Laboratorios Simulados de Redes de Datos

Laboratorios Simulados Para Estudiantes De Ingeniería De Sistemas De La Universidad EAFIT

Marcela Vélez Pulgarín
Juan Pablo Vergara Villarraga

Universidad EAFIT
Pregrado De Ingeniería De Sistemas
Medellín
2007

Contenido

Marcela Vélez Pulgarín mvelezpu@eafit.edu.co
Juan Pablo Vergara Villarraga jvergar7@eafit.edu.co

Introducción

En una era digital como la que se vive en este momento el las herramientas de software se han convertido en un apoyo para muchas áreas del conocimiento, entre ellas para la docencia. Dentro de esta área en particular, estas herramientas han sido utilizadas no sólo para enfatizar la teoría sino también para la realización de prácticas de laboratorio, que manualmente son complejas, conllevan riesgos como el daño de equipos por configuraciones erradas o por otras razones relevantes para el docente.

Específicamente en la universidad EAFIT, dentro del pensum del pregrado de Ingeniería de Sistemas encontramos diferentes aplicaciones que son usadas como soporte para el aprendizaje. Entre ellas tenemos Simuproc usada en la materia Programación Digital, REM y Enterprise Architect para las asignaturas de Ingeniería de Software y Taller de Ingeniería de Software, Microsoft Project usada en Gestión de Proyectos Informáticos, Statgraphics para Estadística, etc.

Telemática I y en general su línea de énfasis, no cuentan con una herramienta de este tipo, aún cuando tienen uno de los riesgos más altos por el préstamo y la manipulación constante que los estudiantes realizan sobre los equipos del laboratorio. Estos riesgos se materializan en los laboratorios de dos formas:

- La no comprensión de la totalidad de lo que sucede en una topología real al estar enfocada hacia una tecnología particular.

- El riesgo de desconfigurar gravemente los equipos al ser el primer acercamiento a un equipo de esta categoría.

Este documento, a través de sus diferentes secciones, expone la propuesta que se realiza para enfrentar este problema. En la sección *Planteamiento del problema* y *Justificación* se encuentran los elementos de necesidad, motivación y justificación del producto obtenido, mientras que en el apartado; por su parte, en el apartado *Desarrollo de la propuesta* se encuentra todo lo relacionado con la ejecución de lo que se propone realizar de acuerdo a las motivaciones indicadas en las secciones indicadas inicialmente; el desarrollo de la propuesta tiene en cuenta los dos frentes involucrados en este proyecto de grado, a saber: a) la construcción de un

módulo de enrutamiento dinámico para una plataforma de software libre utilizada para realizar simulaciones de topologías de red y todo lo relacionado con el análisis, diseño e implementación de este módulo desde el punto de vista del desarrollo de software, y b) el análisis del impacto de una herramienta de este tipo si fuera utilizado en las asignaturas de Telemática en la universidad EAFIT.

En las secciones *Conclusiones* y *Trabajo Futuro* se exponen los resultados del trabajo realizado, así como también los posibles trabajos a realizar a partir del desarrollo de nuestra propuesta.

Marcela Vélez Pulgarín **mvelezpu@eafit.edu.co**
Juan Pablo Vergara Villarraga **jvergar7@eafit.edu.co**

2. Planteamiento del problema

El programa de Ingeniería de Sistemas de la universidad EAFIT, tiene como una de sus materias obligatorias para todos los estudiantes del pregrado una asignatura llamada Telemática, en la cual entre muchos otros temas, se cubre la teoría básica de enrutamiento y el diseño básico de redes, componente esencial en la formación básica de un profesional de este programa. En la actualidad este tema se cubre en horario de clases, mediante exposición magistral dictada por el docente a cargo del grupo, exponiendo ejemplos valiéndose de recursos como el tablero, video beam y otros que permitan mostrar de manera introductoria los principales componentes de esta teoría de enrutamiento. Una vez se ha dictado en clase los elementos mínimos necesarios se procede con una práctica de laboratorio en las instalaciones de la universidad, la cual permite mostrar los conceptos enseñados anteriormente utilizando máquinas reales que deben ser configuradas de acuerdo a un enunciado propuesto por el docente.

