sus clientes.

Estos factores ilustran la necesidad de la medición del desempeño de una aplicación o un sistema, sin embargo, esta no ha sido una tarea sencilla y aunque se han creado varias herramientas y técnicas para calcularlo, el tema no se considera resuelto completamente.

El desempeño en términos muy generales se puede decir que se encuentra directamente en función de diversas métricas que varían dependiendo del tipo de aplicación, por ejemplo, si la aplicación es un servidor de correo el desempeño se mediría en función de la cantidad de correos enviados y recibidos por alguna cantidad de tiempo, tiempo de respuesta y latencia en este caso particular.

1.4 DESCRIPCIÓN DE LA PROBLEMÁTICA

En la actualidad los servidores que proveen servicios a usuarios poseen una gran cantidad de recursos de hardware y en el mejor de los casos una conexión a Internet de alta velocidad. Por estos motivos algunos administradores o personal técnico consideran innecesario preocuparse por el desempeño, sin embargo es necesario mencionar que una gran cantidad de recursos y una conexión a Internet ultra rápida no implican un buen desempeño en el servicio que se preste.

Debido a estas circunstancias, el desempeño del servicio de un servidor es un factor al cuál se le debe dar la importancia pertinente, e idealmente un administrador debe preocuparse por cuan bien se brinda un servicio y no simplemente en si se brinda o no.

Existen muchos factores que pueden ocasionar un pobre desempeño en un servidor, algunos autores han hecho análisis de la importancia de ellos, desarrollando a partir de ese análisis alguna aplicación para medir el desempeño de la red principalmente. A estos factores se les conoce como métricas, que son los parámetros que reflejan el desempeño que experimenta un servidor al prestar algún servicio. En la tabla 1.1 se presentan los artículos, autores y métricas que se han revisado para la realización de la presente investigación:

Autor	Título Articulo	Métrica Definida
Cisco Systems Inc.	Transforming Network Capacity Planning from an Art to a Science [1]	Tráfico en la red
Compuware Corporation	Measuring Application Performance in SOAs [2]	Código Fuente de una aplicación
NetQoS Inc.	Best Practices for NetFlow/IPFIX Analysis and Reporting [3]	Tráfico en la red
NetQoS Inc.	Performance-Based Network Management Keeps Organizations	Desempeño punto a punto, flujos de tráfico, utilización de

	Functioning at Optimal Levels [4]	los dispositivos
Network Physics	Ending the Network vs. Application Blame Game [5]	Paquetes entrantes y salientes, Pérdida de paquetes, Latencia, Tiempo de respuesta del usuario, Tiempo de respuesta del servidor, Retardo de la aplicación ocasionada por la pérdida de paquetes, Tasa de conexión, Número de Conexiones, Conexiones fallidas, Desempeño Punto a Punto, Desempeño del ISP, Historial de rutas y rutas trazadas, Tiempo de configuración de la conexión, Tiempo de respuesta del servidor, Tiempo de transferencia de carga útil, Retardo en la transmisión.

Tabla 1.1 Métricas Generales

Una descripción general de los parámetros analizados por los autores se describe a continuación:

- **Tráfico de una red**. Cuando en la red viajan demasiados paquetes generando una acumulación excesiva, el medio puede no darse abasto para que viajen todos ellos libremente, lo que ocasiona un asentamiento, o bien tráfico, que se ve reflejado en el usuario como una lentitud de servicio. [1]

- **Latencia**. Lapso de tiempo necesario para que un paquete de información viaje desde la fuente hasta su destino. Una alta latencia, naturalmente indicaría una anomalía en el desempeño del servidor. La latencia y el ancho de banda, juntos, definen la capacidad y la velocidad de una red. Una herramienta que puede determinar la latencia es el comando *ping*. [5]

- **Código fuente de un programa**. Mediante el análisis de la ejecución de cierto programa, se puede determinar que instrucciones o fragmentos del programa se ejecutan con mayor frecuencia, así como saber cuales toma más tiempo su ejecución. De esta forma, el programador puede optimizar el programa mediante la modificación del código fuente del mismo en los fragmentos más recurrentes. [2]

- **Desempeño punto a punto**. Consiste en elegir dos puntos en la red y analizar el desempeño exclusivamente de estos. Esta técnica otorga una muestra del desempeño en general de la red. [4]

- **Utilización de los dispositivos**. Mediante la medición de utilización de los dispositivos que se encuentran brindando algún servicio en la red, se puede obtener una muestra del desempeño general de esta principalmente. [4]

- **Flujos de tráfico**. Consiste en analizar los flujos de datos que ocasionan más tráfico dentro de la red, ya que es posible que solo algunos tramos de la red sean los que ocasionan el tráfico en toda ella. De esta forma, se pueden establecer rutas alternas, para que así los paquetes que tienen que llegar a esos puntos tengan una opción más rápida, reduciendo así el tráfico en esos tramos, mejorando el desempeño de la red en general. [4]

- **Paquetes entrantes o salientes**. Este método consiste en medir la cantidad de paquetes que entran y salen de los nodos de la red, o inclusive del servidor. Se puede saber de esta forma el volumen de información que transita en la red o el servidor y tener un parámetro para determinar el uso de la red. [5]

- **Pérdida de paquetes**. La pérdida de paquetes ocurre cuando algunos paquetes transmitidos jamás llegan a su destino, esto puede ocurrir porque un ruteador tenga demasiada carga y se sature, y si algún paquete llega en ese momento entonces el ruteador tendrá que desechar éste o algún otro que se encuentre en espera. La naturaleza del protocolo TCP permite que los paquetes perdidos sean retransmitidos, pero si el porcentaje de paquetes perdidos es muy grande, la retransmisión de todos ellos ocasionaría una congestión en la red, retardando los tiempos de respuesta y decrementando naturalmente el desempeño de la misma. [5]

- **Tiempo de respuesta del usuario**. Es la cantidad de tiempo que le toma a un cliente el enviar una petición al servidor, en algunas ocasiones puede afectar directamente si es que el servidor espera demasiado tiempo por la petición del cliente. [5]

- **Tiempo de respuesta del servidor**. Es la cantidad de tiempo que le toma a un servidor el procesar la llegada de información a través de un paquete, y posteriormente enviarlo al origen. Este parámetro sirve entre otros para determinar el desempeño del servidor. [5]

- **Retardo de la aplicación ocasionado por la pérdida de paquetes**. Cuando la pérdida de paquetes ocurre y una aplicación necesita alguno de ellos para continuar con su ejecución, tiene que esperar a que el paquete sea retransmitido desde el origen y que este llegue satisfactoriamente para continuar con la ejecución. Este fenómeno se ve reflejado como un retardo

de la aplicación, aunque la raíz del mismo es ocasionada por problemas en la red, posiblemente tráfico. [5]

- **Tasa de conexión**. Es la velocidad de transferencia de datos a través del medio de comunicación, se mide en cantidad de bits transmitidos por segundo. [5]

- **Número de conexiones**. La cantidad de conexiones que se han establecido en el servidor para atender alguna petición de servicio. [5]

- **Conexiones fallidas**. Las conexiones que se han establecido y que por algún motivo tuvieron que ser desconectadas o que simplemente no pudieron realizarse en ningún momento. [5]

- **Desempeño ISP**. Este es el desempeño del servicio que otorga el proveedor de Internet, el cuál puede afectar directamente al desempeño que brinde el servidor. [5]

- **Historial de rutas y rutas trazadas**. Bitácora que contiene las rutas que han sido determinadas para los paquetes enviados o recibidos. [5]

- **Tiempo de configuración de la conexión**. Tiempo promedio necesario para el protocolo TCP en configurar la conexión. [5]

- **Tiempo de respuesta del servidor**. Tiempo promedio que le toma el atender y procesar una petición TCP realizada por el cliente. [5]

- **Tiempo de transferencia de carga útil**. Tiempo promedio en entregar carga útil satisfactoria para cada petición IP, esto es sin incluir la retransmisión de paquetes. [5]

- **Retardo en la retransmisión**. Cantidad de tiempo perdida ocasionada por la retransmisión de los paquetes perdidos. [5]

- **Uso del CPU**. Indica cuan ocupada se encuentra la Unidad Central de Procesamiento resolviendo operaciones en todo momento. Una alta carga de operaciones implica que la atención y respuesta del servidor sea más lenta.

- **Uso de memoria RAM**. Indica cuanta memoria están ocupando los programas en ejecución en el servidor, si la cantidad de memoria libre es pequeña, la nueva ejecución de un programa puede ser lenta y por lo tanto la respuesta a alguna petición también.

Todos los parámetros listados son apropiados estimar el desempeño, sin embargo, se selecciona un subconjunto de estas métricas que proporcione un caso genérico y aplicable a todas las circunstancias, además que el análisis implicará una menor cantidad de información.

1.5 OBJETIVOS

1.5.1 OBJETIVOS GENERALES

- Diseñar un mecanismo de evaluación y medir un conjunto de métricas propuestas que permitan esbozar el desempeño que experimenta un servidor de aplicaciones.

- Implementar una herramienta de software, mediante la cuál se podrá auditar la calidad de servicio que está ofreciendo un servidor de aplicaciones.

1.5.2 OBJETIVOS ESPECÍFICOS

- Proponer una selección de métricas que sean capaces de reflejar el desempeño que experimenta un servidor de aplicaciones

- Crear un sistema de monitoreo permanente de las métricas propuestas, para tener datos medidos en cualquier momento que se requieran

- Mostrar datos medidos en tiempo real en el servidor de aplicaciones

- Diseñar e implementar una interfaz de usuario que muestre la medición de las métricas mediante gráficas y sean mostradas por medio de un navegador Web, además que sea sencilla su utilización

- Mostrar la información recolectada a través de un servidor Web de tal forma que permita consultar los datos del monitoreo de las métricas en cualquier parte del mundo a través de Internet

- Diseñar las herramientas de monitoreo de forma tal que consuman la menor cantidad de recursos de hardware y software del servidor, de tal suerte que no alteren el desempeño de éste

- Diseñar la aplicación de manera tal que sea configurable para el usuario, permitiendo especificar algunos parámetros específicos para alguna de las métricas a monitorear

1.6 JUSTIFICACIÓN

Es muy evidente ahora el alcance que ha tenido Internet en el mundo y el impacto en la sociedad, ya que los servicios que se ofrecen a través de el resultan muy atractivos, cómodos, sencillos y seguros para la mayoría de la gente.

Entre los servicios que existen disponibles por los servidores de aplicaciones sobresalen el correo electrónico, sitios Web, telefonía mediante Voz IP, televisión en línea, transferencias bancarias, compras en línea, mapas mundiales, música y videos en línea, mensajería instantánea, acceso a bibliotecas virtuales o electrónicas, visitas virtuales a museos y lugares históricos, entre otros. La figura 1.4 muestra algunos de estos servicios que son ofrecidos a través de Internet.

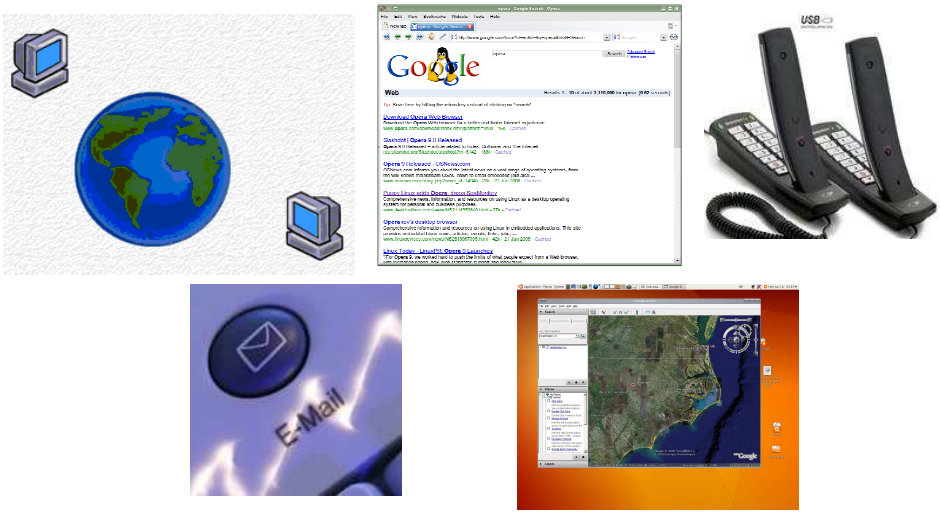


Figura 1.4 Algunos servicios en Internet

Un factor que afecta directamente a la calidad de los servicios que brinda alguno de estos servidores es el desempeño, ya que si un servidor por alguna razón tiene un desempeño pobre, el usuario tendrá un acceso deficiente o nulo al servicio. Por lo tanto el desempeño es un factor que naturalmente se debe vigilar con mucho cuidado, para garantizar un correcto funcionamiento en el servidor y a la postre una buena calidad en el servicio que se presta.

Una labor fundamental de un administrador es verificar que el servidor se encuentre atendiendo eficazmente a las peticiones solicitadas por los clientes, que sean resueltas efectivamente en el menor tiempo posible y sin pérdida de información. Esta labor es realizada comúnmente con una serie de pruebas en el servidor que le permitan tener una idea acerca del desempeño que se experimenta en esos momentos. A través de los resultados obtenidos, el administrador es capaz de estimar el desempeño y detectar alguna falla en el funcionamiento que altere directamente al servicio que se presta mediante él.

El administrador selecciona las pruebas a realizar en función del tipo de servicio que se ofrezca a través del servidor, y mediante éstas se encontrará obteniendo muestras de las métricas involucradas en los servicios brindados. Por ejemplo, si el servidor brinda correo electrónico, la métrica que le interesará es la cantidad de correos que entran y salen, además de la cantidad de ancho de banda consumida por los clientes o bien por los correos que se encuentran ingresando a los buzones de los usuarios.

La intención de la aplicación es simplificar este tipo de tareas administrativas y proporcionar la medición de las métricas que con mayor frecuencia se examinan para discernir la calidad de los servicios brindados. Dicha información será presentada mediante graficas que facilitarán la visualización, comprensión y análisis de los datos, además que serán obtenidos a razón de un periodo de tiempo determinable por el administrador mismo.

1.7 BENEFICIOS ESPERADOS

Los beneficios que se esperan obtener a través del presente trabajo son los siguientes:

- **Presentar una técnica para la medición del desempeño**. Mediante la propuesta se podrá medir el comportamiento de un servidor de aplicaciones.

- **Contribuir a la calidad de servicio QoS**. En la actualidad no basta con brindar un servicio, es muy importante considerar la calidad con la que se presta éste. La presente aplicación permite aportar a la calidad de servicio (QoS) al monitorear las métricas propuestas.

- Presentar al usuario resultados en forma gráfica, para que sean fácilmente entendibles.

- Implementación de una arquitectura con capacidades de modularidad y expansión.

1.8 ALCANCES Y LÍMITES

Se tienen contemplados los siguientes alcances y límites:

Alcances:

- Desarrollar una aplicación capaz de monitorear y medir el desempeño del servidor de aplicaciones

- Mediante el monitoreo permanente, poder hacer un cálculo del desempeño en cualquier momento.

- Desarrollar una interfaz gráfica que le permita al usuario que le permita acceder a los resultados del monitoreo, el índice de comportamiento y las bitácoras con datos antiguos.

- Utilizar características del sistema operativo Unix y empleo de tecnologías de código abierto.

- Realizar pruebas de funcionalidad de la herramienta.

Límites:

- **La propuesta solo considera un conjunto de métricas propuestas que no puede ser expandido a menos que se agreguen módulos para ellas**. Sin embargo, ésta es la propuesta de investigación.

1.9 ORGANIZACIÓN DE LA TESIS

- ✓ *Capítulo 1*. Se describen los antecedentes, así como algunos conceptos teóricos fundamentales de la temática, y la problemática. Se plantean los objetivos, beneficios esperados, alcances y limites. Finalmente se detalla la justificación.

- ✓ *Capítulo 2*. Definición del marco teórico y descripción del estado del arte en investigaciones sobre la medición del desempeño en servidores de aplicaciones.
- ✓ *Capítulo 3*. Se presenta el análisis, así como el diseño de la aplicación mediante diagramas de flujo que la describen a detalle.

- ✓ *Capítulo 4*. Especificación de los módulos implementados: programas de monitoreo permanente y registro de datos e interfaz generadora de gráficas.

- ✓ *Capítulo 5*. Etapa de pruebas y análisis de resultados a la aplicación realizada.

- ✓ *Capítulo 6*. Listado de conclusiones, aportaciones, objetivos alcanzados y trabajos futuros sobre esta línea de investigación

CAPÍTULO 2

Marco Teórico

2.1 INTRODUCCIÓN

La medición del desempeño de los servidores de aplicaciones es una temática que se encuentra en constante desarrollo. Dicho desarrollo ha sido realizado principalmente por empresas particulares con la intención de crear metodologías y herramientas de monitoreo muy robustas para fines comerciales. Estas herramientas son pensadas para administradores de red que dominan muchos conceptos de redes y servidores en general.

Es importante considerar dichas investigaciones para la realización de la presente, con la finalidad de saber cuales han sido los puntos que han sido realizados satisfactoriamente y los que han representado un reto mayor, o inclusive que no se han podido resolver. El objetivo además es indagar en las aplicaciones que ya existen para tenerlas como un antecedente al presente trabajo y justificar así cual es la aportación al campo de la calidad del servicio.

En este capítulo se presenta mediante el estado del arte una visión general acerca de las investigaciones previas acerca de la temática que aborda la presente. A continuación se mostrarán las investigaciones más cercanas a la presente, sustentadas con el desarrollo de una aplicación que apoya a estas.

2.2 ESTADO DEL ARTE

Existen en la actualidad herramientas de medición de calidad en el servicio, que sirven para corregir problemas de configuración, uso y seguridad en servidores y en la red misma. Estas herramientas tienen un carácter comercial propiamente, ya que han sido desarrolladas por empresas que se dedican a vender aplicaciones relacionadas con este tema.

2.2.1 REPORTE Y ANÁLISIS DE TRÁFICO PARA NETFLOW/IPFIX [3]

Esta investigación muestra una técnica de monitoreo de red y como es que el tráfico en ella afecta directamente a los servidores de aplicaciones que se encuentran en ella.

- *Importancia del Análisis del Tráfico*

Esta herramienta se encarga de analizar en tráfico en la red, ya que para los ingenieros de redes que toman decisiones acerca de un cambio en la configuración, es indispensable comprender primero el comportamiento del tráfico. Los datos de la red que hacen posible dicho análisis son los siguientes:

- ✓ Cantidad de tráfico en tiempo real y velocidad de transferencia de datos generados por el protocolo de la aplicación, los hosts, y la comunicación host a host
- ✓ Las fuentes de tráfico, ya sea que estén agrupadas por oficinas, geográficamente, subredes o alguna otra forma útil

Los análisis futuros pueden llevar a una correcta planeación de capacidad, valoración en la prontitud de una red para nuevas aplicaciones de empresas, contabilidad, utilización de la red, identificación y eliminación de tráfico no deseado, planeación e implementación de QoS (calidad de servicio) y presupuesto para equipos de red y soporte.

- *Instrumentación de Red*

La instrumentación de red implica el identificar puntos estratégicos donde los datos a analizar serán recogidos.
Estas tomas de datos son realizadas mediante los siguientes instrumentos.

- ➢ Sondas de Red RMON2

Las Sondas RMON2 son instrumentos dedicados a monitorear los paquetes mientras estos viajan en la red hacia los puntos clave de recolección de datos. Al observar y medir el comportamiento de los paquetes, las sondas recolectan el desempeño del protocolo y las aplicaciones que se ejecutan a través de la red. Las ventajas de estas pruebas son:

- o Captura de Tráfico en Tiempo Real
- o No se introduce una carga adicional a la red
- o La información puede ser recolectada en una amplia gama de protocolos, tales como TPC/IP, UDP/IP, ICMP, IPX y NetBEUI

Las sondas de recolección tienen una estación central de control en la cuál los datos son almacenados por todas las sondas que se encuentran en la red.

- NetFlow/IPFIX

Dada la creciente popularidad en las tecnologías Cisco NetFlow, ha surgido un estándar para los flujos de datos. La IETF (Internet Engieneering Task Force) ha creado un estándar IP Flujo de Exportación de Información (IPFIX).
Con esta tecnología, los ruteadores que existen en la red son utilizados para recolectar datos NetFlow/IPFIX brindando así las siguientes ventajas:

- No se requiere inversión en equipo adicional para la medición
- Bajos costos de implementación
- Fuentes completas de datos
- No se requiere mantenimiento periódico
- La implementación de esta tecnología incrementa la utilización del CPU en los dispositivos configurados entre 1 y 2% solamente

- *Utilización de los datos NetFlow/IPFIX*

Cada flujo de datos es único y tiene siete características con las que puede ser identificado: dirección IP origen, dirección IP destino, número de puerto origen, número de puerto destino, protocolo de la capa 3, tipo de servicio y la interfaz lógica de entrada.
Los datos pueden ser analizados para reportar:

- Los hosts que transmiten más datos en la red
- Los hosts que transmiten más datos con algún otro host
- Las aplicaciones que generan más tráfico en la red
- Volumen de datos por entidad
- Velocidades de transmisión por entidad
- Marcado de ToS (Tipo de Servicio) por aplicación o entidad

- *La solución NetQoS*

La empresa NetQoS ha desarrollado una herramienta para el reporte de datos NetFlow/IPFIX llamada NetQoS Reporter Analizer. Debido a que se desarrolló utilizando las tecnologías de Cisco, esta herramienta esta diseñada para tomar ventaja del conjunto amplio de datos que el estándar de datos ofrece.
Algunas de las capacidades que esta herramienta ofrece son:

- Vista de operaciones de Alto Nivel
- Reportes de Análisis de Tráfico

- Información de Aplicaciones definidas por una combinación de puertos, direcciones IP y ToS
- Reportes en tiempo real y alarmas por minuto para cada interfaz
- Establecimiento de velocidad de transferencia y volumen en función de flujos, paquetes o bytes
- Rastreo del 100% del tráfico en toda la red
- Despliegue de reportes por interfaz, protocolo, host o inclusive conversaciones (comunicación host a host)
- Establecimiento de umbrales a partir de desempeño anterior

Ejemplo de uso de esta herramienta:

Problema: En los últimos 10 minutos los usuarios en las oficinas de Nueva York han comenzado a llamar quejándose porque no pueden acceder al servidor de recursos en el centro de datos de Londres.

Análisis: El ingeniero de red que recibe las llamadas necesita saber como está la red en ese momento, por lo que abre un reporte en tiempo real de la interfaz de red apropiada, tal como muestra la imagen la figura 2.1:

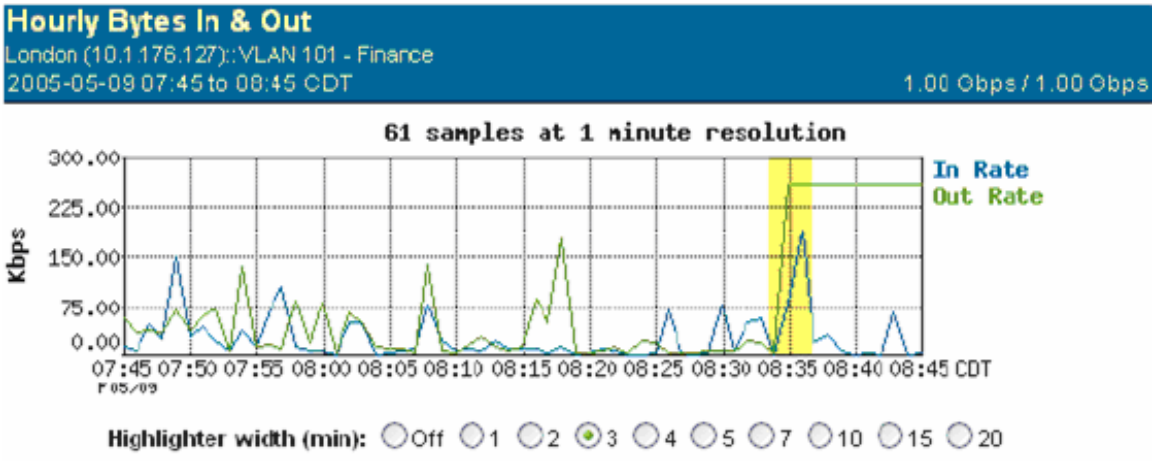
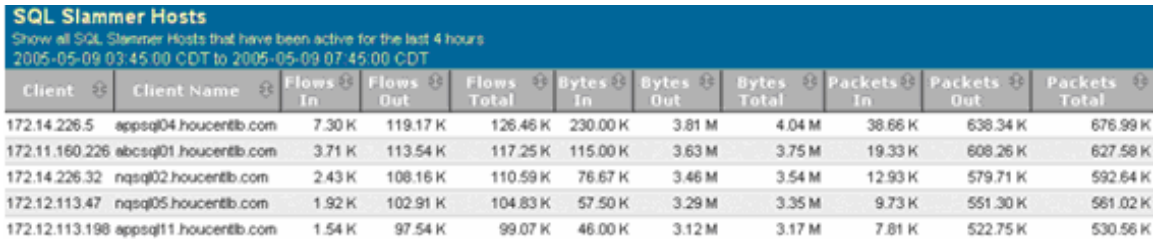


Figura 2.1 Gráfica de bytes entrantes y salientes

El ingeniero ve entonces un pico en la gráfica que corresponde justamente a los datos salientes en los últimos 10 minutos. Selecciona entonces ese fragmento de la gráfica y abre un reporte más detallado para obtener más información. Dicho reporte se muestra en la figura 2.2:

Figura 2.2 Reporte detallado

En esta figura se puede observar entonces que el protocolo que ha sido más utilizado fue el *ms-sql-m*, por lo que el ingeniero concluye que el virus *SQL Slammer* se encuentra activo en la red. El sistema es capaz de mostrarle una lista con los equipos que han sobrepasado el límite normal de uso de ese protocolo en las últimas 4 horas, la figura 2.3 muestra ese resultado:



Client	Client Name	Flows In	Flows Out	Flows Total	Bytes In	Bytes Out	Bytes Total	Packets In	Packets Out	Packets Total
172.14.226.5	appsql04.houcentlib.com	7.30 K	119.17 K	126.46 K	230.00 K	3.81 M	4.04 M	38.66 K	638.34 K	676.99 K
172.11.160.226	abcsql01.houcentlib.com	3.71 K	113.54 K	117.25 K	115.00 K	3.63 M	3.75 M	19.33 K	608.26 K	627.58 K
172.14.226.32	nqsql02.houcentlib.com	2.43 K	108.16 K	110.59 K	76.67 K	3.46 M	3.54 M	12.93 K	579.71 K	592.64 K
172.12.113.47	nqsql05.houcentlib.com	1.92 K	102.91 K	104.83 K	57.50 K	3.29 M	3.35 M	9.73 K	551.30 K	561.02 K
172.12.113.198	appsql11.houcentlib.com	1.54 K	97.54 K	99.07 K	46.00 K	3.12 M	3.17 M	7.81 K	522.75 K	530.56 K

Figura 2.3 Lista de uso de recursos

Solución: En este ejemplo el analizador de reportes le permite al ingeniero ver un reporte del tráfico en la red en la última hora en tiempo real. La razón de este comportamiento inusual es un virus que esta activo en la red, por lo que la solución consiste en aplicar las respectivas actualizaciones a los programas antivirus con lo que se cuente.

Conclusión: Podemos ver que para la realización de esta aplicación, el autor lleva a cabo un monitoreo realizado por las Sondas RMON2 que se encargan de monitorear y recaudar información de las métricas correspondientes. Una vez obtenida la información, la aplicación hace un análisis de todos los datos y los presenta finalmente en una pantalla que será la que el usuario podrá ver.

2.2.2 COMO TRANSFORMAR LA PLANEACIÓN DE LA CAPACIDAD DE UNA RED DE UN ARTE A UNA CIENCIA [1]

- *Antecedentes*

De todas las cuestiones que enfrentan las compañías al administrar sus redes, la planeación de la capacidad es una de las más importantes. Mas que un arte es recientemente una ciencia, ya que la planeación implica todo acerca del balance que se necesita para cubrir las expectativas del usuario en contra de la realidad del presupuesto con el que se cuenta.

- *QoS: Una alternativa*

Un sorprendente número de ingenieros de red en empresas grandes creen que las redes construidas con switches de alta capacidad, redes LAN o WAN de alta velocidad no requieren administración de QoS en lo absoluto. Ellos creen que el mayor ancho de banda que se posea implica menor administración de QoS.

El uso de QoS no incrementa el ancho de banda, sin embargo, ayuda a reducir los picos y valles (cuando no existe un uso del ancho de banda) en el uso de la red. De esta forma QoS provee un rendimiento con mayor consistencia desde el punto de vista de los usuarios.

Desde un punto de vista de planeación, el desarrollo uniforme de QoS uniformemente a través de la red, protege las aplicaciones de tiempo real, garantizando el ancho de banda y una baja latencia, ya que los picos frecuentes pueden afectar directamente al desempeño.

- *Solución*

Categorizar el tráfico en la red. Los ingenieros de red de Cisco iniciaron sus esfuerzos en mejorar la planeación de red creando categorías del tráfico de red en tres tipos:

- Legítimo, tráfico ocasionado por uso de negocios
- Tráfico Inapropiado
- Tráfico no deseado

- *Normas de utilización y medición*

Los planeadores de Cisco creían que era vital establecer normas de utilización y medición, de forma que sirvieran como base para la administración de la capacidad de la red. El equipo de planeación encontró que las normas iniciales basadas en sus experiencias en el campo eran las apropiadas para casi todas las oficinas de ventas de Cisco.

- *Las herramientas*

Cisco caracterizó, analizó y detectó anomalías en los flujos de tráfico de red utilizando la tecnología Cisco IOS NetFlow, además el Analizador de Reportes NetQoS. Esta aplicación utiliza los datos recaudados por NetFlow para crear un reporte del tráfico en la red.

La aplicación NetFlow se ha convertido en la tecnología principal en detección de anomalías de red en la industria. Tanto así que en el año 2003, Cisco IOS NetFlow versión 9 fue elegido como un estándar, el Flujo de Exportación de Información (IPFIX). IPFIX define el formato mediante el cual el flujo de información IP es transferido desde un exportador, como puede ser un ruteador Cisco, a una aplicación que analiza los datos. La figura 2.4 muestra un escenario de uso de la aplicación NetFlow:

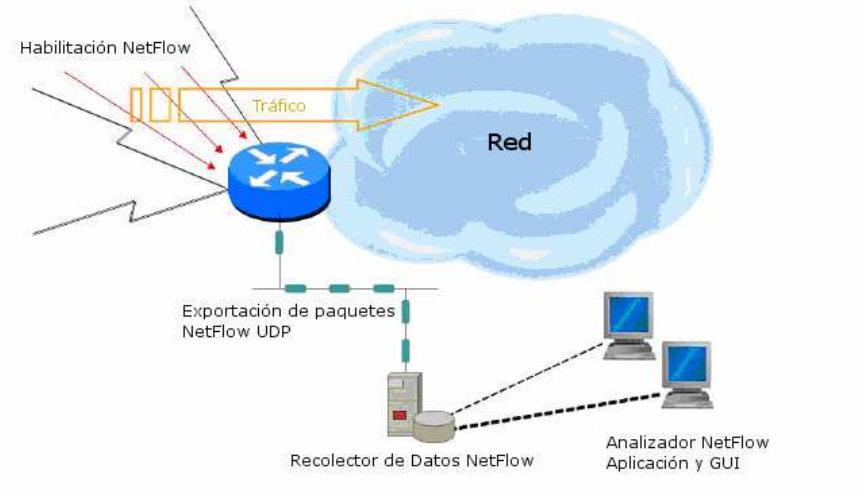


Figura 2.4 Esquema del uso de NetFlow

- *Presentación de Reportes*

Cisco realiza frecuentemente planeaciones de capacidad en sus oficinas, sin embargo no existe actualmente un método general para analizar cada circuito. Los reportes creados por NetFlow y adicionalmente las normas de utilización y medición, ayudan a los planeadores a determinar cuales circuitos deberán revisar primero y que tanto aumento del ancho de banda es necesario.

- *Resultados*

Cuando el equipo de planeación de capacidad determina si es necesario incrementar el ancho de banda, se hace una recomendación a los ingenieros de red de Cisco, quienes la revisan. Una vez que se determina que en efecto es necesario un incremento en el ancho de banda, el procedimiento se vuelve una decisión y trámite meramente administrativos.

Conclusión: Para la realización de esta aplicación, el autor concluyó que una forma para planear la capacidad de una red es categorizar el tipo de tráfico que circulará en ella, para posteriormente poder monitorearla y analizarla. Si bien el tráfico no es algo que se pueda controlar directamente, en este estudio realizado se demuestra que una categorización del tráfico, una etapa de análisis de cada una de estas y un reporte detallado del análisis ayudan al administrador a saber donde se encuentra el punto que genera más tráfico en la red.

2.2.3 EL DESEMPEÑO ES PRIMERO [2]

Esta investigación presenta una metodología que consiste en el monitoreo de tres métricas para medir el desempeño de un servidor de aplicaciones.

- *Administrar la Disponibilidad de la Red*

En las ultimas dos décadas los ingenieros se han concentrado en administrar la disponibilidad de la red. Hoy en día los productos de red y los vendedores de sistemas de administración se concentran principalmente en decirles a los administradores de red si los componentes de la infraestructura están funcionando o no.

Sin embargo, la mayoría de las empresas y los proveedores de servicios de red operan con un 99.9% de tiempo útil o incluso más. Una de las razones por la que esto ha ocurrido es que esta tecnología se ha vuelto más fiable en los últimos años.

- *Administración de Red y Desempeño de Aplicaciones*

Mientras que la viabilidad en las redes se ha mejorado, las cuestiones relacionadas con el desempeño han decrecido dramáticamente. La mayor influencia de esta tendencia es que ahora la expectativa del usuario es para un servicio instantáneo de la red. Otras influencias que han incrementado el volumen y complejidad del tráfico, y la necesidad de monitorear que tan bien las aplicaciones son las siguientes:

  - Consolidación del centro de datos
  - Incremento en el número de usuarios remotos
  - El incremento de tráfico de voz y video
  - Aplicaciones heredadas
  - Software como un servicio
  - Aplicaciones más complejas que incluyen Arquitecturas Orientadas a Servicio (SOA)

La convergencia del incremento del uso de WAN con la disponibilidad de dispositivos mejorados, es guiada por los ingenieros de red para darle prioridad al desempeño cuando se trate de administrar redes muy complejas.

- *El Caso de un Acercamiento a una Nueva Administración*

Recientemente un grupo Yankee reportó que el título "desempeño es el nuevo mandato para la administración de redes" se refiere al estudio de administración de aplicaciones en empresas. El estudio encontró que las empresas reportan una caída en el desempeño del 14% en promedio cuando experimentan problemas con las aplicaciones. De esta forma el lema "El Desempeño Primero" se está volviendo un nuevo estándar en la administración de redes.

- *Una nueva filosofía en la Administración de Redes: El Desempeño Primero*

El paradigma *Desempeño Primero* utiliza monitoreo a través de la visibilidad sobre las aplicaciones que se ejecutan sobre la red. Este acercamiento es conducido por el propósito fundamental de la red, transportar datos de un punto a otro tan rápido

como sea posible. El tiempo que tarda en llevarse a cabo esto es la mejor forma para decidir como se puede optimizar la red, así como planificar actualizaciones e identificar cuan severo y profundo puede ser un problema en la red.

Concentrarse solo en el desempeño de las aplicaciones mas importantes que corren sobre la red, es la forma en la que las organizaciones y los ingenieros de red pueden saber en que punto concentrar sus herramientas: informar sobre inversiones en la infraestructura, entrega consistente, tiempos de respuesta aceptables y la resolución de los problemas que impactan en los negocios más fuertemente.

El paradigma *Desempeño Primero* requiere monitorear tres métricas: desempeño punto a punto, flujos de tráfico y la disponibilidad y utilización de los dispositivos. La figura 2.5 muestra un ejemplo de esta situación.

Figura 2.5 Métricas de monitoreo

El monitoreo de punto a punto mide cuan bien la red está entregando los servicios a los usuarios finales y provee una vista de lo que está pasando en la red.

Con las métricas del desempeño punto a punto capturadas en la fuente y fuente de latencia aislada, los análisis posteriores son más específicos. El análisis del tráfico permite a los ingenieros de red entender su composición en enlaces específicos en los que la latencia es más grande de lo normal.

Si la fuente de la latencia es aislada a un componente de infraestructura, los administradores de la red necesitan administrar el desempeño del dispositivo en cuestión y decidir que acción será tomada.

La tecnología de exportación de información NetFlow/IP, que se encuentra integrada en los ruteadores y switches de las empresas hoy en día, hacen posible el análisis del tráfico sin necesidad de invertir dinero en costosos equipos.

- *Los beneficios del paradigma "El Desempeño Primero"*

Este paradigma ayuda a los ingenieros de red a optimizar la entrega de servicios de aplicación críticos a los usuarios, mitigando así los riesgos por un cambio y haciendo más eficiente el uso de los recursos. El paradigma provee medios para:

- Demostrar el desempeño de las aplicaciones que corren sobre la red
- Entregar desempeño de aplicaciones consistentes y medirlos
- Disminuir los riesgos de cambios planeados y eventos inesperados
- Hacer un informe de las inversiones en la infraestructura
- Trabajos colaborados y mas eficientes
- Rápida resolución de problemas

- *Centro de Desempeño NetQoS*

Esta herramienta ha introducido productos que apuntan a las métricas más importantes ya mencionadas anteriormente: desempeño punto a punto, análisis del flujo del tráfico y desempeño de dispositivos.

El centro de desempeño NetQoS es un portal de administración que integra tres productos, permitiendo así a los administradores de red que vean un conjunto de métricas específicas que necesitan, ya sea en tiempo real o en bitácoras, todo en una interfaz basada en Web. En la figura 2.6 se muestra la pantalla principal de la aplicación.

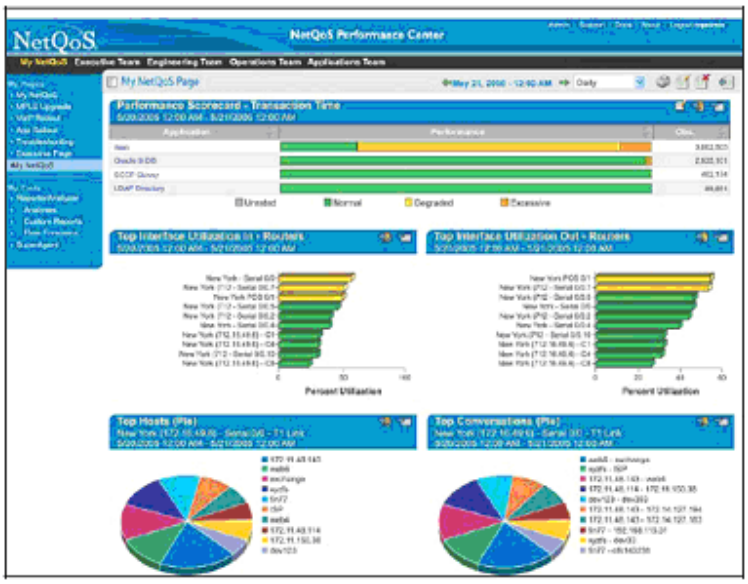


Figura 2.6 Pantalla principal del Centro de Desempeño NetQoS

Para entender el valor que esta herramienta ofrece, es necesario comprender las contribuciones de cada módulo del producto.

- *Módulo de Desempeño Punto a Punto – El Súper Agente*

La medida más útil para medir y establecer desempeño punto a punto desde el punto de vista del usuario es el tiempo de respuesta. El súper agente monitorea todos los paquetes de aplicaciones TCP desde la red al centro de datos y viceversa, esto provee una forma de medir el tiempo de entrega en la red, tiempo de respuesta del servidor, tiempo de transferencia de datos entre otras.

Con el súper agente, un solo dispositivo en el centro de datos puede reportar el desempeño de aplicaciones para todos los usuarios en sus ubicaciones. Además, puede medir la latencia en una aplicación, un servidor y componentes de red comparadas con métricas normales que fueron tomadas por el agente previamente de forma automática.

- *Análisis del Flujo de Tráfico en la Red – Analizador de Reportes*

El tener una visión de la composición del tráfico en cada enlace en la red, le brinda a los ingenieros de red toda la información que necesitan para planificar la instalación de una nueva aplicación o la adición de nuevos usuarios, de forma que puedan tomar decisiones inteligentes en la configuración de los ruteadores, determinación del ancho de banda y actualizaciones. En la figura 2.7 se muestra la pantalla principal de los análisis de reportes.

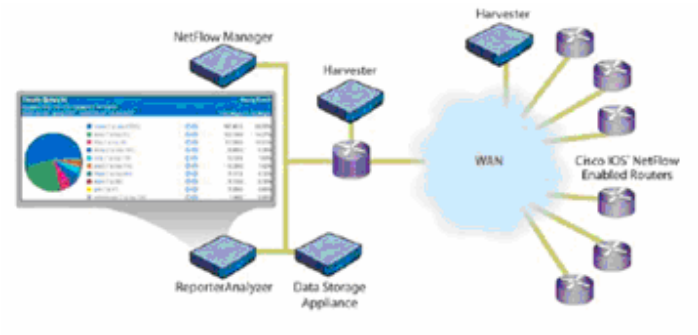


Figura 2.7 Pantalla de Análisis de Reportes

- *Desempeño del Dispositivo – NetVoyant*

Esta herramienta provee métricas de desempeño basadas en SNMP para infraestructura de red, dispositivos y servicios, además que brinda la capacidad de importar y reportar en bases de administración de la información (MIB's).

Adicionalmente ayuda a los ingenieros de red y a los administradores de operaciones a resolver problemas rápidamente, ya que solo analizan las causas que ocasionan el deterioro del desempeño de los dispositivos. También ayuda a los ingenieros de red a administrar la red y la capacidad de lo servidores mediante una comparación del desempeño actual con datos históricos, calculo que se hace de forma automática. Finalmente en la figura 2.8 se muestra la pantalla de análisis de NetVoyant.

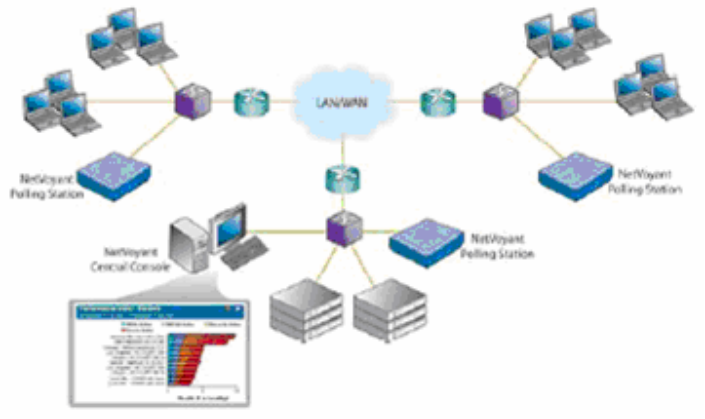


Figura 2.8 Pantalla de Análisis de NetVoyant

Conclusión: En este trabajo el autor concluyó que el paradigma *El desempeño es Primero* puede ayudar mucho a los ingenieros de red a optimizar la entrega de servicios de aplicación críticos a los usuarios. Esto se lleva a cabo mediante el monitoreo y análisis de tres métricas que el consideró como más importantes. Una aplicación que implementa la metodología propuesta, lleva a cabo el análisis y la presentación de los resultados al usuario mediante algunas gráficas que muestran un panorama general del estado en la red, así como el desempeño en la misma.

2.3 RESUMEN

En este capítulo se analizaron algunas investigaciones publicadas sobre la determinación del desempeño y la calidad del servicio. Sin embargo y como pudo notarse, el mayor interés de ellas es el medir el desempeño de la red y algunos protocolos particulares para aplicaciones específicas.

Es aquí donde radica la mayor diferencia y ventaja de la presente investigación a diferencia de las existentes, ya que la intención de esta es tener un esquema genérico para cualquier tipo de servidor de aplicaciones, ya sea correo electrónico, sitio Web, bases de datos, transferencias de archivos (FTP), etcétera o inclusive varios de ellos.

El análisis de las investigaciones previas es un paso fundamental para la realización de una, ya que de esta forma es como se puede compartir el conocimiento y experiencia obtenida por otros investigadores o grupos de investigación, con el fin de integrar todas ellas en una investigación cuyo alcance sea mayor que si fuese individual, y finalmente mediante la publicación, proporcionar a la comunidad el resultado obtenido a través del desarrollo de la investigación realizada.

La técnica que ha sido utilizada por los autores de los trabajos analizados para la determinación del desempeño de los servidores de aplicaciones o la red consiste de una selección de métricas, una etapa de monitoreo, análisis y finalmente presentación de resultados. Las aplicaciones resultantes presentan sus conclusiones y resultados mediante gráficas que ayudan al usuario a un veredicto fácilmente entendible.

CAPÍTULO 3

Análisis y Diseño de la Aplicación: MoniTool

3.1 INTRODUCCIÓN

En este capítulo se hace un análisis de los factores más importantes que involucran al presente trabajo. Dichos factores son sin dudarlo las métricas, y en este capítulo se hará el análisis de cada una de ellas y una justificación de la selección de un conjunto, el cuál será a la postre gran parte de la propuesta del siguiente trabajo.

Una vez hecho el análisis y haber definido el conjunto de métricas a emplear, será posible entonces realizar el diseño de los módulos medidores y la interfaz que mostrará los resultados mediante gráficas.

Finalmente se especificarán las herramientas de cómputo que se utilizarán para la realización de los módulos y la interfaz gráfica.

Como se puede concluir, este capítulo es de gran importancia para el desarrollo del presente trabajo de investigación, por lo que debe tenerse presente la importancia que le corresponde.

3.2 EVALUACIÓN DEL DESEMPEÑO DE LOS SERVIDORES DE APLICACIONES

La forma en como se medirá el desempeño en un servidor de aplicaciones es una tarea clave y quizás la más importante de la tesis, por esta razón es que debe quedar muy clara la forma en la que se llevará a cabo.

De acuerdo a los trabajos analizados en el capítulo 2, se puede extraer la metodología que ha sido utilizada por los autores [1][2][3]. Por lo tanto, la forma en que se medirá el desempeño de los servidores de aplicaciones en la presente tesis, será de la siguiente forma:

- **Selección de las métricas**: En este punto se seleccionan algunas métricas que deben reflejar correctamente lo que se quiere medir. En el capítulo 1 se describieron las diversas métricas que se analizan para hacer estimaciones de desempeño, sin embargo, no todas estas métricas resultan útiles este trabajo, debido a que muchas de ellas describen el comportamiento de

aspectos diversos que no son el objetivo de la presente tesis. Así, para saber el comportamiento de un servidor de aplicaciones resulta indispensable y muy importante elegir las métricas adecuadas que lo reflejen de la forma más acertada posible y además no saturar con información que resultase innecesaria, poco útil o inclusive redundante.

- **Monitoreo**: El monitoreo es el responsable de recaudar todos los datos correspondientes a las métricas seleccionadas previamente. El monitoreo es llevado a cabo por un programa que vigila un parámetro o métrica por un tiempo indefinido, y que obtiene datos de él.

- **Análisis:** Una vez que los datos hayan sido capturados habrá un análisis de ellos para poder determinar el desempeño en el servidor de aplicaciones. Este análisis será el responsable de medir el desempeño de acuerdo al criterio establecido para calcularlo.

- **Presentación de resultados**: Esta es la etapa que se encarga de presentar los resultados del monitoreo y análisis previos. Justo aquí es donde se notifican los resultados y lo común es hacerlo de una manera sencilla, entendible y resumida (de acuerdo a los trabajos analizados en el capítulo 2 [1][2][3]) ya que en la mayoría de los casos al usuario final no le interesarán demasiado los detalles técnicos, a menos que un problema lo requiera.

3.3 ANÁLISIS Y SELECCIÓN DE MÉTRICAS

Las métricas seleccionadas para analizar por la aplicación, así como la razón por la cuál son seleccionadas son las siguientes:

- **Latencia**: Esta puede determinar cuán accesible puede ser el servidor desde cualquier punto de la red, si la latencia resulta ser muy elevada, el servicio que se preste será limitado, así que es una métrica muy importante a tener en cuenta. La latencia es una métrica que refleja muy bien cuando una red se encuentra muy saturada o congestionada, y a los paquetes les toma más tiempo transitar del origen al destino y viceversa, o inclusive podrían hasta nunca llegar a su destino. Debido a que la latencia refleja la disponibilidad del servidor hacia la red, resulta fundamental incluir esta métrica en el análisis.

- **Número de conexiones**: Un servidor puede ser configurado para atender una cantidad ilimitada de peticiones, sin embargo en la práctica no ocurre así, ya que el número se determina propiamente en función de la capacidad y características del hardware, software y red, y como éstos recursos son

finitos y limitados, por ende el número de conexiones también lo será. Si se rebasa el límite natural de conexiones, los usuarios que soliciten servicio posteriormente serán rechazados. De esta forma es muy importante saber cuantos llamados se atienden en todo momento para tener muy presente la cantidad de requisiciones que se pueden responder en ese momento, y saber si el número de conexiones establecidas está dentro de los parámetros óptimos de uso y no crear así una saturación en el servidor, que ocasionaría una nula respuesta posterior de servicio. Inclusive es útil para determinar si el servidor está bajo un ataque de "Denegación de Servicio", el cuál es una saturación en el número de peticiones que el servidor es capaz de atender, lo que ocasionaría una respuesta negativa a cualquier petición posterior. Todas estas situaciones justifican la inclusión de esta métrica.

- **Paquetes entrantes o salientes**: Comúnmente esta métrica sirve para saber si una red se encuentra congestionada, sin embargo es importante monitorearla debido a que también se puede saber si es que un servidor se encuentra sobrecargado de información entrante o saliente, de forma que el servicio no sea lo suficientemente rápido para atender todas las peticiones. Puede además notarse existe una gran diferencia entre la cantidad de paquetes que entran y salen, o si han dejado de circular a través de el, lo cuál ocurriría si el servidor ha dejado de atender a las peticiones de servicio o inclusive si la seguridad en el servidor ha sido corrompida. Adicionalmente, se puede hacer una comparativa entre los paquetes que entran y salen del servidor, y determinar si todas las requisiciones han sido atendidas y devueltas al cliente, si es que se tiene una idea en la cantidad de intercambio de información que hay entre el cliente y el servidor. Estas razones hacen meritoria la inclusión de esta métrica en el análisis.

- **Paquetes Perdidos**: Cuando los paquetes viajan de un punto a otro en la red, ocasionalmente el tráfico que existe en ella impide que estos se trasladen rápidamente o inclusive pueden hasta perderse en el camino. Cuando esto ocurre, la información llega incompleta y es necesaria una retransmisión del paquete perdido, lo que naturalmente toma tiempo, además que hay posibilidad de que el paquete se pierda otra vez. Todo esto se ve reflejado como una respuesta lenta a las peticiones de servicio. Esta situación naturalmente afecta directamente al desempeño, razón suficiente por la será tomada en consideración.

- **Tasa de conexión**: La velocidad de transferencia de datos entre el servidor y el cliente es un factor importante para determinar la calidad de servicio, ya que esta refleja cuan rápida es la respuesta a la requisición solicitada por el cliente. Adicionalmente, a través de la tasa de conexión se puede saber cuanto ancho de banda de la red está ocupando el servidor en atender

alguna tarea en particular, lo que ocasionaría que las futuras conexiones experimenten una lenta respuesta, demora en el servicio o inclusive la negación del mismo. Por estos motivos resulta fundamental la selección de esta métrica para tenerla en cuenta en la estimación del desempeño del servidor e inclusive para saber si estuviese consumiendo el ancho de banda disponible en la red.

- **Uso de CPU**: Cuando el procesador se encuentra muy ocupado atendiendo operaciones todo el tiempo, los servicios se encontrarán en la necesidad de ser encolados para esperar su turno a ser atendidos y finalmente dar una respuesta a la requisición hecha con anterioridad. Cuando esto ocurre, el cliente experimentará una respuesta lenta a su petición de servicio, y naturalmente a mayor cantidad de uso de CPU implica mayor tiempo de espera de respuesta en la atención de los servicios. Este es el motivo principal por el cuál esta es considerada como una métrica digna de un análisis permanente.

- **Uso de RAM**: Cuando los procesos en ejecución en el servidor consumen mucha memoria, el sistema operativo encontrará dificultades en ubicar espacios vacíos en la memoria para los procesos que se ejecutarán posteriormente. En este caso, y aunque el procesador se encuentre relativamente libre, el procesamiento será lento debido a que no será posible encontrar memoria libre con facilidad, tomándole más tiempo guardar los resultados en ella, lo que afectará natural y directamente al tiempo de respuesta hacia los procesos y por consecuencia a los clientes que soliciten servicio. Por esta razón se elije esta métrica como un factor que será necesario un análisis permanente.

3.4 JUSTIFICACIÓN DE LAS MÉTRICAS SELECCIONADAS

Es importante mencionar que un servicio puede afectar directamente una métrica, y otro servicio alguna otra. Por esta razón es que todas las métricas descritas en el capítulo anterior resultan importantes. Debido a que seleccionar un gran número de métricas que cubran todos los servicios que pudiera un servidor brindar significaría una cantidad abrumadora de información, e inclusive consumiendo recursos del servidor mismo, se deben elegir por lo tanto un conjunto de ellas. Además pudiera ser que en la herramienta se estuviese analizando una métrica para un servicio que no se preste en el servidor en particular.

