INFORME FINAL
PROYECTO DE GRADO:

HERRAMIENTA DE MONITOREO Y ALERTA CONFIGURABLE (HMAC)

Ana Carolina Pascual, Marcos Rodríguez

Tutores: Ariel Sabiguero Yawelak, Héctor Cancela

Instituto de Computación
Facultad de Ingeniería
Universidad de la República Oriental del Uruguay

Abril 2004

Resumen

En el mundo de hoy y cada vez más las plataformas informáticas de aplicación en las empresas tienen un crecimiento importante, éstas no solo se refieren a una red de computadoras (que cada vez son más grandes) sin que incluyan recursos de todo tipo ya sean hardware o software.
Los diversos contextos de aplicación abarcan la interconexión de soluciones de distintos fabricantes y de las formas más variadas: computadoras, fuentes de energía, impresoras, switches, servidores, CPU, memoria, discos, procesos en ejecución, aplicaciones etc.

En este sentido es necesario contar con políticas y mecanismos de administración y gestión de estos recursos para asegurar el control y conocimiento del estado de los recursos en forma permanente.
Tales mecanismos deben permitir actuar en caso de ser necesario para evitar o corregir situaciones indeseables detectadas, algunas de estas acciones pueden ser: envió de e-mail, llamadas telefónicas, parar o ejecutar un proceso, prender o apagar un equipo, ejecutar backups, disparar una alarma sonora o lumínica, etc.
También es adecuado el registro de los sucesos relevados para su posterior análisis, por diversos mecanismos como ser archivos de texto o en manejadores de bases de datos.

Hay disponibles distintas soluciones posibles a este problema las que se diferencian por su costo, alcance, funcionalidades, plataforma de aplicación. Aun dentro de la gran variedad, en ciertos contextos es adecuado pensar en definir una solución personalizada según necesidades particulares y que no son cubiertas por una única solución disponible.

El objetivo de este trabajo es la construcción de una herramienta de monitoreo y gestión aplicada a este contexto, específicamente la definición y desarrollo de un prototipo modular de monitoreo y gestión de recursos, manejo de eventos, alertas o acciones y registro (logging).

El informe presenta un estudio de soluciones disponibles y la definición de las funcionalidades básicas del producto a construir.
Se incluye luego una discusión de la tecnología JMX (Java Management Extensions), que ha sido fundamental en la definición de la arquitectura del producto y posterior implementación.

El producto desarrollado es configurable y extensible, y permite incorporar o anexar nuevos módulos o pluggins de recursos, alertas o registro según las necesidades particulares. En esta herramienta se da soporte para monitorear y gestionar recursos mediante SNMP (siempre que el recurso defina el MIB adecuado), u otros mecanismos, distintos mecanismos de alerta y acción (llamadas telefónicas, envió de mail, etc), y distintos métodos de registro posibles (archivos de texto, Tablas en DBMS).

Palabras Clave: monitoreo, gestión, recursos, JMX.

INDICE

Indice de ilustraciones

1. ACERCA DEL INFORME

Este documento presenta el informe final del proyecto de grado "Monitor de Alertas configurable (HMAC)" de la carrera Ingeniería en Computación de Faculta de Ingeniería.
En este capítulo se realizará una introducción al documento, estableciendo su alcance, audiencia y contenido.

1.1 ALCANCE

El documento muestra en forma detallada el proceso de desarrollo del proyecto en su totalidad, describiendo los objetivos, resultados esperados, logros, actividades realizadas y conclusiones para sus distintas etapas. Contiene también documentación e información de aspectos tecnológicos, del estado del arte y de interés en el área de gerenciamiento de recursos.

1.2 AUDIENCIA

Este informe está dirigido tanto a personas interesadas en utilizar el monitor de alertas configurable [HMAC], como aquellas que quieran introducirse en el tema de gestión y administración de recursos informáticos.

Para quienes tengan interés en el uso de la herramienta desarrollada, se detallan los aspectos de implementación y configuración necesarios para su implantación. También se expone con detalle la configuración e instalación de otras herramientas estudiadas en profundidad que pueden ser de utilidad a personas que tengan interés en éstas.
Es recomendable la lectura detallada de las siguientes secciones:
3.2.3 "JMX" donde se describe la tecnología JMX que es fundamental en el desarrollo del producto [JMX].
3.4 "Especificaciones de diseño y arquitectura" donde se detallan aspectos relevantes del diseño, los componentes centrales de la arquitectura integrados con JMX y los procesos que integran cada componente. También son de utilidad los documentos "Guía del desarrollador" [GD], "Guía de instalación y configuración" [GIC] y "Guía de Usuario" [GU] del presente proyecto.

Para el público interesado en introducirse en la problemática que aborda la herramienta, se expone el relevamiento realizado de distintas tecnologías disponibles. Es recomendable la lectura de la sección 3.2 "Relevamiento de estado del arte" y el Apéndice 8.1 donde se presentan las herramientas de monitoreo y gestión estudiadas, en algunos casos se profundiza en la instalación y funcionamiento de productos considerados de interés tanto por su aplicación en el contexto de la empresa como por su aporte a la comprensión de los temas de gestión y administración de recursos. También es de utilidad la lectura del apartado 3.2.3 que desarrolla la tecnología JMX.

En el capitulo 5 se desarrollan las conclusiones generales como resultado del análisis del desarrollo del proyecto en su conjunto, evaluación del producto, del grupo de trabajo, etc. También se plantean una serie de trabajos futuros considerados interesantes pero que no se incluyeron en el alcance del proyecto por la duración del mismo.

En los Apéndices se puede encontrar información sobre: detalle de los casos de uso relevados en la etapa de requerimientos (Apéndice 8.2), discusión sobre posibles arquitecturas del sistema (Apéndice 8.3), el desarrollo de un prototipo de prueba utilizado para probar el uso de JMX (Apéndice 8.4), Evolución de los diagramas de clases (Apéndices 8.5 al 8.7).

Dadas las características que presenta el producto, es recomendable que el lector tenga nociones en las áreas de administración de recursos informáticos, redes, plataformas W2K y Linux, entorno de programación Java.

1.3 CONTENIDO

En el capítulo 1 se describe el alcance y contenido del presente documento.

En el capítulo 2 se da una breve introducción al tema del proyecto y el contexto de trabajo en el cual se desarrolla. Se especifican los objetivos y alcance del mismo.

En el capítulo 3 se detallan las etapas del proceso de desarrollo de la herramienta. Si bien algunas de éstas se superpusieron en el tiempo, es posible diferenciar y evaluar los resultados planificados y esperados para cada una de ellas.

En el capítulo 4 se realiza una evaluación del proyecto en cuanto al cronograma. Se plantean los motivos de atrasos y decisiones tomadas que alteraron el cronograma inicial.

El capítulo 5 presenta una evaluación del proyecto describiendo los logros y conclusiones. Se plantean también posibles trabajos futuros que no se incluyeron en el alcance del trabajo pero que sería interesante incorporar a la herramienta.

En el capítulo 6 se incluyen una serie de conceptos empleados o mencionados a lo largo del presente documento.

En el capítulo 7 se detallan referencias consideradas de interés.

En el capítulo 8 se incluye como apéndice un conjunto de documentación referente al desarrollo del proyecto: complemento para el estudio del “Estado del Arte”, casos de uso relevados en el “Análisis de requerimientos”, desarrollo del prototipo de prueba para validar el uso de la tecnología JMX [JMX] y diagramas de clases.

2. ACERCA DEL PROYECTO

2.1 INTRODUCCIÓN

La gran variedad de soluciones informáticas disponibles, el número de máquinas capaces de ser interconectadas en redes de computadoras, las posibilidades en la actualidad de interoperar soluciones de distintos fabricantes que cooperan en la búsqueda de objetivos comunes, en distintas plataformas y la interacción con distintos componentes de hardware en los esquemas actuales de aplicación imponen la necesidad de administrar y gestionar estos componentes.

En este contexto se hace cada vez más necesario que las empresas incorporen mecanismos de control para monitorear el normal funcionamiento de estos componentes, ya sean piezas de hardware (estado de los nodos de una red, discos, memoria, CPU, impresoras, etc) o recursos de software en aplicación de los cuales es importante estar en conocimiento del estado de ejecución.
Por lo tanto es muy valioso contar con sistemas de monitoreo y alertas que sean capaces de detectar fuentes de posibles problemas y actuar en consecuencia, dando respuesta de la forma más rápida y eficiente posible.

Estos sistemas pueden estar enfocados a monitorear un gran número de componentes ya sean de hardware o software y usar distintos mecanismos de control y alertas.

También es importante en este contexto de cambios permanentes la flexibilidad con que estos sistemas se ajustan a los cambios, permitiendo incorporar nuevos elementos a monitorear, mecanismos de acción o alerta según las necesidades particulares.

Es importante también que estas herramientas se adapten con facilidad a las situaciones particulares de cada empresa y la plataforma de aplicación en particular. Esto se debe a que si bien hay una gran variedad de soluciones disponibles, es muy difícil cubrir todas las necesidades en su conjunto en una única herramienta y al mismo tiempo que tenga un costo económico bajo.

Entre los sistemas de administración y gestión de recursos podemos identificar los siguientes componentes básicos:

- Recursos a ser administrados. Una computadora, una red de computadoras, un sistema compuesto de varios componentes de Hardware y software o una aplicación particular. Es necesario contar con mecanismos que permitan a los sistemas de gestión y monitoreo conocer el estado del recurso, tales mecanismos envían la información por medio de eventos o aceptan requerimientos sobre una propiedad en particular del recurso.

- Agentes. Encargados de interactuar entre los recursos y los sistemas de gestión, canalizan los requerimientos y eventos.

- Sistemas de monitoreo y gestión. Estos proveen la infraestructura para administrar, recibir eventos de los agentes, permiten ejecutar requerimientos sobre los recursos a través de los agentes para monitorear datos o detectar posibles problemas.

El proyecto actual se desarrolla en el marco de la construcción de una herramienta de monitoreo y alertas flexible y extensible a nivel de prototipo funcional.

2.2 CONTEXTO DE TRABAJO

El trabajo se realizó en coordinación con la empresa K&S Asociados SRL, en la cual los alumnos se desempeñan como desarrolladores. Éste se enmarcaba en un proyecto de adquisición de conocimiento en tecnologías de código abierto, Java, y desarrollo para la plataforma UNIX. Por lo tanto, la temática era afín a los planes de desarrollo de software en la misma.
Se dispuso de un ambiente de desarrollo, y la infraestructura acorde con las necesidades del proyecto, por lo que gran parte del desarrollo se realizó en instalaciones de K&S.

También era interés en esta nueva área de trabajo, el desarrollo de una herramienta de monitoreo que se integre al conjunto de los productos desarrollados.
Es de destacar que la empresa ya contaba con un sistema de monitoreo desarrollado para WIN32, que se utilizaba integrada con otras soluciones de software desarrolladas en esa plataforma.

2.3 OBJETIVOS

El objetivo principal de este proyecto fue definir y desarrollar a nivel de prototipo funcional, una plataforma modular de gestión de recursos, con manejo de eventos, alertas y registro (logging) extensible en base a módulos acoplables.
Parte del trabajo incluía desarrollar un motor de comunicaciones que permitiera que diferentes módulos puedan monitorear diferentes recursos utilizando el mecanismo más adecuado disponible, sea SNMP [SNMP], invocación remota de métodos, extracción de información de DBMS [DBMS]. Este monitor configurable debería aceptar diferentes módulos de alerta configurables en función de la prioridad de la alerta y otros parámetros que se definieran pudiendo brindar múltiples caminos de notificación para un mismo evento. Además debería poder modularizar la generación de registros para permitir su inserción en el entorno adecuado para poder, por ejemplo, realizar registros a un DBMS, escritura en archivos de texto plano u otros módulos a definir [HMAC].

2.4 ALCANCE DEL PROYECTO

Como resultado del proyecto se esperaba el desarrollo de un prototipo de motor genérico, fácilmente extensible, y configurable a la vez, que permitiera realizar acciones de monitoreo y gestión de recursos, alertas y registro. Es decir, que el producto no se limitara por un conjunto de funcionalidades determinadas, sino que soportara la integración de nuevos módulos de monitoreo, alerta y registro de forma sencilla, estableciendo las interfases y mecanismos necesarios para la extensión de la herramienta.

El objetivo principal del proyecto era diseñar de forma completa el motor, e implementarlo a un nivel funcionalmente aceptable de acuerdo a los plazos del proyecto y a las necesidades concretas de la empresa.

Si bien no era el objetivo principal del proyecto, se esperaba además, el desarrollo de un conjunto específico de módulos de monitoreo, alerta y registro. En este punto se trataron de contemplar aspectos que permitieran probar el alcance de la herramienta y su potencialidad, y que a la vez fueran de utilidad para la empresa. Algunos de éstos son: monitoreo mediante el uso de SNMP, monitoreo de aplicaciones particulares de la empresa, envío de mail, búsqueda de patrones en archivos, registros en archivos o base de datos.

También se debía generar un conjunto de documentación a lo largo del proyecto, informes intermedios, documentación del código utilizando la herramienta de documentación javadoc [javadoc], y un informe final que incluyera todo el proceso y actividades llevadas a cabo durante el proyecto.

3. ETAPAS DEL PROYECTO

3.1 INTRODUCCIÓN

El cronograma al inicio del proyecto planteaba un período de investigación, desarrollo, verificación y puesta a punto de los informes intermedios, de ocho meses. Se detallaron el conjunto de tareas a realizar a lo largo del proyecto, especificando una estimación de tiempo, en meses, de lo que llevaría cada una de ellas. Igualmente, en el transcurso del trabajo se identificaron cinco etapas bien diferenciadas en su alcance y contenido aunque no hubo un límite bien diferenciado en el tiempo ya que algunas de éstas se superpusieron.
Estas etapas son:

- Relevamiento de estado del arte. El interés básico fue reconocer herramientas de administración y monitoreo disponibles, políticas aplicadas y funcionalidades que se deberían tener en cuenta. También se estudió el monitor que estaba en uso en la empresa.
- Análisis de requerimientos. Se definieron los requerimientos mínimos y deseables que deberían estar contemplados en el prototipo a construir.
- Análisis de diseño y arquitectura. Se discuten distintas opciones de arquitectura y diseño del producto para posteriormente definir la arquitectura del sistema.
- Desarrollo e implementación. Desarrollo del motor de la herramienta y una conjunto de pluggins de monitoreo, alertas y registro.
- Plan de pruebas. Desarrollo de plan de pruebas y verificación del sistema.

De aquí en más se hará referencia a éstas por su papel en el proceso de desarrollo del proyecto y no a las tareas especificadas en el cronograma original.

En esta sección se presenta el proceso de desarrollo de la herramienta, los objetivos planteados en cada etapa, los logros y los resultados, los problemas que se encontraron y cómo se resolvieron.

3.2 RELEVAMIENTO DE ESTADO DEL ARTE

3.2.1 Objetivos

El interés fundamental en esta etapa fue lograr un acercamiento al mundo de las herramientas que intentan dar solución al problema de la administración y monitoreo de recursos en general (servicios, componentes y recursos de red), con el fin de entender la filosofía en que se basan estas aplicaciones e intentar adquirir los conocimientos necesarios para abordar el desarrollo del prototipo.
Además, conocer qué posibilidades hay en el mercado o la Web y qué elementos funcionales deben ser tenidos en cuenta a la hora del desarrollo.

Si bien el estudio de estas herramientas no fue con el interés de conocer cómo estaba implementada cada solución, se buscó reconocer funcionalidades diversas y su alcance. Igualmente se intentó tener comprender las tecnologías aplicadas para su desarrollo.

En general el grado de profundidad en el estudio de cada herramienta vista fue enfocado a sus posibilidades funcionales, salvo algunos casos en los cuales se encontró beneficioso su instalación, prueba y estudio.

Como caso de particular interés se estudió el sistema actual de monitoreo que tiene la empresa para entender su arquitectura y funcionalidades. Además, la herramienta a desarrollar debía soportar las funcionalidades que soporta el sistema actual.

Esta etapa sirvió como punto de partida para determinar qué enfoque se le daría a la recopilación de requerimientos. En las secciones siguientes se detallan un conjunto de herramientas vistas que se consideraron de interés o que fueron aplicadas y probadas, entre éstas: tecnologías de monitoreo de red

(SNMP, RMON, NetFlow, sFlow), herramientas de código abierto (MRTG, GetIf), otras distribuidas bajo licencia (3COM) y el monitor de aplicación en K&S.
En el Apéndice 8.1 se resumen los resultados y se continúa con el estudio de otras herramientas vistas.

3.2.2 Desarrollo

De los protocolos y herramientas vistas algunos fueron sugeridos por los tutores y otros surgieron de búsquedas en Internet. El estudio no fue exhaustivo. De los resultados de las búsquedas, que arrojaban una gran cantidad de herramientas disponibles y con una diversidad de enfoques y aplicaciones, fueron seleccionadas algunas que parecieron interesantes o representativas.

Este proceso de estudio se puede clasificar en tres partes: búsqueda, selección y análisis.
En algunos casos puntuales se consideró una cuarta etapa: instalación y prueba.

La evaluación de qué herramienta probar en general estuvo basada en dos factores principales: su disponibilidad y su utilidad práctica. Esto se debió a que hay un conjunto de herramientas en el área que son muy costosas y por lo tanto difícil de obtener. También fue posible encontrar herramientas de mucha utilidad y de libre distribución. Algunas de éstas fueron instaladas y probadas. En especial MRTG fue de gran utilidad para la empresa. Inclusive al día de hoy se sigue utilizando.

También fue visto en detalle el proceso de configuración y ejecución del monitor que se encuentra en producción en la empresa.

Como el protocolo SNMP es un estándar en el área de gestión y administración de recursos se dedicó un tiempo considerable al estudio de sus posibilidades, con el objetivo de entender su arquitectura y manejo.

Se presentará una breve descripción de alguna de las herramientas vistas. Si se desea profundizar en el tema se puede tomar como referencia el Apéndice 8.1 en el que se detallan las funcionalidades de todas las herramientas vistas, inclusive en algunos casos el proceso de instalación y puesta en funcionamiento. En el capítulo 7 se ofrecen referencias y enlaces a lugares de interés relacionados al tema.

3.2.2.1 Tecnologías de monitoreo de red.

Dentro de las tecnologías de monitoreo de red se eligieron SNMP, RMON, netFlow y sFlow.

SNMP (Simple Network Management Protocol)

Es un protocolo de administración de dispositivos de red, utilizado en redes TCP/IP para monitorear y administrar gran diversidad de software y hardware. Es un estándar y cada vez más las empresas fabricantes tanto de hardware como software implementan mecanismos que permiten el monitoreo de recursos con SNMP. Éstos implementan módulos básicos de información en una base de datos (MIB) que permiten el diálogo con los agentes SNMP.
Se puede monitorear cualquier recurso que implemente la MIB correspondiente, con la salvedad de que es necesario recompilar el archivo con las bases de datos que definen las MIB.

Está implementado con una arquitectura cliente-servidor y consta de cuatro componentes básicos:

1. Elementos administrados. Cualquier dispositivo de red o componente de software.

2. Agentes. Es el software instalado en el dispositivo a gestionar. Funcionan como servidores en el sentido de que dan el servicio de responder requerimientos sobre las variables MIB. Se deben instalar en cada máquina a monitorear (implementados por software o hardware). Mantienen

una base de datos local (MIB) donde se representan todos los objetos que se pueden monitorear.

3. SAR (Sistema de Administración de redes). Es una pieza de software que se instala en el equipo que realiza la tarea de monitoreo. Funciona como cliente consultando a los agentes SNMP y maneja dos operaciones básicas como son leer y escribir objetos de la base de datos MIB. Es el software que ejecutan las aplicaciones de administración y monitoreo.

   Se utilizó el software SNMPUtil para el trabajo en Windows y el paquete ucd-snmp en Linux.

4. El protocolo SNMP en sí mismo es la definición de la forma en que los dispositivos y el SAR intercambian información.

Es un protocolo no orientado a conexión, utiliza UDP como un protocolo de transporte de mensajes, utilizando el puerto 161 para todos los mensajes, excepto para las traps, que llegan el puerto 162 de UDP. Los agentes reciben sus mensajes del administrador a través del puerto UDP 161 del agente.

SNMP está definido formalmente en tres *Request For Comments* (RFC):
1. RFC 1157 Simple Network Management Protocol (SNMP)
2. RFC 1155 Structure Management Information (SMI)
3. RFC 1213-1156 Base Information Administrable (MIB-II)

RMON (Monitor Remoto)

Es un estándar de la Internet Engineering Task Force (IETF) que especifica un dispositivo de monitoreo de tráfico remoto. Un dispositivo RMON monitorea y descifra cada paquete de la red y crea tablas de medidas que después pueden transmitirse a alguna aplicación de administración de red [RMON].
Al igual que SNMP tiene una arquitectura cliente servidor.
El siguiente cuadro resume la arquitectura.

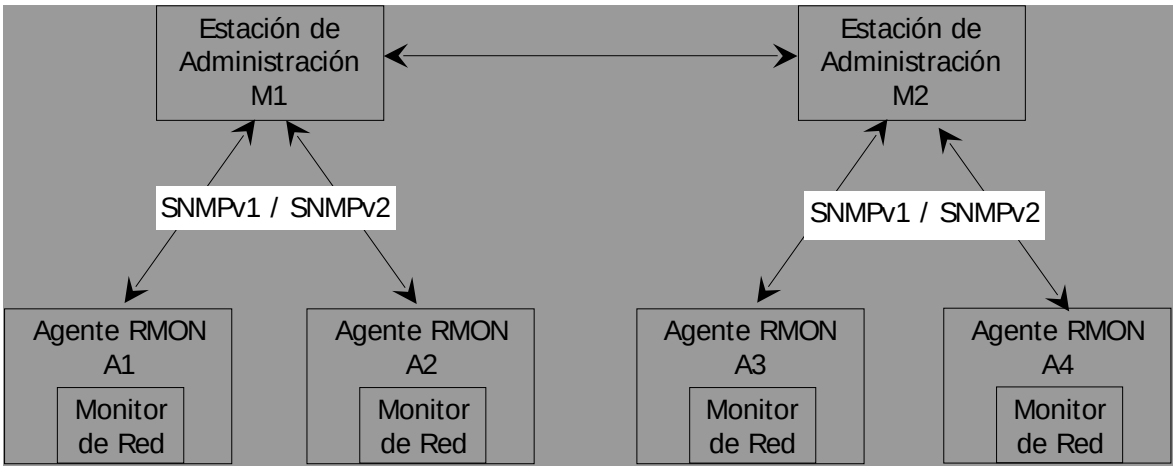


Figura 1. Arquitectura RMON

Los datos de administración recolectados por los Monitores de red son enviados a las Estaciones de administración. Los Monitores remotos de red se ubican en los Agentes RMON, que proveen funciones de soporte de protocolos de comunicación con las estaciones de administración y recuperación de errores, y actúan como intermediarios entre los Monitores de red y las Estaciones de administración. Para transferir los datos a las estaciones de administración pueden usar protocolos SNMPv1 o SNMPv2.

Un Agente RMON con un monitor remoto de red puede enviar datos a una o más estaciones de administración y pueden ubicarse en hosts, routers y bridges.
Esta arquitectura permite que varios monitores de red y varias consolas de monitoreo intercambien datos sobre los elementos de la red.

La especificación RMON es fundamentalmente una ampliación de MIB. Proporciona información sobre uso de la red, tráfico, carga, diagnóstico de fallas, datos para planificación, etc.

netflow

Es una tecnología para el monitoreo de tráfico de red desarrollada por Cisco Networks. Permite recolectar, analizar información de los flujos que pasan por switches y routers y luego exportarla a aplicaciones tanto para software de contabilidad, sistemas de facturación, aplicaciones de administración de redes [netFlow].

Su arquitectura se compone de tres elementos:

1. Flow catching: analiza y recolecta cada paquete IP de los flujos que entran por switches y routers y prepara datos para exportar (Netflow datagrams). Permite acumular datos de flujos identificados por características únicas.

2. Flow Collector: captura la información exportada de múltiples routers, filtra y agrega valor a los datos según criterios y políticas del cliente, para luego almacenar esa información.

3. Network Data Analyzer: es una herramienta de análisis de tráfico de red, que combina una interfaz gráfica de usuario con otros módulos que pueden recuperar, mostrar y analizar los datos recolectados por el Flow Collector. Esta herramienta permite a los usuarios ver tendencias de los flujos o visualizarlos casi en tiempo real. Los datos se pueden visualizar con gráficas de barras, de torta o histogramas. También se puede exportar los datos a aplicaciones externas como ser Microsoft Excel para poder generar reportes.

La tecnología NetFlow puede ser útil en redes donde es esencial ver cada flujo, y quizás para monitoreo de seguridad o un conteo específico flujo a flujo.

sFlow

Es una tecnología multi-vendedor, basada en el sampling de paquetes, embebida en switches y routers. Constantemente puede monitorear el flujo de tráfico de aplicaciones a gran velocidad, sin afectar el desempeño de la red. Combina contadores exactos de paquetes con muestreo estadístico del estado de las tablas de ruteo para que switches y routers envíen paquetes de forma aleatoria a un colector central para ser analizado [sFlow].

Los elementos básicos en la arquitectura son el Agent y el Collector sFlow.

1. Agent: Es un proceso de software que corre como parte del software de administración de red dentro de un switcher o router. Combina contador de interfases y muestras de flujos en datagramas sFlow que son enviados a través de la red hacia el Collector.
   El muestreo de paquetes es típicamente realizado por switcheo/ruteo ASICs, proveyendo gran desempeño. El Agente realiza muy poco procesamiento, simplemente empaqueta datos en los datagramas sFlow que son enviados inmediatamente a la red, minimizando así los requerimientos de memoria y CPU del Agente.

2. Colector: El colector se encarga de recolectar y analizar los datos, pudiendo generar información detallada del tráfico en tiempo real.

SFlow permite medir, recolectar, guardar y analizar los datos del tráfico de la red, brindando información detallada de los paquetes, switches y routers en tiempo real.
La técnica de muestreo estadístico utilizado permite administrar un gran número de agentes, muchos puertos de switches por un solo servidor. El bajo costo de los agentes y las ventajas técnicas lo hacen ideal para el monitoreo y reporte continuo de una gran red.
La especificación de sFlow fue publicada como RFC 3176.

3.2.2.2 Herramientas Open Source.

3.2.2.2.1 Agentes SNMP.

En el mundo Open source hay disponibles varias implementaciones de agentes SNMP que permiten interactuar con las bases de datos MIB definidas en cada equipo, estos disponen de los mecanismos necesarios para leer y escribir los objetos del MIB (get y set).
En el transcurso de la fase de investigación se consideró usar el paquete ucd_snmp para trabajar con Linux y en Windows se instaló el servicio SNMP donde fue necesario, las consultas a la MIB se realizaron con la herramienta SNMPUtil.

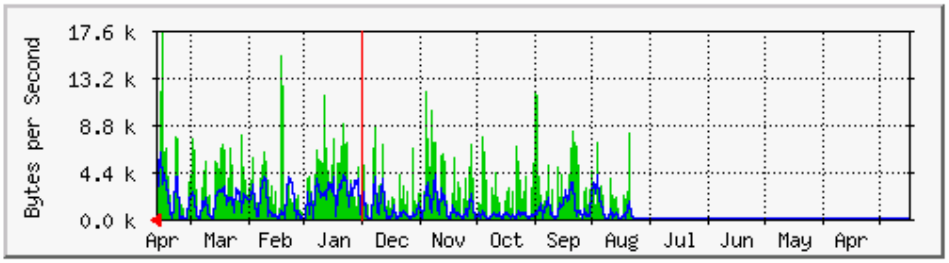
3.2.2.2.2 MRTG (Multi Router Traffic Grapher).

MRTG es una herramienta de gestión diseñada en sus inicios para monitorear el tráfico entrante y saliente de interfaces de red en tiempo real. Puede monitorear máquinas remotas que tengan habilitado el servicio SNMP. [MRTG]
La implementación de SNMP está escrita en Perl y el programa principal en C lo que optimiza el proceso de edición y generación de imágenes Gif. Presenta esta información en una página principal HTML que contiene las imágenes generadas.
MRTG necesita el intérprete de Perl para su normal funcionamiento, por lo que debe estar instalado en la máquina donde corre el servicio MRTG. [Perl]
La toma de datos sigue el siguiente mecanismo:

- Consulta a las máquinas definidas, ejecutando la operación Get de SNMP sobre los OID determinados en el archivo de instalación para cada máquina.
- Actualiza las imágenes de los gráficos con los nuevos valores y borra los viejos.
- Almacena los nuevos valores en las bases de datos de log.

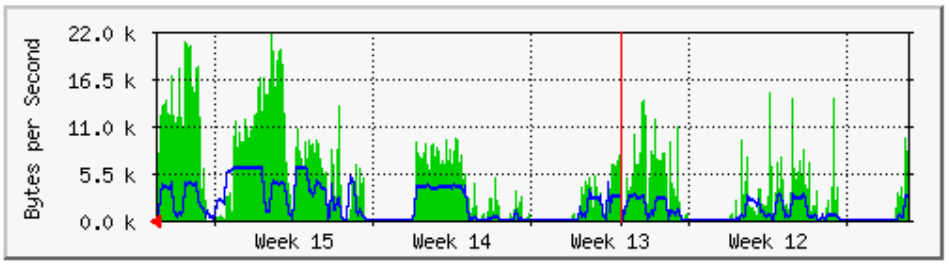
Esta herramienta interactúa con los agentes SNMP consultando los objetos del MIB configurados. Con una frecuencia determinada registra todos los valores obtenidos en archivos y presenta los resultados en 4 gráficos diferenciados por período de tiempo, diario, semanal, mensual y anual.
Como parte del mecanismo de estudio y aprendizaje de esta herramienta se consideró positiva su instalación con el fin de monitorear el ancho de banda utilizado por el servidor de Internet de la empresa y los servidores de base de datos.
Se encuentra información de la instalación de esta herramienta en el Apéndice 9.2.

A continuación se presentan los gráficos creados por el uso de esta herramienta por un período de 8 meses.



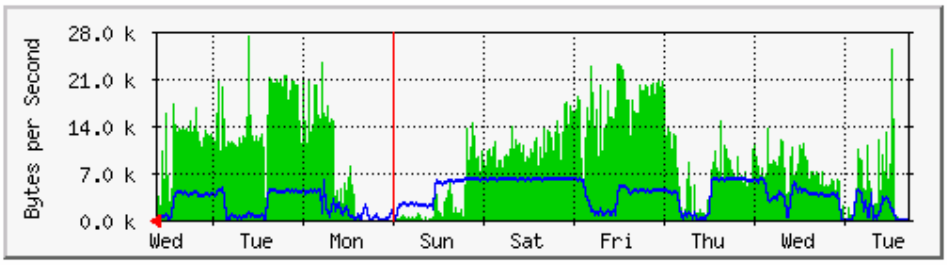
Max In:17.5 kB/s (1.4%) Average In:3528.0 B/s (0.3%) Current In:5147.0 B/s (0.4%)
Max Out:6295.0 B/s (0.5%) Average Out:1367.0 B/s (0.1%) Current Out:1494.0 B/s (0.1%)

Figura 2. Gráfico de ancho de banda por año.



Max In:21.8 kB/s (1.7%) Average In:4569.0 B/s (0.4%) Current In:4492.0 B/s (0.4%)
Max Out:6361.0 B/s (0.5%) Average Out:1725.0 B/s (0.1%) Current Out:495.0 B/s (0.0%)

Figura 3. Gráfico de ancho de banda por mes.



Max In:27.7 kB/s (2.2%) Average In:9722.0 B/s (0.8%) Current In:994.0 B/s (0.1%)
Max Out:6405.0 B/s (0.5%) Average Out:3565.0 B/s (0.3%) Current Out:424.0 B/s (0.0%)

Figura 4. Gráfico de ancho de banda por semana.

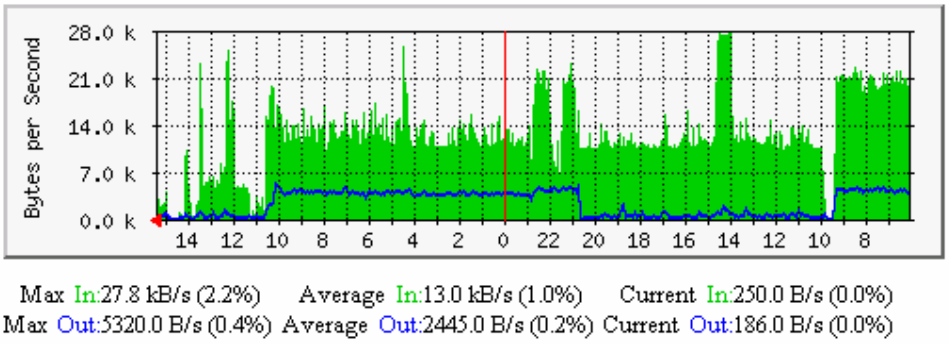


Figura 5. Gráfico de ancho de banda por día.

3.2.2.2.3 Ethereal.

Es un analizador de paquetes de red (sniffer) con una interfase gráfica. Permite analizar paquetes de una gran cantidad de protocolos. Es posible analizar el tráfico entrante y saliente por la tarjeta de red o por medio de un archivo de captura previamente generado [ETH].

3.2.2.2.4 Getif 2.x Network Tool

Es una herramienta con interfase gráfica que descubre información de un equipo en la red con solo ingresar su dirección IP, chequea y recupera un conjunto muy completo de datos. Está basado en la realización de consulta SNMP a la base de datos MIB instalada en el equipo [GIF].
Entre estos datos se pueden mencionar: descripción del equipo, puertos de configuración del Agente SNMP, control de puertos específicos (HTTP, SMTP, POP3, Telnet, etc) con la ejecución del comando PING.
Una sección a destacar es su utilidad de navegación por la MIB. Permite recuperar información muy completa sobre cada objeto instalado en ésta y gráficos de los valores de un objeto en función del tiempo.

Se comienza una sesión con el ingreso de la dirección IP del equipo que se quiere consultar la cual despliega en pantalla información del equipo. Se muestra a continuación:

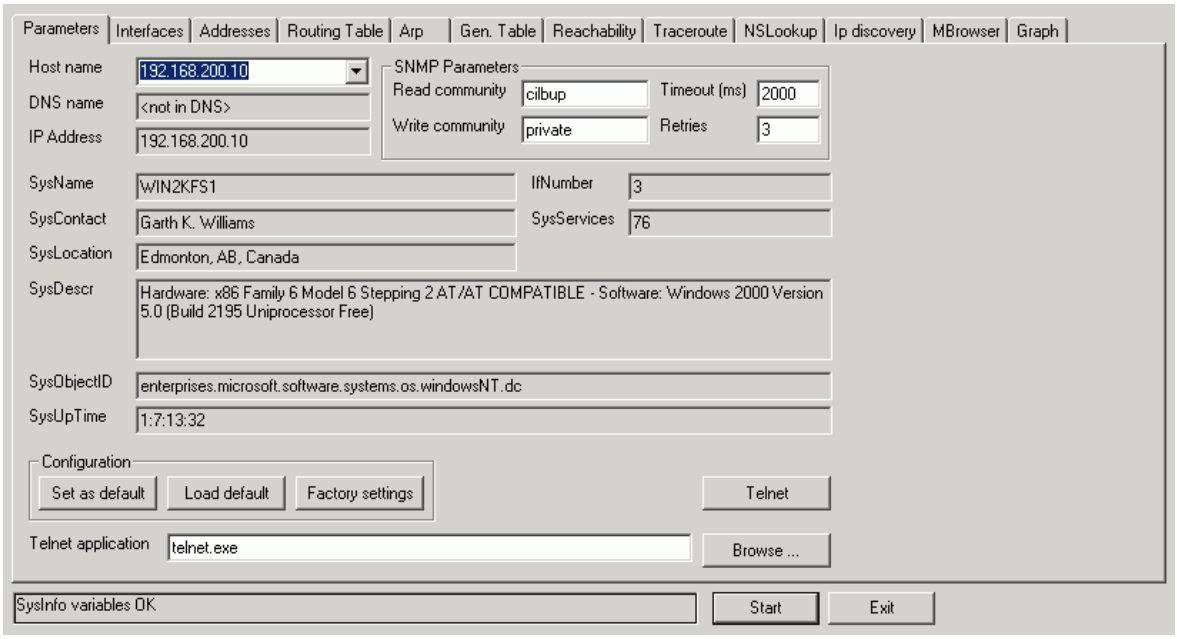


Figura 6. Inicio de sesión.

Otro ejemplo de sus posibilidades es el navegador de MIB. Éste permite descubrir información ejecutando la operación de recuperación sobre toda la base de datos o sobre un objeto en particular, también para los objetos que lo permitan se puede ejecutar el comando set cambiando el valor actual del objeto.

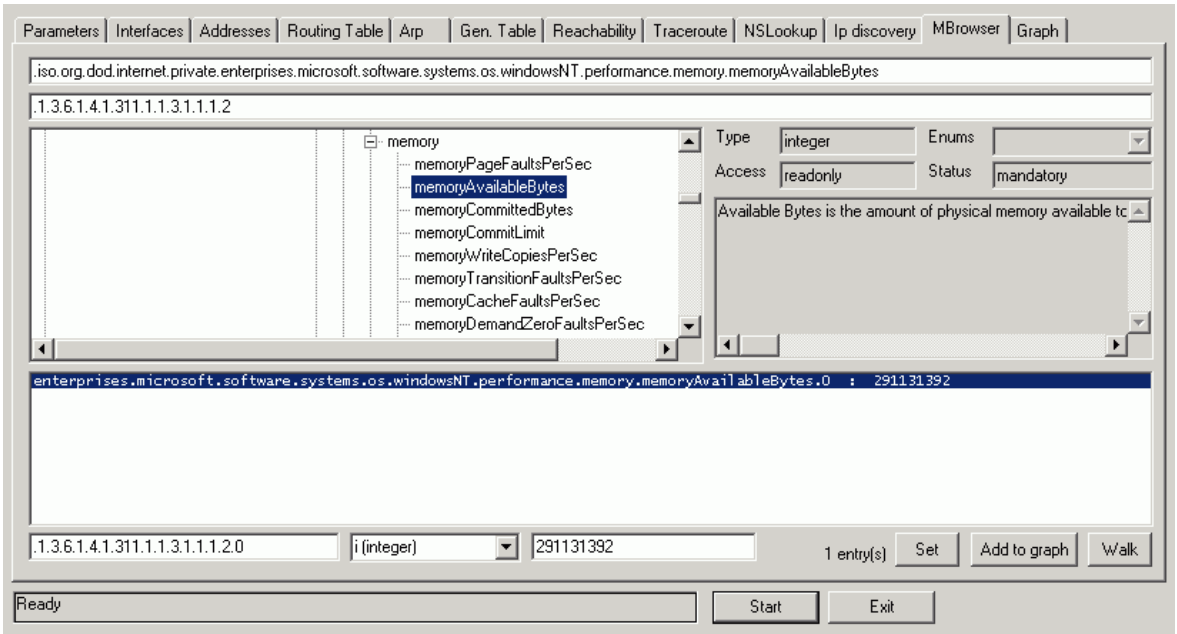


Figura 7. Navegador de MIB.

Es una herramienta muy útil, de fácil uso y de libre distribución.
Permite incorporar nuevos objetos a su base de datos de una forma simple, sólo es necesario:

- Detener la ejecución del programa.
- Copiar el archivo de definición de los nuevos objetos en el directorio que contiene las MIBs que reconoce getif y nuevamente ejecutar el programa.

3.2.2.3 Herramientas propietarias

3.2.2.3.1 Network Supervisor (3Com)

Este producto no es de libre distribución y fue probado un demo con permiso de uso de un mes.

Descripción

Este producto ofrece ciertas posibilidades que facilitan la gestión de recursos de red. Permite monitorear toda la red desde un solo equipo, sin agregar carga a la red y se comunica con los dispositivos ejecutando un comando y esperando la respuesta (por ejemplo PING a un equipo). Ejecuta un requerimiento sobre un puerto o servicio, recupera el valor y recupera el tiempo de demora de la respuesta. A diferencia de SNMP en que el requerimiento de la operación get es sobre un contador en particular y no se mide el tiempo de respuesta.

Permite configurar rangos en cada equipo sobre cada servicio que se quiera monitorear y acciones en caso de que se superen los valores máximos y mínimos.
Las acciones son configurables, pueden ser sonoras, en pantalla mandar un e-mail, etc.
Se puede instalar en equipos de escritorio por lo que no requiere equipos costosos.

Características Generales

Descubrimiento automático y mapeo de red.
AL iniciar una sesión permite la opción de realizar un mapeo automático de todos los dispositivos activos en la red o de trabajar en base configuraciones de redes guardadas en archivos de sesiones anteriores, realizando un descubrimiento de todos los dispositivos existentes (puertos y conexiones, hubs, switches, routers).
Permite ver las características de cada nodo mapeado en particular, dirección IP, dirección MAC, sistema operativo, etc.

Monitorea los niveles de carga de toda la red, subredes y dispositivos en general (hubs, switches, etc)
En forma automática despliega en forma gráfica los nodos críticos o sobrecargados de la red en tiempo real con un manejo de colores (blanco, verde, amarillo, rojo (crítico)).

Manejo de umbrales. Para cada ítem a monitorear es posible definir un rango de medidas para las cuales alertar o notificar. Esto permite controlar nodos críticos, definiendo específicamente los umbrales críticos (por ejemplo, tiempo de respuesta para el comando PING, carga excesiva para el tráfico de red o servicio de correo electrónico). En equipos críticos estos rangos se podrían definir un poco más restrictivos como forma de prevenir o tomar acciones en forma rápida.

Lista de eventos posibles y registro en una base de conocimientos histórica.
Permite anticiparse a problemas, aprende de éstos guardando en bases de datos históricas la información y facilita la toma de decisiones en caso que ya se hayan almacenado en la base de datos. Realiza mediciones de los niveles de carga en dispositivos y puertos (puede monitorear hasta 3000 puertos

En estas bases de datos históricas almacena datos sobre los problemas y las soluciones empleadas en cada caso, si se repite algún suceso ya registrado en la base de datos el sistema sugiere al usuario posibles acciones a tomar en función de los históricos.

Servicio de notificación de sucesos configurables y alertas (e-mail, audio, o comunicación con otras aplicaciones). Es necesario que los administradores de red estén informados de problemas o potenciales problemas antes que los usuarios para poder corregirlos a tiempo.
El producto proporciona un sistema de alerta que asegura la notificación inmediata de problemas. Se pueden escoger varios métodos visuales y/o auditivos para notificar la actividad de la red que se defina como crítica.