Si bien el procedimiento actual cumple con las expectativas tanto del docente como de los alumnos, si se examina un poco a fondo el proceso de enseñanza del tema de enrutamiento, se puede observar que existe un gran vacío al momento de dar el paso del salón de clases al laboratorio, ya que cuando el estudiante va a enfrentarse a los enrutadores y conmutadores de la universidad, los conocimientos acerca de enrutamiento están aún en el papel, lo que representa una dificultad adicional al momento de configurar uno de estos equipos para lograr el objetivo planteado en el enunciado de la práctica.

Esta dificultad tiene varias consecuencias al momento de la elaboración de la práctica de laboratorio, a saber:

a) Los estudiantes pueden tardar más tiempo del necesario para lograr los objetivos planteados.
b) Al ser la primera aproximación a un equipo, se corre un riesgo realmente grande de averiarlo.
c) Existe un riesgo latente que podría llevar al estudiante a no comprender los conceptos de enrutamiento, ni la configuración de los equipos.

Lo anterior afecta, no sólo el desarrollo de la práctica, sino también a otros usuarios del laboratorio; pues la universidad cuenta con unos recursos físicos limitados que son utilizados, tanto por los estudiantes que cursan la materia Telemática como por los otros de las diferentes asignaturas que componen la línea de énfasis del mismo nombre, entre las que se encuentran: Redes y Protocolos, TCP/IP y Computación móvil. Esta situación genera una congestión en el laboratorio que afecta directamente a estudiantes y docentes, limitando el número de prácticas posibles a realizar y el tiempo de estadía en los laboratorios por cada grupo.

Los retrasos y riesgos asumidos en las prácticas de laboratorio, como ya se ha mencionado, radican principalmente, en que no se han asimilado los conceptos de enrutamiento en su totalidad y por tanto, la destreza en el manejo de los equipos no se desarrollará en el mismo tiempo que sucedería si el estudiante tuviera una primera aproximación que le permitiera prepararse un poco antes de enfrentarse a las máquinas como tal. Este entrenamiento previo consistiría principalmente en lograr un primer contacto con los dispositivos necesarios en la topología y configuración de una red, sin encontrar en los comandos de los equipos o en la falta de recursos una dificultad que no le permita progresar en su proceso de aprendizaje.

Una vez identificada esta situación, se encontró que la necesidad de una primera aproximación a las topologías de red y a casos de estudio que no se realizaran sobre equipos reales podría ser suplida por una herramienta de software, que permitiera interactuar con simulaciones de equipos como los que se tienen que manejar en el laboratorio y que eliminara los riesgos citados en este mismo apartado. Durante la revisión del estado del arte se determinó que existen herramientas reconocidas en el mercado pero que no cumplen las condiciones y los planteamientos requeridos para el desarrollo que se busca, pues algunas tienen un costo de licenciamiento o poseen un enfoque diferente al educativo, como es el caso de OPNet y PacketTracer; sin embargo se logró localizar una herramienta que tiene desarrolladas en gran parte las características descritas para el estudio de redes básico y que, adicionalmente, es un software libre que permite la distribución y modificación de sus códigos fuentes, esta aplicación lleva el nombre de KivaNS y fue desarrollada por estudiantes de la Universidad de Alicante, España.

Complementar el código de KivaNS e implantar la aplicación dentro del laboratorio ofrecería un gran apoyo a los docentes del área de Telemática de

la universidad EAFIT, permitiendo de esta manera lograr un afianzamiento de los conceptos de enrutamiento en primera instancia y apoyando la labor de enseñanza de configuración de equipos reales.

2.1 Formulación del problema de investigación

¿Cuál sería el impacto que generaría para los estudiantes de la universidad la implantación de una herramienta de software que permita simular topologías de red y estudiar el proceso de enrutamiento entre los dispositivos utilizando protocolo de enrutamiento RIP v1 que la conforman?

3. Antecedentes

3.1 Estado del Arte

Se realizó una investigación sobre diferentes herramientas de simulación disponibles en el medio y cuyo objetivo fuera el análisis o estudio de las topologías de red y los diferentes dispositivos que lo componen, así como la configuración de equipos y los diferentes eventos que ocurren en una simulación.

Dentro de las principales herramientas encontradas se analizaron las siguientes por ser conocidas en el área de redes o respaldadas por empresas reconocidas en este medio. A continuación se describen las principales características, ventajas y desventajas de las aplicaciones analizadas:

- Packet Tracer

  Es un simulador gráfico que permite la configuración y estudio de topologías de red y dispositivos CISCO; su principal foco es el entrenamiento para obtener la certificación CCNA[1].