De esta forma, la selección de estas métricas son los parámetros que en conjunto reflejan de la manera más acertada el comportamiento de un servidor de aplicaciones. Se considera entonces que resultan ser los parámetros más

descriptivos de un servidor de aplicaciones en términos muy generales, es decir, para cualquier tipo de servicio o servicios simultáneos que se brinden en el servidor en cuestión, y que estos serán cubiertos por el análisis de alguna de las métricas seleccionadas.

Es claro entonces que el acierto en la estimación del desempeño que se puede medir mediante la presente aplicación, y su veracidad depende directamente de la selección de las métricas correspondientes. Aquí surge la importancia de una selección apropiada de las mismas.

A pesar de que la latencia y los paquetes perdidos son parámetros de la red y no del servidor propiamente, es muy importante considerarlos ya que comprometen directamente la calidad del servicio que ofrece el servidor, por lo que es necesario saber su estado a pesar de que no se puedan controlar. El monitoreo de estos parámetros le permitirá al administrador notificar al responsable (si es posible) de algún problema en la red o deslindar responsabilidades.

3.5 DESCRIPCIÓN GENERAL DE LA APLICACIÓN: MONITOOL

En este trabajo de tesis se realiza una herramienta de software que sea capaz de medir el desempeño de un servidor de aplicaciones en una red, primordialmente Internet. La estimación se determina a partir del monitoreo permanente de algunas métricas específicas, las cuales han sido seleccionadas cuidadosamente para que reflejen en las estadísticas el desempeño que experimenta en todo momento el servidor.

MoniTool es una herramienta de software configurable, ya que mediante un archivo de configuración permite al administrador ajustar el comportamiento de esta de acuerdo a las necesidades particulares de su ambiente de red o servicios brindados mediante el servidor.

Monitool es un sistema que es capaz de recopilar información de las métricas definidas, registrar en bitácoras, controlarlas, desplegarlas y hacer una gráfica de los datos recaudados, de forma que al administrador del servidor en cuestión le resulte muy sencillo el interpretarla al verla en pantalla. Además cuenta con una interfaz Web mediante la cuál puede revisar los datos desde cualquier computadora en la red o inclusive en algún punto remoto de Internet.

La aplicación MoniTool dejará a juicio del administrador del servidor el calificar el desempeño de éste, utilizando toda la información que obtendrá de las gráficas que se trazan mediante el monitoreo de las métricas seleccionadas. Las gráficas solo son el medio que permite determinar la eficiencia del desempeño, y no el juez

que lo determina por si mismo mediante ellas. Adicionalmente debe considerarse que dependiendo el tipo de servicios que ofrezca un servidor, la magnitud de las métricas varía de acuerdo a éstos, por lo que un servicio puede reflejar un valor en alguna métrica mientras que otro servicio reflejaría un valor distinto en la misma métrica.

3.6 DESCRIPCIÓN DE LOS MÓDULOS A DESARROLLAR

La arquitectura consta principalmente de cuatro grandes módulos. El primero de ellos consiste en la recopilación y registro en bitácoras de los datos en cada una de las métricas, el segundo controla los archivos de bitácoras, el tercero visualiza las bitácoras en la interfaz Web y finalmente el cuarto presenta los resultados mediante la generación de las gráficas descriptivas para cada una de las métricas.

La estructura general del sistema es representada mediante el diagrama 3.1:

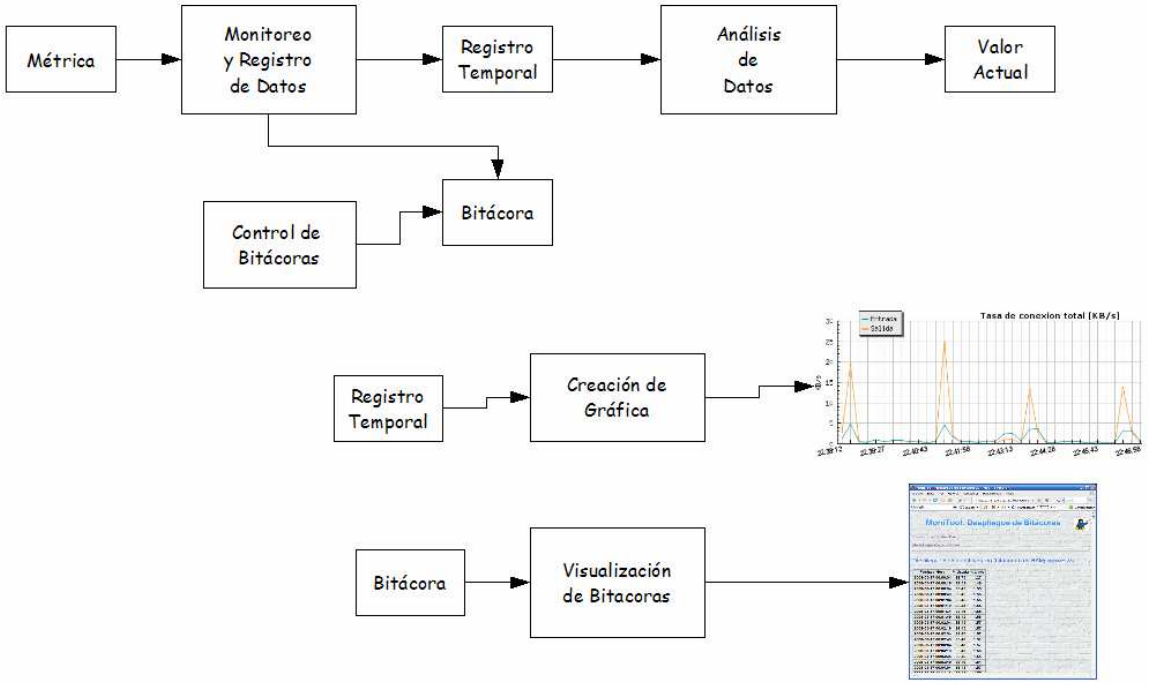


Figura 3.1 Esquema General de la Aplicación MoniTool

Este esquema es la estructura general, la cuál se implementa para cada una de las métricas seleccionadas, lo que significa que cada métrica tiene cuatro módulos de estos para un completo funcionamiento independiente y modular.

Se puede observar que la aplicación esta seccionada en dos partes independientes, en la primera de ellas se lleva a cabo el monitoreo, registro de datos y desplazamiento de las bitácoras y en la segunda la visualización de bitácoras y generación de gráficas.

3.6.1 MÓDULO DE MONITOREO Y REGISTRO DE DATOS

Este módulo realiza el trabajo más complejo de la aplicación, el cuál consiste de dos fases que monitorean la métrica correspondiente y obtiene un dato nuevo, se guarda en un registro temporal y se registra de bitácoras. En la figura 3.2 se ilustran las fases que este módulo contiene.

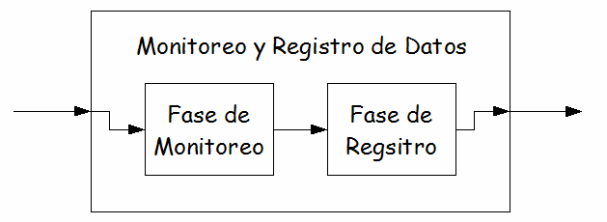


Figura 3.2 Fases intermedias del módulo de Monitoreo

3.6.1.1 FASE DE MONITOREO

Esta fase consiste en una serie de programas e instrucciones que vigilan alguna métrica y obtienen una medición de la misma por una unidad de tiempo definida por el usuario en un archivo de configuración. La función fundamental en esta fase es recopilar información, y se hace mediante un programa en ejecución permanente en el servidor, el cual vigila y recopila los datos que le corresponden de acuerdo a la métrica respectiva y lo almacena en un archivo temporal, que será leído por la siguiente fase. La figura 3.3 muestra un esquema que detalla esta fase.

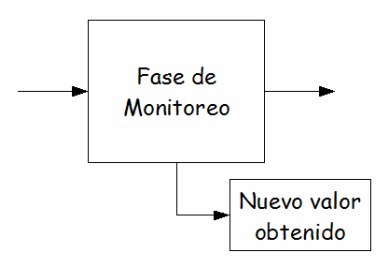


Figura 3.3 Detalles de la Fase de Monitoreo

3.6.1.2 FASE DE REGISTRO

Esta fase lee el valor obtenido por el monitor desde un archivo temporal y lo agrega al registro temporal de datos, así como al archivo bitácora con una marca de tiempo de ese momento. La figura 3.4 muestra detalles del diseño de esta fase.

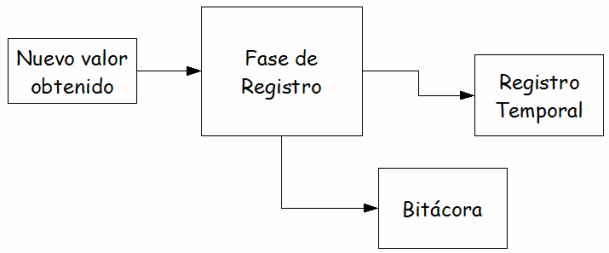


Figura 3.4 Detalles de la Fase de Registro

3.6.2 MÓDULO DE CONTROL DE BITÁCORAS

Existen seis niveles de bitácoras en los que se almacenan los datos registrados de acuerdo a su antigüedad.

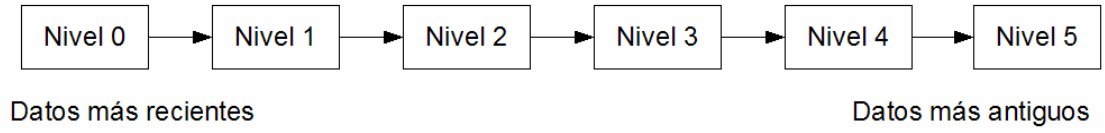


Figura 3.5 Jerarquía de los archivos de bitácoras

Este módulo consiste en un programa que vigila la fecha, en cuanto detecta la transición de un día a otro desplaza los datos contenidos en las bitácoras hacia los archivos que almacenan información de días anteriores. De esta forma en el nivel cero siempre estarán datos del día actual, en el nivel uno datos del día anterior, en el nivel dos datos de dos días antes y así sucesivamente.

Mediante un archivo de configuración, MoniTool permite elegir al usuario si desea que en el nivel cinco se acumulen todos los datos antiguos que han sido desplazados, o bien que sean descartados una vez que hayan alcanzado este nivel.

3.6.3 MÓDULO DE ANÁLISIS DE DATOS

El módulo de análisis toma calcula un promedio de los datos registrados últimamente contenidos en el registro temporal para obtener un estado reciente de

la métrica. Este promedio es comparado posteriormente con un valor máximo definido previamente por el usuario, si es mayor entonces se dice que el valor ha sobre pasado el umbral óptimo, por lo que habrá que reportar esta anomalía. La figura 3.6 muestra un esquema de este módulo.

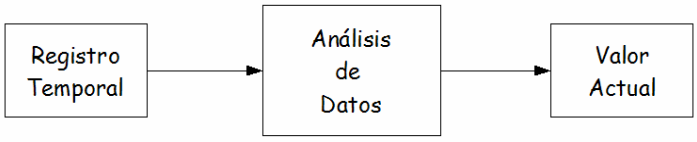


Figura 3.6 Módulo de Análisis de Datos

3.6.4 MÓDULO GENERADOR DE GRÁFICAS

Este módulo se encarga de generar las gráficas de desempeño a partir de los datos que le son entregados en la fase de registro, y son ajustadas de acuerdo a estos. De tal suerte se calcula dinámicamente la escala para que los datos se acomoden adecuadamente en la gráfica.

El registro temporal es el que se emplea en este módulo para realizar la gráfica. En aquella fase los datos se recorren conforme se genera uno nuevo, de forma que en la primera posición del registro siempre se encontrará el dato más reciente del monitoreo. En términos de la gráfica significa que los datos más nuevos estarán en la orilla derecha de ésta.

Una vez leídos los datos desde el registro temporal, la gráfica es elaborada y enviada a la interfaz Web. La figura 3.7 muestra detalles de este módulo.

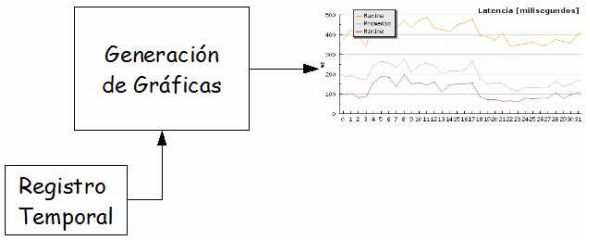


Figura 3.7 Módulo Generador de Graficas

3.6.5 MÓDULO VISUALIZADOR DE BITÁCORAS

Este módulo despliega los archivos de bitácoras en la pantalla mediante una aplicación Web, permitiendo elegir al usuario la métrica y el nivel de antigüedad de los datos que desee consultar. La figura 3.8 muestra detalles del módulo.

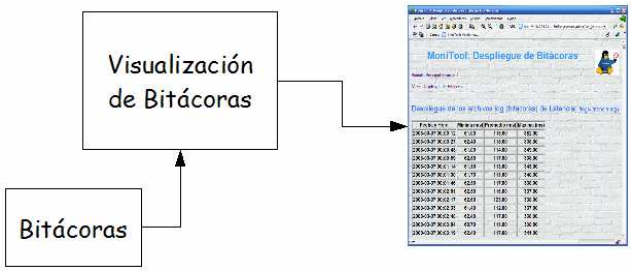


Figura 3.8 Módulo Visualizador de Bitácoras

3.7 DISEÑO DE LOS MÓDULOS

A continuación se especificarán los detalles de diseño de los módulos desarrollados para la aplicación MoniTool.

3.7.1 MÓDULO DE MONITOREO Y REGISTRO DE DATOS

Como ya se mencionó anteriormente, este módulo consiste de dos fases: la fase de monitoreo y la fase de registro. Se harán entonces el análisis de diseño de estas fases.

La aplicación fue diseñada para que tener propiedades modulares y secuenciales, dado que estas características son fácilmente entendibles y explicables por medio de un diagrama de flujo, estos han sido seleccionados para exponer esta parte del diseño.

3.7.1.1 FASE DE MONITOREO

La fase de monitoreo consiste en una instrucción que obtiene datos del servidor. Es una directriz que se ejecuta periódicamente en el servidor obteniendo un resultado que será enviado a la siguiente fase a través de un archivo temporal. El diagrama de flujo de la figura 3.9 detalla el diseño de esta operación.

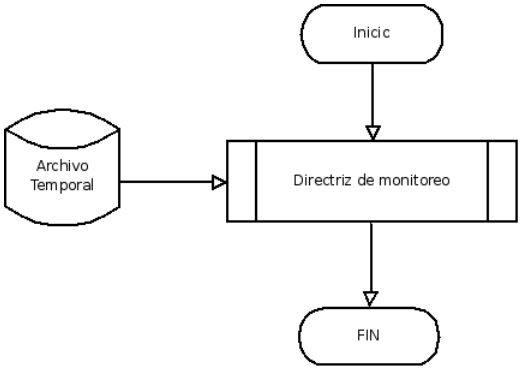


Figura 3.9 Diseño Módulo Fase de Monitoreo

3.7.1.2 FASE DE REGISTRO

Esta fase realiza la tarea de almacenar los datos generados por la fase de monitoreo en el registro temporal así como en la bitácora correspondiente. Los detalles del diseño de esta fase se muestran de acuerdo al diagrama de flujo de la figura 3.10.

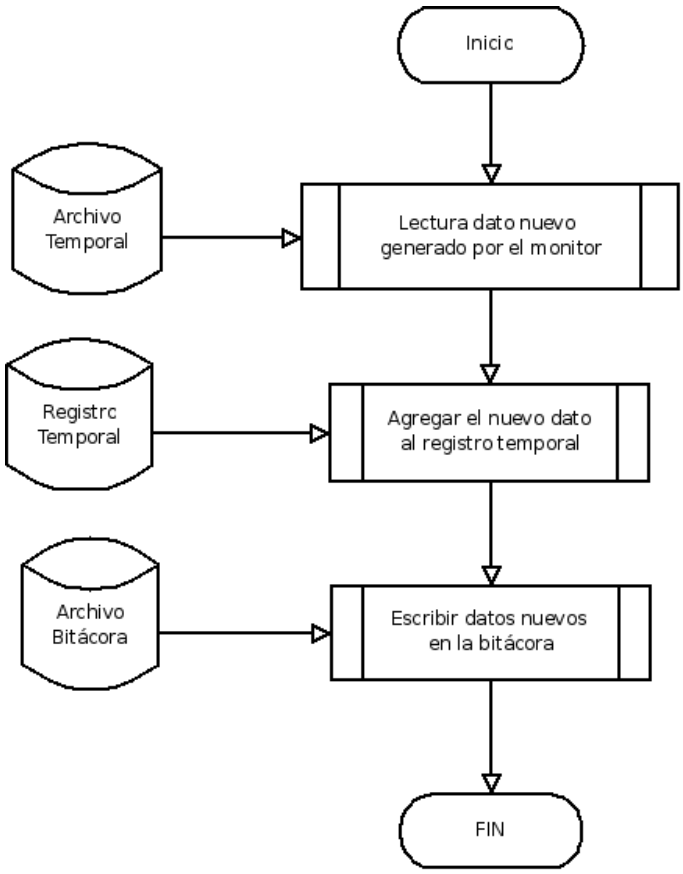


Figura 3.10 Diseño Módulo Fase de Registro

Esta etapa leerá el dato nuevo generado por la fase anterior, que será a través de un archivo temporal. Una vez leído este dato, será agregado al registro temporal contenido en un archivo. Posteriormente se escribirá este dato en el archivo bitácora que corresponda de acuerdo a la métrica en cuestión.

3.7.2 MÓDULO ANÁLISIS DE DATOS

Este módulo se encarga de calcular un promedio de los datos que han sido obtenidos a través del monitoreo durante la última hora. Una vez calculado este promedio, se compara con un valor máximo definido por el usuario en el archivo de configuración, si se ha sobrepasado ese valor máximo se mandará un reporte

por medio de un correo electrónico. La figura 3.11 muestra detalles del diseño de este módulo.

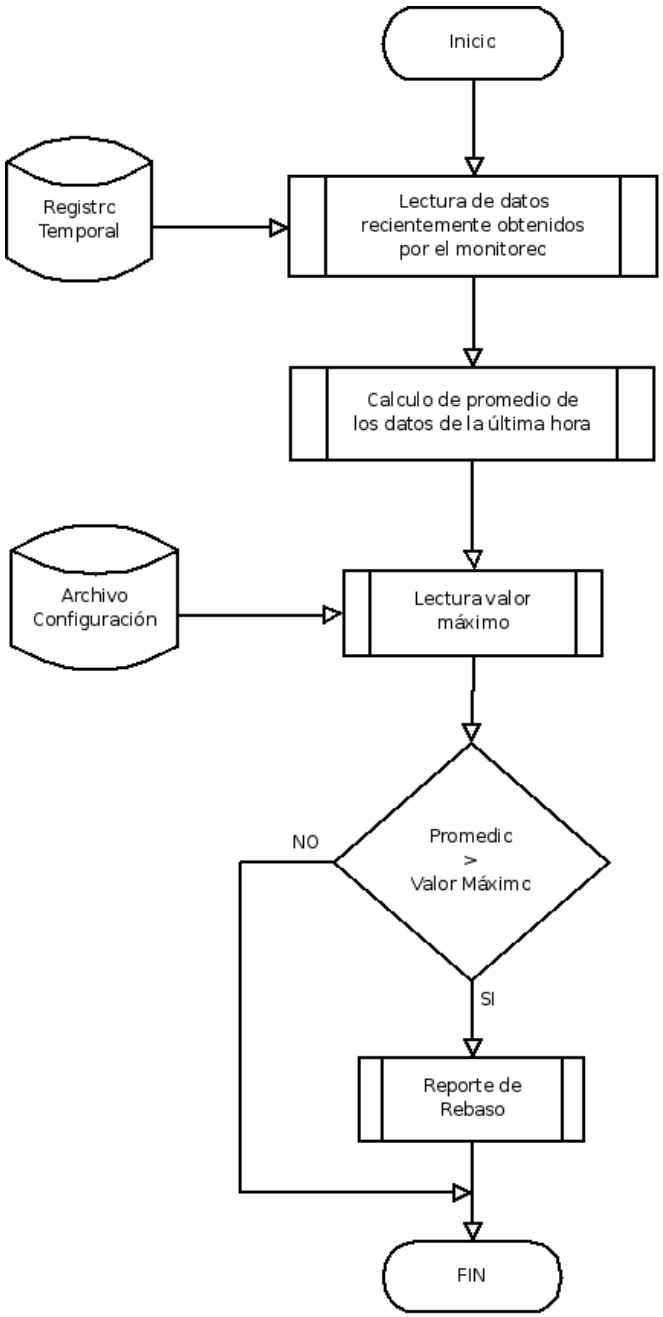


Figura 3.11 Diseño Módulo Análisis de Datos

3.7.3 MÓDULO VISUALIZACIÓN DE BITÁCORAS

Para realizar la visualización de alguno de los archivos de bitácoras se debe conocer la métrica y nivel de información que se desea desplegar. Una vez hecho esto se procede a leer el archivo correspondiente a la métrica y nivel seleccionados y finalmente se despliega en la pantalla los datos leídos desde la bitácora en cuestión. La figura 3.12 muestra los detalles del diseño de este módulo.

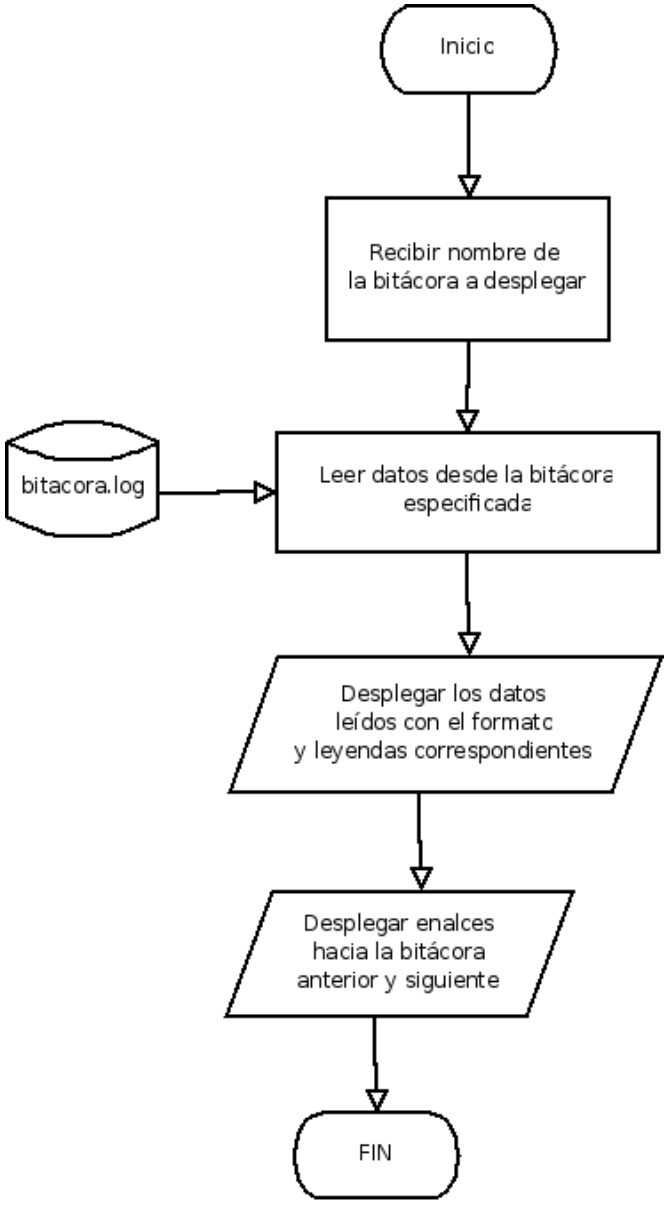


Figura 3.12 Diseño Módulo Visualizador de Bitácoras

3.7.4 MÓDULO CREACIÓN DE GRAFICAS

Este módulo es quizás el que tendrá mayor importancia en la aplicación, ya que será el que principalmente observe el usuario final. En este nivel de la aplicación se utilizarán resultados de etapas y fases previas.

El primer paso será leer los datos contenidos en el registro temporal de acuerdo a la métrica, se ajustarán las propiedades visuales y finalmente se creará la gráfica a través de un archivo de imagen, el cuál será desplegado posteriormente en el navegador Web.

La figura 3.13 muestra detalles del diseño de este módulo.

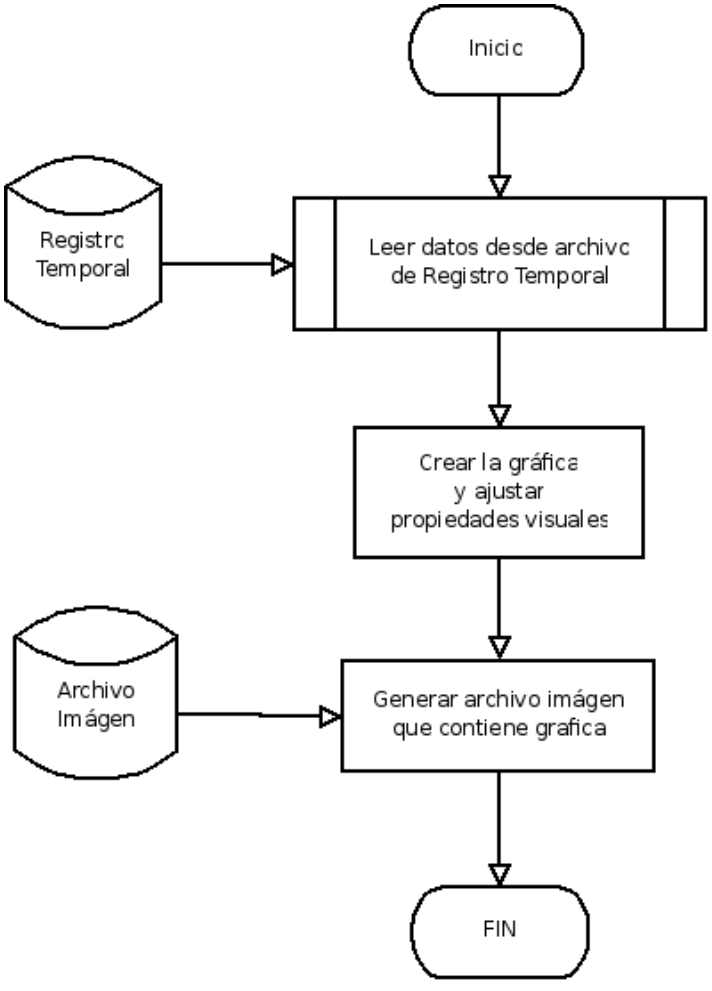


Figura 3.13 Diseño del Módulo Generador de Gráficas

3.8 MÓDULO CÁLCULO DEL ÍNDICE DE DESEMPEÑO

El índice del desempeño es un valor que indica al usuario el estado general del servidor de aplicaciones. El usuario definirá los valores máximos que podrán alcanzar las métricas antes de comprometer la calidad de servicio del servidor, y este módulo se encargará de calcular el índice y desplegarlo. Este índice de desempeño resumirá cuantas métricas han rebasado los valores máximos previamente definidos. La figura 3.14 muestra el diagrama de flujo del diseño de este módulo.

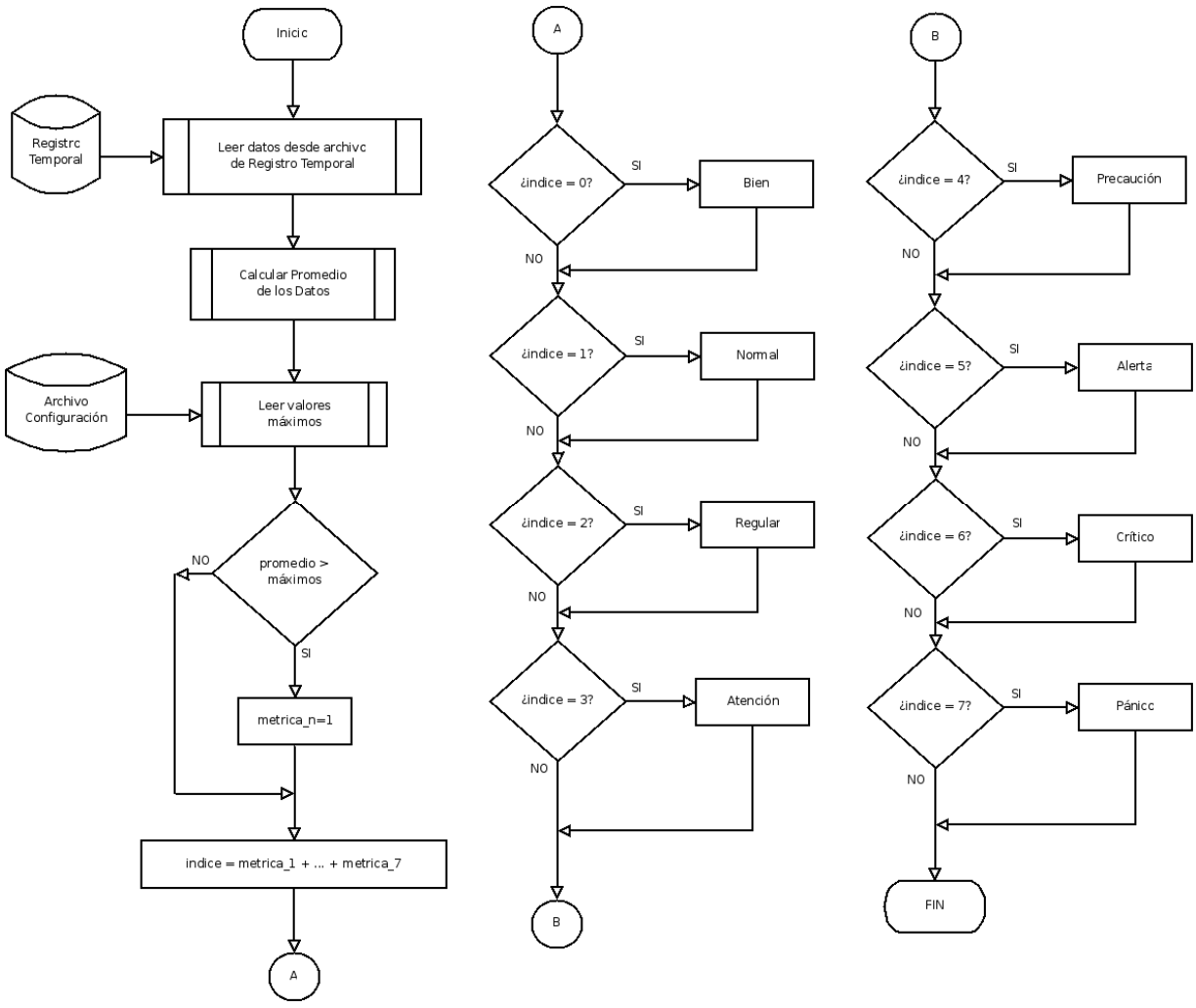


Figura 3.14 Diseño del Módulo Cálculo del Índice de Desempeño

3.9 PUNTOS CLAVE DEL DISEÑO

Dadas las necesidades descritas en el capítulo 1 y las métricas selectas en este capítulo, se ha hecho un diseño que permita la construcción de una herramienta que cumpla principalmente con facilidad de uso, configurable, escalable, requisitos mínimos de operación en el servidor y minúsculos en el cliente

- ✓ **Configurable**. La aplicación cuenta con varios archivos de configuración que controlan su funcionamiento. Estos archivos permiten al administrador adaptarla de acuerdo a sus necesidades, y dado el diseño de esta, dichos cambios pueden realizarse en cualquier momento, inclusive en tiempo de ejecución. De esta forma no es necesario reiniciar la ejecución para que los cambios en la configuración sean llevados a cabo.

- ✓ **Modular**. Una característica muy importante y destacable de esta aplicación es su naturaleza modular, que posteriormente reflejará un funcionamiento meramente secuencial. Esta característica permite que todas las operaciones sean realizadas por pequeños fragmentos de programas, los cuales fueron diseñados para realizar pequeñas tareas que en conjunto efectúan una gran operación. Dada esta naturaleza es fácil encontrar un problema en caso de que exista alguna falla en la aplicación.
  Otro punto importante a destacar acerca de esta característica es que la modularidad otorga independencia de ejecución, lo que además permite de manera muy natural que sean agregados nuevos módulos para otras métricas, sin afectar al funcionamiento de las ya existentes.

- ✓ **Jerarquía en archivos de bitácoras**. Las bitácoras son almacenadas en un conjunto de seis archivos, los cuales están clasificados en una jerarquía de antigüedad de datos. Esta clasificación permite al usuario poder acceder a la información de los últimos seis días de monitoreo para cada una de las métricas en distintos archivos.

- ✓ **Sistema inteligente de bitácoras**. El sistema hace el desplazamiento de los datos en forma de cascada en los archivos de bitácoras en la transición de un día a otro, de forma que en el archivo bitácora nivel 0 siempre estarán los datos del día en curso.

- ✓ **Repote de valores rebasados**. Con el sistema de alarmas, la aplicación es capaz de determinar cuando se ha rebasado el límite definido por el usuario en el archivo de configuración. El sistema es capaz de discernir e ignorar cuando se trata de un pico aislado y avisar cuando sea un problema persistente, por lo que avisará al administrador enviando un reporte por correo electrónico de la métrica rebasada.

- ✓ **Requisitos mínimos para el cliente**. El diseño esta hecho para que las gráficas sean desplegadas mediante un archivo tipo imagen y los datos de las bitácoras en texto. Por esta razón, el cliente solo debe contar con un navegador Web que sea capaz de desplegar texto e imágenes estándar. Dada esta característica, el cliente podrá ser una computadora con escasos recursos o incluso un dispositivo móvil.

3.10 RESUMEN

En este capítulo se han presentado los diagramas de flujo de datos que han sido elaborados como requerimientos de diseño. En estos diagramas se han especificado los detalles pertinentes de cada una de las etapas de la aplicación.

También se han especificado los módulos que serán implementados, mostrando así la naturaleza modular y secuencial de la aplicación. De igual forma se han detallado algunos puntos del diseño de la aplicación que son importantes para la siguiente etapa en el desarrollo de la tesis.

CAPÍTULO 4

Implementación de la Aplicación: MoniTool

4.1 INTRODUCCIÓN

En este capítulo se describen los detalles de la implementación y funcionalidad de los módulos descritos en el capítulo anterior y de la aplicación completa en general. Se describen además las características de las herramientas de desarrollo y programas de software empleados para dicha implementación.

4.2 CONSIDERCIONES DE IMPLEMENTACIÓN

La aplicación MoniTool funciona bajo la arquitectura Cliente-Servidor dado que el servidor de aplicaciones a monitorear hará la función de servidor para esta aplicación, mientras que el cliente será el quipo en el que se encontrará en ejecución el navegador que despliegue las gráficas.

Ya se han establecido y explicado los elementos que forman el sistema en conjunto, el paso siguiente es mencionar las herramientas que se utilizan para la implementación del diseño. Las herramientas computacionales con las que se deben contar para la implementación de la presente aplicación son las siguientes:

- Un servidor de aplicaciones bajo alguna distribución de GNU/Linux
- Servidor Web Apache Software Foundation
- PHP
- Biblioteca Jpgraph de PHP
- C++
- Bash Script

4.3 IMPLEMENTACIÓN DE LOS MÓDULOS

Ya se han detallado en el diseño las funciones que llevan a cabo todos los módulos de la aplicación. Sin embargo en esta sección se especificará la forma en la que interactúan operativamente estos módulos y generan un resultado factible, que resultará en las gráficas de monitoreo.

Dada la implementación y en el sentido operativo, la aplicación consiste de tres etapas, las cuales pueden verse en la figura 4.1.

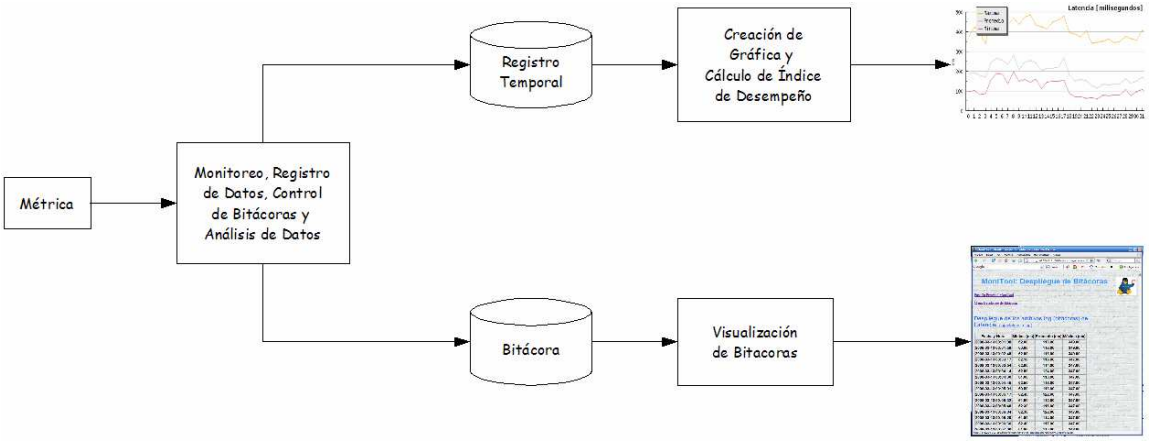


Figura 4.1 Estructura Operativa de la Aplicación MoniTool

En la implementación se ha construido la aplicación en tres etapas principales, la primera de ellas consiste en el monitoreo, registro de datos y control de bitácoras, la segunda en la generación de las gráficas y la última el despliegue de las bitácoras.

4.3.1 MÓDULO DE MONITOREO, REGISTRO DE DATOS, ANÁLISIS DE DATOS Y CONTROL DE BITÁCORAS

Todos estos módulos, descritos en el capítulo anterior, se concentran en la parte operativa en uno solo, el cuál estará en ejecución en el servidor. Ya se han detallado en el diseño las funciones que llevan a cabo cada uno de estos módulos, sin embargo, en esta sección se especificará la forma en la que interactúan operativamente estos módulos y generan un resultado factible, que resultará en las gráficas de monitoreo.

Los módulos antes analizados operan de manera conjunta, de tal suerte que la suma de las tareas unitarias de cada una de ellas resulta un sistema de monitoreo, registro, análisis y control de bitácoras que se encuentra en ejecución en el servidor de aplicaciones que se pretende monitorear.

La ejecución de este conjunto de operaciones deberá ser periódica y permanente, debido a que se requiere que el monitoreo se lleve a cabo en todo momento en el servidor de aplicaciones, incluso en días y horas no laborables en las que el

administrador no se encuentre frente a la máquina. De esta forma el esquema será un bucle sin fin para lograr este cometido.

En el diagrama de la figura 4.2 podrá notar la existencia de un bloque de retardo que permite que la aplicación tome muestras cada periodo controlado de tiempo. Este retardo obedece a un valor establecido por el usuario, el cuál está contenido en un archivo de configuración de la aplicación.

De esta forma, la implementación de estos módulos queda descrita mediante el diagrama de flujo de datos de la figura 4.2.

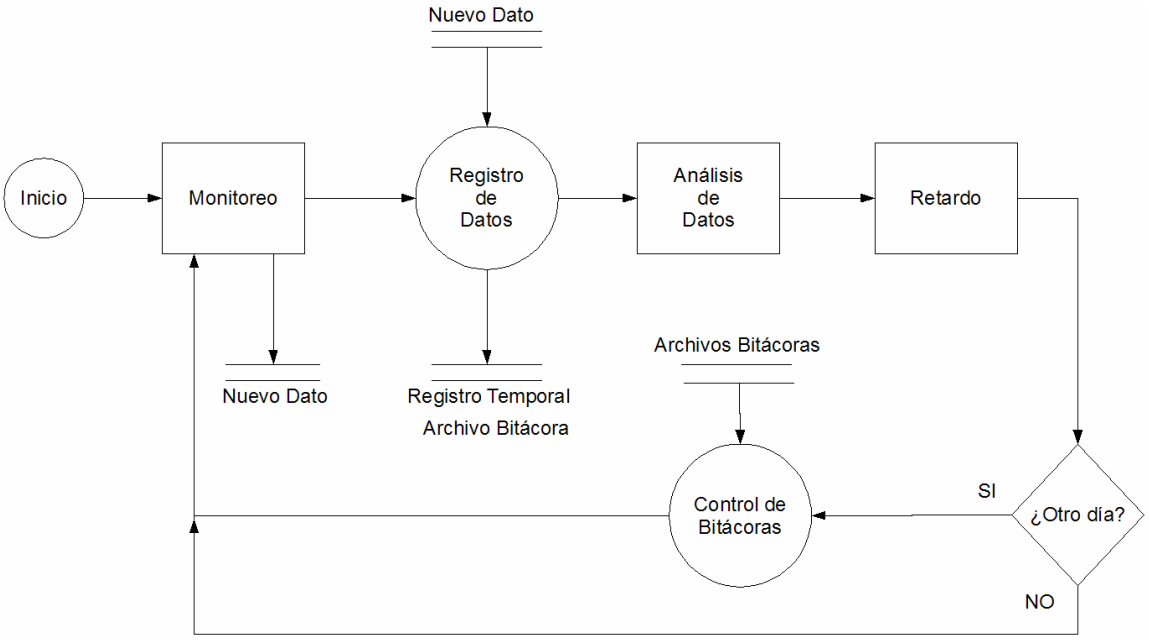


Figura 4.2 Arquitectura Operativa del Monitoreo, Registro, Análisis y Control de Bitácoras

En el diagrama de flujo de datos de la figura 4.2 se puede observar que existe un inicio y no un final, lo que implica que este es un ciclo infinito. El primer módulo de monitoreo vigila la métrica en cuestión y arroja un resultado que será almacenado en un archivo temporalmente. Posteriormente la fase de registro de datos, leerá ese dato nuevo y lo agregará al registro temporal y al archivo bitácora del día en curso (nivel 0). El análisis de datos acumulará todos los datos registrados, con el cuál se hará un promedio posteriormente. El siguiente bloque es un retardo en la aplicación, bloque que lee desde el archivo de configuración el tiempo que debe detener el flujo de la aplicación. Cuando el tiempo de guarda ha terminado, el control de bitácoras detecta si ha habido una transición de un día a

otro, si ocurrió hace un desplazamiento de los datos contenidos en los archivos en forma de cascada; cuando esto concluya, o bien si no ocurrió la transición, se sigue el flujo hacia el módulo de monitoreo para generar un nuevo dato y repetir la operación nuevamente.

En la etapa de análisis se realiza un promedio de los datos durante cada hora de monitoreo, y cuando ha pasado ese tiempo y le promedio ha sido calculado es comparado con el valor máximo definido en el archivo de configuración de la aplicación. Si el promedio es menor al máximo no se hace nada, pero si es mayor se construye un reporte con la métrica, promedio y fecha en la que ha ocurrido esto. Dicho reporte es finalmente enviado por correo electrónico a la dirección del administrador para que este notificado de la anomalía y pueda arreglar esta situación.

Este esquema de implementación es el mismo que se utiliza para todas las métricas, por tanto esta arquitectura es una generalización de todas ellas. Si se deseara agregar una nueva métrica a monitorear, habría que obedecer esta estructura para poder añadirlo como un módulo más a la aplicación completa.
Nótese que esta etapa almacena todos sus resultados en un par de archivos, lo cuáles serán leídos por la etapa posterior. De esta forma es que la siguiente etapa depende de los resultados que genere su predecesora, y no de su correcto flujo o funcionamiento.

Cabe añadir finalmente que los programas de monitoreo consumen memoria y tiempo de ejecución del procesador (como cualquier programa) del servidor a vigilar. Aquí se hace presente un principio físico que dice “no se puede realizar una medición sin afectar lo que se mide”.

Sin embargo, y a pesar de que no se puede eliminar este fenómeno, si es posible minimizar su efecto. Para esto se hace uso de una gran característica del sistema operativo, la cuál permite asignarle prioridades de ejecución a las aplicaciones, de forma que el calendarizador se encarga de darle mayor tiempo de ejecución a las aplicaciones que tengan una mayor prioridad de ejecución, y a los que tengan prioridad baja menor tiempo de ejecución.

De esta forma, se ha diseñado la aplicación MoniTool para que todos los procesos que ésta genere, tengan la prioridad más baja posible, de manera que consuman menos recursos y su efecto sea mínimo en el monitoreo de los datos “reales” del servidor.

Adicionalmente, los programas de monitoreo son ejecutados en segundo plano, de forma que estarán en la lista de procesos en ejecución que mantiene el sistema operativo.

4.3.2 IMPLEMENTACIÓN OPERATIVA DEL MONITOREO

Tal como se mostró en el capítulo 3, en la sección 3.7.1.1 el monitoreo es llevado a cabo mediante una directriz. Esa directriz o comando, que varía de acuerdo a la métrica correspondiente, entrega una salida que es filtrada para obtener así los datos que se necesitan.

4.3.2.1 CPU

Para obtener el porcentaje de procesador que está siendo utilizado por los procesos en ejecución en el servidor de aplicaciones, desde las primeras versiones de Unix existe un comando que permite desplegar los procesos en ejecución, así como información que es usualmente útil. Para el caso del monitoreo la forma en que se utilizó este comando y las opciones son las siguientes:

```
$ ps –eo %C > .lista_procesos
```

Este comando entrega una lista con el porcentaje de uso de todos los procesos que se encuentran en ejecución en el servidor. Cabe mencionar que esta salida es redireccionada a un archivo para poder ser leída y procesada por la etapa de monitoreo de la aplicación.

4.3.2.2 RAM

Existen varias formas de saber la cantidad de memoria RAM utilizada en el servidor, sin embargo se utiliza el siguiente comando:

```
$ free –k | grep Mem > .salida
```

Esta instrucción despliega la cantidad de memoria total, utilizada y libre además de mostrar el uso de la memoria de intercambio. Con la opción "*-k*" el comando agrupa los datos en kilobytes para una mejor comprensión, podría ser agrupado inclusive en megabytes, pero dado que estos datos se utilizarán para calcular un porcentaje, esto no es necesario. La salida es redireccionada hacia un archivo para su procesamiento posterior.

4.3.2.3 LATENCIA

En el caso de la latencia, la técnica empleada para calcular este parámetro es enviar una serie de paquetes ICMP de prueba a diversos puntos de interés en la red

(definidos por el usuario en el archivo sitios.conf) y se mide el tiempo que tarda en regresar la respuesta de ellos. El comando y las opciones utilizadas son los siguientes:

```
$ fping -c 1 -s < ../config/sitios.conf > .salida
2>> .salida
$ cat .salida | grep "min round" > .min_latencia
$ cat .salida | grep "avg round" > .avg_latencia
$ cat .salida | grep "max round" > .max_latencia
```

Este comando envía un solo paquete ICMP a los sitios contenidos en el archivo *sitios.conf* y espera su retorno. Una vez que la respuesta ha llegado, reporta el tiempo que tomó el trayecto así como el tiempo máximo, mínimo y promedio de todos los paquetes. Esta salida es redireccionada hacia un archivo que será posteriormente procesada.

Cabe mencionar que algunos ruteadores bloquean los paquetes ICMP, por lo que podría ser una desventaja para medir de esta forma la latencia. Sin embargo, dado que es muy fácil medir este factor con este tipo de comandos y de que la aplicación permite seleccionar al usuario los puntos de la red que desea utilizar como referencia para medirlo, se considera el método más apropiado y dejar a criterio del administrador los sitios que utilizará como referencia, sabiendo de antemano que sean accesibles para recibir paquetes ICMP.

4.3.2.4 PAQUETES PERDIDOS

Para calcular el porcentaje de paquetes extraviados en la red, se envía un conjunto de paquetes ICMP y se espera una respuesta. En caso de que la respuesta no llegue, indica que el paquete no llegó al destino o bien que la respuesta se perdió. Dado que el mecanismo de medición es similar al caso de la latencia, se utiliza la misma respuesta para realizar este cálculo, por lo tanto ese comando tiene un doble propósito.

Una vez que la salida nos dice cuantos paquetes se enviaron y cuantos se recibieron, esta información es filtrada y redirigida hacia un archivo, el cuál será procesado posteriormente.

```
$ cat .salida | grep "Echos sent" > .paq_env
$ cat .salida | grep "Echo Replies received" > .paq_rcbd
```

4.3.2.5 TASA DE TRANSFERENCIA

Para medir la tasa de transferencia de las interfaces de red del servidor de aplicaciones, se utiliza el siguiente comando:

```
$ ifstat –z –T | cat > .salida
```

Este comando reporta la cantidad de información que entra y sale de cada una de las interfaces de red. Con las opciones "*-z –T*" se le indica que despliegue la cantidad de información que entra y sale para cada una de las interfaces activas y un total de todas ellas. Dicha salida es redireccionada hacia un archivo para procesarla después.

4.3.2.6 NÚMERO DE CONEXIONES

Para obtener el número de conexiones que han sido establecidas con el servidor existe un comando en Unix desde las primeras versiones que permite desplegar en pantalla esta información:

```
$ netstat –n > .salida
$ cat .salida | grep tcp  | wc -l > .internet
$ cat .salida | grep unix | wc -l > .unix
```

El comando despliega un listado de todas las conexiones que han sido establecidas con el servidor. Reporta dos tipos de conexiones, tipo unix y tcp. Las tipo unix son las basadas en servicios o demonios de unix y las tcp se refieren a las que han sido implementadas bajo el protocolo de transporte y control (tcp). La salida es filtrada hacia un par de archivos que posteriormente serán procesados.

4.3.2.7 PAQUETES ENVIADOS Y RECIBIDOS

Una vez más se utiliza el mismo comando que en el apartado anterior, pero con una opción distinta que modifica la información desplegada:

```
$ netstat -s > .salida
$ cat .salida | grep "total de paquetes recibidos" >.paq_rcbd
$ cat .salida | grep "peticiones enviadas" > .paq_env
```

La opción "-s" ocasiona que el comando despliegue estadísticas sobre cada uno de los protocolos que soporta el sistema operativo. En este caso se filtran todos los paquetes que entran y salen en el protocolo IP y la salida es redireccionada hacia un archivo que posteriormente será procesado.

4.3.3 MÓDULO VISUALIZACIÓN DE BITÁCORAS

Esta etapa simplemente se leen los datos contenidos en las bitácoras, que han sido generados por la etapa anterior. La figura 4.3 muestra la forma operativa en que la visualización de las bitácoras es llevada a cabo.

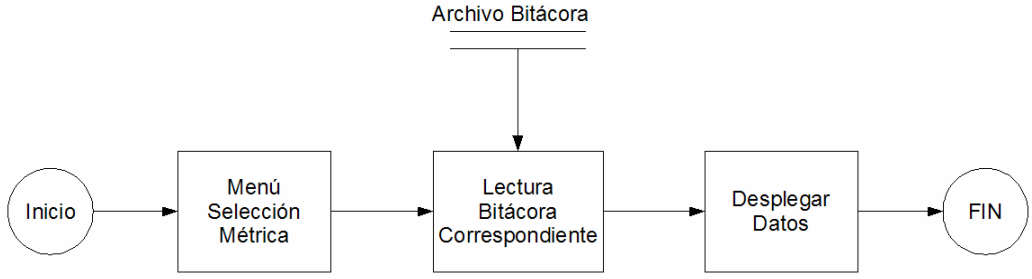


Figura 4.3 Arquitectura Operativa del Módulo Visualización de Bitácoras

En la figura 4.3 se puede observar además que este procedimiento consiste de tres etapas, a continuación se especifican los detalles de la implementación de cada una de estas etapas:

4.3.3.1 MENÚ DE SELECCIÓN DE MÉTRICAS

Para cuestiones operativas, el usuario debe especificar cuál métrica desea que se despliegue, por esta razón es que en el diagrama de observa un bloque "Menú Selección Métrica" que funge tal propósito. La figura 4.4 muestra la pantalla de selección.

Figura 4.4 Menú de Selección de Métrica

Como se puede observar, en la figura 4.4 esta fase es una interfaz con el usuario y se despliega ya en el navegador Web. El menú consiste en una serie de ligas que enlazan hacia las métricas disponibles por la aplicación, y una liga hacia la pantalla inicial de despliegue de gráficas.

4.3.3.2 LECTURA DE BITÁCORA CORRESPONDIENTE

Para la lectura del archivo correspondiente de acuerdo a la métrica selecta, se realiza de la forma definida en el diseño de este módulo especificado en el capítulo anterior.

El archivo que se leerá inicialmente será el correspondiente al del día en curso, es decir el nivel 0, ya que son los datos que quizás sean de mayor interés para el usuario.

4.3.3.3 DESPLEGAR DATOS

Una vez leído el archivo correspondiente será desplegado en la pantalla del navegador Web. La figura 4.5 muestra un ejemplo del despliegue de las bitácoras de la latencia.

Figura 4.5 Despliegue de Datos de la Bitácora Seleccionada

Esta también es una interfaz con el usuario, tanto en la parte superior como en la inferior de la pantalla hay dos enlaces. Los superiores llevan hacia la pantalla principal y al menú de selección de métricas, mientras que los inferiores despliegan las bitácoras anteriores y siguientes de acuerdo a la antigüedad de los datos que contienen. De esta forma, en cada pantalla del navegador se mostrarán datos de un sólo día.

4.3.4 MÓDULO GENERADOR DE GRÁFICAS

Como se detalló en el diseño de este módulo, este lee el contenido del registro temporal de alguna métrica en particular y crea las gráficas. La implementación de esta acción, así como los detalles de su funcionamiento son especificados en el siguiente diagrama de flujo de datos (figura 4.6).