Monitoreo en tiempo real del ancho de banda usado, errores generados, disponibilidad de servicios y tiempo de respuesta de éstos.
Distintos tipos de reportes configurables.
Facilidad para agendar tareas.
Escalabilidad Este producto se puede acoplar fácilmente como un módulo a otro tipo de herramientas de gestión de distintos fabricantes como por ejemplo HP OpenView [HP].
Portabilidad. Se puede aplicar en varios sistemas operativos Wnt, W2K, Unix, etc.

Instalación

Fue descargada una versión de prueba de este producto e instalada en una estación de trabajo en la red interna de la empresa.
El proceso de instalación consiste en correr un setup y mediante un diálogo se van especificando las opciones de configuración, tipo de red o subred, dispositivos en general, frecuencia de actualización.
El último paso permite el descubrimiento automático de la red definida en los pasos anteriores. Éste proceso tarda unos minutos y como resultado reconoce los dispositivos y máquinas conectados a la red.
Sobre cada dispositivo se reconoce: nombre, dirección IP, dirección MAC, sistema operativo y automáticamente el ícono que lo representa refleja el estado mediante colores (blanco, verde, amarillo y rojo). El rojo indica el estado más grave. Cuando se normalizan los valores críticos se cambia automáticamente el estado.
Estos mapeos de red se pueden guardar para sesiones futuras o hacer un descubrimiento automático en cada inicio de sesión.

Mapeo y descubrimiento automático de la red

El siguiente dibujo muestra el mapeo de la red conseguido mediante el descubrimiento automático de la red y las distintas operaciones que se pueden realizar sobre cada nodo en particular (monitorear o no, ver carga, ver servicios monitoreados, configurar alertas o eventos, etc).

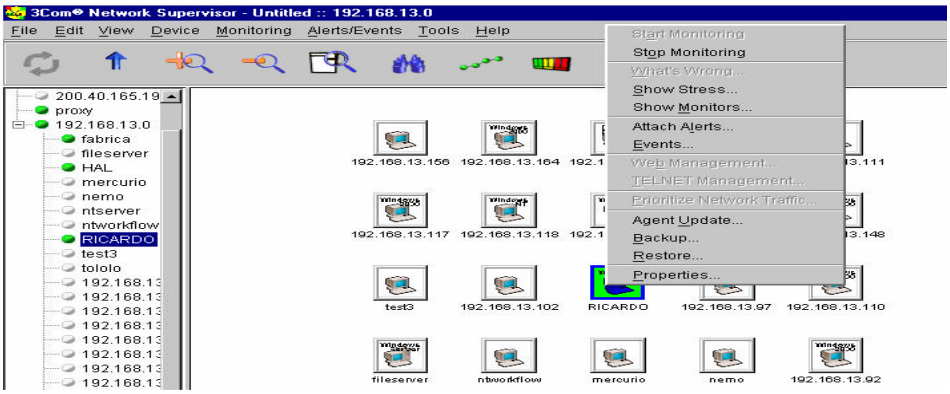


Figura 8.Descubrimiento de nodos red.

Servicios Monitoreados

El siguiente esquema muestra los servicios o ítems que pueden monitorearse sobre cada nodo y cuales tiene disponibles. Con esto se puede tener una idea del estado de cada servicio y de los rangos o umbrales definidos sobre cada uno.
En el gráfico se pueden apreciar los colores que reflejan la gravedad del estado actual de cada servicio.

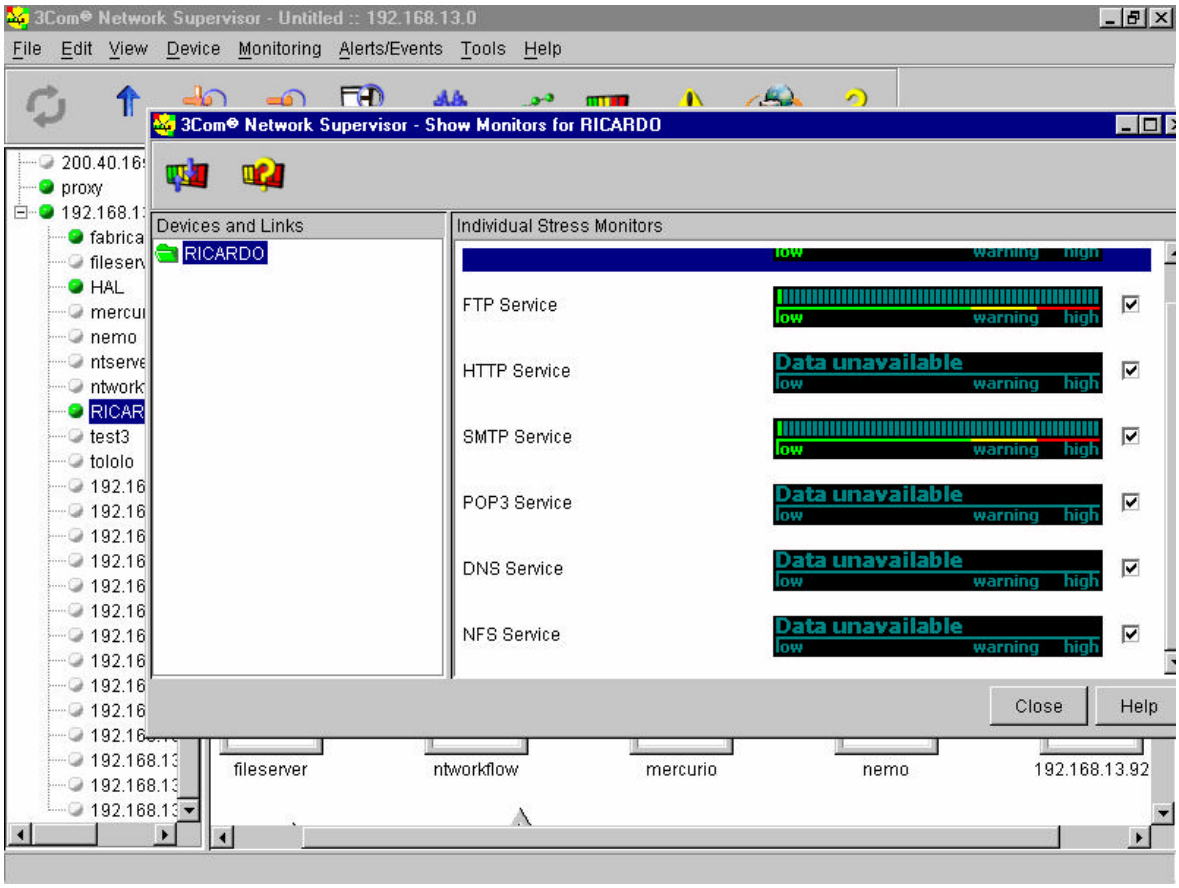


Figura 9.Chequeo de puerto y servicios.

Configuración de alertas y notificación

El siguiente escenario ejemplifica la configuración de una alerta. En el ejemplo se eligió la acción enviar un e-mail, la cual permite configurar quien envía el e-mail, destinatario, copias, día, hora del suceso etc. La parte derecha del escenario varía en función del tipo de alerta elegido.

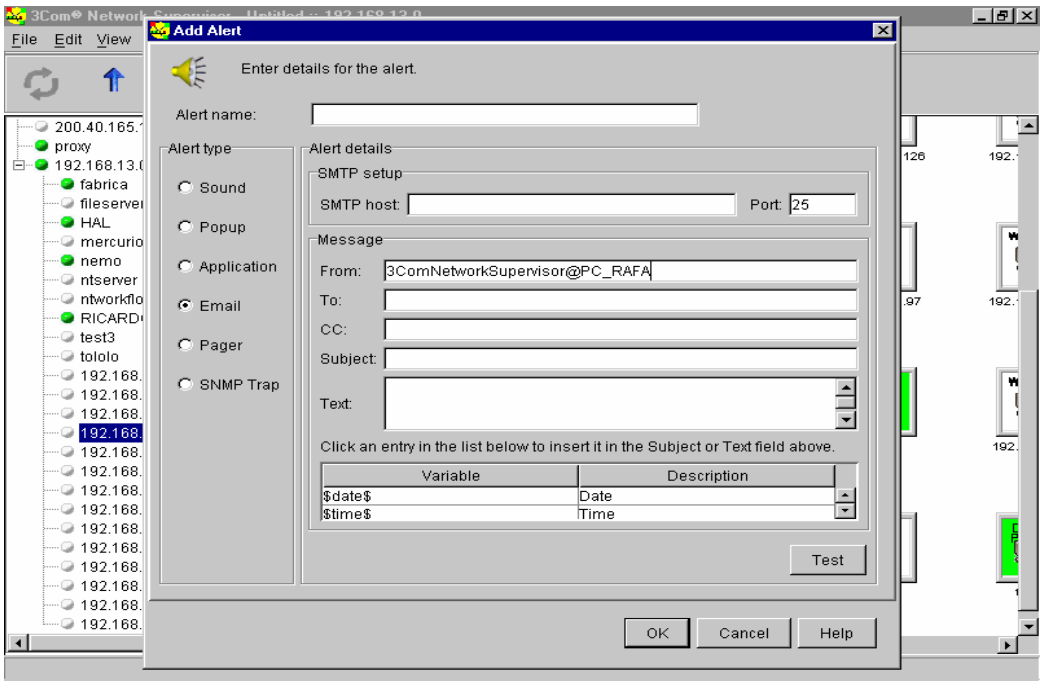


Figura 10. Configuración de Alertas.

3.2.2.3.2 Monitor de K&S

En esta sección se presenta una breve reseña de las características y funcionalidades básicas del monitor que la empresa tiene en uso en la actualidad.

Introducción

Es un herramienta diseñada y desarrollada por K&S asociados, con el objetivo de monitorear instalaciones de IT del negocio. El alcance no se limita al monitoreo de dispositivos físicos, sino que también llega al software y a los datos que el software procesa.

Basado en tecnología Microsoft, usa Microsoft Management Console Framework (MMC) y Snap-In, en la consola. Un servicio de NT, el Monitor Engine Manager (MEM) se encarga de tomar medidas y agendar tareas orientadas a monitorear el funcionamiento de una instalación, previamente definida en una base de conocimiento.
El Scheduler se encarga de disparar tareas de muestreo agendadas a frecuencias especificadas por el MEM.
Las muestras tomadas de una instalación son registradas para su posterior análisis y el respectivo control del desempeño, de la instalación objeto del monitoreo.

El producto permite la especificación de alarmas, con acciones configurables asociadas cuando una alarma se activa, y métodos (comandos) a ejecutar desde la consola para los dispositivos visualizados.

El monitor permite el acceso autenticado de usuarios, cada usuario que ingresa tiene un perfil asociado que determina la visibilidad en la consola y las acciones que éste puede realizar.
En la consola se representan todos los objetos asociados al perfil que ingresó con sus componentes, alarmas y acciones disponibles sobre los dispositivos de hardware y software que se están monitoreando.

Funcionamiento

Existe una base de conocimiento donde se definen los objetos a monitorear, sus variables, los instrumentos con que se toman las medidas, así como las alarmas asociadas a cada elemento crítico de la instalación que deba informarse en caso de malfuncionamiento, las acciones que tomará el monitor cuando se disparan alarmas y los métodos disponibles desde la consola para cada uno de los componentes.

Los objetos a monitorear se denominan EBSM acrónimo de Entidades Básicas Sujeto de Medición.

Es preciso definir, si los EBSM son multi-instancias y como se identifican cada una de ellas, estos valores con sus respectivos íconos se podrán visualizar en la consola.

La herramienta permite que sobre los objetos se definan alarmas y métodos. Las alarmas son expresiones lógicas en base a las mediciones realizadas que disparan automáticamente acciones. Los métodos son acciones como las que disparan las alarmas pero estos son disparados de forma interactiva desde la consola de usuarios, para un objeto seleccionado.

Instrumentos de medición y acciones disponibles

1. Instrumentos de medición

- **SNMP** – Simple Network Management Protocol. Acceso a datos del MIB.
- **Consultas OLE-DB**. Acceso a información en bases de datos y soportes de datos que admiten OLE-DB
- **Ping**. Uso del PING para verificar visibilidad de red.
- **Echo Test**. Envío de mensajes de echo a sistemas y dispositivos que soportan esta característica.
- **Peticiones de Middleware**. Invocación de peticiones de Middleware en instalaciones que tengan dicho software.

2. Acciones asociados con alarmas

- Notificaciones vía e-mail.
- Ejecución de peticiones de Middleware.
- Ejecución de comandos sobre el Service Control Manager de Windows NT.
- Ejecución de procesos RPC.
- Ejecución de COM, DCOM y Aplicaciones.
- Registro (logging) de información.

Instalación y funcionamiento

Se instaló la última versión del K&S Monitor disponible (K&S Monitor setup 1.5) en una máquina con WindowsNT y se seleccionaron tres elementos para monitoreo:

- PING a una máquina de la red (nemo).
- Chequear que el servicio K&S Middleware que se inició previamente en una máquina de la red (nemo) esté activo.
- Chequear que el servicio K&S Launcher que se inició previamente en una máquina de la red (nemo) esté activo.

A cada uno de los elementos se le definió la acción envío de mail, de forma que si se diera la condición de alarma se enviaría un mail a una cuenta específica notificando del error. A su vez la consola de monitoreo de la herramienta desplegaría todos los elementos de monitoreo con un color indicando su estado, verde indicando que el chequeo fue satisfactorio, rojo indicando que se dio una condición de error. Todo esto se configuró en tablas de la base de conocimiento asociada al monitor, por medio de herramientas de configuración que cuenta la empresa, de las cuales se tenía buen manejo.

Para realizar las pruebas en principio se mantuvo el servicio de Middleware activo y el servicio de Launcher bajo, y se observó el estado de la consola para la situación descripta.
En la figura siguiente se puede apreciar.

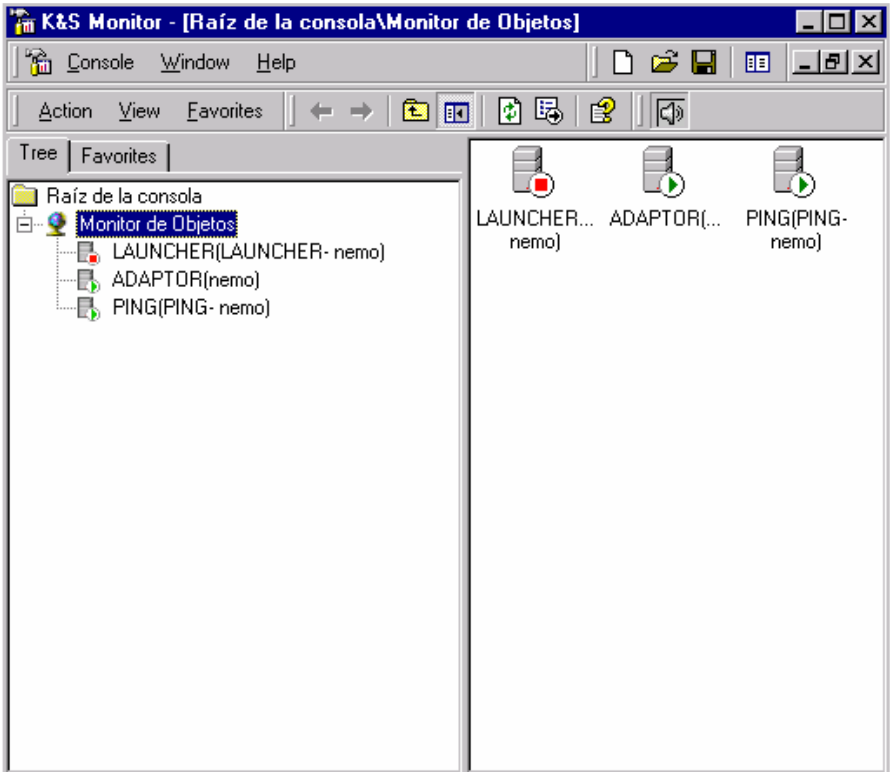


Figura 11. Consola de monitoreo.

Luego se detuvo el servicio de Middleware. En consecuencia, se observó el nuevo estado de la consola y se recibió el correo electrónico en la cuenta especificada notificando que el servicio no estaba activo.

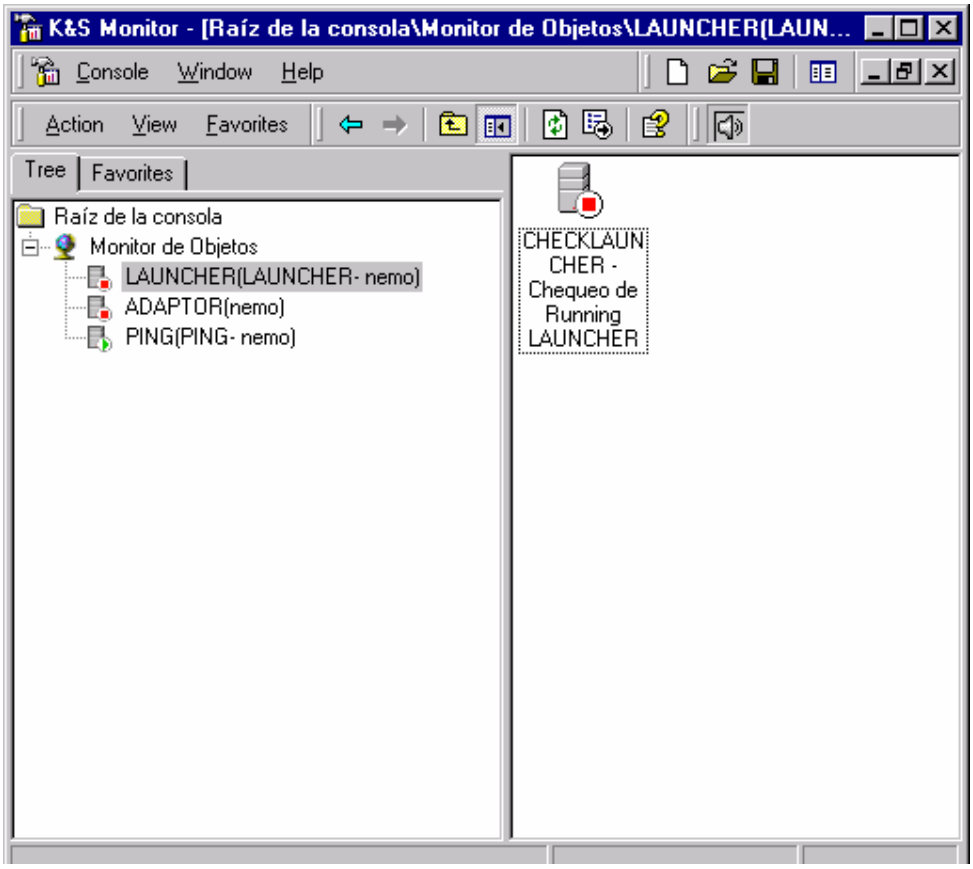


Figura 12. Cambios de estado de un nodo.

3.2.3 JMX

La tecnología Java Management Extensions (JMX) de la empresa Sun Microsystems, se presenta como una potente herramienta base para desarrollar productos enfocados a la gestión de recursos.
JMX define la arquitectura, los patrones, la API, los servicios y patrones de diseño para aplicaciones, monitoreo y gestión de red utilizando el entorno de programación Java.

Es de destacar que esta especificación permite definir los recursos a administrar en una forma simple y amplia. Es posible definir como manejar dichos recursos y luego especificarlos en una forma acorde a la especificación que define JMX.
Permite definir Agentes y gestores en Java, implementar aplicaciones de gestión distribuida e integrar estas a otras aplicaciones existentes utilizando APIs que implementan protocolos estandarizados.

Dentro de las posibilidades que JMX ofrece se puede mencionar: monitoreo de recursos tanto de hardware como software, comunicación remota ya sea vía RMI, HTTP y SOAP, facilidades para implementar servicios de notificación y registro de eventos, posibilidades de interactuar con aplicaciones hechas en java y también la posibilidad de interactuar con aplicaciones no puramente Java por medio de JNI. Permite integrar soluciones de gestión existentes con el uso de la API apropiada (por ejemplo SNMP, WBEM, TMN).

A continuación se presenta una breve reseña de JMX y sus principales características para luego pasar a definir los componentes básicos del sistema en función de esta especificación.

3.2.3.1 Arquitectura de JMX

JMX se construye sobre una arquitectura de tres niveles (Instrumentación, Agentes, Administración), también se permite una suerte de capa que permite el dialogo e integración con la API para el manejo de protocolos adicionales de administración (como ser SNMP, TMN, CIM).
Cada uno de estos niveles puede ser utilizado en forma independiente de los otros, lo que permite una gran flexibilidad.

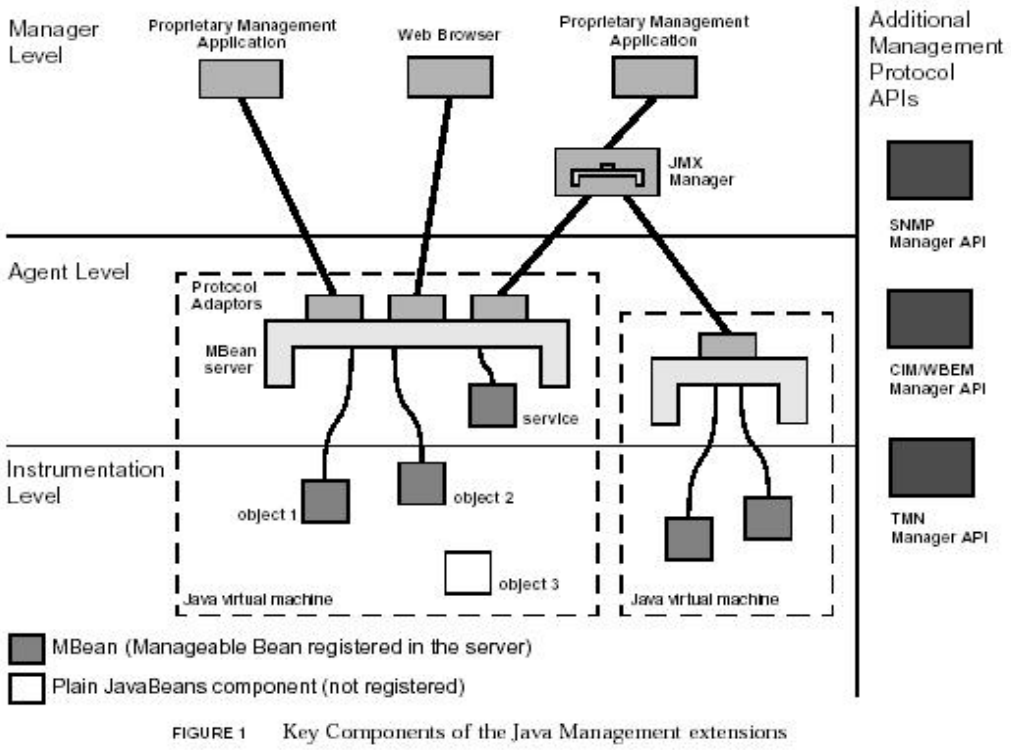


FIGURE 1 Key Components of the Java Management extensions

Figura 13. Arquitectura JMX.

Nivel de Instrumentación (Instrumentation Level): este nivel proporciona una especificación para implementar recursos gerenciables a través de JMX. Estos recursos pueden ser aplicaciones, la implementación de un servicio, puerto, usuario o pieza de hardware, u otros recursos manejados por medio de otros protocolos de administración por medio de una API específica provista por éstos.

Nivel de Agentes (Agent Level): en este nivel se proporcionan los agentes de administración. Los Agentes JMX son el servicio central de la administración que provee JMX. Éstos pueden ser extendidos dinámicamente incorporando nuevos recursos JMX.

Nivel de Administración (Manager Level): componentes que pueden operar como agentes o como administradores en los cuales se pueden distribuir o consolidar todos los servicios de administrados.

Los niveles de Agentes y Administración permiten facilidades para administrar recursos por medio de la aplicación de la tecnología Java, mientras que el nivel de instrumentación permite gestionar la tecnología Java.

3.2.3.2 Componentes de JMX

Se distinguen tres componentes básicos:

- Recurso a ser administrado
- Agente
- Administrador (Manager)

Recursos gerenciables: es el recurso a administrar, el cual ha sido implementado de acuerdo con la especificación provista por el nivel de instrumentación de JMX. Éste debe ser desarrollado en Java completamente o debe implementarse un nexo entre el recurso real y la plataforma.
El término empleado para referirse a estos recursos es Mbean (Manager Bean) y una vez definidos son fácilmente acoplables en cualquier agente JMX.

Agente JMX: es la unidad de management implementada de acuerdo a la especificación de la arquitectura. Está compuesto por un servidor (MBeanServer), un conjunto de MBeans que representan los recursos a ser manejados, y una lista de adaptadores o conectores sobre protocolos de comunicación que permiten a las aplicaciones de management acceder al Agente y manipular los Mbeans que éste contiene.

MBeanServer: oficia como registro de los MBeans en el agente y es el componente que proporciona los servicios que permiten la manipulación de los MBeans.

Adaptadores y conectores: permiten a las aplicaciones acceder a los agentes JMX y manipular los MBeans que contiene cada agente.
Los adaptadores acceden directamente a la representación de los MBeans en otro protocolo (SNMP, HTML). Se implementan del lado del agente y este se comunica con la aplicación correspondiente por medio de los protocolos específicos.

Los conectores tienen dos componentes, uno en el agente y otro en el server, permiten establecer la comunicación punto a punto con componentes remotos con los agentes sobre varios protocolos (RMI, HTTP, HTTPS, IIOP). Estos están basados en interfaces lo que brinda gran flexibilidad a la hora de elegir el conector apropiado según la aplicación a implementar

Manager JMX: provee una interfase para que las aplicaciones puedan relacionarse con los agentes y distribuir o consolidar la información a administrar. La posibilidad de controlar varios agentes permite simplificar estructuras de administración distribuidas y complejas.

Servicios de administración: tanto Agentes como Managers JMX integran servicios que poseen e inteligencia propia. Estos servicios habilitan a los agentes a manejar sus propios recursos y a los managers intercambiar información entre los agentes y las aplicaciones de administración.
Los agentes y managers pueden manejar sus propias alarmas y tareas especificas, lo cual permite reducir la carga en la red.
Estos servicios también se modelan como MBeans, por lo tanto pueden agregarse y removerse a medida que se necesiten.
La especificación JMX define la interfase básica para que estos servicios sean registrados como MBeans. Ajustándose a esto se pueden incorporar nuevos servicios.

API's de administración de Protocolos adicionales: Proveen un mecanismo estándar para que las aplicaciones JMX interactúen con otras tecnologías de administración. Una aplicación debería usar una de estas API's para acceder a estos sistemas y representar sus recursos gerenciables como un recurso JMX para que sea manejado a través de un agente JMX.

3.2.3.3 Elementos centrales de JMX

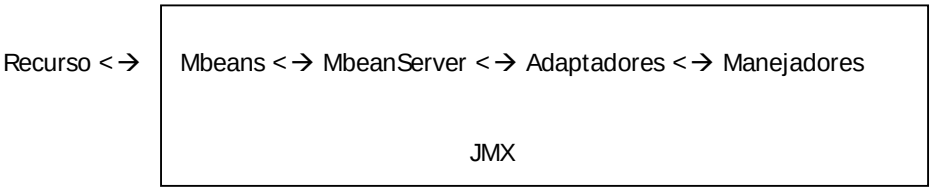
3.2.3.3.1 MBeans

Los elementos básicos para modelar los recursos a ser administrados son los Mbeans. Éstos pueden ser de cuatro tipos: Standard Mbean, Dynamic Mbean, Open Mbean, Model Mbean, aunque los tipos básicos de Mbean son estándar y dinámicos.

La especificación JMX define un Mbean como una clase concreta que:

- Tiene uno o más constructores públicos.
- Implementa su propia interface Mbean o la Interface DynamicMBean
- Opcionalmente implementa la interface NotificationBroadcaster

El esquema básico de como se modela y se administra un recurso en el entorno JMX es el siguiente.



El Mbean instrumenta la interfase con los recursos manejados, esto consiste en definir atributos, operaciones y notificaciones que son usadas para administrar cada recurso (MBeanInfo).
Estos están soportados en la capa de implementación de JMX.

Standard Mbean: Puede ser un javaBean o un javaBean style registrado en el MbeanServer. Los Mbean y sus interfases se definen en tiempo de desarrollo y deben implementar la interfase classnameMBean. Si las clases que implementan los recursos de una aplicación particular a administrar son compatibles con la definición de alguna interfase provista por un MBean entonces estas se pueden registrar directamente en el MBeanServer.

Dynamic MBean: Permite definir o generar la interfase en tiempo de ejecución, esto puede ser posible por medio de un wrapper para recursos no Java o para un estilo no-Bean.
Definen su propia interfase de management en tiempo de ejecución pero mantienen sus propios objetos MbeanInfo. Las implementaciones de los Mbean dinámicos se pueden desarrollar por las aplicaciones directamente.

Open Mbean: Funcionan como los Mbeans dinámicos pero con una restricción adicional de la cantidad de tipos de datos que pueden manejar. De esta forma de elimina la necesidad de uso de carga de clases (class loading) facilitando el desarrollo en sistemas distribuidos.
Los tipos de datos permitidos son:

Tipos primitivos: int, boolean, float, double, etc.
Clases wrappers para tipos de datos primitvos: Integer, Boolean, Float, String, etc.
Tablas: arrays de filas del mismo tipo.
Compuestos: tipos de datos compuestos.

Model Mbean: Es una extensión de un Mbean dinámico, pero se deben implementar completamente. Este más que un conjunto de interfases define como se debe escribir completamente lo que permite que sea fácilmente personalizado y estandarizado. Da soporte para requerimientos tales como persistencia, registro, y administración de servicios varios.

Está presente en todas las implementaciones de Agentes JMX lo que permite su implementación en pocas líneas de código, casi exclusivamente en la configuración.
Son atractivos de usar por las siguientes características:

- Fácil y simple uso: al estar implementados dan soporte en varios aspectos como registro, notificaciones, manejo de errores, persistencia, etc.
- Fácil de adaptar a herramientas de desarrollo: son completamente configurables por medio de metadatos lo que simplifica su especificación con un archivo XML o una API, estos son fácilmente manejables con herramientas de desarrollo.
- Encapsula servicios estándar: soporta o implementa un grupo de actividades que pueden ser implementados o personalizados según la necesidad del desarrollador.
- Comportamiento dinámico: permite definir o modificar su comportamiento en tiempo de desarrollo o en tiempo de ejecución. ModelMBeanInfo soporta API´s para personalizar el comportamiento y las interfaces de gestión del Model MBean.

Estos son los cuatro mecanismos básicos para modelar y administrar recursos mediante el desarrollo de aplicaciones con la tecnología JMX.
Para hacer posible la administración de éstos con una aplicación JMX se debe considerar lo siguiente:

- Diseñar los MBean que modelan de la mejor forma posible los recursos en cuestión
- Crear los MBean correspondientes
- Registrarlos en el MBeanServer

3.2.3.3.2 MBeanServer

Como se ha mencionado, un MBean representa el recurso que el sistema de gestión debe monitorear y controlar. A modo de ejemplo, procesos en ejecución, dispositivos, equipos, volumen de datos de entrada o salida de una interfase de red, espacio en discos, etc.

Se conoce como interfase de gestión al conjunto formado por los constructores del MBean, los atributos, las operaciones definidas por la interfase de este y las notificaciones que este genera.
Esta interfase refleja el comportamiento de las clases java que modelan el MBean, pero éstas no deben estar disponibles para el sistema de gestión.

Para poder acceder a los datos de cada MBean disponible en el sistema es necesario disponer de algún mecanismo que lo permita, éste es el MBeanServer.
El sistema tiene acceso a los datos de los MBean por medio del MBeanServer, tiene acceso directo a la interfase del MBeanServer pero necesita comunicarse con la interfase de gestión de los propios MBeans para poder manejar los recursos modelados por los MBeans.

Una de las principales funciones del MBeanServer es la de permitir el acceso a las interfases de gestión de los MBean por medio de la propia interfase estándar de gestión.

El MBeanServer es el componente central de la capa de Agentes JMX ya mencionada, este provee un registro con un mecanismo predefinido y estandarizado de nombrado para los MBean que son usados para controlar y monitorear los recursos JMX manejables.
Todas las operaciones o consultas a ejecutar sobre los MBeans se administran vía el MBeanServer. Este provee métodos para crear, manipular MBeans y también para recuperar información sobre estos (query métodos).

El acceso a este puede ser local o remoto, el acceso remoto se define según una API estándar JSR 160(Java Specification Request).

Para que un MBean esté disponible para interactuar con las aplicaciones de gestión, este debe ser incorporado en un registro del MbeanServer. Cuando esto pasa los métodos, atributos y operaciones de su interfase son accesibles por medio de métodos del MBeanServer.
Se puede decir entonces que un MBean cumple con un cierto ciclo de vida el cual es modelado con métodos del MBeanServer.

- **Instanciación**. Para poder registrarlo este debe ser instanciado invocando sus constructores
- **Registración**. Registro del MBean con su correspondiente objectName
  Creación. Invocación de los objetos registrados cuando es necesario, en forma automática
  Interfase para registración. Se dan métodos para conocer el estado del MBean en un momento dado, antes y después de registrarlo o desregistrarlo
- **Desregistración**. Se quita del registro.

El MBeanServer también es responsable por la administración de las notificaciones asociadas a cada MBean, permite enviar las notificaciones a los procesos que están escuchando por ellas (listeners).

3.2.3.3.3 Conectores y adaptadores

Es necesario definir e implementar mecanismos que permitan la comunicación entre el MbeanServer y los agentes que interactúan con las aplicaciones de gestión y que éstos sean estandarizados.
Estos mecanismos son los conectores y adaptadores.
Un conector expone una interfase similar a la de un MbeaServer para la invocación remota y manejar los requerimientos que son hechos por los procesos remotos que administran los agentes, estos respetan la semántica definida por JMX en el MbeanServer.

La diferencia básica entre los conectores y adaptadores es que los últimos adaptan los medios de comunicación a protocolos que tienen una semántica no definida en los estándares de JMX. Los agentes remotos implementan la comunicación con aplicaciones que poseen mecanismos propios, esto hace necesario trasladar la implementación de las vías de comunicación JMX para poder interactuar con estos protocolos distintos (tales pueden ser, SNMP, SOAP, HTTP, etc).

La implementación de conectores JMX tiene dos componentes fundamentales:

- Conector representado por un Mbean registrado en el MbeanServer que declara el conector y tiene la funcionalidad de servidor lógico entre los agentes, implementa el mecanismo de comunicación entre éstos y el componente servidor JMX mencionado anteriormente.

- Cliente lógico. Conector que canaliza envíos y recepción de requerimientos entre el servidor y un agente JMX.

La comunicación entre estos dos componentes es una conexión punto a punto.

3.2.3.3.4 Monitores

El manejo de control de los valores definidos para los atributos es realizado por otro componente de la implementación JMX que es la entidad o interfase Monitor, esta tiene varias implementaciones que se comportan distinto, y tienen diferente implementación según el tipo de datos del atributo a monitorear, estas son String Monitor, Gauge Monitor, Counter Monitor.
Estos monitores se definen con el mismo esquema que los MBeans (para JMX son MBeans), es decir, están definidos por una interfase general MonitorMBean que extiende la clase abstracta Monitor encapsulando los elementos comunes del comportamiento, a su vez se especializan con la definición de la interfase correspondiente implementando los aspectos particulares. Por ejemplo, StringMonitorMBean y se implementa en la clase StringMonitor con los métodos específicos para manejar el monitoreo de atributos del tipo String.

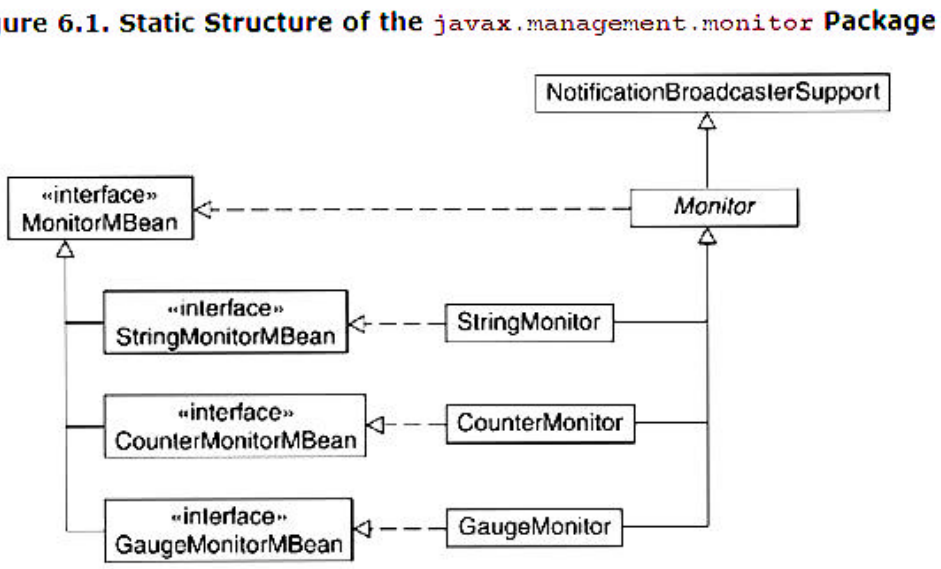


Figura 14. Estructura de la implementación de monitores.

Las propiedades comunes que maneja el MBeanMonitor son las siguientes:

Atributos comunes:

- Granularidad. Valor numérico que define la frecuencia entre observaciones medido en milisegundos.
- Atributo Observado. Nombre del atributo que se esta monitoreando.
- Objeto Observado. Nombre del MBean cuyo atributo esta siendo monitoreado.
- Activo. Indica si esta activo o no.

Operaciones comunes:

- Activar. Activa el monitor para comenzar a chequear el atributo observado.
- Detener. Detiene la actividad de monitoreo.

A continuación se da una breve descripción de los tipos de monitor y sus características generales.

StringMonitor. Este tipo de monitor chequea el valor de un atributo en un MBean de tipo String pudiendo controlar dos casos, uno si el valor medido del atributo coincide con un valor prefijado y otro si el valor no coincide.
Se maneja los siguientes atributos propios del tipo de monitor y notificación:

El ultimo valor observado de l atributo
Tiempo en milisegundos desde la última notificación
Notificación por diferencia. Si es verdadero se envía notificación por diferencia
Notificación por igualdad. Si es verdadero se envía notificación por igualdad

El uso de estos dos últimos atributos esta definido por la conveniencia según el caso particular de lo que se quiera controlar para el atributo, si se quiere controlar que el atributo es igual a un cierto valor o si hay diferencia.
También las notificaciones que maneja son por diferencia o igualdad del atributo con un valor.

CounterMonitor. Se chequea si el valor del atributo no pasa de cierto tope prefijado, los atributos monitoreados varia el valor en forma monótona con el tiempo
Maneja los siguientes atributos particulares:

Valor actual del atributo observado.
Tiempo en milisegundos desde la última observación.
Modo diferencia. Atributo booleano que indica si el atributo "valor actual" la diferencia con el valor medido en la observación anterior (verdadero) o si solamente es el valor medido (falso)
Módulo. Número de observaciones necesario antes para reinicializar el contador.
Notificación. Decide cuando si se envía una notificación o no al exceder el valor tope.
Tope. Valor tope a comparar con el valor medido del atributo.

Este tipo de monitor en su funcionamiento más simple recoge datos del atributo en forma periódica y cuando se excede el valor tope del contador se envía una notificación.

GaugeMonitor. Chequea el valor de un atributo numérico cuyo valor puede crecer o decrecer en el tiempo, se puede usar para chequear que el valor se mantenga dentro de un rango aceptable de valores.

Valor actual del atributo observado.
Tiempo en milisegundos desde la última observación.
Modo diferencia. Atributo booleano que indica si el atributo "valor actual" la diferencia con el valor medido en la observación anterior (verdadero) o si solamente es el valor medido (falso)
Tope máximo. Valor tope máximo a comparar con el valor medido del atributo.
Notificación por máximo. Decide si se envía una notificación o no al exceder el valor tope.
Tope. Valor tope a comparar con el valor medido del atributo.

Notificación por mínimo. Decide si se envía una notificación o no al decrecer a menos del valor mínimo.
Tope. Valor tope a comparar con el valor medido del atributo.
Este tipo de monitor maneja dos tipos de notificaciones, una si se sobrepasa el máximo valor y otra si el atributo monitoreado toma valores por debajo del tope mínimo.

Es responsabilidad de los monitores alertar a las aplicaciones de gestión cuando de dan condiciones de excepción en los atributos que están siendo monitoreados, el mecanismo natural que se da JMX para resolver esto es la notificación.

3.2.3.3.5 Notificaciones y "listeners"

Dentro de la información genérica de notificación que manejan los monitores se pueden mencionar los atributos:

- Valor de la medida: valor medido del atributo.
- Nombre del atributo observado: nombre del atributo.
- Nombre del atributo observado nombre del objeto propietario del atributo.
- La condición de disparo de la notificación: condición dad para el disparo de la notificación.

A su vez cada tipo particular de monitor maneja tipos de notificación particulares que son enviados y que dependen del tipo de monitor.

El StringMonitor maneja la notificación si el atributo coincide con un valor o si hay diferencia.
El CounterMonitor si el valor del atributo supera cierto valor tope fijado
El GaugeMonitor maneja dos tipos, si se supera el valor máximo o si se baja del valor mínimo
También se manejan notificaciones ante varios tipos de errores.

Una vez que un monitor queda activo es necesario contar con un mecanismo que comunique a la aplicación de gestión la información necesaria sobre las condiciones de excepción que se den respecto al atributo que se está monitoreando. Esto se logra con un manejador de notificaciones en conjunto con los procesos listeners (escuchan).

El manejador de notificaciones canaliza las notificaciones hacia los procesos declarados como listeners que son los que se comunican con los procesos de gestión o derivan la información hacia los mecanismos de acción predefinidos.

3.2.4 Resultados y conclusiones

Dentro de los objetivos planteados para esta etapa se ha completado el documento esperado, en el que se describen una serie de herramientas estudiadas.

Se encontró gran cantidad de información relacionada al tema de gestión y administración de recursos. Tomando en cuenta el conocimiento respecto al tema antes de abordar el taller, se puede evaluar que realmente fue positivo el hecho de investigar distintas soluciones y aplicaciones. Esto dio pautas sobre cómo encarar o pensar en las etapas sucesivas del desarrollo, así como también determinar qué factores sería importante tener en cuenta y cuales serían secundarios.
Luego de esta etapa se comprendió mejor el problema a resolver y se logró un acercamiento con un conjunto de herramientas que planteaban distintas soluciones al problema.

En cuanto a la formación de los integrantes del grupo se tomó contacto con el entorno de programación Java, plataforma de desarrollo en la cual se desarrollaría el producto y en forma conjunta con esto se llegó a tener experticia en el uso y resolución de problemas de configuración de los entornos de desarrollo utilizados, entre estos NetBeans y Eclipse.
A su vez se cierra esta etapa con cierta fluidez en el manejo del sistema operativo Linux, sobre el que no se tenía mucha experiencia previa.

