Integración de herramientas par la Simulación y modelado de redes WAN en el desarrollo de aplicaciones distribuidas

Orlando Micolini
Laboratorio de Arquitectura de Computadoras, FCEFyN
Universidad Nacional de Córdoba
Córdoba, Argentina
lac@efn.uncor.edu

Adriana Damiani
Laboratorio de Arquitectura de Computadoras, FCEFyN
Universidad Nacional de Córdoba
Córdoba, Argentina
adamiani@efn.uncor.edu

Abstract

Actually there is a great development interest and use of distributed system. In those systems the communication network has the most important role in its operation. The complexity of setting up these systems (effort, time, and money) is strongly influenced by different parameters and network configurations. This takes us to propose an instrument for the net simulation of a distributed system.
In this paper we present a multi-platform tool, implemented with a parallel architecture, for the evaluation and testing of the parameters in the net for tuning system performance.
Our purpose constitutes an integration and extension of existing software for the net simulation. This tool allows the messages capture between subsystems, the injection of these messages into the simulator and its recovery to send them to the target subsystem.

Keywords: Testing Distributed Systems, Simulation, Communications and Networks, Parallel Software.

Resumen

Actualmente hay gran interés en desarrollo y uso de sistemas distribuidos. En estos, la red de comunicaciones cumple un rol fundamental para su funcionamiento. La complejidad de la puesta a punto (esfuerzo, tiempo y dinero) de dichos sistema esta fuertemente influenciada por los distintos parámetros y configuraciones de la red. Esto nos lleva a proponer un instrumento para la simulación de la red de un sistema distribuido.
En este trabajo presentamos una herramienta multi-plataforma, implementada con una arquitectura paralela, para la evaluación de la influencia de los parámetros de la red en las prestaciones del sistema. Nuestra propuesta constituye una integración y extensión de software existentes para la simulación de redes. Esta herramienta permite la captura de mensajes entre susbsistemas, la inyección de estos mensajes al simulador y su recuperación para enviarlos al subsistema destino.

Palabras Claves: Prueba de sistemas distribuidos, Simulación, Comunicaciones y Redes, Software Paralelo.

1 INTRODUCCIÓN

El presente trabajo aborda los requerimientos y la implementación de una herramienta para el desarrollo, prueba y validación de aplicaciones en sistemas distribuidos. El comportamiento de sistemas distribuidos es más complejo que los sistemas que se ejecutan en un único nodo, puesto que una simple operación puede involucrar a cientos de nodos y el envío y recepción de miles de mensajes, con el agravante de que los componentes se encuentran distribuidos geográficamente.
Por ejemplo, si queremos determinar el comportamiento de una implementación frente a la perdida de datos y o variaciones de los retardos, se hace difícil realizar y observar el experimento debido a la distribución y a la variabilidad de comportamiento de la red WAN.
Para esto hemos implementado una herramienta, que permita realizar el experimento en una LAN y arroje los resultados del sistema en evaluación para su análisis, donde los tiempos de ejecución son similares a los del sistema real.

1.1 Objetivos

Principalmente se pretende:

- Modelar una red de comunicaciones de datos para aplicaciones distribuidas, flexible en su topología y que permitan recolectar el contenido de los paquetes para su posterior análisis
- El Simulador de red permite que los programas a probar corran sin modificaciones en la implementación ni el ambiente de trabajo. El sistema real puede acoplarse al simulador sin modificaciones.
    - Inyección de Fallos como perdida de información y variación en los retardos, para evaluar las soluciones de sistemas distribuidos

Como objetivos secundarios: el simulador esta paralelizado, permitiendo su implementación en sistemas distribuidos, tanto débilmente acoplados como fuertemente acoplados.

2 EL CONTEXTO

Existen muchas herramientas complementarias para la detección de errores e inyección de errores en sistemas distribuidos.
Algunas de las herramientas mas usadas son: gdb y gprof para bugs de bajo nivel y un nodo, caja negra [1] para sistemas los que no se tiene el código fuente, impresiones y asserts para la detección de errores por análisis de log, model checking [2], herramientas basadas en agentes [3] para el manejo y o recolección de path event e información del sistema.
Ninguna de estas herramientas permite la realizar de la evaluación del sistema distribuido mientras varían los parámetros de la red y sin ningún cambio de configuración o código.