  Actualmente soporta tecnologías como Ethernet, FastEthernet, Gigabit Ethernet, VLAN, NAT, PAT y protocolos como RIP, DHCP, STP limitado.

  El usuario puede construir sus propias redes y analizar la simulación de la red, así como crear y programar el envío de paquetes. La interfaz gráfica

[1] CCNA - Cisco Certified Network Associate

permite al usuario arrastrar los dispositivos al escenario de la topología y configurarlos con los diferentes protocolos soportados, la figura 3.1 muestra un ejemplo de las topologías que se pueden realizar.

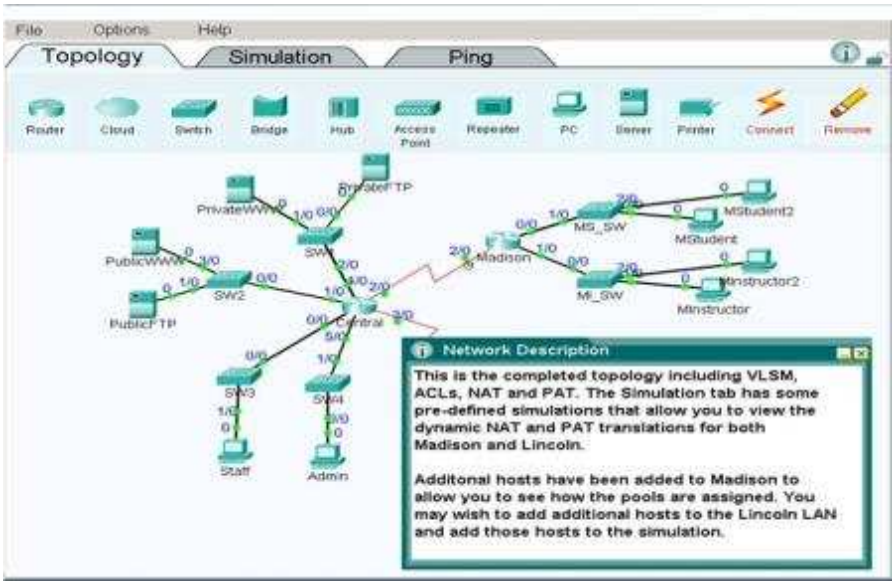


Figura 3.1. Topología de red, interfaz gráfica Packet Tracer

Adicional a la ventana de topología, posee una ventana de simulación que permite hacer un seguimiento paso a paso de la transferencia de los datos a través de dispositivos, explicando visualmente el flujo que se presenta en la red.

Trabaja con Windows como sistema operativo y no tiene requisitos especiales de CPU o disco. Para poder obtenerlo debe ser bajado desde la página de Academy Connection a la que tienen acceso los estudiantes de los cursos para la certificación CCNA.

- Ventajas
  - Posee una interfaz gráfica completa y fácil de manejar, con una documentación completa para su manejo.
  - Permite analizar el desarrollo de la simulación, analizando la transferencia de los datos en los diferentes niveles del modelo OSI.
- Desventajas
  - Es software propietario de CISCO, por lo que requiere autorización para descargarse. Usado para estudiantes o instructores de CISCO.

        - No permite la creación de topologías que involucren tecnologías diferentes a Ethernet.
        - Enfocado a dispositivos CISCO, incluye protocolos particulares de esta marca.

- OPNET Modeler

Es una herramienta que permite el diseño y estudio de las redes de comunicación, dispositivos, protocolos y aplicaciones con una gran escalabilidad y flexibilidad.

Cuenta con cientos de protocolos y dispositivos de diferentes vendedores. Permite al usuario el estudio del comportamiento de prototipos de aplicaciones generándolas dentro de una topología de red simulada.

Es una aplicación bastante flexible en términos de diseño de topologías, creación de protocolos y pruebas a los mismos, incluye varías librerías de modelos, y el código fuente de estas librerías puede ser modificado para realizar pruebas de nuevos protocolos.

    - Ventajas
        - Soporta tecnologías bases y nuevas, cuenta entre otros con: VoIP, TCP, OSPFv3, MPLS, IPv6.
        - Se facilita la modificación del código fuente para la programación de nuevos protocolos con los API's expuestos.
        - Útil para el análisis de comportamiento y desempeño en redes de gran tamaño.
    - Desventajas
        - Herramienta propietaria de OPNET, requiere licencia para su uso.
        - No enfocada al estudio básico de redes.