Se investigaron y se realizaron pruebas con varias herramientas de administración de recursos de red, las cuales algunas quedaron instaladas en la empresa y se están usando en la actualidad, como es el caso de MRTG y agentes SNMP.

Dadas sus características, se estudió en profundidad las posibilidades de la tecnología JMX lo cual fue muy positivo para el desarrollo en las siguientes etapas.

3.3 ANÁLISIS DE REQUERIMIENTOS

3.3.1 Objetivos

El principal objetivo en esta etapa fue definir y especificar el alcance de la herramienta a construir. Se determinó un conjunto de funcionalidades básicas, recursos a monitorear, mecanismos de acciones y alertas, registro que ésta debería incluir.

Se esperaba como resultado un documento de Análisis de Requerimientos que determinara las características del producto y alcance del proyecto, que sirva como punto de partida para el Análisis de Diseño y arquitectura del sistema a desarrollar.

3.3.2 Definición del problema

La gran variedad de soluciones informáticas aplicadas en las empresas requieren un permanente flujo de información que detalle el estado de los diversos componentes de su plataforma informática. Por este motivo es altamente apreciado el contar con herramientas de monitoreo que se adapten a las necesidades particulares.
Tales herramientas de monitoreo deben permitir configurar nuevos recursos a controlar y también definir mecanismos de alerta, y registros.

Una motivación muy fuerte en la empresa está enfocada en construir herramientas que den solución a estos aspectos en dos direcciones. Por un lado acoplar la herramienta a los productos que la empresa provee a sus clientes y por otro ofrecerla en el futuro como un producto independiente del conjunto de soluciones.

3.3.3 Desarrollo.

Se realizaron reuniones con varios miembros de la empresa y los tutores del proyecto para establecer los requerimientos de la herramienta. A partir de estas reuniones se definieron un conjunto de escenarios que planteaban situaciones sujetas a posibles acciones de monitoreo. Éstos no fueron todos los casos posibles, pero se determinó un conjunto representativo y de especial interés para la empresa.
El conjunto de todos los escenarios relevados se encuentra el Apéndice 8.2.
De la lista de escenarios posibles se seleccionó un conjunto más reducido, con el fin de determinar el alcance del proyecto.

La empresa tenía especial interés en desarrollar el motor de la herramienta, más que un conjunto considerable de pluggins de monitoreo, alerta y registro. Por este motivo se hizo hincapié en las características generales que se pretendía tuviera la herramienta. Por características generales se entiende: mecanismo de configuración, mecanismos de muestreo de los datos a monitorear, plataforma en que debe correr, sistemas de registro, etc.

3.3.4 Requerimientos específicos

Como resultado del proceso de análisis de requerimientos se definieron un conjunto de requerimientos que permitieron establecer el alcance real del proyecto y que sirvieron como punto de partida para definir la arquitectura y diseño de la herramienta.

El interés fundamental para este proyecto es el desarrollo del prototipo del motor de la herramienta de monitoreo, alarmas, y registro. En este sentido el principal objetivo fue la construcción de una herramienta que cumpla con las siguientes características generales:

- Permita realizar el monitoreo de recursos.
- Posibilidades de manejo de alarmas.
- Manejo de Registro (logging).
- Independiente de la plataforma.
- Fácilmente extensible.
- Fácilmente configurable (ya sea poder configurar nuevos elementos sujetos a monitoreo, alarmas según niveles de severidad establecidos, etc.)
- Capacidades diversas de interacción con los usuarios (ej: visual, sonido, llamada telefónica, mail, etc.)

3.3.4.1 Requerimientos Funcionales

Los siguientes requerimientos funcionales describen el conjunto de funcionalidades que se pretenden desarrollar en el proyecto. A éstos se les estableció una prioridad. Se dividieron en requerimientos mínimos y deseables. Como el objetivo principal era definir el motor, se prefirió seleccionar un conjunto mínimo de requerimientos funcionales, que se deberían implementar para completar la herramienta a desarrollar, y otro conjunto denominado requerimientos deseables, que sería bueno implementar durante el proyecto si los tiempos lo permitían.

3.3.4.1.1 Monitoreo y registro

1. Mínimos
    a. Monitoreo del estado de una máquina, porcentaje en uso de CPU, consumo de memoria, espacio en disco. Se considerara la implementación para el sistema operativo Linux.
    b. Monitoreo de dispositivos de red.
    c. Monitoreo de datos sobre procesos en una máquina, uso de CPU, consumo de memoria, chequear si dicho proceso esta en el estado adecuado. Se considerara la implementación para el sistema operativo Linux.
    d. Monitoreo y análisis, búsqueda y reconocimiento de patrones en archivos (txt, xml).

2. Deseables
    a. Monitorear el estado de una máquina con las mismas características que las implementadas para Linux, pero en la plataforma Microsoft.
    b. Monitorear procesos con las mismas características implementadas para Linux, para la plataforma Microsoft.
    c. Monitoreo y análisis de bases de datos.

3.3.4.1.2 Alarmas

Por alarma se entiende una condición que permiten establecer el estado no deseado de un recurso. Mediante la evaluación de la condición de alarma en el contexto adecuado se puede determina el estado del recurso. Cuando se cumple la condición se dice que se genera una alarma.

1. Mínimos
    a. Se podrán especificar expresiones booleanas de un solo operador, de los tipos:
    <, <=, >, >=, = .

2. Deseables
    a. Especificar expresiones booleanas compuestas con más variedad de operadores.

Una expresión booleana consiste de la combinación de operadores lógicos y aritméticos, funciones y variables, cuya evaluación resulta en un tipo boolean (*true* o *false*).

3.3.4.1.3 Acciones

Operación que permite reaccionar ante la notificación de una condición de alarma. Ejemplos de acción son envío de mail, llamada telefónica, registro en archivos, etc.

1. Mínimos
    a. Envío de correo electrónico.
    b. Ejecutar acciones sobre los dispositivos monitoreados mediante SNMP (Start , Stop)
    c. Señalización de procesos en Linux (Kill, Signal)

2. Deseables
    a. Llamadas telefónicas, opcionalmente mensajes SMS.

3.3.4.1.4 Registro

1. Mínimos

    a. Registro histórico de sucesos en archivos de texto y opcional en bases de datos.
    b. Mantener archivos de log de acuerdo a un período configurable, a modo de ejemplo si el período es una semana y se mantienen archivos diarios, solo se almacenan uno por cada día y se sobrescribe un archivo por día. La forma de escritura de archivos sería circular.
    c. Se estudiará la posibilidad de mantener históricos de promedios. Es decir, almacenar el promedio de un conjunto de medida a intervalos regulares de tiempo.

3.3.4.2 Requerimientos de Interfase

3.3.4.2.1 Interfase con el usuario

1. Visualizar los ítems monitoreados. Es necesario representar en una consola gráfica los elementos monitoreados. Se debe poder visualizar información sobre el estado en tiempo real y configuración de cada ítem monitoreado. De acuerdo a las propiedades de cada ítem se debe presentar la posibilidad de poder analizar históricos de datos asociados a dicha propiedad.

2. Se considerará para la etapa de diseño el manejo de vistas asociadas a permisos sobre el usuario que opera la interfaz gráfica, dicho usuario tendrá asociado un rol específico y de acuerdo a este rol se le permitirá ver y actuar sobre un conjunto determinado de ítems de monitoreo. Este requerimiento no entra dentro del alcance del proyecto.

3. Configuración
   En forma opcional se puede implementar una interfaz para la especificación y configuración de los ítems a monitorear, las alarmas que se dispararán, las acciones que se realizarán. Es recomendable que esta sea clara y amigable facilitando su uso, y permitiendo tener un total control sobre la herramienta.
   Estos escenarios no están exigidos como obligatorios en el marco de este proyecto, y las configuraciones básicas serán especificadas en archivos.

3.3.4.3 Requerimientos sobre Atributos del Software

En esta sección detallaremos los atributos con los que deberá contar la herramienta.

1. Mantenibilidad

   La mantenibilidad es imprescindible, su buscará el desarrollo modular y dado que se necesitan pluggins fácilmente acoplables se pondrá especial interés en facilitar esta operación.

2. Confiabilidad

   La herramienta deberá ser confiable en el sentido que los usuarios puedan depender de todos los datos generados por ésta.

3. Robustez

   En caso de que ocurra alguna situación imprevista la herramienta deberá notificar la misma, sin emitir datos de los cuales no se pueda confiar.

3.3.4.4 Actores

Distintos tipos de actores que utilizarán la herramienta.

Es posible identificar distintos roles según la relación de los usuarios con la herramienta a construir. Entre éstos podemos mencionar: Configurador, Administrador y Operador.

- Configurador. Encargado de la especificación y configuración. Para esto debe tener conocimientos técnicos en el área de aplicación de la herramienta.
- Administrador. Encargado de recibir las notificaciones que el monitor pueda generar, por ejemplo vía e-mail, llamada telefónica. No es necesario que tengan una formación avanzada en informática. Debe ser capaz de interpretar los mensajes recibidos o los archivos de registro generados.
- Operador. Interactúan con la interfase gráfica para visualizar los elementos monitoreados. Deberán ser capacitados para su utilización, pero no necesitarán conocimientos o formación avanzada en informática.

3.3.5 Resumen del trabajo

Se encontró especial dificultad al definir el alcance de la herramienta en lo que respecta a funcionalidades. El concepto “genérico” por su dimensión hacía difícil especificar casos de uso concretos los que se aclaraba no eran la parte esencial del producto a construir.

La dirección de trabajo principal fue definir o especificar una herramienta tan general y extensible que dificultaba la definición y elección de los pluggins de monitoreo a implementar.

Una vez aclarado esto se define una serie de casos específicos a implementar como un subconjunto de los casos de uso relevados, en el Apéndice 8.2 se detalla la totalidad de los casos de uso relevados.

3.4 ESPECIFICACIONES DE DISEÑO Y ARQUITECTURA

3.4.1 Objetivos

Definir la arquitectura y el diseño de la aplicación tomando como base los resultados obtenidos en etapas anteriores, y que sirviera como guía para la etapa de implementación.

3.4.2 Visión general

Esta etapa fue una de las más importantes del proyecto ya que en ella se estudió y decidió la arquitectura de la herramienta. Durante el desarrollo de la misma se encontraron ciertas dificultades que determinaron que fuera una etapa más larga de lo que se había estimado en el cronograma inicial.

Como punto de partida se intentó reconocer los componentes básicos que poblarían el sistema. Este proceso no fue exhaustivo, el objetivo fue definir un conjunto de componentes básicos y comenzar a analizar sus posibles relaciones y distribución como base para el desarrollo de la arquitectura.

En función de estos componentes y tomando como base la arquitectura del monitor actualmente en uso en la empresa, se plantearon cuatro arquitecturas posibles de las cuales se analizaron sus ventajas y desventajas.

Una vez definida la arquitectura y con el diseño de clases ya casi finalizado se toma contacto con la tecnología JMX [JMX].

Éste fue un punto crítico desde varios puntos de vista.

El hecho de que el diseño de clases estuviera avanzado era un factor de duda sobre el hecho de invertir tiempo en el estudio de esta tecnología. La bibliografía consultada destacaba que esta tecnología fue desarrollada justamente para el desarrollo de aplicaciones de administración y gestión, por lo cual se decidió investigarla más en detalle.

En principio se dedicó un tiempo a profundizar en la posibilidades de JMX, la documentación consultada daba idea de resolver un conjunto de problemas ya evaluados en forma relativamente sencilla. Entre éstos, mecanismos de comunicación entre componentes, mecanismos de definición de nuevos pluggins de recursos, monitoreo y alertas, facilidad para extender las funcionalidades, etc.

Al terminar el período de evaluación, para el que se dedicaron entre dos y tres semanas, se tenía la seguridad de que con la aplicación de JMX se daba solución a todos los problemas planteados.

Un único motivo de resistencia por parte de los estudiantes fue el hecho de que adoptar esta tecnología no fuera un impedimento para posibles mejoras futuras de la herramienta, sobre todo por la eventualidad de que en el análisis se hubiera pasado por alto algún detalle de importancia.

La evaluación terminó con el desarrollo de un prototipo de prueba que permitió verificar ciertos aspectos que se consideraron claves.

Fue necesario buscar implementaciones disponibles de la especificación JMX que fueran de libre distribución. Se encontraron dos implementaciones con éstas características, XMojo y MX4J [MX4J], se decidió utilizar la implementación MX4J (los fundamentos de esta decisión se detallan en la sección 3.4.4.1).

Los resultados fueron satisfactorios en el conjunto de los aspectos a verificar, por lo que se decidió incorporar la tecnología JMX a la arquitectura de la herramienta con la implementación MX4J.

Se entendió adecuado redefinir la arquitectura de manera de incorporar las posibilidades de JMX para aprovechar las características de este marco de trabajo. Algunos aspectos que antes pudieron haber sido muy complejos y no se incluyeron en la arquitectura, con la incorporación de JMX se simplificarían.

El desarrollo de la aplicación de prueba llevó aproximadamente tres semanas, lo que sumado a las tres semanas de investigación de JMX fueron un mes y medio de atraso de acuerdo al cronograma definido.

3.4.3 Reconocimiento de componentes básicos

En el proceso de identificación de los elementos básicos se tomaron como referencia los requerimientos definidos durante la etapa de Análisis de Requerimientos, el relevamiento realizado de las distintas herramientas en la etapa de investigación y también el estudio realizado sobre el monitor existente en K&S.

Tomando como punto de partida el documento de Análisis de requerimientos se identificaron un grupo básico de componentes según su funcionalidad, recursos que se quiere administrar o monitorear, acciones a tomar en caso de ser necesario, entidades que chequean el estado de cada recurso, y otras que evalúan estos estados, mecanismos de registro y clientes que consumen todos estos datos.

Estos componentes son los siguientes:

Ítem de monitoreo (Recurso a monitorear): toda entidad que contenga una propiedad medible. Esta entidad tiene asociadas condiciones de alarma que alertan de un mal funcionamiento o estado indeseable del ítem monitoreado, las cuales serán notificadas mediante alguna acción.

Recolector de medidas: se encarga de medir y recolectar las medidas que se realiza a los ítems.

Evaluador de condiciones: evalúa las condiciones de alarma asociadas al ítem para la medida tomada.

Acciones: encargado de ejecutar acciones. Ejemplos de acciones pueden ser envío de mail, realizar llamadas telefónicas.

Manejo de Registro: encargado del registro de las medidas tomadas o de las acciones en caso de ser necesario.

Consolas gráficas: vistas gráficas que muestran del estado de los ítems de monitoreo.

Mecanismos de conexión: implementan las vías de comunicación entre los distintos componentes.

Para la definición de la arquitectura de la aplicación se evaluaron una serie de aspectos que tienen que ver con la distribución de los componentes básicos definidos y sus funcionalidades. La discusión tomó como punto de partida la arquitectura del monitor actual en uso en la empresa y posteriormente se plantearon distintas posibilidades manejando qué tan distribuida podía llegar a ser o como distribuir los componentes en la misma.

3.4.4 Arquitectura

Se manejaron cuatro posibles arquitecturas buscando definir parámetros que permitieran determinar la arquitectura que se ajustara mejor con los requerimientos definidos, y a su vez que estuviera dentro del marco del proyecto y sus posibilidades.

La discusión fue hecha desde un punto de vista teórico y sin tener referencias como para poder evaluar las distintas posibilidades de organización entre los posibles componentes del sistema.
Uno de los casos evaluados fue la arquitectura del monitor desarrollado en la empresa, el cual presenta una arquitectura centralizada, tanto el monitoreo, evaluación de condiciones de alarma, disparo de acciones, registro, y consola de monitoreo para visualizar el estado de los ítems monitoreados se realiza en una sola estación.

Los puntos centrales en los que se basó la discusión fueron:

- El hecho de que los componentes estuvieran más o menos distribuidos como afectaría en:
  - el desempeño y consumo de recursos.
  - las posibilidades de extender la herramienta.
  - la incorporación de forma sencilla de nuevos módulos de monitoreo, acciones y registro.
  - la complejidad de la solución.
  - la escalabilidad de la herramienta.

- Qué mecanismo registro sería más eficiente, si centralizado o distribuido.
- Manejo de distintas políticas para agendar tareas.
- Posibilidad de manejo de varias aplicaciones cliente independientes. En este punto se tenía conocimiento de la dificultad en el monitor actual para mantener actualizada la información sobre el estado de los recursos a monitorear.

Se puede profundizar esta discusión en el Apéndice 8.3.

Como resultado de la discusión se eligió una arquitectura que se constituye de tres componentes básicos:

Agentes: Éstos mantienen la inteligencia necesaria para agendar con cierta frecuencia los recursos a monitorear, y mantener la política de registro. Además proveen mecanismos de comunicación con el Servidor.

Servidor: Este componente tiene el control de leer la configuración e inicializar todo el sistema, inclusive a los Agentes. Se encarga de evaluar los resultados de las medidas tomadas sobre los recursos, así como la ejecución de acciones cuando se den condiciones de alarma. Mantiene y administra información de todos los recursos que se monitorean en los Agentes, y solo a través de él es que los Clientes acceden a los recursos de los Agentes. También incluye una inteligencia similar a la del Agente, es decir puede monitorear recursos si es necesario. Implementa vías de comunicación con los Agentes y con los Clientes.

Clientes: Componentes que consumen información sobre el estado de los distintos recursos monitoreados ya sea en los Agentes o en el propio Servidor. Éstos pueden ser tanto consolas gráficas o aplicaciones de gestión. Se comunican únicamente con el Servidor donde se concentra la información de todos los agentes.

La siguiente figura resume los componentes de la arquitectura.

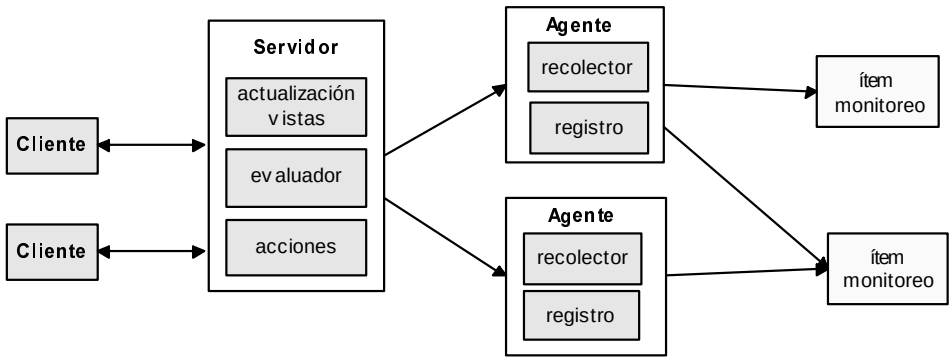


Figura 15. Arquitectura y componentes.

La arquitectura elegida mantiene centralizada gran parte de la inteligencia en el Servidor, pero permite tener distribuida en los Agentes la actividad de monitoreo y los mecanismos de registro de datos, lo cual permitía que sea escalable. Los Agentes son independientes entre sí, si alguno fallara no afectaría en la actividad del resto. Permite mantener varios clientes independientes entre sí, que se comunican únicamente con el Servidor.
La limitante mayor de esta arquitectura es que si ocurre algún daño en el Server se detiene la actividad de monitoreo.
Tomando como punto de partida esta arquitectura se construyó un primer diseño de clases, que se detalla en Apéndice 8.5.

3.4.4.1 Proceso de definición

Como se mencionó en la sección 3.4.2 "Visión general", una duda no menor planteada en ese momento era no perder la posibilidad de futuras ampliaciones del producto por no haber tenido en cuenta detalles en el momento de su evaluación. En otras palabras, que las características de la herramienta no se vieran limitadas por el uso de la tecnología JMX. En un principio la documentación consultada sobre este marco de trabajo permitió suponer que la arquitectura elegida se ajusta a esta especificación, además presentaba facilidades para implementar aspectos de comunicación y definición de nuevos pluggins.
En este sentido se considera que su aplicación provee facilidades para la implantación de la herramienta concentrándonos en los aspectos centrales (definición de los recursos acorde a la especificación, mecanismos de configuración inicial y dinámica, sistemas de registro a emplear, etc).

Para determinar su uso o no, se analizaron tres aspectos básicos:

1. Estudio en profundidad de la plataforma, su arquitectura y posibilidades.
2. Búsqueda de implementaciones de JMX disponibles.
   Se encontraron dos implementaciones de JMX de libre distribución y de código abierto.
   a. XMojo [XMOJO]
   b. MX4J [MX4J]

   Luego de un análisis de las mismas en cuanto a sus posibilidades, se decidió testear la implementación MX4J por ser la que implementa la especificación de forma más completa, brindando mayores posibilidades en cuanto a mecanismos de comunicación y configuración.
   Ésta es una implementación open source de JMX que cumple con JSR 003 y JSR 160. La especificación JMX (JSR 003) actualmente proporciona los métodos para crear agentes basados en Java, a través de técnicas de instrumentación y servicios de agente estándar. Pero no define un estándar para el acceso remoto a dichos agentes.
   El JSR 160 definirá cómo es la API remota de cliente y cómo se comporta. También va a definir protocolos de transporte estándar que debe soportar la implementación JSR para que puedan interoperar distintas implementaciones.
   Java Specification Request 3 era una API que definía operaciones para aplicaciones sobre management basadas en Java locales, lo cual es un problema si se quiere tener un sistema distribuido.
   Por lo tanto, para implementar esto se crean o se permiten crear conectores en el cliente que basados en RMI, HTTP o SOAP permiten la comunicación cliente-servidor con la dificultad no resuelta de cómo lograr que estas aplicaciones sean interoperables.

   Para esto surge una extensión de la API denominada JSR 160 que permite comunicar en forma remota aplicaciones java implementando conectores con soporte para RMI (JRMP, IIOP), en forma opcional permite la comunicación por medio de sockets y serializacion java.
   Este mecanismo facilita el manejo de consolas de management comunicando vía RMI clientes y servidores con independencia de la implementación de JMX que se use.

   Un punto de especial interés es el de la configuración de los ítems de monitoreo, dado que un requerimiento de la herramienta a construir es que sea fácilmente configurable y extensible.

La configuración de nuevos ítems se presenta como un proceso sencillo, especificando las características del Mbean correspondiente en un archivo que puede ser incorporado dinámicamente si es necesario.
Estas incluyen atributos a monitorear, operaciones, notificaciones y acciones a ejecutar sobre cada MBean en particular.

3. Luego de elegida la implementación a usar, verificar que ésta permite desarrollar y cumplir con los requerimientos básicos para cumplir con el diseño y alcance del proyecto, para lo cual se desarrolló un prototipo de prueba utilizando la arquitectura y API provista por MX4J. En el Apéndice 8.4 se detalla la aplicación.

Esta comprobación se vio favorecida por el hecho de tener identificado los componentes básicos, la arquitectura y la definición de qué herramienta se debía construir.

Las pruebas tomaron como base una serie de aspectos considerados como clave para el desarrollo de la aplicación. Sobre éstos se desarrolló un pequeño ambiente de prueba sobre el cual se probaron una serie de aspectos entre un grupo de componentes desarrollados en el contexto de JMX utilizando la API que provee MX4J. Detalles del la aplicación de prueba se encuentran en el Apéndice 8.4.

Este ambiente estuvo formado por una pequeña aplicación con tres módulos básicos:

1. Agente. En él se definen los MBeans que representan los recursos a ser monitoreados y los monitores sobre cada atributo a controlar.
2. Servidor. En él se definen y registran los conectores para implementar la comunicación remota con el Cliente y los Mbeans del Agente. También se definen las acciones a ejecutar en caso de que los monitores detecten alguna condición de alarma.
3. Cliente. El cliente consume información sobre los MBeans del Agente. Esta información la obtiene a través del Servidor, que oficia de proxy entre el cliente y el Agente.

Como se detalla en el documento “Arquitectura y especificaciones Diseño” se dedicó tiempo a desarrollar un ambiente de prueba donde verificar estos puntos y su aplicación satisfactoria determinó que se adaptara el diseño, inclusive la arquitectura para utilizar la tecnología JMX.

Los aspectos que se verificaron se mencionan a continuación:

De arquitectura:

Se investigó acerca de las posibilidades de desarrollar la arquitectura previamente definida.
Se comprobó que sí es posible, se desarrolló un Agente, un Servidor donde se centraliza la información y clientes que consumen ésta a través del Servidor. Se encontró que mediante el registro remota de componentes en los MbeanServers, se pueden mantener referencias remotas que hacen transparente la comunicación entre los componentes.

De comunicación:

Otro punto de gran interés fueron los mecanismos de comunicación soportados, especialmente RMI que era el que se deseaba utilizar, y las posibilidades de incorporar nuevos mecanismos.
Se comprobó en forma satisfactoria la comunicación de los componentes vía RMI.

Ítems de monitoreo:

Una de las características básicas de la herramienta a desarrollar era que fuera genérica en el sentido que se pueda monitorear todo tipo de recursos tanto de software como de hardware, por lo que resultó de particular interés buscar posibles limitantes en la representación de los recursos a través de los denominados MBeans. Se encontró que un Mbean podría a llegar a ser cualquier clase en Java que implementara su propia interfase Mbean y cumpliera ciertas reglas de nomenclatura, por lo que no habría limitantes en ese aspecto. También por la bibliografía estudiada se vio que era posible monitorear

recursos de software de aplicaciones Java y no-Java.
Los servicios de monitoreo ofrecidos por MX4J permiten definir la frecuencia de monitoreo por ítem de monitoreo, y a su vez cada servicio corre de forma independiente del resto.

Registro:

Posee un mecanismo de registro que podría configurarse tanto para registro de archivos planos, o bases de datos. Pero permite de forma sencilla redirigir el registro a otras aplicaciones de registro.

Alarmas:

La API provee un servicio de monitoreo con tres tipos evaluación de condiciones de alarma. El StringMonitor, CounterMonitor y GaugeMonitor. Se analizaron las posibilidades de ampliar el espectro de condiciones utilizando lo provisto o generando un nuevo módulo y no se encontraron limitantes al respecto.

Acciones:

Se analizó cómo se ubicarían las acciones en esta nueva arquitectura, y que mecanismos de notificación se utilizarían para avisarle a las mismas que se ejecuten. En la arquitectura se pueden definir distintos tipos de servicios y registrarlos en el MbeanServer para interatuar con otros MBEans registrados. En el caso de las acciones estas podrían ser servicios que se registarían en el MbeanServer, y se les asociaría al servicio de monitoreo a través de notificaciones que éstos dispararían cuado se presenten condiciones de alarma.
Esto se pudo implementar de forma sencilla ya que la API provee mecanismos de notificación y escucha (llamados listeners) que facilitan esta comunicación. También se vio la posibilidad de ejecutar notificaciones remotas, y se comprobó de forma satisfactoria la funcionalidad.

Las aplicación de prueba permitió verificar de forma satisfactoria que la API resolvía aspectos fundamentales para el desarrollo de la herramienta y brindaba además muchas facilidades relacionadas con la comunicación y notificación local y remota, invocación remota de métodos. Se decidió entonces incorporar la implementación MX4J para el desarrollo de la herramienta.

3.4.5 Integración de la Arquitectura elegida con JMX

Una vez decidida la aplicación de JMX fue beneficiosa una redefinición de ciertos aspectos para compatibilizar el diseño hecho con las posibilidades del nuevo marco de trabajo y remodelar los componentes básicos del diseño definido adoptando los conceptos de la arquitectura provista por JMX.

Los agentes JMX permiten implementar los mecanismos de comunicación, definir los recursos, la forma en que se monitorean, con que frecuencia propia de cada recurso, que acciones es necesario disparar para cada caso, como se monitorea cada propiedad del recurso y como se registran los valores monitoreados. Cada monitor se encarga de medir con una frecuencia determinada el valor de una propiedad y en caso de que se de una condición de alarma se envía una notificación.

Dadas estas posibilidades se decidió dar más autonomía a los Agentes, permitiendo que en éstos, no solo se agendaran, recolectaran y registraran las medidas de los recursos, si no que también se evalúen las medidas recuperadas y en caso de que se den condiciones de alarma ejecutar las acciones. De esta forma se independizan los Agentes del Servidor. Si hubieran problemas en el Servidor la actividad de monitoreo de cada Agente no se detendría.

El Servidor mantiene una referencia remota de cada recurso (MBean) registrado en los Agentes para los cuales estableció una vía de comunicación. De esta forma los Clientes pueden contar con información del estado de todos los recursos de los Agentes relacionados al Servidor desde un único punto de acceso. Ésta es la función básica que cumplen los servidores en la arquitectura de la aplicación definida. Igualmente el Servidor puede comportarse como un Agente y monitorear directamente los recursos.

Los Clientes consumen la información sobre los recursos registrada en los servidores a través del Servidor o directamente desde los Agentes.

La siguiente figura resume la distribución de los componentes en la nueva arquitectura.

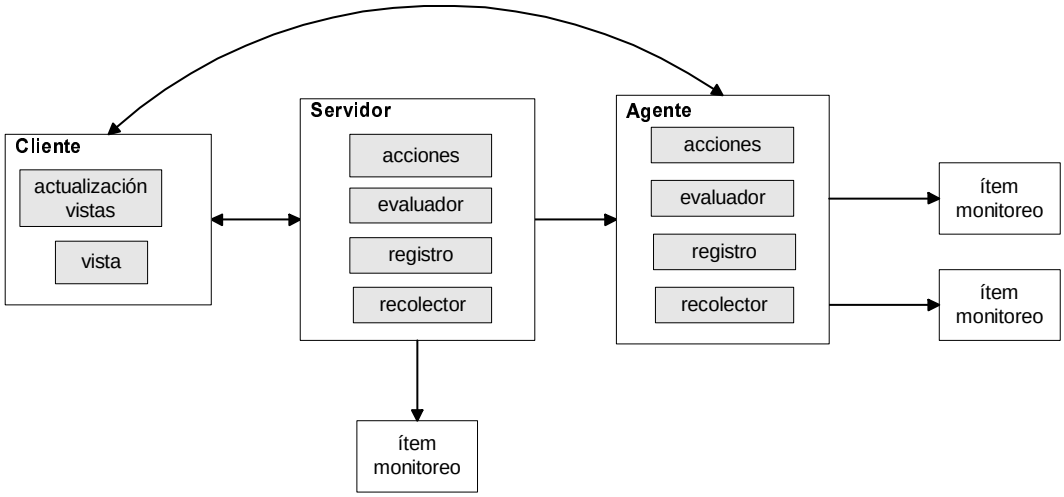


Figura 16. Distribución de componentes.

En el Apéndice 8.6 se puede ver el diseño de clases tentativo.

3.4.6 Componentes del sistema

La aplicación de esta arquitectura permite que los agentes tengan una independencia mayor y también más inteligencia. Cada MBean y el monitor asociado especifican el comportamiento del recurso monitoreado, mecanismo de registro, frecuencia de la toma de medidas, acciones a tomar, operaciones.

El sistema consiste de tres componentes principales:

- Agente
- Servidor
- Cliente

3.4.6.1 Agente

El Agente de monitoreo se compone de un MBeanServer, un conjunto de MBeans para representar los recursos a monitorear, servicios de monitoreo, acciones y registro que se registrarán en el MBeanServer como MBeans, y por lo menos un conector para comunicarse con el Servidor o directamente con el Cliente. También contiene un componente evaluador, y uno para manejo de configuración.

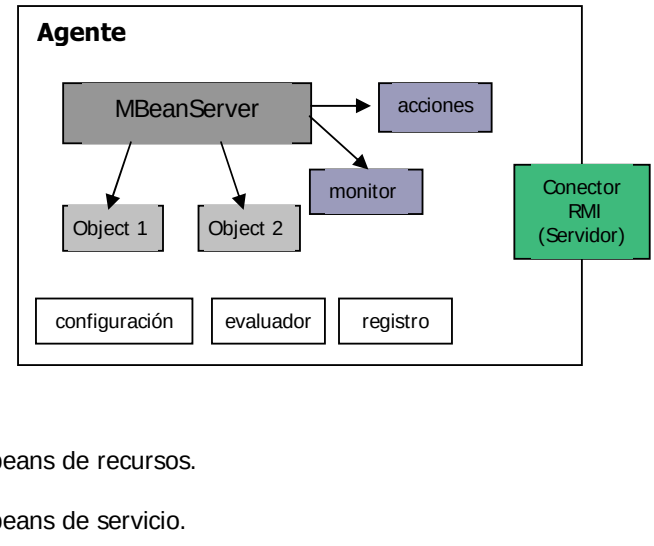


Figura 17. Componentes en el Agente.

3.4.6.1.1 MbeanServer

El Agente se compone de un único MBeanServer en el cual se registrarán los MBeans, tanto de recursos como de servicios.

3.4.6.1.2 Conectores

En el proyecto se utilizarán conectores RMI. Dado que el área de aplicación del producto es básicamente en redes locales (LAN) y éste es desarrollado en su totalidad en el entorno de programación Java consideramos que el empleo de RMI es adecuado y suficiente para esta etapa aunque pensamos tener en cuenta hacer un diseño de estos componentes que permita su fácil extensión hacia los otros mecanismos de conexión. MX4J incorpora en su implementación los conectores HTTP y SOAP [SOAP]. Éstos no se utilizarán, pero su integración al producto sería un proceso relativamente sencillo.

3.4.6.1.3 Monitor

Se utilizará el servicio de monitoreo brindado por JMX, el cual ofrece tres tipos de monitoreo, String, Counter y Gauge Monitor, y además se incorporará un nuevo tipo de monitoreo, Generic, el cual permite evaluar expresiones booleanas, de forma de ampliar el espectro de condiciones de alarma.
Se utilizará el componente Evaluador para implementar el nuevo monitor.

3.4.6.1.4 Evaluador

Encargado de evaluar expresiones booleanas. Se utilizará para evaluar las condiciones de alarma asociadas a los recursos de monitoreo. Se piensa utilizar un wrapper del evaluador de expresiones desarrollado en K&S en el producto AdaptorJava, el cual se desarrolló previamente con otros integrantes de la empresa.

3.4.6.1.5 Acciones

Este componente será el encargado de ejecutar las acciones especificadas cuando se cumplan las condiciones de alarma. Cada acción estará escuchando las notificaciones enviadas por los monitores (será un listener). Cuando el monitor determine una condición de alarma enviará notificaciones a todas las acciones que se registraron en el MBeanServer como listeners del monitor, y éstas ejecutarán la acción correspondiente.

3.4.6.1.6 Registro

Este componente se encargará de realizar el registro de las medidas que se van tomando. Soportará registro a BD y a archivos planos. Para el registro a bases de datos, la idea es que tanto los Agentes como el Servidor lo hagan a un repositorio común. Por un tema de desempeño se considera tener dos bases, una para los datos más recientes, y otra para mantener un histórico. De esta forma se podrán conservar datos históricos sin influir en el desempeño de la herramienta.
El registro de archivos será local al Agente.

3.4.6.1.7 Manejo configuración

Este componente se encarga de leer la especificación de los componentes de la aplicación y su comportamiento. Está pensado para soportar fuentes de datos archivos XML, o bases de datos, pero en el proyecto se va a implementar solamente manejo de archivos XML.

3.4.6.1.8 Inicializador

Se encarga de inicializar el agente, habilitando los conectores y adaptadores para que el Servidor acceda y manipule los MBeans registrados en el Agente.

3.4.6.2 Servidor

Este componente oficia de apoderado de todos los agentes de monitoreo de forma que los posibles Clientes interactúen con un solo MBeanServer y no tengan la dificultad de tener que encontrar en qué MBeanServer de que Agente de monitoreo se encuentra el MBean requerido.
El Servidor consiste de un MBeanServer el cual es responsable de la administración de todos los MBeans del sistema ya sean locales al Servidor o remotos en un Agente.
Para los Clientes la ubicación de los Agentes de monitoreo es transparente ya que sólo interactúan con el Servidor.

También contiene un componente para el manejo de configuración, inicialización y registro, y en caso de administrar MBeans a recursos locales, entonces tendrá los componentes Evaluador, Acciones, y Monitor, como en los Agentes de monitoreo.

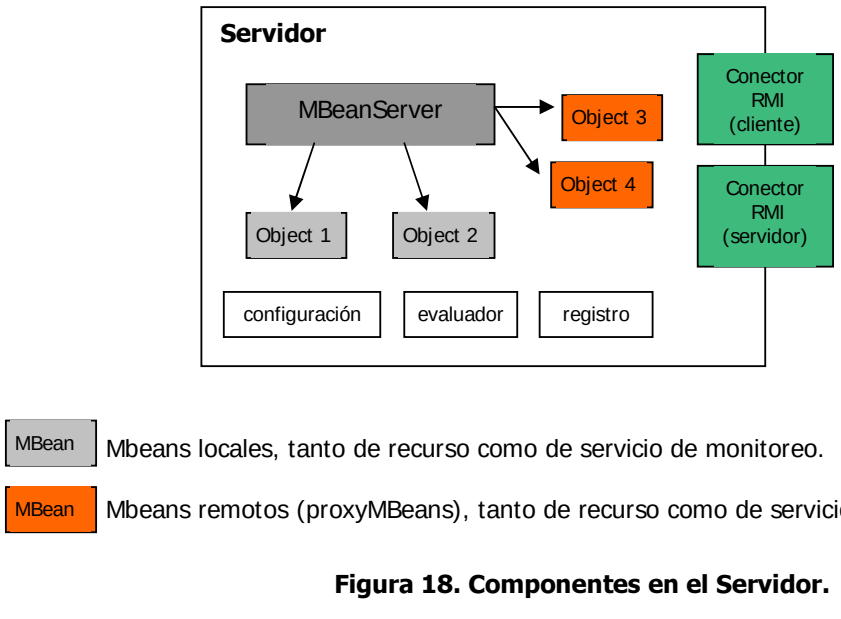


Figura 18. Componentes en el Servidor.

3.4.6.2.1 MBeanServer

El MBeanServer registra MBean locales y referencias a los MBeans remotos (proxyMBeans). Para esto el agente debe ofrecer un conector para poder realizar invocaciones remotas desde el Servidor que oficiaría de cliente en la comunicación establecida.

3.4.6.2.2 Conectores y Adaptadores

Debe contener al menos un conector para que los Clientes accedan y manipulen los recursos (MBeans) registrados en el MBeanServer. Al igual que en los Agentes se van a utilizarán conectores RMI.
No se va implementar ningún Adaptador en el proyecto, pero la arquitectura y utilización de MX4J permitirían de forma sencilla incorporarlos en el futuro.

3.4.6.2.3 Inicializador

Se encarga de inicializar el Servidor.
La inicialización consiste en habilitar los conectores para que los clientes accedan a los MBeans, registrar los MBeans locales, conectarse a cada Agente habilitado para el Servidor y registrar en el MbeanServer una referencia remota a éstos.

3.4.6.3 Cliente

Se desarrollará un Cliente que consistirá de una interfase gráfica para visualizar el estado de un grupo elegido de MBeans.
Para acceder al Servidor, realizar invocaciones remotas o recibir notificaciones remotas requiere de un conector. Tendrá un componente para el manejo de configuración, un inicializador y otro que se encargará de refrescar la consola, que con cierta frecuencia chequea el estado del MBean a monitorear o es notificado cada vez que se cumpla una condición de alarma.

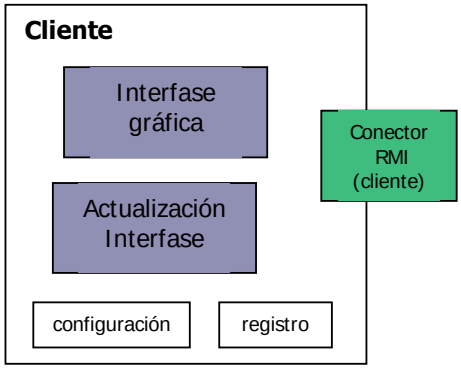


Figura 19. Componentes en el Cliente.

3.4.6.4 Interacción a nivel de componentes

En el siguiente esquema se presenta la relación entre todos los componentes del sistema.

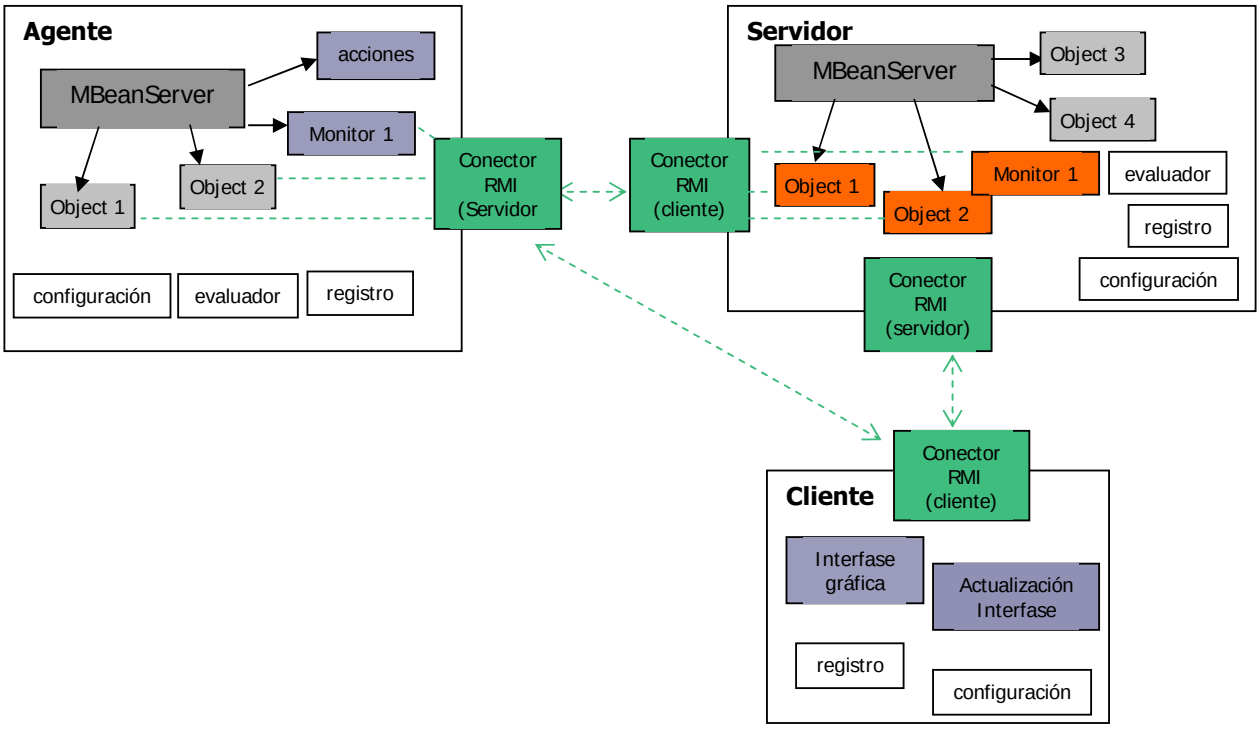


Figura 20. Interacción entre componentes del sistema.