3 REQUERIMIENTOS

Para lograr una arquitectura adecuada, es necesario explicitar los requerimientos. Dichos requerimientos están implícitos en los objetivos [16].
Para la simulación de la red, se deben cumplir con las siguientes condiciones:

1. Implementación de la red por segmentos independientes con el fin de distribuirla en distintos procesadores
2. Mecanismo de comunicación estándar que permita el acoplamiento, tanto a otros segmentos de red, como a los sistemas que se quiere evaluar
3. Debe ser multi-plataforma, puesto que esta herramienta puede requerir múltiples nodos con diferentes SO
4. La herramienta tiene que ejecutarse en la maquina local, adonde se prueba la aplicación sin modificar
5. La herramienta tiene que poder acoplarse a las aplicaciones distribuidas, capturando los mensajes. Esto se realizara por TCP/IP

4 ARQUITECTURA

En la figura 1, los subsistemas 1 y 2 conforman el sistema distribuido a evaluar. Estos se comunican a través de una red, la que puede ser LAN o WAN.
La herramienta propuesta es un sistema que realiza:

- La intercepción de los paquetes que el **Subsistema1** envía al **Subsistema2**
- Inyecta los paquetes en una red simulada
- Recupera los paquetes y los inyecta en el **Subsistema2 (**Figura 1)

De esta forma no se requieren cambios en los subsistemas como recopilación de código y/o cambios de configuración.

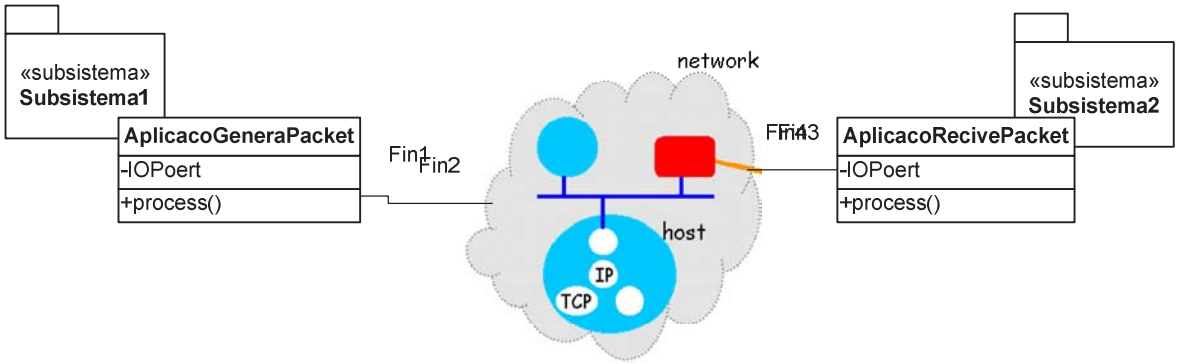

Figura 1

Con respecto a la red simulada, se tiene control de sus paramentos y topología, por lo que es posible extenderla y/o paralelizarla. Esto constituye el aporte fundamental de la herramienta desarrollada.
De la Figura 1 y de los requerimientos se desprende que las partes principales del sistema son:

1. Modulo de captura: captura los mensajes de la aplicación fuente (paquetes) y los envía al simulador
2. Modulo de simulación: es programable según la topología y los parámetros de la red. Es divisible en partes con el fin poder ejecutarlos en distintos procesadores
3. Modulo de transmisión: extrae los mensajes (paquete) del simulador, reconstituye el mensaje y lo envía al subsistema destino
4. Modulo de almacenamiento: almacena la información contenida en los mensajes (paquetes de comunicación) con el fin de recuperarlos para su análisis posterior, y recuperación de los mensajes
5. Modulo de configuración y gestión: permite la configuración del sistema y de la topología de la red, las direcciones de los host y la distribución de los procesos de simulación y captura

Todos estos componentes, mostrados en la figura 2, se tienen que poder ejecutar en múltiples plataformas sin modificación.