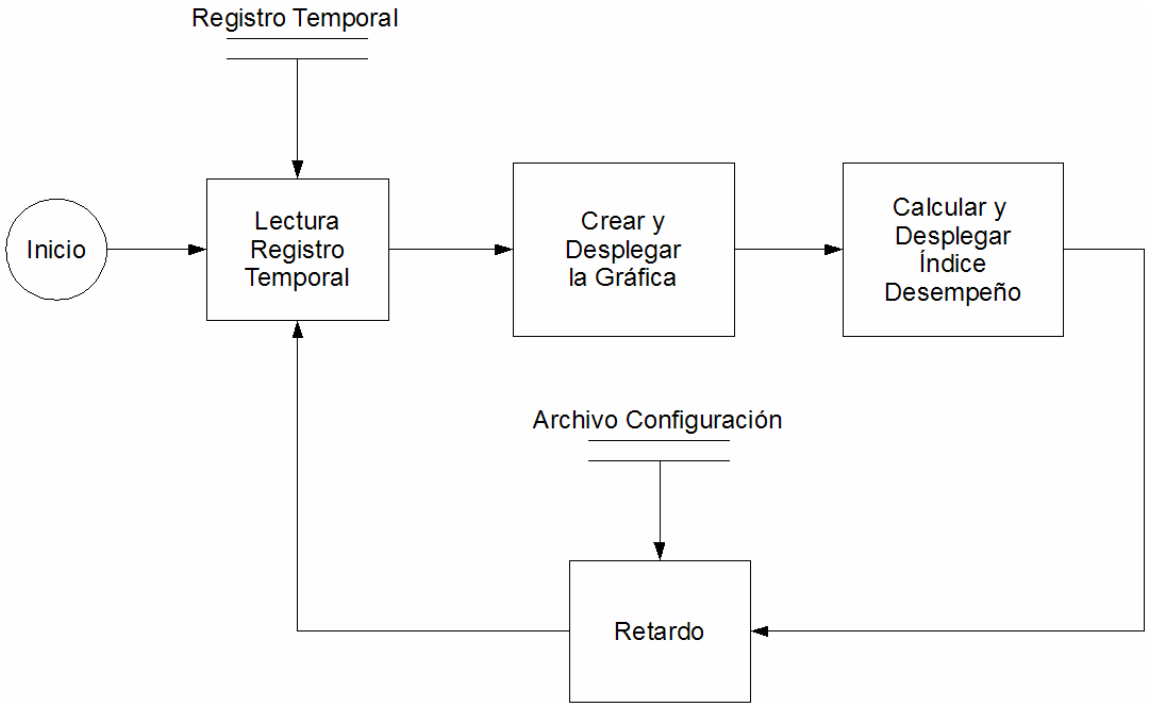


Figura 4.6 Creación de las Gráficas

El primer proceso de este diagrama lee el contenido del registro temporal generado por la etapa de monitoreo. Posteriormente y ya con los datos leídos, se crea la gráfica de acuerdo a las características de la métrica que corresponda, es decir, las leyendas, colores, dimensiones y tipo de gráfica. Finalmente se despliega la gráfica.

Se puede observar que finalmente existe un retardo, el cuál espera un tiempo predefinido por el usuario en el archivo de configuración. Esta espera existe para que la etapa de monitoreo genere un nuevo dato y lo almacene en el registro temporal, y cuando esto ocurra, se generará nuevamente la gráfica con los datos más recientes del monitoreo. De esta forma es como se encuentran sincronizadas las dos etapas y funcionan de manera paralela. La figura 4.7 es un ejemplo de este resultado.

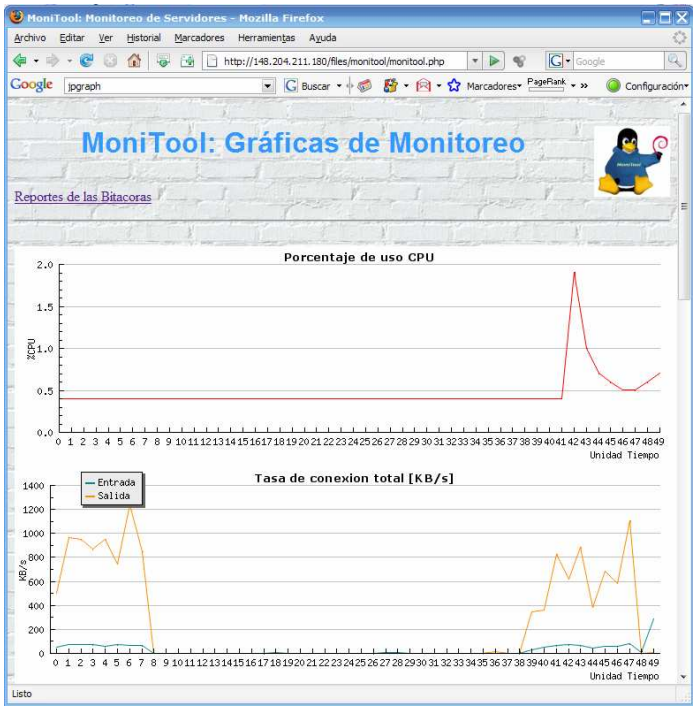


Figura 4.7 Gráficas de Monitoreo

La salida que verá el usuario será una pantalla como esta, en un navegador Web, la cuál es la pantalla principal de la aplicación y la que se abrirá en primera instancia cuando de acceda a ella. En esta pantalla estarán las siete métricas selectas para la aplicación MoniTool, y se refrescará cada tiempo predefinido en el archivo de configuración.

En la parte superior de la página existe un enlace hacia el menú de selección de métricas de la parte del despliegue de bitácoras.

4.3.5 MÓDULO CÁLCULO ÍNDICE DE DESEMPEÑO

En el diagrama mostrado en la figura 4.6 se puede observar el módulo que se encarga de calcular y desplegar el índice de desempeño junto con la gráfica de la métrica correspondiente. Este módulo calcula el promedio de los datos obtenidos recientemente y que se encuentran almacenados en el registro temporal, si el promedio de una métrica es mayor al establecido por el usuario en el archivo de configuración de la aplicación, el índice se incrementa en uno (el cuál inicialmente vale cero). De esta forma el índice solo podrá tomar valores entre creo y siete, el cuál indica el número de métricas que han rebasado su valor máximo. Posteriormente se le asigna una leyenda de acuerdo al valor del índice, la cuál es desplegada junto con las métricas que han rebasado el nivel máximo.

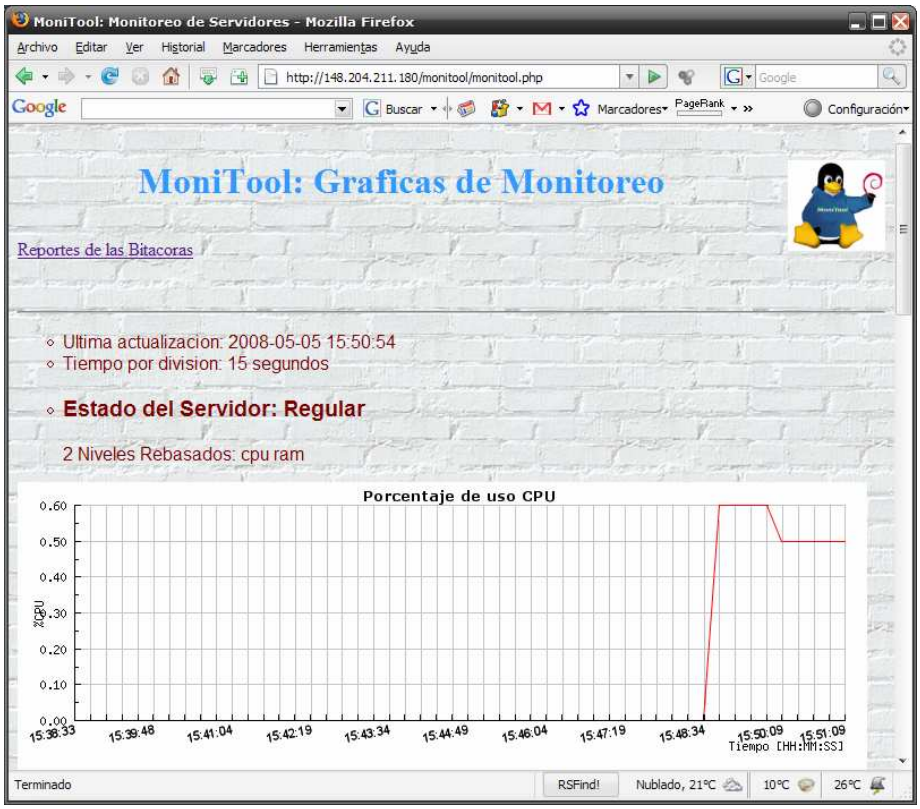


Figura 4.8 Despliegue del Índice de Desempeño

4.3.6 ESTADOS DEL ÍNDICE DE DESEMPEÑO

Como ya se ha mencionado, el índice revela la cantidad de métricas que han sobrepasado el valor máximo definido por el usuario en los datos monitoreados

recientemente. El índice vale inicialmente cero y se incrementa en uno por cada métrica que rebasa el límite, por lo tanto tendrá valores entre cero y siete. A cada valor le corresponde una leyenda que será desplegada en la pantalla como una palabra clave indicadora que resume el desempeño actual que experimenta el servidor de aplicaciones. La tabla 4.1 muestra todos los valores que pueden tomar el índice de desempeño y la leyenda correspondiente a cada uno de ellos.

Índice de Desempeño	Leyenda
0	Bien
1	Normal
2	Regular
3	Atención
4	Precaución
5	Alerta
6	Crítico
7	Pánico

Tabla 4.1 Valores y Leyendas del Índice de Desempeño

4.4 CARACTERÍSTICAS OPERATIVAS DE MONITOOL

MoniTool es una herramienta de monitoreo que posee varias características de funcionamiento que ofrecen diversas capacidades al usuario. Estas características serán descritas a continuación.

4.4.1 ARCHIVO DE CONFIGURACIÓN

Una característica destacable de esta aplicación consiste en su capacidad de configurabilidad mediante un archivo de configuración. Este archivo afecta directamente al comportamiento de la aplicación y permite que el usuario la ajuste a sus necesidades o características de operación de su red o servidor de aplicaciones.

Los parámetros que se pueden configurar en este archivo y una breve descripción de ellos se muestran en la siguiente tabla (4.2).

Archivo	Función
borrar_bitacora	Borrar o agregar datos en la bitácora nivel 5
num_datos	Cantidad de datos en el registro temporal
num_tarjetas_red	Numero de tarjetas de red configuradas en el sistema
retardo	Tiempo de espera general de la aplicación
cpu	Porcentaje máximo de uso del CPU
ram	Porcentaje máximo de uso de la memoria RAM
latencia	Valor máximo de latencia
rate	Valor máximo de tasa de transferencia
paq_perdidos	Porcentaje máximo de paquetes perdidos
paq_rcbd_env	Cantidad máxima de paquetes enviados y recibidos
conexión	Cantidad máxima de conexiones
sitios.conf	Sitios de prueba para latencia y paquetes perdidos

Tabla 4.2 Archivos de configuración de MoniTool

- *borrar_bitacora*. Este parámetro deberá ser un '0' para indicar al sistema que se desean acumular todos los datos antiguos en la bitácora nivel 5; y con '1' se sobrescribirán su contenido.

- *num_datos*. Mediante este parámetro se define el tamaño del registro temporal. Recuerde que este valor es la cantidad de datos que habrán en el eje X de las gráficas.

- *num_tarjetas_red*. Con este parámetro se especifica el número de tarjetas de red que se encuentran configuradas en el sistema operativo, es decir, el número de tarjetas de red que tienen dirección de red válida.
- *retardo*. Este parámetro permite especificar el tiempo en segundos de espera de la aplicación general. Los bloques de monitoreo y generación de gráficas lo obedecen.

- *cpu*. Mediante este parámetro se configura el porcentaje máximo permitido de uso del CPU.

- *ram*. A través de este parámetro se define la cantidad máxima de uso de memoria RAM en el servidor de aplicaciones.

- *latencia*. Este parámetro configura la latencia máxima en milisegundos permitida en el servidor.

- *rate*. Por medio de este valor se especifica el máximo permitido de kb/s en las interfaces de red.

- *paq_perdidos*. Este es el porcentaje máximo de paquetes perdidos que se tolerará en el servidor de aplicaciones.

- *paq_rcbd_env*. La cantidad de paquetes enviados y recibidos máximos permitidos en el servidor de aplicaciones.

- *conexión*. El número máximo de conexiones activas que se pueden establecer al servidor de aplicaciones.

- *sitios.conf*. Este archivo contiene una lista de sitios o direcciones IP de los lugares en la red que se desean utilizar para determinar la latencia y los lugares de prueba para calcular el porcentaje de paquetes perdidos en la red.

Cabe mencionar que la intención del parámetro *borrar_bitacora* es impedir que en el archivo bitácora de menor nivel se acumulen una gran cantidad de datos y que posteriormente sea incomprensible la cantidad de información contenida en el, además que el tamaño de este podría incrementarse demasiado y desperdiciar espacio del disco duro en información que quizás no sea útil.

Se recomienda tener 50 o incluso 75 datos máximo en el registro temporal, ya que con un valor mayor podría ser un poco ilegible la gráfica generada. Dado que no existe un límite natural para el tamaño de este registro, será cuestión de que el usuario pruebe algunos valores y decida el que mejor le convenga.

El tiempo de retardo recomendado es de 30 segundos, el cuál es suficientemente eficaz para mostrar anomalías de servicio en el equipo, además que la cantidad de información generada en los archivos de configuración no será demasiada.

Para el cálculo de la latencia se recomienda utilizar sitios relativamente cercanos y lejanos en la red, de forma que el promedio resultante sea lo más acertado posible.

4.4.2 SCRIPTS DE INICIO Y PARO

MoniTool dispone de un par de archivos que inician y detienen la ejecución del sistema de monitoreo, registro, análisis y control de bitácoras especificado en la figura 4.2 para cada una de las métricas.

Estos archivos son bash-scripts que contienen una secuencia de programas y comandos que ejecutan todos los procesos de monitoreo. La figura 4.9 muestra la secuencia de ejecución de tales procesos.

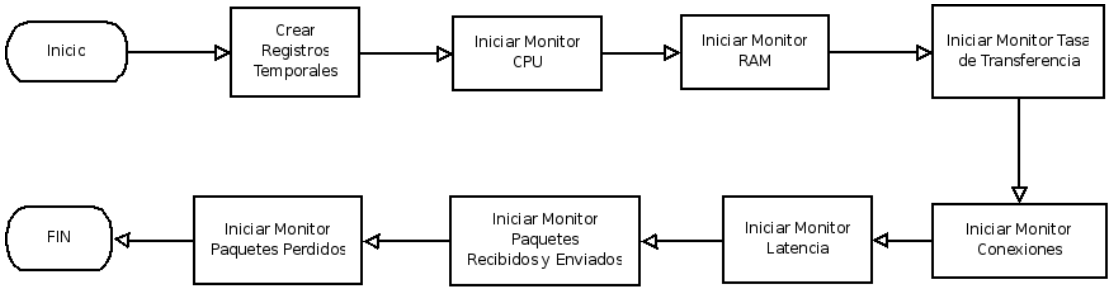


Figura 4.9 Script para Inicio de Monitoreo

A su vez existe otro bash-script que finaliza todos los procesos que genera el monitoreo, evitando así que queden procesos perdidos en ejecución en el sistema. La secuencia de detención se muestra en la figura 4.10.

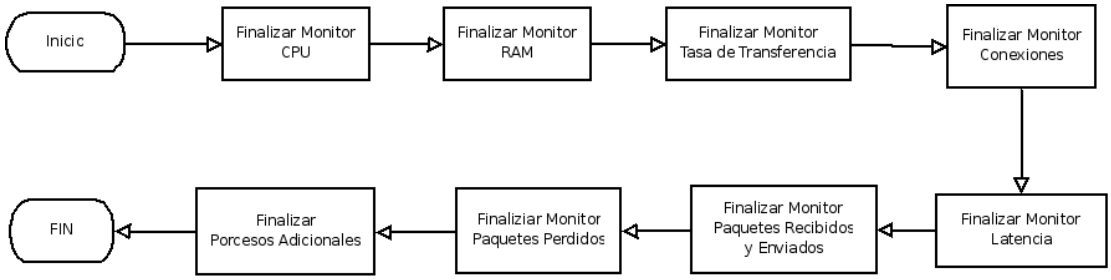


Figura 4.10 Script para Finalizar todos los procesos de Monitoreo

4.4.3 SCRIPT DE COMPILACIÓN DE CÓDIGO FUENTE

A pesar de que los programas que vigilan las métricas fueron creados y compilados previamente, y simplemente podrían ser ejecutados en el servidor sin

requerir alguna complicación adicional, y debido a las cómodas bondades de transportabilidad que ofrece el lenguaje C/C++, es recomendable que el servidor cuente con un compilador de C/C++ instalado. Esta recomendación es hecha debido a que el código fuente de los programas debería ser compilado en la arquitectura particular del servidor en cuestión, y generar el archivo ejecutable con los parámetros específicos de este y funcionar de una manera más óptima.

Adicionalmente, si el usuario desea realizar alguna modificación al código fuente de la aplicación, serpa necesario por lo tanto una nueva compilación de los archivos fuente.

Cualquiera sea el caso, existe un script que compila todos los archivos fuente de la aplicación, genera los binarios ejecutables y los ubica en el directorio correspondiente con el debido nombre de archivo.

Este script tiene la finalidad de simplificar el hecho de tener que compilar uno a uno los archivos con su respectivo nombre y tener que colocar el ejecutable en la ruta específica que le corresponda, ahorrando tiempo del usuario y garantizando que la aplicación encuentre los archivos ejecutables en los lugares ya definidos.

Si se desean agregar opciones para el proceso de compilación, como por ejemplo la arquitectura del procesador, este es el archivo que será necesario modificar.

Para detalles de operación de este script, consúltelo en el Anexo B.

4.5 CONSIDERACIONES ADICIONALES PARA MONITOOL

Un punto fundamental para el despliegue de las gráficas a través del servidor Web, implica que es necesario colocar los scripts de php en la carpeta adecuada de publicación definida en el servidor Apache o bien crear enlaces hacia el directorio correspondiente. De esta forma, los archivos locales podrán ser exportados a través del protocolo http y ser accesibles por un navegador Web.

Si el usuario lo prefiere, puede modificar la configuración del servidor Apache para modificar la raíz publicada por defecto por alguna otra en la que el usuario tenga sus archivos. Para mayores detalles de esto, deberá consultar la documentación de Apache.

Es importante aclarar que los programas de monitoreo deberán estar en ejecución para que los datos nuevos sean graficados. Si por alguna razón los programas de monitoreo no estuviesen en ejecución, en el navegador Web observaríamos la

gráfica de los últimos datos capturados sin renovación; el módulo haría la recarga respectiva de acuerdo al tiempo definido, pero los datos serían los mismos.

Para evitar esta situación, se recomienda agregar a la secuencia de arranque del sistema operativo la ejecución de los programas analizadores de las métricas para que de esa forma, la herramienta esté disponible cada vez que se inicie el servidor y no sea necesario que se ejecuten todos los programas de monitoreo manualmente por el administrador. En el caso del servidor Web, cuando es instalado se configura automáticamente para ser iniciado por defecto al arranque de la computadora.

4.6 RESUMEN

En este capítulo se han especificado las razones por las cuales se eligieron todas las herramientas y programas utilizados para esta aplicación.

Adicionalmente se puntualizaron los detalles operativos realizados al momento de la implementación de la aplicación en el sistema, de forma que el usuario sepa como se realiza todo el proceso.

Finalmente se señalaron los archivos de configuración, su función y la forma en que deben ser utilizados.

CAPÍTULO 5

Pruebas y Resultados

5.1 INTRODUCCIÓN

El objetivo principal de este capítulo es realizar pruebas de control en la aplicación con el fin de demostrar la confiabilidad de los datos obtenidos por esta. Para ello se ha configurado un ambiente de trabajo de ejemplo que servirá como prueba de control de referencia para futuras aplicaciones.

En este esquema de ejemplo se pretende tener un sistema funcional que se asemeje lo más posible a las condiciones que experimentaría un servidor en un caso real, de forma que la prueba sea lo más fiable que se pueda obtener, y por ende resultados más confiables. Se hará una prueba para cada una de las métricas que monitorea la aplicación.

Todos los resultados de las pruebas serán comparados con algunas otras herramientas que realizan la misma medición, para que de esta forma además se pueda comprobar que todos los valores reportados por la aplicación son confiables y verídicos.

5.2 EQUIPO DE CÓMPUTO

Para el desarrollo de las pruebas se utilizó el siguiente equipo de cómputo:

- **Servidor de Aplicaciones**: Computadora con procesador Intel Pentium D 2.80 GHz, 512 MB memoria RAM, disco duro de 160 GB, dos tarjetas de red y sistema operativo Debian GNU/Linux.

- **Cliente Local 1:** Computadora con procesador AMD Athlon XP 2700, 1GB de memoria RAM, disco duro 160 GB, una tarjeta de red y sistema operativo Ubuntu GNU/Linux.

- **Cliente Local 2:** Laptop Sony VAIO PGG-K43F, procesador Intel Celeron D 3.06 GHz, 512 MB memoria RAM, disco duro 40 GB, y dos tarjetas de red (alámbrica e inalámbrica) y sistema operativo Windows XP Profesional SP2.

- **Cliente Remoto 1:** Computadora con procesador Intel Pentium D 2.80 GHz, 512 MB memoria RAM, disco duro de 160 GB, dos tarjetas de red y sistema operativo Debian GNU/Linux.

5.3 ESQUEMA UTILIZADO COMO EJEMPLO

Para las pruebas de la aplicación se ha utilizado un servidor de aplicaciones implementado sobre Debian GNU/Linux 4.0 con kernel 2.6.18-i686. Los servicios configurados en este servidor son: Security Shell 4.7p1-8, NFS 1.1.2-2, Samba 3.0.28, Apache 2.2.8-3, PHP 5.2.5-3, MySQL 5.0.5-1a-3 y Exim 4.69-2. Estos servicios proveen Terminal remota virtual, compartición de archivos para Linux/Unix, compartición de archivos para Windows/Linux, servidor http, preprocesador de hipertextos PHP, base de datos y correo electrónico respectivamente.

5.4 ESPECIFICACIONES DEL ESQUEMA DE EJEMPLO

La figura 5.1 muestra los servicios que se han instalado y configurado en el servidor de aplicaciones implementado para las pruebas de control en esta etapa.

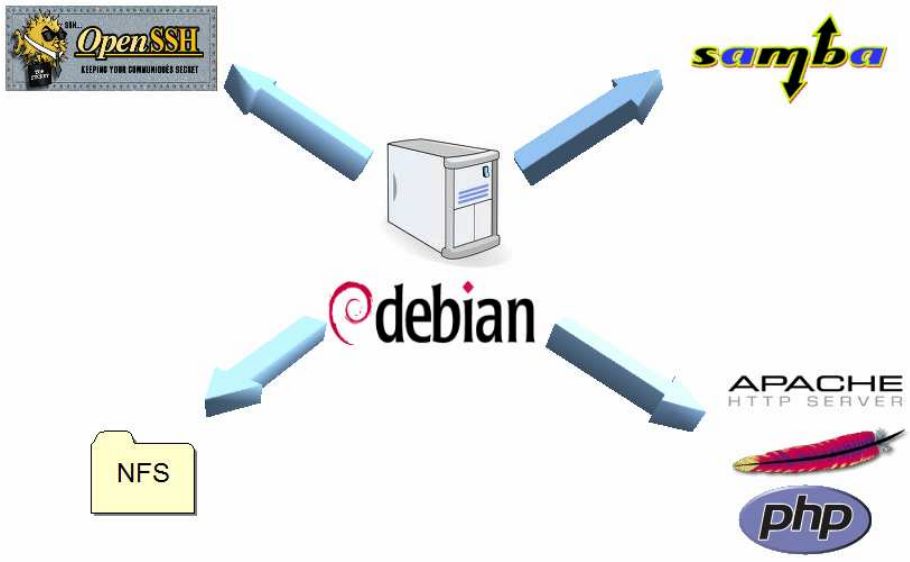


Figura 5.1 Los servicios del Servidor de Aplicaciones de Prueba

El servidor de aplicaciones fue configurado en la red con acceso a Internet. En la prueba se tienen clientes tanto en la red local, así como también otro a través de Internet.

Un esquema de este escenario se muestra en la figura 5.2.

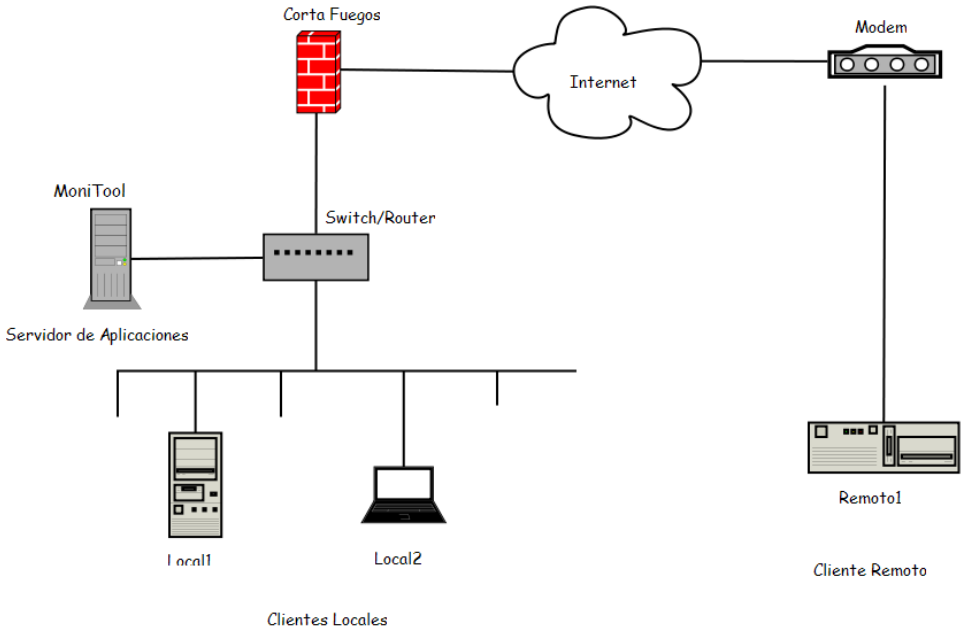


Figura 5.2 Configuración de red en la prueba

5.5 PRUEBAS Y RESULTADOS

Una vez configurados los servicios descritos en el servidor de aplicaciones y el entorno de red se inicia el mecanismo de pruebas.

Cabe mencionar que antes de realizar cualquier prueba será necesario configurar el servidor de acuerdo al esquema mostrado en la figura 5.2. Será indispensable que se tenga una dirección IP fija y pública, para que los clientes locales y remotos puedan tener acceso a cualquiera de los servicios prestados o bien a la aplicación.

Los métodos de prueba consisten en operaciones de uso común que pondrán en evidencia la funcionalidad de la aplicación y en ese momento podremos verificar que la veracidad de los datos generados por ella.

5.5.1 ACCESO AL SERVIDOR DE APLICACIONES

Para iniciar la aplicación MoniTool manualmente es necesario antes que nada abrir una Terminal en el servidor, la cuál nos permite ejecutar comandos. El acceso a esa Terminal podrá ser directamente en el servidor o vía remota a través de Security Shell.

5.5.2 AJUSTE DEL ARCHIVO DE CONFIGURACIÓN

La configuración del archivo *monitool.conf* que se utilizará en la aplicación para realizar la prueba, está en la tabla 5.1 así como la descripción de cada parámetro especificado.

Parámetro	Valor	Descripción
borrar_bitacora	1	Los datos antiguos en la bitácora de nivel 5 serán descartados
num_datos	50	50 datos en el registro temporal
num_tarjetas_red	1	Una tarjeta de red configurada en el servidor
retardo	15	15 segundos de retardo en la aplicación
cpu	80	Porcentaje máximo permitido de uso de CPU
ram	95	Porcentaje máximo permitido de uso de RAM
latencia	100	Latencia máxima en milisegundos
rate	100	Cantidad de transferencia de datos total máxima permitida
paq_perdidos	25	Porcentaje máximo de perdida de paquetes permitido
conexión	50	Número máximo de conexiones

Tabla 5.1 Configuración de la aplicación

5.5.3 INICIALIZACIÓN DE LA APLICACIÓN MONITOOL EN EL SERVIDOR

Ya que los valores han sido asignados apropiadamente a los archivos de configuración, podrá ejecutarse entonces la aplicación en la misma Terminal. Para esto basta con ejecutar el script de inicio:

```
$ cd monitool/init/
monitool/init$ sh monitool_start.sh
Iniciando Procesos Preliminares ......
[ OK ]
Iniciando monitor de CPU ......
[ OK ]
Iniciando monitor de RAM ......
[ OK ]
```

```
Iniciando Monitor de Tasa de Transferencia ......
[ OK ]
Iniciando Monitor de Conexiones ......
[ OK ]
Iniciando Monitor de Latencia ......
[ OK ]
Iniciando Monitor de Paquetes Recibidos y Enviados ......
[ OK ]
Iniciando Monitor de Paquetes Perdidos ......
[ OK ]
Iniciando Sistema de Alarmas ......
[ OK ]

MoniTool is running :)
```

Obsérvese que la aplicación devuelve la leyenda "OK" después de la correcta ejecución de cada uno de los monitores de las métricas.

5.5.4 MÉTODO DE COMPARACIÓN DE RESULTADOS

Para comparar y validar los resultados obtenidos en las pruebas realizadas, se calculará un porcentaje de error tomando como referencia el valor obtenido por la herramienta de control. Dicho porcentaje se calculará mediante la siguiente fórmula:

$$P.Error = \left(1 - \frac{Valor\ MoniTool}{Valor\ Control}\right) \cdot 100$$

5.5.5 MÉTODO DE PRUEBA PARA LA TASA DE TRANSFERENCIA

El método de prueba para la tasa de transferencia consiste en transferir un archivo grande (50 MBytes) hacia el servidor desde el cliente local (Local1) y descargar el mismo archivo hacia el cliente remoto (Remoto1) para observar y registrar los valores medidos por la aplicación. El esquema de la figura 5.2 especifica ese esquema.

Adicionalmente y de manera paralela a la aplicación, utilizaremos otro programa para medir la tasa de transferencia con la finalidad de comparar los datos medidos y corroborar su validez. El programa de control que se utilizará es "*dstat*", el cuál mide la tasa de transferencia y algunos otros parámetros.

- ✓ *Cliente Local (Local2).* Para este caso nos conectaremos hacia el servidor a través de Samba. Una vez hecho esto procedemos a transferir el archivo

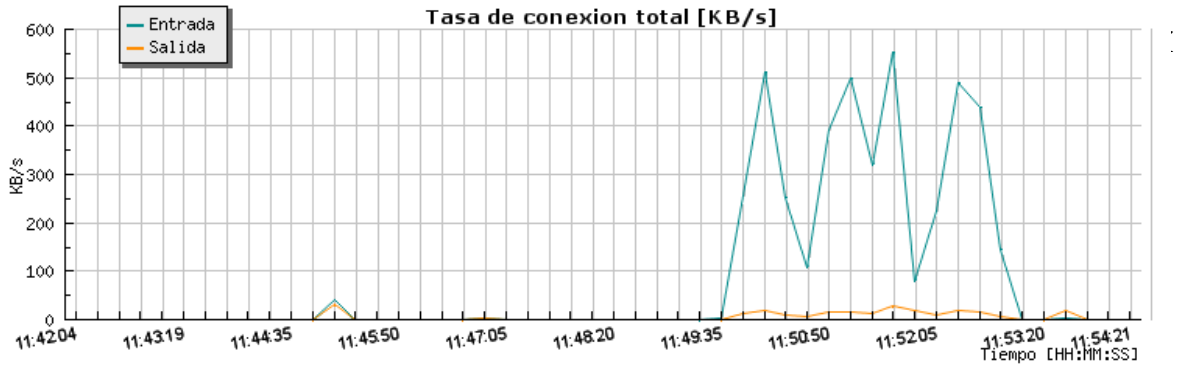


Figura 5.3 Resultado de la Primera Prueba a Tasa de Transferencia

En la gráfica resultante se puede observar que la transferencia inició en 11:49:35 y finalizó poco antes de 11:53:20. Para tener un detalle más claro de los valores registrados se utilizará la bitácora de esta métrica y se hará una comparación con los datos registrados por el programa de control "*dstat –n 15*" que con las opciones establecidas reportará medidas cada 15 segundos. Véase la tabla 5.2.

MoniTool			dstat –n 15	
Fecha/Hora	Entrada	Salida	Recibidos	Enviados
2008-05-06 11:49:35	1.90	1.00	421B	1025B
2008-05-06 11:49:50	255.80	12.40	239k	14k
2008-05-06 11:50:05	511.40	17.70	362k	14k
2008-05-06 11:50:20	253.50	9.80	297k	11k
2008-05-06 11:50:35	105.80	5.40	275k	11k
2008-05-06 11:50:50	391.60	15.10	272k	13k
2008-05-06 11:51:05	499.80	17.30	346k	16k
2008-05-06 11:51:20	318.60	13.20	296k	14k
2008-05-06 11:51:35	552.40	28.30	288k	13k
2008-05-06 11:51:50	80.10	19.80	335k	13k
2008-05-06 11:52:05	223.10	8.50	254k	12k
2008-05-06 11:52:20	489.40	18.60	336k	15k
2008-05-06 11:52:35	438.40	16.50	359k	17k
2008-05-06 11:52:50	145.70	7.30	308k	15k
2008-05-06 11:53:05	0.00	0.00	22k	8919B
Promedio Entrada 328.12 kb/s			Promedio Recibidos 305.15 kb/s	
Porcentaje de Error			**-7.52%**	

Tabla 5.2 Comparación de los Resultados de la Primera Prueba a la Tasa de Transferencia

- ✓ *Cliente Remoto (Remoto1).* Para este caso haremos una descarga de un archivo desde el cliente remoto mediante un navegador Web a través de Apache. La gráfica resultante de la operación es la siguiente (figura 5.4):

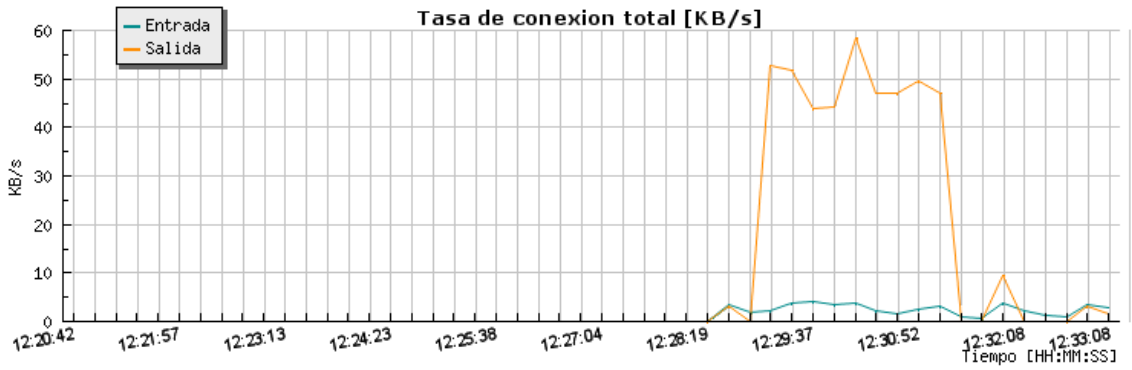


Figura 5.4 Resultado de la Segunda Prueba a Tasa de Transferencia

En la gráfica resultante se puede observar que la transferencia inició después de 12:28:19 y finalizó poco antes de 12:32:08. Nuevamente se comparan los datos recopilados por la aplicación contenidos en las bitácoras con los datos registrados por el programa de control *"dstat –n 15"*. Véase la tabla 5.3.

MoniTool			dstat –n 15	
Fecha/Hora	Entrada	Salida	Recibidos	Enviados
2008-05-06 12:28:52	1.80	0.00	0	0
2008-05-06 12:29:07	2.30	52.70	3359B	26k
2008-05-06 12:29:22	3.80	51.90	2581B	49k
2008-05-06 12:29:37	4.10	44.00	2906B	46k
2008-05-06 12:29:52	3.60	44.10	3234B	45k
2008-05-06 12:30:07	3.80	58.30	3202B	49k
2008-05-06 12:30:22	2.20	47.00	2888B	49k
2008-05-06 12:30:37	1.50	47.00	3053B	45k
2008-05-06 12:30:52	2.60	49.60	3226B	49k
2008-05-06 12:31:07	3.30	47.00	3131B	48k
2008-05-06 12:31:22	1.10	0.00	2938B	6574B
Promedio Salida 40.14 kb/s			Promedio Enviados 36.9 kb/s	
Porcentaje de Error			**8.78%**	

Tabla 5.3 Comparación de los Resultados de la Segunda Prueba a la Tasa de Transferencia

5.5.6 MÉTODO DE PRUEBA PARA EL USO DE CPU

El método de prueba para esta métrica consiste en ejecutar dos aplicaciones grandes a la vez en el servidor y observar el estado desde el cliente remoto 1. Las aplicaciones serán Open Office y Mozilla Firefox. La herramienta de control con la que se compararan los datos es nuevamente "*dstat*". La gráfica resultante es la siguiente:

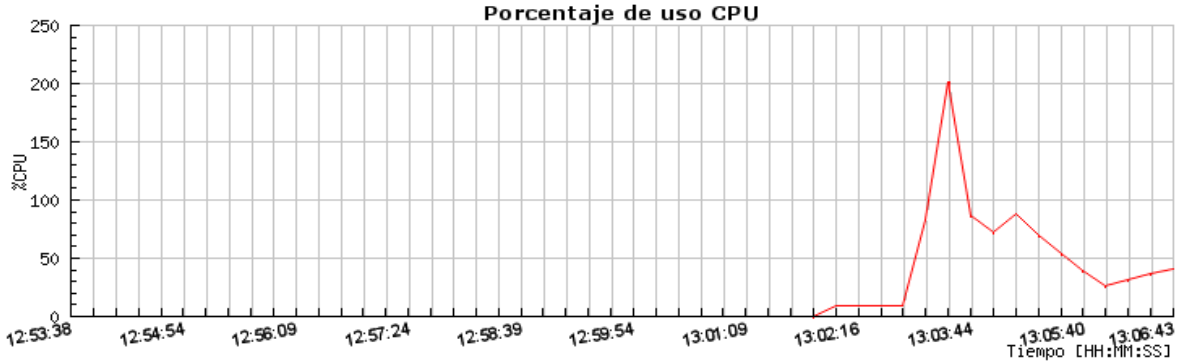


Figura 5.5 Resultado de la prueba para el uso de CPU

En la gráfica de la figura 5.5 se puede observar que la actividad inicia poco antes de 13:02:16 y se estabiliza al final de 13:06:43. Es notorio un pico que pasa del 100% del uso, y esto ocurrió porque existían en el servidor dos aplicaciones que ejecutaron procesos con prioridad alta y solicitaban toda la atención del CPU, por eso ocurre este efecto. La herramienta de control y validación de los datos es "*dstat –c 15*". La tabla 5.4 muestra la comparación de los datos obtenidos por ambas herramientas.

MoniTool		dstat –n 15
Fecha/Hora	%	% Uso
2008-05-06 13:01:09	8.60	11
2008-05-06 13:01:24	10.60	6
2008-05-06 13:01:39	8.90	10
2008-05-06 13:01:54	50.3	45
2008-05-06 13:02:01	86.0	83
2008-05-06 13:02:16	86.0	81
2008-05-06 13:02:31	89.0	78
2008-05-06 13:02:46	80.8	78
2008-05-06 13:03:04	82.80	79

2008-05-06 13:03:24	201.40	82
2008-05-06 13:03:44	86.50	83
2008-05-06 13:04:07	73.00	86
2008-05-06 13:04:32	87.90	80
2008-05-06 13:04:53	70.10	85
2008-05-06 13:05:23	54.50	89
2008-05-06 13:05:40	40.00	94
2008-05-06 13:05:56	26.80	97
2008-05-06 13:06:11	32.10	98
Promedio 65.29%		Promedio 70.27%
Porcentaje Error		**7.08%**

Tabla 5.4 Comparación de los Resultados de la Prueba a % uso de CPU

5.5.7 MÉTODO DE PRUEBA PARA LA LATENCIA

Para el caso de la latencia se utilizarán un conjunto de sitios en Internet como puntos de referencia para la prueba. La herramienta de control que se utilizará para medir la latencia será fping, que es la misma que se utiliza para esta aplicación. La diferencia radica en que se realizará esta medición desde el cliente local 1 para comparar los resultados. Los sitios de control que se utilizarán en esta prueba son los establecidos en la siguiente tabla:

Sitio	Descripción
uam.es	Universidad Autónoma de Madrid, España
google.com	Compañía ubicada en California, Estados Unidos
ipn.mx	Instituto Politécnico Nacional, México
kernel.org	Organización alojada en California
www.uba.ar	Universidad de Buenos Aires, Argentina
www.ox.ac.uk	Universidad de Oxford, Inglaterra

Tabla 5.5 Sitios de prueba que se utilizan para medir la latencia

Los sitios han sido elegidos debido a la cercanía con los backbones principales y además de ser puntos de relevancia o interés popular. La siguiente gráfica (figura 5.6) es el resultado de la operación.

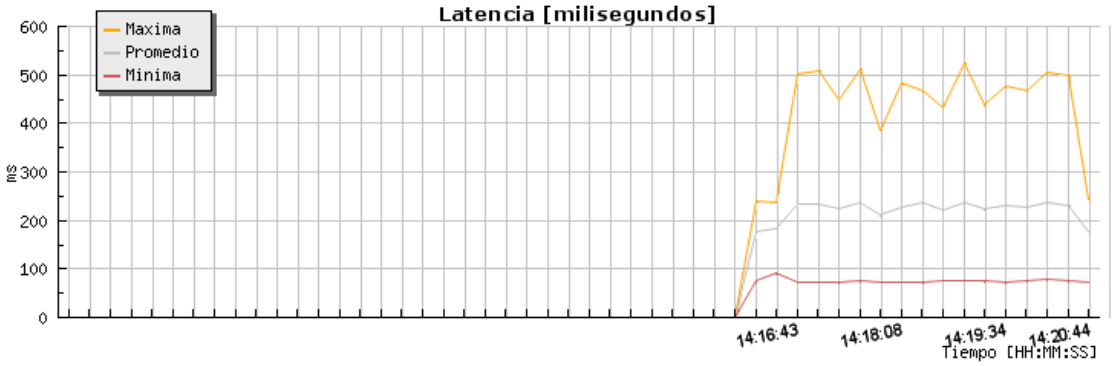


Figura 5.6 Resultado de la prueba para la Latencia

En la gráfica de la figura 5.6 se puede observar que la actividad inicia poco antes de 14:16:43. se ejecutó el comando *fping* en el cliente local 1 en 14:16:46. La salida de ese comando es la siguiente:

```
$ fping -c 1 -s < ../config/sitios.conf ; date
72.14.207.99 : [0], 84 bytes, 136 ms (136 avg, 0% loss)
kernel.org   : [0], 84 bytes, 73.9 ms (73.9 avg, 0% loss)
www.uam.es   : [0], 84 bytes, 234 ms (234 avg, 0% loss)
www.uba.ar   : [0], 84 bytes, 243 ms (243 avg, 0% loss)
www.ox.ac.uk : [0], 84 bytes, 210 ms (210 avg, 0% loss)
ipn.mx       : [0], 84 bytes, 472 ms (472 avg, 0% loss)

www.uam.es   : xmt/rcv/%loss = 1/1/0%, min/avg/max = 234/234/234
72.14.207.99 : xmt/rcv/%loss = 1/1/0%, min/avg/max = 136/136/136
ipn.mx       : xmt/rcv/%loss = 1/1/0%, min/avg/max = 472/472/472
kernel.org   : xmt/rcv/%loss = 1/1/0%, min/avg/max = 73.9/73.9/73.9
www.uba.ar   : xmt/rcv/%loss = 1/1/0%, min/avg/max = 243/243/243
www.ox.ac.uk : xmt/rcv/%loss = 1/1/0%, min/avg/max = 210/210/210

       6 targets
       6 alive
       0 unreachable
       0 unknown addresses

       0 timeouts (waiting for response)
       6 ICMP Echos sent
       6 ICMP Echo Replies received
       0 other ICMP received

 73.9 ms (min round trip time)
 228 ms (avg round trip time)
 472 ms (max round trip time)
        0.603 sec (elapsed real time)

mar may  6 14:16:46 CDT 2008
```

La salida muestra en las últimas tres líneas los resultados de la latencia medida en ese momento, se busca en las bitácoras el valor más cercano para hacer una comparación. La tabla 5.4 muestra la comparación de los datos obtenidos por ambas herramientas.

MoniTool				Fping –c 1 –s < sitios.conf		
Fecha/Hora	Mín	Promedio	Max	Mínima	Promedio	Máxima
2008-05-06 14:16:43	72.90	233.00	501.00	73.9	228	472
Porcentaje de Error				**2.19%**		

Tabla 5.6 Comparación de los Resultados de la Prueba a Latencia

5.5.8 MÉTODO DE PRUEBA PARA PAQUETES PERDIDOS

Para la prueba de los paquetes perdidos en la red, se realizará un procedimiento muy similar al punto anterior. Se considerarán los mismos sitios de prueba y la misma herramienta de control. La única variante en el procedimiento consiste en observar la respuesta a los paquetes de prueba enviados. La gráfica después de iniciar la prueba es la mostrada en la figura 5.7.

Figura 5.7 Resultado de la prueba para la Paquetes Perdidos

Podemos observar en la gráfica que no ha ocurrido perdida de paquetes. Por otro lado, la salida del comando de control es la siguiente:

```
$ fping -c 1 -s < ../config/sitios.conf ; date
72.14.207.99 : [0], 84 bytes, 137 ms (137 avg, 0% loss)
kernel.org   : [0], 84 bytes, 77.5 ms (77.5 avg, 0% loss)
```

```
www.uam.es   : [0], 84 bytes, 253 ms (253 avg, 0% loss)
www.ox.ac.uk : [0], 84 bytes, 214 ms (214 avg, 0% loss)
www.uba.ar   : [0], 84 bytes, 283 ms (283 avg, 0% loss)

www.uam.es   : xmt/rcv/%loss = 1/1/0%, min/avg/max = 253/253/253
72.14.207.99 : xmt/rcv/%loss = 1/1/0%, min/avg/max = 137/137/137
ipn.mx       : xmt/rcv/%loss = 1/0/100%
kernel.org   : xmt/rcv/%loss = 1/1/0%, min/avg/max = 77.5/77.5/77.5
www.uba.ar   : xmt/rcv/%loss = 1/1/0%, min/avg/max = 283/283/283
www.ox.ac.uk : xmt/rcv/%loss = 1/1/0%, min/avg/max = 214/214/214

       6 targets
       5 alive
       1 unreachable
       0 unknown addresses

       0 timeouts (waiting for response)
       6 ICMP Echos sent
       5 ICMP Echo Replies received
       0 other ICMP received

 77.5 ms (min round trip time)
 193 ms (avg round trip time)
 283 ms (max round trip time)
        0.625 sec (elapsed real time)

mar may  6 14:38:04 CDT 2008
```

Podemos observar que la respuesta del sitio *ipn.*mx nunca llegó, por lo que ese paquete se perdió en el trayecto, a diferencia de la prueba realizada por la aplicación. Los resultados de las pruebas se resumen en la tabla

MoniTool			fping –c 1 –s < sitios.conf	
Fecha/Hora	Enviados	Recibidos	Paquetes Enviados	Recibidos
2008-05-06 14:38:10	12.00	12.00	6	5
(12/12) * 100 = 100%			(5/6) * 100 = 83.3%	
Porcentaje de Error			**-20.48%**	

Tabla 5.7 Comparación de los Resultados de la Prueba a Paquetes Perdidos

5.5.9 MÉTODO DE PRUEBA PARA EL NÚMERO DE CONEXIONES

Para probar el número de conexiones se utilizará una aplicación que mide la cantidad de conexiones que han sido establecidas en el servidor; ésta herramienta es "*netstat –n*". La siguiente gráfica (figura 5.8) muestra los resultados reportados por la aplicación:

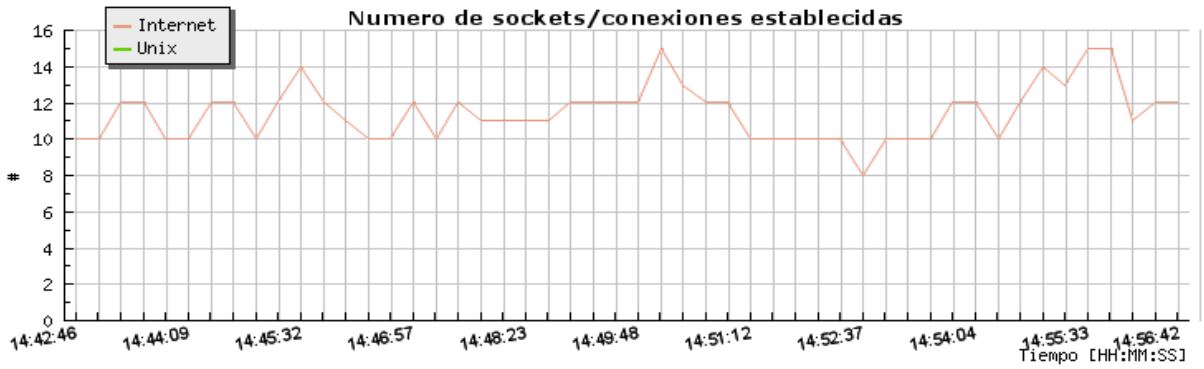


Figura 5.8 Resultado de la prueba para el número de conexiones

El resultado del comando de control es el siguiente:

```
$ netstat -n | grep tcp ; date
tcp        0   0 189.141.89.88:445    192.168.1.146:1193      ESTABLECIDO
tcp6       0   0 189.141.89.88:80     192.168.1.146:3078      TIME_WAIT
tcp6       0   0 189.141.89.88:80     192.168.1.146:3077      TIME_WAIT
tcp6       0   0 189.141.89.88:80     192.168.1.146:3076      TIME_WAIT
tcp6       0   0 189.141.89.88:80     192.168.1.146:3075      TIME_WAIT
tcp6       0   0 189.141.89.88:80     192.168.1.146:3074      TIME_WAIT
tcp6       0   0 189.141.89.88:80     192.168.1.146:3082      ESTABLECIDO
tcp6       0   0 189.141.89.88:80     192.168.1.146:3083      ESTABLECIDO
tcp6       0   0 189.141.89.88:80     192.168.1.146:3075      TIME_WAIT
tcp6       0  48 189.141.89.88:22     192.168.1.146:1839      ESTABLECIDO
mar may  6 14:51:21 CDT 2008
```

En la salida del comando podemos observar que existen 10 conexiones activas en el servidor de aplicaciones. Para comparar los resultados apropiadamente, compararemos el valor registrado en las bitácoras que se encuentre mas cerca al momento en el que este comando fue ejecutado. La comparación de los resultados se muestra en la tabla 5.8.

MoniTool		netstat –n | Grez tcp
Fecha / Hora	# Conexiones	# Conexiones
2008-05-06 14:51:12	10.00	10
Porcentaje de Error		**0%**

Tabla 5.8 Comparación de los Resultados de la Prueba a Número de Conexiones

5.5.10 MÉTODO DE PRUEBA PARA EL USO DE MEMORIA RAM

El método de prueba de la RAM utilizada consiste en comparar la cantidad de memoria reportada en algún momento y comparar el valor obtenido por la aplicación con una herramienta de control. Esta herramienta es "*top*" la cuál determina la cantidad de memoria total, usada y libre entre otros parámetros.

La figura 5.9 muestra la gráfica presentada por la aplicación:

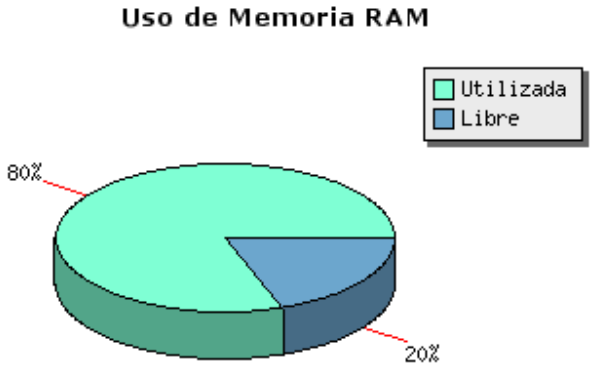


Figura 5.9 Resultado de la prueba para la RAM

La salida del comando *top* es la siguiente:

```
  top - 15:13:59 up  5:51,  2 users,  load average: 1.85, 1.73, 1.67
Tasks: 158 total,   2 running, 156 sleeping,   0 stopped,   0 zombie
Cpu(s): 98.3%us,0.3%sy,0.3%ni,0.0%id,0.0%wa,0.0%hi,1.0%si,  0.0%st

Mem:   1035400k total,   833004k used,   202396k free,   101492k buffers
Swap:   473876k total,      300k used,   473576k free,   458244k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
29816 hselley   20   0 50768 6424 5160 S 93.0  0.6 115:05.91 evolution-
data-
 7236 hselley   20   0 86268  30m 8304 S  1.3  3.1   6:35.57 compiz.real
 6546 root      20   0 79032  34m 8312 S  1.0  3.4  13:16.98 Xorg
 7280 hselley   20   0 24452  11m 8576 S  0.3  1.1   1:05.03 multiload-
apple
13632 www-data  20   0 23912 8940 3412 S  0.3  0.9   1:12.69 apache2
    1 root      20   0  2844 1692  544 S  0.0  0.2   0:01.24 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:00.18 ksoftirqd/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:00.46 events/0
```

```
  7 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
 42 root      15  -5     0    0    0 S  0.0  0.0   0:00.32 kblockd/0
 45 root      15  -5     0    0    0 S  0.0  0.0   0:02.46 kacpid
 46 root      15  -5     0    0    0 S  0.0  0.0   0:02.16 kacpi_notify
123 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
161 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
162 root      20   0     0    0    0 S  0.0  0.0   0:01.08 pdflush
163 root      15  -5     0    0    0 S  0.0  0.0   0:00.20 kswapd0
206 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
499 root      20   0  7464 1260  868 S  0.0  0.1   0:00.12 nmbd
```

En las primeras líneas de la salida podemos ver que se despliega la cantidad de memoria total, usada y libre. Después de calcular el porcentaje correspondientes se podrá comparar el resultado de esta herramienta con el obtenido por la aplicación.