3.4.6.5 Procesos durante la ejecución

La intención de este apartado es describir como se relacionarán los principales componentes, (Agentes, Servidor, Clientes) a través de los procesos que los componen.
Se presentan diagramas de actividad y de secuencia, utilizando el lenguaje SDL ya que consideramos permite describir de forma simple los proceso y sus relaciones.

Descripción:

En el Agente, cada atributo de un MBean tendrá asociado uno o varios monitores corriendo cada cierta frecuencia especificada, que evaluarán las condiciones de alarma.

En caso de que se de una condición de alarma, el monitor enviará notificaciones a todas los MBeans que estén escuchando, en particular las acciones.

Se ejecutan las acciones que están escuchando. Estas acciones son locales al Agente.

Los clientes están escuchando a través de Servidor, por las notificaciones que envían los monitores, de modo que cuando se da una condición de alarma el cliente actualice su consola.

En paralelo a lo anterior los clientes están actualizando la consola cada cierto tiempo especificado, mediante invocaciones remotas al Servidor y este realiza una invocación remota al Agente.

Diagramas de actividad asociados a los procesos descriptos:

3.4.6.5.1 Procesos en el Agente

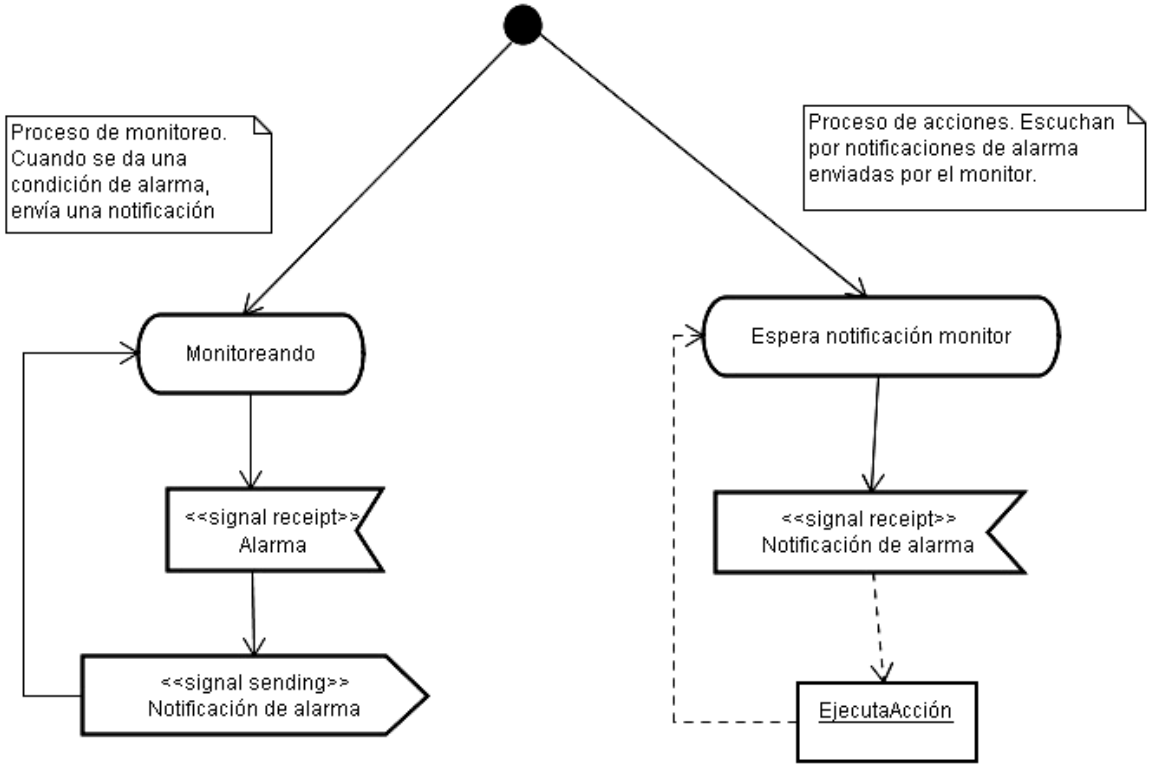


Figura 21. Procesos en el Agente.

3.4.6.5.2 Procesos en el Servidor

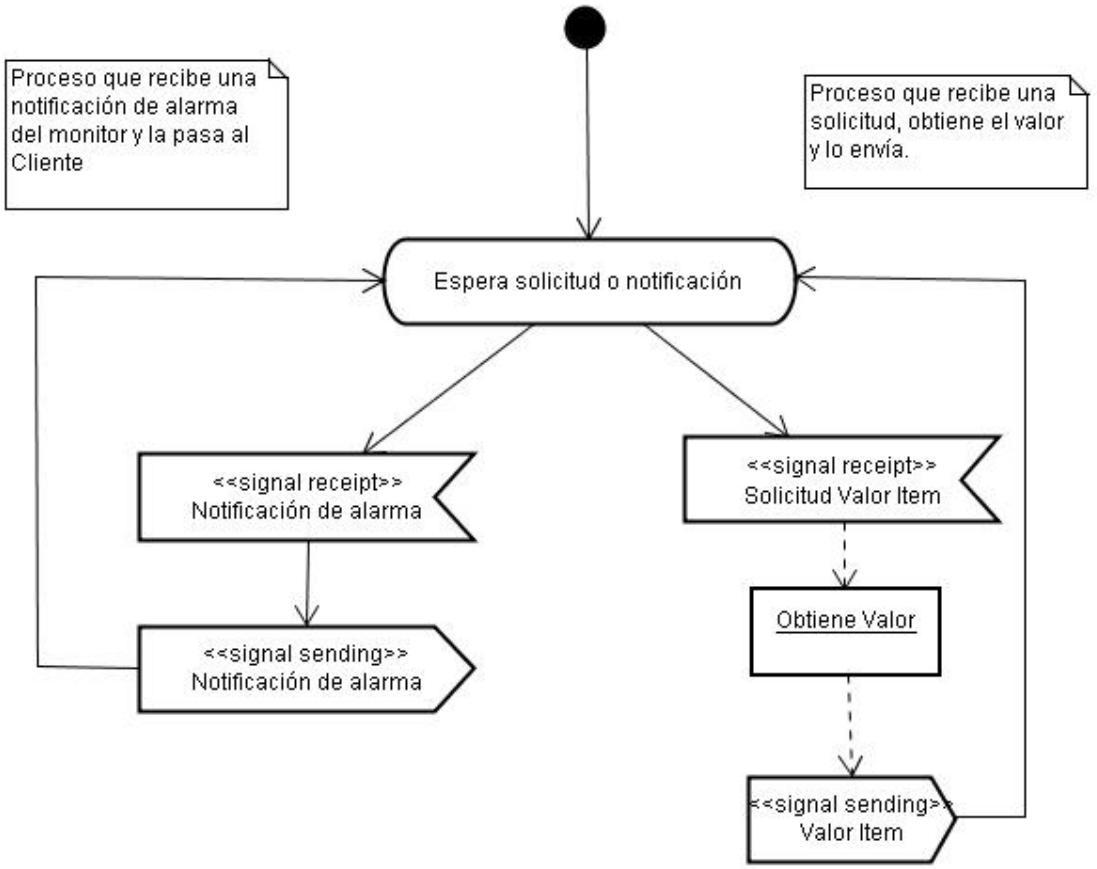


Figura 22. Procesos en el servidor.

3.4.6.5.3 Procesos en el Cliente

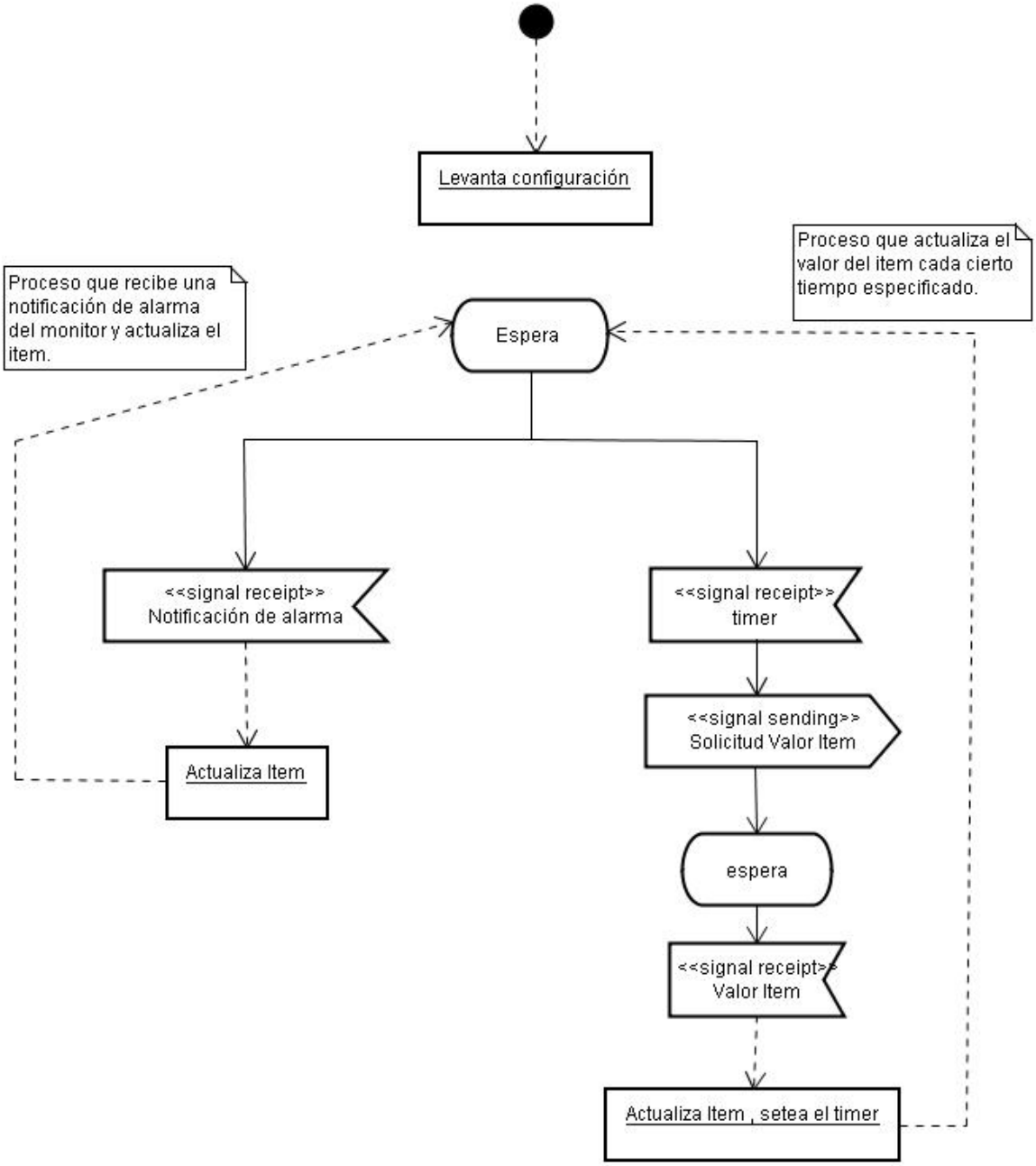


Figura 23. Procesos en el cliente.

3.4.7 Resumen del trabajo

Se comenzó esta etapa con:

- conocimientos generales en el tema producto de la etapa de investigación.
- conocimiento sobre los problemas y la discusión del proyecto en java que se daba en la empresa.
- requerimientos y alcance del proyecto bien definidos.
- poca afinidad con el mundo de las herramientas Open Source.

Producto de esta etapa tenemos discusión sobre la arquitectura de la aplicación con el manejo de una serie de posibilidades, la discusión sobre el uso de la tecnología JMX y la aplicación de esta al rediseño de lo que ya se tenía.
Esto fue un hecho fundamental para el desarrollo futuro no solo de la aplicación sino también para el balance de que posibilidades de abren con el uso de JMX, por lo que se considera que en realidad el mes y medio que se le dedicó al estudio y prueba de esta tecnología fue una inversión de tiempo por varios motivos.

Entre otros:

- Simplifica el hecho de configurar los recursos a monitorear y sus atributos.
- Simplifica la programación de la formas de monitoreo ya que brinda monitores específicos.
- Es posible extender la lógica de los monitores para crear otros según la necesidad.
- La frecuencia de muestreo esta embebida en el monitor lo que simplifica su programación.
- Ofrece herramientas para levantar la configuración desde archivos XML, este mecanismo fue adaptado según las necesidades particulares de la herramienta.
- Permite definir mecanismos que procesos escuchen (listeners) en forma automática por condiciones de excepción las que se reflejan a través de los listeners
- Hace transparente los mecanismos de conexión y configurable el protocolo de conexión.

Por lo tanto, al final de esta etapa no hay dudas de las posibilidades que se abren con el uso de JMX y que no sería una limitante a la flexibilidad de la herramienta a construir.

Se completó y desarrolló el documento de diseño el cual sirvió como punto de partida para el proceso de implementación.

El trabajo realizado fue mayor de lo planificado para esta etapa debido al tiempo dedicado al estudio de JMX y al desarrollo de la aplicación de prueba. Igualmente derivó en una reducción del tiempo de implementación de algunos componentes que ya están soportados por la plataforma, como ser, mecanismos de conexión, de monitoreo, de notificación. De no ser así se deberían haber implementado en su totalidad.

3.5 DESARROLLO

3.5.1 Objetivos

El objetivo principal de esta etapa fue desarrollar el motor de la herramienta, y un conjunto de pluggins de monitoreo, alerta y registro, a partir de los conocimiento adquiridos y decisiones tomadas en las etapas previas. Como resultado de este desarrollo se esperaba contar con un conjunto de programas que permitieran instalar el producto, principalmente en la plataforma Linux.
También fue objetivo para esta etapa la generación de documentación del código utilizando la herramienta javadoc [javadoc], documentación de usuario, para el desarrollador e instalación.

3.5.2 Desarrollo

3.5.2.1 Recursos informáticos

Parte del desarrollo se realizó en K&S, donde se contaba con los recursos informáticos necesarios. Esto permitió contar con el apoyo de otros integrantes de la empresa tanto para realizar intercambio de ideas, como cuando se encontraron dificultades al instalar o utilizar nuevas herramientas.

El desarrollo fue realizado bajo las plataformas WIN32 (W2K Service Pack 4, XP Service Pack 1) y Linux (SUSE 8.2 [SUSE]), esto se vio simplificado por el hecho de que los IDE de desarrollo tiene el mismo comportamiento para ambas plataformas, con la salvedad de aspectos de configuración del entorno de trabajo

Se utilizó la herramienta de desarrollo Eclipse 2.1.2 [Eclipse], en lugar de la sugerida, SUN ONE (o NetBeans), debido a que los estudiantes ya la estaban utilizando en otro proyecto de la empresa, y con la cual habían conseguido un buen manejo y familiaridad.

Para el control de versionado se utilizó la herramienta sugerida, CVS [CVS].
Esto fue de real utilidad ya que minimizó los errores al momento de realizar cambios en el repositorio. Esta herramienta permite analizar las diferencias entre distintas versiones del mismo elemento (clase, documento, etc). Para que esta funcionalidad sea de utilidad es necesario que todas las personas que trabajan en el mismo repositorio utilicen el mismo formato de código, independientemente de la herramienta de desarrollo.

3.5.2.2 Modalidad de trabajo

En la etapa Especificaciones de arquitectura y diseño se realizó el diseño de los componentes principales (Agente, Servidor, Cliente), pero se dejó para las etapas posteriores el diseño concreto de los módulos que lo integran. A partir de ésta no se siguió un modelo en cascada estricto, sino que se adoptó un modelo en cascada con retroalimentación para diseño, desarrollo y prueba como se muestra en la figura siguiente.

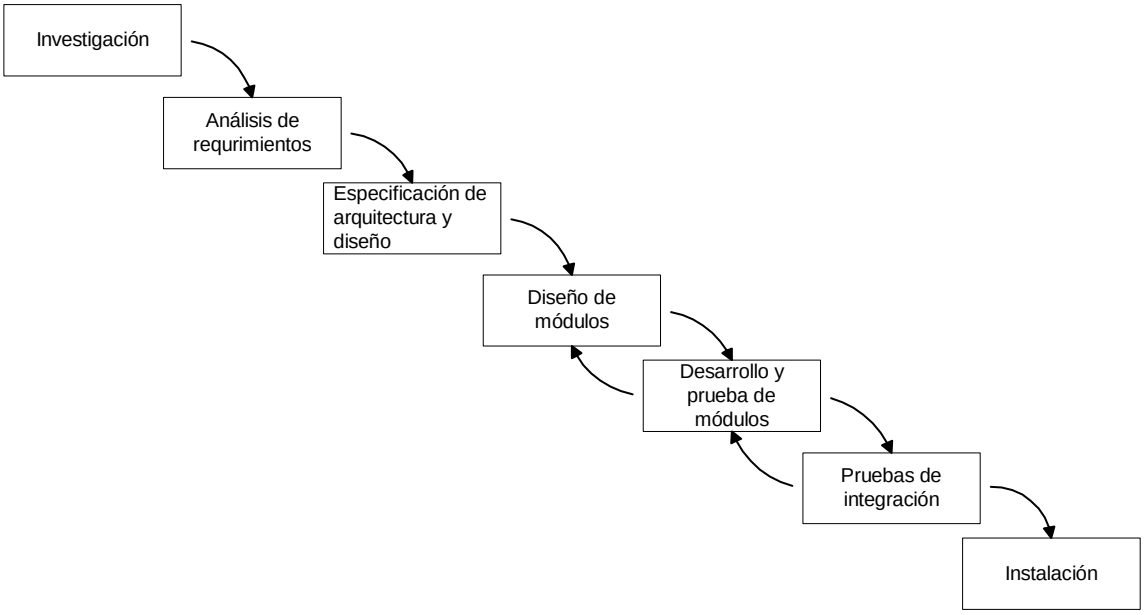


Figura 24. Etapas del proyecto.

La dinámica de trabajo se basó en la comunicación permanente de los integrantes del grupo, mediante reuniones frecuentes y de corta duración en las cuales se planteaban dudas y se discutían los avances diarios. Generalmente éstas eran en la empresa. A su vez se realizaban reuniones semanales o quincenales en la que se evaluaban los resultados obtenidos del período anterior y se planteaban objetivos concretos para el siguiente período.
En éstas se distribuía el trabajo de acuerdo a las posibilidades de tiempo de cada integrante del grupo, y contemplando en qué área se sentía más a gusto o se desenvolvía mejor. El tener un diseño modular y trabajar con un lenguaje orientado a objetos facilitó la división de trabajo. Si bien en general se trabajó sobre distintos módulos, cuando fue necesario hacerlo sobre los mismos, CVS permitió trabajar de forma clara, ordenada, y segura.

Se acordaron ciertas reglas en la utilización del CVS para lograr un mejor trabajo, y un producto de mayor calidad. Algunas de éstas son:

- no actualizar una clase sin antes probarla.
- no actualizar una clase con errores.
- comentar el cambio realizado.
- formatear el código antes de actualizar.
- utilizar el mismo formato de código.

3.5.2.3 Metodología de Validación

A medida que se fueron incorporando las distintas clases asociadas a los pluggins de monitoreo y acciones, se fueron implementando casos de prueba utilizando la herramienta JUNIT [JUNIT]. Su uso se consideró ventajoso en varios aspectos. Por un lado permite implementar casos de prueba escritos en java, de forma simple, sin invertir mucho tiempo en crearlos, lo cual se encontró muy importante. Se puede tener la certeza de que cambios que se hagan en el código se verán reflejados en las pruebas. Como JUNIT chequea los resultados de forma automática y provee información inmediata sobre los resultados de las pruebas, correr las pruebas resulta en una tarea simple.

3.5.2.4 Detalles de implementación

Uno de los objetivos de esta etapa fue generar la documentación del código con la herramienta javadoc, a nivel de paquete java, clases, y métodos. Esta documentación junto con el diseño de clases, proveen información de cómo se organizaron las clases en java y las relaciones entre las mismas.
En el Apéndice 8.6 se encuentra el diseño de clases.

El javadoc producido se incluirá en el material de apoyo al Desarrollador, junto con el documento "Guía para el Desarrollador" en el cual se detallan aspectos del desarrollo tanto para aquellas personas que deban mantener la herramienta, o quieran continuar desarrollándola [GD].

Para la codificación se siguieron los estándares utilizados en la empresa, más bien en cuanto a nomenclatura de métodos y variables.

En esta sección se describen algunos puntos que pareció interesante mencionar, o que fueron de particular dificultad al implementar, tanto en los componentes del motor como en los distintos pluggins.

Las distintas áreas se presentan en distintas subsecciones para más claridad.

3.5.2.4.1 Motor

Muchos de los problemas a resolver en la implementación del motor fueron vistos previamente en la aplicación de prueba de MX4J, lo cual facilitó su desarrollo. Igualmente se siguió profundizando en JMX viendo las distintas posibilidades que ofrecía y de esa forma encontrar la solución más adecuada al problema a resolver.

Configurador

Dentro del motor se incluye el módulo configurador. Éste en particular llevó más tiempo del esperado, debido a la dificultad que se encontró al momento de definir si generar un módulo de configuración genérico o más específico para esta aplicación.
Se decidió dividir el módulo configurador en dos módulos independientes, el *loader* y el *reader*. El *loader* que permitiera a partir de una lectura de un archivo XML adecuadamente configurado inicializar, cargar los ítems, acciones, monitores y conectores. Así como poder realizar operaciones de *start* y *stop* sobre conectores y monitores. Este módulo se pensó que fuera específico del producto, pero genérico en el sentido que soportara la incorporación de nuevas clases de ítems, acciones, monitores, o conectores sin que requiriera modificaciones de código. Para su implementación se reutilizó en gran parte la solución que ofrece la MX4J para carga de mbeans a partir de archivos XML.

Luego se implementó el módulo *reader*, el cual que permite realizar recorridas en archivos XML. Es más genérico que el *loader* pero su propósito es otro, simplemente obtener información del archivo.

Logger

Para el componente de registro se reutilizó el implementado en K&S en el proyecto AdaptorJava, pero se realizaron pequeños cambios para adecuarlo a las necesidades de esta herramienta. Especialmente se realizaron cambios para el registro a base de datos y en la inicialización del componente. Éste permite realizar registro en base de datos, en archivos o enviar a la salida estándar. Es configurable, se puede por ejemplo habilitar o deshabilitar cierto tipo de registro a nivel de componente Se pueden establecer diferentes niveles de registro. Este módulo se utiliza tanto para registrar la actividad del motor, como en los distintos pluggins.

Conectores

JSR 160 provee un API estándar para conectarse de forma remota a una aplicación basada en JMX.
En la herramienta se utilizó la implementación del estándar que provee MX4J basado en conectores RMI.
El hecho de utilizar la implementación que provee MX4J no limita el uso de otras implementaciones. Es posible por ejemplo utilizar la implementación que ofrece MX4J de JSR160 del lado del cliente, y tener otra implementación del estándar del lado del servidor. Esto permitiría por ejemplo crear una consola de administración que podría interoperar con el Server o el Agente sin importar la implementación de JMX que se utilice.

3.5.2.4.2 Desarrollo de pluggins de monitoreo de recursos.

SNMP

Para implementar el pluggin de monitoreo para SNMP se buscaron implementaciones open source del protocolo SNMP, realizadas en java, que permitieran ejecutar las operaciones básicas del cliente y agente SNMP (gets y sets de OIDs). En estas búsquedas, en Internet, se encontraron tres implementaciones con estas características. Java SNMP package [SNMPPkg], NETSNMP [NETSNMP]y snmp Construction kit [SKC]. De las encontradas elegimos SNMP package ya que era la que brindaba la interfaz de comunicación más simple, soportaba SNMP versión 1 y 2, y además implementaba la captura de traps SNMP.

Las pruebas realizadas con esta librería fueron exitosas, por lo cual se decidió utilizarla.
Algunos de los OID que se necesitaban monitorear (carga de CPU, memoria libre en disco, espacio libre en disco) no se encontraban definidos en la MIB que la plataforma WIN ofrece por defecto, por lo que hubo que extender la MIB para que los soportara.
En linux ya que se contaba con estos OIDs.

El desarrollo de este pluggin requirió más esfuerzo del esperado. Primero, encontrar los OID adecuados requirió investigar mucho, extender la MIB y realizar varias pruebas. Segundo, se encontró un bug en las librerías de SNMP utilizadas, lo que implicó inspeccionar el código para encontrar el problema, que no se creía estuviera en un principio allí.

PING

La API de Java no permite construir paquetes ICMP para envío y recepción de los mismos emulando el comportamiento del programa PING. Dada esta dificultad se optó por ejecutar el programa PING de la misma forma que se realiza desde el shell que ofrece el sistema operativo. Un problema de esta solución es que es dependiente de la plataforma, pero es correcta y simple a la vez.
No se descarta en un futuro cambiar la implementación para que sea independiente de la plataforma.

Chequeo del servicio de K&S Middleware.

Este pluggin monitorea tanto el servicio de Middleware desarrollado en la plataforma WIN32, como el implementado en java en el producto AdaptorJava.
Generalmente en la empresa se realiza este control de forma manual desde una aplicación cliente, desde la cual se envía una petición bien conocida por el servicio y se observa la respuesta.
El pluggin realiza esta misma actividad, envía la petición al servicio, analiza la respuesta y determina si el servicio está respondiendo correctamente.
Para realizar el envío de la petición se utilizan el cliente RMI o cliente socket que ofrece el producto AdaptorJava implementado en la empresa.

Chequeo de puerto

Este pluggin permite chequear si se puede establecer una conexión punto a punto con determinado puerto en cierto host utilizando el protocolo TCP. La implementación simplemente intenta crear un socket al host y puerto especificado.

Inspección de archivos y búsquedas.

El pluggin provee información del tamaño y fecha de última modificación del archivo.
Permite buscar un texto en todo el archivo, así como definir secciones de búsqueda dentro del mismo determinadas por un texto de comienzo y otro de fin si se desea.
Para hacer más eficiente la búsqueda se guarda un cursor con la última posición de búsqueda.

3.5.2.4.3 Desarrollo de pluggins de acciones.

Envío de correo electrónico

Se utilizó el servicio ofrecido por MX4J que usa el protocolo SMTP. Este punto se quiso destacar ya que sin mucho esfuerzo se pudo realizar envío de correo electrónico.
Es necesario definir los datos para el armado del mensaje como: dirección de envío del mensaje, direcciones de destinatarios y copias, servidor de correo y sujeto del mensaje. El contenido del mensaje es generado por el manejador de notificaciones cuando es invocado por cada monitor y posteriormente es encapsulado por el manejador de las acciones.

Envío de mensajes SMS

Se resolvió enviando correo electrónico.
El mecanismo de definición de este tipo de mensajes se resuelve con la misma interfase del envío de correo electrónico con la salvedad de que en la dirección de correo del destinatario se debe ingresar el número de teléfono celular más un código clave de cuatro dígitos. Este mecanismo se beneficia de las facilidades de un proveedor en particular y se estudia como extender este servicio a otros proveedores.
A modo de ejemplo: ***"NumeroTel"+ "XXXX"@ancelinfo.com.uy***

Parar un proceso

Permite matar un proceso del sistema.
Solo se implementó en la plataforma linux y de forma similar al pluggin para PING, pero en este caso utilizando los comandos kill y pkill.
El comando kill permite matar un proceso dado su Id de proceso (número único que le asigna el sistema operativo al proceso para identificarlo).
El comando pkill permite matar todo proceso dado su nombre de proceso. En este caso puede haber más de un proceso vinculado al mismo nombre, en cuyo caso se eliminarían todos.

3.5.2.4.4 Desarrollo de pluggins de alarma.

Se reutilizó el servicio de monitoreo que ofrece MX4J, StringMonitor, CounterMonitor y GaugeMonitor pero se debieron realizar algunos cambios para ajustarse a las necesidades de la herramienta. También se agregó un nuevo monitor que permite evaluar expresiones booleanas. Parte de esta implementación requería desarrollar un componente que permitiera evaluar dichas expresiones. La empresa ya contaba con un evaluador de expresiones, el cual se desarrolló durante el proyecto AdaptorJava en el que uno de los integrantes del grupo había participado, por lo que se decidió utilizarlo. Para desarrollar el pluggin se implementó un wrapper del componente evaluador.

3.5.3 Estadísticas

La herramienta Statcvs-xml [Statcvs-xml] provee reportes interesantes acerca del uso de CVS basado en los registros que CVS genera. Provee estadísticas a nivel de usuario, de archivos por módulos, y también información acerca de la evolución del producto.

En esta sección presentaremos alguno de los resultados arrojados durante el desarrollo.

Cantidad de archivos ingresados al repositorio desde la creación del repositorio.

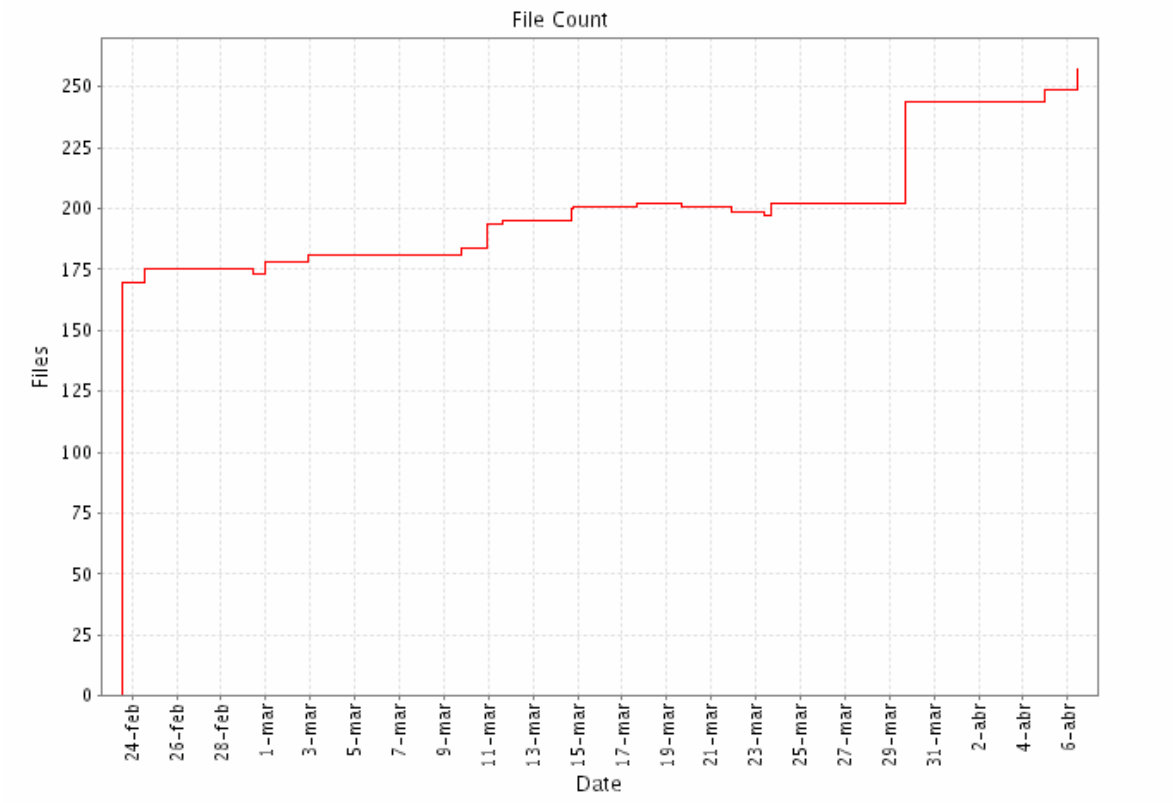


Figura 25. Cantidad de archivos ingresados al repositorio

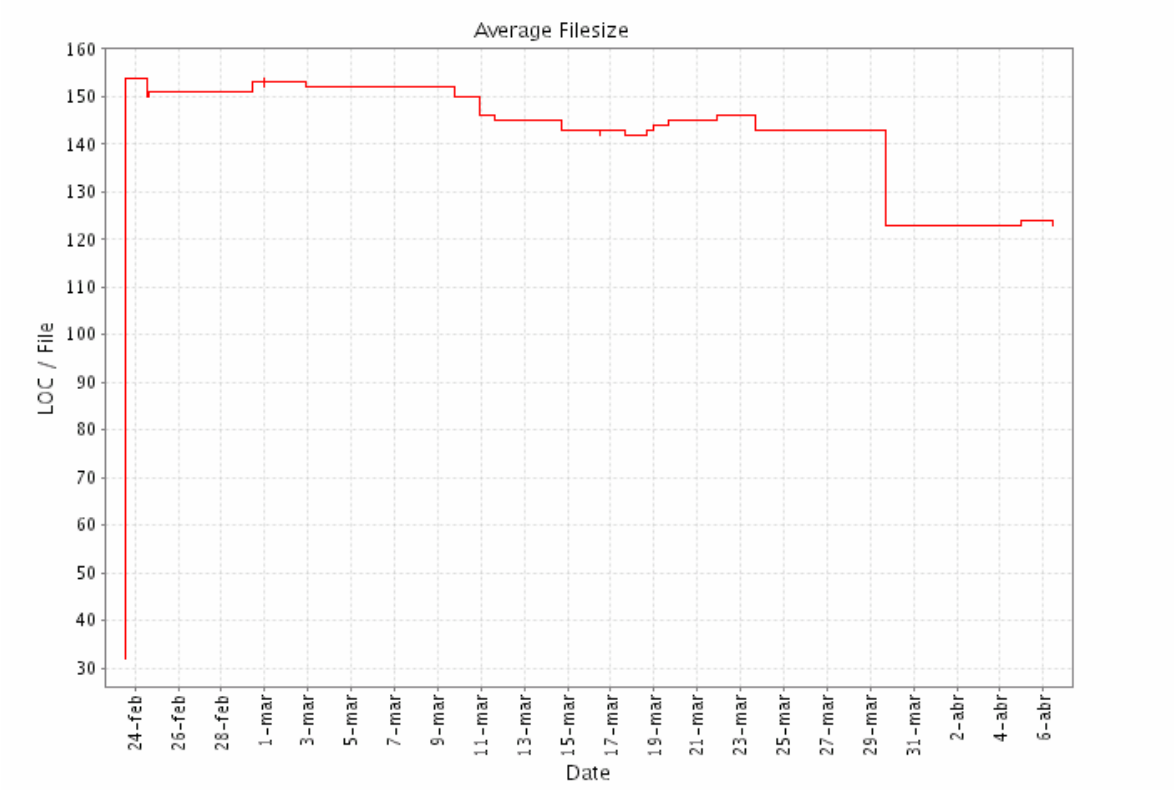


Figura 26. Promedio del tamaño de los archivos.

3.5.4 Resumen del trabajo

Resultados

Se consiguió implementar el motor, un subconjunto de los pluggins especificados en la etapa "Análisis de requerimientos".
No se pudo completar el desarrollo de todos los requerimientos mínimos, pero se creyó conveniente depurar asuntos del funcionamiento del motor, completar la etapa de pruebas y finalizar todos los documentos requeridos.

Los requerimientos mínimos no implementados son los siguientes:

1. Mantener archivos de log de acuerdo a un período configurable, a modo de ejemplo si el período es una semana y se mantienen archivos diarios, solo se almacenan uno por cada día y se sobrescribe un archivo por día. La forma de escritura de archivos sería circular.

Se generó la documentación planteada en los objetivos para esta etapa, "Guía de Instalación y Configuración" [GID], "Guía del Desarrollador" [GD], "Documentación de Usuario" [DU] y "Javadoc" [JDOC].

Evaluación

En esta etapa se adquirió formación en el uso de herramientas de desarrollo (Eclipse, NetBeans), versionado (CVS), documentación (javadoc), diseño (Jude) y prueba (JUnit), esto es valorado muy positivamente por el hecho de que los integrantes del grupo no contaban con experiencia previa en el uso de estas herramientas.
En particular, utilizar la herramienta CVS permitió experimentar una modalidad de trabajo diferente a la habitua, la cual se encontró sumamente positiva.

El trabajo en la plataforma Linux en la cual no se había tenido mucho contacto previamente fue de gran aporte para la formación de los integrantes del grupo, lográndose al final de esta etapa un grado de familiaridad importante tanto en configuración de herramientas en este ambiente como en manejo del sistema operativo.

Se tomó contacto con el mundo de las herramientas open source, que hasta el momento no formaba parte de la cultura de trabajo de la empresa, explotando sus facilidades y ventajas. Son el caso de la utilización de MX4J, la implementación de SNMP, Eclipse, postgreSQL, Linux, JUDE, CVS, etc. Éstas permitieron un grado de flexibilidad importante ya que fue posible modificar el código de la implementación para adaptarlo a las necesidades particulares de la herramienta. En algunos casos fue necesario implementar secciones enteras de código y en otras con pequeñas modificaciones fue suficiente.

Fueron reutilizadas soluciones ya implementadas en K&S cuando se creyó conveniente, así como soluciones encontradas en otras fuentes.

Internet se constituyó como una fuente muy importante de respuesta a problemas, ya sea a través de las FAQs o foros de preguntas y discusión.

En conclusión, se adquirió experticia en el manejo de un conjunto de herramientas y se accedió a nuevas fuentes de conocimiento, que fueron de gran utilidad tanto para el proyecto en sí como para trabajos futuros.

La etapa duró el tiempo estipulado en el cronograma original que era aproximadamente de tres a cuatro meses, con la salvedad de que se comenzó bastante más tarde en el tiempo debido a atrasos en etapas anteriores.

3.6 PLAN DE PRUEBAS

3.6.1 Objetivos

En esta etapa se definieron los casos de prueba que se consideraron básicos para revisar el correcto funcionamiento del sistema.
Se definió un conjunto mínimo de pruebas que permitieran, por un lado descubrir errores que pudiera contener el programa, por otro tomar medidas para evaluar las limitantes o posibilidades del producto en cuanto a carga y distribución.

Alcance

Primero se verificó el código asociado a las clases que modelan los pluggins de recursos a monitorear y pluggins de las acciones, asegurándose la correcta ejecución de la toma de medidas y ejecución de las distintas acciones. Para estas pruebas se utilizó JUNIT. La idea fue que los casos de pruebas se fueran incorporando de forma incremental a medida que se iban desarrollando nuevos pluggins. Esta actividad realizó en paralelo con el desarrollo.

Luego se realizó una verificación del comportamiento de los principales módulos del sistema: Agente de Monitoreo, Servidor, Aplicación Cliente.

Una vez finalizadas estas pruebas se realizaron pruebas de integración de los diferentes módulos. En éstas se incluyeron pruebas de carga, distribución y estabilidad durante cierto período de tiempo.

3.6.2 Desarrollo

En esta sección se describen brevemente los casos de prueba que se definieron como básicos para verificar el correcto funcionamiento del sistema. En el Apéndice 8.8 se realiza el detalle de los mismos, indicando los archivos de configuración utilizados en caso de que se necesitaran, y los resultados de las pruebas. Los archivos se adjuntan con este informe.

3.6.2.1 Casos de prueba para el componente de registro.

3.6.2.1.1 En archivos

1. El archivo de log no existe.
2. Inhabilitar el registro a archivos.
3. Registro de archivos habilitado.

3.6.2.1.2 En base de datos

Se intenta verificar el acceso a base de datos y la correcta ejecución de las operaciones de registro.

1. El Registro a base de datos inhabilitado.
2. Registro de base de datos habilitado.

3.6.2.2 Casos de prueba para pluggins de ítems de monitoreo.

3.6.2.2.1 Búsqueda en archivos

1. El archivo no existe.
2. La lectura de archivo se retoma en el último fin de lectura de la lectura anterior.
3. El archivo no contiene el texto de error buscado.

4. El archivo contiene el texto buscado pero no dentro de la sección determinada.
5. El archivo contiene el texto buscado dentro de la sección determinada.

3.6.2.2.2 Consultas SNMP

Se implementaron una serie de casos de prueba en JUnit para comprobar la normal ejecución de los casos planteados en los puntos 1, 2 y 3.

1. El agente SNMP no existe o no está activo.
2. OID no válido o no disponible.
3. Para cada OID validos, chequear que retorna los valores esperados.
   Se realizan consultas SNMP con OID validos en la base de datos MIB, no es un aprueba exhaustiva.

3.6.2.2.3 Ejecución de petición al servicio de K&S Middleware.

1. El servicio de Middleware no está disponible.
2. Canal de comunicación no soportado.

3.6.2.2.4 Chequeo de puerto

Se realizaron casos de prueba JUnit para verificar una serie de puertos correctos y uno con un valor incorrecto.

1. Número de puerto incorrecto.
2. No se puede establecer una conexión con el puerto.
3. Puerto habilitado.

3.6.2.2.5 Ejecución del comando PING

Se Implementa un caso de prueba que verifica el funcionamiento de la ejecución del comando PING a una dirección valida y a una dirección no valida.

1. La dirección IP a la que se hace el PING no existe.
2. La dirección IP a la que se hace es inalcanzable.
3. La dirección IP a la que se hace PING es correcta.

3.6.2.3 Casos de prueba de monitores.

Estas pruebas se repetirán para cada tipo de monitor: String Monitor, Counter Monitor, Gauge Monitor y Generic Monitor.

1. Verificar que cada monitor se ejecuta con la granularidad especificada.

2. Verificar que en caso de darse nuevamente una condición de alarma luego de haber pasado por una situación aceptable, se notificará nuevamente la acción asociada al monitor.

3.6.2.4 Casos de prueba para pluggins de acciones.

3.6.2.4.1 Registro

1. La ruta de destino del archivo de log no existe.
2. El archivo existe.

3.6.2.4.2 Ejecución de e-mail

Se diseño un caso de prueba JUnit que verifica el correcto funcionamiento del envió de e-mail con una serie de características definidas.

1. El correo se envía exitosamente. Se busca probar solamente el envió de correo
2. Cuando los monitores tienen asociada la acción SendMail, esta se ejecuta correctamente con los datos especificados y el mensaje preestablecido.

3.6.2.4.3 Modificación del valor de un objeto SNMP

Se diseño un caso de prueba JUnit que verifica el resultado de la operación set sobre un objeto habilitado en la base de datos, se ejecuta un caso con un OID valido y otro con un OID no disponible para escritura.

1. OID no válido o no disponible para escritura.
2. OID válido.

3.6.2.4.4 Parar un proceso del sistema operativo

1. No existe el proceso, no debe ejecutar ninguna acción.
2. Existe el proceso. Luego de pararlo, verificar en la lista de procesos del sistema que no exista un proceso del sistema con ese identificador.

3.6.2.5 Casos de prueba en el componente "Agente"

3.6.2.5.1 No existe el archivo de configuración

No existe el archivo de configuración desde el cual el Agente lee la configuración.