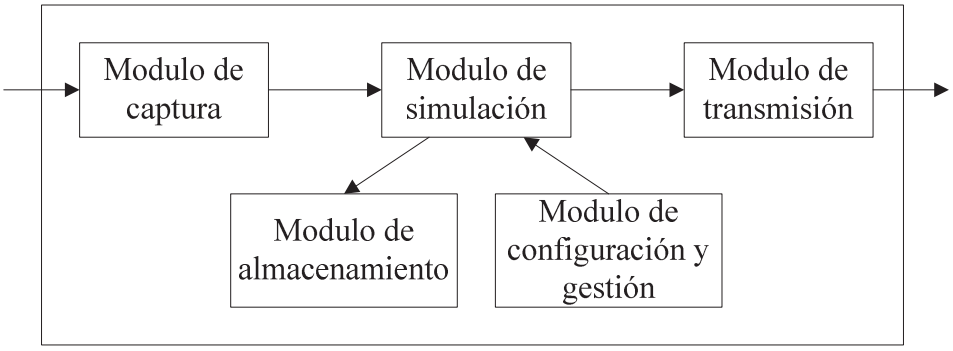

Figura 2: Arquitectura del sistema

5 ANÁLISIS DE HERRAMIENTAS

5.1 Modulo de Simulación

Se realizo una búsqueda de software de simulación orientado a redes, según los requerimientos expuestos. En primer lugar se enumeran algunas de las herramientas mas usadas y probadas, seguidamente se selecciono la que mas se adecue a nuestro proyecto.

Nombre	Network Simulator Tesbed (NEST) [4]
Desarrollado por	Computer Science de la Columbia University
Aplicación	Ambiente de simulación para sistemas de redes distribuidas y protocolos básicos
Implementado	En C y permite a los usuarios ejecutar sus propios códigos en C
Otras características	Interfaz grafica que permite controlar la simulación Basado en una arquitectura cliente/servidor, lo que permite simular complejos escenarios.

Nombre	Maryland Routing Simulator (MaRS) [6]
Desarrollado por	Computer Science de la University of Maryland
Aplicación	Estudio de algoritmos de ruteo en redes WAN
Implementado	En C en una plataforma Unix
Otras características	Es una evolución de un simulador mas antiguo llamado NetSim

Nombre	Realistic And Large Network Simulator (Real) [12]
Desarrollado por	Hani T. Jamjoom de Cornell University
Aplicación	Estudio del comportamiento dinámico de flujos y de control de

	congestión en redes de datos packet switched
Implementado	En C y también se ha desarrollado una interfaz de usuario grafica (GUI) en Java
Otras características	Estudio de sistemas o parámetros que no afecten de forma directa al flujo de conexiones TPC/IP principal. Esto limita la capacidad de modelar un sistema real

Nombre	Network Simulator 2 (ns-2) [7]
Desarrollado por	DARPA, la Universidad del Sur de California y patrocinado por Xerox
Aplicación	
Implementado	En C++, pero para realizar las simulaciones usa un lenguaje interpretado llamado Tcl
Otras características	Es una evolución de REAL network Simulator.Posee capacidades de routing y multicast en redes estructuradas y gíreles. Posee un visualizador llamado Nam, que permite ver en forma simple los resultados de la simulación

Nombre	S3 project, Scalable Simulation Framework [8]
Desarrollado por	Renesys Corporation, el Institute for Security Technology Studies at Dartmouth y DARPA
Aplicación	Permite el uso de prácticamente todos los protocolos en Internet.
Implementado	En C++ y posee dos interfaces de programación, una en Java y otra C++, así como en DML
Otras características	Es altamente escalable pero tiene escaso rendimiento en las versiones gratuitas. La interacción con la simulación solo puede hacerse a través del DML

Nombre	J-Sim Java Simulator [5]
Desarrollado por	NSF, DARPA, CISCO y las Universidades de Illinois y Ohio
Aplicación	
Implementado	En Java, basado en el modelo de programación de componentes autónomos
Otras características	Posee una interfaz de script para la integración con lenguajes de script como Perl, Tcl o Pitón Soporta la mayoría de los protocolos usados en Internet, así como multicast y QoS.

Nombre	Gíreles IP Simulator (WIPSIM) [9]
Desarrollado por	Departament of Communication Technology de la Aalborg University (Dinamarca), en curso
Aplicación	Simulación de redes wireless en IPv6. Su uso principal es en investigación y estudio de mecanismos de control de congestión, escenarios móviles, protocolos y descubrimientos de rutas.
Implementado	En C++.
Otras características	Es fácil escribir protocolos en las capas de transporte, red, link y MAC, y tiene implementaciones para UDP, DiffServ, ISMA, CSMA y Blue Tooth.

El software de simulación elegido fue J-Sim debido a que soporta tanto múltiples plataformas y permite la descomposición para distribuir las cargas de simulación en múltiples procesadores. La implementación de simulador de res se hace por componentes. Esto se logra debido a su arquitectura orientada a componentes [17].