MoniTool		top
Fecha / Hora	Usada	% Memoria Usada
2008-05-06 15:13:59	80.45 %	(202396/1035400) * 100 = 80.45%
Porcentaje de Error		**0%**

Tabla 5.8 Comparación de los Resultados de la Prueba a Latencia

5.5.11 MÉTODO DE PRUEBA PARA PAQUETES ENVIADOS Y RECIBIDOS

El método utilizado para probar la cantidad de paquetes enviados y recibidos en las interfaces de red consiste en comparar el valor obtenido por la aplicación en algún momento y compararlo con la herramienta de control "*netstat -s*". La gráfica mostrada en la figura 5.9 es la medición generada por la aplicación.

Figura 5.9 Resultado de la prueba para Paquetes Enviados y Recibidos

La salida del comando de control es la siguiente:

```
$ netstat -s | head -8 ; date
Ip:
    318437 total de paquetes recibidos
    202 con direcciones incorrectas
    0 reenviados
    0 paquetes entrantes desechados
    318234 paquetes entrantes servidos
    413522 peticiones enviadas
    58 descartados debido a una ruta inexistente
mar may  6 16:02:56 CDT 2008
```

Finalmente se hará un análisis y comparación de los resultados obtenidos por ambas herramientas. La tabla 5.9 muestra estos resultados:

MoniTool			Netstat -s	
Fecha / Hora	Recibidos	Enviados	Recibidos	Enviados
2008-05-06 16:02:55	318432	413516	318437	413522
Porcentaje de Error P. Recibidos			**0.0015%**	
Porcentaje de Error P. Enviados			**0.0014%**	

Tabla 5.9 Comparación de los Resultados de la Prueba a Paquetes Enviados y Recibidos

5.5.12 DETENCIÓN DE LA APLICACIÓN MONITOOL

Para probar que la aplicación detiene su ejecución y que se finalizan todos los procesos que fueron lanzados, se ejecuta el siguiente bash-script en la Terminal de la siguiente forma:

```
$ cd monitool/init/
monitool/init$ sh monitool_stop.sh
Finalizando Monitor CPU ...
[ OK ]
Finalizando Monitor RAM ...
[ OK ]
Finalizando Monitor de Tasa de Transferencia ......
[ OK ]
Finalizando Monitor de Conexiones ......
[ OK ]
Finalizando Monitor de Latencia ......
[ OK ]
Finalizando Monitor de Paquetes Recibidos y Enviados ......
[ OK ]
Finalizando Monitor de Paquetes Perdidos ......
[ OK ]
```

5.6 CONCLUSIONES

Las pruebas se han hecho para cada métrica considerando un valor de control obtenido mediante una herramienta de monitoreo distinta para esa métrica. Una vez obtenidos los datos de la métrica en cuestión, se compara con los datos obtenidos por la aplicación MoniTool y se calcula un porcentaje de error.

Cabe mencionar que a pesar de que las pruebas fueron efectuadas con mucho cuidado, en algunos casos la medición realizada por la aplicación y por la herramienta de control, no fueron totalmente sincronizadas. Esto puede ocasionar que exista una diferencia en los datos obtenidos, que en un instante distinto la métrica arroje un valor diferente y por ende provocar un porcentaje de error un poco mayor.

A pesar de esta situación, los porcentajes de error obtenidos por la aplicación en la mayoría de las pruebas no son muy grandes y demuestran que los datos pueden ser confiables para un propósito general en el que no se requiere una gran aproximación en los datos obtenidos.

5.7 RESUMEN

En este capítulo se han descrito y realizado diversas pruebas de funcionamiento a la aplicación MoniTool mediante un método que compara los resultados obtenidos por esta contra algunas herramientas de control. Se hizo un análisis de los resultados obtenidos, mostrando un porcentaje de error que muestra la veracidad de los datos obtenidos.

CAPÍTULO 6

Conclusiones

6.1 LOGROS ALCANZADOS

Para hablar de algún logro conseguido es imprescindible recordar cuales fueron los objetivos planteados en primera instancia. Por tal motivo, se mostrará la forma en que fueron cumplidos los objetivos de acuerdo al desarrollo del presente trabajo de tesis.

De acuerdo al objetivo general, se propuso diseñar una metodología de evaluación, tal que por medio de un conjunto de métricas seleccionadas a partir de un análisis profundo del estado del arte se pudiera medir el desempeño que experimenta un servidor de aplicaciones cuando está dando servicio a una gran cantidad de usuarios. Se planteó además desarrollar la herramienta con la que se pudiera auditar la calidad de servicio ofrecida.

De esta forma, y usando como base el diseño y los resultados de las pruebas, se puede observar que la herramienta realiza un monitoreo permanente por medio de un conjunto de métricas justificadas. Tal monitoreo proporciona valores que son datos suficientes para que el usuario pueda observarlos y concluir cual es el desempeño que está experimentando el servidor en algún momento. Además dada la independencia de la ubicación de los datos de las distintas métricas, permite que el usuario pueda observar alguna métrica en especial para poder diseñar a partir de ellos algunas medidas correctivas.

Para el resto de los logros se hará un listado de las capacidades que posee la herramienta y que a su vez cumplen con los objetivos específicos planteados inicialmente.

- ✓ Se seleccionó un conjunto de métricas que reflejan el comportamiento general que experimenta un servidor de aplicaciones.

- ✓ Se propuso una metodología de medición del desempeño de los servidores de aplicación, la cuál fue implementada con la herramienta desarrollada.

- ✓ La herramienta posee un sistema de monitoreo permanente que genera nuevos datos en todo momento, para que el usuario tenga a su disposición la información cuando la requiera.

- ✓ La herramienta muestra los datos recién capturados por el monitoreo, de manera que los datos desplegados serán en tiempo real.

- ✓ La herramienta cuenta con una interfaz Web que permite acceder al usuario a los datos capturados por la etapa de monitoreo que son mostrados mediante gráficas que permiten que el usuario pueda tener una fácil comprensión y seguimiento de ellos. La pantalla de despliegue de gráficas es muy sencilla por lo que el usuario solo tendrá que ingresar a ella y observar el movimiento de las gráficas, ya que la herramienta se actualiza automática de manera periódica mostrando los últimos datos obtenidos en la etapa de monitoreo.

- ✓ Otra gran ventaja de la interfaz Web, es que mediante el servidor Apache es posible ingresar a la herramienta desde cualquier punto de la red, o bien si el servidor está configurado apropiadamente, desde cualquier lugar del planeta a través de Internet.

- ✓ Gracias a una gran característica del calendarizador del sistemas operativo Linux/UNIX es posible asignarle a un proceso una prioridad de ejecución, de esta forma en el diseño de esta herramienta se ha especificado que todos los procesos que genere tendrán una prioridad baja y por lo tanto perturbarán de manera mínima al desempeño del servidor de aplicaciones.

- ✓ MoniTool posee un conjunto de archivos de configuración que permiten al usuario ajustar la aplicación de acuerdo a sus necesidades específicas o bien la configuración de la red local en la que se encuentra el servidor, de manera tal que el comportamiento de la herramienta obedece a estos archivos, lo que le da la característica de configurabilidad.

Una característica muy importante que no fue planteada en los objetivos iniciales, pero que fue implementada en el desarrollo de la herramienta, es el sistema de bitácoras.

Este sistema permite tener un conjunto de archivos que almacenan todos los datos generados por el monitoreo constante de la aplicación, y además permite que el usuario pueda consultar posteriormente estos datos sin la necesidad de estar frente a la pantalla en el momento en el que ocurran.

Adicionalmente el sistema es capaz de detectar la transición de un día a otro, y cuando esto ocurre, desplazar el contenido de los datos almacenados en las bitácoras de tal suerte que en el primer archivo siempre se encuentren los datos correspondientes al día en curso.

6.2 APORTACIONES

La metodología propuesta es eficaz para la medición del desempeño de aplicaciones de acuerdo a los resultados obtenidos en el capítulo 5.

A partir de tal metodología, se desarrolló una herramienta capaz de monitorear un servidor de aplicaciones, mostrar los resultados del monitoreo realizado mediante gráficas a través de un servidor Web que permite visualizar la información en cualquier punto de la red o inclusive en Internet. La aplicación además almacena los datos capturados en bitácoras que podrán ser visualizadas mediante la misma interfaz Web.

6.3 PUBLICACIÓN DE LOS RESULTADOS DE LA TESIS

Artículos Nacionales:

Monitoreo del Comportamiento de Servidores de Aplicaciones
Segundo Congreso Mexicano de Ingeniería en Comunicaciones y Electrónica 2007
AMIICE 6, 7 y 8 de Junio del 2007. México D.F.
Héctor Julián Selley Rojas, Dr. Felipe Rolando Menchaca García

MoniTool: Herramienta de Monitoreo de Servidores de Aplicaciones
II Magno Congreso Internacional de Computación CIC-IPN 2007
Exposición de Proyectos de Laboratorios, CIC-IPN 6, 7 y 8 de Noviembre del 2007
Héctor Julián Selley Rojas, Dr. Felipe Rolando Menchaca García

MoniTool: Herramienta de Monitoreo de Servidores de Aplicaciones
1er. Ciclo de Conferencias en Tecnología Informática y Software Libre 2007
COTISOL 14 de Diciembre del 2007. CONALEP, Iztapalapa, México D.F.
Héctor Julián Selley Rojas, Dr. Felipe Rolando Menchaca García

MoniTool: Herramienta de Monitoreo de Servidores de Aplicaciones
Segundo Foro de Software Libre 2008
2, 3, y 4 de Abril del 2008, Tlaxcala, México
Héctor Julián Selley Rojas, Dr. Felipe Rolando Menchaca García

MoniTool: Herramienta de Monitoreo de Servidores de Aplicaciones
Festival Lationoamericano de Instalación de Software Libre 2008
FLISOL 26 de Abril 2008, México D.F.
Héctor Julián Selley Rojas, Dr. Felipe Rolando Menchaca García

6.4 TRABAJOS FUTUROS

A continuación se mencionan algunas posibilidades de crecimiento de la presente tesis.

- ***Autenticación de acceso a la aplicación***. La herramienta despliega en una página Web las gráficas sin restricción de algún tipo, una posibilidad sería implementar alguna metodología que permita tener un acceso restringido a la información, mediante un sistema de autenticación de usuario.

- ***Almacenamiento de las bitácoras en una base de datos***. El almacenamiento de los datos de las bitácoras en una base de datos podría elevar la funcionalidad de ellas, ya que con esto se podrían realizar búsquedas de fechas o valores pico específicos y mostrar una cantidad reducida y concisa de los datos monitoreados. Esto podría facilitar la tarea de estimar el desempeño y encontrar fallas en el servidor.

- ***Compatibilidad con otros sistemas operativos***. Actualmente la aplicación funciona sobre sistemas Linux/UNIX, implementar la compatibilidad con otros sistemas operativos incrementaría la versatilidad y utilidad que puede tener esta herramienta de monitoreo. Debido a que las herramientas de software empleadas para el desarrollo del software son de código abierto, existe una versión para sistemas basados en Windows es viable implementar la aplicación para tales sistemas operativos.

- ***Agregar nuevas métricas a la herramienta***. A pesar de que el conjunto selecto de las métricas empleadas son parte fundamental de la propuesta del presente proyecto, el usuario final podría tener la necesidad de agregar nuevas métricas dadas algunas necesidades que surjan en su ambiente de trabajo. Debido a la naturaleza intrínseca modular y secuencial de la aplicación, permite que sean agregados nuevos módulos sin afectar directamente al funcionamiento de las existentes.

6.5 COMENTARIOS FINALES

En la actualidad la calidad de servicio es un parámetro que se ha vuelto una necesidad, un factor al que la industria le ha dado la importancia que merece, ya que un usuario tiene la capacidad de decidir entre diversos proveedores de acuerdo a la calidad con la que ofrezcan sus servicios.

La herramienta que se desarrolló tiene la intención de ayudar a los administradores de servidores de aplicaciones a determinar la calidad del servicio que se ofrece, mediante un monitoreo constante de los parámetros que en términos generales dictan el desempeño de sus servicios.

El sistema de bitácoras permite que el usuario pueda analizar el comportamiento del servidor de días anteriores, y por lo tanto no preocuparse por estar todo el tiempo observando la pantalla y si existe algún cambio importante en los parámetros medidos.

Debido a ser ligera y transportable, esta aplicación resulta muy útil e imprescindible para cualquier tipo de servidor de aplicaciones, permitiendo tener un esbozo general del comportamiento de la aplicación rápida y fácilmente.

ANEXO 

Glosario

Ada. Lenguaje de programación estructurado diseñado por Jean Ichbiah de CII Honeywell Bull por encargo del Departamento de Defensa de los Estados Unidos. Es un lenguaje multipropósito orientado a objetos y concurrente, pudiendo llegar desde la facilidad de Pascal hasta la flexibilidad de C++

Apache. Este proyecto es un esfuerzo por desarrollar y mantener un servidor HTTP para los sistemas operativos modernos. El objetivo de este proyecto es proveer un servidor seguro, eficiente y extensible que provea servicios HTTP de acuerdo a los estándares actuales.

Base de datos. Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su uso posterior.

Bash. Es un intérprete de comandos del proyecto GNU. Es un acrónimo de Bourne Again Shell, donde Bourne fue el primer intérprete de comandos importante en UNIX.

Bit. Acrónimo de Binary Digit, un bit es la unidad mínima en la numeración binaria y solo puede tomar el valor 1 ó 0.

BSD. (Berkeley Software Distribution) Sistema operativo derivado del sistema Unix nacido a partir de las aportaciones realizadas a ese sistema por la Universidad de California en Berkeley.

Byte. Conjunto de ocho bits. Es el prefijo que se usa como base en combinación de otros prefijos de cantidad. Utilizado en computación y electrónica.

C. Lenguaje de programación creado en 1972 por Thompson y Dennis M. Ritchie y sucesor del lenguaje B. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel.

C++. Lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. Es un lenguaje que

abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Cisco Systems. Empresa multinacional ubicada en San José California, Estados Unidos, dedicada principalmente a la fabricación, venta, mantenimiento y consultoría de equipos de telecomunicación.

Código abierto (Open Source). También llamado como Software Libre, es el término mediante al cuál se define a todas las aplicaciones computacionales que se desarrollan y distribuyen libremente. Todo el software desarrollado de esta forma brinda libertad de uso, instalación, modificación y redistribución.

CPU. (Central Processing Unit) Unidad Central de Procesamiento, se refiere al micro-procesador de una computadora el cuál se encarga de realizar todas las operaciones lógicas y aritméticas.

Debian GNU/Linux. Es un sistema operativo libre para su computadora. Utiliza el núcleo Linux (el corazón del sistema operativo), pero la mayor parte de las herramientas básicas vienen del Proyecto GNU; de ahí el nombre GNU/Linux.
En el Proyecto Debian una comunidad conformada por desarrolladores y usuarios, pretende crear y mantener un sistema operativo GNU basado en software libre precompilado y empaquetado, en un formato sencillo en múltiples arquitecturas de computador y en varios núcleos. Nace como una apuesta por separar en sus versiones el software libre del software no libre.

DHCP. (Dynamic Host Configuration Protocol) Protocolo de Configuración Dinámica de Host, es el protocolo que se encarga de asignar la configuración de red automáticamente a los clientes que se conecten a la red donde este se encuentre.

DNS. (Domain Name Server) Servidor de Nombres de Dominio. Este servidor se encarga de resolver los nombres de host solicitadas por los clientes y entregarle la dirección IP que corresponda para poder establecer una comunicación.

Exim. (Experimental Internet Mailer). Es un agente de transporte de correo (MTA) desarrollado por la Universidad de Cambridge para su uso en sistemas Unix conectados en Internet. Es distribuido libremente dada la licencia GNU – GPL.

Fortran. (Formula Translator). Lenguaje de programación desarrollado en los años 50 y activamente utilizado desde entonces. Se utiliza principalmente en aplicaciones científicas y análisis numérico. Si bien el lenguaje era inicialmente un lenguaje imperativo, las últimas versiones incluyen elementos de la programación orientada a objetos.

GCC. (GNU Compiler Collection) Es una colección de compiladores creados por el proyecto GNU. Es software libre y lo distribuye la Free Software Foundation bajo la licencia GPL. Estos compiladores se consideran estándar para los sistemas operativos derivados de UNIX, de código abierto o también de propietarios, como Mac OS X.

GNU. (GNU's Not Unix) Este proyecto fue creado en 1984 con la intención de desarrollar un sistema operativo entero tipo Unix que fuese software libre: el sistema GNU. Dado que el kernel GNU no se encontraba terminado, se comenzó a utilizar con el kernel Linux, a esta combinación se le conoce como sistema operativo GNU/Linux, que es utilizado por millones en la actualidad. Algunas veces se utiliza incorrectamente la combinación como solamente Linux.
A la conjunción de los programas de código abierto o software libre GNU y el kernel de Linux se le conoce como una distribución, las cuales contienen software 100% libre.

GPL. (General Public License) La Licencia Pública General de GNU, llamada comúnmente GNU GPL, la usan la mayoría de los programas de GNU y más de la mitad de las aplicaciones de software libre. Esta licencia otorga derechos de uso, instalación, modificación y redistribución, siempre y cuando se mantenga la licencia del software en cuestión.

Hardware. Se refiere a todos los componentes físicos (que se pueden tocar), en el caso de una computadora personal serían los discos, unidades de disco, monitor, teclado, la placa base, el microprocesador, etcétera

HP. (Hewlett Packard) Es la mayor empresa de tecnologías de la información del mundo con sede en Palo Alto, California. Fabrica y comercializa hardware y software además de brindar servicios de asistencia relacionados con la informática. La compañía fue fundada en 1939 y se dedicaba a la fabricación de instrumentos de medida electrónica y de laboratorio. Hoy en día es la empresa líder en venta de impresoras.

HTML. (HyperText Markup Lenguaje) El Lenguaje de Etiquetas de Hiper Texto es el lenguaje de marcado predominante para la construcción de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

HTTP. (Hyper TexT Protocol). El Protocolo de Transferencia de Hiper Texto es usado en cada transacción de la Web. Fue desarrollado por el consorcio W3C y la IETF. Define la sintaxis y la semántica que utilizan los elementos software de la arquitectura Web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

IBM. (International Business Machines Corporation) Es una corporación multinacional en tecnología y consultoría de cómputo cuyas oficinas centrales se encuentran en Armonk, Nueva York. Construye y vende hardware y software para computadoras, y ofrece servicios de infraestructura y consultoría en áreas que van desde nanotecnología hasta supercomputadoras.

Internet. Red mundial de computadoras con un conjunto de protocolos, siendo TCP/IP el más destacado. Aparece por primera vez en 1969, cuando ARPAnet establece su primera conexión entre tres universidades en California y una en Utah. Cuando se dice red de redes se hace referencia a que es una red formada por la interconexión de otras redes menores.

IP. (Internet Protocol) Protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

ISP. (Internet Service Provider) Un Proveedor de Servicios de Internet es una compañía que se encarga de brindar servicios de Internet.

Kernel/núcleo. Es el centro o corazón del sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

Latencia. Es la cantidad de tiempo de traslado que le toma a un paquete el viajad del origen al destino. Parámetro usado para estimar el tráfico en una red.

LDAP. (Lightweight Directory Access Protocol) Es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorios ordenado y distribuido para acceder a diversa información en un entorno de red.

Linux. Es el nombre de un kernel desarrollado como software libre basado en Unix. Cuando es integrado con un conjunto de aplicaciones GNU se forma un sistema operativo llamado “distribución Linux/GNU”.

Métrica. Es un parámetro que afecta directamente a la calidad que ofrece algún servicio otorgado.

Middleware. Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.

MoniTool. (Monitoring Tool) Herramienta de software que permite medir el desempeño de un servidor de aplicaciones a través de graficas, generadas mediante el monitoreo de un conjunto de métricas.

MySQL. Sistema de gestión de base de datos relacional, multihilo y multiusuario desarrollado por MySQL AB con un esquema de licenciamiento dual, GNU y privativo.
NFS. (Network File System) Es un servicio para compartir un sistema de archivos en Unix. Permite que distintos sistemas puedan acceder a archivos remotos tal como si fueran locales.

OpenSSH. Versión libre de las herramientas de conectividad SSH, desarrollado por el proyecto OpenBSD.

Paquete. Es la unidad mínima en que se divide la información cuando es transferida mediante Internet. Posee una estructura y tamaño diferente dependiendo del protocolo que lo utilice.

Pascal. Lenguaje de programación desarrollado por el profesor suizo Niklaus Wirth a finales de los años 60. Su objetivo era crear un lenguaje que facilitara el aprendizaje de la programación a sus alumnos. Sin embargo con el tiempo su utilización excedió el ámbito académico para convertirse en una herramienta para la creación de aplicaciones de todo tipo.

Perl. (Practical Extraction and Reporting Language) Lenguaje de programación diseñado por Larry Wall creado en 1987. Toma características de C, del lenguaje interpretado shell, AWK, sed, Lisp y, en un grado inferior, muchos otros lenguajes de programación.

PHP. (PHP Hypertext Pre-processor) Lenguaje interpretado de uso general ampliamente utilizado. Diseñado para el desarrollo de aplicaciones Web dinámicas y puede ser integrado en el código HTML.

Python. Lenguaje de programación creado por Guido van Rossum en el año 1990. En la actualidad se desarrolla como un proyecto de código abierto, administrado

por la Python Software Foundation. Considerado como la "oposición leal" a Perl, lenguaje con el cual mantiene una rivalidad amistosa.

QoS. (Quality Of Service) Son las tecnologías que garantizan la transmisión de cierta cantidad de datos en un tiempo dado (throughput). Calidad de servicio es la capacidad de dar un buen servicio.

RAM. (Random Access Memory) Compuesta por chips y se utiliza como memoria de trabajo para programas y datos. Es un tipo de memoria temporal que pierde sus datos cuando se queda sin energía, por lo cual es una memoria volátil.

Ruby. Lenguaje de programación reflexivo y orientado a objetos, creado por Yukihiro Matsumoto en 1995. Combina una sintaxis inspirada en Python, Perl con características de programación orientada a objetos similares a Smalltalk. Ruby es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre.

Samba. Una implementación libre del protocolo de archivos compartidos de Microsoft Windows para sistemas de tipo UNIX. De esta forma, es posible que ordenadores con Linux o Mac OS X se vean como servidores o actúen como clientes en redes de Windows.

Sendmail. Popular "agente de transporte de correo" (MTA) en Internet, cuya tarea consiste en encaminar los mensajes correos de forma que estos lleguen a su destino. Se afirma que es el más popular MTA, corriendo sobre sistemas Unix y el responsable de la mayoría de envío del correo de Internet, aunque se le critica su alto número de alertas de seguridad (la mayoría de ellas parchadas a las pocas horas), además de no ser sencillo de configurar.

Servidor de archivos. Tipo de servidor en una red de ordenadores cuya función es permitir el acceso remoto a archivos almacenados en él o directamente accesibles por este.

Shell. Es el intérprete de comandos, el cual es un programa informático que actúa como Interfaz para comunicar al usuario con el sistema operativo mediante una ventana que espera órdenes escritas a través del teclado, los interpreta y los entrega al sistema operativo para su ejecución. La respuesta del sistema operativo es mostrada al usuario en la misma ventana.

Software. Es el conjunto de los componentes intangibles de una computadora, es decir, el conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica

SSH. (Security Shell) Protocolo que permite ejecutar una Terminal remota virtual. Sirve para acceder a máquinas remotas a través de una red y manejar por completo la computadora mediante un intérprete de comandos.

SSL. Protocolo criptográfico que proporciona comunicaciones seguras en Internet. Proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía.

Sun Microsystems. Empresa informática de Silicon Valley, fabricante de semiconductores y software. Fue constituida en 1982 por el alemán Andreas von Bechtolsheim y los norteamericanos Vinod Koshla, Bill Joy, Scott McNealy y Marcel Newman. Las siglas SUN se derivan de «Stanford University Network», proyecto que se había creado para interconectar en red las bibliotecas de la Universidad de Stanford.

TCP. (Transmission Control Protocol) Es uno de los protocolos fundamentales en Internet, fue creado entre los años 1973 - 1974 por Vint Cerf y Robert Kahn. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. TCP da soporte a muchas de las aplicaciones más populares de Internet, incluidas HTTP, SMTP y SSH.

Tux. Nombre de la mascota del proyecto GNU/Linux. Creado por Larry Ewing en 1996, es un pequeño pingüino de aspecto risueño y cómico. La idea de que la mascota de kernel de Linux fuera un pingüino provino del mismo Linus Torvalds, creador de kernel de Linux. Según se cuenta, cuando era niño le picó un pingüino, y le resultó simpática la idea de asociar un pingüino a su proyecto.

UNIX. Sistema operativo portable, multitarea y multiusuario, desarrollado en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

Voz IP. Voz sobre Protocolo de Internet, también llamado Voz sobre IP, VozIP, VoIP (por sus siglas en inglés), o Telefonía IP, es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP.

WebDAV. Este protocolo proporciona funcionalidades para crear, cambiar y mover documentos en un servidor remoto (típicamente un servidor Web). Esto se utiliza sobre todo para permitir la edición de los archivos en un servidor Web, pero puede también aplicarse a sistemas de almacenamiento generales basados en Web, que pueden ser accedidos desde cualquier lugar. La mayoría de los sistemas

operativos modernos proporcionan soporte para WebDAV, haciendo que los ficheros de un servidor aparezcan como un directorio local.

WWW (World Wide Web) Nombrado como "Web" solamente es un sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas Web que pueden contener texto, imágenes, vídeos u otros contenidos multimedia, y navega a través de ellas usando hiperenlaces

ANEXO B

Manual de Usuario

- **Instalación**

El procedimiento de instalación de la aplicación es realmente sencillo, solo descomprima y desempaquete el archivo en algún punto del directorio personal del servidor que desea monitorear. Asegúrese que el procedimiento sea llevado a cabo sin errores.

Posteriormente deberá instalar los siguientes paquetes en el sistema operativo, que son necesarios para la ejecución de la aplicación.

- Apache HTTP Server 2
- Módulo php de apache
- Php 5
- Biblioteca jpgraph de php
- Módulo GD de php
- Fuentes Msttcorefonts
- Ifstat
- Fping
- Free

El procedimiento para instalar los paquetes en Debian es el siguiente:

```
#apt-get install apache2 libapache2-mod-php5 php5 \
libphp-jpgraph php5-gd msttcorefonts ifstat fping free
```

Apt se encargará de resolver todas las dependencias que sean necesarias instalar en el sistema para cada uno de los paquetes.

- **Configuración**

Este punto se harán los ajustes necesarios en el sistema operativo para que la aplicación pueda ejecutarse.

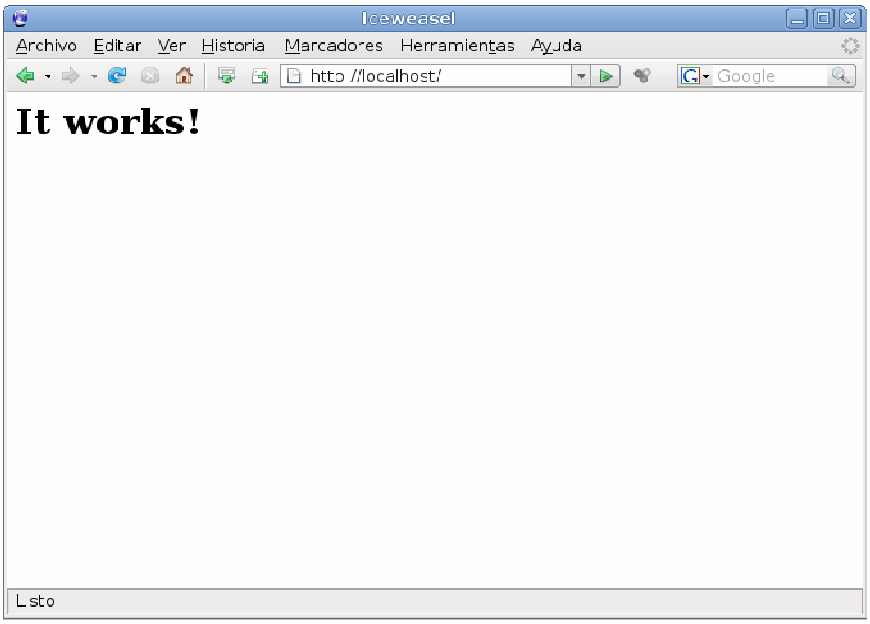
- Abra el archivo */etc/php5/apache2/php.ini* y en la sección *"Paths and Directories"* añada la ubicación de las bibliotecas *jpgraph*. Deberá quedar de la siguiente forma:

```
; UNIX: "/path1:/path2"
include_path = ".:/usr/share/php:/usr/share/jpgraph/"
```

- Reinicie el demonio de Apache para cargar los cambios:

```
# /etc/init.d/apache2 restart
Restarting Web server: apache2.
```

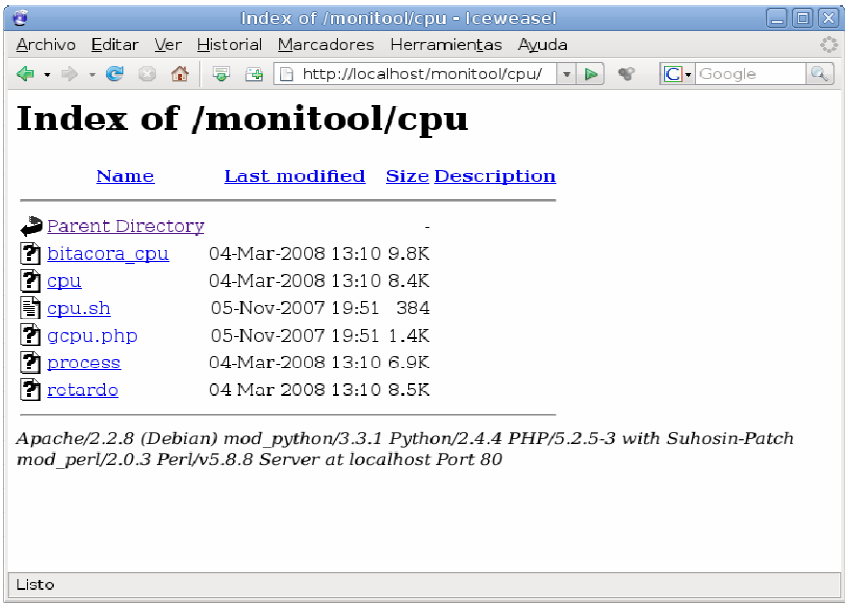
- Compruebe el funcionamiento de Apache. Abra un navegador Web y escriba "localhost" en la barra de direcciones. La salida deberá ser similar a la siguiente:



- Publicación vía Apache. Para poder acceder a los scripts PHP a través de Apache, en el servidor deberá crear un enlace suave del directorio donde se encuentra la aplicación hacia la ubicación predeterminada de publicación en Apache.

```
# ln -s /home/usuario/monitool/ /var/www/
```

- Compruebe la correcta publicación escribiendo *http://localhost/monitool/cpu/* en la barra de direcciones del navegador Web, la salida deberá será similar a esta:



Note en las líneas inferiores, después de *Apache/2.x.x (Debian)* que aparezca *PHP/5.x.x*. Esto comprueba que el módulo de PHP ha sido cargado en Apache.

- Asegúrese que todos los archivos binarios tengan permisos de ejecución, para saber cuáles son revise el Anexo E. Si necesita cambiar los permisos, el comando correspondiente será similar a este

```
#chmod 777 monitool/cpu/cpu /monitool/cpu/process …
```

o Archivos de Configuración

Analice su caso y determine los valores de configuración correspondientes. Posteriormente escríbalos en los archivos correspondientes con su editor de texto preferido. En estos archivos no se permite otra cosa distinta al valor correspondiente que es un número, así que no agregue comentarios o texto (excepto en el archivo *sitios.conf* donde si se acepta texto)

o Scripts de Inicio y Paro

Para iniciar la aplicación, es necesario posicionarse primero en el directorio *init* en donde se encuentra el script correspondiente. Una vez en ese directorio ejecute el *script shell*:

```
$ cd monitool/init/
monitool/init$ sh monitool_start.sh
Iniciando monitor de CPU ......
[ OK ]
Iniciando monitor de RAM ......
[ OK ]
Iniciando Monitor de Tasa de Transferencia ......
[ OK ]
Iniciando Monitor de Conexiones ......
[ OK ]
Iniciando Monitor de Latencia ......
[ OK ]
Iniciando Monitor de Paquetes Recibidos y Enviados ......
[ OK ]
Iniciando Monitor de Paquetes Perdidos ......
[ OK ]
```

Como podrá ver, la aplicación devuelve una salida que confirma su correcta ejecución. Tal como se hizo en la etapa de pruebas en el Capítulo 5, puede ejecutar el comando *ps* para darle seguimiento a los procesos que fueron lanzados por el script.

```
$ ps -fl
UID        PID  PPID  C PRI  CMD
usuario   6468  6467  0  75 -bash
usuario   7493     1  0  99 sh cpu.sh
usuario   7494     1  0  99 sh ram.sh
usuario   7495     1  0  99 sh rate.sh
usuario   7496     1  0  99 sh conexion.sh
usuario   7497     1  0  99 sh latencia.sh
usuario   7502     1  0  99 sh paq_rcbd_env.sh
usuario   7503     1  0  99 sh paq_perdidos.sh
usuario   8015  7494  0  99 ./retardo
usuario   8022  7503  0  99 ./retardo
usuario   8027  7493  0  99 ./retardo
usuario   8035  7502  0  99 ./retardo
usuario   8039  7495  0  94 ifstat -T -z
usuario   8040  7495  0  94 cat
usuario   8043  7495  0  99 ./retardo
usuario   8053  7496  0  99 ./retardo
usuario   8068  7497  0  99 ./retardo
usuario   8074  6468  0  77 ps -fl
```

Note que este script ejecuta otros scripts y estos mantienen un PID mientras estén en ejecución (dado que son un ciclo infinito), si usted deseara ejecutar solo uno de ellos, puede hacerlo desde la ubicación correspondiente de acuerdo a la métrica deseada. En el caso de los retardos, estos finalizan y son lanzados nuevamente, así que su PID cambia constantemente. También existen un par de procesos adicionales en la lista, estos son utilizados para el monitoreo de la tasa de transferencia.

Si usted deseara terminar la ejecución de la aplicación podría hacerlo enviando una señal de finalización a todos los procesos listados, sin embargo existe un script que los termina "amablemente" a todos ellos. Para ejecutarlo, es necesario ubicarse nuevamente en el directorio *init*:

```
$ cd monitool/init/
monitool/init$ sh monitool_stop.sh
Finalizando Monitor CPU ...
[ OK ]
Finalizando Monitor RAM ...
[ OK ]
Finalizando Monitor de Tasa de Transferencia ......
[ OK ]
Finalizando Monitor de Conexiones ......
[ OK ]
Finalizando Monitor de Latencia ......
[ OK ]
Finalizando Monitor de Paquetes Recibidos y Enviados ......
[ OK ]
Finalizando Monitor de Paquetes Perdidos ......
[ OK ]
```

El script de paro devuelve un estado OK después de finalizar los procesos correspondientes a cada una de las métricas. Puede ejecutar *ps* para comprobar que efectivamente hayan sido finalizados.

Tanto el inicio o la detención de la aplicación son realizadas desde la consola, no es posible hacerlo vía Web debido a que esto comprometería la seguridad en el servidor.

- **Otros scripts**

Como ya se mencionó, existen otros scripts que realizan el monitoreo de una métrica únicamente. De esta forma, existe un script por cada métrica, esto dada la

característica modular de la aplicación. Es posible entonces iniciar uno de estos scripts desde la ubicación correspondiente de acuerdo a la métrica deseada, por ejemplo, el script de CPU se encontrará en el directorio *monitool/cpu*.

Además de estos existe otro par de scripts.

El primero de ellos llamado *creacion_regs.sh* se encarga de crear los registros temporales de cada una de las métricas, de acuerdo al valor definido en el archivo de configuración *num_datos.conf*. Este script se ejecuta una sola vez cuando se inicia la aplicación.

El segundo de ellos se explicará en el siguiente punto.

- **Compilación de código fuente**

La aplicación posee un directorio que contiene los archivos con el código fuente de los programas realizados en C para el monitoreo. Si el usuario hace alguna modificación al código fuente, debería compilarlo y ubicar el binario en la ruta donde la aplicación lo buscará. Si hace modificaciones a varios archivos deberá repetir esta operación.

De esta forma, existe un script que se encarga de compilar todos los archivos de código fuente y ubica los binarios en el directorio correspondiente.

Además de facilitar este procedimiento, este script tiene la intención de que el usuario pueda enviarle parámetros al compilador de acuerdo a la arquitectura del servidor o a las necesidades particulares.

Este script se encuentra en monitool/src/make.sh, en el mismo lugar donde se encuentran los archivos de código fuente. Antes de ejecutarlo, la aplicación no deberá estar en ejecución.

```
$ cd monitool/src/
monitool/src $ sh make.sh
```

Este script no retorna mensaje alguno en caso de que todo salga bien, cualquier mensaje desplegado será de parte del compilador, ya sea una advertencia o error.

- **Archivos de Registro Temporal**

Cuando la aplicación es ejecutada, el script *creacion_regs.sh* crea los registros temporales que contienen los datos que serán graficados. Estos registros pueden

ser consultados desde la Terminal y observar como los datos nuevos desplazan los anteriores.

Los registros temporales son archivos ocultos (su nombre comienza con '.') y se encuentran en el directorio correspondiente a la métrica que se refieren. El nombre de los archivos tiene el prefijo "*.datos_*" seguido de una cadena descriptora del dato que contengan. Por ejemplo el registro temporal de cpu es "*.datos_cpu*" o el de la latencia promedio "*.datos_avg*".

Puede entonces ser desplegado el contenido, o bien contar las líneas para comprobar que la cantidad de datos en ellos es la debida de acuerdo al archivo de configuración *num_datos.conf*.

```
$ tail -10 monitool/cpu/.datos_cpu
49.600002
49.600002
49.500004
49.600002
49.500000
49.500000
49.500004
49.600002
49.300007
49.300007
$ tail -10 monitool/latencia/.datos_avg
0.000000
664.000000
669.000000
464.000000
510.000000
591.000000
649.000000
832.000000
549.000000
636.000000
$ cat  monitool/rate/.datos_in | wc -l
50
```

- **Configuración dinámica**

La aplicación permite que los archivos de configuración sean modificados en "tiempo de ejecución", es decir mientras esta se encuentra en ejecución y no requieren que sea reiniciada. Sin embargo, se recomienda que los valores sean

escritos mediante el comando *echo*. Por ejemplo, supongamos que se quiere cambiar el retardo a 60 segundos, se tiene entonces que ejecutar lo siguiente:

```
$ echo 60 > monitool/config/retardo.conf
```

Supongamos que quiere reemplazar los sitios de prueba para el cálculo de la latencia por *google.com* y *www.ipn.mx*, se tiene que ejecutar lo siguiente:

```
$ echo google.com > monitool/config/sitios.conf
$ echo www.ipn.mx >> monitool/config/sitios.conf
```

El primer comando reemplaza el contenido del archivo, y el segundo lo agrega al final.

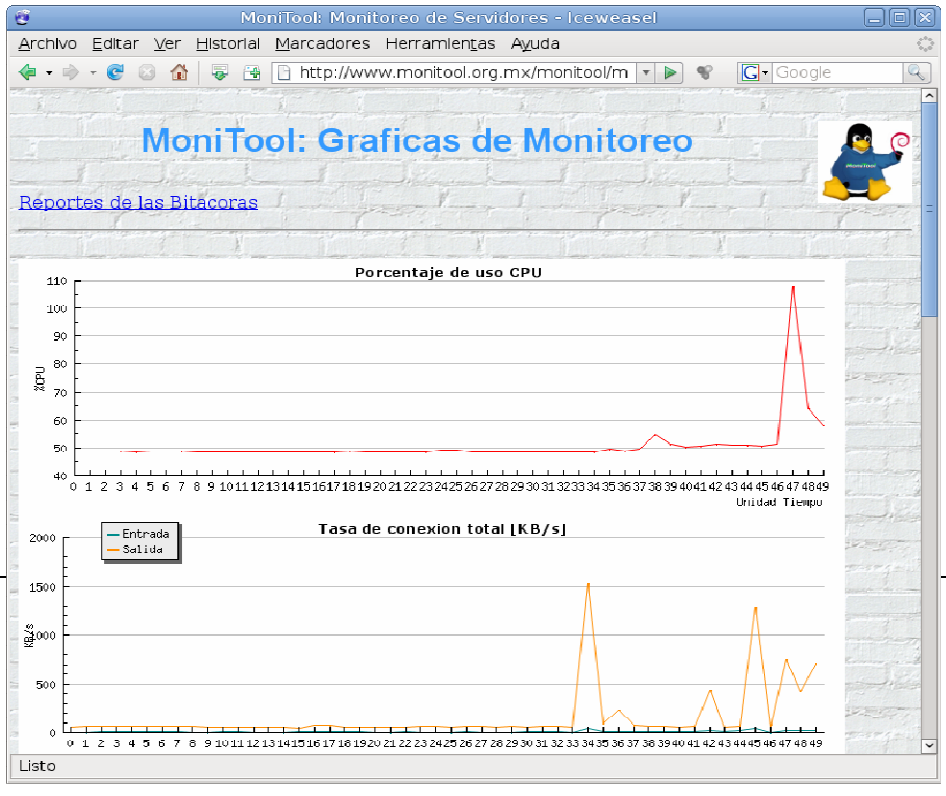
Es posible utilizar un editor de textos como *vi* o *emacs*, sin embargo este método es el más recomendado para un ajuste rápido.

Nota: A pesar de que puede ser modificado sin problemas, el cambio realizado en el archivo *num_datos.conf* no será realizado hasta que la aplicación sea reiniciada.

- **Acceso Web**

El acceso a través de un navegador Web puede ser realizado desde el servidor, desde una computadora en la red o incluso desde Internet.

Para poder acceder a las gráficas deberá escribir en la barra de direcciones, el nombre de host del servidor (si existe una entrada en los DNS) o la dirección IP seguido de *monitool*. Por ejemplo: *http://www.monitool.org.mx/monitool/*. Recuerde que el directorio *monitool* fue agregado en la ubicación de publicación por defecto de Apache.



El enlace Reportes de las Bitácoras en la parte superior permitirá acceder al menú de selección de métricas del visualizador de bitácoras.
Una vez seleccionada una métrica y que los datos contenidos en las bitácoras, existen cuatro enlaces que permiten cambiar navegar entre las pantallas.

En todas las pantallas del visualizador de bitácoras, el enlace Pantalla Principal MoniTool permite regresar a la pantalla principal de la aplicación.

El enlace Menu Despliegue de Bitacoras permite retornar al menú que permite seleccionar la métrica de la cuál se desplegarán las bitácoras.
Los enlaces Ver Bitacoras de CPU Anteriores desplegará los datos contenidos en el archivo de nivel de antigüedad inferior y el enlace Ver Bitacoras de CPU Siguientes muestra los datos del archivo de nivel de antigüedad superior.

- **Inicio Automático**

Con la finalidad de evitar que el usuario tenga que iniciar la ejecución de la aplicación manualmente cada vez que el servidor inicia, es posible crear un script en el directorio de arranque automático del sistema. Este script permitirá que la aplicación sea ejecutada cuando el sistema operativo inicie todos los demonios y servicios que se tengan, de manera que el monitoreo ocurra siempre que se encienda el equipo.
Para mayor información, consulte la documentación del sistema operativo para la implementación de esta funcionalidad.

- **Cambiar el fondo predeterminado**

La aplicación tiene el directorio *monitool/img/* que contiene todas las imágenes que la interfaz Web despliega. La aplicación establece por defecto lee el archivo *fondo.png*, si se desea cambiar el fondo bastará con reemplazar este archivo en este

directorio. El nombre, extensión y formato deberán coincidir para que pueda ser leído.

- **Ajustar valores visuales**

En el script principal *monitool.php* existe código HTML que determina las propiedades visuales, tales como tipo, tamaño y colores de las fuentes. Todos estos valores pueden ser ajustados de acuerdo a las preferencias personales, editando directamente el código HTML que las establece.