3.6.2.5.2 Sección del archivo de configuración mal configurada.

1. Declaración de un tipo no soportado.
2. Declaración incorrecta de los argumentos para una sección.

3.6.2.5.3 Problemas al crear el conector.

3.6.2.5.4 Problemas con el atributo a monitorear.

1. El atributo no es soportado por el ítem de monitoreo.
2. El atributo no es compatible con el tipo de monitor.
3. Problemas para obtener el valor del atributo.

3.6.2.6 Casos de prueba en el componente "Servidor"

3.6.2.6.1 No existe el archivo de configuración.

No existe el archivo de configuración desde el cual el Server levanta la configuración.

3.6.2.6.2 Falta una sección en el archivo de configuración.

1. Falta sección de agentes.
2. Falta sección de conectores.

3.6.2.6.3 Problemas al crear el conector.

3.6.2.6.4 Problemas para conectarse a alguno de los “Agentes”.

3.6.2.6.5 Inicializar el componente ya estando inicializado.

3.6.2.7 Casos de prueba en el componente “Cliente”

En este capítulo, cuando hacemos referencia a Cliente, nos referimos a la consola gráfica de administración implementada.

3.6.2.7.1 No existe el archivo de configuración.

No existe el archivo de configuración desde el cual el Cliente levanta la configuración.

3.6.2.7.2 Falta la sección conectores en el archivo de configuración.

3.6.2.7.3 Problemas para conectarse al “Servidor” o al “Agente”.

3.6.2.8 Pruebas de integración

Para realizar estas pruebas consideramos exitosas las pruebas de los distintos componentes detalladas anteriormente.

3.6.2.8.1 Caso de prueba de integración 1

Contexto
Las pruebas se realizarán en la plataforma LINUX.
Habrá dos Agentes corriendo cada uno en una máquina distinta, un Server y un Cliente corriendo en otras máquinas.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se parará de forma manual uno de los Agentes.

3.6.2.8.2 Caso de prueba de integración 2

Contexto
Condiciones idénticas al caso de prueba de integración 1.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se parará de forma manual el Server.

3.6.2.8.3 Caso de prueba de integración 3

Contexto
Condiciones idénticas al caso de prueba de integración 1.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se parará de forma manual el Cliente.

3.6.2.8.4 Caso de prueba de integración 4

Contexto
Las pruebas se realizarán en la plataforma LINUX.
Habrá un Agente, un Server y un Cliente corriendo en la misma máquina. Se quiere determinar la incidencia en el desempeño del equipo en el cual corren los componentes.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se harán mediciones de consumo de CPU y memoria para este esquema de funcionamiento.

3.6.2.8.5 Caso de prueba de integración 5

Contexto
Las pruebas se realizarán en la plataforma WIN.
Habrá un Agente, un Server y un Cliente corriendo en distintas máquinas. Se quieren evaluar las mismas medidas tomadas en el caso 4 pero en esta configuración particular.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se harán mediciones de consumo de CPU y memoria para este esquema de funcionamiento.

3.6.2.8.6 Caso de prueba de integración 6

Contexto
Habrá un Agente, un Server y un Cliente corriendo en distintas máquinas y sobre distintos sistemas operativos. Se busca evaluar que no haya problemas de comunicación y de funcionamiento entre los componentes debido a estar corriendo en distintas plataformas.

Detalle del caso
Se inicializarán los componentes según la siguiente tabla:

	Agent	Server	Client
Caso1	WIN	LINUX	LINUX
Caso2	LINUX	LINUX	WIN
Caso3	LINUX	WIN	WIN
Caso3	WIN	WIN	LINUX

3.6.3 Resumen del trabajo

Los casos de prueba se implementaron a medida que se fueron desarrollando los módulos de ítems y acciones. Éste fue un proceso ágil y simple debido al uso de JUnit como método para realizar las pruebas unitarias de tales módulos.

La realización de las pruebas de los principales componentes y las de integración permitió determinar que la herramienta puede ejecutarse de forma correcta con independencia de la plataforma, pudiendo distribuir también los componentes en diferentes plataformas a la vez de forma correcta.

A medida que se detectaron problemas durante la etapa de desarrollo, se fueron solucionando por el propio desarrollador permitiendo agilizar en gran medida el control y corrección de errores.
Aunque la mayor parte de los casos de pruebas especificados fueron satisfactorios, se encontró un grupo reducido de problemas sobre los cuales se sigue trabajando.

Estos son:

- El componente Servidor debe registrar y notificar la pérdida de conexión con alguno de los Agentes.
- El cliente deberá reflejar la pérdida de conexión con los Agentes o Servidores a los que está conectado.

4. EVALUACIÓN DEL CRONOGRAMA

Como resumen general del tiempo empleado en el proyecto se puede decir que este duró dos meses más de lo planificado en el cronograma original y si bien se cumplieron los objetivos planteados, no hubo un seguimiento estricto de las etapas planteadas en éste.
Se pueden clasificar en dos grupos los factores que motivaron el atraso respecto a los plazos establecidos en el cronograma original.

1. Situaciones de carácter personal que implicaron comenzar el trabajo cerca de 20 días más tarde del plazo oficial.

2. Atrasos por motivos inherentes al desarrollo en los cuales fue necesario evaluar su conveniencia y tomar la decisión a conciencia del atraso que provocaría. Tal es el caso de la decisión tomada de invertir un mes y medio en el estudio de JMX y desarrollo del prototipo de prueba, lo que se vio reflejado en el corrimiento del plazo de finalización. Esta decisión fue crítica para el desarrollo del proyecto dado que una vez definido el uso de JMX se simplificaron varios aspectos como manejo de conexiones o desarrollo de pluggins de recursos, acciones o monitores, y se potenciaron otros que se plantean como posibilidades de desarrollo futuro para incorporar al producto.

Una vez vistos los resultados se puede evaluar que la decisión tomada fue realmente positiva.

El siguiente cuadro presenta el cronograma original que resume las tareas determinadas como fundamentales para la correcta ejecución del proyecto.

Tareas \ (meses)	Jun	Jul	Ago	Set	Oct	Nov	Dic	Ene
1 - Relevamiento	x							
2 - Estudio Java	x	x						
3 - Definición del motor		x	x	x				
4 - Desarrollo prototip. Motor			x	x	x			
5 - Definición pluggins				x	x			
6 - Desarrollo prototip. Pluggins				x	x	x		
7 - Implantación y testing					x	x	x	
8 - Redacción de informes intermedios	x	x		x	x		x	
9 - Redacción del informe final						x	x	x

En el transcurso real del desarrollo, y por los motivos antes mencionados fue necesario redefinir la duración de las etapas y el tiempo para el cierre del proyecto.

En la siguiente tabla se muestra el cronograma modificado.

Tareas \ (meses)	Jun	Jul	Ago	Set	Oct	Nov	Dic	Ene	Feb	Mar	Abr
1 - Relevamiento del estado del arte	x	x									
2 - Estudio Java		x	x	x	x						
3 - Definición del motor				x	x	x					
4 - Desarrollo prototip. Motor						x	x	x			
5 - Definición pluggins							x	x	x		
6 - Desarrollo prototip. Pluggins								x	x	x	
7 - Implantación y testing									x	x	x
8 - Redacción de informes intermedios		x	x	x		x	x				
9 - Redacción del informe final										x	x
10- Relevamiento de requerimientos			x								

A continuación se discuten brevemente los tiempos empleados en cada etapa y una evaluación del manejo de los plazos en cada una.

El tiempo de dedicación en horas no fue estable durante todo el proyecto y fue en dependencia de la disponibilidad de los integrantes del grupo así como también de las exigencias puntuales en el desarrollo del mismo. La cantidad de horas aumentó conforme se iban viendo los resultados en el desarrollo de la herramienta, se comenzó con un promedio de 10 o 12 horas y se terminó con un mínimo de 20 horas semanales.

El período de investigación fue más largo de lo planificado (2 meses) debido al gran volumen de información recopilada, su análisis y clasificación. También en esta etapa fueron probadas varias de las herramientas encontradas lo que en algunos casos consumió bastante tiempo.
El estudio del entorno de desarrollo Java fue básicamente realizado en paralelo a las etapas de investigación y análisis de requerimientos pero se extendió a lo largo de todo el proyecto siendo más intenso en las etapas iniciales.

En el análisis de requerimientos resultó difícil precisar el alcance de la herramienta a construir, determinar el desarrollo del motor genérico y extensible, pero sobre todo que la elección de los pluggins a desarrollar fuera representativa de las posibilidades del producto. El tiempo insumido en esta etapa es destacado por el hecho de que si bien no estaba diferenciada en el cronograma inicial fue necesario dedicar un tiempo considerable a definir en forma precisa los requerimientos.

En las etapas anteriores la dedicación horaria fue menor a la estipulada inicialmente, esto también fue un factor que incidió en que no se cumpliera el cronograma original en forma estricta.

Según el cronograma original el trabajo de desarrollo estaba concentrado en los meses cuatro y cinco, en el nuevo cronograma se prolonga desde los meses seis al diez con un tiempo importante de pruebas paralelo al desarrollo en los meses nueve y diez. Es decir que se prolongó el trabajo planificado inicialmente a cinco meses y se dedicó más tiempo en los dos últimos meses (diez y once) a realizar las pruebas de integración y la redacción del informe final.

La definición y desarrollo del motor empezó un mes más tarde por los atrasos previos y se extendió por un período de cinco meses en total. Fue dedicado al estudio de la tecnología JMX y desarrollo de la aplicación de prueba para MX4J. Este tiempo en realidad posibilitó que con la elección de JMX y el prototipo de prueba se minimizara el tiempo de desarrollo de etapas posteriores. En el desarrollo del prototipo se resolvieron un conjunto de problemas adelantándose al desarrollo futuro del motor.

Se fueron haciendo pruebas en paralelo con el desarrollo lo cual permitió distribuir el tiempo dedicado a realizar las mismas, de acuerdo a este método de trabajo el tiempo estipulado en el cronograma para esta actividad no fue cumplido en forma estricta.

5. CONCLUSIONES

5.1 Evaluación del grupo de trabajo

Desde el punto de vista de la formación de los integrantes del grupo, se adquirió formación en el manejo de un conjunto de herramientas y se accedió a nuevas fuentes de conocimiento que fueron de gran utilidad tanto para el proyecto en sí como para trabajos futuros.

La fase de investigación permitió ampliar conocimientos en el área de las herramientas de administración y monitoreo de redes y gestión en general.

Se adquirió familiaridad con tecnologías y formas de trabajo en el área de las herramientas open source, explotando sus facilidades y ventajas. El hecho de disponer del código y modificarlo permitió un grado de flexibilidad importante ya que fue posible adaptarlo a las necesidades particulares de la herramienta a desarrollar.

Se encontró a Internet como una fuente muy importante de respuesta a problemas, ya sea a través de las FAQS o foros de preguntas y discusión.

El entorno de desarrollo Java y las herramientas empleadas, desde la utilización de herramientas de desarrollo, trabajar en el sistema operativo Linux, el manejo de versiones con CVS, extracción de métricas desde el código del producto, manejo de herramientas para consultas SNMP y aplicación de herramientas de monitoreo, aportó en gran medida a la formación de los integrantes del grupo.

Se adoptó una forma de trabajo ágil basada principalmente en la comunicación fluida, la que permitió un trabajo de intercambio permanente, planteando y resolviendo los problemas de forma rápida.

Por lo antes mencionado, la evaluación personal es muy positiva y creemos que se han cumplido las expectativas planteadas al comienzo del proyecto.

5.2 Evaluación del producto

Se cumplió con el principal objetivo del proyecto de construir el motor con las características requeridas. Este permite acoplar distintos módulos de recursos y sus atributos a monitorear, alertas y mecanismos de registro, de forma simple, lo que hace que la herramienta sea fácilmente extensible y mantenible.

El producto es multiplataforma, permite combinar los principales componentes que lo integran con independencia de la plataforma, es decir que un componente funcionando en windows (Por ejemplo: un Agente) se puede comunicar con un servidor o clientes que se están ejecutando en Linux, a su vez este agente puede estar monitoreando equipos con la plataforma Linux.
También los servidores se pueden comunicar con Agentes ejecutando en Linux y otros en Windows.
Se dedicó especial interés en verificar este comportamiento y hay una sección en el plan de pruebas dedicada a probar el funcionamiento de la integración de los componentes multiplataforma.

Se han desarrollado casi la totalidad de los pluggins de monitoreo detallados en los requerimientos específicos mínimos, con la salvedad de:

- Para algunas medidas de interés se estudiará la posibilidad de mantener históricos de promedios.
- Mantener archivos de log de acuerdo a un período configurable, a modo de ejemplo si el período es una semana y se mantienen archivos diarios, solo se almacenan uno por cada día y se sobrescribe un archivo por día. La forma de escritura de archivos sería circular.

La elección por una arquitectura distribuida amplía las posibilidades de uso de la herramienta, permitiendo soportar distintos tipos de escenarios según la aplicación específica, lo que la hace flexible.

Por otro lado es una arquitectura simple ya que se puede extender fácilmente mediante la incorporación de nuevos pluggins de monitoreo, de alarmas, acciones, o nuevos tipos de conectores.

Se generó documentación la cual provee información:
- De Instalación y configuración.
- Para el desarrollador proveyendo información de cómo integrar nuevos módulos.
- Para el usuario.
- De la API.

Todas éstas colaboran en sintetizar un producto de mejor calidad.

JMX es una nueva tecnología orientada a la administración y gestión de recursos muy potente. Define la arquitectura, los patrones, la API, los servicios y patrones de diseño para aplicaciones, monitoreo y gestión de red utilizando el entorno de programación Java. Ésta permitió desarrollar una herramienta con un grado de complejidad alto de forma simple y en un proceso relativamente corto.

5.3 Trabajos Futuros

Se dejan planteados una serie de trabajos futuros los que potenciarían en gran forma al resultado obtenido en este taller, pero que no se incluyeron en el alcance del proyecto.

- Extensión del esquema de monitoreo, con incorporación de una política de scheduling que permita establecer reglas de tiempo para el monitoreo de los recursos.

- Evolucionar la consola gráfica de monitoreo para:

    - Permitir ejecutar de forma manual, operaciones definidas sobre para cada ítem de monitoreo.
    - Desplegar distintos tipos de gráficos a partir de los datos registrados en la base de datos durante la actividad de monitoreo.
    - Ser más amigable e interactiva.

- Creación de nuevos pluggins de monitoreo y alerta. El desarrollo de nuevos pluggins de monitoreo y alertas que sean incorporados a los que ya han sido desarrollados, como forma de ampliar la cantidad de recursos monitoreados.

- Establecer prioridades asociadas a las alertas que permitan según éstas brindar múltiples caminos de notificación para un mismo evento.

- Monitoreo de reglas del negocio. Es de interés por parte de la empresa poder aplicar el sistema de monitoreo para el control de reglas del negocio en aplicaciones particulares que desarrolla o mantiene la empresa.

- Carga dinámica. Es deseable en un futuro desarrollar un mecanismo para la carga dinámica de nuevos recursos y alertas, esto implica implementar un mecanismo que permita reconocer e incorporar la especificación de un recurso y todos los elementos necesarios sin necesidad de detener el proceso del monitor o recompilar los fuentes de la aplicación.

- Monitor Pasivo. Escucha por eventos que disparan aplicaciones particulares, eliminando la necesidad de pooling sobre el recurso en particular. Es decir, implementar una interfase que permita a las propias aplicaciones avisar al monitor cuando se presenta una condición determinada.
  Esto se complementa con el monitoreo de reglas del negocio porque permite el seguimiento del normal funcionamiento de procesos de aplicaciones y mediante el uso de ciertos puntos de control tener conocimiento del estado del proceso o simplemente mientras no genera un evento avisando de que hay problemas es que el proceso sigue su curso normal.

6. GLOSARIO

API: Aplication Programming Interface.

HTTP: HyperText Transfer Protocol.

XML: Extensible Markup Languaje.

SOAP: Simple Object Access Protocol protocolo basados e XML que describe como se envían y reciben mensajes desde servicios WEB.

TMN: Telecommunication Managed Network, entorno que garantiza la interconexión y comunicación entre sistemas y redes de telecomunicaciones heterogéneas.

IIOP: Internet Inter-ORB Protocol.

JNI: Java Native Interface.

JSR: Java Specification Request.

RMI: Remote Method Invocation.

MBeans de recursos: Representan un recurso que el sistema puede monitorear y controlar Puede ser un dispositivo, recursos del sistema, proceso, componente de una aplicación.

MBeans de servicios: MBeans utilizados por otros MBeans, pero no representan un recurso del sistema.

SDL: Lenguaje de Descripción y Especificación. Lenguaje orientado a especificar y describir el comportamiento de Sistemas de Telecomunicaciones. Ver referencias.

ICMP: protocolo de control de mensajes de Internet.

ARM: Application Response Measurement

DSI: Data Source Integration.

IETF: Internet Engineering Task Force. Más información en http://www.ietf.org/

ISPs: Internet Service Providers.

IT: Information Technology.

RFC (Request For Comments): Conjunto de documentos que contiene los protocolos de Internet y discusiones de temas relacionados.

SNMP (Simple Network Management Protocol, Protocolo simple de administración de red)

SNMP v2c: Consiste del protocolo SNMP v2 con el modelo de seguridad especificado en RFC 1910.

SNMP Community (Comunidad SNMP): Es una especie de usuario o password, que permite realizar solicitudes SNMP a un dispositivo que soporte SNMP. Cuando se realiza una consulta, si la comunidad es correcta, el dispositivo responde, de lo contrario se descarta la solicitud y no se responde.
MIB. Management Information Base. Base de datos de objetos que permite la gestión mediante SNMP.

OID: Object Identifier, identificador de un objeto en la MIB.

UDP (User Datagram Protocol, protocolo de datos de usuario): Protocolo estándar TCP/IP que permite a un programa de aplicación en una máquina enviar un datagrama hacia el programa de aplicación en otra máquina. El UDP utiliza el protocolo de Internet (IP) para entregar datagramas. El envío no es confiable, y los mensajes no son entregados necesariamente en el orden en que se envían.

TCP/IP (Transmission Control Protocol /Internet Protocol): grupo de protocolos de Internet utilizado para comunicarse a través de cualquier grupo de redes interconectadas.

Transmission Control Protocol (TCP): es el protocolo que arma los datos en paquetes de manera tal, que la red los pueda manejar eficientemente, verifica que todos los paquetes lleguen a su destino, ordenándolos a medida que los va recibiendo con la secuencia correcta y si un paquete está dañado reconstruye los datos a su forma original. Se lo llama orientado a conexión, porque establece una conexión lógica entre hosts antes de iniciar una conversación.

Internet Protocol (IP): Protocolo estándar que define a los datagramas IP como la unidad de información que pasa a través de una red de redes y proporciona las bases para el servicio de entrega de paquetes sin conexión.

Host (anfitrión): Cualquier sistema de computadora de usuario final que se conecta a una red. Los anfitriones abarcan desde computadoras personales hasta supercomputadoras, impresoras de red, cámaras de video, etc.

Flujo de red: Secuencia unidireccional de paquetes entre un origen y un destino de una terminal.

Paquete: Se trata, en términos generales, de bloques pequeños de datos enviados a través de una red de conmutación de paquetes.

Tabla de ruteo: Tabla utilizada en algoritmos para el ruteo IP. Almacena información sobre posibles destinos y sobre cómo alcanzarlos.

Socket: Abstracción proporcionada por el sistema operativo Unix que permite a un programa de aplicaciones acceder los protocolos TCP/IP.

PING (Packet Internet Groper): Nombre de un programa utilizado con las redes TCP/IP que se usa para probar la accesibilidad a un destino, enviando una solicitud de eco ICMP y esperando una respuesta.

Hub Dispositivo electrónico al que se conectan varias computadoras, por o general mediante un cable de par trenzado. Un concentrador simula una red que interconecta a las computadoras conectadas.

Router (ruteador): Computadora dedicada, de propósito especial, que se conecta a dos o más redes y envía paquetes de una red a otra. Un ruteador utiliza las direcciones de destino en un datagrama para decidir el próximo salto al que enviará el datagrama.

Switch: Dispositivo que permite filtrar y enviar paquetes entre segmentos LAN. Opera a nivel de capa 2 en el modelo de referencia OSI.

Datagrama IP: unidad básica de información que pasa a través de una red de redes TCP/IP. Contiene las direcciones de fuente y destino junto así como los datos.

Sampling de paquetes: ver http://www.sflow.org/packetSamplingBasics/index.htm

Recurso: en este contexto consideramos recurso a toda entidad que necesite ser administrada. Podría ser una aplicación de negocio, un dispositivo, un servicio.

K&S Middleware: Producto desarrollado por la empresa K&S, que consiste en una pieza de software capaz de brindar conectividad e integración en los niveles físico y lógico entre distintos sistemas, que posiblemente funcionen en distintas plataformas.
Visto como una interfase universal puede definirse como una capa de software capaz de abstraer la complejidad y la heterogeneidad de las actuales infraestructuras distribuidas, para ofrecer una interfase de más alto nivel a las aplicaciones finales, esta interfase permite consumir los diferentes objetos del negocio de una forma consistente y segura.

K&S Adaptor Java: Producto en desarrollo en la empresa K&S, implementado en Java, principalmente para utilizar en la plataforma Linux. Es compatible con el K&S Middleware en cuanto a su funcionalidad.

Petición de Middleware: Es la unidad básica de ejecución del Middleware. El rol principal del servicio MiddleWare es estar a la espera de conexiones de clientes (establecimiento de una sesión), una vez establecida la conexión este puede ejecutar peticiones y desconectarse.
Una *petición* al MiddleWare puede verse como una abstracción de alto nivel de los procesos del negocio. Estas se componen de un conjunto de tareas, especificadas adecuadamente en una base de conocimiento, que el MiddleWare ejecuta al recibir un requerimiento de ejecución de la *petición*.
Estas tareas pueden ser funcionalmente: consultas a distintas fuentes de datos, ejecución de procedimientos almacenados, escritura y recuperación de colas de mensajes, ejecución de componentes locales o remotos, instanciación de procesos.

7. REFERENCIAS

[HMAC] Definición del proyecto de grado: Monitor de Alertas configurable.

[GID] Guía de Instalación y configuración del Monitor de Alertas configurable.

[GD] Guía del Desarrollador del Monitor de Alertas configurable.

[DU] Documentación de Usuario del Monitor de Alertas configurable.

[JDOC] JavaDoc Generado durante el desarrollo del proyecto de grado Monitor de Alertas configurable.

[JMX]
http://java.sun.com/products/JavaManagement/ fecha último ingreso: 12/04/2004

[JSR 160]
http://jcp.org/en/jsr/detail?id=160 fecha último ingreso: 12/04/2004

[JSR 003]
http://jcp.org/en/jsr/detail?id=003 fecha último ingreso: 12/04/2004

[MX4J]
http://mx4j.sourceforge.net fecha último ingreso: 12/04/2004

[XMOJO]
http://www.xmojo.org fecha último ingreso: 12/04/2004

[SNMP]
http://edge.mcs.drexel.edu/GICL/people/sevy/snmp/ fecha último ingreso: 12/04/2004

[RMON]
http://www.chaco.gov.ar/UTN/AdmRedes/Traduccion/cap9.doc fecha último ingreso : 04/08/2003
Federico D. R. Navarro, capítulo 9.

[netFlow]
http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/iosnf_ds.htm
fecha último ingreso : 04/08/2003
http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm
fecha último ingreso : 04/08/2003
Posted:15/07/2002
All contents are Copyright © 1992--2002 Cisco Systems, Inc. All rights reserved.

[sFlow]
http://www.sflow.org/sFlowOverview.pdf fecha último ingreso : 04/08/2003
All contents are Copyright © 2003 sFlow.org, Inc. All rights reserved.

[RMON, netFlow, sFlow]
http://www.inmon.com/PDF/EmbeddedTM.pdf fecha último ingreso : 04/08/2003
Copyright © 2003 InMon Corp.

CA- UNICENTER
http://www.cai.com/offices/spain/local/unicenter.htm fecha último ingreso : 04/08/2003
http://www.cai.com/offices/spain/local/unicenter.pdf fecha último ingreso : 04/08/2003
http://www.tecnopro.com.ar/esp/ca/unicenter.html fecha último ingreso : 04/08/2003

3COM NetworkSupervisor
http://www.3com.com/products/en_US/detail.jsp?tab=features&pathtype=purchase&sku=3C15100D
fecha último ingreso : 04/08/2003
http://www.3com.com/products/en_US/detail.jsp?tab=prodspec&sku=3C15100D&pathtype=purchase
fecha último ingreso : 04/08/2003
http://www.3com.com/other/pdfs/products/en_US/400731.pdf
fecha último ingreso : 04/08/2003

exe trial
http://support.3com.com/software/3com_network_supervisor_v4_0_1.exe
fecha último ingreso : 04/08/2003

Herramientas Open Source de Gestión
http://public.planetmirror.com/pub/opennms/whitepapers/OpenSourceNetworkManagementTools.pdf
fecha último ingreso : 31/12/2003

[HP]
http://www.openview.hp.com/ fecha último ingreso : 12/04/2004

[HP1]
http://www.openview.ru/products/ovperf_pb_mar03.pdf fecha último ingreso : 04/08/2003
Copyright © 2003 Hewlett- Packard Development Company. LP. Febrero 2003.

[HP2]
http://www.managementsoftware.hp.com/products/extensible_snmp_agent/briefs/snmp_pb.pdf
fecha último ingreso : 04/08/2003
Copyright © 2001 Hewlett- Packard Company. LP. Junio 2001.

[MRTG]
http://people.ee.ethz.ch/~oetiker/webtools/mrtg/ fecha último ingreso : 12/04/2004

[Perl]
http://www.perl.com/ fecha último ingreso : 02/05/2004

[GIF] Getif 2.x Network Tool
http://www.wtcs.org/snmp4tpc/getif.htm fecha último ingreso: 19/04/2004

[SKC] SNMP Construction Kit
http://membres.lycos.fr/ysoun/ fecha último ingreso: 12/04/2004

NetSNMP
http://www.net-snmp.org/ último ingreso: 12/04/2004

[Javadoc]
http://java.sun.com/j2se/javadoc/ fecha último ingreso: 12/04/2004

[CVS]
http://www.cvshome.org/ fecha último ingreso: 12/04/2004

[Statcvs-xml]
http://statcvs-xml.berlios.de/ fecha último ingreso : 02/05/2004

[Jude]
http://objectclub.esm.co.jp/Jude/jude-e.html fecha último ingreso: 12/04/2004

[JUNIT 1] Junit Primier
http://www.clarkware.com/articles/JUnitPrimer.html fecha último ingreso: 12/04/2004

[Eclipse]
http://www.eclipse.org/ fecha último ingreso: 02/05/2004

[SUSE]
http://www.suse.com/us/ fecha último ingreso: 02/05/2004

Bibliografía

Redes de Computadoras.
Andrew S.Tanenbaum.
Tercera Edición.

Redes globales de información con Internet y TCP/IP
Principio básicos, protocolos y arquitectura.
Douglas E.Comer.
Tercera edición.

8. APÉNDICES

8.1 ESTADO DEL ARTE

8.1.1 INTRODUCCIÓN

Se va a hacer un estudio de algunas soluciones existentes en el mercado. El objetivo de documento es una conocer el estado del arte de las formas en que se intentan resolver los problemas de gestión y administración de recursos. De las soluciones que se van a ver, algunas fueron propuestas por los tutores, otras surgieron a partir de búsquedas en Internet que se considera interesante incluir.

Esta discusión se presenta como complemento a la sección desarrollada en la etapa de "Relevamiento de estado del arte" en el presente informe, en la que se desarrollaron las herramientas consideradas más relevantes o las que se estudiaron más en profundidad.

Existen distintos tipos de soluciones que difieren tanto en precio como en los problemas que resuelven.
Algunas de estas herramientas se encargan de administrar los recursos de red, y otras abarcan la gestión y administración de distintas áreas de la empresa, como ser monitoreo de aplicaciones, monitoreo de utilización de recursos y aplicaciones por usuarios, análisis y planificación estratégica de recursos.

8.1.2 ALCANCE

En este documento se presentarán distintas herramientas que dan solución a los distintos problemas o puntos de interés que se presentan de administrar servicios, componentes y recursos de red.

En una primera parte se mencionaran el estudio de herramientas open source en la administración de redes y sus distintos enfoques.

En un segunda parte estudiaremos un grupo de herramientas con una tecnología propietaria en algunos aspectos y que son muy costosas económicamente. Describiremos sus principales características desde el punto de vista funcional, no se profundizará en la implementación o arquitectura de las mismas.

Como tercera parte se estudiarán diferentes tecnologías que se aplican en el monitoreo de red, SNMP, RMON, netFlow y sflow.

A modo de conclusión haremos un resumen de las posibilidades y limitantes de las herramientas estudiadas.

8.1.3 HERRAMIENTAS OPEN SOURCE

En este capítulo presentaremos un conjunto de herramientas open source, basándonos en el artículo "Network Management: Open Source Solutions to Proprietary Problems".

Varias empresas ofrecen soluciones integradas para el manejo de distintos tipos de recursos, ofreciendo alta performance, correlación de eventos y altos grados de automatismo.
Un problema básico en este tipo de soluciones es el alto precio, es por esto que surgen soluciones alternativas que intentan reducir esos costos con herramientas que cumplan con las funcionalidades básicas o reemplazan las soluciones costosas con alternativas no cotosas disponibles en la web que proveen una solución rápida, parcial o temporal a problemas determinados.

Es en este sentido es que surgen herramientas open source que enfocan distintos tipos de necesidades, en el caso de gestión y administración de redes.

En el artículo de referencia se plantea una clasificación de las herramientas en cinco categorías:

1. Agentes SNMP
    i. cmusnmp
    ii. ucdsnmp
    iii. netsnmp

2. Herramientas de presentación de colección de datos.

La base de estas herramientas es que fácilmente permiten recolectar datos con snmp. Recogen los datos mediante snmp, los guardan en una base de datos consolidada y grafican los resultados en formato html

    i. MRTG
    ii. RDDtoll
    iii. Cricket. Posterior a rddtoll, combina MRTG y webTV

3. Herramientas de mapeo de redes
   Este grupo de herramientas monitorea la red y la presenta en una consola gráfica
    i. Scottty/thinked
    ii. Cheops. Escanea puertos e identificación(seguridad)

4. Herramientas de análisis de protocolos de red
    i. TCPdump
    ii. SNORT. Captura nodos y ejecuta logging de análisis de la red (basado en TCP/IP) y detección de intrusos
    iii. Ethereal
    iv. Iptraf. Información estadísticas de conversación entre protocolos, análisis de alto nivel para la captura de paquetes
    v. SNMP_sniff

5. monitores de redes y sistemas
    i. GxSnmp
    ii. Event monitor proyect. Centraliza la administración de eventos.
    iii. MON. Monitoreo y notificación (cliente/servidor), componentes, interfase web y productos independientes del monitor que son invocados por el cliente. Muy portables.
    iv. BigSister. Cliente/servidor
        1. Agente.
            a. monitoreo distribuido.
            b. actualiza el Colector de Estado.
            c. es fácilmente extensible la cantidad de servicios de plataformas que no están siendo monitoreadas por defecto.
        2. Colector de estado. Log de eventos.
        3. Servicio de notificación.

8.1.4 HERRAMIENTAS PROPIETARIAS

En este capítulo presentaremos dos herramientas propietarias.
Para la elección de las mismas nos enfocamos en herramientas que permitieran la administración de recursos, tanto para análisis como para planificación, que no solo estuvieran enfocadas en el monitoreo de redes en sí.

8.1.4.1 HP Openview [HP]

8.1.4.1.1 HP Openview Performance Manager, Performance Monitor, Performance Agent.

Los productos HP OpenView Performance ofrecen una solución flexible para la administración de recursos y performance, tanto para un único sistema o una red de varios sistemas. HP OpenView Performance Manager, Performance Monitor y Performance Agents ofrece una única interfaz de monitoreo central, para análisis y prevención de la utilización de recursos para ambientes distribuidos multi-vendedor[HP1].

Estos productos permiten responder preguntas como; ¿la aplicación está activa? ,
¿son aceptables los tiempos de respuesta de un usuario final?, ¿son aceptables los niveles de servicio ofrecidos?.

Las áreas de las que se encarga son las siguientes:

1. Administración de Performance y resolución de problemas de forma pro-activa.

El Performance Agent recolecta datos para detectar condiciones de excepción.
Las condiciones de excepción se pueden basar en medidas individuales o por combinación de medidas y pueden ser definidas usando umbrales y duración, permitiendo detección de problemas de forma pro-activa. Por ejemplo, una condición de excepción puede definirse cuando el tiempo de respuesta de una aplicación cliente/servidor excede cierto umbral predefinido o cuando el uso de CPU sobrepasa el 75% y largo de la cola de espera es más que 3 por más de 5 minutos. En tal caso el Performance Agent detecta y alarma en una condición de excepción.

El Performance Monitor recibe y mantiene una lista de alarmas de performance que hayan ocurrido en distintos sectores del ambiente distribuido.
Para solucionarlo, selecciona la alarma en el log de mensajes y observa como el Performance Manager dispara y provee información detallada describiendo la condición de excepción, incluyendo una vista de medidas de performance necesarias para identificar las causas de los problemas.

2. Planificación de recursos y administración de servicios.

HP OpenView Performance provee información para determinar cómo es la performance de los recursos y cómo están siendo usados.

El Performance Agent se encarga de recolectar, sumarizar y registrar mediciones de recursos y performance de aplicaciones, bases de datos, redes y sistemas operativos. Usando los datos históricos, permite examinar la utilización de los recursos y la tendencia de performance.

Con esta información, se pueden por ejemplo descubrir cuellos de botella, comparando actividades se puede balancear sobrecarga de actividades, permitiendo ubicar recursos y distribuir servicios.

Ahora vamos a describir de que se encarga cada uno de los componentes.

8.1.4.1.2 HP OpenView Performance Manager

Para UNIX

Es una herramienta gráfica de análisis y planificación diseñada para analizar y hacer proyecciones futuras sobre la utilización y ver las tendencias de performance de los recursos.

Utiliza los datos históricos recolectados por los Performance Agents para proveer:

- la comparación y correlación de medidas de recursos de aplicación y del sistema
- métricas o gráficos para períodos específicos de tiempo
- creación de gráficos personalizados y plantillas que contienen métricas de distintos sistemas
- creación de copias de gráficos, incluso soporta impresoras de PostScript
- prever la capacidad basándose en datos coleccionados por el HP OpenView Performance Agent, incluyendo proyecciones lineales, exponencial.

Para Windows

Es una herramienta de análisis y visualización diseñada para analizar las tendencias de performance de las aplicaciones, sistemas y servicios.

Utiliza los datos recolectados por los Performance Agents y otras fuentes para aislar cuellos de botella y maximizar el tiempo de actividad de los recursos. Además provee:

- una interfase gráfica de usuario configurable
- visualización de reportes separado por cliente
- gráficas en tiempo real

8.1.4.1.3 HP OpenView Performance Monitor para UNIX

Permite configurar acciones según las alarmas. Cuando recibe una alarma, se puede iniciar el monitor para automáticamente crear un diagrama conteniendo métricas para resolver problemas de performance. También provee, filtro de alarmas según la severidad, por tipo o nodo del sistema y actualizaciones de estado de alarma a intervalos.

8.1.4.1.4 HP OpenView Performance Agent

Es el componente encargado de recolección, logueo y alarma. Se instala en cada sistema a ser monitoreado y provee:

- recolección y logueo de aplicaciones, procesos y métricas del sistema.
- sistema de alarma basado en combinación de métricas, definición de síntomas y/o por duración de tiempo o umbrales.
- soporte del estándar Application Response Measurement (ARM) para administración end-to-end.
- incorporar tecnología Data Source Integration (DSI), pudiéndose integrar los datos a otros orígenes.
- exportar datos en distintos formatos standard, incluyendo ASCII y binario.
- generar y enviar alarmas como SNMP.
- tomar acciones locales disparadas por alarmas.
- soporta múltiples plataformas, incluyendo HP-UX, SunOS/Solaris, IBM AIX, Tru64 UNIX y Microsoft Windows.

8.1.4.1.5 Integración con otras soluciones HP OpenView

Los productos HP OpenView Performance pueden ser usados standalone, o integrados con HP OpenView Operations and Network Node Manager (NNM). Los Agentes pueden desplegarse desde el Operations Manager. Una vez desplegado, las alarmas son enviadas al manager de operaciones, permitiendo tener en una misma interfase la combinación de datos de operaciones y performance.
Seleccionando un evento, se puede activar el Performance Manager para ver ciertas condiciones del sistema en el momento que se produce el evento.

También, si Network Node Manager y HP OpenView Performance son instalados en el mismo sistema, el Performance Monitor se integra de forma transparente en el mapa topológico del NNM.
Cuando una alarma es recibida, el Network Node Manager identifica el operador, cambiando el estado del ícono del dispositivo con problemas. El Performance Manager puede ser activado automáticamente desde el NNM para obtener más información referente a la alarma.

8.1.4.1.6 HP OpenView Extensible SNMP Agent

El Agente Extensible SNMP amplía las capacidades de administrar de SNMP para controlar los dispositivos básicos de la red, los sistemas y aplicaciones críticas [HP2].
Además de administrar dispositivos como routers, bridges y hubs, permite administrar aplicaciones, impresoras, usuarios, y bases de datos.

Es fácil de extender ya que se pueden configurar nuevos objetos SNMP sin programar. Los objetos SNMP definidos por el usuario en el Agente Extensible, pueden ser configurados tanto para SNMP GET o SNMP SET. Esto permite a un Administrador de red no solo monitorear sino controlar los recursos de la red y de los sistemas.

El uso de estándares como Internet Protocol (IP), User Datagram Protocol (UDP), Simple Network Management Protocol Versión 1 (SNMPv1), y Community-based SNMPv2 (SNMPv2C) permiten que se integre a cualquier entorno de administración de redes, tanto si es administrado por el HP OpenView Network Node Manager u por otra aplicación de administración de red basada en SNMP.

El Agente Extensible se adapta a las especificaciones de SNMP para facilitar la interoperabilidad con soluciones de gestión tanto de HP como de otros vendedores. Cumple con RFC 1157 (SNMP), RFC 1187 Bulk Table Retrieval con SNMP, RFCs 1901-1908 Community-based SNMPv2 protocol y RFCs 1212 y 1213 que cubren los formatos y conjuntos de objetos de la MIB II.

El Agente Extensible es soportado en varias plataformas, está disponible en versiones de servidores y estaciones de trabajo HP y en sistemas SunSPARC.

También se pueden definir interrupciones SNMP. El usuario tiene que establecer cuales son las condiciones de alarmas, preparar un procedimiento para monitorear la condición, y notificar al administrador si la condición ocurre.
Esto provee al administrador de más información inmediata sobre problemas potenciales.
Esta funcionalidad se podría combinar con la de configurar eventos del Network Node Manager, para tomar acciones especiales al arribar las interrupciones.

El agente puede enviar alarmas para interrupciones estándar a múltiples administradores al mismo tiempo, de forma automática. También puede responder a una solicitud de GET y SET de más de un administrador, permitiendo al usuario personalizar el sistema para utilizar de forma eficiente los recursos de la red.

8.1.4.2 CA Unicenter (TNG)

Sistema de Administración, Gerenciamiento y Seguridad de Recursos Informáticos, Gestión empresarial.

8.1.4.2.1 Descripción

Este producto se presenta como una solución escalable, flexible, totalmente automatizada y con posibilidades de emplear módulos acoplables para gestionar distintas áreas de una empresa con un rendimiento máximo. Algunas de las áreas sobre las que brinda soluciones son:

Gestión empresarial multiplataforma que incluyen Gestión de Red y Sistemas, Automatización de Operaciones, Gestión de Recursos TI, Gestión de Bases de Datos, Gestión de Infraestructura Web y Gestión de Aplicaciones.

Es una herramienta modular de instalación directa que unifica distintos componentes de gestión y control a demanda (seguridad, almacenamiento, manejo de eventos, planificación de recursos, entorno de desarrollo integrando distintas plataformas).

Este producto se presenta como independiente de plataformas y aplicable en cualquiera de ellas, esto es destacable ya que permite gestionar recursos y dispositivos de todos los fabricantes, favoreciendo la interoperabilidad.

Los usuarios se pueden adaptar a cualquier mecanismo de trabajo independizándose en cierto sentido del hardware y programas de aplicación. Tiene el mismo aspecto en todas las plataformas manteniendo una terminología y funcionalidad consistente, permitiendo de esta manera la adopción de políticas uniformes en todo el sistema.

8.1.4.2.2 Características generales

1. Gestión de red y sistemas
    i. Disponibilidad del SO y elementos de infraestructura como dispositivos de red.
    ii. Control de eficiencia del servicio IT, generación de reportes.
    iii. Rendimientos de servidores, dispositivos, tiempos de respuestas de la red, ancho de banda y rendimiento de aplicaciones.
    iv. Garantiza buen estado y fiabilidad de redes TCP/IP, Ipx, SNA, DecNet, ATM.
    v. Planificación de recursos.
2. Gestión de infraestructura web
3. Gestión de aplicaciones
    i. Monitorea en forma automática parámetros específicos de las aplicaciones y asegura la disponibilidad y el rendimiento de aplicaciones como Peoplesoft, SAP R/3 así como de los sistemas, redes y bases de datos subyacentes.
4. Gestión de recursos IT
5. Gestión de Operaciones automatizadas
6. Gestión de bases de datos
    i. Rendimiento de bases de datos

ii. Administración de bases de datos

iii. La solución Unicenter Database Administration automatiza y simplifica las tareas cotidianas del DBA a la vez que le permite mantener el control total del entorno de bases de datos en sus manos.

iv. Backup y recuperación. La solución Unicenter Database Backup and Recovery ofrece la tranquilidad que se obtiene al contar con rutinas de backup y recuperación rápidas, eficientes y fiables.

8.1.4.2.3 Funciones generales de CA-Unicenter

Permite realizar la administración del sistema desde un solo punto, de esta manera el administrador puede controlar la funcionalidad del Unicenter desde una única estación de trabajo.

1. **Administración de la Seguridad del Sistema.** Fácil de administrar y seguro. Integridad y privacidad de los datos y programas. Prevenir accesos de individuos no autorizados al sistema. Asegurar que los datos y recursos solo sean accedidos por personal autorizado. Protección de los datos y recursos ante posibles abusos de privilegios por los usuarios del sistema. Auditorias de uso de los recursos.

2. **Manejo de Eventos, Estado (status) y Excepciones del Sistema**. Cada consola recibe mensajes del sistema operativo y de los procesos los cuales pueden corresponder a variedad de situaciones, desde un agotamiento de la capacidad de disco a una falla en un proceso. Unicenter permite: rutear los mensajes a una consola centralizada según reglas especificas; resaltar, codificar con colores, e incluso retener mensajes críticos; Permite definir acciones a tomar en caso de recibir mensajes específicos o si un mensaje llega con una frecuencia determinada.