5.2 Modulo de Captura

El objetivo es el diseño e implementación del modulo de acoplamiento de recepción y transmisión de captura de mensajes en el sistema distribuido a evaluar.
La comunicación entre procesos se basa en un Middleware, que esta conformado por diferentes capas. Dichas capas pueden implementar cualquiera de los paradigmas de programación distribuida. En su mayoría están soportados por los protocolos UDP y TCP [13].
Por ejemplo:

	Aplicaciones Servicios	
	RMI y RPC	Capa Middleware
	Protocolo Petición-Respuesta Empaquetado y representación externa de Datos	
	UDP y TCP	

RMI invocación de métodos remotos, permite que un objeto invoque métodos de otro objeto remoto (ej. CORBA y Java RMI).
RPC llamada a procedimientos remotos, permite que un cliente ejecute procedimiento de un servidor remoto.
La capa que da servicio para comunicación Petición-Respuesta entre procesos, valiéndose de TCP, UDP. Los cuales están implementados con sockets tanto en Java como en UNIX.

5.2.1 Características de la Comunicación entre Procesos

El proceso emisor envía un mensaje al proceso receptor, esto implica la comunicación de datos y también puede implicar la sincronización entre procesos.
Las operaciones básicas son *send* y *receive*. Como se puede ver en el esquema de capas, estas están soportadas por protocolos de comunicaciones, principalmente por paquetes TCP y UDP, según sean sincrónicas o asincrónicas.
Lo antedicho implica que nuestro modulo tiene que ser capaz de capturar y filtrar paquetes UDP y TCP para inyectarlos al modulo de simulación. También tiene que poder identificar sus destinos, para rutearlos dentro del simulador y al subsistema de destino. Para esto se deben capturar otros tipos de paquetes como ARP, RARP con el fin de determinar direcciones IP de los distintos [14].
De lo expuesto anteriormente se desprende que el manejo de sockets no es suficiente y se requiere una librería que permita acceder tanto a los datos de los paquetes como a la información de sus encabezados.
Uno de los modulo mas usados y probados para la captura de paquetes es PCap, pero esta desarrollado en C, por lo que debe ser recompilado para distintas plataformas. La versión Java, que nos permite ejecutarlo en múltiples plataformas, es JPcap [10], y es la que mas se adecua a nuestro proyecto, por lo que se eligió el JPcap para el desarrollo del modulo de captura.

5.3 Modulo de Configuración, Almacenamiento y Gestión

El modulo de configuración realiza la persistencia de:

a. La topología de la red, la subdivisión en segmentos y los procesos para ser ejecutados en los distintos hosts
b. Los puertos y sus direcciones para realizar la comunicación con los subsistemas en evaluación
c. Los mensajes contenidos en los paquetes de comunicación para su posterior recuperación

Estos procesos estarán distribuidos entre los hosts que conforman la herramienta y se deben comunicar y sincronizar con el fin de realizar la simulación y captura de datos.
Siendo que tanto los módulos de simulación como de captura están implementados en java, aquí también se realizo la implementación en java.

6 LA SOLUCIÓN QUE SE PROPONE

Para la comunicación y sincronización de los mensajes entre los distintos módulos de la herramienta hay dos alternativas claras que son socket y RMI [11] [15]:
Se decidió hacer la primera la primera aproximación con RMI, con el fin de mantener un diseño de objetos.
En ambas alternativas, hay que serializar [15] los objetos para almacenarlos y transmitirlos entre procesos, estos son los packetes capturados UDP, TCP, ARP y RARP. Lo importante a tener en cuenta

es que los paquetes capturados por j-pcap no son serializables, para lo que se deben discriminar y extraer la información para crear un nuevo objeto paquete serializable. Se procesa dicha información para luego reconstituir el paquete que se envía al proceso destino.