- BOSON NetSim

La tecnología virtual de BOSON genera paquetes individuales que son enrutados y conmutados a través de la red simulada, permitiendo a la aplicación construir una tabla virtual de rutas apropiada y así simular una red de comunicaciones.

La herramienta esta enfocada para obtener certificación CISCO CCNA y CCNP más que herramienta de aprendizaje.

Incluye laboratorios y exámenes cercanos a la preparación de un curso de CISCO.

- Ventajas
  - Soporta diferentes tipos de dispositivos físicos que permiten una mayor compresión.
  - Cuenta con laboratorios y exámenes que permiten poner a prueba el conocimiento.
- Desventajas
  - Herramienta propietaria de BOSON, requiere licencia para su uso y tiene un costo elevado.
  - No enfocada al estudio básico de redes en la academia, sino en la certificación de CISCO.

- KivaNS

Es una herramienta principalmente enfocada a simular el comportamiento del protocolo IP. Al utilizarlo también puede estudiarse el comportamiento de protocolos auxiliares como ICMP, ARP y de redes Ethernet.

El objetivo principal del entorno es ayudar a diseñar y comprender el funcionamiento de redes de datos, y en especial el encaminamiento de paquetes en la arquitectura TCP/IP, sin necesidad de una infraestructura real y de herramientas de análisis de tráfico. KivaNS también es capaz de simular distintos tipos de errores en el funcionamiento de las redes, como la pérdida de paquetes o fallos en tablas de encaminamiento. La figura 3.2 muestra una topología generada con KivaNS.

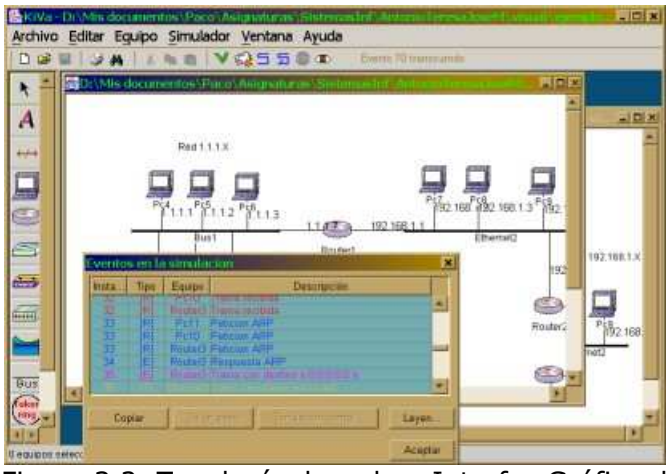


Figura 3.2. Topología de red en Interfaz Gráfica de KivaNS
Fuente: http://www.disclab.ua.es/kiva/

Posee una API de simulación totalmente desligada de la parte gráfica, permitiendo que el programador genere una nueva forma de interpretar los resultados de la simulación de acuerdo a su conveniencia.

Por estar desarrollado en Java es una aplicación multiplataforma y puede ser usado en sistemas operativos Windows y Linux. Sin embargo, para la instalación de la interfaz gráfica se requiere los archivos ejecutables que incluyen el jar[2] de la API de simulación.

- Ventajas
  - Su orientación académica hace que sirva de ayuda para el diseño y comprensión del funcionamiento de redes de datos.
  - Permite analizar los eventos ocurridos durante una simulación, para lograr su análisis y comprensión.
  - Por ser un software libre permite la modificación del código fuente para la inclusión de nuevos elementos.
- Desventajas
  - Para investigación o desarrollo de nuevos prototipos el usuario debe realizar programación en JAVA.
  - La interfaz gráfica requiere la ejecución del conjunto de clases bajo la cual esta implementada.

[2] JAR Formato utilizado para reunir los componentes que requiere una aplicación hecha en lenguaje de programación JAVA.

3.2 Estado de KivaNS

Previo al inicio de la modificación de KivaNS, según lo especificado como objetivos del trabajo, se realizó un análisis del estado de la aplicación con el fin de analizar la modularidad y cambios puntuales que se debían realizar para la implementación de RIP y los scripts de configuración. KivaNS se compone de dos partes, enteramente implementadas en JAVA: la API [3] de simulación y la interfaz gráfica relacionadas en la figura 3.3.