3. **Seguimiento de Problemas (Help Desk)**. Repositorio de problemas, con los datos de los mismos, y las soluciones adoptadas, permitiendo realizar consultas de información histórica. Permite asignar los problemas para que los resuelva determinada persona, relacionándolo a la causa o a algún hardware o software determinado. Los problemas tienen prioridades, los cuales son escalados o reasignados a una nueva persona si no se resuelven dentro de un período de tiempo determinado. Permite la generación automática de problemas basada por ejemplo en eventos del sistema.

4. **Manejo de Cintas**. La destrucción inadvertida de información almacenada en cinta puede ser de alto costo para una empresa, tanto en dinero como en pérdida de productividad, de confianza por parte de sus clientes. Marca las cintas con una identificación digital, de tal manera de identificar y proteger los datos. Ante un intento de reusar la cinta el Unicenter descubre la marca e impide su uso. Mantiene una base de datos de cintas que contiene los archivos almacenados y su ubicación en depósito. Detecta automáticamente el momento en que la cinta puede ser reusada, ya que a cada una se le anexa un tiempo de expiración, el cual estará ligado al tiempo de expiración asignado a cada archivo almacenado en ella. El período de retención puede ser mensual, diario, anual, permanente o por cantidad de versiones. Permite (Opcional) impresión de código de barras para etiquetar las cintas.

5. **Manejo del File System.** La información producida diariamente por la empresa es almacenada en disco. Toda organización necesita realizar copias de seguridad de sus datos en forma regular, de tal manera de poder recuperarlos en caso de una emergencia. Los procedimientos deben estar lo suficientemente aceitados para permitir una rápida recuperación de la información archivada en el momento que se necesite. Unicenter puede identificar los datos a ser respaldados, determinar con que frecuencia varían esos datos, y decidir cada cuanto tiempo realizar una copia de seguridad, así como determinar cuantas versiones anteriores y cuales se quiere guardar. Permite realizar respaldos totales, incrementales y diferenciales. Como medio de sustento físico el Unicenter es compatible con los estándares del mercado: cinta, diskette, disco, disco óptico WORM, disco óptico borrable. Automatiza los pasos a seguir para una recuperación total desde cero de una máquina.

6. **Automatización de la Carga de Maquina** (Automating System Workload). Permite automatizar los procesos que corren en background, creando dependencias entre los mismos, y teniendo en cuenta en qué nodo se debe ejecutar el mismo. El Unicenter ejecuta el proceso si se cumplen determinados prerrequisitos como ser: prioridad en el scheduling, que se cumpla un calendario, es decir a partir de una hora determinada y/o antes de una hora determinada, que hayan terminado otros procesos, que se libere un recurso como el caso de una impresora, etc. Cuando se automatiza un determinado proceso, puede haber tareas que se deban realizar manualmente, para lo cual el sistema permite incorporar una tarea de confirmación manual pre y post CPU. Permite una visualización detallada de los procesos en espera, en ejecución, finalizados, y de los bloqueados por alguna causa.

7. M**anejo de la cola (spool) de impresión** Habilitar o deshabilitar la impresión, aceptar o no nuevos trabajos, históricos de las impresiones.

8. **Seguimiento o Tracking de los Recursos del Sistema.** Monitorear los recursos y la performance del sistema. Una evaluación del consumo de recursos puede ayudar a una organización a evaluar los costos de los mismos y las necesidades futuras. Determinar qué usuarios utilizan determinados recursos. Determinar si los recursos actuales son suficientes para requerimientos futuros.

8.1.4.3 CA- Unicenter TNG

Se han incorporado unas nuevas funcionalidades al CA-Unicenter , llamado TNG, que incorpora las siguientes características:

1. ***Posibilidad de Administración centralizada de toda la infraestructura***, incluyendo sistemas, redes, bases de datos, y aplicaciones.

2. La habilidad para visualizar el ambiente computacional, con vistas por proceso, o arquitectura.

3. ***Monitoreo y administración basados en el paradigma Agent-Manager***, de toda la red desde computadoras de escritorio hasta el servidor.

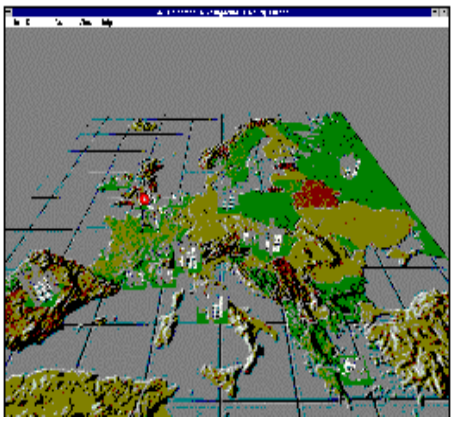
4. Las publicaciones en forma abierta y gratuita del sistema de desarrollo ***SDK***, permite que día a día haya disponibles en el mercado nuevas aplicaciones que trabajen íntimamente con el Unicenter.

5. El Unicenter TNG contiene una herramienta de desarrollo de agentes denominada ***Agent Factory*** que permite a los administradores desarrollar sus propios agentes para monitorear lo que deseen, enviando y recibiendo mensajes SNMP (Simple Network Management Protocol). El master realiza un pooling de los agentes y les envía comandos. Ante algún problema el agente envía un mensaje de reporte al master.

6. ***End To End Management:*** El Unicenter TNG integra la administración de recursos de la empresa: Sistemas, redes, bases de datos, aplicaciones. Permite una administración global de los recursos, con políticas de administración, seguridad y respaldo claras y concisas.

7. ***Business Views:*** Permite filtrar y agrupar los recursos de la empresa según la actividad productiva, permitiendo de esta manera controlar que todos los recursos computacionales clave para la actividad, se encuentren funcionando correctamente.

8. ***Real World Interface:*** Permite presentar la información de los recursos computacionales en forma gráfica para una fácil y rápida visualización. De esta manera se puede acceder a la información navegando por los recursos. Permite conocer rápidamente el estado de toda la estructura, por ejemplo si un router tiene algún problema se verá con un cambio de color en el icono correspondiente al mismo.



9. ***El Unicenter es una Solución Completa:*** El TNG tiene un conjunto de herramientas de administración desarrolladas sobre una estructura muy flexible. La arquitectura del TNG es basada en objetos, tiene una base de datos que actúa como repositorio de las características de los objetos, y una infraestructura del tipo manager/agent. La interfase 2D y 3D permite en forma intuitiva el acceso a los datos críticos de la red completa de la empresa. Visualizando los procesos que corren en cada máquina, con un golpe de vista se puede conocer el estado de una impresora, si una torre de cintas necesita el reemplazo de sus cintas, o si determinado usuario se encuentra logueado en la red. Simplemente la interfase 3D mapea los estados posibles de los objetos en información visual de fácil interpretación. Una administración conjunta y coherente, independiente de la plataforma, permite la implementación de una administración basada en reglas uniformes para toda la empresa. Evitando uno de los problemas mas graves de cualquier ambiente con máquinas diversas como es el de administrar cada plataforma en forma coherente con el resto.

8.1.5 Tecnologías de monitoreo de red

8.1.5.1 RMON, NetFlow, Sflow

En esta sección se resume un estudio realizado por InMon Corp sobre tecnologías para el monitoreo del tráfico para redes que utilizan switches [RMON, netFlow, sFlow]. Según el estudio, actualmente hay principalmente tres opciones RMON, Netflow y Sflow. Previo a la comparación de las mismas vamos a describir un poco más en detalle cada una de las tecnologías.

Background

En el pasado una gran cantidad de máquinas solían estar conectadas a una red compartida. Este tipo de red permite que un dispositivo conectado a la red pueda monitorear todo el tráfico (figura 1).
Esto fue cambiando con el tiempo debido a requisitos de aumentar el ancho de banda, cambios en los patrones de tráfico, baja de los precios de paquetes de switches y routers, por lo que ha habido un movimiento hacia redes altamente segmentadas donde el tráfico no es visible desde un solo punto (figura 2). La idea es que en estas redes un switch dirige los paquetes a puertos específicos basados en el destino del paquete, entonces para obtener una vista completa del tráfico de la red es necesario monitorear cada puerto dentro de los switches.

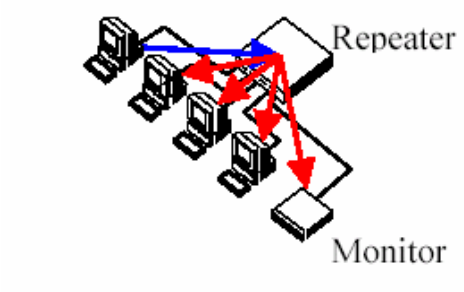


Figura 1

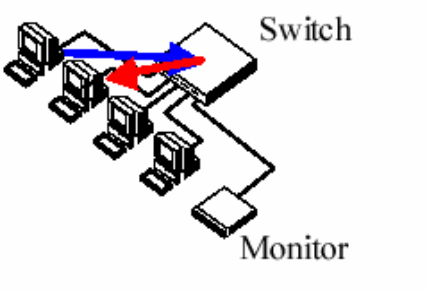


Figura 2

Tecnologías

Las tres tecnologías que se presentan en el artículo son:

RMON (Monitor Remoto) es un estándar de Internet Engineering Task Force (IETF) que especifica un dispositivo de monitoreo de tráfico remoto. Un dispositivo RMON monitorea y descifra cada paquete de la red y crea tablas de medidas que después pueden transmitirse a alguna aplicación de administración de red.

NetFlow® Cisco, es un sistema de monitoreo el cual envía información del flujo del tráfico a un colector central. Descifra cada paquete IP, mantiene tablas de flujos activos, y envían información periódicamente a una aplicación de administración de red.

sFlow sFlow combina contadores exactos de paquetes con muestreo estadístico del estado de las tablas de ruteo, para que el switch envíe paquetes de forma randómica. La muestra seleccionada es enviada inmediatamente a un colector central para ser analizado.

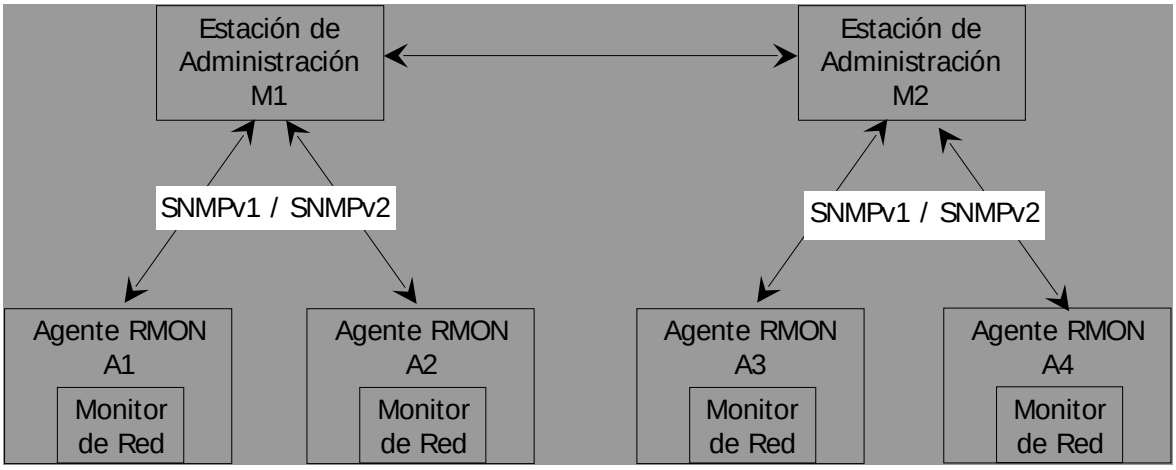
8.1.5.1.1 RMON

Descripción

Es una especificación de Monitoreo estándar que permite que varios monitores de red y sistemas de consola intercambien datos sobre el monitoreo de la red. RMON ofrece a los administradores de red mayor libertad al seleccionar sensores y consolas para el monitoreo de red con características que satisfagan sus necesidades particulares de conectividad. La especificación RMON es fundamentalmente una ampliación de MIB. El efecto sin embargo es el de definición de un conjunto de estadísticas y funciones que se pueden intercambiar entre los administradores de consola que cumplen con RMON y los sensores de red. RMON proporciona a los administradores de red información muy completa acerca del diagnóstico de fallas de red, planificación y puesta a tono del sistema.
El RMON es capaz de monitorear un segmento Ethernet y transmitir información estadística de regreso a la consola para RMON.

RMON se convirtió en 1992 en una proposición de standard para Ethernet en el RFC 1757. Luego en 1995 se volvió un standard en RFC 1757.

Administración con RMON



Los datos de administración recolectados por un monitor de red son enviados a las estaciones de administración. La ubicación de los monitores de red es dependiente de la implementación: puede colocarse un monitor de red por segmento, por subred, o por red token-ring.

Los monitores remotos de red se ubican en los Agentes RMON, que proveen funciones de soporte de protocolos de comunicación con las estaciones de administración y recuperación de errores, y actúan como intermediarios entre los monitores de red y las estaciones de administración. Para transferir los datos a las estaciones de administración pueden usar protocolos SNMPv1 o SNMPv2.
Un Agente RMON con un monitor remoto de red puede enviar datos a una o más estaciones de administración y pueden ubicarse en hosts, routers y bridges.

Propiedades

El tener un monitor exclusivamente para propósitos de administración de red tiene sus ventajas:

- El monitor de red puede recolectar datos continuamente y enviarlos selectivamente a las estaciones de administración.
- Cuando se cruzan umbrales, se pueden generar eventos y enviar alarmas a las estaciones administradoras.
- Puede ayudar a reducir el tráfico de red, ya que los datos recopilados se almacenan en los agentes con monitores de red. Cuando la estación administradora necesita más detalles, puede acceder y recolectar la información desde los agentes con RMONs.

Escenarios para Monitoreo Remoto de Redes

RMON puede usarse en los siguientes escenarios:

- Operación Offline: se da cuando el monitor de red opera como un dispositivo de monitoreo standalone (único, independiente), ya sea por fallas de comunicación con la estación de administración, o para recolección selectiva de datos. El RMON puede operar independientemente de la estación administradora: puede recolectar y almacenar datos continuamente; la estación de administración puede recuperar los datos almacenados cada vez que lo requiera.

- Comunicación con estaciones de administración en base a excepciones: Cuando se observan problemas en la red, debido a que se cruzan umbrales o por fallas, se pueden enviar notificaciones a las estaciones de administración. El RMON puede reunir datos a cerca de la red y almacenarlos en un registro para analizarlos más tarde.

- Recopilación de datos relevantes: Un monitor puede concentrarse en hosts de una porción de red (críticos, que tienen mucho tráfico o errores) o en loguear los datos requeridos por la estación administradora.

- Múltiples estaciones de administración: Si las estaciones de administración se organizaran en base a las acciones que realizan, podría necesitarse que un monitor trate concurrentemente con más de una estación de administración para reportarse. La descomposición por funciones (contabilidad, configuración, performance, etc.) puede ayudar a centralizar en una estación administradora los datos requeridos para ciertas funciones de administración.

Otros

Grupos RMON

Para RMON se definieron nuevos grupos de objetos, bajo el subárbol {***mib-2*** *16*} de la jerarquía de registración.

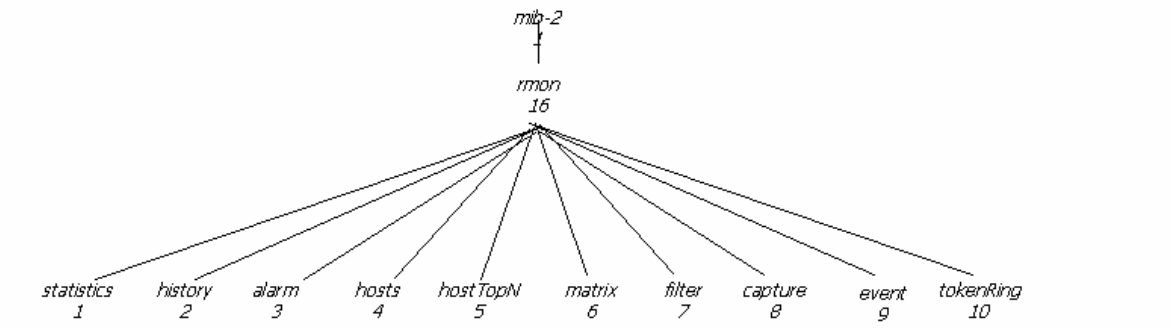


Figura. Objetos RMON

Cada grupo provee un conjunto específico de datos. Los grupos son opcionales, por lo tanto los vendedores no necesitan soportar todos los grupos que se incluyen en la MIB.
Algunos grupos de RMON precisan el soporte de otros grupos de RMON para funcionar adecuadamente.
La siguiente tabla resume los nueve grupos especificados en RFC 1157 Ethernet RMON MIB.

Grupo	**Función**
Estadísticas	Contiene estadísticas tomadas por el sensor de cada interfase monitoreada en este dispositivo.
Historia	Periódicamente toma muestras estadísticas de una red y las guarda para utilizarlas mas adelante.
Alarma	Cada cierto tiempo toma muestras estadísticas de las variables en el sensor y las compara con los niveles previamente configuradas. Si la variable monitoreada cruza el umbral, se genera el evento.

Host	Contiene estadísticas asociadas con cada anfitrión descubierto en la red
HostTopN	Prepara tablas que describen los host top de una lista ordenada por una de sus estadísticas, en cierto intervalo especificado por las estaciones administradoras. Son estadísticas rate-based.
Matriz	Almacena estadísticas de conversación entre conjuntos de dos direcciones. A medida que el dispositivo detecta una nueva conversación, crea un nuevo parámetro en su tabla.
Filtros	Permiten la comparación de los paquetes a una ecuación de filtro. Estos paquetes comparados forman una ráfaga de datos que se debe de capturar o generar nuevos eventos.
Captura de paquetes	Permite la captura de paquetes después de que han fluido a través de un canal.
Eventos	Controla la generación y notificación de eventos de este dispositivo.

MIB RMON

Los objetos definidos en la MIB RMON son principalmente para el protocolo Ethernet (puede ser expandida para cubrir los protocolos Token-Ring y FDDI).

La MIB de RMON soporta dos nuevas funciones, configuración e invocación de acciones.

- **Configuración**

Se determina el tipo, el formato y la cantidad de los datos que deben ser recolectados.

En los grupos de la MIB RMON hay una tabla de control y una o más tablas de datos.
La tabla de control es de lectura-escritura, contiene parámetros que describen los datos en la tabla. Las tablas de datos son de sólo lectura.

Fase de configuración: la estación gestora da instrucciones al monitor remoto para que recolecte los datos deseados, añadiendo una nueva fila a la tabla de control.
La información se recolecta y almacena en varias filas de las tablas de datos según la configuración descrita en una fila de la tabla de control.
Cada fila de la tabla de control tiene asociadas varias filas en una o varias tablas de datos.

- **Invocación de acciones**

SNMP no tiene mecanismos para ordenar a un agente que ejecute una determinada acción, sólo se pueden leer y escribir valores de objetos. Sin embargo es posible mediante operaciones SNMP enviar un mandato, empleando un objeto para representar un estado de modo que se realice una determinada acción cuando dicho objeto cambia de estado.

8.1.5.1.2 NetFlow

Descripción

Netflow es una tecnología para el monitoreo de tráfico de red desarrollada por Cisco Networks. Permite recolectar, analizar información de los flujos que pasan por switches y routers y luego exportarla a aplicaciones tanto para software de contabilidad, sistemas de facturación, aplicaciones de administración de redes.

Un flujo de red se define como una secuencia unidireccional de paquetes entre un origen y un destino de una terminal. Los flujos son altamente granulares y son identificados tanto por direcciones IP como por los números de puerto de aplicación de la capa de transporte.
Netflow también utiliza el tipo de protocolo IP, el tipo de servicio (ToS) y el identificador de interfase de entrada para identificar de forma única a los flujos.

Arquitectura

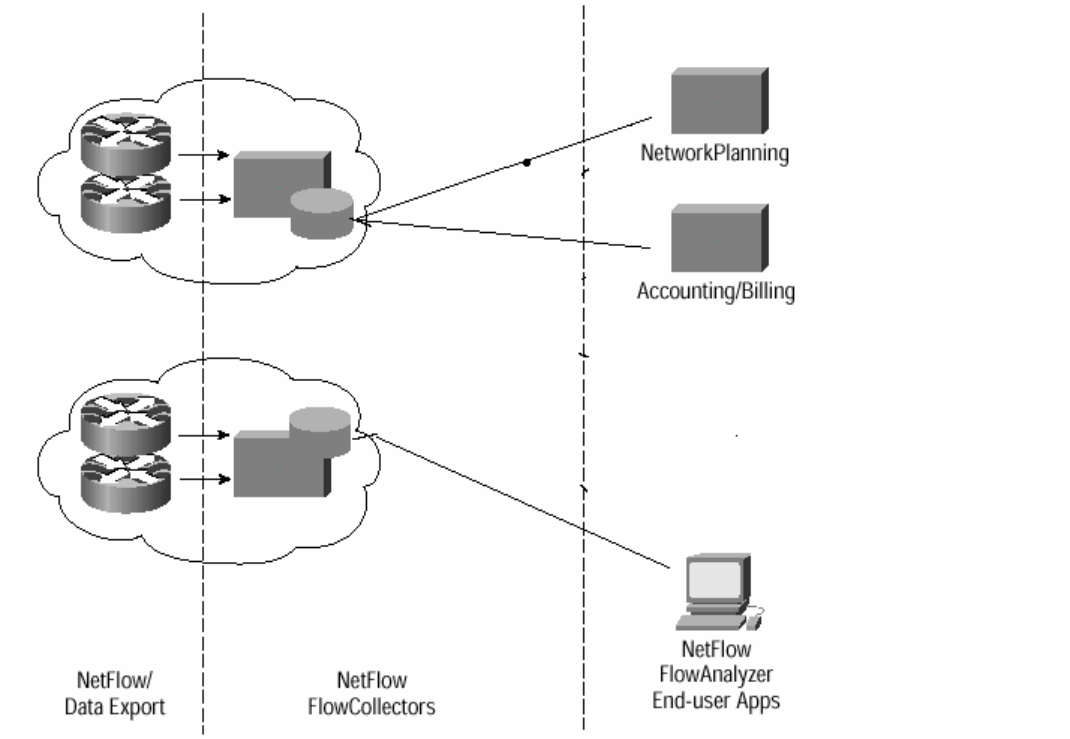
NetFlow tiene 3 componentes clave:

Flow catching: analiza y recolecta cada paquete IP de los flujos que entran por switches y routers y prepara datos para exportar (Netflow datagrams). Permite acumular datos de flujos con características únicas.

Flow Collector: captura la información exportada de múltiples routers, filtra y agrega valor a los datos según criterios y políticas del cliente, para luego almacenar esa información.

Network Data Analyzer: es una herramienta de análisis de tráfico de red, que combina una interfaz gráfica de usuario con otros módulos que pueden recuperar, mostrar y analizar los datos recolectados por el Flow Collector. Esta herramienta permite a los usuarios ver tendencias de los flujos o visualizarlos casi en tiempo real. Los datos se pueden visualizar con gráficas de barras, de torta o histogramas.
También se puede exportar los datos a aplicaciones externas como ser Microsoft Excel para poder generar reportes.

La figura siguiente permite visualizar la infraestructura de netflow.



Usos

- **Contabilidad y cobro (Accounting/Billing)** al proveer métricas finas por ejemplo direcciones IP, cantidad de paquetes y bytes, timestamps, tipo de servicios y puertos de aplicación, permite gran flexibilidad y detallado seguimiento del uso de recursos.

- **Planificación y análisis de la red** provee información clave que puede ser utilizada por herramientas para planificación y toma de decisiones estratégica, minimizando el costo total de las operaciones de la red y maximizando la performance, capacidad y confiabilidad.

- **Monitoreo de red** al dar información casi en tiempo real Netflow permite realizar monitoreo de red. Las técnicas de análisis basado en flujos pueden utilizarse para visualizar los patrones de tráfico asociados a routers o switches individuales, siendo útil para detectar fallas y actuar de forma más rápida para solucionarlas.

- **Monitoreo y Profiling de aplicaciones** provee a los administradores de red una vista detallada y basada en tiempos del uso de las aplicaciones en la red.

- **Monitor de usuario y profiling** provee a los administradores de red información de la utilización de los recursos de red y de aplicaciones de los usuarios o clientes. Esta información puede servir para planificar el uso de recursos de aplicación, como también detectar y resolver potenciales violaciones de seguridad.

- **NetFlow Data Warehousing and Mining** los datos exportados o la información derivada de Netflow puede ser analizada para soportar de forma proactiva programas para servicios de marketing

o clientes. Esto es especialmente útil para los Internet Service Providers (ISPs), ya que Netflow permite a éstos indagar en sus paquetes de servicio.

8.1.5.1.3 Sflow

Descripción

sFlow es una tecnología multi-vendedor, basada en el sampling de paquetes, embebida en switches y routers. Constantemente puede monitorear el flujo de tráfico de aplicaciones a gran velocidad, de forma simultánea en todas las interfases. Provee medidas de tráfico cuantitativas y detalladas, con velocidad en gigabit sin impactar en la performance de la red.
Switches y routers con tecnología sFlow están disponibles desde 2001.
La especificación de sFlow fue publicada como RFC 3176.

Arquitectura

Los elementos básicos de un sistema sFlow son el Agente y el Collector.

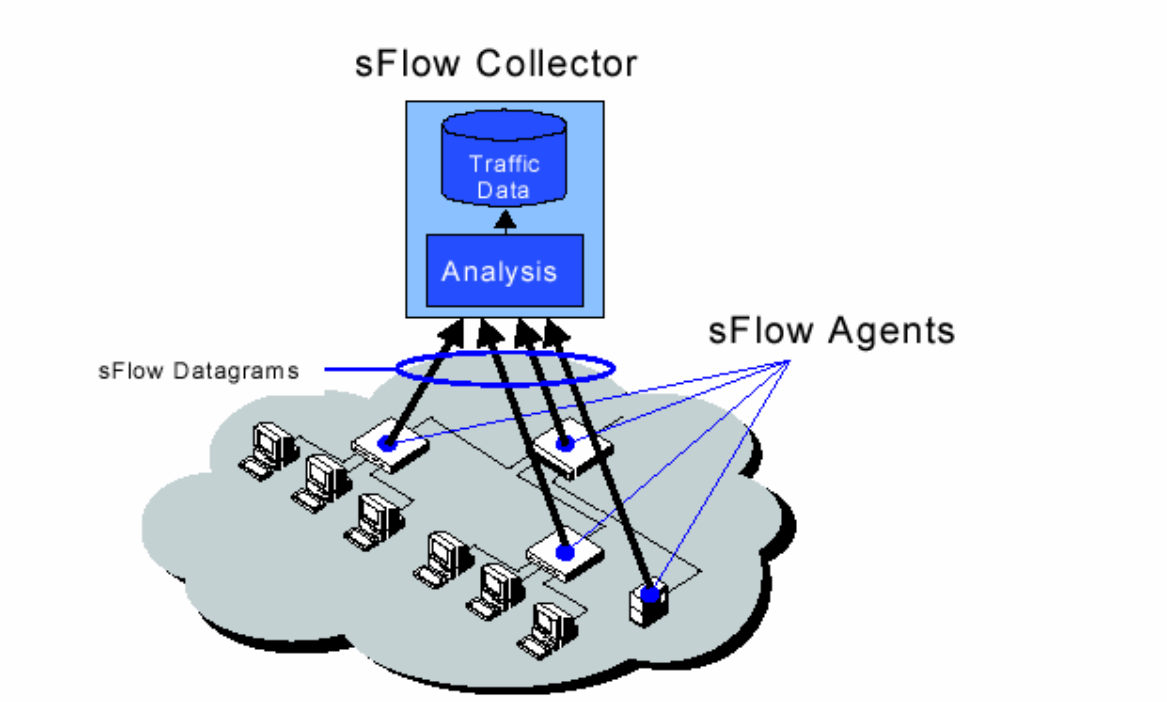


Figura. Agente y Colector sFlow.

El Agente sFlow es un proceso de software que corre como parte del software de administración de red dentro de un switcher o router. Combina contador de interfases y muestras de flujos en datagramas sFlow que son enviados a través de la red hacia el Colector sFlow.
El muestreo de paquetes es típicamente realizado por switcheo/ruteo ASICs, proveyendo gran performance.

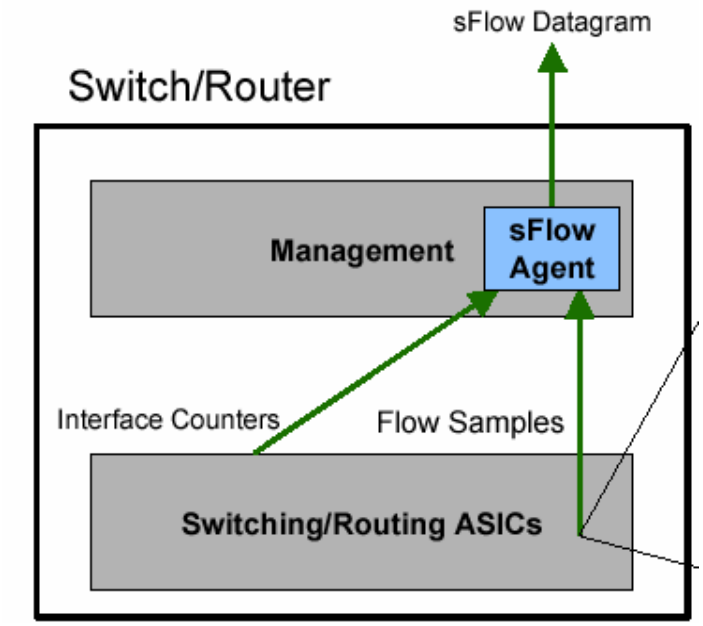


Figura. Agente sFlow embebido en switch/router.

El Agente sFlow realiza muy poco procesamiento, simplemente empaqueta datos en los Datagramas sFlow que son enviados inmediatamente a la red. Este envío inmediato minimiza los requerimientos de memoria y CPU del Agente.

El colector se encarga de recolectar y analizar los datos, pudiendo generar información detallada del tráfico en tiempo real.

Propiedades

- sFlow provee una amplia vista del uso de la red y de los routers. Permite medir el tráfico de red, recolectar, guardar y analizar los datos del tráfico.
- **Preciso** El muestreo es simple como para ser realizado en hardware. Además, el sistema sFlow está diseñado de manera que puede ser determinada con exactitud distintos tipos de medidas.
- **Detallado** El contar con información completa del header de los paquetes y del switching y routing permite un detallado análisis del flujo de tráfico de las capas L2-L7.
- **Escalable** En tamaño y velocidad.
  Es capaz de monitorear redes a 10Gbps, 100Gbps y más, sin sobrecargar la red. Cientos de dispositivos pueden ser monitoreados por un único Colector sFlow.
- **Bajo costo** El Agente sFlow es simple de implementar y agrega bajo costo a un switch o router. Ha sido implementado en un amplio rango de dispositivos, sin requerir memoria y CPU extra.
- **Oportuno** El Colector sFlow tiene una vista del tráfico de la red de los últimos minutos. Esta propiedad es particularmente importante si fuera necesario proveer controles en tiempo real.

Usos

sFlow da una amplia visibilidad del uso de la red al monitorear los flujos de tráfico en todos los puertos. Algunos de sus usos son:

- Soluciona problemas de tráfico (Troubleshooting Network Problems)
  Cualquier uso de la red genera tráfico. Consecuentemente, los problemas son observados en patrones anormales de tráfico. sFlow hace visibles estos patrones anormales, con el detalle como para rápidamente identificar, diagnosticar, y corregir.

- Control de congestionamiento (Controlling Congestion)
  Al monitorear el flujo de tráfico en todos los puertos, se pueden ver los links de congestionamiento, identificar el origen del tráfico, y las conversaciones entre aplicaciones.

- Seguridad y Audit Trail Analysis
  Permite de forma rápida, controlar y hacer un seguimiento de ataques y amenazas internas y externas. Para llevar la historia detallada del tráfico se crea una baseline del comportamiento normal, a partir del cual anomalías o actividades sospechosas se pueden identificar. De esta forma se puede prevenir ataques intencionales, minimizar errores no intencionales, y proteger la información.

- Perfil de ruteo
  Como sFlow contiene información anticipada, puede usarla para perfilar las rutas más activas y los flujos que corren por esas rutas. Al comprender las rutas y flujos permite optimizar la rutas mejorando la conectividad y performance.

- Contabilidad y cobro del uso (Accounting and Billing for Usage)
  Provee información detallada del uso de los servicios de red, y así poder contabilizar y facturar el uso por cliente. También puede ser usado para proveer a los clientes una lista de su tráfico total, indicar los usuarios y aplicaciones más frecuentes.

8.1.5.2 Comparación de RMON, NetFlow and sFlow en ciertas áreas relevantes:

8.1.5.2.1 Recursos del Agente

Cuando se comparan tecnologías embebidas en aplicaciones de monitoreo, se deben considerar tres recursos principales que la aplicación consumirá: CPU, memoria y ancho de banda.
Estos recursos son caros, por lo tanto, la solución debe minimizar el uso, y así hacer una solución menos costosa.

CPU

Los requerimientos computacionales en el monitoreo del tráfico impactan significativamente en los costos del agente y escalabilidad. Las técnicas de monitoreo requieren más alta performance en el manejo de CPU para switches y routers. Además un solo procesador quizás no soporte la demanda de monitorear un gran número de puertos contenidos en muchos switches y por ende más CPU es requerida para monitorear.

Los agentes RMON y NetFlow construyen matrices de tráfico decodificando cada paquete.
Esto puede agregar una carga impredecible de CPU, dependiendo de la naturaleza de los patrones de tráfico. En el caso de RMON la CPU además tiene que establecer un diálogo con la estación administradora para cargar los resultados del último período. Ordenar la información en un formato ASN.1 para SNMP implica más sobrecarga.
En contraste, la función de muestreo del agente sFlow puede ser fácilmente implementada en hardware. Además como el agente de lo único que se encarga es de enviar el cabezal del paquete al servidor, esa tarea no implica una sobrecarga del CPU.

Memoria

La memoria requerida para construir las medidas de tráfico afectan al costo del agente.
Los agentes RMON y NetFlow construyen matrices en la RAM del agente. El tamaño de las tablas depende mucho de los patrones de tráfico.
NetFlow tiene la ventaja de que puede ser configurado automáticamente para descargar flujos individuales cada quince minutos o más, y también para descargar bloques de flujos para prevenir quedarse sin memoria suficiente.
En cambio el agente RMON debe mantener toda la matriz de tráfico en memoria y crea una nueva para el período siguiente mientras espera que el servidor tome los resultados.

El agente sFlow solo necesita RAM para guardar un paquete. Cuando una muestra es tomada es enviada inmediatamente al servidor.

Ancho de banda

La transferencia de datos desde los agentes de monitoreo al colector central consume considerable ancho de banda. Este ancho de banda es quitado a las aplicaciones de la red.
Hay que considerar también que picos altos de transferencia pueden causar congestionamiento en la red.

La carga de red de un agente Netflow se muestra como streams de largos paquetes UDP. Una vez que la tabla de flujos se llena, un gran número de flujos pueden ser descargados para liberar espacio.

Los agentes RMON recolectan toda la matriz de tráfico en memoria. Luego al final del período de recolección hay un disparo de la actividad del flujo de red debido a que el servidor solicita los datos del período anterior a todos los agentes de la red, lo más pronto posible.
Esto limita la cantidad de agentes que pueden ser administrados por un solo servidor.

Todos los agentes sflow realizan un envío sostenido de paquetes hacia el servidor. Cada agente puede controlar el nivel de tráfico ajustando la frecuencia de muestreo.

8.1.5.2.2 Recursos del servidor

El agente de monitoreo es solo una parte de la solución de administración. Un servidor central se precisa para configurar los agentes, bajar y archivar los datos, mostrar los datos en una interfase de usuario.
Un factor clave al determinar el costo y escalabilidad de un sistema de monitoreo es determinar los recursos del servidor para gestionar un solo agente.
El consumo de recursos en el servidor determina la cantidad de agentes que un solo servidor puede administrar.

Los servidores de RMON y NetFlow tienen el problema del gran tamaño de las matrices de tráfico de los agentes. Aparte del esfuerzo de recibir la información desde la red, también tienen que consolidar los datos en matrices de tráfico por sitio sin mezclar la información de los distintos agentes.

El tamaño de las matrices construidas por el muestreo de paquetes es mucho menor, consiste típicamente de los flujos más ocupados, más una selección de los flujos más pequeños. Por lo tanto la tarea de consolidar las matrices de tráfico es mucho más liviano.

8.1.5.2.3 Conclusiones

RMON define esencialmente un analizador de protocolo remoto, con la habilidad de filtrar, capturar y descifrar paquetes, lo que lo hace aplicable para resolver problemas que solo pueden ser resueltos viendo las secuencias paquetes.

La tecnología NetFlow puede ser útil en enlaces WAN, donde es esencial ver cada flujo, y quizás para monitoreo de seguridad o un conteo específico flujo a flujo.

La técnica de muestreo estadístico utilizado por sFlow permite administrar un gran número de agentes, muchos puertos de switches por un solo servidor. El bajo costo de los agentes y las ventajas técnicas lo hacen ideal para el monitoreo y reporte continuo de un gran red.

8.1.5.3 conclusiones

Trataremos de respondernos la siguiente pregunta, ¿Por qué hacer algo nuevo con tantas soluciones existentes?

En las empresas es necesario el monitoreo de una gran variedad de servicios y componentes, tanto de software como hardware. Por ejemplo, respaldos de bases de datos, consumo de CPU, carga en la red, consumo de recursos de las aplicaciones, desempeño y estado de aplicaciones. Además es importante dar una respuesta rápida a cualquier problema que pueda surgir, asegurando más confiabilidad de los servicios y su disponibilidad cuando sea necesario.
También es importante poder alertar lo más rápido posibles de problemas reales o potenciales (notificación y alerta), realizar el seguimiento de sucesos por medio de registro de información.

De las soluciones que encontramos vimos que de las que se presentan como una solución muy completa que abarcan muchas de las características que buscamos concretar en nuestra herramienta, son muy costosas y propietarias, por lo tanto difíciles de integrar en sistemas heterogéneos.
Otras que son más baratas son muy específicas del monitoreo en redes, y no abarcan la totalidad de los problemas que nos interesa resolver.
Por las características de estas herramientas, es difícil pensar en integrar más de una de estas opciones para lograr una solución integrada, para administrar de forma centralizada y a su vez con una interfaz de usuario simple y amigable.

Por estos motivos tiene sentido pensar en desarrollar una herramienta de monitoreo que reúna distintas funcionalidades, entre estas, que sea configurable, fácil de mantener y extender, multiplataforma, que permita configuración de alarmas y registro, monitorear componentes de software.

8.1.6 MRTG – instalación, configuración, puesta en marcha

Información para instalar, configurar y correr MRTG se puede encontrar en el sitio web *http://people.ee.ethz.ch/~oetiker/webtools/mrtg/*, por esto recomendamos visitarlo. Allí se consiguen links para bajar los componentes y también guías claras de instalación tanto para unix como para Windows NT.

Igualmente consideramos útil describir nuestra experiencia al instalar, configurar y utilizar la herramienta. Antes que nada es importante recordar que tanto para instalar o configurar archivos se precisa tener permisos de Super usuario en Unix y de Administrador en Windows.

Instalación

Todas las pruebas las realizamos en la empresa K&S IT.
La idea inicial fue instalar MRTG en una máquina en unix para monitorear los servidores de bases de datos (WinNt) y el servidor de Internet (Unix).
Para poder monitorear esos servidores precisábamos que en cada uno corriera el servicio SNMP y configurarlo para que pudieran responder solicitudes a la máquina donde corriera el MRTG.

1. Instalación SNMP Windows

En
http://resource.intel.com/telecom/support/releases/winnt/SR511/docs/htmlfiles/userguide/getting4.htm_-1001777 se detalla como instalar y configurar el servicio de SNMP para Windows 2000 y también un referencia para Windows nt.
Seguimos los pasos que en la página indican.

Luego de instalado, tuvimos ciertos problemas con la versión de Windows Service Pack que tenía el servidor, pero por razones de seguridad de K&S no instalamos una actualización.

2. Instalación SNMP Unix

El servidor de Internet tenía una versión más vieja de SUSE (SUSE 6.4) en la cual tampoco estaba instalado el servicio SNMP. Como la idea no era actualizar el SUSE lo que hicimos fue bajar el paquete **ucdsnmp.rpm** *de ftp://ftp.sunet.se/pub/Linux/distributions/suse/suse/i386/6.4/suse/n1 (Fecha último acceso:30/07/2003).*
Para instalar, corrimos via telnet al servidor de Internet el comando*: rpm -i ucdsnmp.rpm*.
Para levantar el servicio: *rcsnmp restart*.
Para que la máquina donde instalamos MRTG pueda realizar consultas hubo que modificar el archivo snmp.cfg. Agregamos la línea *rocommunity public 192.168.13.1.*

Las máquina destinada a correr MRTG tenían instalado SUSE 8.2 el cual trae una versión de MRTG, por lo que no fue necesario hacer ninguna instalación, solo realizamos trabajo de configuración.
En *http://people.ee.ethz.ch/~oetiker/webtools/mrtg/unix-guide.html* se puede encontrar información detallada de cómo instalar.

Configuración

En el sito *http://people.ee.ethz.ch/~oetiker/webtools/mrtg/reference.html* se describe en detalle como crear el archivo de configuración.