6.1 Implementación del Modulo de Recepción y Transmisión

6.1.1 Modulo de Recepción

En la figura 3 se muestra el diagrama de objetos de este modulo. El objeto CapturePacketNIC implementa PacketReceiver (del J-pcap); el JpcapCaptor captura los paquetes y los manda a este objeto. CapturePacketNIC contiene una referencia a un objeto del tipo ComNICtoRMI. Dicho objeto realiza la discriminación de paquetes y los convierte a objetos serializable para poder hacer la comunicación interproceso por RMI

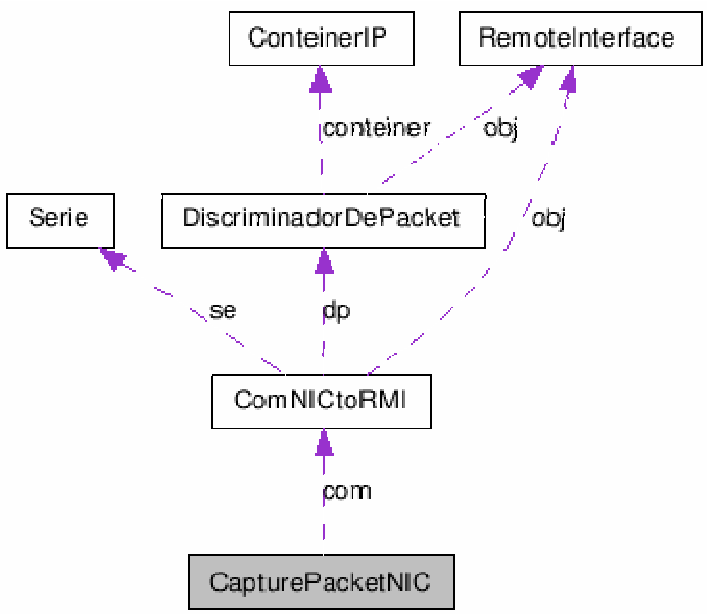

Figura 3: Diagrama de clases del modulo de recepción

El discriminador de paquetes extrae los paquetes ARP para armar las tablas de ruteo y los paquetes UDP/TCP de donde saca la dirección de destino para inyectarlos a la red simulada y al container para su almacenamiento.

La secuencia de flujo de mensajes del modulo de recepción se muestra en la figura 4.

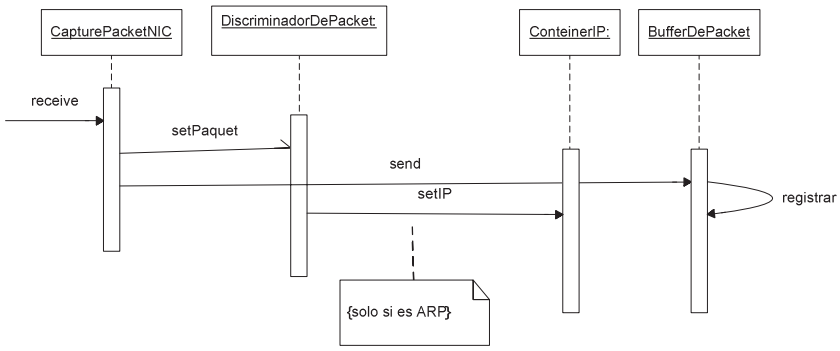

Figura 4: Diagrama de secuencia del modulo de recepción

6.1.2 Modulo de Transmisión

Este modulo es invocado por el simulador cuando un paquete sale del mismo y debe ser inyectado a la red para alcanzar el subsistema de destino.
La clase ContinerIP mantiene una tabla con las relaciones IP destino a NIC del host.
La clase Send se encarga de recuperar el paquete de BufferDePacket (contenedor de los paquetes). En la figura 5 se muestra el diagrama de clases de este modulo.

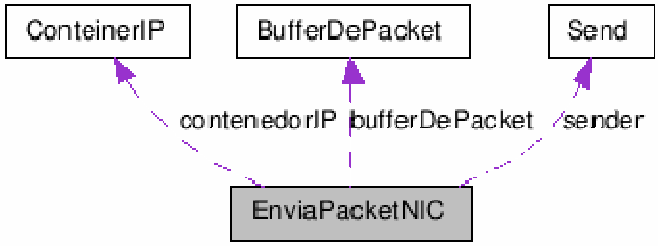

Figura 5: Diagrama de clases de Modulo de Transmisión

6.2 Implementación del Modulo de Simulación

Para el modulo de simulación se utilizo J-Sim. Con el fin de realizar la comunicación se crearon las siguientes clases: GenPacket y RecvPacket. Ambas extienden Component de JSim y poseen puertos de entrada salida para inyectar/recibir paquetes.

En la figura 6 se muestra una configuración elemental del simulador, la cual puede ser escalada en distintas topologías de red.