ANEXO C

Diagramas de Flujo / Diagramas de Flujo de Datos

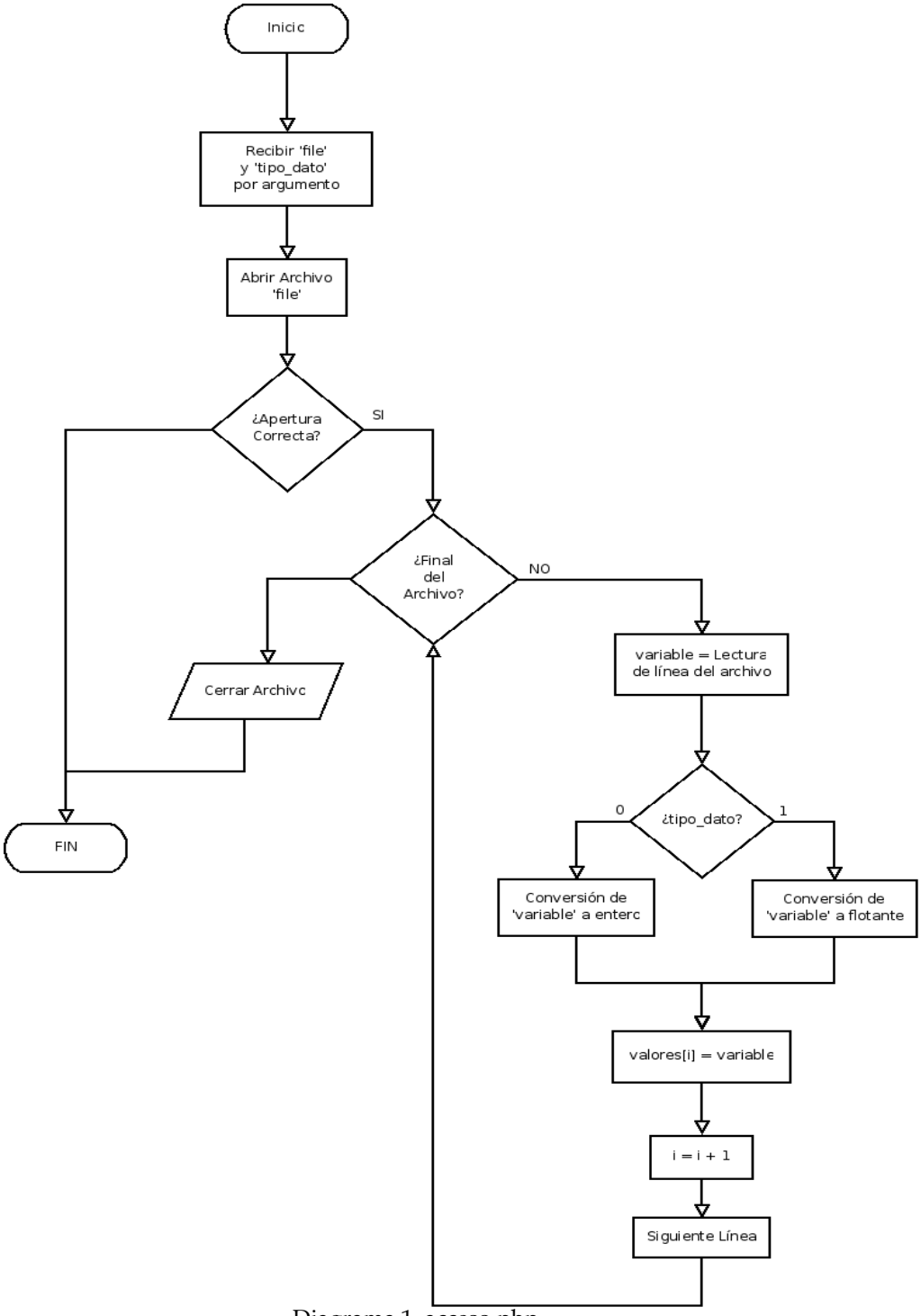


Diagrama 1. acceso.php

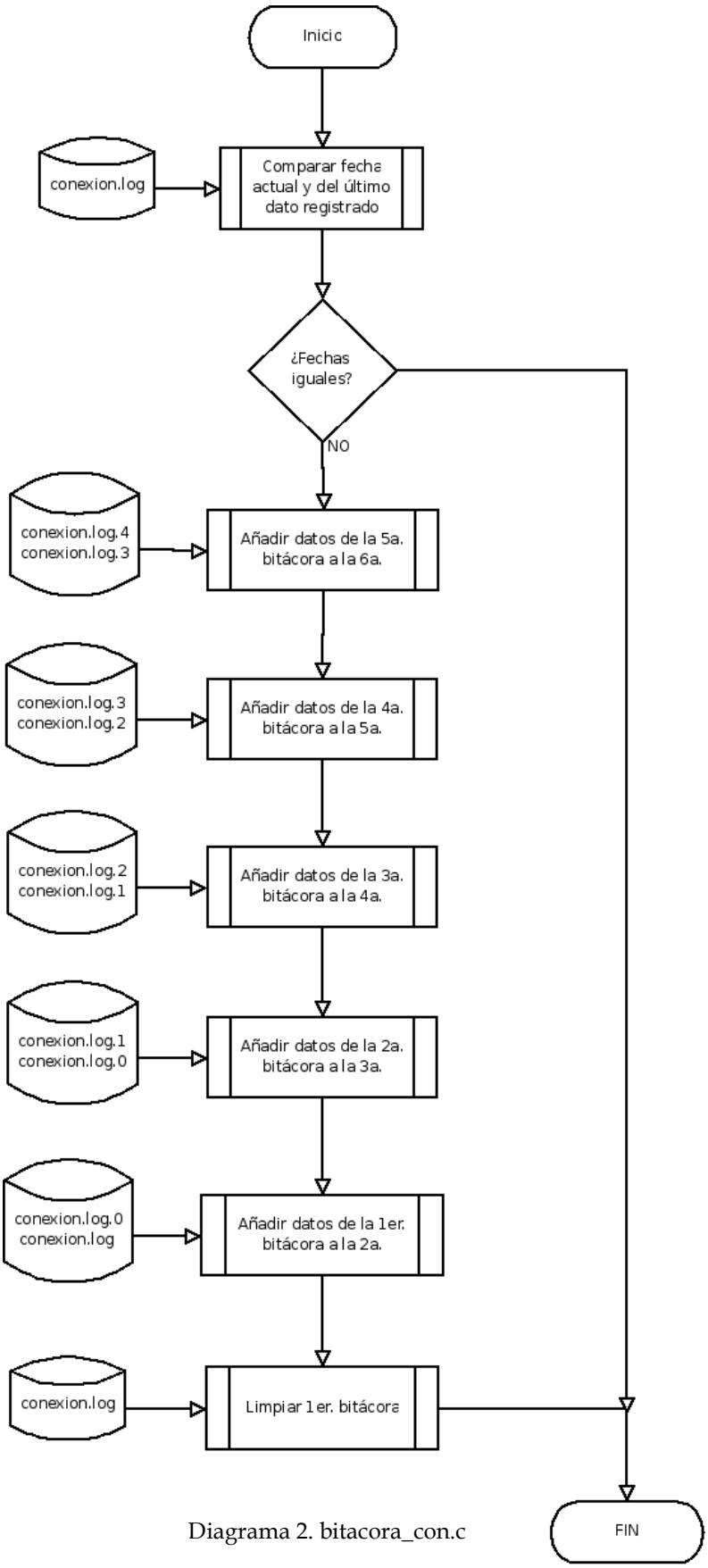


Diagrama 2. bitacora_con.c

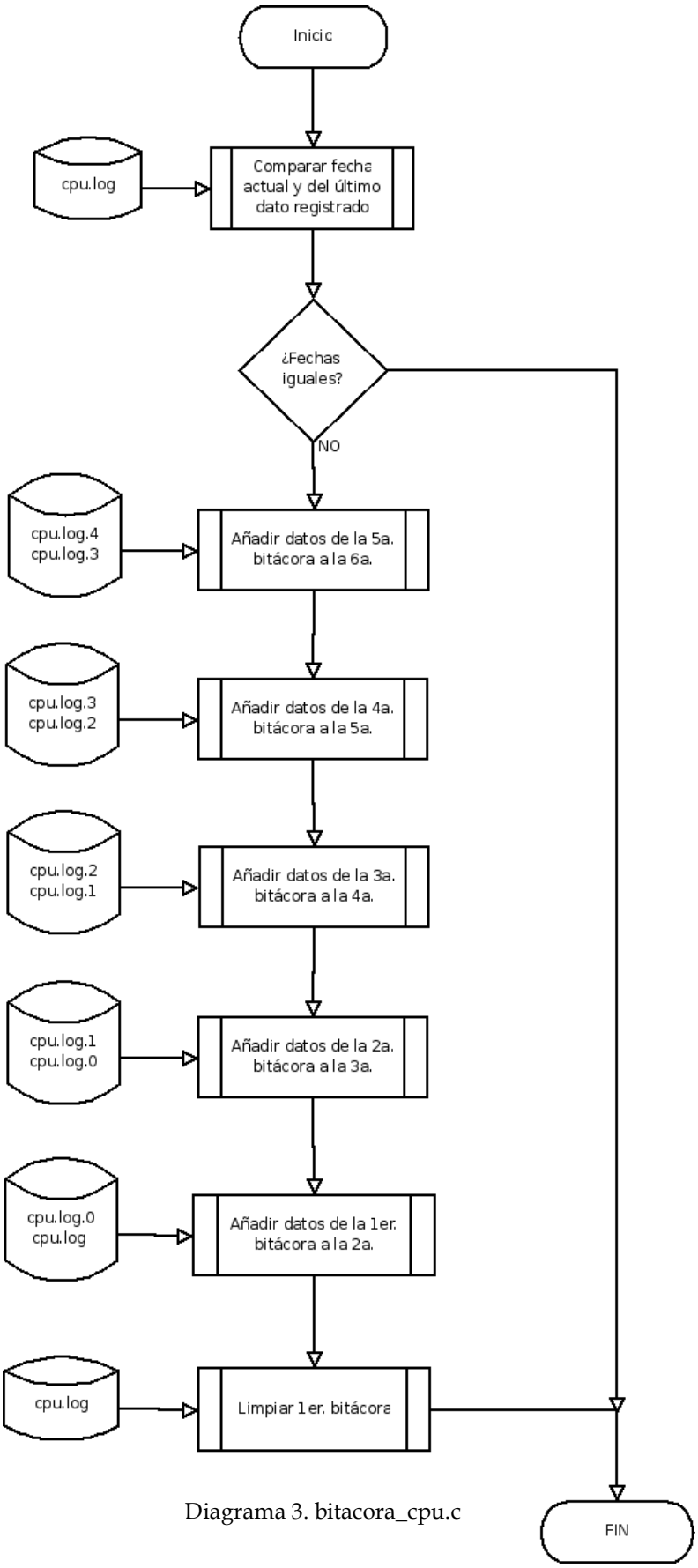


Diagrama 3. bitacora_cpu.c

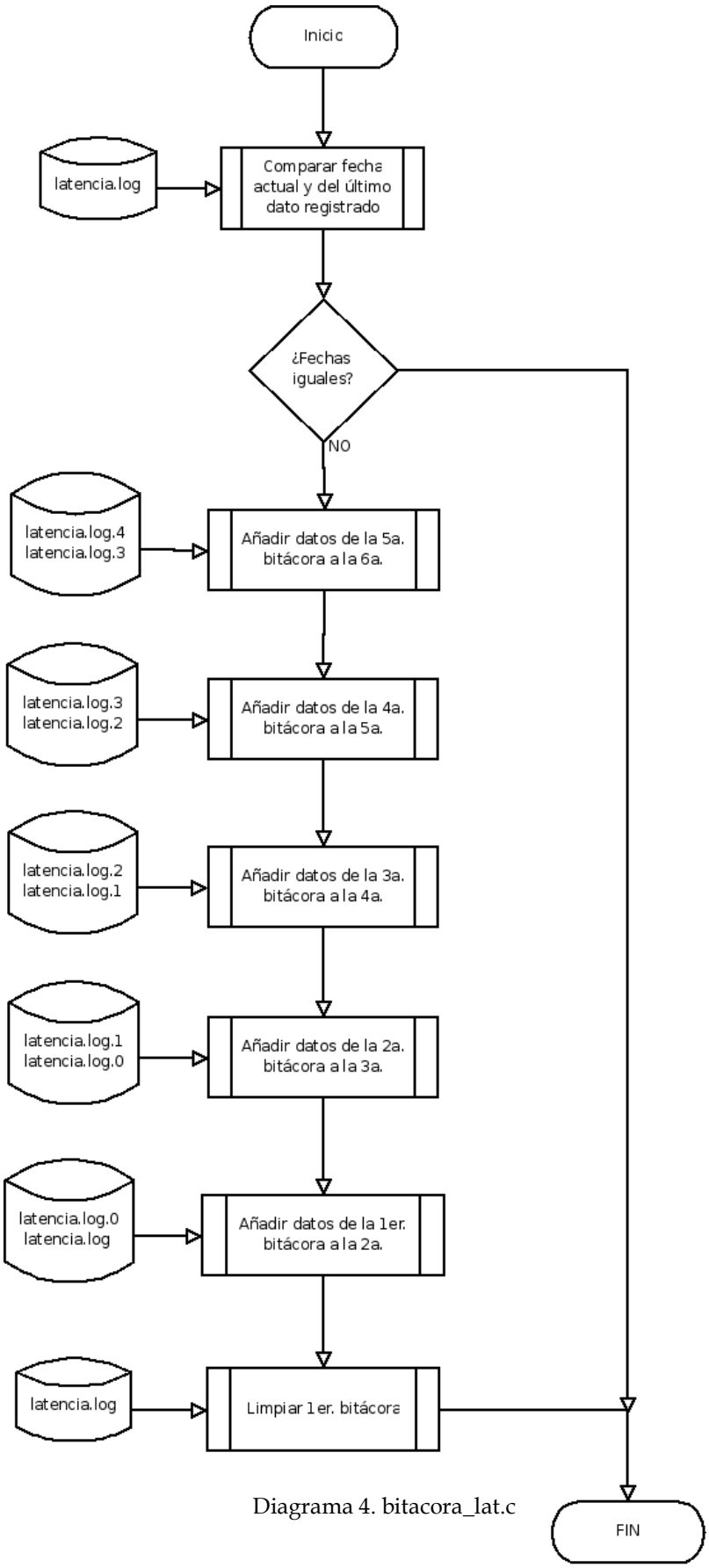


Diagrama 4. bitacora_lat.c

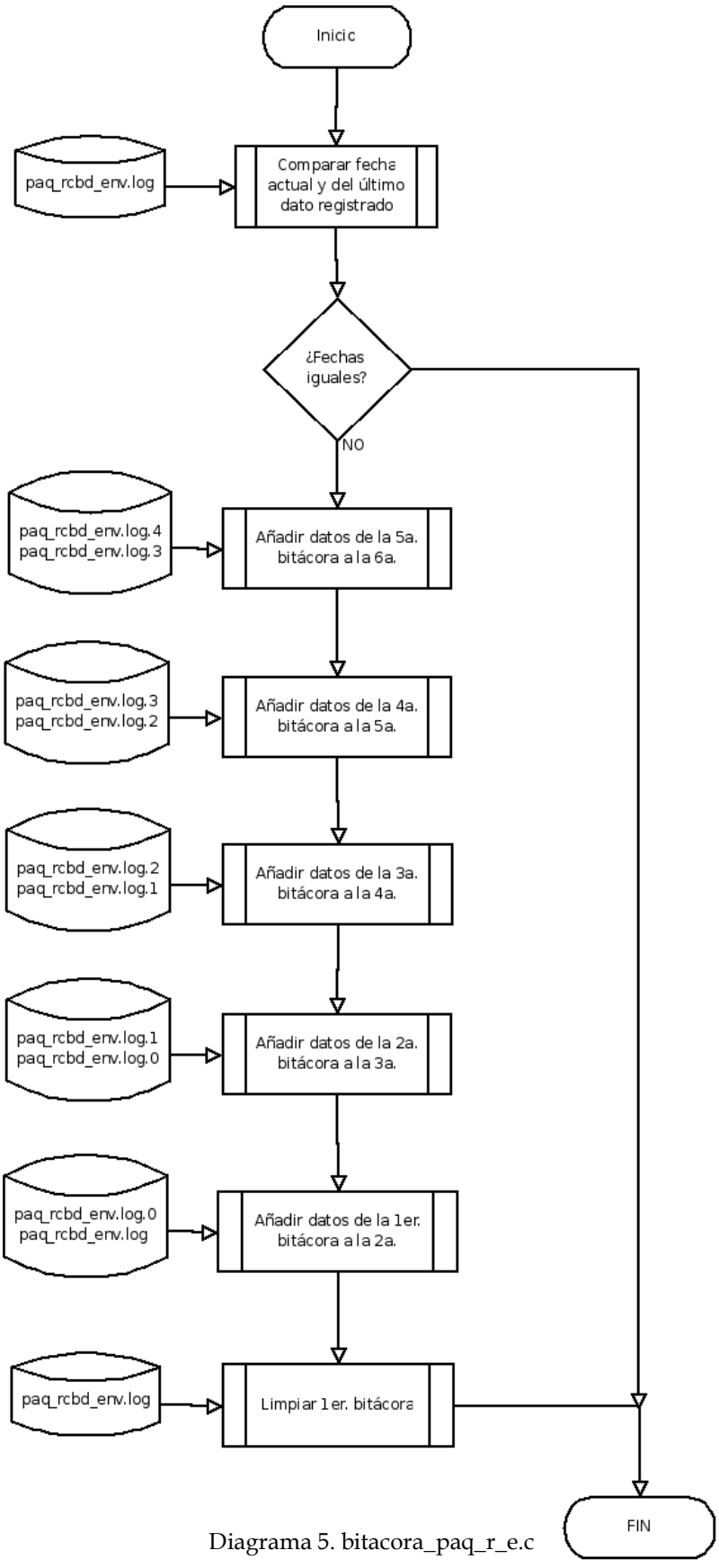


Diagrama 5. bitacora_paq_r_e.c

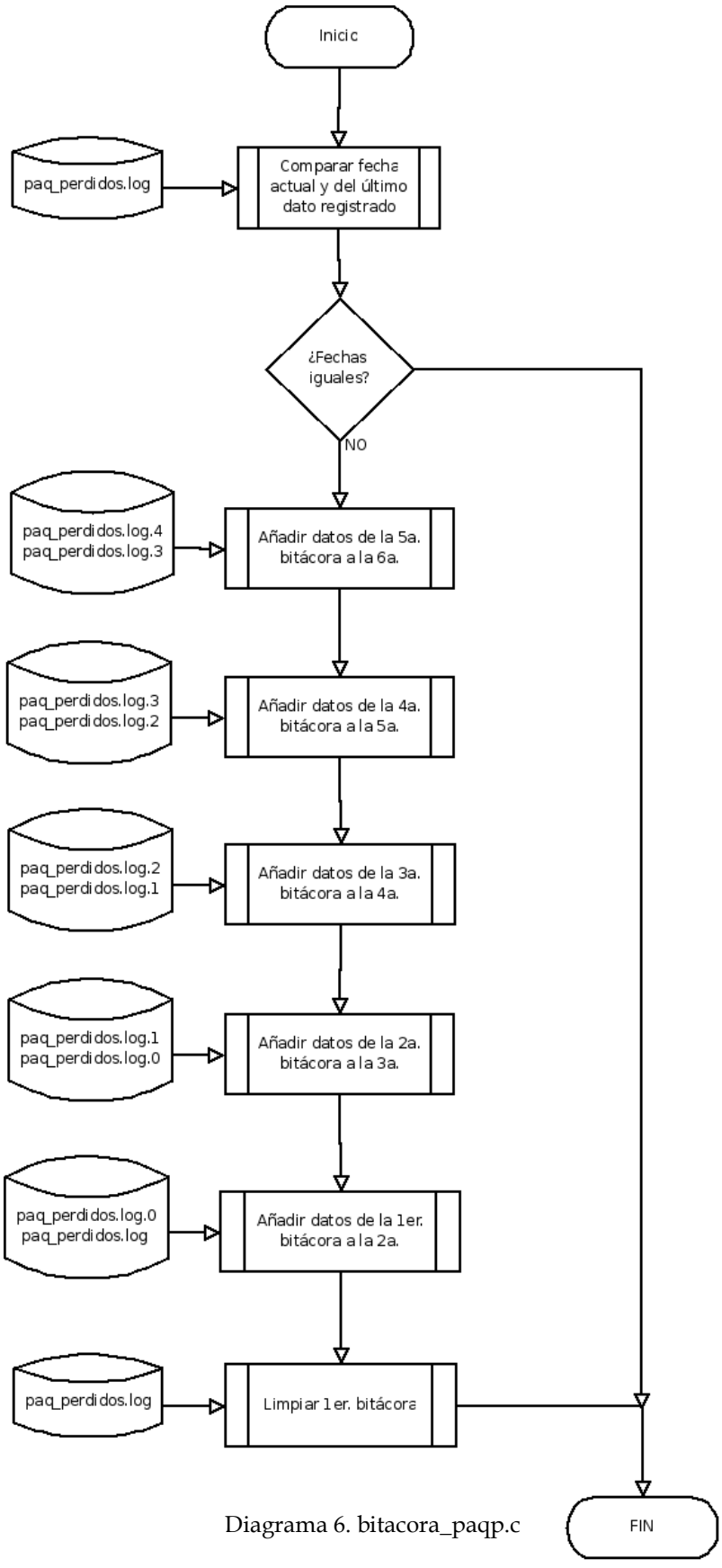


Diagrama 6. bitacora_paqp.c

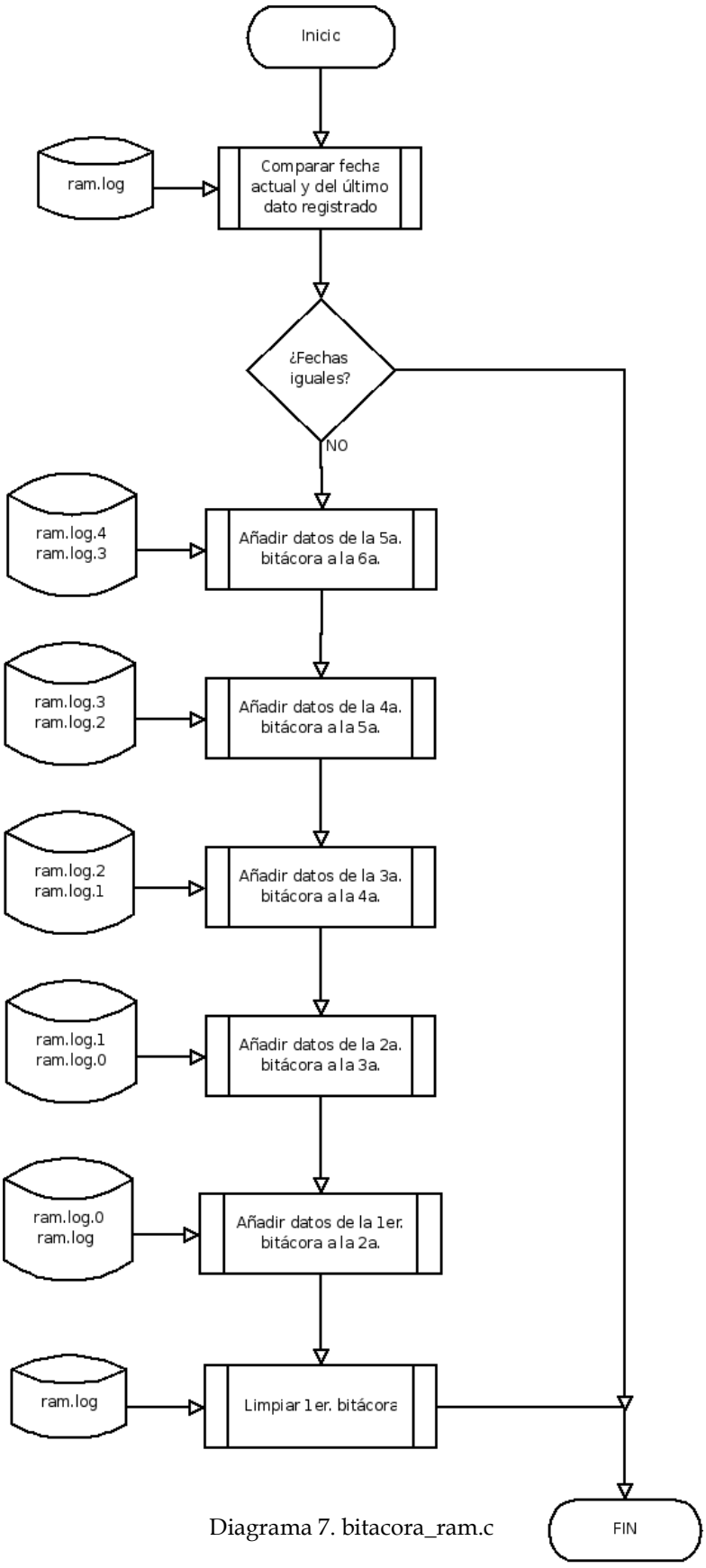


Diagrama 7. bitacora_ram.c

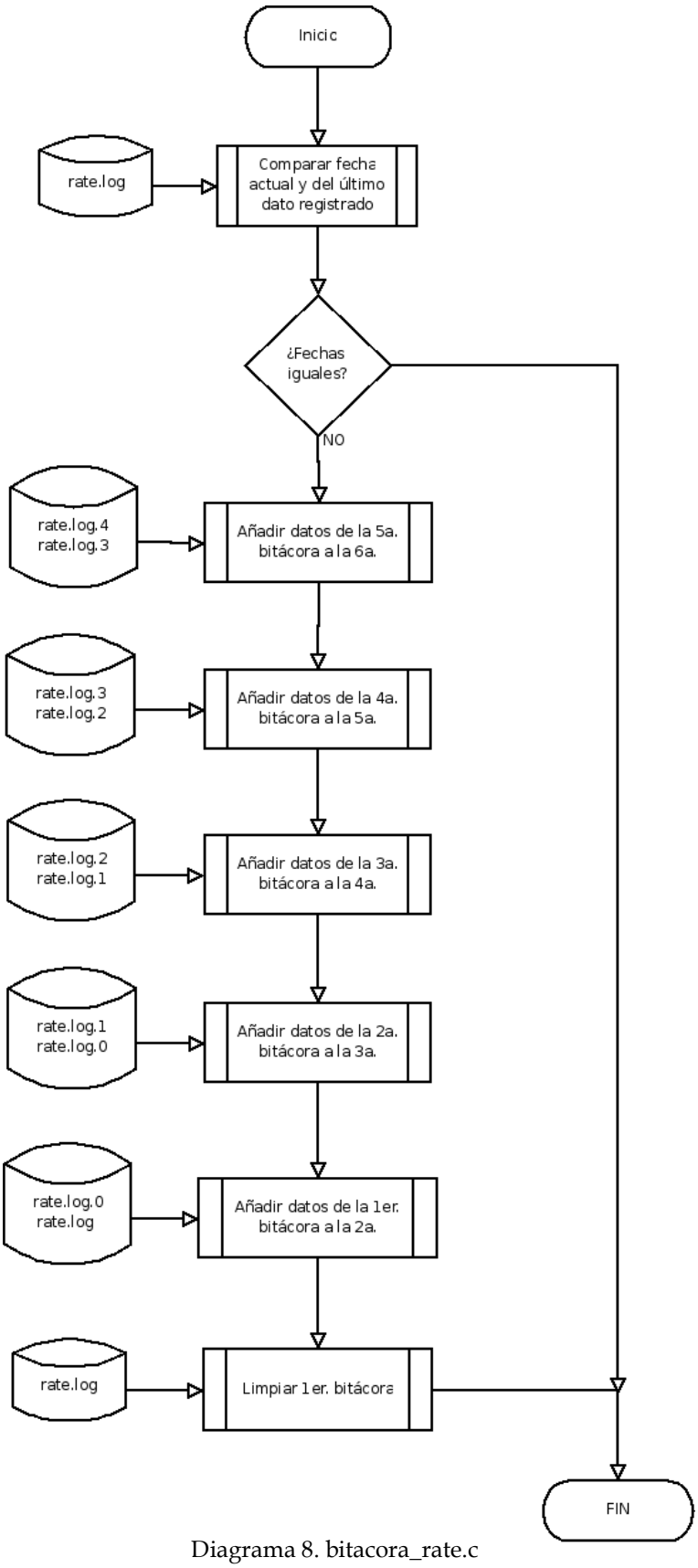


Diagrama 8. bitacora_rate.c

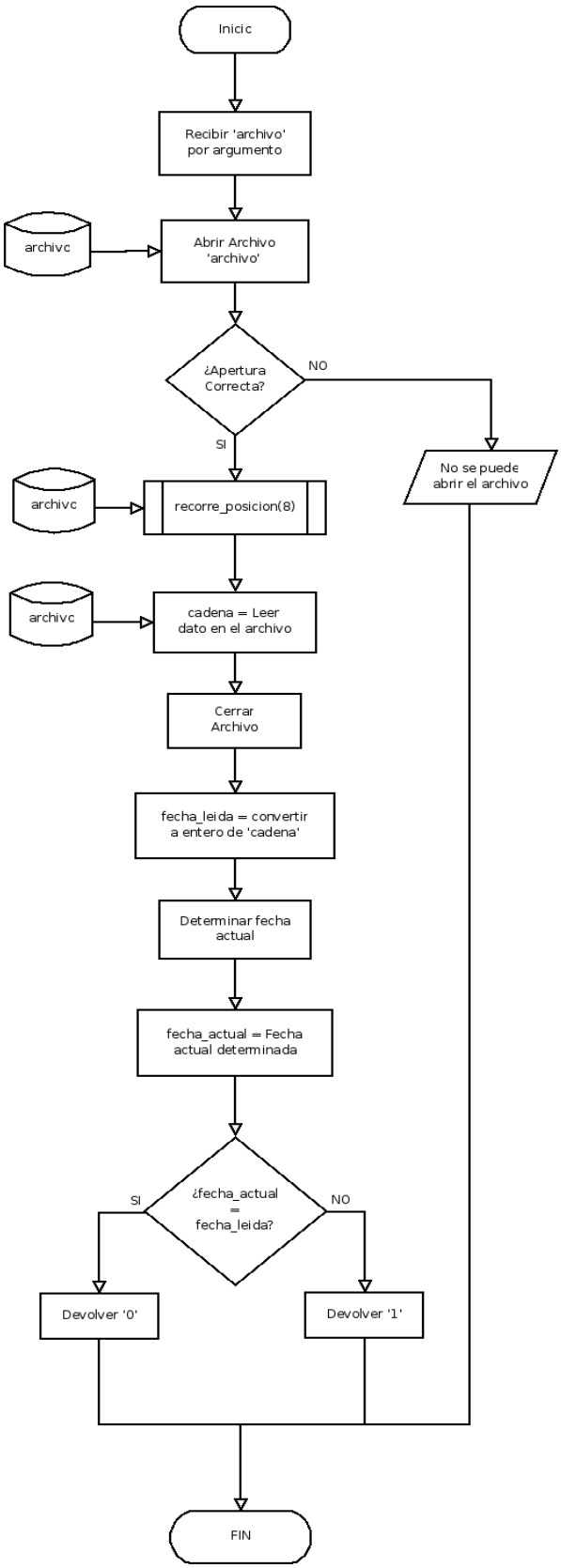


Diagrama 9. comparar_fechas.c

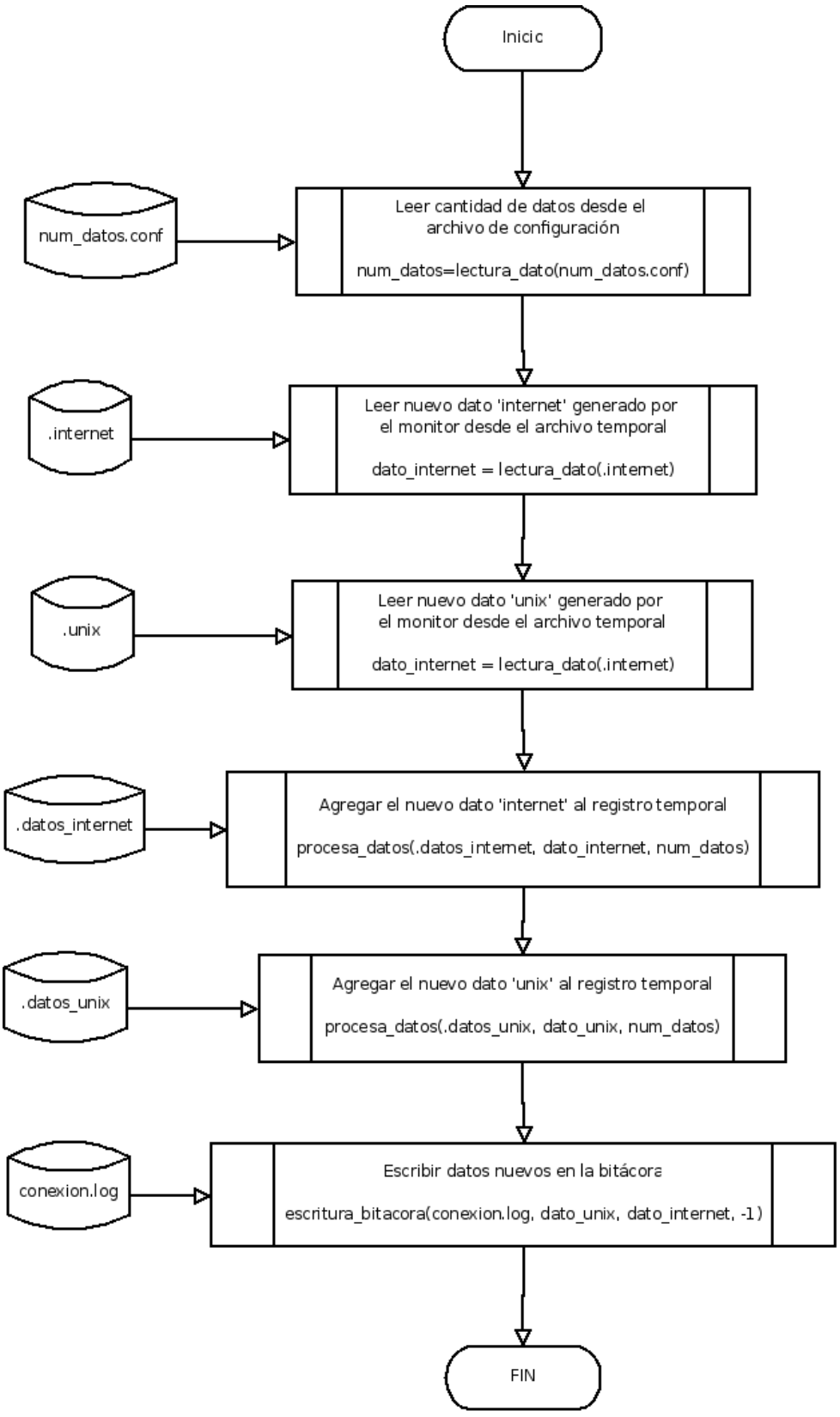


Diagrama 10. conexion.c

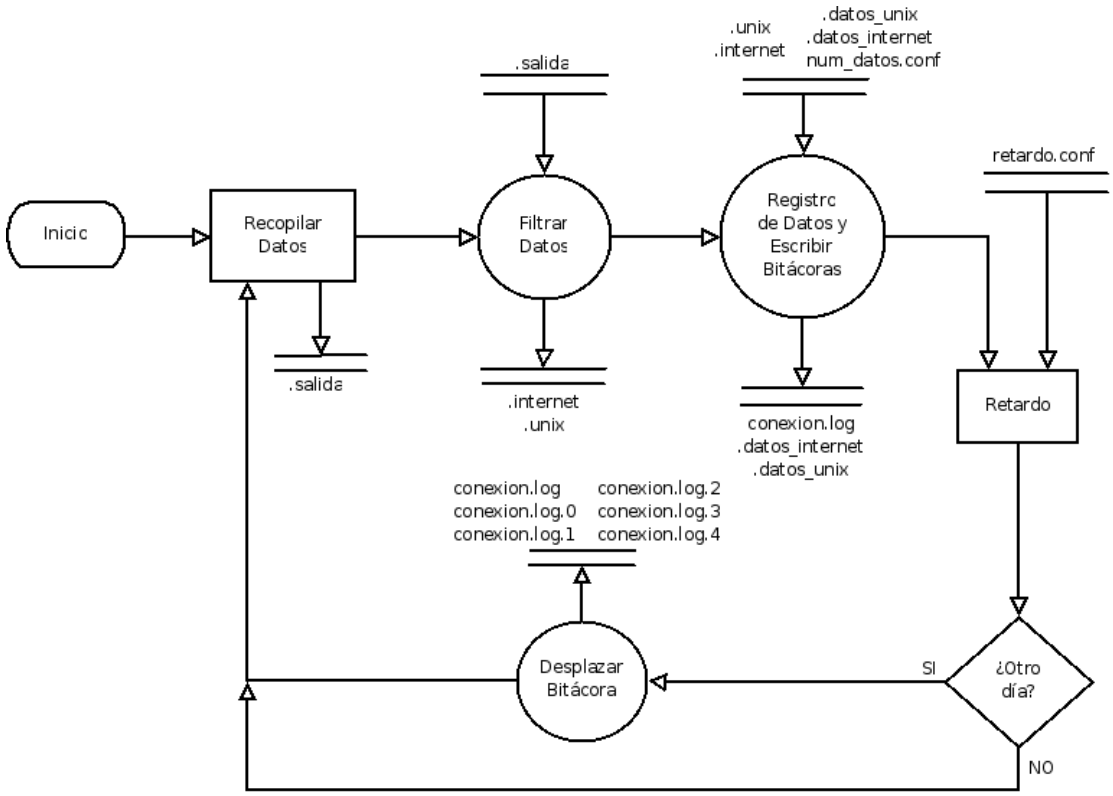


Diagrama 11. conexion.sh

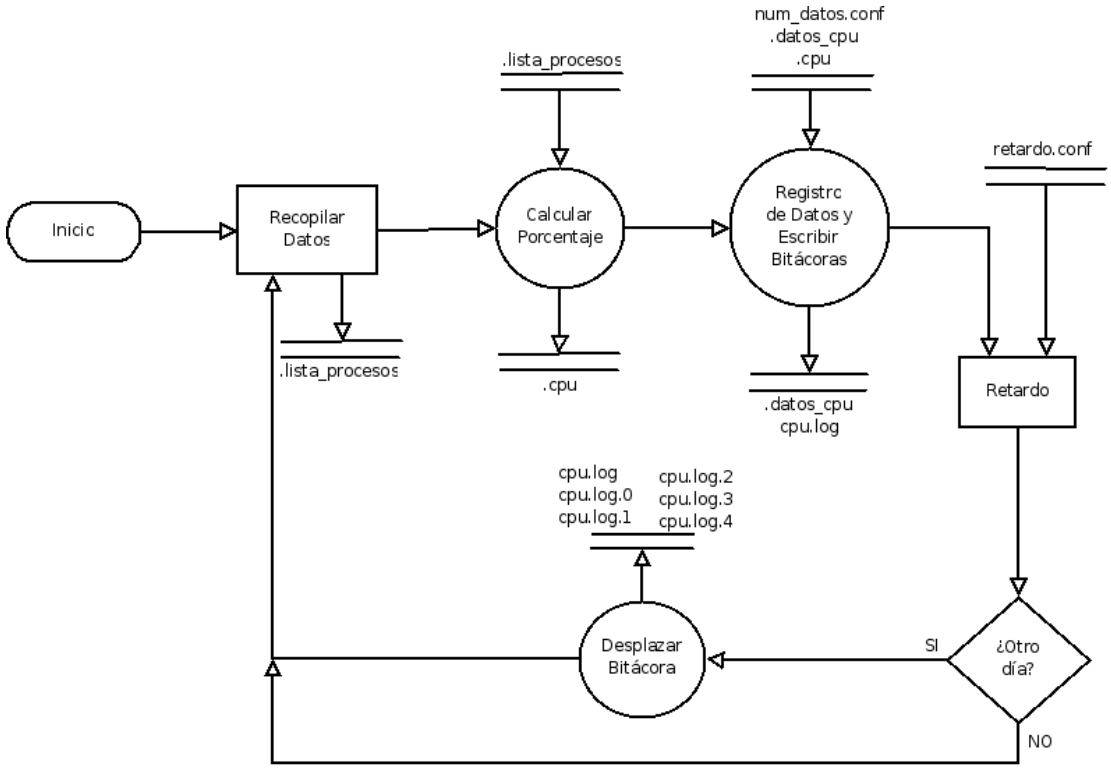


Diagrama 12. cpu.sh

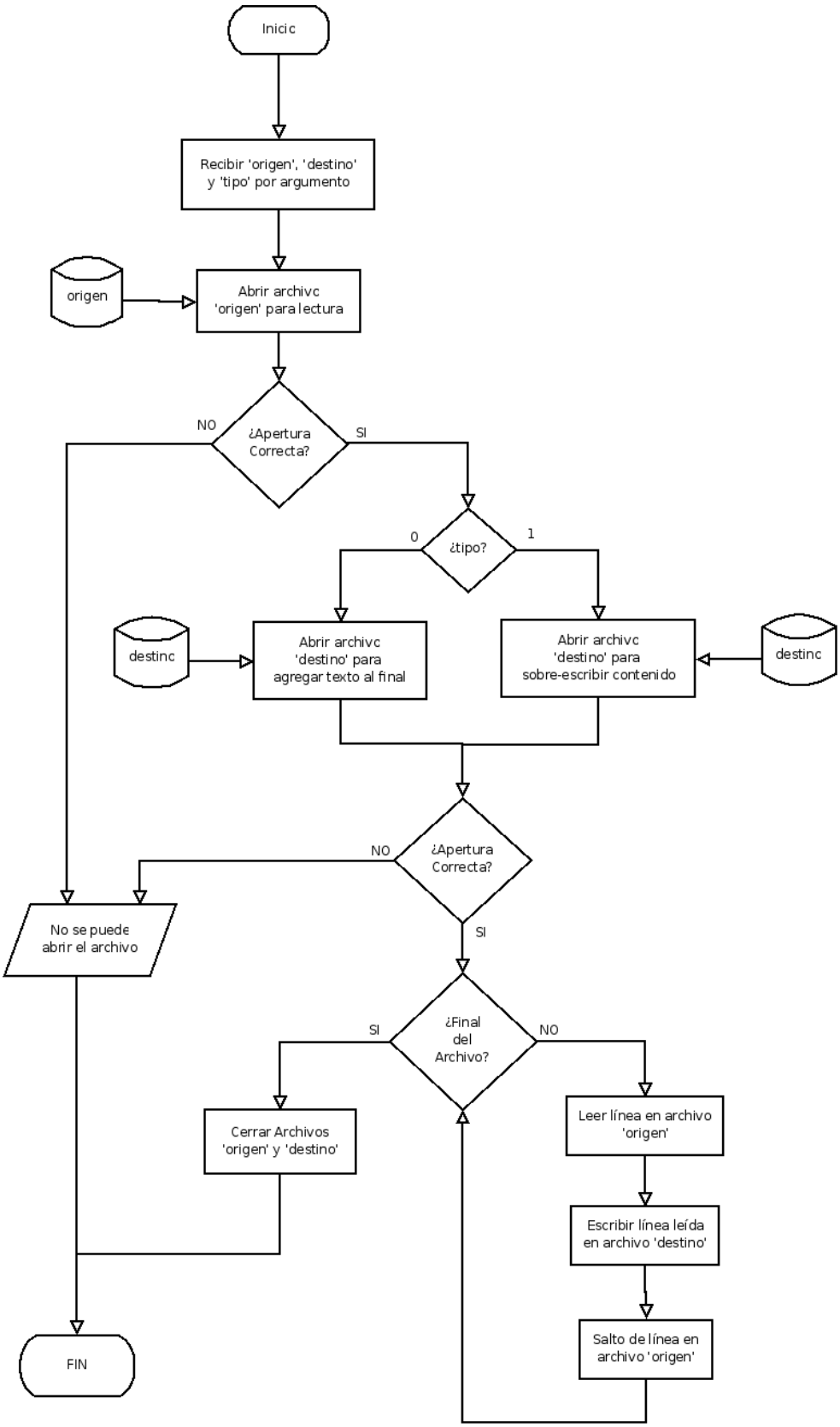


Diagrama 13. copia_archivo.c

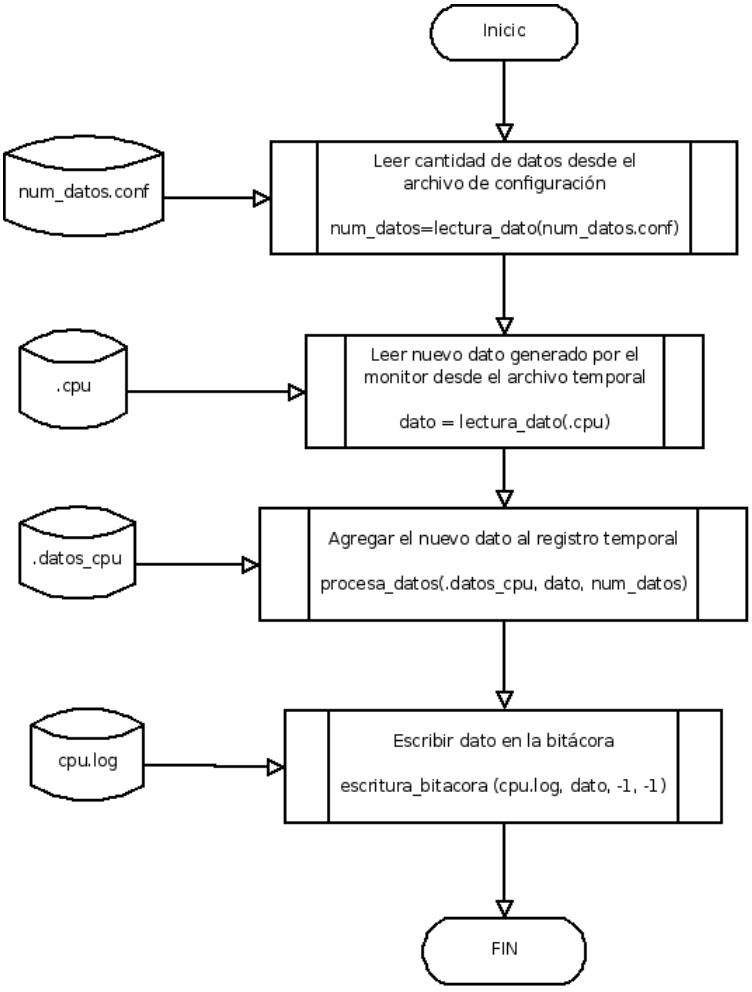


Diagrama 14. cpu.c

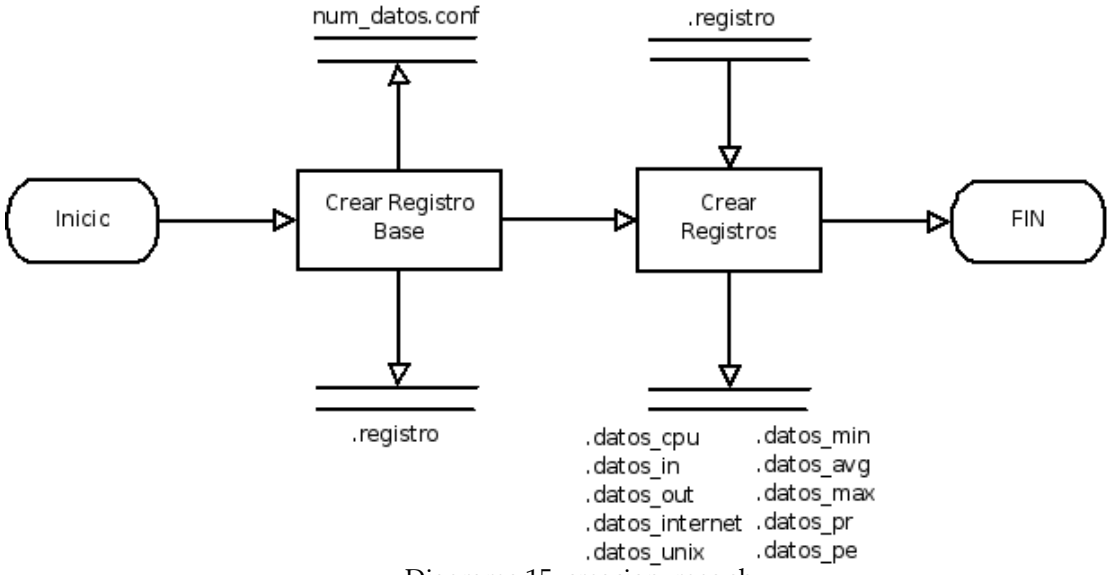


Diagrama 15. creacion_regs.sh

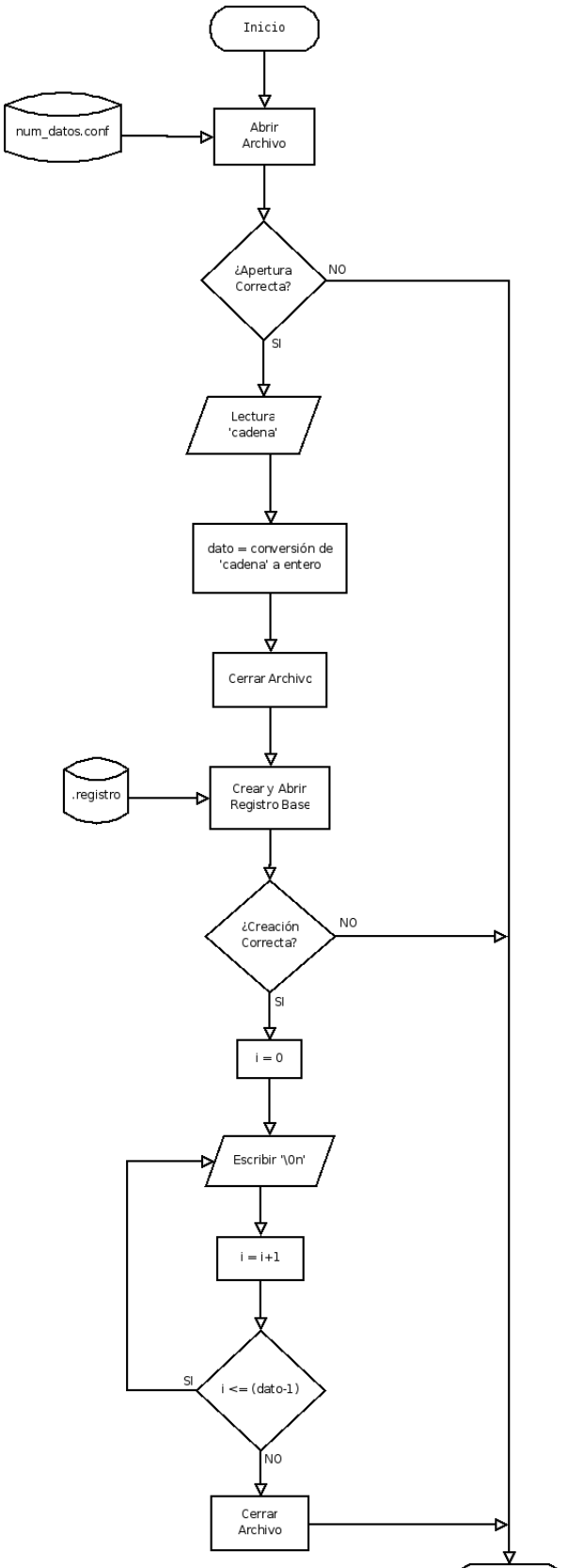


Diagrama 16. creacion_regs.c

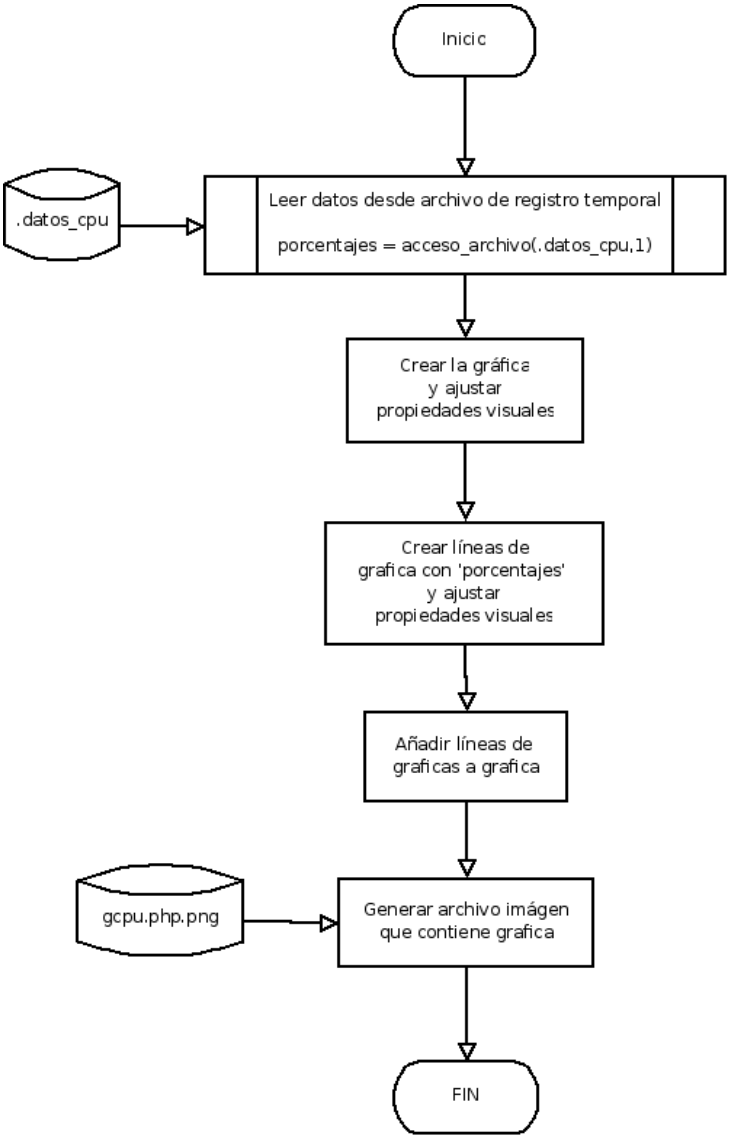


Diagrama 17. gcpu.php

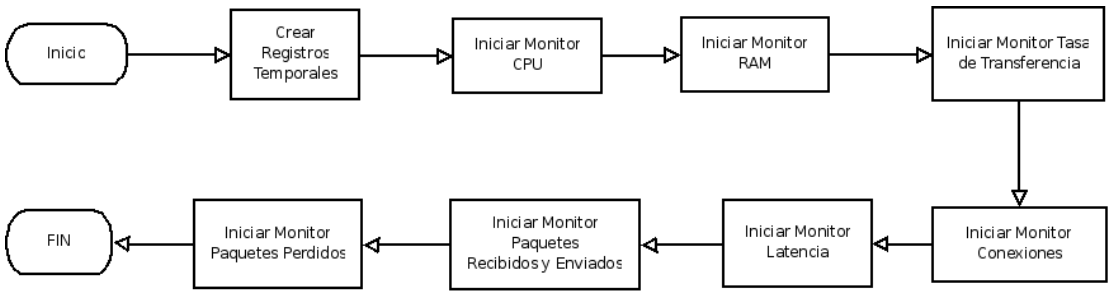


Diagrama 18. monitool_start.sh

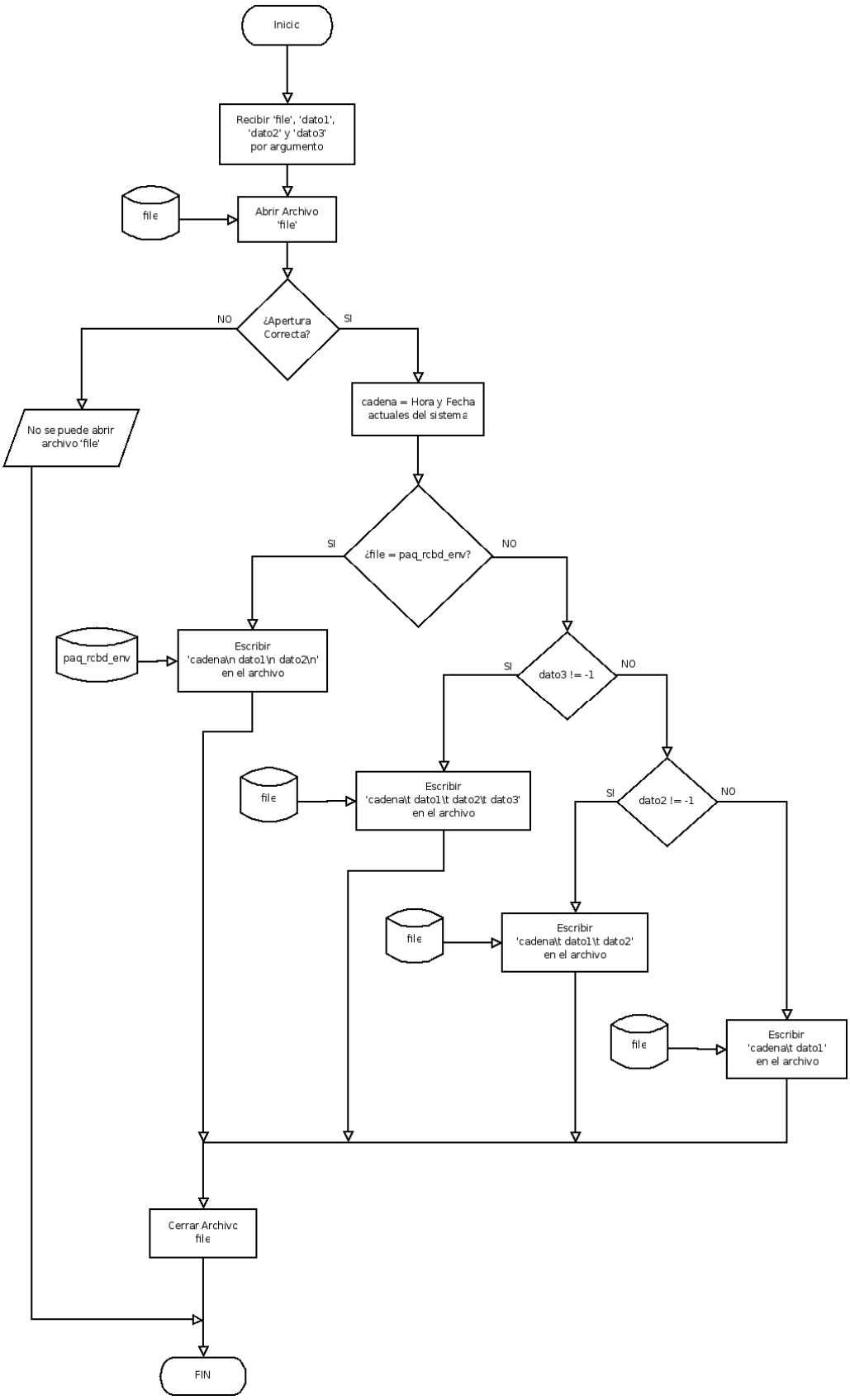


Diagrama 19. escritura_bitacora.c

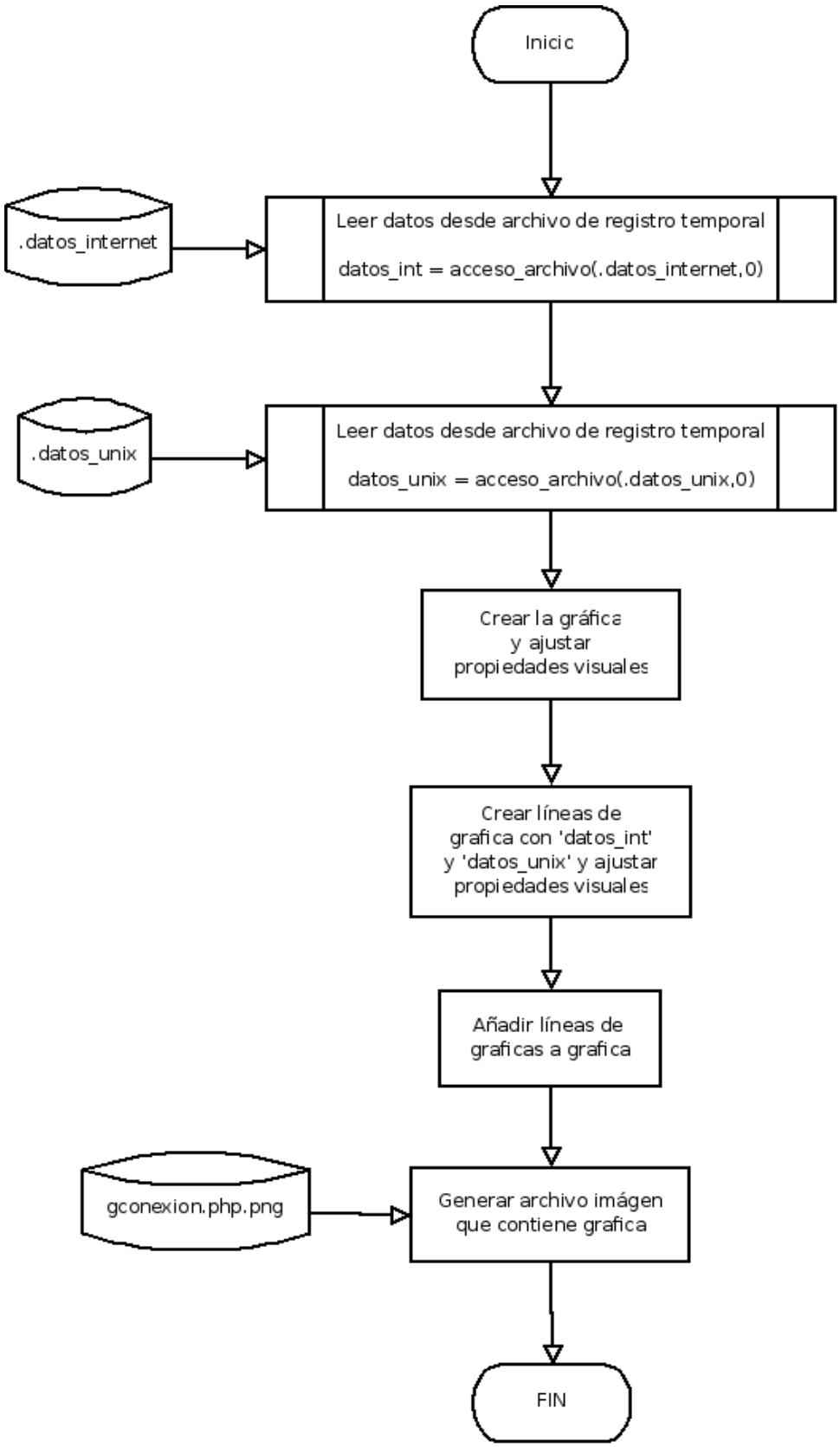


Diagrama 20. gconexion.php

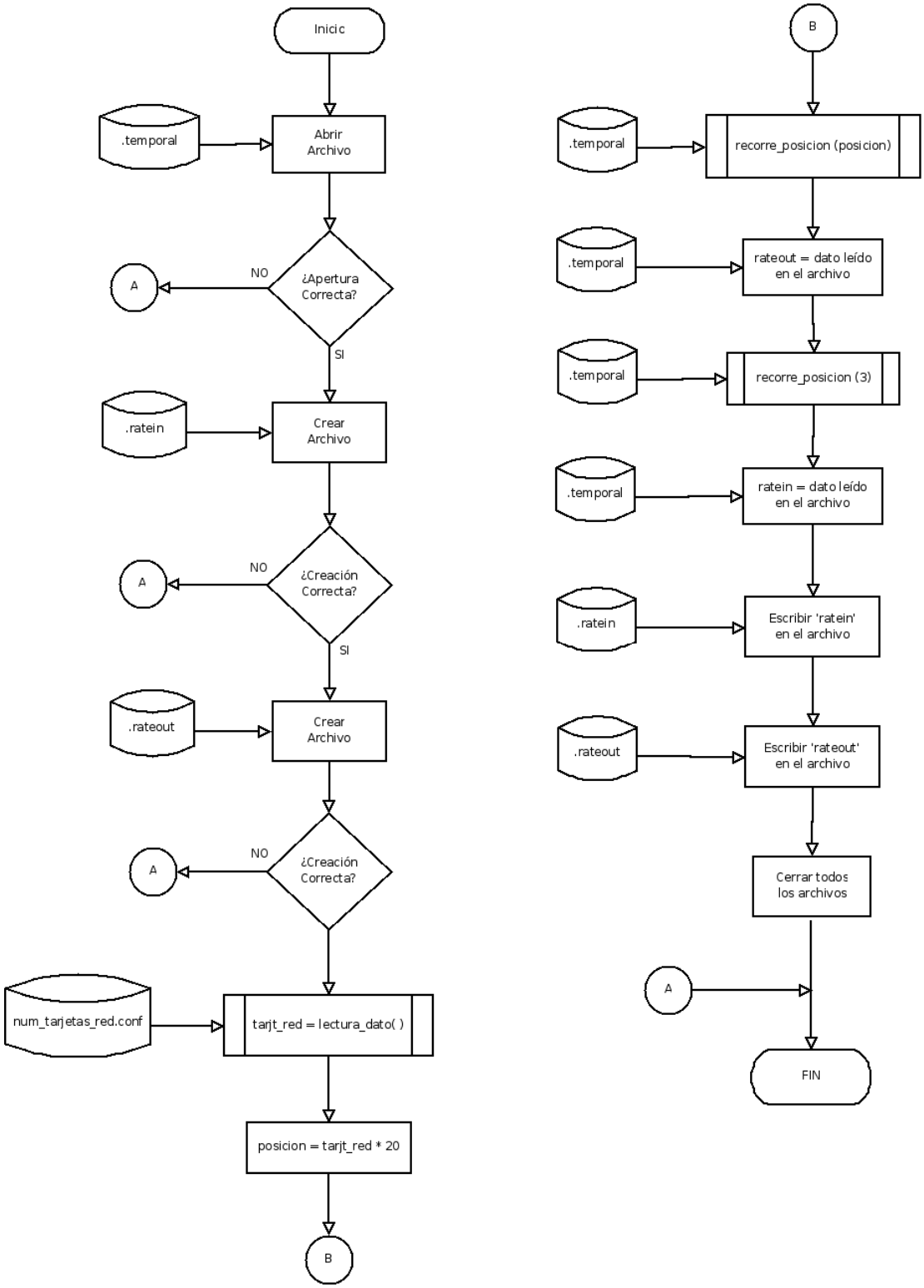


Diagrama 21. get_rate.c

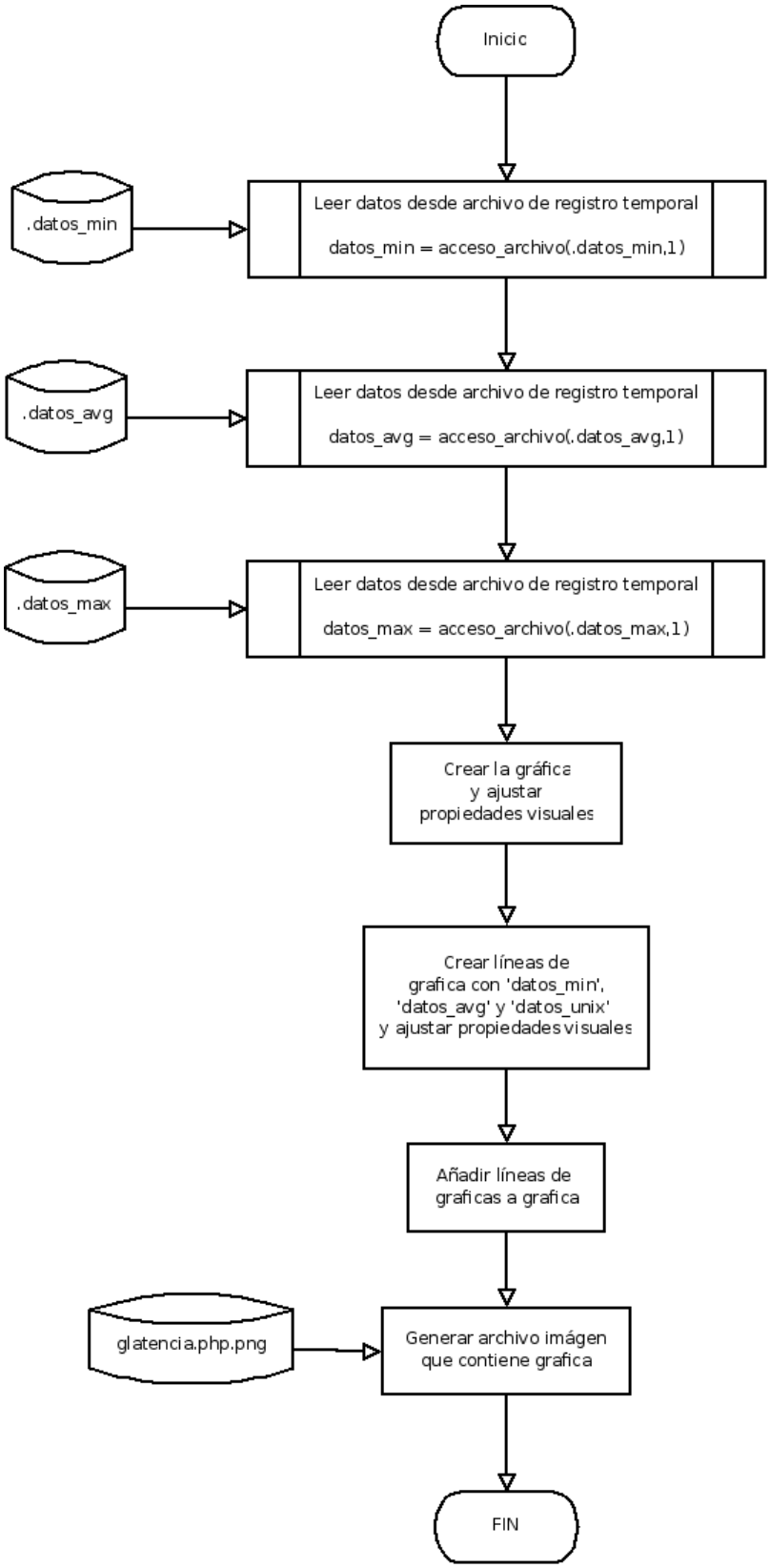


Diagrama 22. glatencia.php

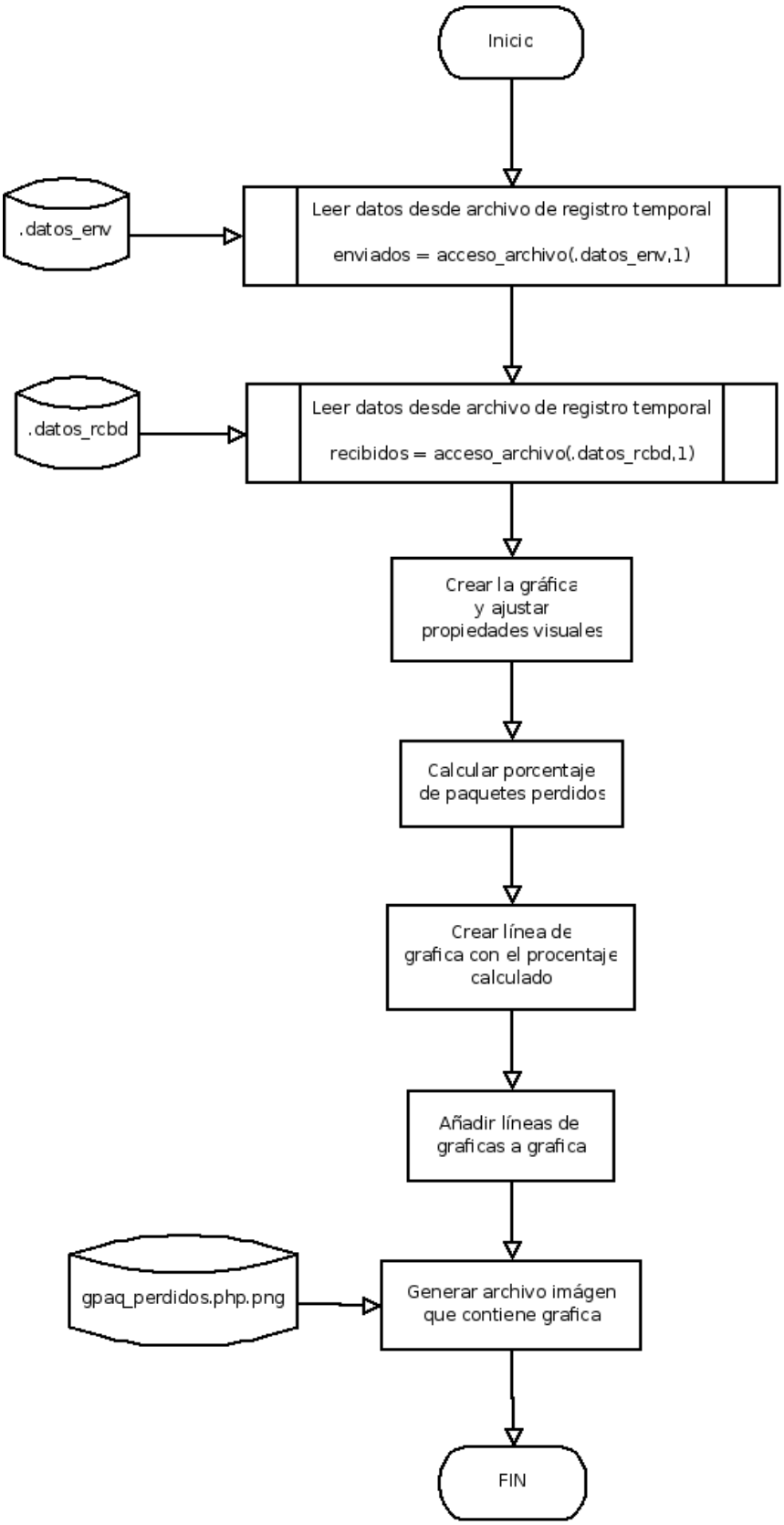


Diagrama 23. gpaq_perdidos.php

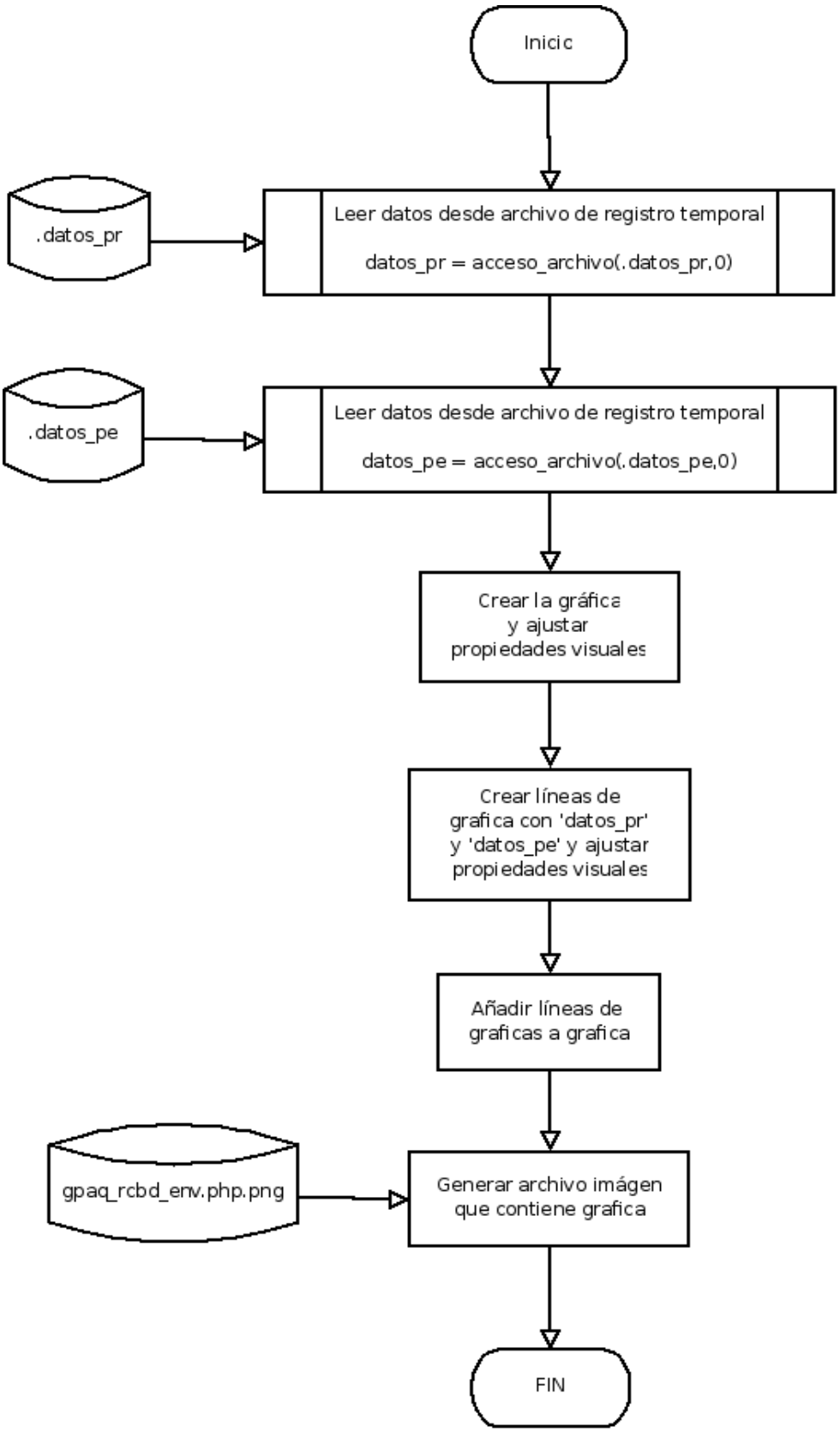


Diagrama 24. gpaq_rcbd_env.php

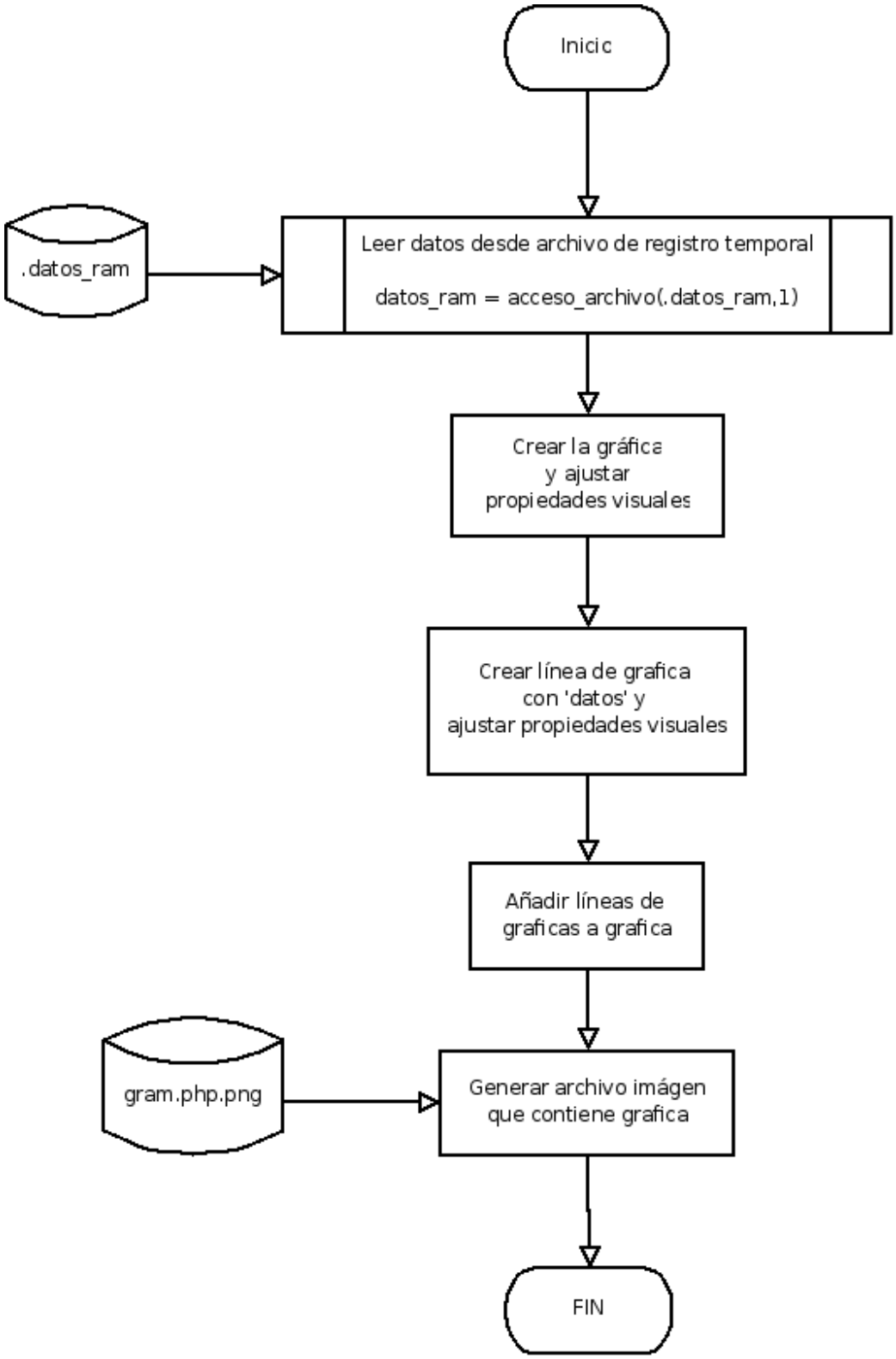


Diagrama 25. gram.php

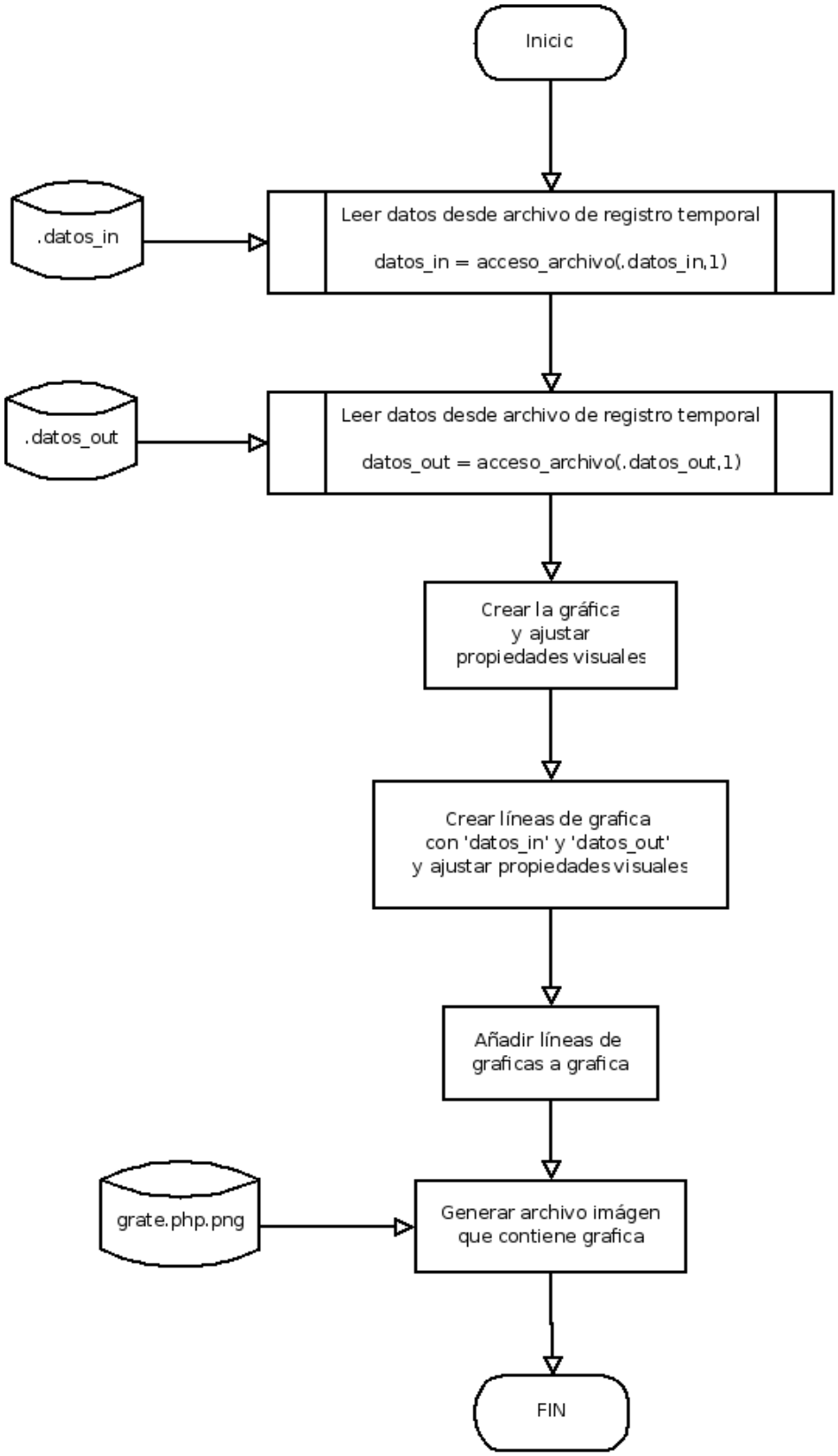


Diagrama 26. grate.php

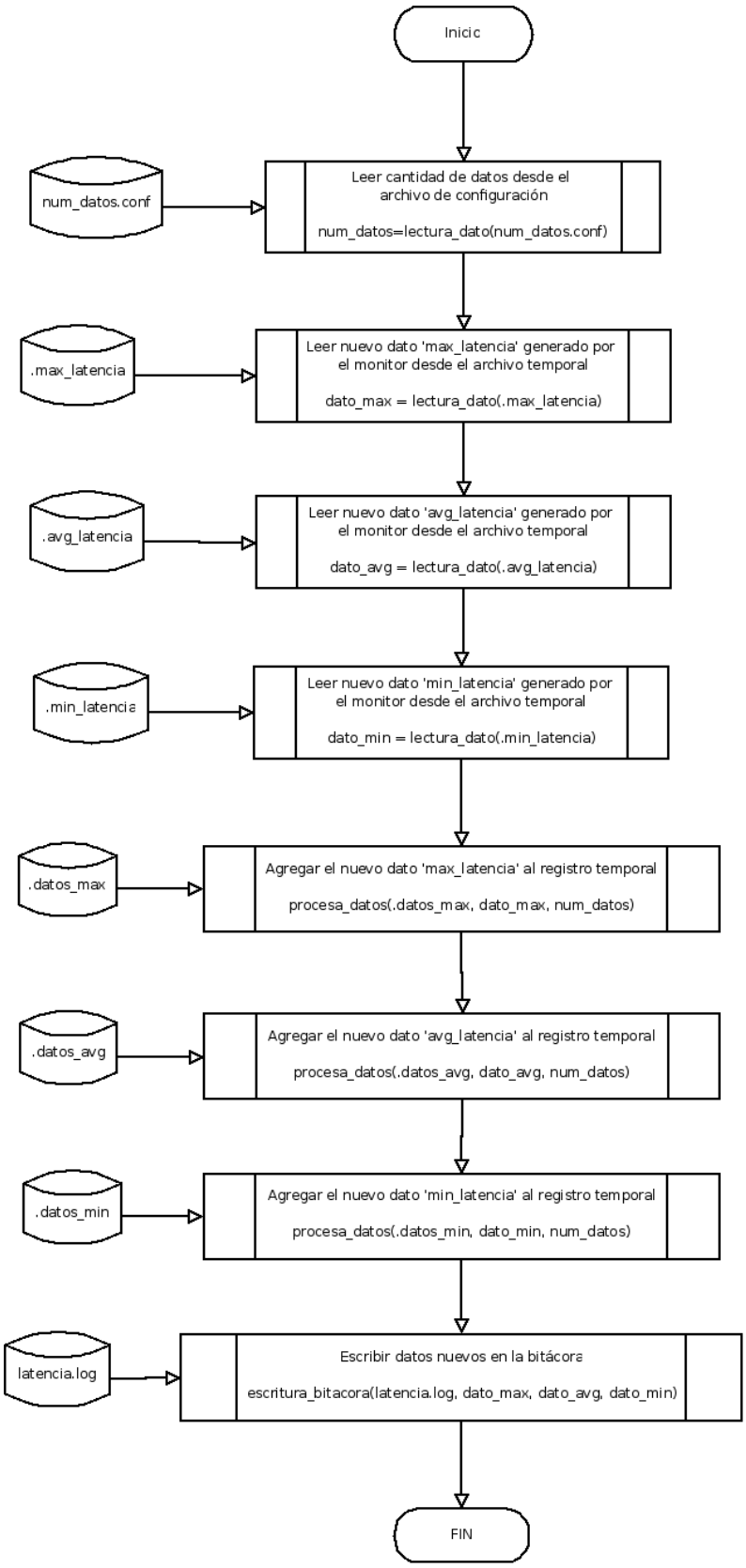


Diagrama 27. latencia.c

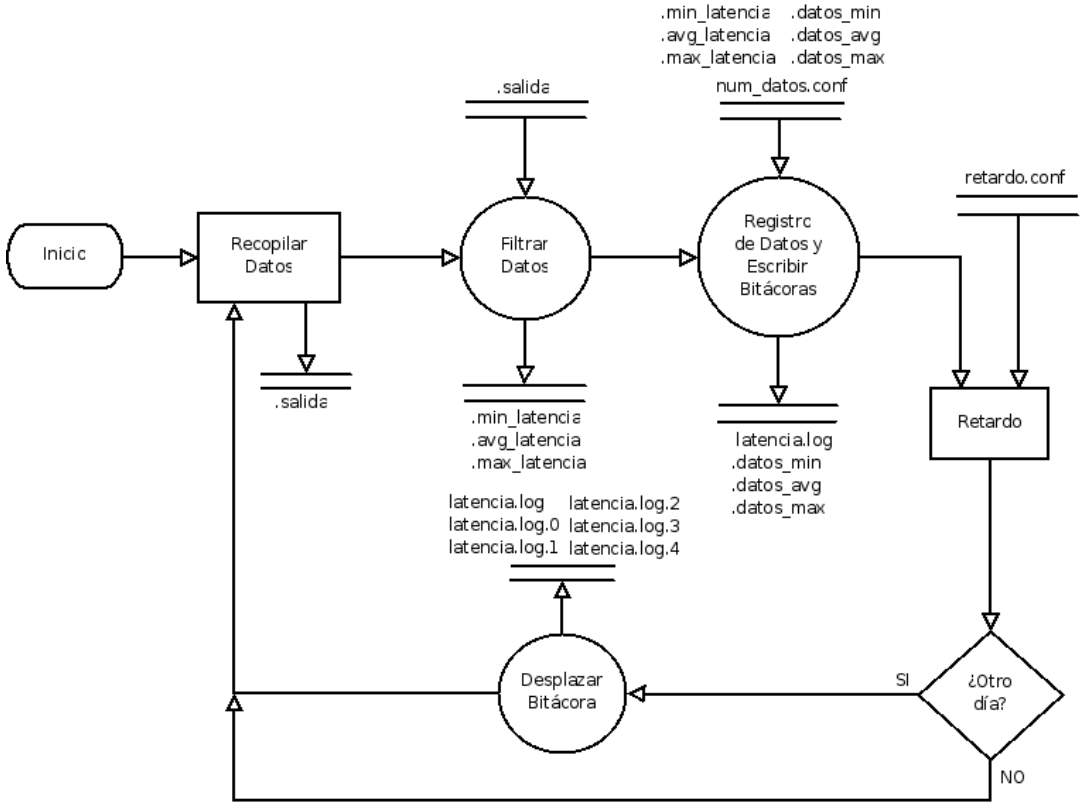


Diagrama 28. latencia.sh

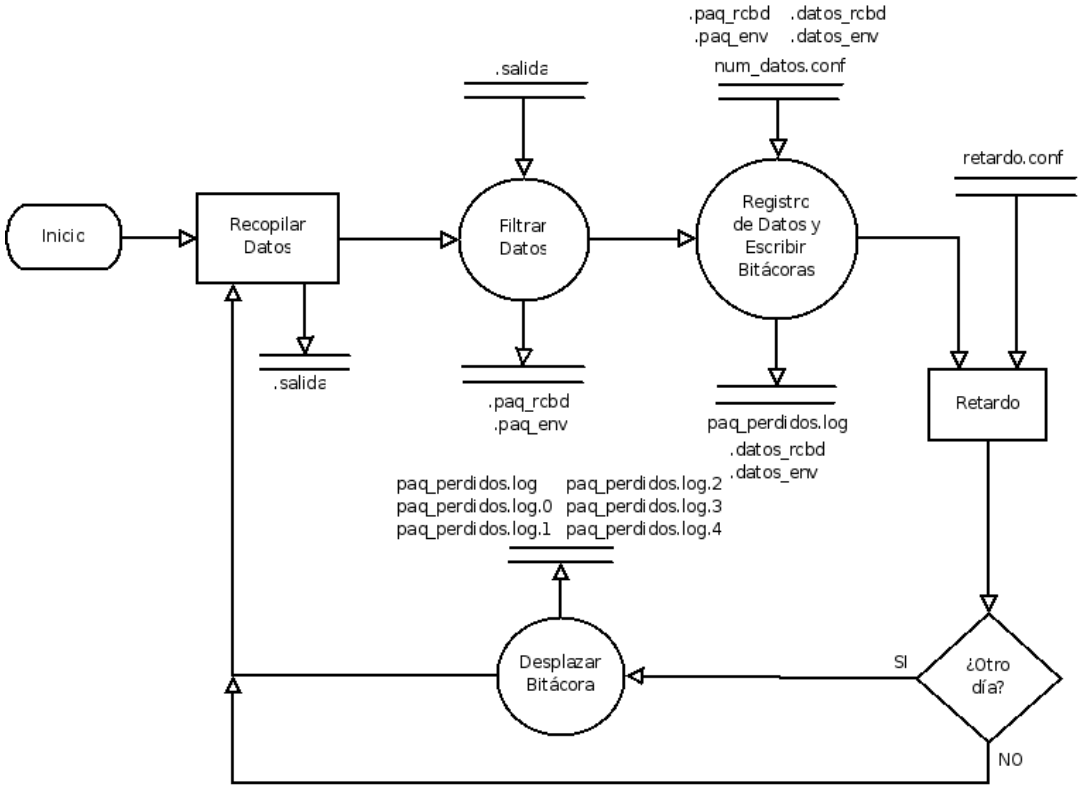


Diagrama 29. paq_perdidos.sh

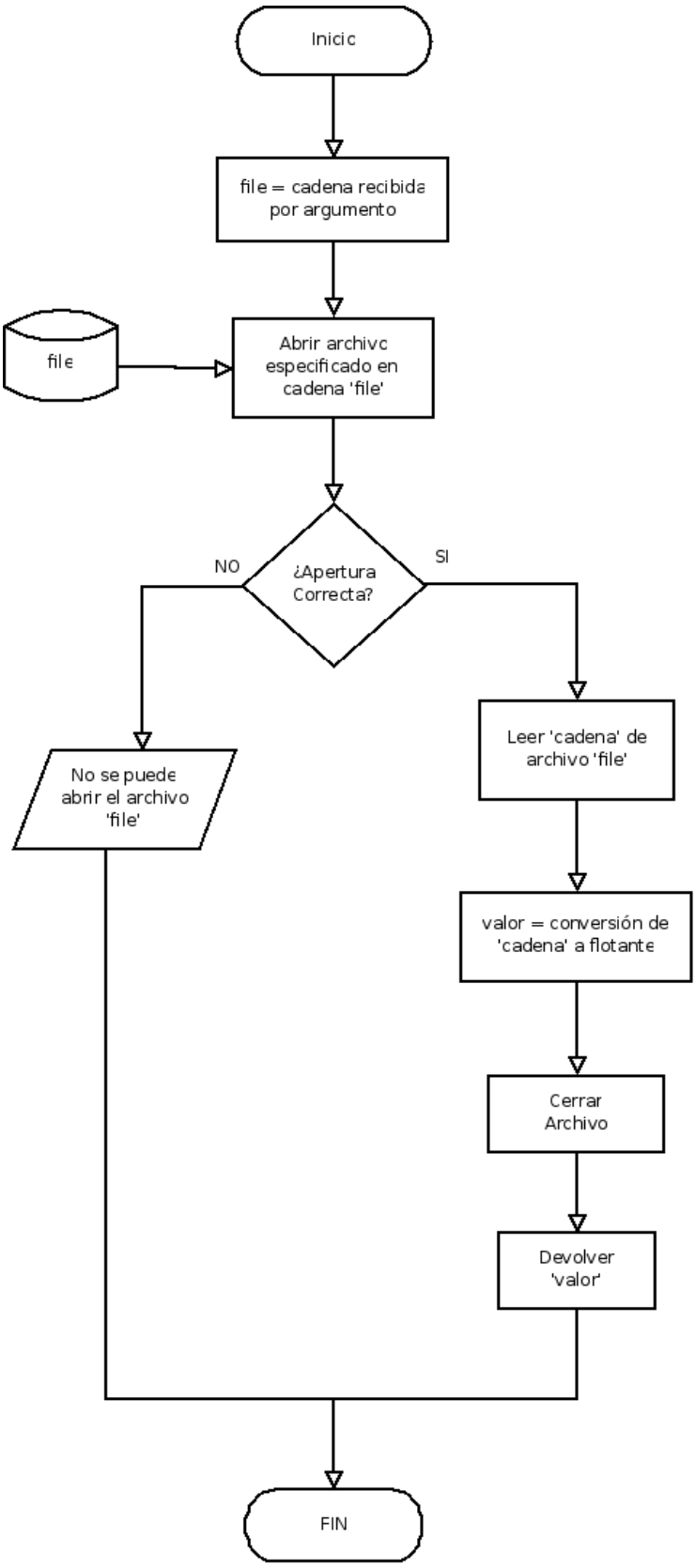


Diagrama 30. lectura_dato.c

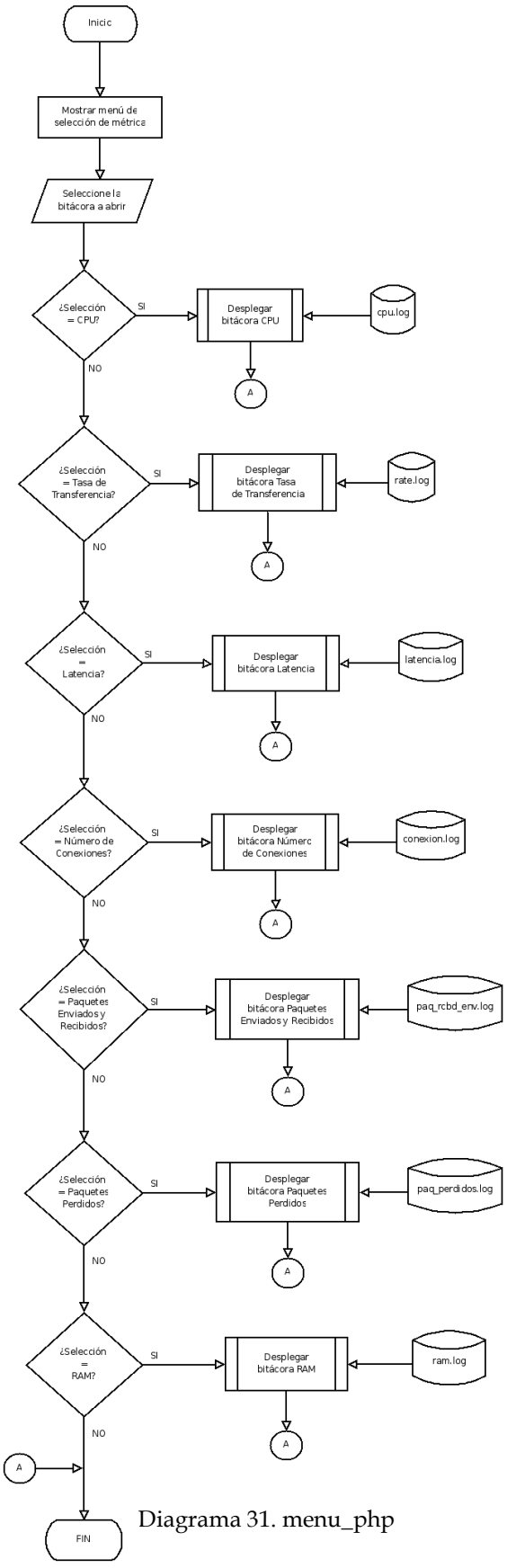


Diagrama 31. menu_php

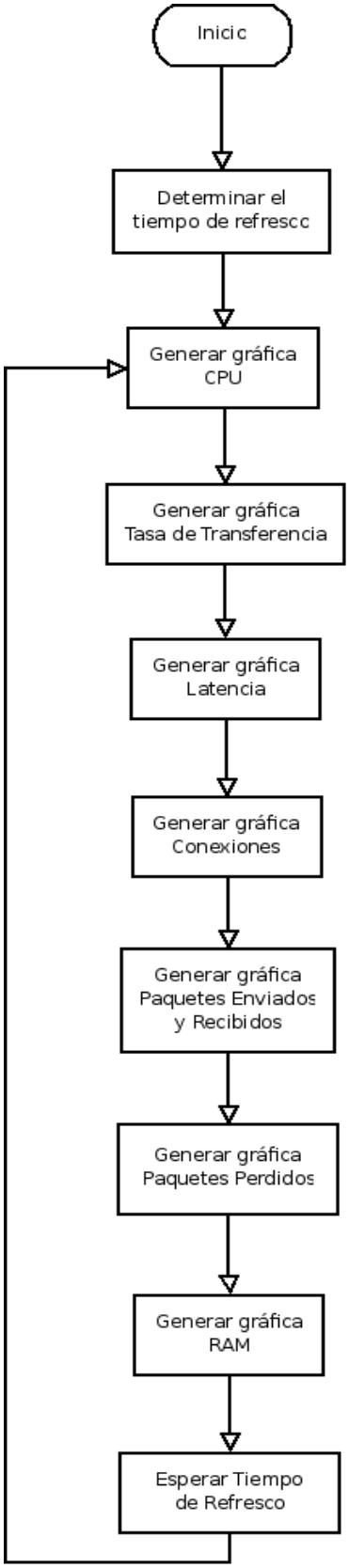


Diagrama 32. monitool.php

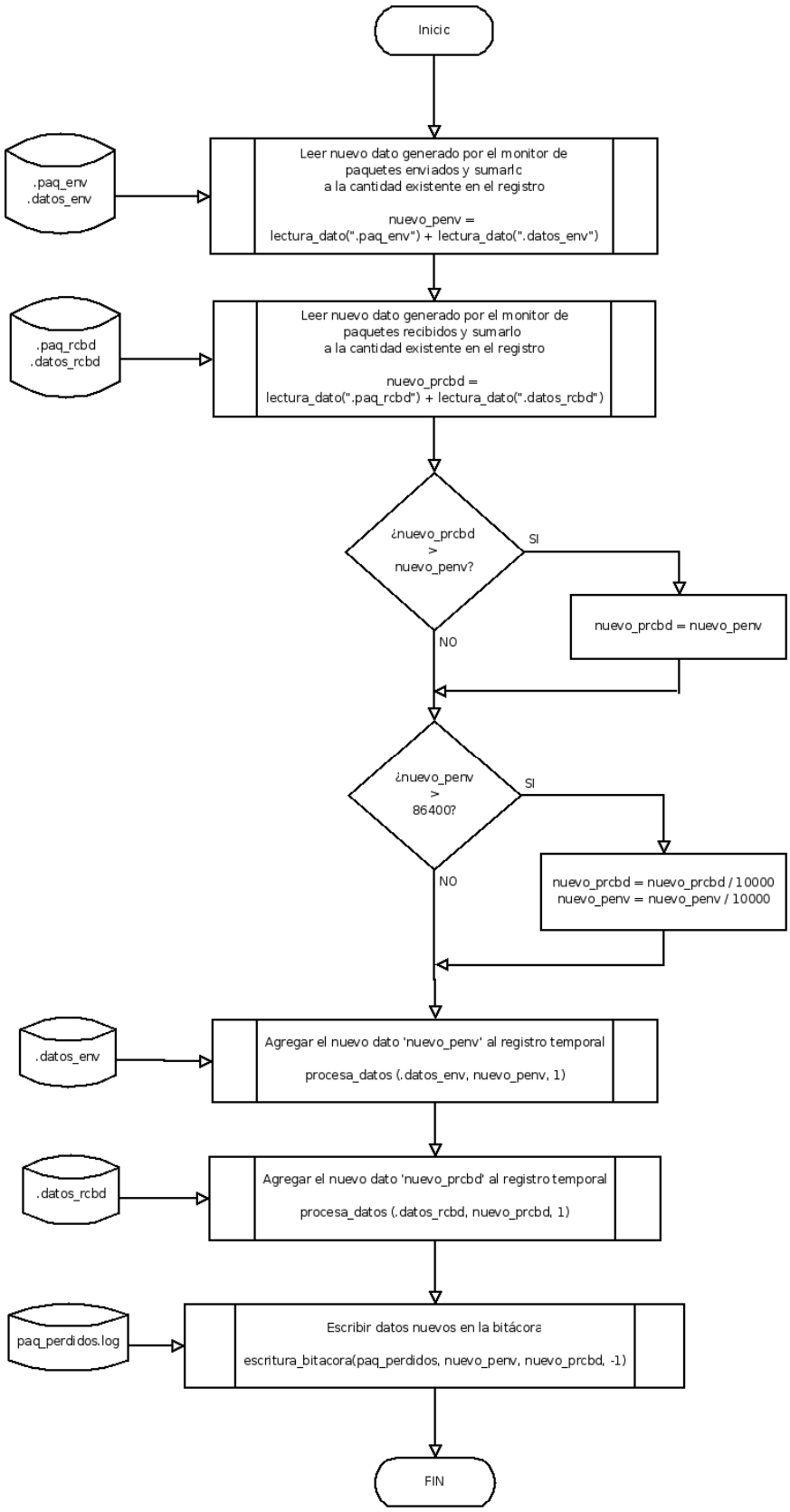


Diagrama 33. paq_perdidos.c

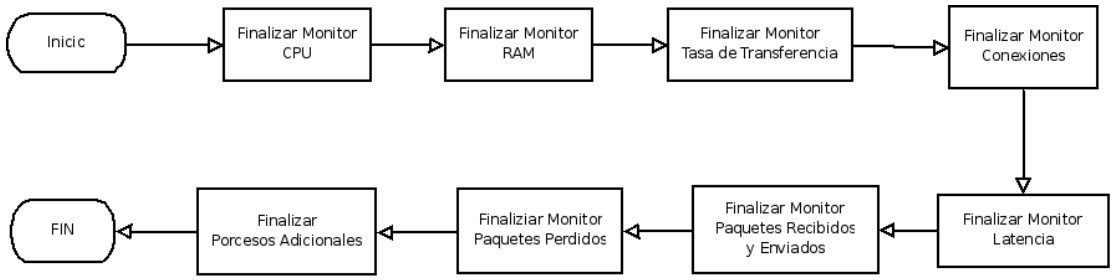


Diagrama 34. monitool_stop.sh

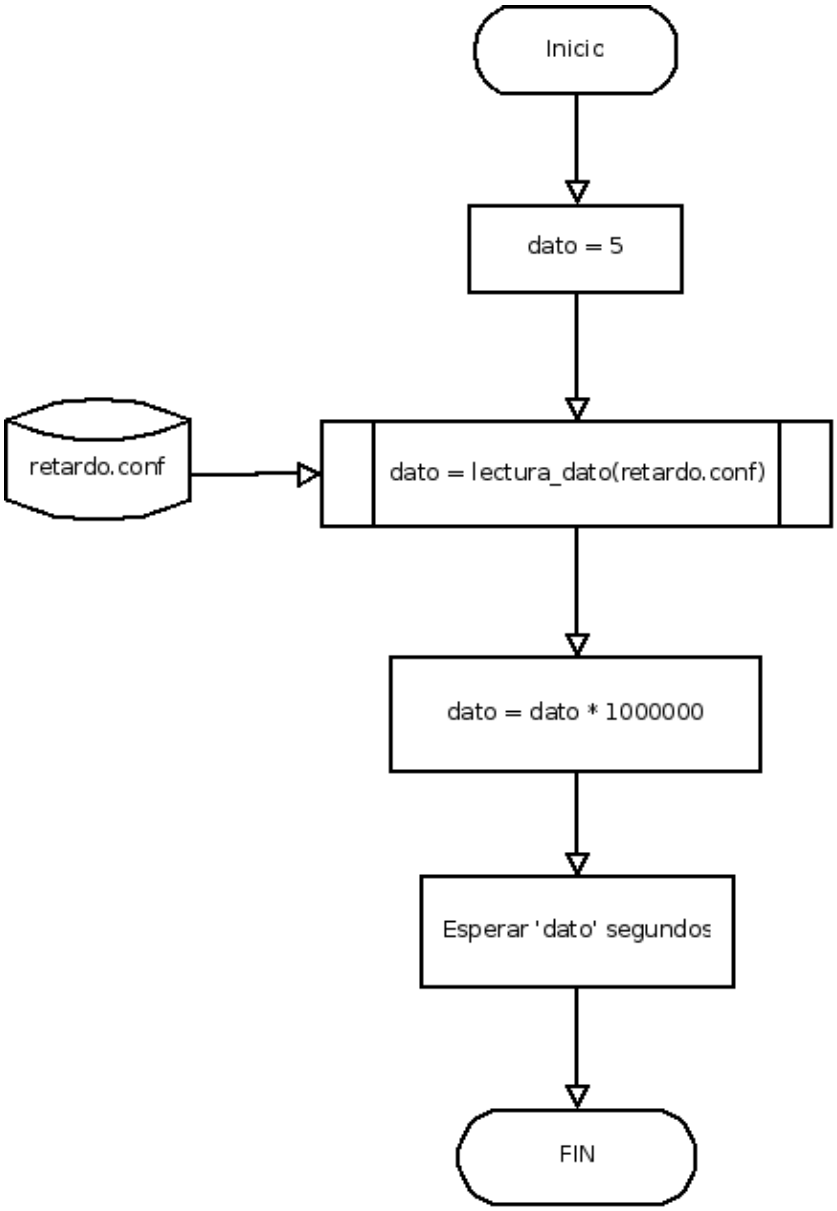


Diagrama 35. retardo.c

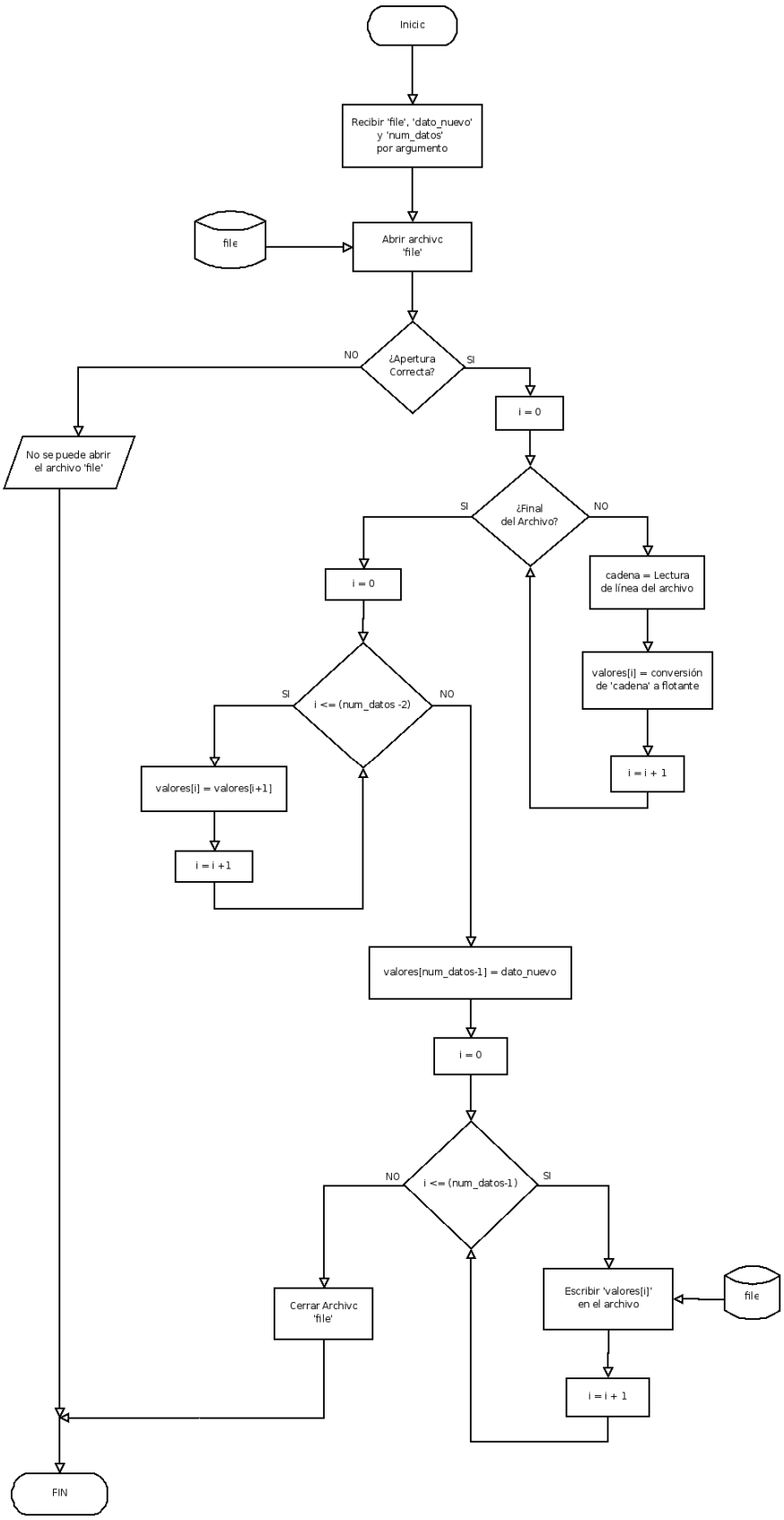


Diagrama 36. proceso_datos.c

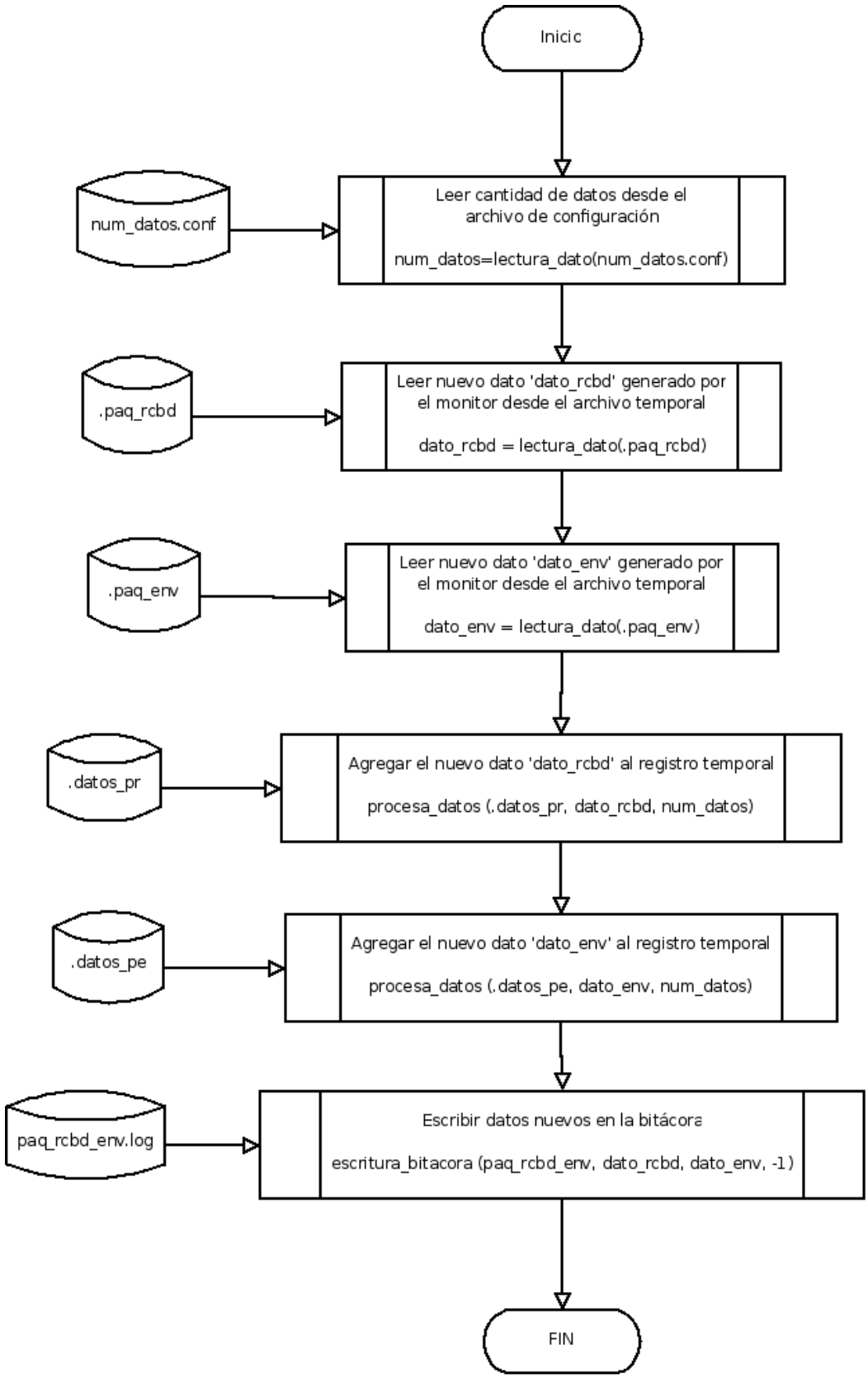


Diagrama 37. paq_rcbd_env.c

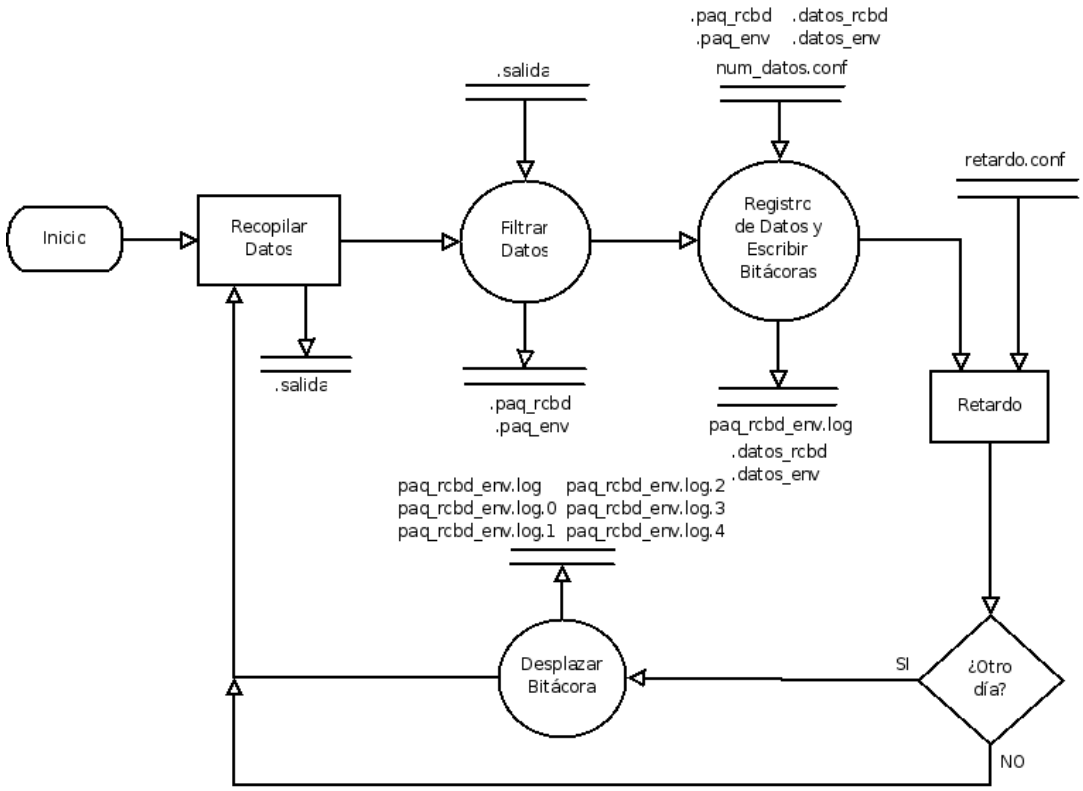


Diagrama 38. paq_rcbd_env.sh

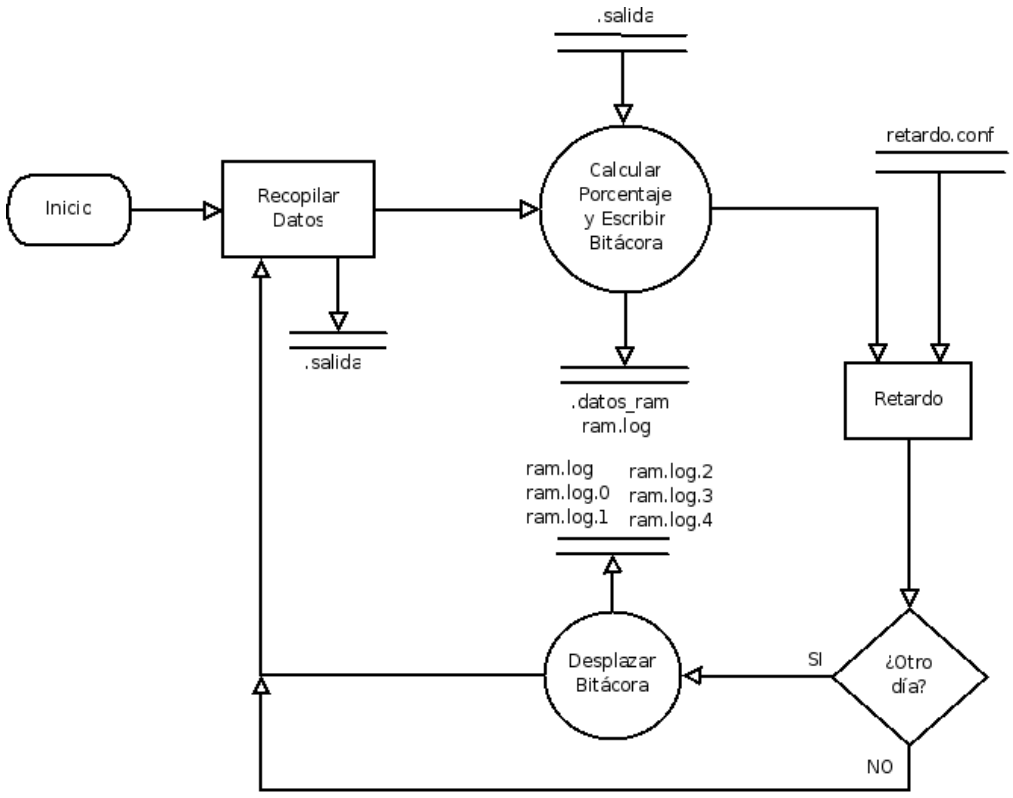


Diagrama 39. ram.sh

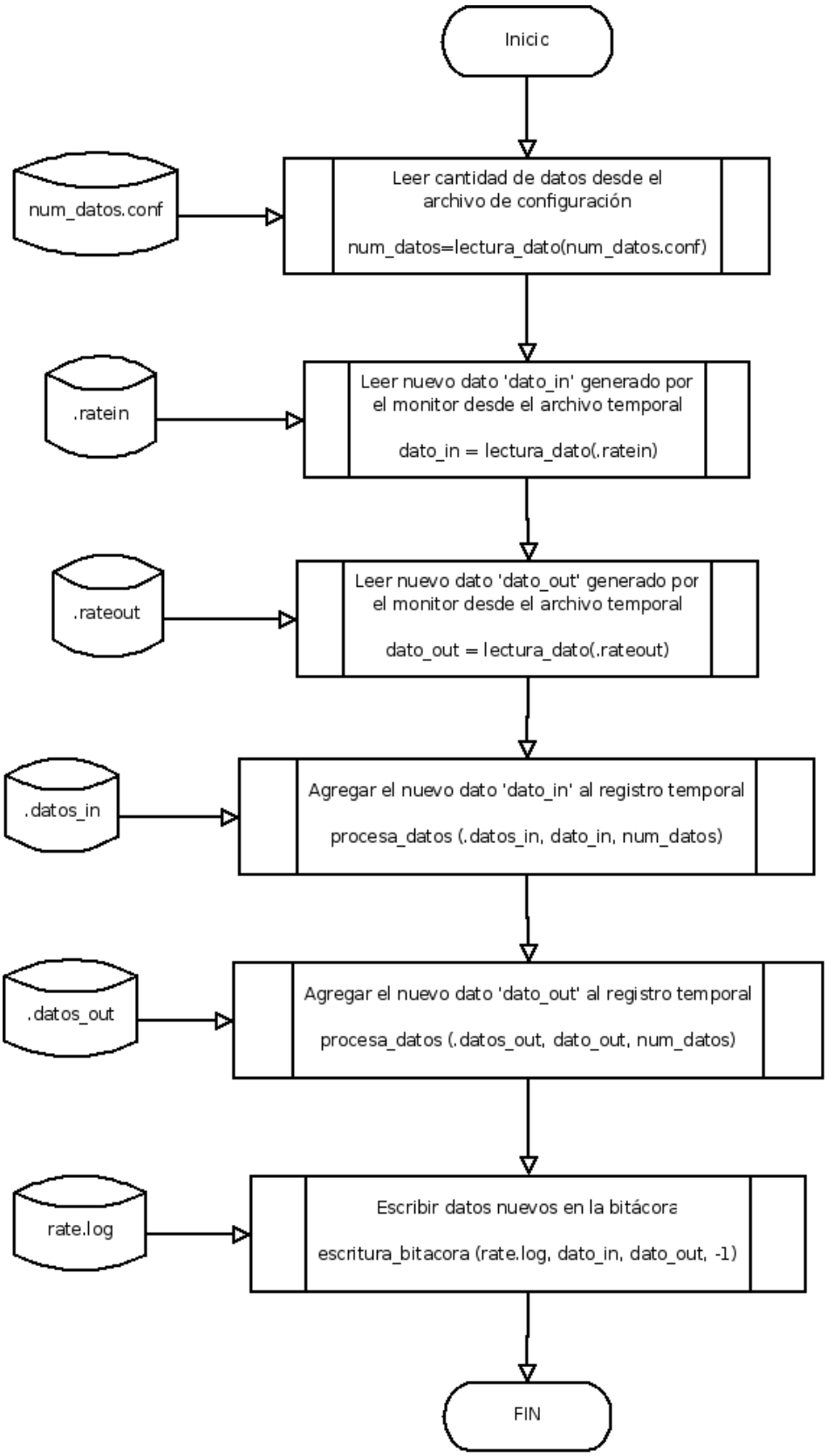


Diagrama 40. rate.c

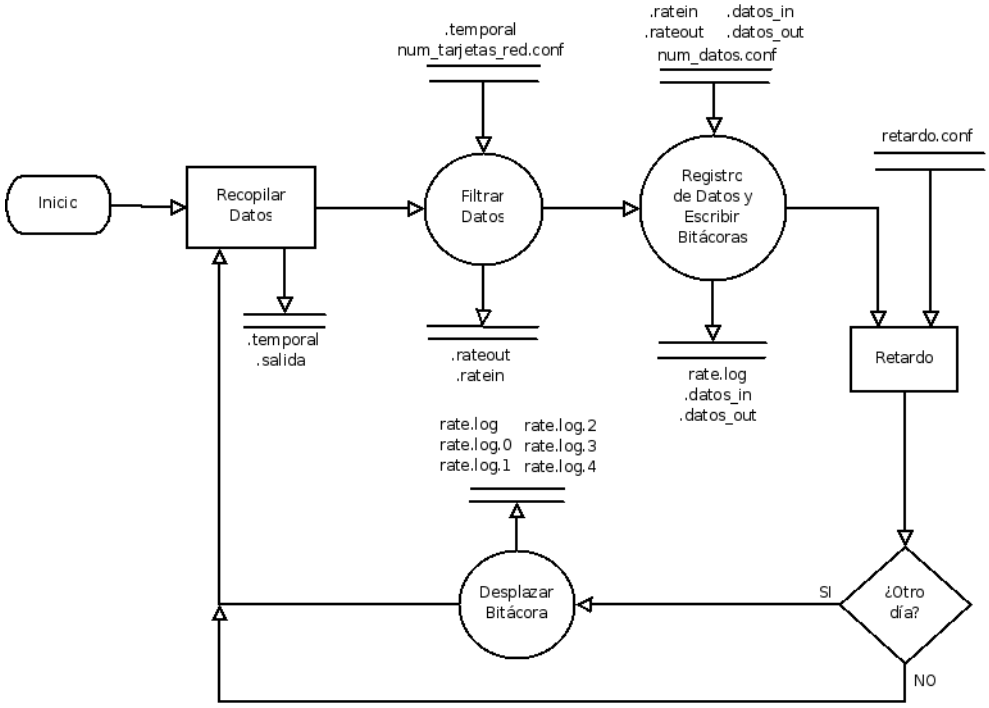


Diagrama 41. rate.sh

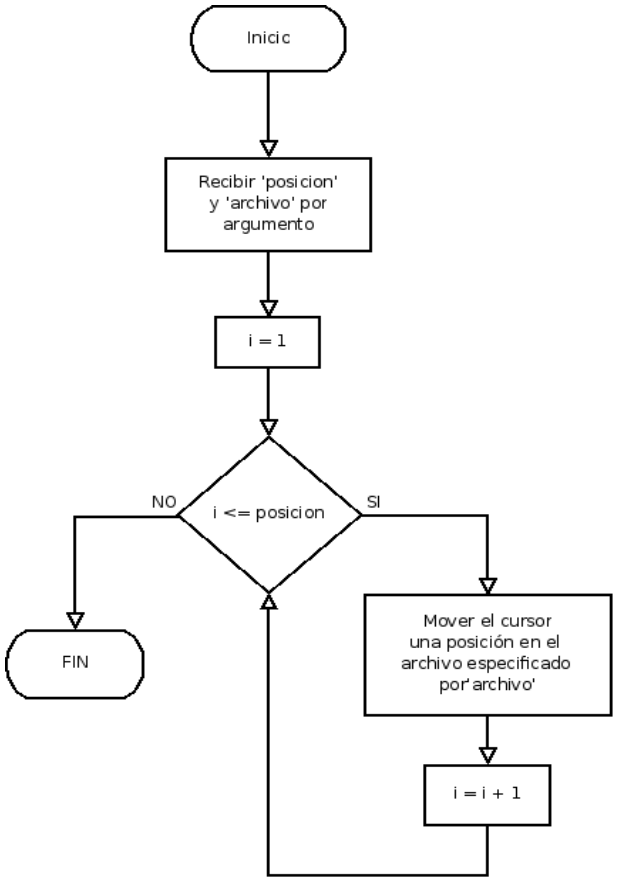


Diagrama 42. recorre_posicion.c

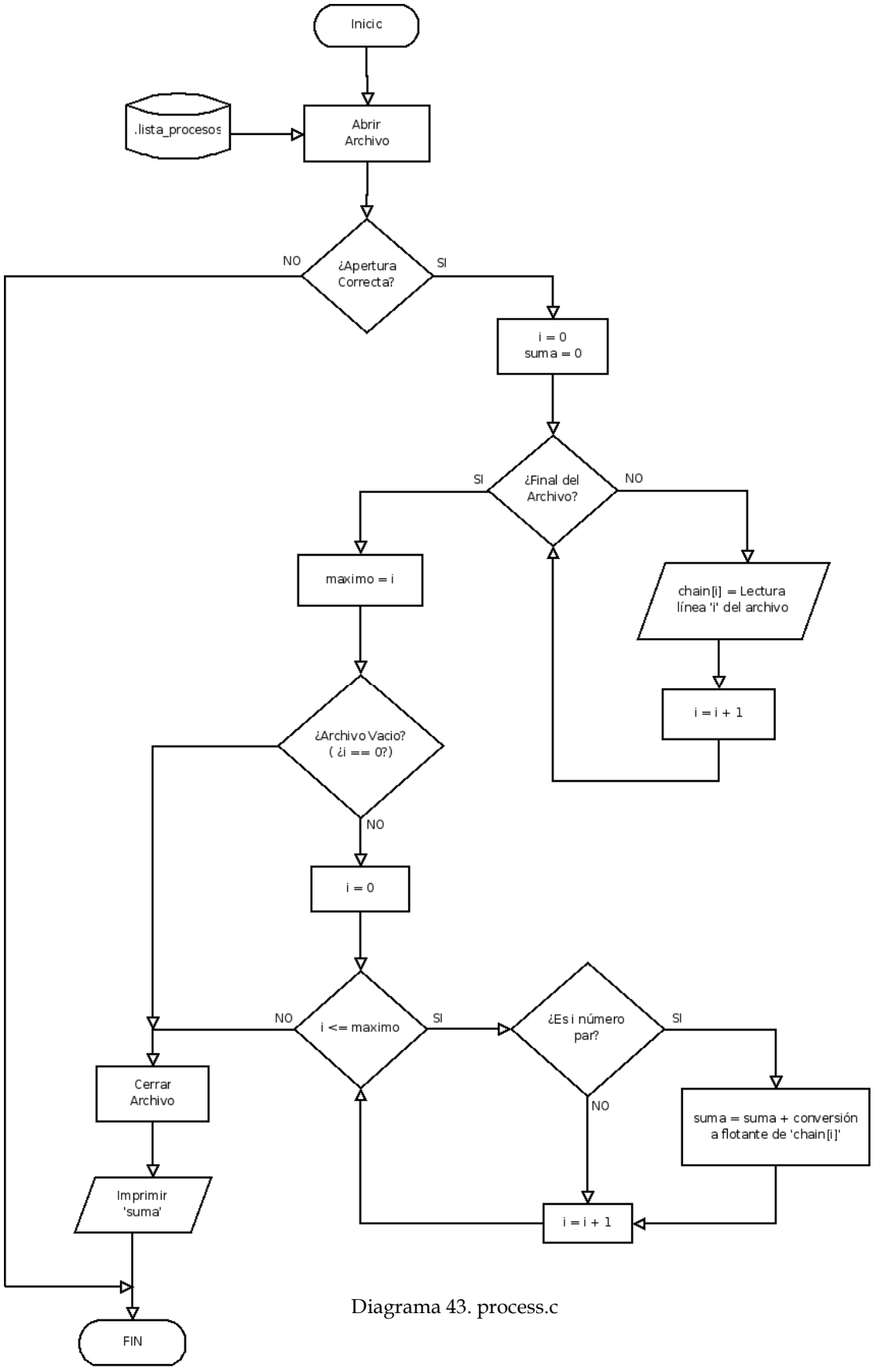


Diagrama 43. process.c

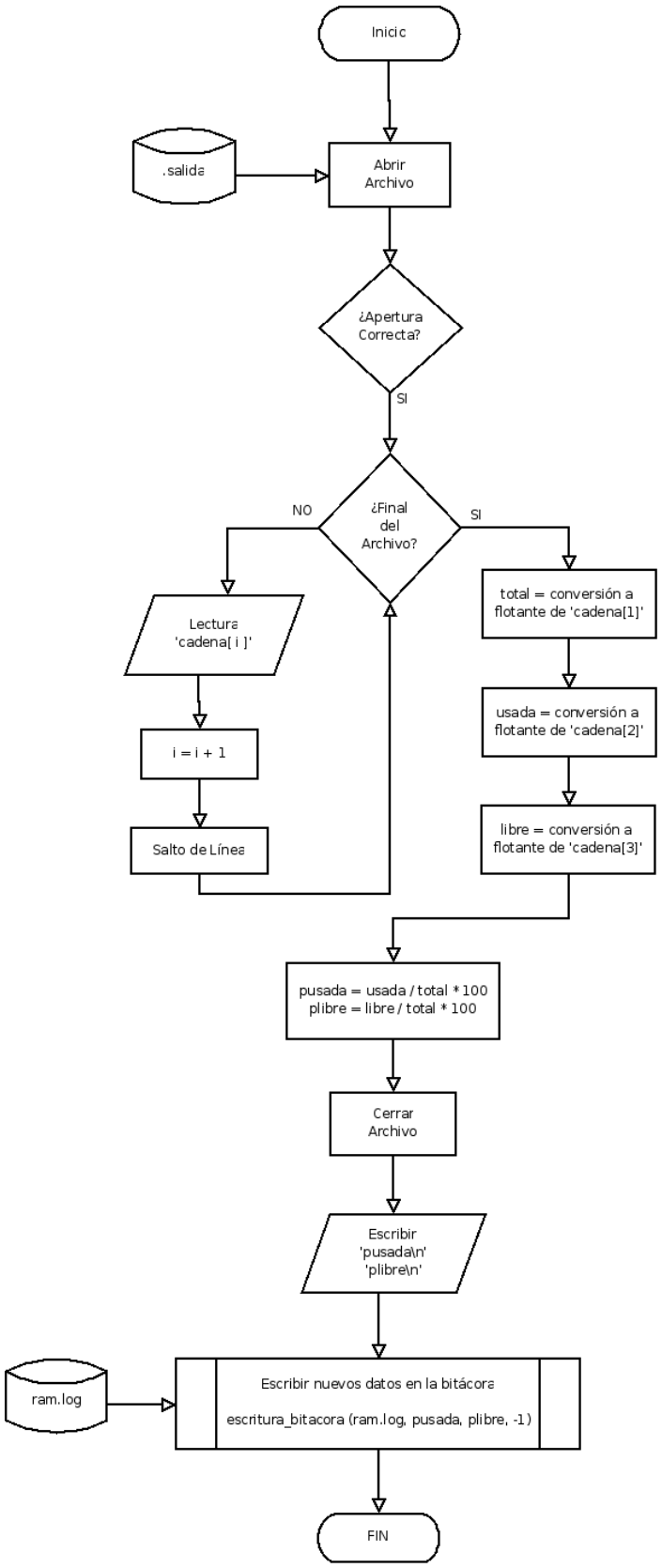


Diagrama 44. ram.c

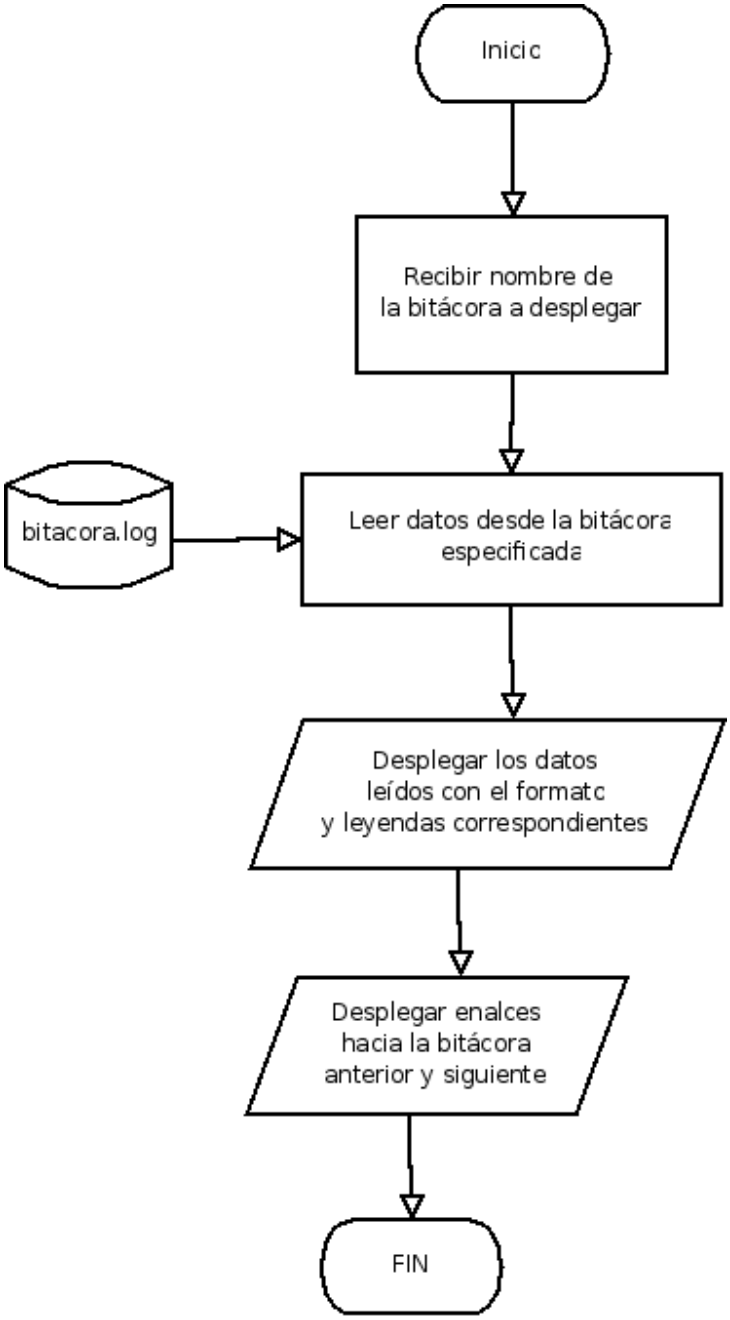


Diagrama 45. ver_bitacora.php

ANEXO D

Código Fuente

Archivo *alarma.c*:

```
#include "funciones.h"

main()
{
	int i=0;
	float contador;
	float cpu=0, conexion=0, latencia=0, rate=0, ram=0, paq_p=0;
	FILE *archivo;
	char cadena[20];
	time_t tiempo;
	struct tm *tmPtr;

	if (lectura_dato("../cpu/.contador") <= 1) { // Nuevo promedio
		// Se crea el archivo que contendrá¡ el cuerpo del correo electrónico
		if( (archivo = fopen("alarma.txt", "w")) == NULL) {
			printf("C: No se puede abrir el archivo\n");
			return 0;
		}
		tiempo = time(NULL);
		tmPtr = localtime(&tiempo);
		strftime( cadena, 20, "%F %H:%M:%S", tmPtr );
		// La leyenda contenida en el reporte junto con la fecha en que se genera
		fprintf(archivo, "Reporte de los valores excedidos en la ultima
		hora\n%s\n\n", cadena);
		fclose(archivo);

		// Se abre nuevamente el archivo para agregarle los reportes de cada una
		if( (archivo = fopen("alarma.txt", "a+")) == NULL) {
			printf("C: No se puede abrir el archivo\n");
			return 0;
		}

		// Promedio de uso de CPU en la ultima hora
		cpu = lectura_dato("../cpu/.promedio_cpu");
		if(cpu >= lectura_dato("../config/.cpu_max")) { // Valor Máximo
		fprintf(archivo, "Promedio de CPU en la ultima hora: %f\%\n", cpu);
			i++;
		}

		// Promedio de Conexiones en la ultima hora
		conexion = lectura_dato("../conexiones/.promedio_con");
		if(conexion >= lectura_dato("../config/.conexion_max")) {
		// Valor Máximo
			printf(archivo, "Promedio de Conexiones en la ultima hora: %f
			conexiones\n", conexion);
			i++;
		}

		// Promedio de Latencia en la ultima hora
		latencia = lectura_dato("../latencia/.promedio_lat");
		if(latencia >= lectura_dato("../config/.latencia_max")) { // Valor Máximo
			fprintf(archivo, "Promedio de Latencia en la ultima hora: %f
			msegundos\n", latencia);
			i++;
		}

		// Promedio de Tasa de Transferencia en la ultima hora
		rate = lectura_dato("../rate/.promedio_rate");
		if(rate >= lectura_dato("../config/.rate_max")) { // Valor Máximo
			fprintf(archivo, "Promedio de Tasa de Transferencia en la ultima
			hora: %f kb/s\n", rate);
			i++;
		}

		// Promedio de RAM Usada en la ultima hora
		ram = lectura_dato("../ram/.promedio_ram");
		if(ram >= lectura_dato("../config/.ram_max")) { // Valor Máximo
			fprintf(archivo, "Promedio de RAM Usada en la ultima hora:
			%f\%\n", ram);
			i++;
		}

		// Promedio de Paquetes Perdidos en la ultima hora
		paq_p = lectura_dato("../paq_perdidos/.promedio_paqp");
		if(paq_p >= lectura_dato("../config/.paq_perdidos_max")) {
		// Valor Máximo
			fprintf(archivo, "Promedio de Paquetes perdidos en la ultima
			hora: %f\%\n", paq_p);
			i++;
		}
		fclose (archivo);
	}
	if (i > 0) {
		system("cat alarma.txt | mail -s Alarma root@localhost");
	}
	return 0; }
```

Archivo *bitacora_con.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que compara la fecha del primer dato de la bitácora principal
            y si es igual a la fecha actual no hace nada, pero si es distinta recorre
            los datos de las bitácoras, acumulando los datos mas antiguos en la
            bitácora "log.4" y limpia la "log" para datos recientes.
            Acumula o desplaza los datos antiguos en la bitácora "log.4" de
            acuerdo al valor en el archivo de configuración "borrar_bitacora.conf"
*/

#include "funciones.h"
#include "funciones2.h"

main()
{
            int valor = 0;

            if( comparar_fechas("../logs/conexion.log") == 1 ) { // Si las fechas son distintas (==1)

                        // Desplazamiento de los datos de las bitácoras

                        // Se determina si los datos se acumulan o
                        // borran desde el archivo de configuración
                        valor = lectura_dato("../config/.borrar_bitacora.conf");

                        // Se hace lo correspondiente de acuerdo al archivo
                        copia_archivo("../logs/conexion.log.3" , "../logs/conexion.log.4", valor);


                        // Se sobre-escriben en el destino
                        copia_archivo("../logs/conexion.log.2" , "../logs/conexion.log.3", 1);
                        copia_archivo("../logs/conexion.log.1" , "../logs/conexion.log.2", 1);
                        copia_archivo("../logs/conexion.log.0" , "../logs/conexion.log.1", 1);
                        copia_archivo("../logs/conexion.log"   , "../logs/conexion.log.0", 1);

                        // Limpia la bitácora mas reciente
            }

            else; // Si las fechas son iguales (==0)

            return 0;
}
```

Archivo *bitacora_cpu.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que compara la fecha del primer dato de la bitácora principal
            y si es igual a la fecha actual no hace nada, pero si es distinta recorre
            los datos de las bitácoras, acumulando los datos mas antiguos en la
            bitácora "log.4" y limpia la "log" para datos recientes.
            Acumula o desplaza los datos antiguos en la bitácora "log.4" de
            acuerdo al valor en el archivo de configuración "borrar_bitacora.conf"
*/

#include "funciones.h"
#include "funciones2.h"

main()
{
            int valor = 0;

            if( comparar_fechas("../logs/cpu.log") == 1) { // Si las fechas son distintas  (==1)

                        // Desplazamiento de los datos de las bitácoras

                        // Se determina si los datos se acumulan o
                        // borran desde el archivo de configuración
                        valor = lectura_dato("../config/.borrar_bitacora.conf");

                        // Se hace lo correspondiente de acuerdo al archivo
                        copia_archivo("../logs/cpu.log.3" , "../logs/cpu.log.4", valor);

                        // Se sobre-escriben en el destino
                        copia_archivo("../logs/cpu.log.2" , "../logs/cpu.log.3", 1);
                        copia_archivo("../logs/cpu.log.1" , "../logs/cpu.log.2", 1);
                        copia_archivo("../logs/cpu.log.0" , "../logs/cpu.log.1", 1);
                        copia_archivo("../logs/cpu.log"   , "../logs/cpu.log.0", 1);

                        // Limpia la bitácora mas reciente
                        copia_archivo("/dev/null" , "../logs/cpu.log", 1);
            }

            else; // Si las fechas son iguales (==0)

            return 0;
}
```

Archivo *bitacora_lat.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que compara la fecha del primer dato de la bitácora principal
            y si es igual a la fecha actual no hace nada, pero si es distinta recorre
            los datos de las bitácoras, acumulando los datos mas antiguos en la
            bitácora "log.4" y limpia la "log" para datos recientes.
            Acumula o desplaza los datos antiguos en la bitácora "log.4" de
            acuerdo al valor en el archivo de configuración "borrar_bitacora.conf"
*/

#include "funciones.h"
#include "funciones2.h"

main()
{
            int valor = 0;

            if( comparar_fechas("../logs/latencia.log") == 1) { // Si las fechas son distintas (==1)

                        // Desplazamiento de los datos de las bitácoras

                        // Se determina si los datos se acumulan o
                        // borran desde el archivo de configuración
                        valor = lectura_dato("../config/.borrar_bitacora.conf");

                        // Se hace lo correspondiente de acuerdo al archivo
                        copia_archivo("../logs/latencia.log.3" , "../logs/latencia.log.4", valor);


                        // Se sobre-escriben en el destino
                        copia_archivo("../logs/latencia.log.2" , "../logs/latencia.log.3", 1);
                        copia_archivo("../logs/latencia.log.1" , "../logs/latencia.log.2", 1);
                        copia_archivo("../logs/latencia.log.0" , "../logs/latencia.log.1", 1);
                        copia_archivo("../logs/latencia.log"   , "../logs/latencia.log.0", 1);