Nosotros ejecutamos:

cfgmaker --global WorkDir: net/homes/carolina/public_html/
--subdirs= HOSTNAME
--output mrtg.cfg
public@192.168.13.3

El contenido del archivo que se creó es :

```
# Created by
# /usr/bin/cfgmaker --global WorkDir:/net/homes/carolina/public_html --subdirs=HOSTNAME --output
mrtg.cfg public@192.168.13.3


### Global Config Options

#  for UNIX
# WorkDir: /home/http/mrtg

#  or for NT
# WorkDir: c:\mrtgdata

### Global Defaults

#  to get bits instead of bytes and graphs growing to the right
# Options[_]: growright, bits

WorkDir:/net/homes/carolina/public_html

######################################################################
# System: proxy
# Description: Linux proxy 2.2.14 #1 Sat Mar 25 00:45:35 GMT 2000 i586
# Contact: root@localhost
# Location: unknown
######################################################################

### Interface 1 >> Descr: 'lo0' | Name: '' | Ip: '127.0.0.1' | Eth: '' ###
### The following interface is commented out because:
### * it is a Software Loopback interface
#
Target[192.168.13.3_1]: 1:public@192.168.13.3:
SetEnv[192.168.13.3_1]: MRTG_INT_IP="127.0.0.1" MRTG_INT_DESCR="lo0"
Directory[192.168.13.3_1]: 192.168.13.3
MaxBytes[192.168.13.3_1]: 1250000
Title[192.168.13.3_1]: Traffic Analysis for 1 -- proxy
PageTop[192.168.13.3_1]: <H1>Traffic Analysis for 1 -- proxy</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>proxy in unknown</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>root@localhost</TD></TR>
   <TR><TD>Description:</TD><TD>lo0  </TD></TR>
   <TR><TD>ifType:</TD>     <TD>softwareLoopback (24)</TD></TR>
   <TR><TD>ifName:</TD>     <TD></TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>1250.0 kBytes/s</TD></TR>
   <TR><TD>Ip:</TD>         <TD>127.0.0.1 (localhost)</TD></TR>
 </TABLE>


### Interface 2 >> Descr: 'eth0' | Name: '' | Ip: '192.168.13.3' | Eth: '00-a0-24-d7-7c-e4' ###

Target[192.168.13.3_2]: 2:public@192.168.13.3:
SetEnv[192.168.13.3_2]: MRTG_INT_IP="192.168.13.3" MRTG_INT_DESCR="eth0"
Directory[192.168.13.3_2]: 192.168.13.3
MaxBytes[192.168.13.3_2]: 1250000
Title[192.168.13.3_2]: Traffic Analysis for 2 -- proxy
PageTop[192.168.13.3_2]: <H1>Traffic Analysis for 2 -- proxy</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>proxy in unknown</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>root@localhost</TD></TR>
   <TR><TD>Description:</TD><TD>eth0  </TD></TR>
   <TR><TD>ifType:</TD>     <TD>ethernetCsmacd (6)</TD></TR>
   <TR><TD>ifName:</TD>     <TD></TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>1250.0 kBytes/s</TD></TR>
   <TR><TD>Ip:</TD>         <TD>192.168.13.3 (proxy.kslinux)</TD></TR>
 </TABLE>
```

```
### Interface 3 >> Descr: 'eth1' | Name: '' | Ip: '192.168.0.99' | Eth: '00-40-33-30-07-82' ###

Target[192.168.13.3_3]: 3:public@192.168.13.3:
SetEnv[192.168.13.3_3]: MRTG_INT_IP="192.168.0.99" MRTG_INT_DESCR="eth1"
Directory[192.168.13.3_3]: 192.168.13.3
MaxBytes[192.168.13.3_3]: 1250000
Title[192.168.13.3_3]: Traffic Analysis for 3 -- proxy
PageTop[192.168.13.3_3]: <H1>Traffic Analysis for 3 -- proxy</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>proxy in unknown</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>root@localhost</TD></TR>
   <TR><TD>Description:</TD><TD>eth1  </TD></TR>
   <TR><TD>ifType:</TD>     <TD>ethernetCsmacd (6)</TD></TR>
   <TR><TD>ifName:</TD>     <TD></TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>1250.0 kBytes/s</TD></TR>
   <TR><TD>Ip:</TD>         <TD>192.168.0.99 ()</TD></TR>
 </TABLE>


### Interface 4 >> Descr: 'ppp0' | Name: '' | Ip: '200.40.165.149' | Eth: '' ###
### The following interface is commented out because:
### * has a speed of 0 which makes no sense
#
Target[192.168.13.3_4]: 4:public@192.168.13.3:
SetEnv[192.168.13.3_4]: MRTG_INT_IP="200.40.165.149" MRTG_INT_DESCR="ppp0"
Directory[192.168.13.3_4]: 192.168.13.3
MaxBytes[192.168.13.3_4]: 32000
Title[192.168.13.3_4]: Traffic Analysis for 4 -- proxy
PageTop[192.168.13.3_4]: <H1>Traffic Analysis for 4 -- proxy</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>proxy in unknown</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>root@localhost</TD></TR>
   <TR><TD>Description:</TD><TD>ppp0  </TD></TR>
   <TR><TD>ifType:</TD>     <TD>ppp (23)</TD></TR>
   <TR><TD>ifName:</TD>     <TD></TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>32000 Bytes/s</TD></TR>
   <TR><TD>Ip:</TD>         <TD>200.40.165.149 (r200-40-165-149.adsl.anteldata.net.uy)</TD></TR>
 </TABLE>
```

Corrida

Es recomendable correr MRTG de forma manual previo a correrlo de forma automática.
Para correrlo manualmente simplemente hay que ejecutar: mrtg mrtg.cfg.
En las primeras corridas pueden aparece algunos mensajes sobre archivos de log perdido, esto se debe a que se intentan abrir estos archivos suponiendo que ya existen.

Si en cambio siguen los mensajes después de unas cuantas corridas habrá que investigar más a fondo el origen del problema.

Luego de que consideramos que el archivo estaba configurado de acuerdo a lo que queríamos, automatizamos el proceso para que corra en intervalos regulares de 5 minutos. Para esto modificamos el cronntab de la siguiente manera:

crontab -e 0,5,10,15,20,25,30,35,40,45,50,55 * * * * /usr/bin/mrtg /root/mrtg.cfg

8.2 CASOS DE USO RELEVADOS

Se detallan un conjunto de casos de uso relevados, estos se tomarán en su conjunto como guía para el diseño y arquitectura de la herramienta.
Estos buscan determinar qué elementos estarán sujetos a acciones de monitoreo, las distintas propiedades que se quieren medir de cada elemento y las acciones que se tomarán sobre cada caso. Es deseable agrupar casos similares que permitan determinar subconjuntos de situaciones similares con el objetivo de definir como requerimientos mínimos un subconjunto representativo de esta tabla.

Como resumen de este agrupamiento se implementaron algunos casos que se entienden como más representativos o que son más interesantes en la realidad de la empresa.

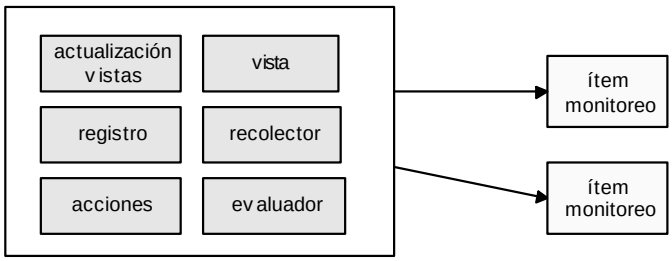
Caso	Entidad de monitoreo	Propiedades a medir	Registro	Acciones a realizar
1	Impresora compartida	Controlar si la impresora está disponible En caso de estar disponible mostrar la cantidad de trabajos encolados para imprimir		En caso de no estar disponible notificar por ejemplo vía email a quien corresponda según el motivo de la no disponibilidad
2	Servidor de internet	Cada cierto período de tiempo el flujo de entrada y salida del servidor de internet, quizás discriminando por puerto	Registrar medidas tomadas, quizás no llevar un histórico sino ir calculando promedios	Tanto para el flujo de entrada o salida, si pasa cierto umbral establecido, notificar a quien corresponda
3	Bases de datos	Monitorear: ▪ Resultados de jobs schedulados ▪ Estados de backups realizados ▪ Tamaño real y el establecido de la BD ▪ Espacio total libre en disco donde se ubican las BD		Para los primeros dos puntos en caso de encontrar un error notificar. Si un backup, o job no se terminó bien disparar nuevamente el proceso. Para los últimos dos puntos, establecer umbrales y si se superan notificar vía una petición de Middleware a quien corresponda.
4	Componentes de hardware que cuentan con Agente SNMP	Controlar el estados del dispositivo de Cintero de backups Controlar una UPS, router, hub o switch.		En caso de haber una falla notificar En caso de haber una falla realizar una llamada telefónica a quien corresponda Dar la posibilidad de prender o apagar el componente
5	Componentes de hardware que no cuentan con Agente SNMP	Monitorear ventilador de una máquina.		En caso de haber una falla notificar.
6	Actividad de Logging de usuarios	Monitorear actividad de logging de usuarios.	Registrar información relevante en un archivo, en caso de por ejemplo encontrar más de tres intentos de ingreso fallidos consecutivos a un sistema, o algún otro tipo de actividad sospechosa	
7	Puertos	Monitorer el uso de un puerto crítico.		En caso de detectar un uso indebido del mismo, notificar.
8	Estado de una máquina	Monitorear: ▪ Carga de CPU ▪ Memoria ▪ Espacio libre en disco		Fijar umbrales para cada una de las medidas y si los sobrepasan notificar.
9	Procesos	Monitorear: ▪ consumo de CPU y memoria de	Registrar medidas si sobrepasa cierto	Si sobrepasa cierto umbral por un período determinado de tiempo

		un proceso dado. ▪ Estado de un proceso	umbral.	notificar . En caso del proceso no estar activo reactivarlo.
10	Análisis de orígenes de datos como ser archivos , BD.	Analizar algún archivo del sistema , por ejemplo eventviewer de windows nt, buscando ciertos patrones.	En caso de encontrar algún patrón, registrar en un archivo en una carpeta según el día de la semana. La idea sería guardar una carpeta para cada día de la semana, e ir sobrescribiéndolas cada semana.	
11	"Responsiveness"	Monitorear el tiempo de respuesta de alguna actividad x, si se cuenta con alguna operación que se ejecute cada cierto tiempo(Hacer ping, echo request, Html, smtp).	Calcular promedios con medidas anteriores y registrar.	Si ping falla o no hay echo notificar
12	Procesos de una máquina	Monitorear los proceso que corren en una máquina.	Registrar cuando un proceso consume recursos en un cierto umbral	Matar o levantar un proceso
13	Monitoreo de transacciones distribuidas (a nivel de negocio)	Poder realizar el seguimiento de transacciones que pasan por distintos sistemas, con distintos estados		En caso de encontrar alguna transacción en algún estado no consistente (habría que establecer las condiciones de inconsistencia) notificar a quien corresponda vía email.
14	Consistencia de fuentes de datos en sistemas distribuidos	Establecer reglas de consistencia y chequear que se cumplan.		En caso de encontrar inconsistencias notificar.
15	Sitios web	Monitorear sitios web que visitan los usuarios de una empresa.	Registrar en una BD cada vez que aparezca un sitio nuevo.	En caso de que en cierto período de tiempo aparecieran más de cierta cantidad de sitios nuevos, notificar.
16	Inspección de Mails	Monitorear envío de mail. Quién envíal y a quién.	Si un usuario excede cierta cantidad establecida de envío de mails a alguna cuenta no conocidas por la empresa, registrar.	
17	Control de Llamadas telefónicas	Controlar las llamadas telefónicas en una empresa, ya sea de interno a interno, como llamadas hacia afuera de la empresa, y medir su duración.	Tener un log histórico de las llamadas que se realizan.	

8.3 DISCUSIÓN DE POSIBLES ARQUITECTURAS

Los componentes a los que se hace referencia son los definidos en el apartado, “Reconocimiento de componentes” del capítulo “Especificación de diseño y Arquitectura”.

8.3.1 Centralizada (Opción 1)



Descripción
La opción 1 consiste de una arquitectura centralizada. Tanto el componente recolector de medidas como el resto de los componentes donde está la inteligencia del sistema (evaluador, acciones, registro, actualización de vistas) se encuentran en la misma estación de monitoreo.
El scheduler se encarga de disparar las tareas de monitoreo de forma secuencial cada cierto período de tiempo fijado.
Cuenta con una única vista de todo el conjunto de items monitoreados, que se encuentra también en la misma estación de monitoreo.

Principales ventajas
Es una arquitectura simple al estar centralizada la inteligencia, por lo cual sería más simple su implementación.

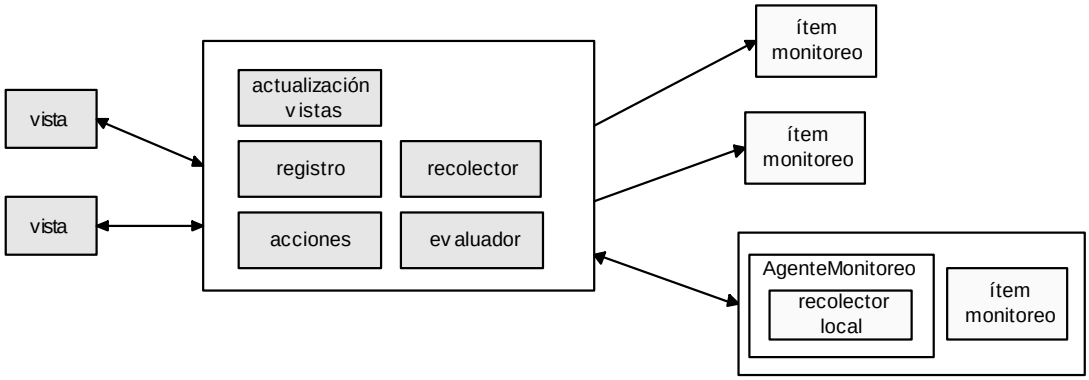
Principales desventajas
La cantidad de items a monitorear se limita a los recursos que disponga la máquina donde se encuentra el monitor, ya que toda la actividad se concentra allí.

Tener una única vista de los items monitoreados, y que se encuentre en la misma máquina de monitoreo, podría limitar las posibilidades de uso.

La carga en la estación de monitoreo podría llegar a ser alta ya que en una única máquina se concentra toda la actividad.

Si ocurre algún daño en la única estación de trabajo, se detiene la actividad de monitoreo.

8.3.2 Centralizada con agentes de monitoreo externo (Opción 2)



Descripción
En la opción 2 los componentes que dan inteligencia al monitor se encuentran en la misma estación de monitoreo. También en este esquema existe un recolector central de medidas, pero a diferencia de la opción 1 este monitor acepta la recepción de traps SNMP y otros eventos por lo cual la actividad de recolección no es efectuada solo por el recolector central. Para el manejo de eventos habrían componentes "Agente de monitoreo" que se encargarían de monitorear items de forma local con un recolector local, que enviaría eventos al monitor para notificar las medidas.

A diferencia de la opción 1 cuenta con distintas vistas de los items monitoreados que a su vez pueden correr en otras máquinas.

En este esquema habría un scheduler por cada item a monitorear. Cada scheduler sería independiente del resto. Habría si un componente que se encargaría de dar comienzo a los distintos schedulers y de asegurarse de que estén activos durante la vida del monitor.

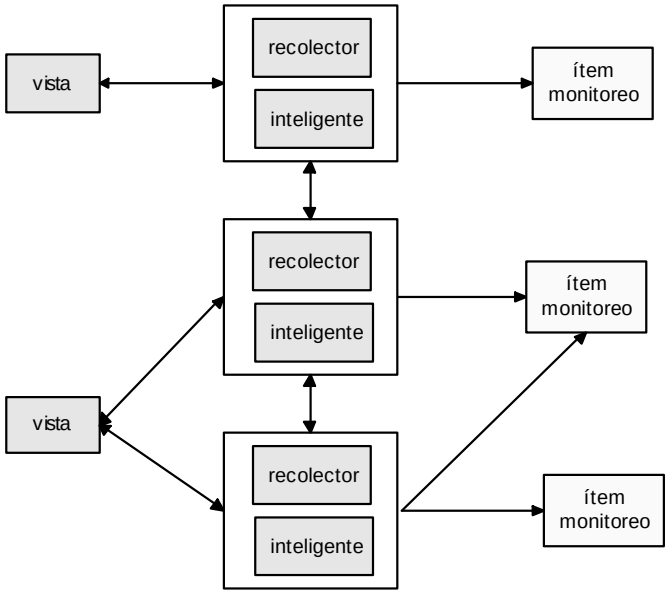
Principales ventajas
Es una arquitectura simple al estar centralizada la inteligencia.

La posibilidad de recepción de eventos y traps SNMP, permite liberar parte de la carga de estación de monitoreo, ya que parte de la actividad de recolección de medidas se delega a los Agentes de Monitoreo que se serían locales al item a monitorear.

Contar con distintas vistas independientes en otras estaciones que no son la de monitoreo daría más posibilidades de uso a la herramienta.

Principales desventajas
Si ocurre algún daño en la única estación de trabajo, se detiene la actividad de monitoreo.
A pesar de delegar un poco de la actividad de recolección a los "Agentes de Monitoreo", todo el resto de las actividades se concentran en una única estación de monitoreo lo cual puede ser una limitante de la cantidad de items a monitorear.

8.3.3 Distribuida (Opción 3)



Descripción:
La opción 3 es una arquitectura distribuida.
La inteligencia y recolección se distribuye en distintas estaciones de monitoreo.
Cada estación se encargaría de monitorear un subconjunto del total de items a monitorear, evaluar condiciones de alarma, disparar las acciones especificadas, registrar, y refrescar las vistas.
En caso de que alguna de las estaciones quede inactiva, el resto de las estaciones de alguna manera sustituiría a la estación inactiva.

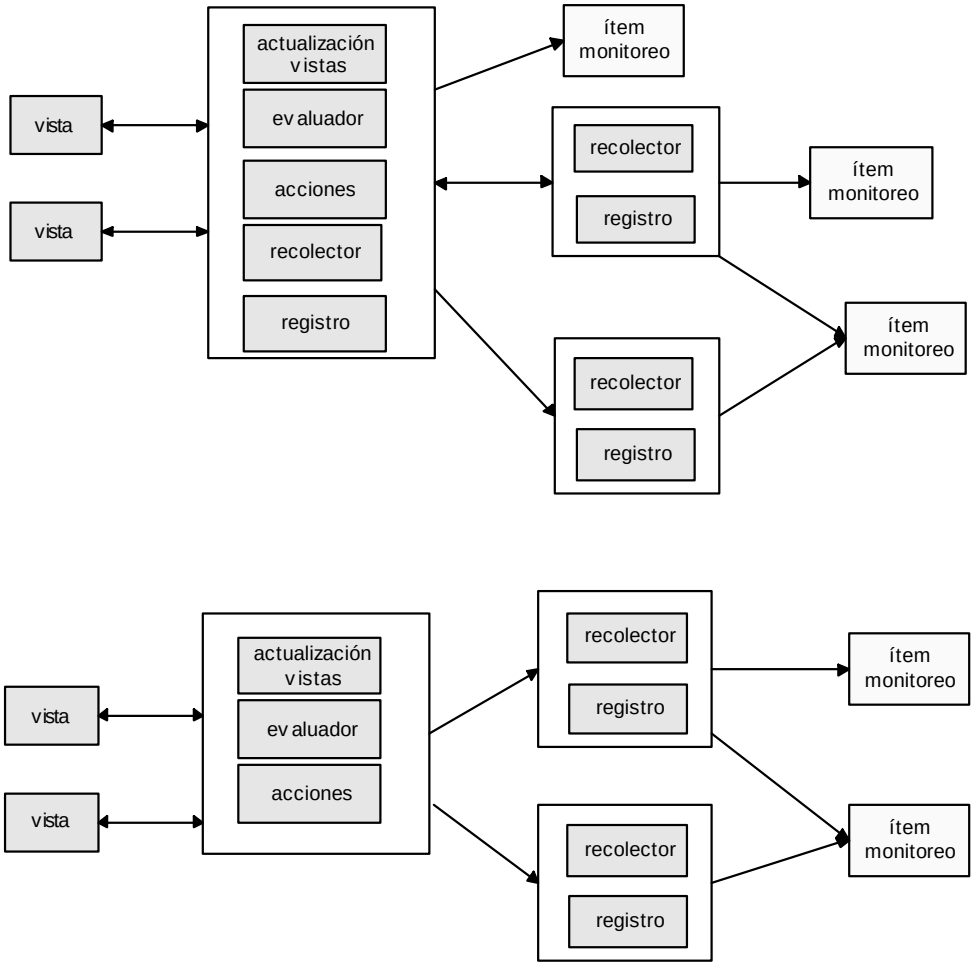
Principales ventajas:
A diferencia de los casos anteriores no hay dependencia de una única estación de monitoreo, si una falla otras la podrían reemplazar. Además no hay una limitante de los items a monitorear por los recursos de la estación de monitoreo, ya que si se quisieran agregar más items y no dieran los recursos, se podrían agregar nuevas estaciones.

Tener distintas vistas de los items monitoreados.

Principales desventajas:
Es una arquitectura más compleja en la cual se tienen que resolver temas que no se presentaban en los casos anteriores, como ser: establecer mecanismos de comunicación entre las distintas estaciones, sincronizar la actualización de las vistas, determinar cómo se va a efectuar la inicialización del sistema, determinar si el registro de actividades va a ser local o centralizado, mantener la configuración independiente de las estaciones que se utilicen, etc.
Esta arquitectura más compleja implica una implementación más compleja, la cual es más propicia a que ocurran más errores que en una arquitectura más simple.

La comunicación entre las estaciones de monitoreo puede generar en la red carga no deseada.

8.3.4 Distribuida (Opción 4)



Descripción
La opción 4 es una arquitectura centralizada en inteligencia, pero distribuida en cuanto a la recolección y registro de las medidas. Presentamos dos esquemas, el primero en donde hay varias estaciones de recolección y registro y en una de éstas además está la inteligencia (se encarga de evaluar condiciones, disparar acciones, actualizar las vistas, inicializar los recolectores), y el segundo esquema donde habría una estación exclusiva donde estaría la inteligencia, y otras estaciones exclusivas para recolección y registro.

En esta arquitectura también se aceparían traps SNMP y eventos como en la opción 2, y la política de scheduler sería la misma.

Principales ventajas
Al no concentrar toda la actividad en una sola estación de monitoreo, permitiendo acoplar nuevas estaciones de recolección y registro en caso de que se requieran y contar también con los "Agentes de Monitoreo", hace que la arquitectura no sea una limitante de la cantidad de items a monitorear, a diferencia a las opciones 1 y 2 en que si lo era.
Tener distintas vistas de los items monitoreados.

Principales desventajas
Si ocurre algún daño en la estación de monitoreo donde está la inteligencia, se detiene la actividad de monitoreo.

La comunicación entre las estaciones de monitoreo puede agregar carga en la red.

8.3.5 CONCLUSIONES

Las opciones 1 y 2 tienen como principal ventaja que su arquitectura es simple, pero la limitante mayor es que no es una arquitectura escalable. La cantidad de items a monitorear estaría limitado por los recursos de la estación de monitoreo, y si ésta sufre algún daño la actividad de monitoreo se detiene.

La opción 3 en contraste, soluciona los problemas de las opciones 1 y 2, pero a costa de tener una arquitectura muy compleja.

Se considera que las opciones 1 y 2 tienen muchas limitantes y no permitirían implementar el monitor con las características deseadas detalladas en el capítulo “Análisis de Requerimientos”, la opción 3 se adecuaría, pero luego de discutida su implementación con los tutores se decidió no aplicarla en este proyecto debido a su complejidad y al tiempo disponible.

En conclusión, se consideró que la opción 4 sería la más adecuada. Es una arquitectura distribuida en cuanto a la recolección de medidas lo que permite que sea escalable, pero centralizada en inteligencia lo cual la hace de menor complejidad que la opción 3. La principal desventaja de esta arquitectura es que si ocurre algún daño en la estación de monitoreo donde está la inteligencia se detiene la actividad de monitoreo. Otra limitante podría ser los recursos de la estación de monitoreo donde se concentra la inteligencia, pero como ésta no realiza todas las actividades, no sería en principio un cuello de botella.

8.4 APLICACIÓN PARA PRUEBA DE MX4J

8.4.1 Descripción

Esta aplicación consiste de tres componentes: un cliente, un servidor al que accede el cliente, y un agente.

En el agente se monitorean dos recursos. Uno es un archivo del cual interesa controlar cada 10 segundos su existencia y cada 15 segundos que el tamaño del archivo no supere cierto umbral especificado. El otro es el espacio libre en disco de una máquina de la red, la cual se monitorea con SNMP. En caso de que alguna de las condiciones de alarma se cumpla se ejecuta una acción local al agente, en el ejemplo será un mensaje que se despliega en la salida estándar.
El agente se compone principalmente de un MBeanServer en el cual se registran los MBeans a monitorear, y los monitores y las acciones también se registran como MBeans.

A los atributos de los MBean de recursos se les asocia un monitor. Creamos dos MBeans, FileMBean para representar al recurso archivo y SNMPMBean para representar un OID SNMP.

El FileMBean tiene dos atributos: Status que puede tomar los valores TRUE y FALSE, y length que retorna el tamaño del archivo.

Al atributo status se le asoció un StringMontor el cual controlará la existencia del archivo.
Al atributo length se le asoció un CounterMonitor, y se estableció un tope, si supera ese tope se enviará una notificación.

El SNMPMBean tiene un atributo value que retorna el valor para el OID SNMP especificado.
Este atributo tiene asociado un StringMonitor. Para el manejo SNMP utilizamos una implementación open source del protocolo SNMP en un paquete java. Este paquete provee soporte para las operaciones básicas del cliente SNMP definidas en SNMP versión 1 y 2.

Del lado del Server se mantienen referencias (proxyMBeans) a los MBeans registrados en el agente de tal forma de poder obtener información, recibir notificaciones o realizar alguna operación que ofrezcan los MBeans, todo de forma remota. El Servidor oficia de proxy entre el agente y el cliente. Para el cliente es transparente la existencia del agente, solo se comunica con el servidor tanto para obtener información de los MBeans como para recibir notificaciones.

Se registraron proxyMBeans tanto para los MBeans de recursos, como para los de Monitor.

El Cliente por su lado está escuchando por notificaciones que envían los monitores del agente. Esto se hace de forma remota a través del Servidor.
El cliente también solicita información de los MBeans haciendo invocaciones remotas al Servidor.

Las conexiones, tanto para Agente-Servidor, como para Servidor-Cliente son establecidas mediante RMI.
Se realizaron pruebas ubicando los componentes en la misma máquina y también en máquinas distintas.

8.4.2 Aspectos probados satisfactoriamente

Arquitectura distribuida: se consiguió implementar tres componentes (Agente, Servidor, Cliente) que corriendo en distintas máquinas virtuales, inclusive en distintas máquinas físicas, se comunicaron de forma exitosa.

Conectores:
Los componentes se comunicaron con conectores RMI, tanto para notificaciones remotas, invocaciones remotas, como para mantener referencias remotas a MBeans.

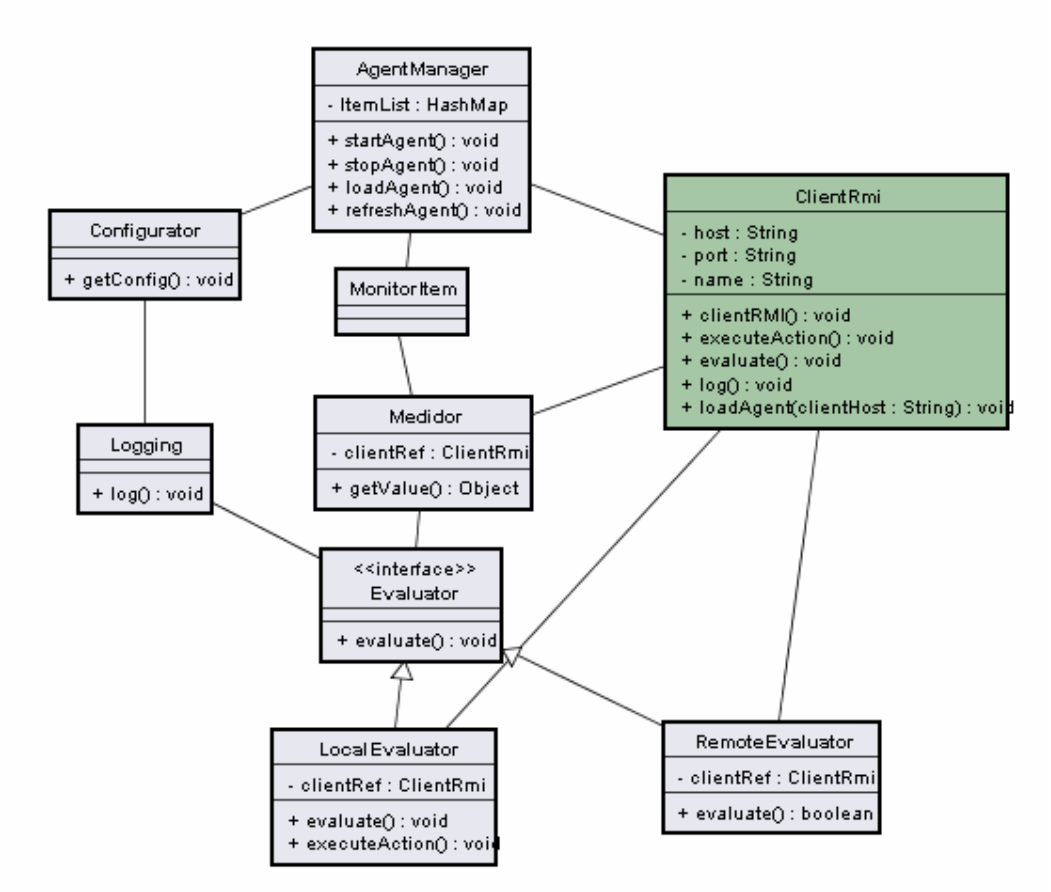
Monitoreo:

Se utilizó el servicio de monitoreo ofrecido por la MX4J. Se probaron el StringMonitor y el Gauge Monitor. Vimos la posibilidad de extender este servicio para otro tipo de evaluación, complementándolo con un componente evaluador.

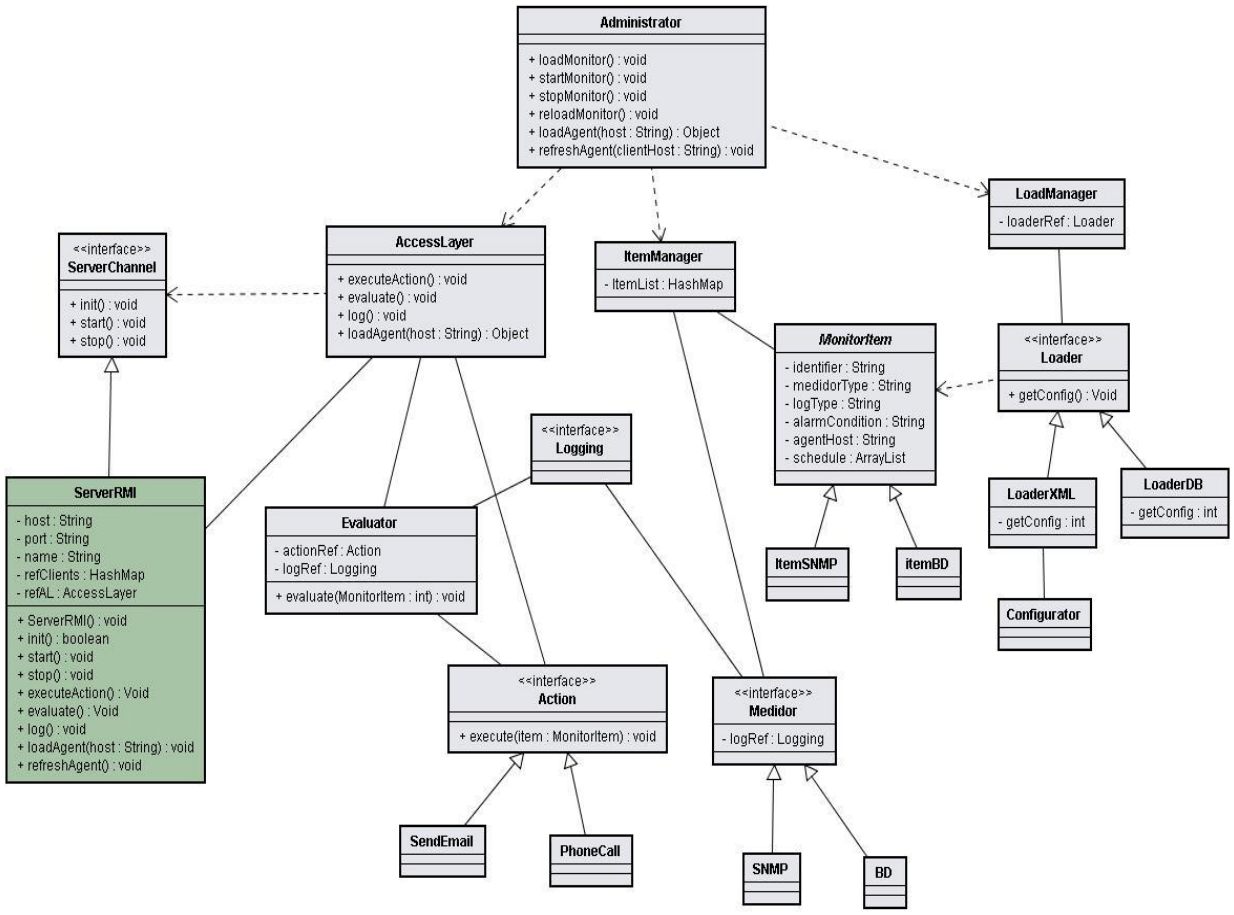
Manejo con SNMP: se realizaron operaciones GET y SET.