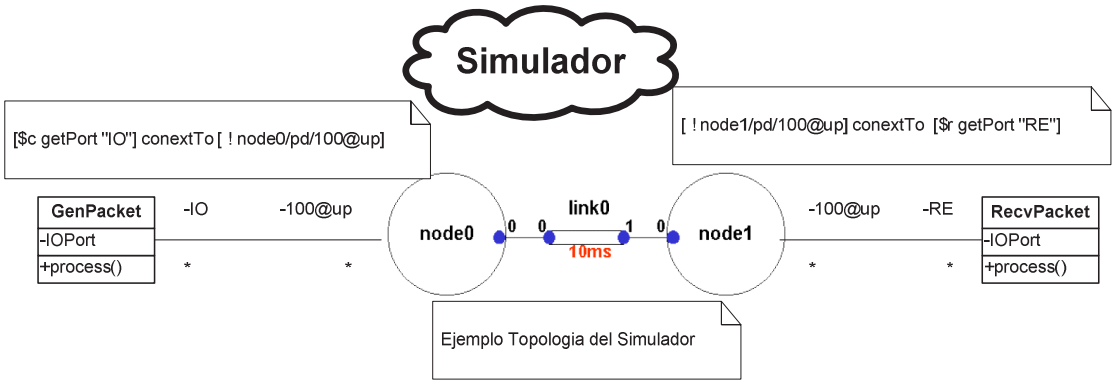

Figura 6: Diagrama de acoplamiento del Simulador

6.3 Implementación del Modulo de Almacenamiento

El objeto BufferDePacket (figura 5) realiza la persistencia de los paquetes que conforman el tráfico del simulador. Cuando el simulador determina que un paquete arribo a su destino, RecvPacket se lo comunica a EnviaPacketNIC, quien lo recupera de BufferDePacket y le coloca una marca de tempo. Para luego enviarlo a un buffer de persistencia definitivo.

7 Conclusión

La solución propuesta se adecua a los requerimientos planteados: multiplataforma, no requiere cambios del sistema a evaluar y permite distintas topologías de red.
El hecho de implementar el simulador por componentes simplifica la paralelización de este modulo.
Una de las principales fortalezas de la herramienta es su escalabilidad. Al dividir y desacoplar la implementación en captura y simulación, es posible hacer uso de un procesador para realizar esta la simulación y que varios procesadores realicen la captura. Esto es recomendable puesto que la captura se debe hacer en tiempo real.

7.1 Mejoras

Realizar un plug-in para herramientas como eclipse con el fin de integrarlo a un framework para así darle al conjunto una mayor funcionalidad.
Implementar una interfase grafica para la configuración y la creación de la topología de la red simulada. Implementar un visualizador grafico para los resultados.

Bibliografía

[1] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed systems of black boxes. In Proc. SOSP, Bolton Landing, NY, Oct. 2003.
[2] P. Godefroid. Software model checking: the VeriSoft approach. FormalMethods in System Design, 26(2):77–101, Mar. 2005.
[3] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed systems of black boxes. In Proc. SOSP, Bolton Landing, NY, Oct. 2003
[4] Sitio oficial de real. http://www.cs.cornell.edu/skeshav/real/overview.html
[5] Sitio oficial de j-sim. http://www.j-sim.org/.
[6] Sitio oficial de MaRS http://www.ccs.neu.edu/home/matta/software.html
[7] Sitio oficial de ns-2. http://www.isi.edu/nsnam/ns/.
[8] Sitio oficial de s3. http://dimacs.rutgers.edu/Projects/Simulations/darpa/.
[9] Sitio oficial de wipsim. http://sourceforge.net/projects/wipsim/.
[10] Sitio oficial de Jpcap. http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html
[11]Sitio oficial de Sun, RMI http://java.sun.com/docs/books/tutorial/rmi/.
[12] Sitio oficial de nctuns. http://nsl10.csie.nctu.edu.tw/.
[13] George Coulouris, Jean Dollimore Y Tim Kindberg, ``Sistemas Distribuidos. Coceptos Y Diseño". 3ª Edición. Pearson Educación, 2001.
[14] W. Richard Stevens, The Protocols TCP/IP Illustrated, Volume 1, Addison-Wesley, 1994
[15] William Grosso, Java RMI, Publisher: O'Reilly, First Edition October 2001
[16] SOMMERVILLE IAN, INGENIERIA DEL SOFTWARE ,PEARSON EDUCACION, Edición 2005
[17] Clemens Szyperski , Component Software, Addison-Wesley Professional, 1997