                        // Limpia la bitácora mas reciente
                        copia_archivo("/dev/null" , "../logs/latencia.log", 1);
            }

            else; // Si las fechas son iguales (==0)

            return 0;
}
```

Archivo *bitacora_paq_r_e.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que compara la fecha del primer dato de la bitácora principal
            y si es igual a la fecha actual no hace nada, pero si es distinta recorre
            los datos de las bitácoras, acumulando los datos mas antiguos en la
            bitácora "log.4" y limpia la "log" para datos recientes.
            Acumula o desplaza los datos antiguos en la bitácora "log.4" de
            acuerdo al valor en el archivo de configuración "borrar_bitacora.conf"
*/

#include "funciones.h"
#include "funciones2.h"

main()
{
            int valor = 0;
            if( comparar_fechas("../logs/paq_rcbd_env.log") == 1) {
                        // Si las fechas son distintas (==1)
                        // Desplazamiento de los datos de las bitácoras
                        // Se determina si los datos se acumulan o
                        // borran desde el archivo de configuración
                        valor = lectura_dato("../config/.borrar_bitacora.conf");

                        // Se hace lo correspondiente de acuerdo al archivo
                        copia_archivo("../logs/paq_rcbd_env.log.3" ,
                                                "../logs/paq_rcbd_env.log.4", valor);

                        // Se sobre-escriben en el destino
            copia_archivo("../logs/paq_rcbd_env.log.2" , "../logs/paq_rcbd_env.log.3", 1);
            copia_archivo("../logs/paq_rcbd_env.log.1" , "../logs/paq_rcbd_env.log.2", 1);
            copia_archivo("../logs/paq_rcbd_env.log.0" , "../logs/paq_rcbd_env.log.1", 1);
            copia_archivo("../logs/paq_rcbd_env.log"   , "../logs/paq_rcbd_env.log.0", 1);

            // Limpia la bitácora mas reciente
            copia_archivo("/dev/null" , "../logs/paq_rcbd_env.log", 1);
            }

            else; // Si las fechas son iguales (==0)

            return 0;
}
```

Archivo *bitacora_paqp.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que compara la fecha del primer dato de la bitácora principal
            y si es igual a la fecha actual no hace nada, pero si es distinta recorre
            los datos de las bitácoras, acumulando los datos mas antiguos en la
            bitácora "log.4" y limpia la "log" para datos recientes.
            Acumula o desplaza los datos antiguos en la bitácora "log.4" de
            acuerdo al valor en el archivo de configuración "borrar_bitacora.conf"
*/

#include "funciones.h"
#include "funciones2.h"

main()
{
            int valor = 0;
            if( comparar_fechas("../logs/paq_perdidos.log") == 1) {
                        // Si las fechas son distintas (==1)
                        // Desplazamiento de los datos de las bitácoras
                        // Se determina si los datos se acumulan o
                        // borran desde el archivo de configuración
                        valor = lectura_dato("../config/.borrar_bitacora.conf");

                        // Se hace lo correspondiente de acuerdo al archivo
                        copia_archivo("../logs/paq_perdidos.log.3" ,
                                                "../logs/paq_perdidos.log.4", valor);

                        // Se sobre-escriben en el destino
            copia_archivo("../logs/paq_perdidos.log.2" , "../logs/paq_perdidos.log.3", 1);
            copia_archivo("../logs/paq_perdidos.log.1" , "../logs/paq_perdidos.log.2", 1);
            copia_archivo("../logs/paq_perdidos.log.0" , "../logs/paq_perdidos.log.1", 1);
            copia_archivo("../logs/paq_perdidos.log"   , "../logs/paq_perdidos.log.0", 1);

            // Limpia la bitácora mas reciente
            copia_archivo("/dev/null" , "../logs/paq_perdidos.log", 1);
            }

            else; // Si las fechas son iguales (==0)

            return 0;
}
```

Archivo *bitacora_ram.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que compara la fecha del primer dato de la bitácora principal
            y si es igual a la fecha actual no hace nada, pero si es distinta recorre
            los datos de las bitácoras, acumulando los datos mas antiguos en la
            bitácora "log.4" y limpia la "log" para datos recientes.
            Acumula o desplaza los datos antiguos en la bitácora "log.4" de
            acuerdo al valor en el archivo de configuración "borrar_bitacora.conf"
*/

#include "funciones.h"
#include "funciones2.h"

main()
{
            int valor = 0;
            if( comparar_fechas("../logs/ram.log") == 1) { // Si las fechas son distintas (==1)

                        // Desplazamiento de los datos de las bitácoras
                        // Se determina si los datos se acumulan o
                        // borran desde el archivo de configuración
                        valor = lectura_dato("../config/.borrar_bitacora.conf");

                        // Se hace lo correspondiente de acuerdo al archivo
                        copia_archivo("../logs/ram.log.3" , "../logs/ram.log.4", valor);

                        // Se sobre-escriben en el destino
                        copia_archivo("../logs/ram.log.2" , "../logs/ram.log.3", 1);
                        copia_archivo("../logs/ram.log.1" , "../logs/ram.log.2", 1);
                        copia_archivo("../logs/ram.log.0" , "../logs/ram.log.1", 1);
                        copia_archivo("../logs/ram.log"   , "../logs/ram.log.0", 1);

                        // Limpia la bitácora mas reciente
                        copia_archivo("/dev/null" , "../logs/ram.log", 1);
            }

            else; // Si las fechas son iguales (==0)

            return 0;
}
```

Archivo *bitacora_rate.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que compara la fecha del primer dato de la bitácora principal
            y si es igual a la fecha actual no hace nada, pero si es distinta recorre
            los datos de las bitácoras, acumulando los datos mas antiguos en la
            bitácora "log.4" y limpia la "log" para datos recientes.
            Acumula o desplaza los datos antiguos en la bitácora "log.4" de
            acuerdo al valor en el archivo de configuración "borrar_bitacora.conf"
*/

#include "funciones.h"
#include "funciones2.h"

main()
{
            int valor = 0;
            if( comparar_fechas("../logs/rate.log") == 1) { // Si las fechas son distintas (==1)

                        // Desplazamiento de los datos de las bitácoras
                        // Se determina si los datos se acumulan o
                        // borran desde el archivo de configuración
                        valor = lectura_dato("../config/.borrar_bitacora.conf");

                        // Se hace lo correspondiente de acuerdo al archivo
                        copia_archivo("../logs/rate.log.3" , "../logs/rate.log.4", valor);

                        // Se sobre-escriben en el destino
                        copia_archivo("../logs/rate.log.2" , "../logs/rate.log.3", 1);
                        copia_archivo("../logs/rate.log.1" , "../logs/rate.log.2", 1);
                        copia_archivo("../logs/rate.log.0" , "../logs/rate.log.1", 1);
                        copia_archivo("../logs/rate.log"   , "../logs/rate.log.0", 1);

                        // Limpia la bitácora mas reciente
                        copia_archivo("/dev/null" , "../logs/rate.log", 1);
            }

            else; // Si las fechas son iguales (==0)

            return 0;
}
```

Archivo *conexion.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 11 de Septiembre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
   Programa que lee el ultimo numero de conexiones desde el archivo .internet y .unix,
   recorre los valores y coloca los obtenidos en la ultima posición de los registros
   en los archivos .datos_internet y .datos_unix respectivamente
*/

#include "funciones.h"
main()
{
            float dato_internet=0.0, dato_unix=0.0, suma=0.0;
            int contador;

            // Apertura del archivo .internet, si hay error sale del programa
            dato_internet = lectura_dato(".internet");

            // Apertura del archivo .unix, si hay error sale del programa
            dato_unix = lectura_dato(".unix");

            // Se agregan los datos en la ultima posición del archivo registro
            proceso_datos(".datos_internet" , dato_internet);
            proceso_datos(".datos_unix", dato_unix);

            // Almacenamiento del dato obtenido en el archivo registro
            escritura_bitacora("../logs/conexion.log", dato_internet, dato_unix, -1);

            // Calculo del promedio para las alarmas
            contador = lectura_dato(".contador");
            suma = lectura_dato(".suma");

            if (contador < (3600 / lectura_dato("../config/.retardo.conf"))) {
                        contador++;
                        suma += dato_internet;
            }
            else {
                        escritura_dato(".promedio_con", suma/contador);
                        suma = dato_internet;
                        contador = 1;
            }
            escritura_dato(".contador", contador);
            escritura_dato(".suma", suma);
            return 0;
}
```

Archivo *cpu.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 22 de Agosto del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
   Programa que lee el último porcentaje de uso del CPU desde el archivo .cpu
   y lo almacena en la última posición del registro contenido en el archivo .datos_cpu
   además de enviar a la salida estándar el registro completo
*/

#include "funciones.h"

main()
{
        float dato=0.0, suma=0.0;
        int contador;

        // Apertura del archivo .cpu, si hay error sale del programa
        dato = lectura_dato(".cpu");

        // Agrega el nuevo dato en la última posición del registro temporal
        proceso_datos(".datos_cpu", dato);

        // Almacenamiento del dato obtenido en el archivo registro
        escritura_bitacora("../logs/cpu.log", dato, -1, -1);

        // Calculo del promedio para las alarmas
        contador = lectura_dato(".contador");
        suma = lectura_dato(".suma");

        if (contador < (3600 / lectura_dato("../config/.retardo.conf"))) {
                contador++;
                suma += dato;
        }

        else {
                escritura_dato(".promedio_cpu", suma/contador);
                suma = dato;
                contador = 1;
        }

        escritura_dato(".contador", contador);
        escritura_dato(".suma", suma);

        return 0;
}
```

Archivo *creación_regs.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 1 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
        Programa que crea un registros temporal base lleno con 0's para los datos,
   de acuerdo a la cantidad de datos definida en el archivo de configuración
   num_datos.conf