8.5 PRIMER DIAGRAMA DE CLASES

8.5.1 Diagrama de clases del Agente

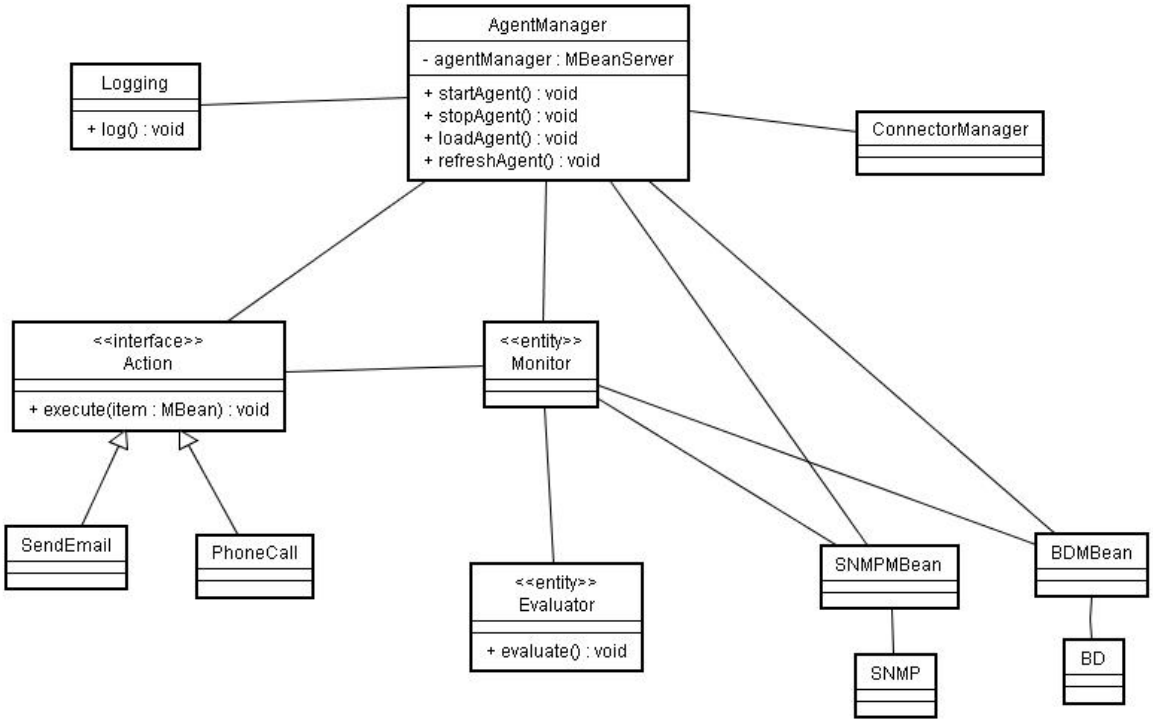


8.5.2 Diagrama de clases del Servidor

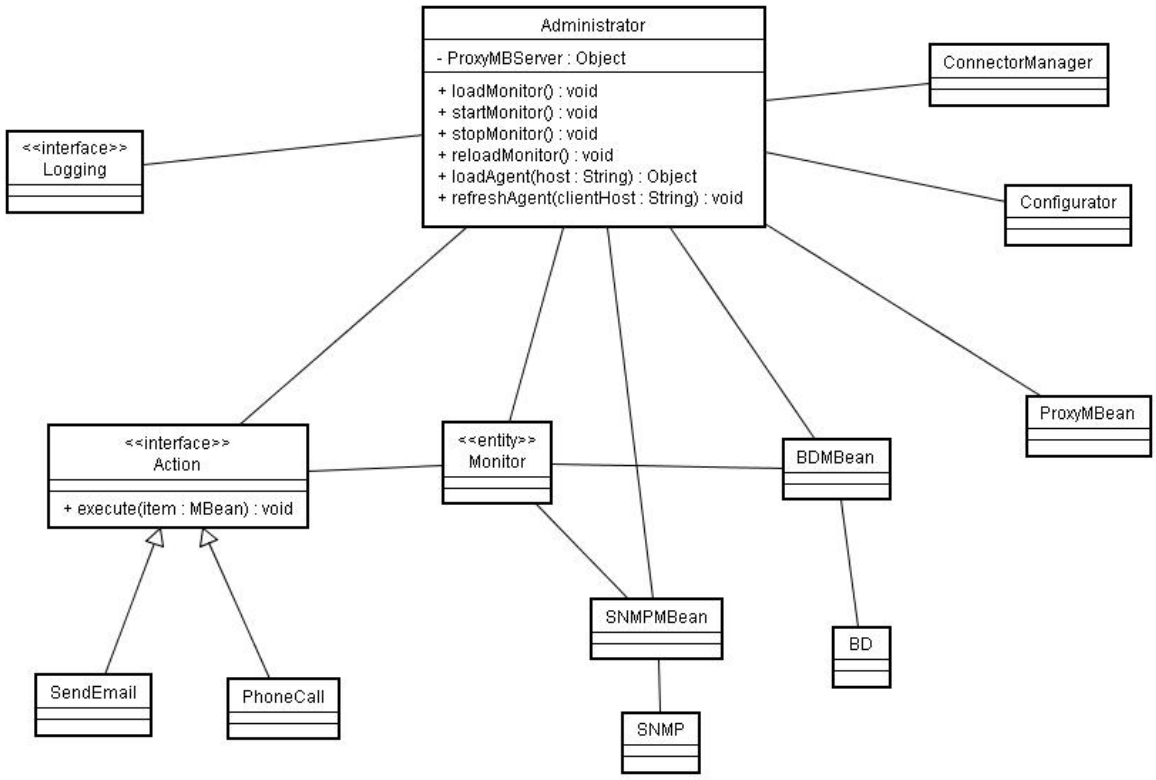


8.6 DIAGRAMA DE CLASES TENTATIVO

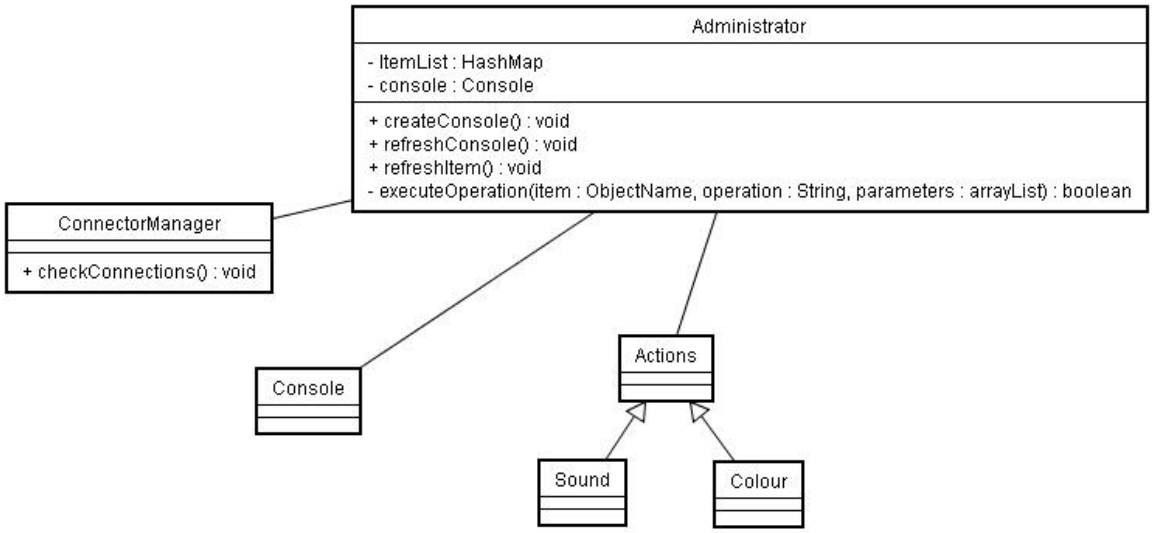
8.6.1 Diagrama de clases del Agente



8.6.2 Diagrama de clases del Servidor

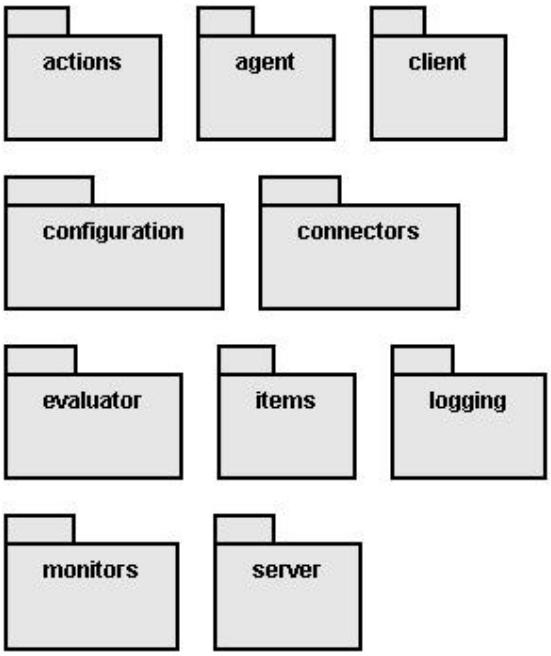


8.6.3 Diagrama de clase del Cliente

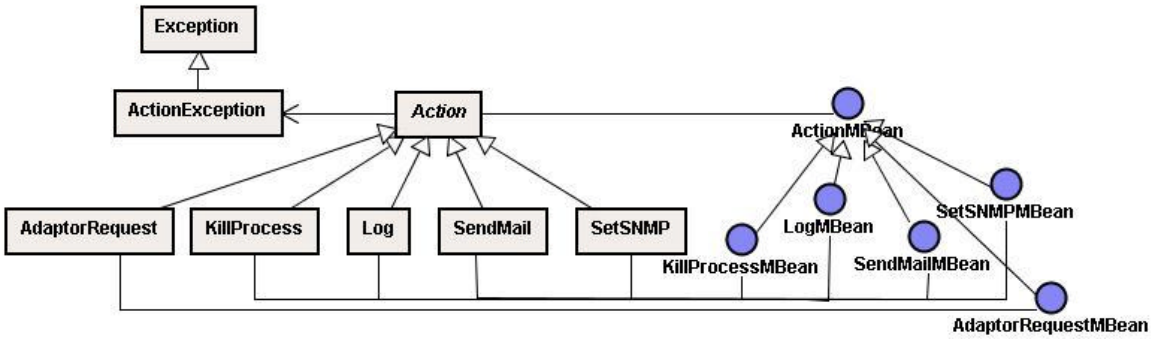


8.7 DIAGRAMA DE CLASES GENERADO

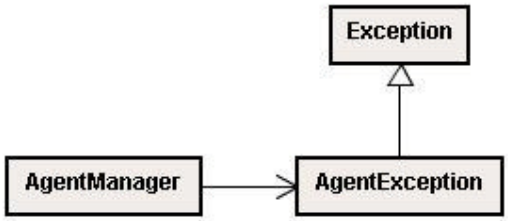
Paquetes de java



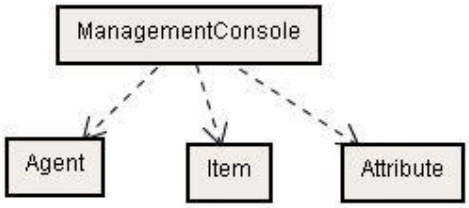
Paquete actions



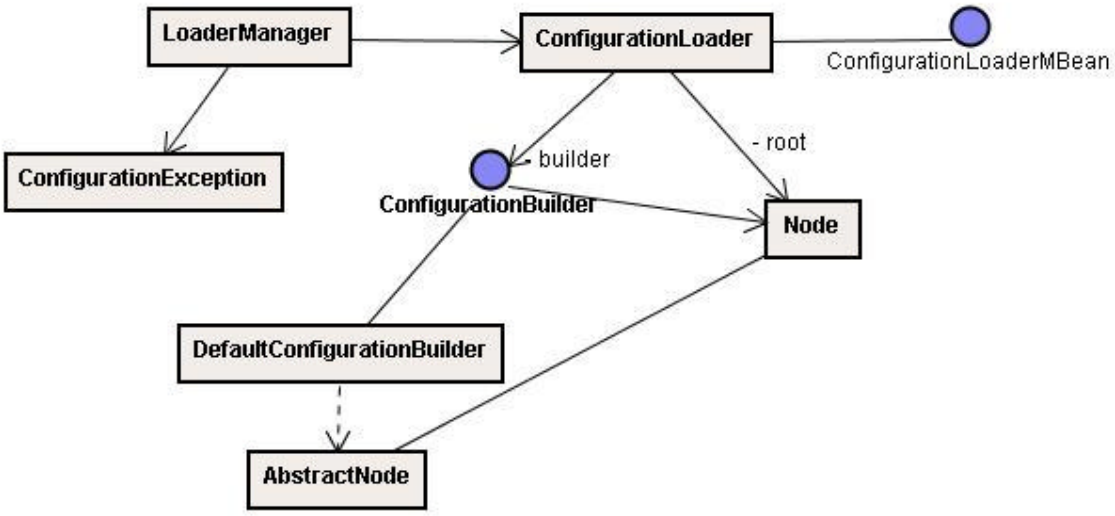
Paquete agent



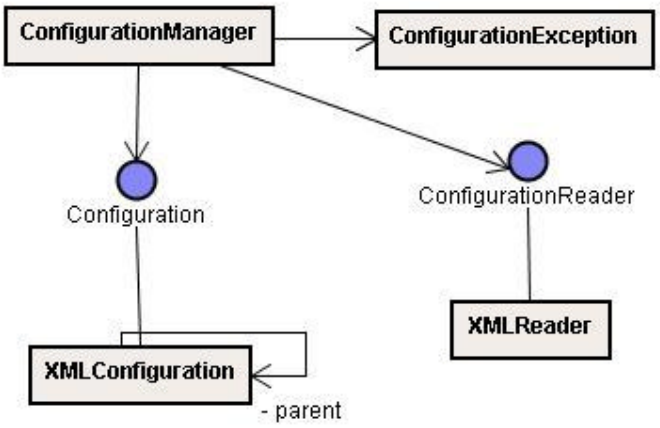
Paquete client



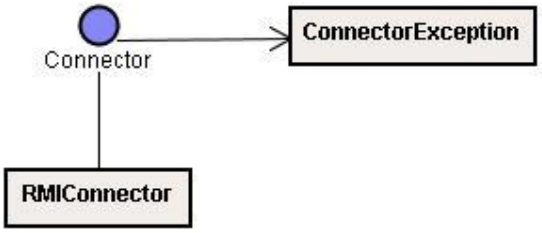
Paquete configuration loader



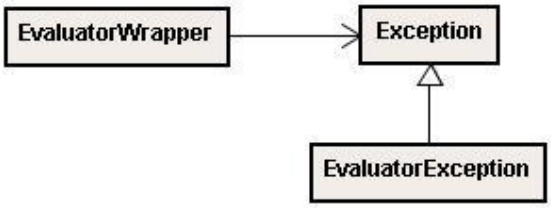
Paquete configuration reader



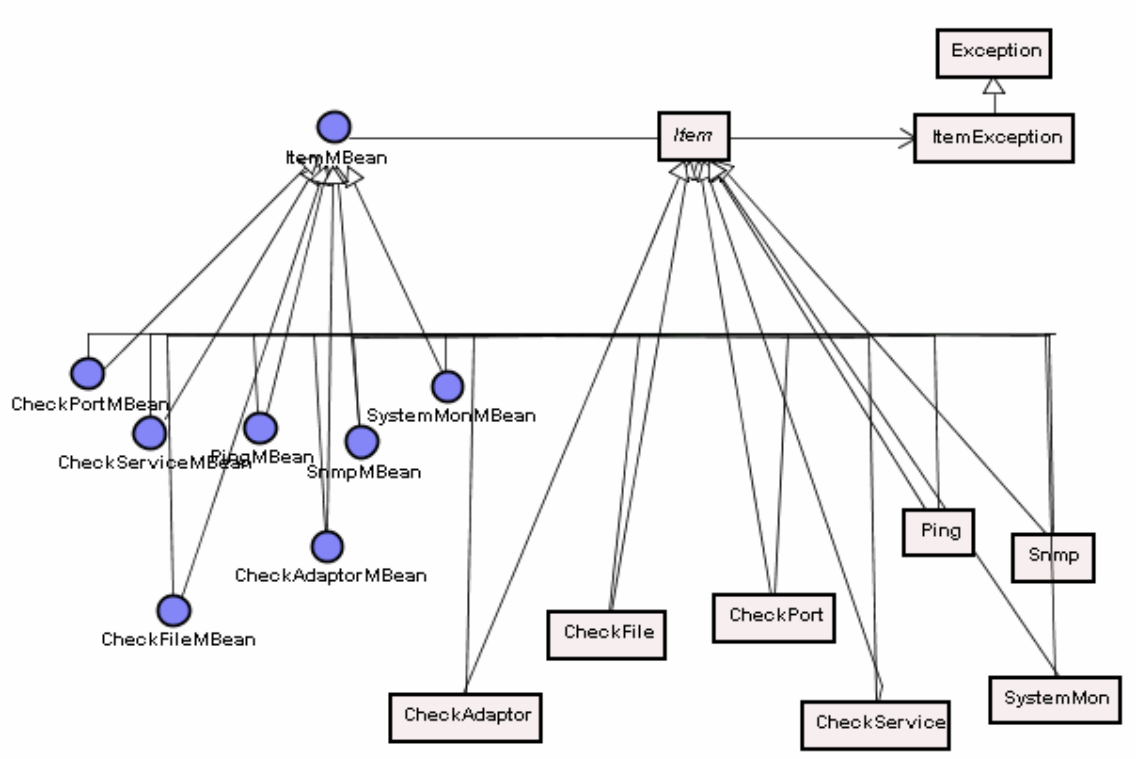
Paquete connectors



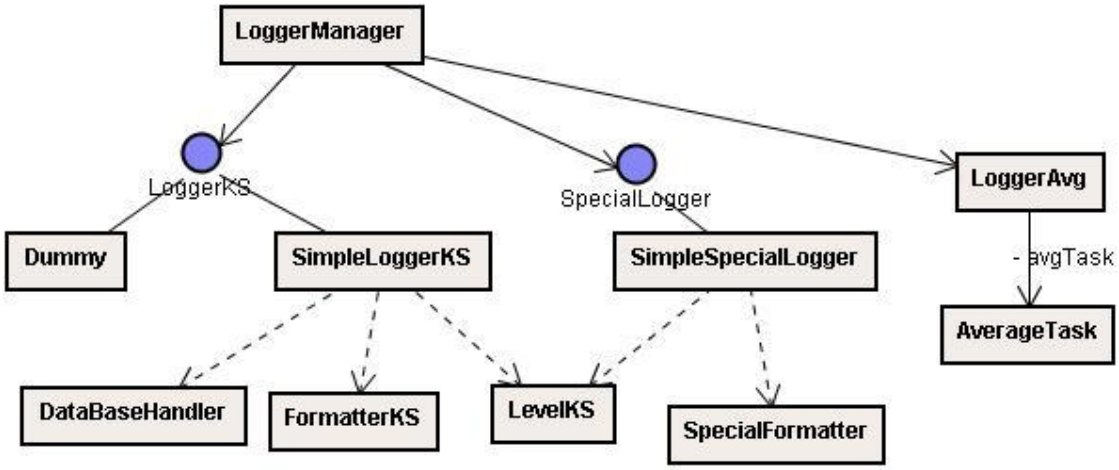
Paquete evaluator



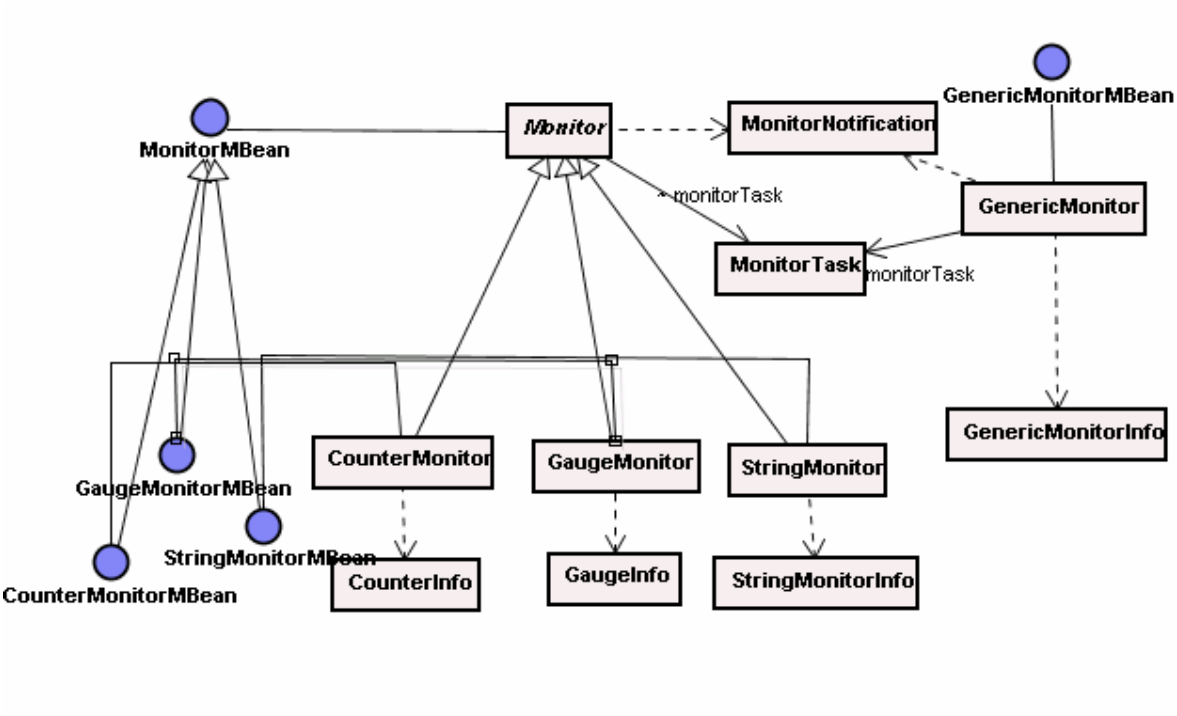
Paquete items



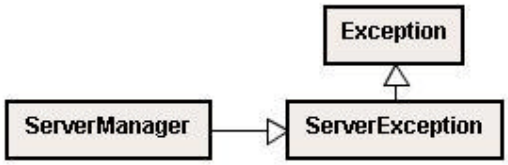
Paquete logging



Paquete monitors



Paquete server



8.8 Casos de prueba

8.8.1 Casos de prueba para el componente de registro.

8.8.1.1 En archivos

Para efectuar estas pruebas se generan las condiciones necesarias y se verifican los resultados.
Los archivos de configuración utilizados se encuentran en la carpeta *Test/Registro/8.8.1.1*.

1. El archivo de log no existe.
   En este caso se deberá registrar en la ruta por defecto definida al momento de inicialización del sistema.

   Resultado
   El resultado fue satisfactorio.
   Se ejecuta la aplicación sin tener creado el archivo de log, se verificó que el registro se hace en la ruta por defecto mencionada.

2. Inhabilitar el registro a archivos.
   Se registrará en cierto archivo existente. Se espera que no se haya registro de log.

   Resultado
   El resultado fue satisfactorio.
   Se verificó que la ejecución de la aplicación no registra datos en archivos cuando está inhabilitada esta opción.

3. Registro de archivos habilitado.
   Se registrará en cierto archivo existente. Se espera encontrar el registro de log en el archivo.

   Resultado
   El resultado fue satisfactorio.
   Con el registro en archivos habilitado se ejecuta la aplicación, se esperaba encontrar el registro en la ruta especificada para tal archivo.

8.8.2 En base de datos

Se intenta verificar el acceso a base de datos y la correcta ejecución de las operaciones de registro.
Los archivos de configuración utilizados se encuentran en la carpeta *Test/Registro/8.8.1.2* .

1. En Registro a base de datos inhabilitado.
   Se registrará en cierta BD para la cual ya se estableció una conexión. Se espera que no haya registro de log en la BD.

   Resultado
   El resultado fue satisfactorio.
   Se verifica que no se generan registros en la base de datos cuando se inhabilita esta opción

2. Registro de base de datos habilitado.
   Se registrará en cierta BD para la cual ya se estableció una conexión. Se espera que haya registro de log en la BD.

   Resultado
   El resultado fue satisfactorio.
   Se comprueba la existencia de registros en la base de datos

8.8.3 Casos de prueba para pluggins de ítems de monitoreo.

8.8.3.1 Búsqueda en archivos

1. El archivo no existe.
   Se debe enviar un mensaje de error.

   Resultado
   El resultado fue satisfactorio.
   Se intenta utilizar la búsqueda en un archivo que no existe, se genera un mensaje de error

2. La lectura de archivo se retoma en el último fin de lectura de la lectura anterior. El mecanismo de lectura del archivo implica que se recuerda la última posición leída en éste. Se quiere verificar que esto funciona realmente así.

   Resultado
   El resultado fue satisfactorio.
   Se comprueba que al ejecutar búsquedas sucesivas, se recuerda la ultima posición leída retomando en la siguiente búsqueda la ultima posición de la búsqueda anterior.

3. El archivo no contiene el texto de error buscado.
   Se quiere comprobar el resultado de la búsqueda en un archivo que no contiene el texto buscado

   Resultado
   El resultado fue satisfactorio. Se comprueba el comportamiento esperado

4. El archivo contiene el texto buscado pero no dentro de la sección determinada.

   Resultado
   El resultado fue satisfactorio. Se comprueba el comportamiento esperado

5. El archivo contiene el texto buscado dentro de la sección determinada.

   Resultado
   El resultado fue satisfactorio. Se comprueba el comportamiento esperado
   Se notifica que el texto fue encontrado

8.8.3.2 Consultas SNMP

Se implementaron una serie de casos de prueba en JUnit para comprobar la normal ejecución de los casos planteados en los puntos 1, 2 y 3. Se hizo la misma prueba con el servicio SNMP activo y detenido. El resultado fue satisfactorio en los tres casos.

1. El agente SNMP no existe o no está activo.
   Se debe enviar un mensaje de error.

   Resultado
   El resultado fue satisfactorio.
   Se genera un mensaje de error que no se pueden recuperar valores del OID requerido.

2. OID no válido o no disponible.
   Se realizan consultas SNMP con números incorrectos y se debe capturar un error declarando este hecho.

Resultado
El resultado fue satisfactorio.
Se genera un mensaje de error avisando que no se pueden recuperar valores para el OID requerido.

3. Para cada OID validos, chequear que retorna los valores esperados.
   Se realizan consultas SNMP con OID validos en la base de datos MIB, no es un aprueba exhaustiva.

   Resultado
   El resultado fue satisfactorio.
   Se recuperaron valores validos para los OID's especificados

8.8.3.3 Ejecución de petición al servicio de K&S Middleware.

Se busca comprobar el normal funcionamiento del pluggin que monitorea la normal ejecución de K&S Middleware, esto implica verificar de alguna forma que el servicio está activo y responde a peticiones específicas. En este caso se ejecuta una petición cada cierta frecuencia y en base a su respuesta se determina el estado del servicio.

1. El servicio de Middleware no está disponible.
   Indicar que el servicio está bajo.

   Resultado
   El resultado fue satisfactorio.
   Se ejecuta la prueba con el servicio bajo y se recibe la notificación de error

2. Canal de comunicación no soportado.
   Solo están implementados los canales RMI y Socket. En caso de especificar el uso de otro canal se debe enviar un mensaje de error.

   Resultado
   El resultado fue satisfactorio.
   Se verifica la correcta ejecución de la prueba para los dos canales, en caso de especificar otro posible canal se notifica este hecho.

8.8.3.4 Chequeo de puerto

Se realizaron caso de prueba JUnit para verificar una serie de puertos correctos y uno con un valor incorrecto.

1. Número de puerto incorrecto.
   Se debe notificar el error.

   Resultado
   El resultado fue satisfactorio.
   Se genera un mensaje de error de que no se puede conectar con ese puerto

2. No se puede establecer una conexión con el puerto.
   Se debe notificar el error.

   Resultado
   El resultado fue satisfactorio.
   Se genera un mensaje de error de que no se puede conectar con ese puerto

3. Puerto habilitado.
   Se debe indicar que se pudo establecer la conexión al puerto.

Resultado
El resultado fue satisfactorio.
El test se realiza correctamente

8.8.3.5 Ejecución del comando PING

Se Implementa un caso de prueba que verifica el funcionamiento de la ejecución del comando PING a una dirección válida y a una dirección no valida.

1. La dirección IP a la que se hace el PING no existe.
   Debe dar un mensaje declarando este hecho.

   Resultado
   El resultado fue satisfactorio.
   Se da un mensaje declarando que no se puede conectar o que la dirección es no valida

2. La dirección IP a la que se hace es inalcanzable.
   Debe dar un mensaje declarando este hecho.

   Resultado
   El resultado fue satisfactorio.
   Se da un mensaje declarando que no se puede conectar o que la dirección es no valida

3. La dirección IP a la que se hace PING es correcta.
   Se debe indicar que el PING se realizó correctamente.

   Resultado
   El resultado fue satisfactorio.
   Se da un mensaje declarando que se pudo conectar correctamente

8.8.4 Casos de prueba de monitores.

Estas pruebas se repetirán para cada tipo de monitor: String Monitor, Counter Monitor, Gauge Monitor y Generic Monitor. Los archivos de configuración utilizados se encuentran en la carpeta *Test/Monitores/*.

1. Verificar que cada monitor se ejecuta con la granularidad especificada.
   Se considerarán distintos períodos de tiempo para hacer este control. Primero se controlará durante 1 día, luego durante una semana. Se inspeccionarán los registros de base de datos para cada monitor, a partir de la cual se calcularán los intervalos de tiempo entre ejecuciones. Estos deberán coincidir con la frecuencia determinada.

   Resultado
   El resultado fue satisfactorio.
   Se verificó que para cada monitor se cumple con el período especificado en su granularidad.
   Fue comprobado en los registros de la base de datos que cada monitor ingresó los registros a la base de datos con la frecuencia esperada. Se comprobó que este comportamiento se mantuvo a lo largo de dos semanas.

   Se realizaron las siguientes consultas a BD luego de dos semanas de actividad del monitor:

```
select Max(IdItem0422), count(*) cantidad, datediff(ss,min(tmstmp0422) , max(tmstmp0422) ) / count(*) intervalo
from LgMonAtr0422(nolock)
where IdItem0422 = 'port110'
and tmstmp0422 > '2004-04-28 21:00:00.000'
and Agent_0422 = 'tololo'

select Max(IdItem0422), count(*) cantidad, datediff(ss,min(tmstmp0422) , max(tmstmp0422) ) / count(*) intervalo
from LgMonAtr0422(nolock)
where IdItem0422 = 'Adaptor'
and tmstmp0422 > '2004-04-28 21:00:00.000'
and Agent_0422 = 'tololo'

select Max(IdItem0422), count(*) cantidad, datediff(ss,min(tmstmp0422) , max(tmstmp0422) ) / count(*) intervalo
from LgMonAtr0422(nolock)
where IdItem0422 = 'ping1'
and tmstmp0422 > '2004-04-28 21:00:00.000'
and Agent_0422 = 'tololo'
```

Resultados:

Item Monitoreado	cantidad de registros	intervalo(segundos)
port110	40632	30
Adaptor	60945	20
ping1	60944	20

También se realizó el seguimiento exhaustivo para algunos de los ítems monitoreados.
A continuación se muestras las consultas realizadas junto con los resultados de las mismas.

```
select IdItem0422 Item, tmstmp0422
from LgMonAtr0422(nolock)
where IdItem0422 = 'ping1'
and Agent_0422 = 'tololo'
and tmstmp0422 > '2004-05-01 00:00:00.000'
and tmstmp0422 < '2004-05-01 00:02:00.000'
```

Item	tmstmp0422
ping1	2004-05-01 00:00:03.760
ping1	2004-05-01 00:00:23.760
ping1	2004-05-01 00:00:43.820
ping1	2004-05-01 00:01:03.760
ping1	2004-05-01 00:01:23.760
ping1	2004-05-01 00:01:43.760

```
select IdItem0422 Item, tmstmp0422
from LgMonAtr0422(nolock)
where IdItem0422 = 'port110'
and Agent_0422 = 'tololo'
and tmstmp0422 > '2004-05-04 10:00:00.000'
and tmstmp0422 < '2004-05-04 10:02:00.000'
```

Item	tmstmp0422
port110	2004-05-04 10:00:21.730
port110	2004-05-04 10:00:51.710
port110	2004-05-04 10:01:22.443
port110	2004-05-04 10:01:51.780

```
select IdItem0422 Item, tmstmp0422
from LgMonAtr0422(nolock)
where IdItem0422 = 'Adaptor'
and Agent_0422 = 'tololo'
and tmstmp0422 > '2004-05-14 04:00:00.000'
and tmstmp0422 < '2004-05-14 04:02:00.000'
```

Item	tmstmp0422
Adaptor	2004-05-14 04:00:04.250
Adaptor	2004-05-14 04:00:24.250
Adaptor	2004-05-14 04:00:44.250
Adaptor	2004-05-14 04:01:04.250
Adaptor	2004-05-14 04:01:24.250
Adaptor	2004-05-14 04:01:44.250

Los resultados permiten confirmar que los intervalos de tiempo coinciden con la granularidad especificada en el archivo de configuración para cada monitor definido.

2. Verificar que efectivamente se notifican y registran las condiciones de error definidas. Para estas pruebas se generarán condiciones de error de las cuales se espera se reporte el error en la base de datos, y se ejecuten las acciones asociadas al monitor.

   Resultado
   El resultado fue satisfactorio.
   Las condiciones de error y los cambios en el estado de cada recurso fueron registradas tanto en BD, como en archivos si es que están habilitados.
   Fue verificada la ejecución de las acciones correspondientes, asociadas a cada una de las situaciones de error generadas.

3. Verificar que en caso de darse nuevamente una condición de alarma luego de haber pasado por una situación aceptable, se notificará nuevamente la acción asociada al monitor.

   Resultado
   El resultado fue satisfactorio.
   Se verifico en la ejecución de cada tipo de monitor, en cada caso se controló el normal funcionamiento del monitoreo, se reprodujo una condición de error verificando que se da la notificación correspondiente. Luego se elimina la condición de error y se verifica que se notifica nuevamente.

   Se muestra a continuación algunos de los resultados generados para el caso de el monitoreo realizando el PING a una máquina. Para esto se desconectó la máquina de la red de lo cual se registró un WARN a las 17:35:28.482 y se recibió un mail. Luego se conectó la máquina a la red por lo cual se recuperó el estado como se indica a las 17:36:08.491. Después se desconectó nuevamente la máquina y se notificó nuevamente el error y se generó otro mail.

```
17:35:08.491  LEVEL: INFO   Execute Monitor: monitor1, Item: ping1, Atribute: Status, Value: TRUE
17:35:28.482  LEVEL: WARN  Execute Monitor: monitor1, Item: ping1, Atribute: Status, Value: FALSE, DESC: Found a match.
17:35:48.542  LEVEL: WARN  Execute Monitor: monitor1, Item: ping1, Atribute: Status, Value: FALSE
17:36:08.491  LEVEL: INFO   Execute Monitor: monitor1, Item: ping1, Atribute: Status, Value: TRUE, DESC: Recover the status OK.
17:36:28.579  LEVEL: INFO   Execute Monitor: monitor1, Item: ping1, Atribute: Status, Value: TRUE
17:35:48.489  LEVEL: WARN  Execute Monitor: monitor1, Item: ping1, Atribute: Status, Value: FALSE, DESC: Found a match.
```

Primer mail:

Notification on 14-may-2004 17:35:28 sent by :

Error in Monitor :Monitors:name=monitor1

Item Checked :Items:name=ping1

Attribute :Status

Value :FALSE
Error condition :found a match

Segundo mail:

Notification on 14-may-2004 17:35:48 sent by :

Error in Monitor :Monitors:name=monitor1
Item Checked :Items:name=ping1
Attribute :Status
Value :FALSE
Error condition :found a match

8.8.5 Casos de prueba para pluggins de acciones.

8.8.5.1 Registro

1. La ruta de destino del archivo de log no existe.
   En este caso se deberá notificar mediante un error.

   Resultado
   El resultado fue satisfactorio. Se comprueba la notificación del error

2. El archivo existe.
   Se espera encontrar un registro en el archivo de log.
   Se generará una condición de alarma para un monitor que tiene asociada una acción de registro.

   Resultado
   El resultado fue satisfactorio. Se verificó que los monitores que tienen asociada una acción de registro efectivamente lo hacen.

8.8.5.2 Ejecución de e-mail

Se diseño un caso de prueba JUnit que verifica el correcto funcionamiento del envió de e-mail con una serie de características definidas.

1. El correo se envía exitosamente. Se busca probar solamente el envió de correo

   Resultado
   El resultado fue satisfactorio.
   El mensaje se envía correctamente y el destinatario lo recibe.

2. Cuando los monitores tienen asociada la acción SendMail, ésta se ejecuta correctamente con los datos especificados y el mensaje preestablecido.

Resultado
El resultado fue satisfactorio.
Los monitores realizan esta acción en forma correcta, detectan la condición de error, encapsula la notificación, ejecutan la acción y esta se envía a el o loas destinatarios.

8.8.5.3 Modificación del valor de un objeto SNMP

Se diseño un caso de prueba JUnit que verifica el resultado de la operación set sobre un objeto habilitado en la base de datos, se ejecuta un caso con un OID valido y otro con un OID no disponible para escritura.

1. OID no válido o no disponible para escritura.

   Resultado
   El resultado fue satisfactorio.
   Se genera un mensaje de que el objeto no esta disponible para escritura o que no se puede escribir sobre el objeto

2. OID válido.

   Resultado
   El resultado fue satisfactorio.
   Se realiza la operación de escribir el valor en el objeto y se verifica que realmente el objeto cambia su valor

8.8.5.4 Parar un proceso del sistema operativo

1. No existe el proceso, no debe ejecutar ninguna acción.
   Resultado
   El resultado fue satisfactorio.

2. Existe el proceso. Luego de pararlo, verificar en la lista de procesos del sistema que no exista un proceso del sistema con ese identificador.

   Resultado
   El proceso es detenido

8.8.5.5 Casos de prueba en el componente "Agente"

8.8.5.6 No existe el archivo de configuración

No existe el archivo de configuración desde el cual el Agente lea la configuración.
Se debe desplegar un mensaje de error y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
El archivo de configuración utilizado se encuentra en *test/Agente/8.8.5.6*. Se ejecutó el componente especificando un archivo inválido para el tag "AGENTMANAGER" en el archivo *configuration.ini*.
El archivo de log generado se encuentra en *test/Agente/8.8.5.6/Resultados*.

8.8.5.7 Sección del archivo de configuración mal configurada.

1. Declaración de un tipo no soportado.
En caso de que se declare un tipo de ítem, monitor, acción o conector que la aplicación no soporta, se debe notificar con un mensaje de error y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
La prueba se realizó modificando las secciones correspondientes en el archivo de configuración.
Se hicieron cuatro pruebas para verificar que se detectan especificaciones de elementos no soportados, una por cada sección, ítems, actions, monitors, connectors.
En los cuatro casos fue verificado que el sistema no se sigue ejecutando y se notifica la condición de error. Los archivos de configuración utilizados se encuentran en *test/Agente/8.8.5.7/1/*.
El archivo de log generado se encuentra en *test/Agente/8.8.5.7/1/Resultados/*.

2. Declaración incorrecta de los argumentos para una sección.
Cada sección, según el tipo, debe contener cierta cantidad de argumentos específicos, en un orden predeterminado. Si los argumentos de una sección no son correctos se debe desplegar un mensaje de error y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
Fueron realizadas cuatro pruebas, una por cada sección especificando en cada una los argumentos para la creación de los objetos. En cada caso estos argumentos fueron mal especificados tratando crear las condiciones de error al reconocer los argumentos.
Cuando se reconoce que los argumentos de la sección correspondiente no son correctos encada caso, se aborta la ejecución del sistema y se registra el error.
Los archivos de configuración utilizados se encuentran en *test/Agente/8.8.5.7/2/*.
El archivo de log generado se encuentra en *test/Agente/8.8.5.7/2/Resultados/*.

8.8.5.8 Problemas al crear el conector.

En caso de no poder crear el conector se debe notificar un mensaje de error con el motivo por el cual no se pudo crear el conector.
La prueba se efectúa declarando en el archivo de configuración de forma incorrecta la sección de conectores, ya sea definiendo un tipo de conector no soportado, o declarando mal los argumentos.

Resultado
El resultado fue satisfactorio.
En ambos casos se verifica el comportamiento esperado.
El archivo de configuración utilizado se encuentra en *test/Agente/8.8.5.8/*.
El archivo de log generado se encuentra en *test/Agente/8.8.5.8/Resultados/*.

8.8.5.9 Problemas con el atributo a monitorear.

1. El atributo no es soportado por el ítem de monitoreo.
   Si el atributo no está definido para el tipo de ítem a monitorear se debe notificar con un mensaje de error, pero no finalizar el sistema, finaliza el monitoreo para ese atributo en particular.

   Resultado
   El resultado fue satisfactorio.
   Se registra la causa del error, se notifica esta y se detiene ese monitor en particular sin detener la ejecución del sistema.

El archivo de configuración utilizado se encuentra en *test/Agente/8.8.5.9/1/*.
El archivo de log generado se encuentra en *test/Agente/8.8.5.9/1/Resultados/*.

2. El atributo no es compatible con el tipo de monitor.
   Chequear que el tipo del atributo y el tipo de monitor son compatibles, si no lo son se debe notificar con un mensaje de error, pero no finalizar el sistema, finaliza el monitoreo para ese atributo en particular.

   Resultado
   El resultado fue satisfactorio.
   Se registra la causa del error, se notifica esta y se detiene ese monitor en particular sin detener la ejecución del sistema.
   El archivo de configuración utilizado se encuentra en *test/Agente/8.8.5.9/2/*.
   El archivo de log generado se encuentra en *test/Agente/8.8.5.9/2/Resultados/*.

3. Problemas para obtener el valor del atributo.
   Si al obtener el valor del atributo se da un error, éste se debe notificar.
   Finaliza el monitoreo para ese atributo en particular, pero no finalizar el sistema.

   Resultado
   El resultado fue satisfactorio.
   Se registra la causa del error, se notifica esta y se detiene ese monitor en particular sin detener la ejecución del sistema.
   El archivo de configuración utilizado se encuentra en *test/Agente/8.8.5.9/3/*.
   El archivo de log generado se encuentra en *test/Agente/8.8.5.9/3/Resultados/*.

8.8.6 Casos de prueba en el componente "Servidor"

8.8.6.1 No existe el archivo de configuración.

No existe el archivo de configuración desde el cual el Servidor levanta la configuración.
Se debe desplegar un mensaje de error y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
Se intento ejecutar el componente sin direccionar archivo de configuración. Se cancela la ejecución y se notifica el error.
El archivo de configuración utilizado se encuentra en *test/Servidor/8.8.6.1/*.
El archivo de log generado se encuentra en *test/Agente/8.8.6.1/Resultados/*.

8.8.6.2 Falta una sección en el archivo de configuración.

1. Falta sección de agentes.

Se debe desplegar un mensaje notificando que la sección no existe y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
Se ejecuta el componente sin especificar cada sección controlando en cada caso la evaluación de que la sección correspondiente no existe.
En cada caso se detiene la ejecución del componente
El archivo de configuración utilizado se encuentra en *test/Servidor/8.8.6.2/1/*.
El archivo de log generado se encuentra en *test/Agente/8.8.6.2/1/Resultados/*.

2. Falta sección de conectores.
Se debe desplegar un mensaje notificando que la sección no existe y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
Se ejecuta el componente sin especificar la sección de conectores, se verifica que este hecho se reconoce y se notifica el error. Se detiene la ejecución del componente.
El archivo de configuración utilizado se encuentra en *test/Servidor/8.8.6.2/2/*.
El archivo de log generado se encuentra en *test/Agente/8.8.6.2/2/Resultados/*.

8.8.6.3 Problemas al crear el conector.

Se debe desplegar un mensaje notificando el motivo y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
Se finaliza la carga del componente y se registra el motivo de error.
El archivo de configuración utilizado se encuentra en *test/Servidor/8.8.6.3/*.
El archivo de log generado se encuentra en *test/Agente/8.8.6.3/Resultados/*.

8.8.6.4 Problemas para conectarse a alguno de los "Agentes".

Se debe desplegar un mensaje notificando el motivo, pero no se debe finalizar el sistema.
Se debe intentar crear conexiones con otros Agentes si las hubiera.

Resultado
El resultado fue satisfactorio.
Se crean las conexiones con los agentes restantes y se notifica el error y el conector o agente con el que no se establece la conexión.
El archivo de configuración utilizado se encuentra en *test/Servidor/8.8.6.4/*.
El archivo de log generado se encuentra en *test/Agente/8.8.6.4/Resultados/*.

8.8.7 Casos de prueba en el componente "Cliente"

En este capítulo, cuando hacemos referencia a Cliente, nos referimos a la consola gráfica de administración implementada.

8.8.7.1 No existe el archivo de configuración.

No existe el archivo de configuración desde el cual el Cliente levanta la configuración.
Se debe desplegar un mensaje de error y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
Se intento ejecutar el componente sin direccionar archivo de configuración. Se cancela la ejecución y se notifica el error.
El archivo de configuración utilizado se encuentra en *test/Cliente/8.8.7.1/*.
El archivo de log generado se encuentra en *test/Cliente/8.8.7.1/Resultados/*.

8.8.7.2 Falta la sección conectores en el archivo de configuración.

Se debe desplegar un mensaje notificando que la sección no existe y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
Se ejecuta el componente sin especificar la sección de conectores, se verifica que este hecho se reconoce y se notifica el error. Se detiene la ejecución del componente.
El archivo de configuración utilizado se encuentra en *test/Cliente/8.8.7.2/*.
El archivo de log generado se encuentra en *test/Cliente/8.8.7.2/Resultados/*.

8.8.7.3 Problemas para conectarse al "Servidor" o al "Agente".

Se debe desplegar un mensaje notificando el motivo por el cual no se pudo establecer la conexión, y finalizar el sistema.

Resultado
El resultado fue satisfactorio.
Para realizar la prueba se intenta establecer la comunicación con una definición de conectores incompatible, se notifica el error y se detiene la ejecución del componente
El archivo de configuración utilizado se encuentra en *test/Cliente/8.8.7.3/*.
El archivo de log generado se encuentra en *test/Cliente/8.8.7.3/Resultados/*.

8.8.8 Pruebas de integración

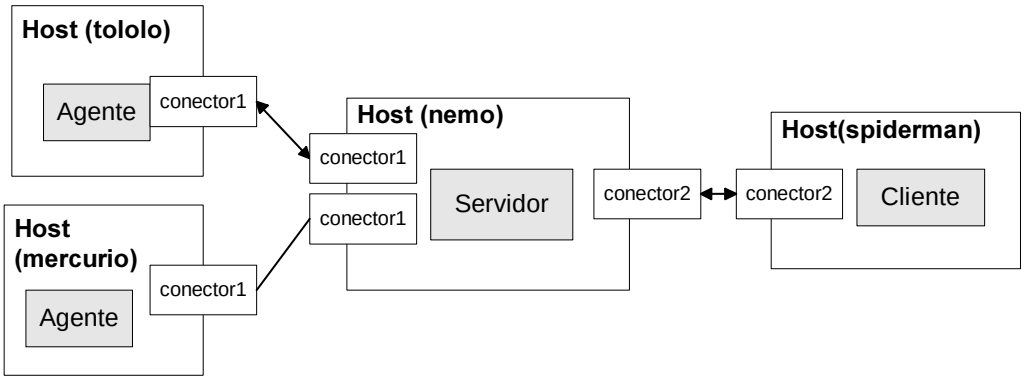
Para realizar estas pruebas consideramos exitosas las pruebas de los distintos componentes detalladas anteriormente.

8.8.8.1 Caso de prueba de integración 1

Contexto
Las pruebas se realizarán en la plataforma LINUX.
Habrá dos Agentes corriendo cada uno en una máquina distinta, un Servidor y un Cliente corriendo en otras máquinas.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se parará de forma manual uno de los Agentes.
Los archivos de configuración utilizados se encuentran en *test/Integracion/8.8.8.1/*.

Esquema:



Resultados esperados
El Servidor deberá notificar en su archivo de log que perdió la conexión con el agente.
Éste suceso no deberá afectar la comunicación con el otro agente.
El Cliente también tendrá que ser notificado y deberá reflejar de alguna forma el hecho.

Resultados obtenidos.
Se verifica en forma satisfactoria que desconectar un agente no afecta el funcionamiento de los otros agentes.
Está pendiente lograr que el Servidor notifique y registre que pierde la conexión, también la notificación al cliente.

8.8.8.2 Caso de prueba de integración 2

Contexto
Condiciones idénticas al caso de prueba de integración 1.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se parará de forma manual el Servidor.

Resultados esperados
Este cambio no deberá afectar el correcto funcionamiento de los agentes.
El Cliente tendrá que ser notificado y deberá reflejar de alguna forma el hecho.
El Servidor deberá registrar si es posible el motivo de su falta de actividad.

Resultados obtenidos.
Se verifica en forma satisfactoria que desconectar el Servidor no afecta el funcionamiento de los agentes.
Esta pendiente la notificación al cliente.

8.8.8.3 Caso de prueba de integración 3

Contexto
Condiciones idénticas al caso de prueba de integración 1.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se parará de forma manual el Cliente.

Resultados esperados
Este cambio no afectará el funcionamiento de los Agentes y el Servidor.

Resultados obtenidos.
Se verifica en forma satisfactoria que desconectar cliente no afecta el funcionamiento de los agentes y del Servidor.

8.8.8.4 Caso de prueba de integración 4

Contexto
Las pruebas se realizarán en la plataforma LINUX.
Habrá un Agente, un Servidor y un Cliente corriendo en la misma máquina. Se quiere determinar la incidencia en el desempeño del equipo en el cual corren los componentes.
Los archivos de configuración utilizados se encuentran en *test/Integracion/8.8.8.4/*.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se harán mediciones de consumo de CPU y memoria para este esquema de funcionamiento.

Resultados esperados
Se espera tener idea de cómo inciden en uso de recursos disponible de la máquina en la cual corren todos los componentes juntos, tales como memoria o consumo de CPU.

Resultados obtenidos.
Se verifica que el funcionamiento de los tres componentes no altera la carga de CPU, si bien tiene requerimientos de memoria propios de el uso de Java. Los picos altos de consumo de CPU se dan en el momento de la inicialización de los componentes. Para la prueba se monitorearon 12 recursos.

8.8.8.5 Caso de prueba de integración 5

Contexto
Las pruebas se realizarán en la plataforma WIN.
Habrá un Agente, un Servidor y un Cliente corriendo en la misma máquina. Se quieren evaluar las mismas medidas tomadas en el caso 4 pero en esta configuración particular.
Los archivos de configuración utilizados se encuentran en *test/Integracion/8.8.8.5/*.

Detalle del caso
Una vez inicializados los distintos componentes, y que estén funcionando de forma correcta, se harán mediciones de consumo de CPU y memoria para este esquema de funcionamiento.

Resultados esperados
Se espera tener idea de cómo inciden en uso de recursos disponibles en la máquina en la cual corre cada los componentes, tales como memoria o consumo de CPU.

Resultados obtenidos.
Se verifica que para la cantidad de recursos monitoreados en la prueba, 12, el funcionamiento de los componentes no altera la carga de CPU, si bien tiene requerimientos de memoria propios de el uso de Java. Los picos altos se dan en el momento de la inicialización de los componentes y como no se están ejecutando en la misma máquina física los requerimientos son menores a los del caso 4.

8.8.8.6 Caso de prueba de integración 6

Contexto

Habrá un Agente, un Servidor y un Cliente corriendo en distintas máquinas y sobre distintos sistemas operativos. Se busca evaluar que no haya problemas de comunicación y de funcionamiento entre los componentes debido a estar corriendo en distintas plataformas.

Detalle del caso
Se inicializarán los componentes según la siguiente tabla:

	Agent	Servidor	Client
Caso1	WIN	LINUX	LINUX
Caso2	LINUX	LINUX	WIN
Caso3	LINUX	WIN	WIN
Caso3	WIN	WIN	LINUX

Resultados esperados
El correcto funcionamiento debe ser independiente de la plataforma.

Resultados obtenidos.
Para todos los casos mencionados en la tabla, se verifica que se cumple con todas las funcionalidades de la herramienta y que no se dan problemas de comunicación entre los componentes.
Se utilizaron los archivos de configuración de los casos de pruebas de integración previos.