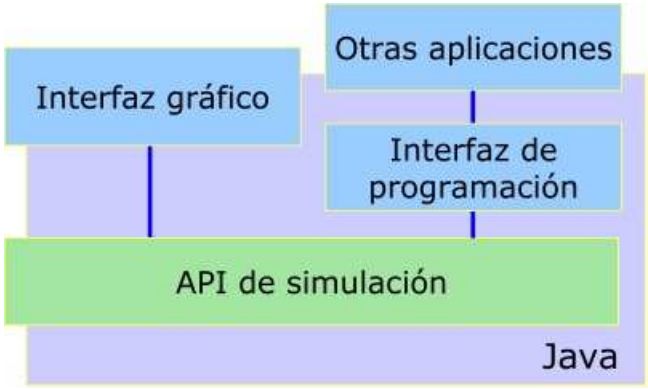


Figura 3.3. Arquitectura general de KivaNS
Fuente: http://www.disclab.ua.es/kiva/

Esta es una arquitectura altamente desacoplada debido a que sus componentes presentan bajas dependencias. De este modo la API de simulación puede ser usada por otras capas desarrolladas que permitan exponer su funcionalidad a cualquier plataforma, un ejemplo podría ser utilizar Web Services basado en la arquitectura SOA[4].

- **API Simulador**

  La API de simulación esta compuesta por cuatro bloques, tal como lo muestra la figura 3.4, claramente definidos de acuerdo a su función dentro del sistema.

[3] API Aplicación Program Interface. Permite utilizar las funcionalidades de un programa a través de su interface.
[4] SOA (Service Oriented Architecture). No sólo usada a través de Web services sino de otras tecnologías que permitan exponer los servicios para ser utlizadas por diferentes aplicaciones independiente de la plataforma. Para más información consulte en el glosario del documento.

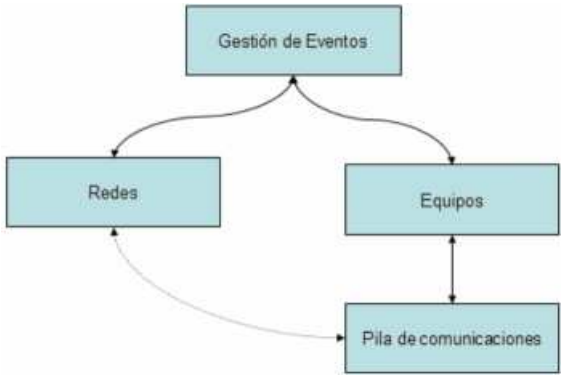


Figura 3.4. Esquema de bloques del sistema de Simulación
Fuente: http://www.disclab.ua.es/kiva/

La gestión de eventos se encarga de coordinar los estados de los equipos y las redes que conforman la topología, llevando un control del tiempo en instantes[5] que le indican al componente que debe avanzar de acuerdo a los eventos programados.

Cada evento controlado por este sistema contiene información la siguiente información del instante en que se produjo, el paquete que generó el evento, la descripción de lo ocurrido y un indicador para determinar si el mensaje entraba, salía o provocó un error en el componente (red o equipo). Dependiendo del evento ocurrido éste podrá ser registrado para que el usuario pueda verlo una vez finalizada la simulación. Esta visualización no esta ligada al componente gráfico pues este sistema lo almacena y puede ser leído por cualquier componente capaz de interpretarlo.

Las redes son encargadas de hacer fluir los datos a través de los equipos. Simula el comportamiento de las redes reales, cuando a una red le llega un paquete de datos, éste es analizado y encaminado de acuerdo a su información. En la versión actual de KivaNS sólo se soportan redes Ethernet.

Los equipos son los encargados de la evolución del sistema, puesto que de su comportamiento y de los paquetes de datos generados depende el desarrollo de la simulación. Adicionalmente los equipos registran los eventos que se presentan dentro de la pila de comunicaciones que tienen.

[5] El control del tiempo se realiza por medio de instantes para el sistema, es decir, la diferencia entre t+1 y t es un instante. Esto no esta supeditado a una medida de tiempo en particular.

Actualmente KivaNS tiene definido dos equipos con estas características que son Ordenadores y Enrutadores.

La pila de comunicaciones es el núcleo del sistema. Este componente esta definido por la arquitectura de redes como las capas y niveles que permiten la transmisión e interpretación de datos a través de una red. Esta formada por diferentes módulos de acuerdo con el equipo en el que se incluya.