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "funciones.h"

main()
{

        FILE *archivo;
        char cadena[20];
        int i, num_datos = 0;

//
//      Lectura del tamaño de los archivos de registro
//

        num_datos = lectura_dato("../config/.num_datos.conf");

//
//      Creación del archivo base para los registros
//
        if ( (archivo = fopen(".registro", "w")) == NULL) {
                printf("C .registro: No se puede abrir el archivo\n");
                return 0;
        }
        else {
                for (i=0; i<=(num_datos-1); i++)
                        fprintf(archivo, "0\n");
        }

        fclose(archivo);

        return 0;
}
```

Archivo *funciones.h*:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que lee el contenido del archivo especificado por la variable file,
            lo convierte a flotante y devuelve mediante la variable archivo.
            Utilizado por los módulos de obtención de manipulación de registros temporales
            de las métricas y obtención de datos
*/

float lectura_dato(char file[20]) {

            FILE *archivo;
            char cadena[20];
            float valor;

            // Apertura del archivo
            if( (archivo = fopen(file, "r")) == NULL) {
                        printf("C %s: No se puede abrir el archivo\n", file);
                        return 0;
            }
            else {
                        // Lee el valor desde el archivo y lo almacena en una variable
                        fgets(cadena, 13, archivo);
                        valor = atof(cadena);
            }
            fclose(archivo);
            return (valor);
}
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que abre el archivo especificado por la variable file,
            lee su contenido y lo almacena en el arreglo valores, desplaza los
            datos una posición y añade el nuevo dato recibido en el último lugar.
            Determina además cuantos datos hay en el registro temporal.
            Utilizada por todos los módulos de manipulación de los registros
            temporales.
*/

float proceso_datos(char file[20], float dato_nuevo) {

            FILE *archivo;
            char cadena[20];
            float valores[100];
            int num_datos;
            int i;

            // Se determina el tamaño del registro temporal en cuestión,
            // se hace excepción para esa métrica
            if ( (strcmp(file, ".datos_env") == 0) || (strcmp(file, ".datos_rcbd") == 0) )
                        num_datos = 1;

            // Lectura del archivo de configuración num_datos.conf
            else        num_datos = lectura_dato("../config/.num_datos.conf");

            // Se abre el registro temporal en cuestión
            if ( (archivo = fopen(file, "r")) == NULL) {
                        printf("C %s: No se puede abrir el archivo\n", file);
                        return 0;
            }

            else {
                        // Guarda el registro en el arreglo valores
                        i=0;
                        while (fgets(cadena, 20, archivo) != NULL) {
                                    valores[i] = atof(cadena);
                                    i++;
                        }

                        // Recorre los elementos del arreglo para agregar en la última posición el
                        // valor leído desde el archivo y descarta el primer valor del arreglo
                        for (i=0; i<=(num_datos-2); i++)
                                    valores[i] = valores[i+1];
                        valores[(num_datos-1)] = dato_nuevo;

            }
            fclose(archivo);

            // Apertura del archivo para sobre-escritura, si hay error sale del programa
            if ( (archivo = fopen(file, "w")) == NULL) {
                        printf("C %s: No se puede abrir el archivo\n", file);
                        return 0;
            }
            else {
                        // Escribe los datos recorridos al archivo
```

```
                        for (i=0; i<=(num_datos-1); i++)
                                fprintf(archivo, "%f\n", valores[i]);
        }
        fclose(archivo);
        return 0;
}
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
          Instituto Politécnico Nacional

   Descripción:
        Programa que determina la fecha local actual, abre el archivo especificado
        por la variable file y escribe en el archivo bitácora correspondiente los
        datos pasados a la función. Si algún dato es -1 es ignorado.
        Utilizado por todos los módulos de las métricas que controlan los registros
        temporales y las bitácoras.
*/

int escritura_bitacora(char file[20], float dato1, float dato2, float dato3) {
        FILE *archivo;
        char cadena[20];
        time_t tiempo;
        struct tm *tmPtr;

        if ( (archivo = fopen(file, "a+")) == NULL) {
                printf("C %s: No se puede abrir el archivo\n", file);
                return 0;
        }
        else {
                // Se agrega una etiqueta de tiempo a la variable cadena
                tiempo = time(NULL);
                tmPtr = localtime(&tiempo);
                strftime( cadena, 20, "%F %H:%M:%S", tmPtr );

                // Se hace una excepción en el formato de escritura en bitácora con este
                // archivo, debido al formato
                if (strcmp(file, "../logs/paq_rcbd_env.log") == 0) {
                        fprintf(archivo, "%s\n%8.2f\n%8.2f\n", cadena, dato1, dato2);
                }

                else { // Los demás archivos son tratados por igual
                        if (dato3 != -1) fprintf(archivo, "%s\t%8.2f\t%8.2f\t%8.2f\n",
                                                cadena, dato1, dato2, dato3);
                        else { // El dato3 no existe para la métrica en cuestión
                                if (dato2 != -1) fprintf(archivo, "%s\t%8.2f\t%8.2f\n",
                                                        cadena, dato1, dato2);
                                else { // El dato2 no existe para la métrica
                                        fprintf(archivo, "%s\t%8.2f\n", cadena, dato1);
                                        }
                                }
                        }
                }
        }
        fclose(archivo);
        return 0;
}
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 28 de Abril del 2008
   Lugar: Centro de Investigación en Computación
          Instituto Politécnico Nacional

   Descripción:
        Programa que escribe el dato recibido en el argumento en el archivo recibido
        de igual forma
*/
escritura_dato(char file[20], float valor) {
        FILE *archivo;

        // Apertura del archivo
        if( (archivo = fopen(file, "w")) == NULL) {
                printf("C %s: No se puede abrir el archivo\n", file);
                return 0;
        }
        else {
                fprintf(archivo, "%f\n", valor);
        }
        fclose(archivo);
        return 0;
}
```

Archivo *funciones2.h*:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
          Instituto Politécnico Nacional

   Descripción:
        Programa que lee la fecha del primer dato del archivo especificado en la
        variable archivo y la compara con la fecha actual. Si son iguales devuelve
```

```
		un cero, si son distintas devuelve un 1
		Función utilizada por el módulo que recorre las bitácoras si el día ha cambiado
*/

int comparar_fechas(char archivo[20]) {
		FILE *fp;
		char cadena[20];
		time_t tiempo;
		struct tm *tmPtr;
		int fecha_actual, fecha_leida;

		// Apertura del archivo
		if( (fp = fopen(archivo, "r")) == NULL) {
			printf("C %s: No se puede abrir el archivo\n", archivo);
			return 0;
		}
		else {
			recorre_posicion(8, fp);  // Utiliza recorre_posicion para ubicarse en el dato
			fgets(cadena, 3, fp);
			fecha_leida = atoi(cadena);
			// Lee y guarda la última fecha escrita en la bitácora
		}
		fclose(fp);

		// Determina la hora y fecha actuales
		tiempo = time(NULL);
		tmPtr = localtime(&tiempo);
		strftime( cadena, 20, "%d", tmPtr );
		fecha_actual = atoi (cadena);

		// Compara la fecha leída y la fecha actual del sistema, devuelve 0 si son iguales
		// y 1 en otro caso
		if(fecha_leida == fecha_actual) return 0;
		else return 1;
}
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Agosto del 2007
   Lugar: Centro de Investigación en Computación
		Instituto Politécnico Nacional

   Descripción:
		Programa que recorre el cursor de lectura las posiciones indicadas por la
		variable posición del archivo especificado por el apuntador al descriptor
		de archivo desc.
		Utilizada por el modulo de obtención de datos de tasa de transferencia y
		por la función comparar_fechas en este mismo archivo
*/

int recorre_posicion(int posición, FILE* desc) {
		int i, letra;

		// Mueve el cursor hasta posición-1
		for(i=0; i<=(posición-1); i++) {
			if ((letra = getc(desc)) == EOF) break;
			// Si llega al final del archivo, sale
		}
		return 0; // Todo bien
}

/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 25 de Octubre del 2007
   Lugar: Centro de Investigación en Computación
		Instituto Politécnico Nacional

   Descripción:
		Programa que copia el contenido del archivo especificado por la variable
		origen al archivo especificado por la variable destino. El parámetro tipo
		controla el tipo de copia, cuando 0 se agregan los datos al final del
		archivo destino y cuando 1 se sobre-escribe el contenido del destino.
		Utilizado por el módulo de desplazamiento de bitácoras cuando el día
		ha cambiado.
*/

int copia_archivo(char origen[20], char destino[20], int tipo) {
		FILE *archivo1, *archivo2;
		char cadena[81];
		// Abrir archivo origen
		if ( (archivo1 = fopen(origen, "r")) == NULL) {
			printf("C %s: No se puede abrir el archivo\n", origen);
			return 0;
		}
		// Abrir archivo destino
		if ( tipo == 1) { // Para sobre-escritura
			if ( (archivo2 = fopen(destino, "w")) == NULL) {
				printf("C %s: No se puede abrir el archivo\n", destino);
				return 0;
			}
		}
		else {		// Para agregar al final del archivo
			if ( (archivo2 = fopen(destino, "a+")) == NULL) {
				printf("C %s: No se puede abrir el archivo\n", destino);
				return 0;
			}
		}
		// Lee el archivo origen y escribe en el destino
		while (fgets(cadena, 80, archivo1) != NULL)
				fprintf(archivo2, "%s", cadena);
```

```
        // Cerrar archivos
        fclose(archivo1);
        fclose(archivo2);
        return 0;  // Todo bien
}
```

Archivo *get_rate.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 5 de Septiembre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
   Programa que lee la última medición de la velocidad de transferencia de datos a través
   de las interfaces de red desde el archivo .temporal y filtra los datos de entrada y
   salida y los envía a un archivo cada uno de ellos
*/

#include <stdio.h>
#include "funciones2.h"
#include "funciones.h"

main()
{
        FILE *temp, *in, *out, *config;
        int tarjt_red=1, posición=21;
        char ratein[10], rateout[10], cadena[10];
        //float valor=0;