Este sistema es el encargado de transmitir las PDU de los niveles de enlace y red, generando los paquetes de datos a ser enviados y procesándolos como corresponda. La pila esta dividida en módulos o niveles independientes en implementación pero no en uso.

Cada módulo posee:

- Una cola de entrada y una de salida de datos.
- Una lista de los niveles inferiores y una de los niveles superiores con los que cuenta.
- Una lista de errores que se pueden simular.
- Una lista de identificadores de niveles basada en los localizadores con los que cuenta el sistema y que serán mostrados más adelante.
- Un enlace al equipo propietario de la pila.
- Lista de propiedades del módulo (Retardos, timeout, etc).

Actualmente esta pila de comunicaciones tiene incluidos un Módulo ARP, un Módulo ICMP y un Módulo IPv4.

Para lograr la modularidad y extensibilidad objeto de la aplicación se aprovecharon las cargas dinámicas de Java, permitiendo el registro de nuevos equipos o redes en unos Localizadores dispuestos para tal fin.

De este modo se eliminan las dependencias de la pila de comunicaciones, por ejemplo cuando se requiere una red o un equipo el localizador se encarga de buscar las clases y cargarlas en tiempo de ejecución, en caso de no encontrarlas el sistema no se detiene y simplemente no son usadas.

Los autores de KivaNS tienen disponible los esqueletos del código fuente para realizar nuevos equipos, interfaces, direcciones, niveles, redes y tipos de trama, facilitando la modificación e inclusión de nuevos componentes a la aplicación.

- **Interfaz Gráfica**

En la versión actual la interfaz de Kiva permite especificar las topologías de redes de datos mediante un editor gráfico, configurando las características de direccionamiento y encaminamiento de los equipos de la red mediante cuadros de diálogo de fácil manejo. En la figura 3.5 se puede observar el estado de la interfaz de una topología de red generada.

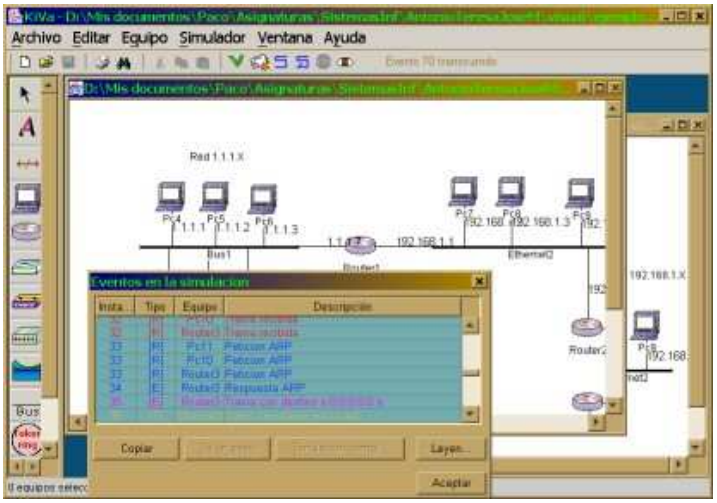


Figura 3.5. Topología de red en Interfaz Gráfica de KivaNS
Fuente: http://www.disclab.ua.es/kiva/

Esta interfaz conserva las características principales de la API de Simulación, permitiendo al usuario acceder a estas sin la necesidad de programar o interpretar el código fuente.

Los diálogos de configuración de cada equipo incluyen:

a. Interfaces. Lista de las interfaces con las que cuenta el equipo que se esta configurando, aquí el estudiante debe incluir las direcciones IP, la máscara, el nombre de la interfaz y la red a la que esta conectada.

b. Rutas. Lista de las rutas que tiene un equipo, simula una tabla de rutas de dispositivos físicos reales. Cada vez que se crea una interfaz se genera una ruta por defecto a esa red directamente conectada.
   Actualmente este diálogo no especifica si la ruta es directa o aprendida por medio de un protocolo, porque todas las rutas son estáticas.
c. Conexiones. Lista de las conexiones con las que cuenta un equipo en un momento dado. Es independiente de que tenga o no configurada la interfaz para dicha red.
d. Errores a Simular. Permite al usuario determinar los errores que le interesa tener en cuenta durante la simulación, de este modo se puede realizar una depuración a la red. Entre los errores a simular se encuentran: No enviar un fragmento de un paquete IP, ignorar las peticiones ARP recibidas, no enviar respuestas ARP, etc.
e. Eventos. Después de una simulación este panel muestra los eventos transcurridos para el equipo seleccionado. Indica los datos mencionados anteriormente para los eventos, como son: instante en que ocurrió, tipo y descripción del evento.