        // Apertura de los archivos, si hay error sale del programa
        if ( (temp = fopen(".temporal", "r")) == NULL) {
                printf("C temporal: No se puede abrir el archivo\n");
                return 0;
        }
        if ( (in = fopen(".ratein", "w")) == NULL) {
                printf("C ratein: No se puede abrir el archivo\n");
                return 0;
        }
        if ( (out = fopen(".rateout", "w")) == NULL) {
                printf("C rateout: No se puede abrir el archivo\n");
                return 0;
        }
        tarjt_red = lectura_dato("../config/.num_tarjetas_red.conf");
        posición = (tarjt_red * 20);
        // Se convierte el numero de tarjetas en posiciones a recorrer

        recorre_posicion(posición, temp); // Utiliza recorre_posicion para ubicarse en el dato
        fgets(ratein, 8, temp);           // Lee el dato del archivo

        recorre_posicion(3, temp);        // Utiliza recorre_posicion para ubicarse en el dato
        fgets(rateout, 8, temp);          // Lee el dato del archivo

        fprintf(in, "%s\n", ratein);      // Se almacena RateIn en el archivo .ratein
        fprintf(out, "%s\n", rateout);    // Se almacena RateOut en el archivo .rateout

        fclose(temp);
        fclose(in);
        fclose(out);
        return 0;
}
```

Archivo *latencia.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 22 de Septiembre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
Programa que lee los valores mínimo, promedio y máximo de la latencia desde los archivos
   .max_latencia, .avg_latencia y .max_latencia respectivamente y los almacena en la ultima
   posición en el registro contenidos en tres archivos .datos_min, .datos_avg y .datos_max
   desplazando al primer valor en cada una de las listas
*/
#include "funciones.h"

main()
{
        float dato_min=0.0, dato_max=0.0, dato_avg=0.0, suma=0.0;
        int contador;

        // Apertura del archivo .max_latencia, si hay error sale del programa
        dato_max = lectura_dato(".max_latencia");

        // Apertura del archivo .avg_latencia, si hay error sale del programa
        dato_avg = lectura_dato(".avg_latencia");

        // Apertura del archivo .min_latencia, si hay error sale del programa
        dato_min = lectura_dato(".min_latencia");

        // Se agregan los datos en la ultima posición del archivo registro
        proceso_datos(".datos_max" , dato_max);
        proceso_datos(".datos_avg" , dato_avg);
        proceso_datos(".datos_min" , dato_min);
```

```
	// Almacenamiento del dato obtenido en el archivo registro
	escritura_bitacora("../logs/latencia.log", dato_min, dato_avg, dato_max);

	// Calculo del promedio para las alarmas
	contador = lectura_dato(".contador");
	suma = lectura_dato(".suma");

	if (contador < (3600 / lectura_dato("../config/.retardo.conf"))) {
		contador++;
		suma += dato_avg;
	}
	else {
		escritura_dato(".promedio_lat", suma/contador);
		suma = dato_avg;
		contador = 1;
	}
	escritura_dato(".contador", contador);
	escritura_dato(".suma", suma);
	return 0;
}
```

Archivo *lectura_config.c*:

```
/*****************************************************************************************************************
*		Autor: Héctor Julián Selley Rojas
*		Fecha: 24 de Abril del 2008
*		Lugar: Centro de Investigación en Computación
*		    Instituto Politécnico Nacional
*
*		Descripción:
*		Programa que lee todos los parámetros de configuración desde el archivo
*		de configuración monitool.conf y guarda los valores en el archivo de configuración
*		dinámico temporal correspondiente
*****************************************************************************************************************/

#include <stdio.h>
#include <string.h>
main()
{
	FILE *config, *arch;
	char cadena[81], parámetro[40][40], *ptr;
	int valor[20], i=0, max=0;

	// Apertura del archivo de configuración
	if( (config = fopen("../config/monitool.conf", "r")) == NULL) {
		printf("C: No se puede abrir el archivo de configuración\n");
		return 0;
	}
	else {	// Lectura del archivo
		while (fgets(cadena, 80, config) != NULL) {
		if(cadena[0] == '\n' || cadena[0] == '#');
		// Si es un comentario se ignora la línea
		else {
		ptr = strtok( cadena, " =" ); // Se parte la cadena donde haya espacios ó #
		if (strcmp(ptr,"retardo") == 0)
			strcpy (parámetro[i], "../config/.retardo.conf");
		if (strcmp(ptr,"num_tarjetas_red") == 0)
			strcpy (parámetro[i], "../config/.num_tarjetas_red.conf");
		if (strcmp(ptr,"borrar_bitacora") == 0)
			strcpy (parámetro[i], "../config/.borrar_bitacora.conf");
		if (strcmp(ptr,"num_datos") == 0)
			strcpy (parámetro[i], "../config/.num_datos.conf");
		if (strcmp(ptr,"cpu") == 0)
			strcpy (parámetro[i], "../config/.cpu_max");
		if (strcmp(ptr,"ram") == 0)
			strcpy (parámetro[i], "../config/.ram_max");
		if (strcmp(ptr,"latencia") == 0)
			strcpy (parámetro[i], "../config/.latencia_max");
		if (strcmp(ptr,"rate") == 0)
			strcpy (parámetro[i], "../config/.rate_max");
		if (strcmp(ptr,"paq_perdidos") == 0)
			strcpy (parámetro[i], "../config/.paq_perdidos_max");
		if (strcmp(ptr,"conexion") == 0)
			strcpy (parámetro[i], "../config/.conexion_max");
		valor[i] = atoi(strtok( NULL, " =" ));
	/* Escribir en el archivo temporal de configuración dinámica el valor leído */
		if ( (arch = fopen(parámetro[i], "w")) == NULL) {
		printf("C %s: No se puede abrir el archivo %d\n", parámetro[i], i);
		return 0;
	}
	else fprintf(arch, "%d\n", valor[i]);
	fclose(arch);
	i++;
			}
		}
	}
	fclose(config);
	return 0;
}
```

Archivo *make.sh*:

```
#!/bin/sh

## Script que compila y ubica los archivos binarios en la ubicación correspondiente
## Simplemente ejecútelo después de hacer alguna modificación de las fuentes
```

```
# Compilación de los programas que controlan las bitácoras
gcc -o ../conexiones/bitacora_con                    bitacora_con.c
gcc -o ../cpu/bitacora_cpu                           bitacora_cpu.c
gcc -o ../latencia/bitacora_lat                      bitacora_lat.c
gcc -o ../paq_perdidos/bitacora_paqp                 bitacora_paqp.c
gcc -o ../paq_rcbd_env/bitacora_paq_r_e              bitacora_paq_r_e.c
gcc -o ../ram/bitacora_ram                           bitacora_ram.c
gcc -o ../rate/bitacora_rate                         bitacora_rate.c

# Compilación del programa que genera los registros
gcc -o ../other/creacion_regs                        creacion_regs.c

# Compilación de los programas que controlan los registros temporales y guardan bitácoras
gcc -o ../conexiones/conexion                        conexion.c
gcc -o ../cpu/cpu                                    cpu.c
gcc -o ../latencia/latencia                          latencia.c
gcc -o ../paq_perdidos/paq_perdidos                  paq_perdidos.c
gcc -o ../paq_rcbd_env/paq_rcbd_env                  paq_rcbd_env.c
gcc -o ../ram/ram                                    ram.c
gcc -o ../rate/rate                                  rate.c

# Compilación de programas adicionales a los monitores
gcc -o ../rate/get_rate                              get_rate.c
gcc -o ../cpu/process                                process.c
gcc -o ../other/lectura_config                       lectura_config.c
gcc -o ../other/alarma                               alarma.c

# Compilación del programa que hace el retardo
gcc -o ../cpu/retardo                                retardo.c
cp ../cpu/retardo ../conexiones/
cp ../cpu/retardo ../latencia/
cp ../cpu/retardo ../paq_perdidos/
cp ../cpu/retardo ../paq_rcbd_env/
cp ../cpu/retardo ../rate/
cp ../cpu/retardo ../ram/
```

Archivo *paq_perdidos.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 28 de Septiembre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
            Programa que lee los últimos paquetes de prueba que se han enviado y recibido
   y los suma con los valores acumulados para determinar cuantos se han perdido en
   el trayecto del destino al origen o viceversa
*/

#include "funciones.h"
main()
{
        float nuevo_prcbd=0, nuevo_penv=0, anterior=0, suma=0;
        int contador;
//
//      Datos de Paquetes de Prueba Enviados
//
        // Se leen los dato anteriores y nuevo y se suman
        nuevo_penv = lectura_dato(".paq_env") + lectura_dato(".datos_env");
//
//      Datos Paquetes de Prueba Recibidos
//
        // Se leen los dato anteriores y nuevo y se suman
        nuevo_prcbd = lectura_dato(".paq_rcbd") + lectura_dato(".datos_rcbd");

        if ( nuevo_prcbd > nuevo_penv ) nuevo_prcbd = nuevo_penv;
        if ( nuevo_penv > 86400 ) {
                nuevo_penv  /= 10000;
                nuevo_prcbd /= 10000;
        }
        // Se agregan los datos en la ultima posición del archivo registro
        proceso_datos(".datos_env" ,  nuevo_penv);
        proceso_datos(".datos_rcbd" , nuevo_prcbd);

        // Almacenamiento del dato obtenido en el archivo registro
        escritura_bitacora("../logs/paq_perdidos.log", nuevo_penv, nuevo_prcbd, -1);

        // Calculo del promedio para las alarmas
        contador = lectura_dato(".contador");
        suma = lectura_dato(".suma");

        if (contador < (3600 / lectura_dato("../config/.retardo.conf"))) {
                contador++;
                suma += (1-(nuevo_prcbd/nuevo_penv))*100;
        }
        else {
                escritura_dato(".promedio_paqp", suma/contador);
                suma = (1-(nuevo_prcbd/nuevo_penv))*100;
                contador = 1;
        }

        escritura_dato(".contador", contador);
        escritura_dato(".suma", suma);


        return 0;
}
```

Archivo *paq_rcbd_env.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 26 de Septiembre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
                Programa que lee los paquetes que se han enviado y recibido  a través de las
   interfaces de red y las guarda en la última posición del registro de datos para
   cada uno de ellos
*/

#include "funciones.h"

main()
{
                float dato_rcbd=0.0, dato_env=0.0;

                // Apertura del archivo .paq_rcbd, si hay error sale del programa
                dato_rcbd = lectura_dato(".paq_rcbd");

                // Apertura del archivo .paq_env, si hay error sale del programa
                dato_env = lectura_dato(".paq_env");

                // Se agregan los datos en la ultima posición del archivo registro
                proceso_datos(".datos_pr" , dato_rcbd);
                proceso_datos(".datos_pe" , dato_env );

                // Almacenamiento del dato obtenido en el archivo registro
                escritura_bitacora("../logs/paq_rcbd_env.log", dato_rcbd, dato_env, -1);

                return 0;
}
```

Archivo *process.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 15 de Agosto del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
   Programa que suma el porcentaje de uso del CPU calculado y almacenado desde
   la línea de comandos a un archivo y lo almacena en un arreglo en otro archivo
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX 1000

main()
{
                FILE *entrada;
                int i=0, máximo=0;
                char chain[MAX][5];
                float valores[MAX], suma=0;

                // Apertura del archivo entrada, si hay error sale del programa

                if ( (entrada = fopen(".lista_procesos", "r")) == NULL ) {
                                printf("C: No se puede abrir el archivo");
                                //exit(1);
                                return 0;
                }
                // Lee todas las líneas del archivo y las almacena en el arreglo
                while ( fgets(chain[i], 5, entrada) != NULL ) i++;

                máximo=i;


                if ( i != 0 ) { // Si el archivo no está vacío
                                i=0;
                                while (i <= máximo) {
                                                // Solamente nos interesan las posiciones pares excepto el 0
                                                if ( (i%2 == 0) && (i != 0) ) {
                                                                suma += atof(chain[i]);
                                                }
                                                i++;
                                }
                }

                else; // El archivo está vacío, por lo tanto no se hace nada

                // Imprime el resultado, el cual será redireccionado hacia un archivo
                printf("%f\n", suma);

                fclose(entrada);

                return 0;
}
```

Archivo *ram.c*:

```
/*
*  Autor:   Héctor Julián Selley Rojas
*  Fecha:   31 de Agosto del 2007
*  Lugar:   Centro de Investigación en Computación
*                   Instituto Politécnico Nacional
*
*  Descripción:
*                   Programa que lee el resumen de utilización de memoria RAM en el sistema
*                   contenido en el archivo .salida. Calcula el porcentaje de memoria
*                   utilizada y libre y los manda a la salida estándar
*/

#include "funciones.h"

main()
{
        FILE *archivo;
        char cadena[10][20];
        int i=0, contador;
        float total, usada, libre, suma=0;
        float pusada, plibre;

        // Apertura del archivo .salida, si hay error sale del programa
        if ( (archivo = fopen(".salida", "r")) == NULL) {
                printf("C .salida: No se puede abrir el archivo\n");
                return 0;
        }

        else {
                // Lee la línea del archivo y almacena los fragmentos en el arreglo
                while (fgets(cadena[i], 12, archivo) != NULL)  i++;

                // Almacenamiento de los datos obtenidos del archivo
                total = atof(cadena[1]);
                usada = atof(cadena[2]);
                libre = atof(cadena[3]);

                // Calculo de los porcentajes
                pusada = usada / total * 100;
                plibre = libre / total * 100;

                // Envío a la salida estándar
                printf("%f\n%f\n", pusada, plibre);
        }
        fclose(archivo);

        // Almacenamiento del dato obtenido en el archivo registro
        escritura_bitacora("../logs/ram.log", pusada, plibre, -1);

        // Calculo del promedio para las alarmas
        contador = lectura_dato(".contador");
        suma = lectura_dato(".suma");

        if (contador < (3600 / lectura_dato("../config/.retardo.conf"))) {
                contador++;
                suma += pusada;
        }

        else {
                escritura_dato(".promedio_ram", suma/contador);
                suma = pusada;
                contador = 1;
        }

        escritura_dato(".contador", contador);
        escritura_dato(".suma", suma);

        return 0;
}
```

Archivo *rate.c*:

```
/*
   Autor: Héctor Julián Selley Rojas
   Fecha: 6 de Septiembre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
   Programa que lee los últimos valores de tasa de transferencia desde los archivos
   .ratein y .rateout y los almacena en la ultima posición de un registro contenido
   en los archivos .datos_in y .datos_out y escribe las bitácoras
*/
#include "funciones.h"

main()
{
        float dato_in=0.0, dato_out=0.0, suma=0.0;
        int contador;

        // Apertura del archivo .ratein, si hay error sale del programa
        dato_in = lectura_dato(".ratein");

        // Apertura del archivo .rateout, si hay error sale del programa
        dato_out = lectura_dato(".rateout");
```

```
        // Se agregan los datos en la ultima posición del archivo registro
        proceso_datos(".datos_in" , dato_in);
        proceso_datos(".datos_out", dato_out);

        // Almacenamiento del dato obtenido en el archivo registro
        escritura_bitacora("../logs/rate.log", dato_in, dato_out, -1);

        // Calculo del promedio para las alarmas
        contador = lectura_dato(".contador");
        suma = lectura_dato(".suma");

        if (contador < (3600 / lectura_dato("../config/.retardo.conf"))) {
                contador++;
                suma += ((dato_in + dato_out)/2);
        }
        else {
                escritura_dato(".promedio_rate", suma/contador);
                suma = ((dato_in + dato_out)/2);
                contador = 1;
        }
        escritura_dato(".contador", contador);
        escritura_dato(".suma", suma);
        return 0;
}
```

Archivo *retardo.c*:

```
/* Autor: Héctor Julián Selley Rojas
   Fecha: 10 de Septiembre del 2007
   Lugar: Centro de Investigación en Computación
                Instituto Politécnico Nacional

   Descripción:
   Programa que ejecuta un tiempo de retardo definido en el archivo
   de configuración monitool/config/retardo
*/
#include <stdio.h>
#include "funciones.h"
main()
{
        FILE *fp;
        char cadena[20];
        int dato=5;
        dato = lectura_dato("../config/.retardo.conf");
        // Conversión a microsegundos
        dato *= 1000000;
        usleep(dato);
        return 0;
}
```

Archivo *conexion.sh*:

```
#!/bin/sh -e
#
# Script que determina el número de sockets/conexiones a Internet y en el sistema
# Unix y los cuenta y redirecciona a dos archivos y recorre las bitácoras si ya ha
# pasado un día de monitoreo
#

TRUE=1
while [ $TRUE -eq 1 ]
do
#
# Ciclo Infinito
#
/bin/netstat -n > .salida
/bin/cat .salida | /bin/grep tcp | /usr/bin/wc -l > .internet
/bin/cat .salida | /bin/grep unix | /usr/bin/wc -l > .unix
./conexion
./retardo
./bitacora_con
done
```

Archivo *cpu.sh*:

```
#!/bin/sh -e
#
# Script que determina la carga del CPU, ejecuta la suma de la carga individual de
# los procesos y la redirecciona a un archivo donde se almacena el resultado final
# y recorre las bitácoras si ya ha pasado un día de monitoreo
#
TRUE=1
while [ $TRUE -eq 1 ]
do
#
# Ciclo Infinito
#
ps -eo %C > .lista_procesos
./process > .cpu
./cpu
./retardo
./bitacora_cpu
#
done
```

Archivo *creación_regs.sh*:

```
#!/bin/sh -e
#
# Script que genera los registros temporales de datos para todas las métricas
```

```
./creacion_regs
cat .registro > ../cpu/.datos_cpu
cat .registro > ../rate/.datos_in
cat .registro > ../rate/.datos_out
cat .registro > ../conexiones/.datos_internet
cat .registro > ../conexiones/.datos_unix
cat .registro > ../latencia/.datos_min
cat .registro > ../latencia/.datos_avg
cat .registro > ../latencia/.datos_max
cat .registro > ../paq_rcbd_env/.datos_pr
cat .registro > ../paq_rcbd_env/.datos_pe
#
```

Archivo *alarma.sh*:

```
#!/bin/sh -e
#
TRUE=1
while [ $TRUE -eq 1 ]
do
#
# Ciclo Infinito
#
./alarma
./retardo
#
done
```

Archivo *paq_perdidos.sh*:

```
#!/bin/sh -e
#
# Script que determina la latencia en la red, se determina mediante el calculo de un
# promedio entre la latencia unitaria de varios hosts y recorre las bitácoras si ya ha
# pasado un día de monitoreo
#

echo 0 > .datos_env
echo 0 > .datos_rcbd
TRUE=1
while [ $TRUE -eq 1 ]
do
#
# Ciclo Infinito
#
cat .salida | grep "Echos sent" > .paq_env
cat .salida | grep "Echo Replies received" > .paq_rcbd
./paq_perdidos
./retardo
./bitacora_paqp
done
```

Archivo *paq_rcbd_env.sh*:

```
#!/bin/sh -e
#
# Script que determina la latencia en la red, se determina mediante el calculo de un
# promedio entre la latencia unitaria de varios hosts y recorre las bitácoras si ya ha
# pasado un día de monitoreo
#

TRUE=1
while [ $TRUE -eq 1 ]
do
#
# Ciclo Infinito
#
netstat -s > .salida
cat .salida | grep "total packets received" > .paq_rcbd
#cat .salida | grep "total de paquetes recibidos" > .paq_rcbd
cat .salida | grep "sent out" > .paq_env
#cat .salida | grep "peticiones enviadas" > .paq_env
./paq_rcbd_env
./retardo
./bitacora_paq_r_e
done
```

Archivo *ram.sh*:

```
#!/bin/sh -e
#
# Script que genera determina el uso de memoria RAM, determina el porcentaje de memoria
# utilizada y libre y la manda al archivo .dataram y recorre las bitácoras si ya ha
# pasado un día de monitoreo
#
TRUE=1
while [ $TRUE -eq 1 ]
do
#
# Ciclo Infinito
#
/usr/bin/free -k | /bin/grep Mem > .salida
./ram > .datos_ram
./retardo
./bitacora_ram
done
```

Archivo *ram.sh*:

```
#!/bin/sh -e
#
# Script que determina la tasa de transferencia de datos a través de las interfaces de red
# y recorre las bitácoras si ya ha pasado un día de monitoreo
#

TRUE=1
while [ $TRUE -eq 1 ]
do
#
# Ciclo Infinito
#
/usr/bin/pkill -f ifstat
/usr/bin/tail -1 .salida > .temporal
/usr/bin/ifstat -T -z | cat > .salida &
./get_rate
./rate
./retardo
./bitacora_rate
done
```

Archivo *iniciar.sh*:

```
#!/bin/sh -e
#
# Script que inicializa los archivos temporales contadores y promedios
#
echo 0 > ../cpu/.promedio_cpu
echo 0 > ../cpu/.contador
echo 0 > ../cpu/.suma
echo 0 > ../conexiones/.promedio_con
echo 0 > ../conexiones/.contador
echo 0 > ../conexiones/.suma
echo 0 > ../latencia/.promedio_lat
echo 0 > ../latencia/.contador
echo 0 > ../latencia/.suma
echo 0 > ../rate/.promedio_rate
echo 0 > ../rate/.contador
echo 0 > ../rate/.suma
echo 0 > ../ram/.promedio_ram
echo 0 > ../ram/.contador
echo 0 > ../ram/.suma
echo 0 > ../paq_perdidos/.promedio_paqp
echo 0 > ../paq_perdidos/.contador
echo 0 > ../paq_perdidos/.suma
#
```

Archivo *monitool_start.sh*:

```
#!/bin/sh –e
#
# Meta-script que ejecuta todos los servicios de monitoreo
# asignándoles  una baja prioridad de ejecución
#
#cd /var/www/monitool/init
echo Iniciando Procesos Preliminares .....
cd ../other
/usr/bin/nice -100 sh creacion_regs.sh
/usr/bin/nice -100 ./lectura_config
echo [ OK ]
#
echo Iniciando monitor de CPU .....
cd ../cpu
/usr/bin/nice -100 sh cpu.sh &
echo [ OK ]
#
echo Iniciando monitor de RAM .....
cd ../ram
/usr/bin/nice -100 sh ram.sh &
echo [ OK ]
#
echo Iniciando Monitor de Tasa de Transferencia .....
cd ../rate
/usr/bin/nice -100 sh rate.sh &
echo [ OK ]
#
echo Iniciando Monitor de Conexiones .....
cd ../conexiones
/usr/bin/nice -100 sh conexion.sh &
echo [ OK ]
#
echo Iniciando Monitor de Latencia .....
cd ../latencia
/usr/bin/nice -100 sh latencia.sh &
echo [ OK ]
#
echo Iniciando Monitor de Paquetes Recibidos y Enviados .....
cd ../paq_rcbd_env
/usr/bin/nice -100 sh paq_rcbd_env.sh &
echo [ OK ]
#
echo Iniciando Monitor de Paquetes Perdidos .....
cd ../paq_perdidos
/usr/bin/nice -100 sh paq_perdidos.sh &
echo [ OK ]
#
echo Iniciando Sistema de Alarmas .....
```

```
cd ../other
/usr/bin/nice -100 sh iniciar.sh
/usr/bin/nice -100 sh alarma.sh &
echo [ OK ]
#
echo
echo "MoniTool is running :)"
#cd ..
#ps -f -C sh > pid.txt
echo
#
```

Archivo *monitool_stop.sh:*

```
#!/bin/sh -e
#
# Script que detiene todos los servicios de monitoreo
#
echo Finalizando Monitor CPU ...
pkill -f cpu.sh
echo [ OK ]
#
echo Finalizando Monitor RAM ...
pkill -f ram.sh
echo [ OK ]
#
echo Finalizando Monitor de Tasa de Transferencia .....
pkill -f rate.sh
echo [ OK ]
#
echo Finalizando Monitor de Conexiones .....
pkill -f conexion.sh
echo [ OK ]
#
echo Finalizando Monitor de Latencia .....
pkill -f latencia.sh
echo [ OK ]
#
echo Finalizando Monitor de Paquetes Recibidos y Enviados .....
pkill -f paq_rcbd_env.sh
echo [ OK ]
#
echo Finalizando Monitor de Paquetes Perdidos .....
pkill -f paq_perdidos.sh
echo [ OK ]
#
# Procesos Adicionales
pkill -f ifstat
pkill -f fping
pkill -f retardo
pkill -f alarma.sh
echo
#
```

Archivo *acceso.php*:

```
<?

/*****************************************************************************************
*       Declaración y definición de acceso_archivo
*
*       $arch           := Nombre del archivo a acceder
*       $tipo_dato      := 0 -> conversión de los datos a entero
*                          1 -> conversión de los datos a flotante
*                          2 -> sin conversión, queda como string
*
*       Descripción:    La función accesa al archivo especificado en la variable
*                       $arch, lee todos los datos encontrados en el, los convierte
*                       dependiendo de $tipo_dato y los almacena en un arreglo que
*                       será devuelto posteriormente. Esta función es utilizada por los
*                       módulos generadores de las gráficas para cada una de las
*                       métricas, accediendo al registro temporal que será graficado.
*
*       $valores    := Arreglo que devuelve la función
*****************************************************************************************/

function acceso_archivo($file, $tipo_dato) {
        // Abre el archivo $file y guarda el registro en el arreglo $valores
        $i=0;
        $fd = fopen($file, "r")
                or die ("PHP $file: No se puede leer el archivo");

        // Lectura del archivo
        while (!feof($fd)) {
                $variable = fgets($fd, 1024);

                // Conversión de string al tipo especificado en $tipo_dato
                if ($tipo_dato == 0) $variable *= 1;
                if ($tipo_dato == 1) $variable *= 1.0;
                else ; // Se deja como string

                //echo "$variable <BR>";
                $valores[$i] = $variable;
                $i++;
        }
        fclose ($fd);

        // Borra el ultimo dato dado que contiene un valor nulo que se guarda como 0
```

```
		unset($valores[$i-1]);

		return $valores;
}

/*********************************************************************************************
*		Declaración y definición de ultima_fecha()
*
*		$file			:= Nombre del archivo a acceder
*
*		Descripción:		La función accesa al archivo especificado en la variable
*					$file, lee todos los datos encontrados en el, y devuelve la
*					ultima línea recortada encontrada en ese archivo.
*
*		$retorno	:= Cadena recortada a imprimir
*********************************************************************************************/

function ultima_fecha($file) {

		// Lectura de los datos del archivo
		$variable = acceso_archivo($file, 2);

		// Retorno de cadena del último dato del arreglo
		$retorno = "Ultima actualización:  " . substr($variable[sizeof($variable)-1], 0, 20);

		return $retorno;
}

/*********************************************************************************************
*		Declaración y definición de fechas()
*
*		$file			:= Nombre del archivo a acceder
*
*		Descripción:		La función accesa al archivo especificado en la variable
*					$file y lee la cantidad de datos desde el archivo num_datos.conf.
*					Posteriormente lee las ultimas fechas de la bitácora y devuelve
*					en un arreglo.
*
*		$datos			:= Arreglo que contiene las ultimas fechas a devolver
*********************************************************************************************/

function fechas($file) {

		$datos = array();

		$arreglo = acceso_archivo("../config/.num_datos.conf", 0);
		$num_datos = $arreglo[sizeof($arreglo)-1];

		// Lectura de los datos del archivo
		$salida = acceso_archivo($file, 2);

		for ($i=0; $i<=($num_datos-1); $i++) {
			$cadena = $salida[(sizeof($salida))-$num_datos+$i] ."<BR>";
			if(($i%5) == 0 || $i == ($num_datos-1)) $datos[$i] = substr($cadena, 11, 8);
			else $datos[$i] = "";
		}

		return $datos;
}

/*********************************************************************************************
*		Declaración y definición de promedio()
*
*		$archivo	:= Nombre del archivo a acceder
*
*		Descripción:		La función accesa al archivo especificado en la variable
*					$archivo, calcula el promedio de los datos leídos y lo
*					devuelve
*********************************************************************************************/

function promedio($archivo) {

		$valores = acceso_archivo($archivo, 2);

		$indice = 1;

		for($i=0; $i<=sizeof($valores); $i++) {
			$suma += $valores[$i];
			if ($valores[$i] != 0) $indice++;
		}
		if (($indice-1) > 0) return ($suma / ($indice-1));
		else return 0;
}


/*********************************************************************************************
*		Declaración y definición de estimacion_indice()
*
*		Descripción:		La función lee los valores máximos configurados por el usuario
*					desde los archivos de configuración dinámicos, así como los
*					valores contenidos en los registros temporales. Se calcula un
*					promedio de estos últimos datos y se comparan con los
*					máximos, por cada valor sobrepasado se incrementa el índice, al
*					que le corresponde una leyenda descriptiva
*********************************************************************************************/

function estimacion_indice() {

		$indice = 0;
		$cpu	= 0;
```

```
$rate       = 0;
$con        = 0;
$lat        = 0;
$paqp       = 0;
$mem        = 0;

// Lectura de los valores máximos desde los archivos de configuración dinámica
$cpu_max = acceso_archivo("config/.cpu_max", 1);
$rate_max = acceso_archivo("config/.rate_max", 1);
$conexion_max = acceso_archivo("config/.conexion_max", 1);
$latencia_max = acceso_archivo("config/.latencia_max", 1);
$pperdidos_max = acceso_archivo("config/.paq_perdidos_max", 1);
$ram_max = acceso_archivo("config/.ram_max", 1);

// Lectura de los datos de ram desde el registro temporal, no se calcula
// el promedio dado que solo hay un valor en el registro temporal
$ram = acceso_archivo("ram/.datos_ram", 1);

// Calculo del indice de desempeño
if (promedio("cpu/.datos_cpu") >= $cpu_max[0]) $cpu=1;
if (promedio("rate/.datos_in") >= $rate_max[0] ||
                promedio("rate/.datos_out") >= $rate_max[0]) $rate=1;
if (promedio("conexiones/.datos_internet") >= $conexion_max[0]) $con=1;
if (promedio("latencia/.datos_avg") >= $latencia_max[0]) $lat=1;
if (((1 - promedio("paq_perdidos/.datos_rcbd") /
        promedio("paq_perdidos/.datos_env")) *100)
                >= $pperdidos_max[0]) $paqp=1;
if ($ram[0] >= $ram_max[0]) $mem=1;

$indice = $cpu + $rate + $con + $lat + $paqp + $mem;
// Los casos posibles
switch ($indice) {
        case 0: $leyenda = "Bien</H3> 0 Niveles Rebasados";
                break;
        case 1: $leyenda = "Normal</H3> 1 Nivel Rebasado: ";
                break;
        case 2: $leyenda = "Regular</H3> 2 Niveles Rebasados: ";
                break;
        case 3: $leyenda = "Atención</H3> 3 Niveles Rebasados: ";
                break;
        case 4: $leyenda = "Precaución</H3> 4 Niveles Rebasados: ";
                break;
        case 5: $leyenda = "Alerta</H3> 5 Niveles Rebasados: ";
                break;
        case 6: $leyenda = "Critico</H4> 6 Niveles Rebasados: ";
                break;
        case 7: $leyenda = "Pánico</H3> 7 Niveles Rebasados: ";
                break;
        default:$leyenda = "Indeterminado";
                break;
        }
        if($cpu == 1)       $leyenda = $leyenda . "cpu ";
        if($rate == 1)      $leyenda .= "tasa transferencia ";
        if($con == 1)       $leyenda .= "conexiones ";
        if($lat == 1)       $leyenda .= "latencia ";
        if($paqp == 1)      $leyenda .= "paquetes perdidos ";
        if($mem == 1)       $leyenda .= "ram";

        // Devolución del índice estimado
        return $leyenda;
}
?>
```

Archivo *ver_bitacora:*

```
<?php

        include "acceso.php";

        switch ($_GET['archivo']) {

                // bitácoras CPU
                case "../logs/cpu.log":
                        $siguiente  = "../logs/cpu.log.0";
                        $anterior   = "../logs/cpu.log";
                        $leyenda = "CPU";
                        $num_datos = 1;
                        break;
                case "../logs/cpu.log.0":
                        $siguiente  = "../logs/cpu.log.1";
                        $anterior   = "../logs/cpu.log";
                        $leyenda = "CPU";
                        $num_datos = 1;
                        break;
                case "../logs/cpu.log.1":
                        $siguiente  = "../logs/cpu.log.2";
                        $anterior   = "../logs/cpu.log.0";
                        $leyenda = "CPU";
                        $num_datos = 1;
                        break;
                case "../logs/cpu.log.2":
                        $siguiente  = "../logs/cpu.log.3";
                        $anterior   = "../logs/cpu.log.1";
                        $leyenda = "CPU";
                        $num_datos = 1;
                        break;
                case "../logs/cpu.log.3":
                        $siguiente  = "../logs/cpu.log.4";
                        $anterior   = "../logs/cpu.log.2";
                        $leyenda = "CPU";
```

```
                $num_datos = 1;
                break;
case "../logs/cpu.log.4":
                $siguiente  = "../logs/cpu.log";
                $anterior   = "../logs/cpu.log.3";
                $leyenda = "CPU";
                $num_datos = 1;
                break;

// bitácoras Rate
case "../logs/rate.log":
                $siguiente  = "../logs/rate.log.0";
                $anterior   = "../logs/rate.log";
                $leyenda = "Tasa de Transferencia";
                $num_datos = 2;
                break;
case "../logs/rate.log.0":
                $siguiente  = "../logs/rate.log.1";
                $anterior   = "../logs/rate.log";
                $leyenda = "Tasa de Transferencia";
                $num_datos = 2;
                break;
case "../logs/rate.log.1":
                $siguiente  = "../logs/rate.log.2";
                $anterior   = "../logs/rate.log.0";
                $leyenda = "Tasa de Transferencia";
                $num_datos = 2;
                break;
case "../logs/rate.log.2":
                $siguiente  = "../logs/rate.log.3";
                $anterior   = "../logs/rate.log.1";
                $leyenda = "Tasa de Transferencia";
                $num_datos = 2;
                break;
case "../logs/rate.log.3":
                $siguiente  = "../logs/rate.log.4";
                $anterior   = "../logs/rate.log.2";
                $leyenda = "Tasa de Transferencia";
                $num_datos = 2;
                break;
case "../logs/rate.log.4":
                $siguiente  = "../logs/rate.log";
                $anterior   = "../logs/rate.log.3";
                $leyenda = "Tasa de Transferencia";
                $num_datos = 2;
                break;

// bitácoras Latencia
case "../logs/latencia.log":
                $siguiente  = "../logs/latencia.log.0";
                $anterior   = "../logs/latencia.log";
                $leyenda = "Latencia";
                $num_datos = 3;
                break;
case "../logs/latencia.log.0":
                $siguiente  = "../logs/latencia.log.1";
                $anterior   = "../logs/latencia.log";
                $leyenda = "Latencia";
                $num_datos = 3;
                break;
case "../logs/latencia.log.1":
                $siguiente  = "../logs/latencia.log.2";
                $anterior   = "../logs/latencia.log.0";

                $leyenda = "Latencia";
                $num_datos = 3;
                break;
case "../logs/latencia.log.2":
                $siguiente  = "../logs/latencia.log.3";
                $anterior   = "../logs/latencia.log.1";
                $leyenda = "Latencia";
                $num_datos = 3;
                break;
case "../logs/latencia.log.3":
                $siguiente  = "../logs/latencia.log.4";
                $anterior   = "../logs/latencia.log.2";
                $leyenda = "Latencia";
                $num_datos = 3;
                break;
case "../logs/latencia.log.4":
                $siguiente  = "../logs/latencia.log";
                $anterior   = "../logs/latencia.log.3";
                $leyenda = "Latencia";
                $num_datos = 3;
                break;

// bitácoras Conexiones
case "../logs/conexion.log":
                $siguiente  = "../logs/conexion.log.0";
                $anterior   = "../logs/conexion.log";
                $leyenda = "Conexiones";
                $num_datos = 2;
                break;
case "../logs/conexion.log.0":
                $siguiente  = "../logs/conexion.log.1";
                $anterior   = "../logs/conexion.log";
                $leyenda = "Conexiones";
                $num_datos = 2;
                break;
case "../logs/conexion.log.1":
```

```
                $siguiente  = "../logs/conexion.log.2";
                $anterior   = "../logs/conexion.log.0";
                $leyenda = "Conexiones";
                $num_datos = 2;
                break;
case "../logs/conexion.log.2":
                $siguiente  = "../logs/conexion.log.3";
                $anterior   = "../logs/conexion.log.1";
                $leyenda = "Conexiones";
                $num_datos = 2;
                break;
case "../logs/conexion.log.3":
                $siguiente  = "../logs/conexion.log.4";
                $anterior   = "../logs/conexion.log.2";

                $leyenda = "Conexiones";
                $num_datos = 2;
                break;
case "../logs/conexion.log.4":
                $siguiente  = "../logs/conexion.log";
                $anterior   = "../logs/conexion.log.3";
                $leyenda = "Conexiones";
                $num_datos = 2;
                break;

// bitácoras Paquetes Recibidos y Enviados
case "../logs/paq_rcbd_env.log":
                $siguiente  = "../logs/paq_rcbd_env.log.0";
                $anterior   = "../logs/paq_rcbd_env.log";
                $leyenda = "Paquetes Recibidos y Enviados";
                $num_datos = 2;
                break;
case "../logs/paq_rcbd_env.log.0":
                $siguiente  = "../logs/paq_rcbd_env.log.1";
                $anterior   = "../logs/paq_rcbd_env.log";
                $leyenda = "Paquetes Recibidos y Enviados";
                $num_datos = 2;
                break;
case "../logs/paq_rcbd_env.log.1":
                $siguiente  = "../logs/paq_rcbd_env.log.2";
                $anterior   = "../logs/paq_rcbd_env.log.0";
                $leyenda = "Paquetes Recibidos y Enviados";
                $num_datos = 2;
                break;
case "../logs/paq_rcbd_env.log.2":
                $siguiente  = "../logs/paq_rcbd_env.log.3";
                $anterior   = "../logs/paq_rcbd_env.log.1";
                $leyenda = "Paquetes Recibidos y Enviados";
                $num_datos = 2;
                break;
case "../logs/paq_rcbd_env.log.3":
                $siguiente  = "../logs/paq_rcbd_env.log.4";
                $anterior   = "../logs/paq_rcbd_env.log.2";
                $leyenda = "Paquetes Recibidos y Enviados";
                $num_datos = 2;
                break;
case "../logs/paq_rcbd_env.log.4":
                $siguiente  = "../logs/paq_rcbd_env.log";
                $anterior   = "../logs/paq_rcbd_env.log.3";
                $leyenda = "Paquetes Recibidos y Enviados";
                $num_datos = 2;
                break;

// bitácoras Paquetes Perdidos
case "../logs/paq_perdidos.log":
                $siguiente  = "../logs/paq_perdidos.log.0";
                $anterior   = "../logs/paq_perdidos.log";
                $leyenda = "Paquetes Perdidos";
                $num_datos = 2;
                break;
case "../logs/paq_perdidos.log.0":
                $siguiente  = "../logs/paq_perdidos.log.1";
                $anterior   = "../logs/paq_perdidos.log";
                $leyenda = "Paquetes Perdidos";
                $num_datos = 2;
                break;
case "../logs/paq_perdidos.log.1":
                $siguiente  = "../logs/paq_perdidos.log.2";
                $anterior   = "../logs/paq_perdidos.log.0";
                $leyenda = "Paquetes Perdidos";
                $num_datos = 2;
                break;
case "../logs/paq_perdidos.log.2":
                $siguiente  = "../logs/paq_perdidos.log.3";
                $anterior   = "../logs/paq_perdidos.log.1";
                $leyenda = "Paquetes Perdidos";
                $num_datos = 2;
                break;
case "../logs/paq_perdidos.log.3":
                $siguiente  = "../logs/paq_perdidos.log.4";
                $anterior   = "../logs/paq_perdidos.log.2";
                $leyenda = "Paquetes Perdidos";
                $num_datos = 2;
                break;
case "../logs/paq_perdidos.log.4":
                $siguiente  = "../logs/paq_perdidos.log";
                $anterior   = "../logs/paq_perdidos.log.3";
                $leyenda = "Paquetes Perdidos";
                $num_datos = 2;
                break;
```

```
                // bitácoras RAM
                case "../logs/ram.log":
                        $siguiente  = "../logs/ram.log.0";
                        $anterior   = "../logs/ram.log";
                        $leyenda = "RAM";
                        $num_datos = 2;
                        break;
                case "../logs/ram.log.0":
                        $siguiente  = "../logs/ram.log.1";
                        $anterior   = "../logs/ram.log";
                        $leyenda = "RAM";
                        $num_datos = 2;
                        break;
                case "../logs/ram.log.1":
                        $siguiente  = "../logs/ram.log.2";
                        $anterior   = "../logs/ram.log.0";
                        $leyenda = "RAM";
                        $num_datos = 2;
                        break;
                case "../logs/ram.log.2":
                        $siguiente  = "../logs/ram.log.3";
                        $anterior   = "../logs/ram.log.1";
                        $leyenda = "RAM";
                        $num_datos = 2;
                        break;
                case "../logs/ram.log.3":
                        $siguiente  = "../logs/ram.log.4";
                        $anterior   = "../logs/ram.log.2";
                        $leyenda = "RAM";
                        $num_datos = 2;
                        break;
                case "../logs/ram.log.4":
                        $siguiente  = "../logs/ram.log";
                        $anterior   = "../logs/ram.log.3";
                        $leyenda = "RAM";
                        $num_datos = 2;
                        break;

                default:

                        break;
        }

        //echo $_GET['archivo'];

        $datos = acceso_archivo($_GET['archivo'], 2);
?>

<HMTL>
<HEAD>
<TITLE> MoniTool: Monitoreo de Servidores </TITLE>
</HEAD>
<BODY BACKGROUND="../img/pared.gif" BGCOLOR="">
<IMG SRC="../img/MoniTool-tiny-mini.png" BORDER="0" ALIGN="RIGHT">
<H1><font face="Arial" color="#3399FF"><P ALIGN="CENTER">
MoniTool: Despliegue de bitácoras </P></font></H1>
<A HREF="../monitool.php"><P ALIGN="LEFT">Pantalla Principal MoniTool</P></A>
<A HREF="menu.php"><P ALIGN="LEFT">Menú Despliegue de bitácoras</P></A>
<HR><BR>
<FONT SIZE=4 COLOR="2767f9" FACE="Arial, Verdana">
<BODY>
<TABLE BORDER=1>
<?php
        echo "<BIG>Despliegue de los archivos log (bitácoras) de $leyenda</BIG>";
        echo "(" . $_GET['archivo'] . ")<BR><BR>";
        echo "<TR>";
        echo "<TH> Fecha y Hora</TH>";
        switch ($leyenda) {
                case "CPU":
                        echo "<TH> %CPU </TH>";
                        break;
                case "Tasa de Transferencia":
                        echo "<TH> Rate IN (kb/s) </TH>";
                        echo "<TH> Rate OUT (kb/s) </TH>";
                        break;
                case "Latencia":
                        echo "<TH> Mínima (ms) </TH>";
                        echo "<TH> Promedio (ms) </TH>";
                        echo "<TH> Máxima (ms) </TH>";
                        break;
                case "Conexiones":
                        echo "<TH> # Tipo Internet </TH>";
                        echo "<TH> # Tipo Unix </TH>";
                        break;
                case "Paquetes Recibidos y Enviados":
                        echo "<TH> # Recibidos </TH>";
                        echo "<TH> # Enviados </TH>";
                        break;
                case "Paquetes Perdidos":
                        echo "<TH> # Enviados </TH>";
                        echo "<TH> # Recibidos </TH>";
                        break;
                case "RAM":
                        echo "<TH> % Usada </TH>";
                        echo "<TH> % Libre </TH>";
                        break;
                default:
                        break;
        }
```

```
		echo "</TR>";

		//foreach ($datos as $d) echo "$d <BR>";

		// Para el despliegue de datos se hace una excepción con esta
		// métrica debido al formato
		if ($leyenda == "Paquetes Recibidos y Enviados") {
			for($i=0; $i<=(sizeof($datos)+1); $i+=3) {
				echo "<TR> <TH> $datos[$i] </TH>";
				echo "<TH>" . $datos[$i+1] . "</TH>";
				echo "<TH>" . $datos[$i+2] . "</TH>";
				echo "</TR>";
			}
		}

		// Las demás métricas son tratadas por igual
		else {
			for($i=0; $i<=(sizeof($datos)+1); $i++) {
				echo "<TR> <TH>" . substr($datos[$i], 0, 20) . "</TH>";
				echo "<TH>" . substr($datos[$i], 20, 9) . "</TH>";
				if ($num_datos >= 2)
					echo "<TH>" . substr($datos[$i], 29, 9) . "</TH>";
				if ($num_datos == 3)
					echo "<TH>" . substr($datos[$i], 38, 9) . "</TH>";
				echo "</TR>";
			}
		}

?>
</TABLE>
</FONT> <BR>

<A HREF="ver_bitacora.php?archivo=<?php echo $anterior; ?>">
		<P ALIGN="LEFT">Ver bitácoras de <?php echo $leyenda; ?> Anteriores</P></A>
<A HREF="ver_bitacora.php?archivo=<?php echo $siguiente;?>">
		<P ALIGN="LEFT">Ver bitácoras de <?php echo $leyenda; ?> Siguientes</P></A>
<HR>

<FONT FACE="Arial"><P ALIGN="RIGHT">
MoniTool ver 1.0 <BR>
Desarrollado por Héctor Selley <BR>
CIC - IPN - México @ 2007
</P></FONT>
<HR>
</BODY>
</HMTL>
```

Archivo *ver_bitacora.php:*

```
<?php

/*************************************************************************************************
*		Autor:		Héctor Julián Selley Rojas
*		Lugar:		Centro de Investigación en Computación
*				Instituto Politécnico Nacional
*
*		Descripción:	Script que lee los datos almacenados en el archivo registro y crea
*				la gráfica correspondiente
*************************************************************************************************/

		include ("jpgraph.php");
		include ("jpgraph_line.php");
		include ("../other/acceso.php");

//		Lectura de los datos del archivo

		// Datos de conexiones a Internet
		$datos_int = acceso_archivo("../conexiones/.datos_internet", 0);

		// Datos de conexiones Unix
		$datos_unix = acceso_archivo("../conexiones/.datos_unix", 0);
//
//		Creación de la gráfica
//
		// Creación de la instancia gráfica
		$grafico = new graph(750,250,"auto");
		$grafico -> img -> SetMargin(50,20,20,40);
		$grafico -> img -> SetAntiAliasing();
		$grafico -> SetScale("textlin");
		$grafico -> SetFrame(false);
		$grafico -> title -> Set("Numero de sockets/conexiones establecidas");
		$grafico -> title -> SetFont(FF_VERDANA,FS_BOLD);
		$grafico -> legend -> Pos(0.1, 0.1, "left" ,"center");
		$grafico -> xaxis -> title -> Set("Tiempo [HH:MM:SS]");
		$grafico -> yaxis -> title -> Set("#");
		$grafico -> xgrid -> Show();

		// Leyendas del eje X
		$datax = fechas("../logs/conexion.log");
		$grafico -> xaxis -> SetTickLabels($datax);
		$grafico -> xaxis -> SetFont(FF_ARIAL,FS_NORMAL,8);
		$grafico -> xaxis -> SetLabelAngle(10);

		// Datos de conexiones a Internet
		$p1 = new LinePlot($datos_int);
		$p1 -> SetColor("darksalmon");
		$p1 -> SetLegend("Internet");
```

```
        $p1 -> SetCenter();
        $grafico -> Add($p1);

        // Datos de conexiones Unix
        $p2 = new LinePlot($datos_unix);
        $p2 -> SetColor("chartreuse3");
        $p2 -> SetLegend("Unix");
        $p2 -> SetCenter();
        $grafico -> Add($p2);

        $grafico->Stroke();

?>
```

Archivo *gcpu.php:*

```
<?php

/*****************************************************************************************
*       Autor:          Héctor Julián Selley Rojas
*       Lugar:          Centro de Investigación en Computación
*                       Instituto Politécnico Nacional
*
*       Descripción:    Script que lee los datos almacenados en el archivo
*                       registro y crea la gráfica correspondiente
*****************************************************************************************/

        include ("jpgraph.php");
        include ("jpgraph_line.php");
        include ("../other/acceso.php");


        // Lectura de los datos del archivo
        $porcentajes = acceso_archivo("../cpu/.datos_cpu", 1);

//
//      Creación de la gráfica
//

        // Creación de la instancia gráfica
        $grafico = new Graph(750,250,"auto");
        $grafico -> SetScale("textlin");
        $grafico -> SetMarginColor('black');
        $grafico -> SetFrame(false);
        $grafico -> SetMargin(30,50,30,30);
        $grafico -> img -> SetAntiAliasing();
        $grafico -> img -> SetMargin(50,20,20,40);
        $grafico -> title -> SetFont(FF_VERDANA,FS_BOLD);
        $grafico -> title -> Set("Porcentaje de uso CPU");
        $grafico -> xaxis -> title -> Set("Tiempo [HH:MM:SS]");
        $grafico -> yaxis -> title -> Set("%CPU");
        $grafico -> xgrid -> Show();

        // Leyendas del eje X
        $datax = fechas("../logs/cpu.log");
        $grafico -> xaxis -> SetTickLabels($datax);
        $grafico -> xaxis -> SetFont(FF_ARIAL,FS_NORMAL,8);
        $grafico -> xaxis -> SetLabelAngle(10);

        // Línea de gráfica y características
        $linea = new LinePlot($porcentajes);
        $linea -> SetColor("red");

        // Añadir la línea de gráfica
        $grafico -> Add($linea);

        // Despliegue
        $grafico -> Stroke();

?>
```

Archivo *glatencia.php:*

```
<?php

/*****************************************************************************************
*       Autor:          Héctor Julián Selley Rojas
*       Lugar:          Centro de Investigación en Computación
*                       Instituto Politécnico Nacional
*
*       Descripción:    Script que lee los datos almacenados en el archivo
*                       registro y crea la gráfica correspondiente
*****************************************************************************************/

        include ("jpgraph.php");
        include ("jpgraph_line.php");
        include ("../other/acceso.php");


//      Lectura de los datos del archivo

        // Datos de Latencia Mínima
        $datos_min = acceso_archivo("../latencia/.datos_min", 1);

        // Datos de Latencia Promedio
        $datos_avg = acceso_archivo("../latencia/.datos_avg", 1);

        // Datos de Latencia Máxima
        $datos_max = acceso_archivo("../latencia/.datos_max", 1);
//
```

```
//      Creación de la gráfica
//
        // Creación de la instancia gráfica
        $grafico = new graph(750,250,"auto");
        $grafico -> img -> SetMargin(50,20,20,40);
        $grafico -> img -> SetAntiAliasing();
        $grafico -> SetScale("textlin");
        $grafico -> SetFrame(false);
        $grafico -> title -> Set("Latencia [milisegundos]");
        $grafico -> title -> SetFont(FF_VERDANA,FS_BOLD);
        $grafico -> legend -> Pos(0.1, 0.15, "left" ,"center");
        $grafico -> xaxis -> title -> Set("Tiempo [HH:MM:SS]");
        $grafico -> yaxis -> title -> Set("ms");
        $grafico -> xgrid -> Show();

        // Leyendas del eje X
        $datax = fechas("../logs/latencia.log");
        $grafico -> xaxis -> SetTickLabels($datax);
        $grafico -> xaxis -> SetFont(FF_ARIAL,FS_NORMAL,8);
        $grafico -> xaxis -> SetLabelAngle(10);

        // Datos Latencia Máxima
        $p1 = new LinePlot($datos_max);
        $p1 -> SetColor("orange");
        $p1 -> SetLegend("Máxima");
        $p1 -> SetCenter();
        $grafico -> Add($p1);

        // Datos Latencia Promedio
        $p2 = new LinePlot($datos_avg);
        $p2 -> SetColor("gray");
        $p2 -> SetLegend("Promedio");
        $p2 -> SetCenter();
        $grafico -> Add($p2);

        // Datos Latencia Mínima
        $p3 = new LinePlot($datos_min);
        $p3 -> SetColor("indianred");
        $p3 -> SetLegend("Mínima");
        $p3 -> SetCenter();
        $grafico -> Add($p3);

        $grafico->Stroke();
?>
```

Archivo *gpaq_perdidos.php:*

```
<?php
/*****************************************************************************************************
*       Autor:          Héctor Julián Selley Rojas
*       Lugar:          Centro de Investigación en Computación
*                       Instituto Politécnico Nacional
*
*       Descripción:    Script que lee los datos almacenados en el archivo
*                       registro y crea la gráfica correspondiente
*****************************************************************************************************/
        include ("jpgraph.php");
        include ("jpgraph_pie.php");
        include ("jpgraph_pie3d.php");
        include ("../other/acceso.php");

//      Lectura de los datos del archivo
        // Datos Enviados
        $enviados = acceso_archivo("../paq_perdidos/.datos_env", 1);
        // Datos Recibidos
        $recibidos = acceso_archivo("../paq_perdidos/.datos_rcbd", 1);
        $porc_perdidos = ((1 - $recibidos[0]/$enviados[0])*100);

        $datos = array ();
        $datos[0] = $porc_perdidos;
        $datos[1] = 100 - $porc_perdidos;

        // Creación de la instancia gráfica
        $grafico = new PieGraph(373,250,"auto");
//      $grafico -> SetShadow();
        $grafico -> SetFrame(false);
        $grafico -> title -> Set("Paquetes Perdidos");
        $grafico -> title -> SetFont(FF_VERDANA,FS_BOLD);

        $p1 = new PiePlot3D($datos);
        $p1 -> SetAngle(40);
//      $p1 = new PiePlot($datos);
//      $p1 -> ExplodeSlice(1);
        $p1 -> ExplodeAll(10);
        $p1 -> SetCenter(0.45);
        $p1 -> SetSize(0.35);
        $p1 -> SetTheme("water");
        $p1 -> SetLegends(array("Perdidos","Respondidos"));

        $grafico -> Add($p1);
        $grafico -> Stroke();

?>
```