La interfaz gráfica que provee KivaNS, permite comprobar la topología creada para validar que los elementos estén correctamente conectados y que tengan sus respectivas interfaces. Adicionalmente, es posible programar envíos entre los equipos, indicando el dispositivo destino, el de origen, el tamaño, el número de copias e incluso si el paquete IP puede ser fragmentado o no.

La aplicación fue concebida desde el inicio con el objetivo de ser modificada fácilmente para soportar nuevas características, esto se ve reflejado no sólo por ser software libre sino también por la claridad de la documentación, el código y la estructura de sus dos componentes principales. Gracias a esto KivaNS es adaptable de acuerdo a los objetivos que se plantearon al inicio de este trabajo. Adicionalmente, tiene un enfoque académico para el aprendizaje y práctica de topologías básicas de red y su funcionamiento.

A partir de este punto, inicia el proceso de análisis de la modularidad descrita anteriormente para concretar la adaptación que se debe realizar dentro de la aplicación. Clarificando así, el proceso de diseño, construcción y pruebas de la propuesta presentada.

4. Justificación

Existen herramientas en el medio que suplen estas necesidades tales como BOSON NetSim, OPNET, PACKET TRACER, y demás revisadas en el apartado *Estado del Arte*; éstas son aplicaciones propietarias o que se pueden usar legalmente comprando su respectiva licencia, lo cual las descarta a la hora de tenerlas en cuenta como posibles soluciones al problema.

KivaNS por otro lado es un software libre que permite la distribución y modificación de su código fuente, lo que facilita la adición de módulos según los objetivos del proyecto. Desde su inicio la herramienta fue concebida como apoyo para el estudio básico de redes, por lo que presenta los conceptos básicos y una excelente guía para el usuario final.

Después de analizar el código fuente y determinar la viabilidad de modificarlo desarrollándole dos módulos adicionales requeridos dentro de los objetivos del proyecto, se determinó que:

- La aplicación constituye una herramienta de apoyo para la enseñanza de la teoría de enrutamiento en los cursos de Telemática I y TCP/IP principalmente.
- Evita el riesgo de daños en los equipos reales en los laboratorios debido a la inexperiencia de los alumnos; los acercaría en primera instancia arrojándoles como resultado los scripts de configuración de los equipos. Esto disminuirá notablemente los costos de mantenimiento y adquisición de nuevos equipos a causa del maltrato.
- Reduciría la congestión que se presenta actualmente en los laboratorios en grupos grandes, ya que todos los alumnos desean probar de manera práctica sus conocimientos y los recursos no son suficientes para todo el grupo en un mismo momento.

Una vez se desarrolle e implante esta herramienta el equipo de docentes del área se vería beneficiado ya que contaría con un gran apoyo al momento de realizar tanto labores de enseñanza, de enrutamiento como labores de dirección de práctica de laboratorio.

5. Alcance

Este proyecto se desarrolló partiendo, como ya se ha mencionado, de una API existente y que tiene por nombre KivaNS, la cual cuenta con una interfaz gráfica y algunos módulos ya desarrollados sobre dicha API.

En este orden de ideas los entregables de este proyecto constan tanto de lo que ya está desarrollado en el proyecto, como los componentes nuevos objeto de este trabajo. Ambos descritos a continuación.

- Implementación de RIP V1 sobre la API y la interfaz gráfica de KivaNS. Esta implementación incluye para el control de bucles en la red, el método "triggered updates" que envía una actualización por fuera del tiempo especificado si una métrica es puesta a 16.
- Implementación de un módulo que simule el comportamiento del protocolo UDP sobre el cual pueda trabajar el módulo RIP V1.
- Actualización de la tabla de rutas en la interfaz gráfica que permita al usuario identificar los pasos de convergencia de la red.
- Implementación del envío de mensajes RIP, dado que KivaNS simula eventos discretos la actualización no se hace cada treinta segundos sino con cada simulación realizada por el usuario.
- Para los mensajes de petición RIP quedarán implementados los esqueletos en la API de simulación pero queda pendiente definir y realizar la implementación de funcionalidad de los mismos. Esto porque se determinó que no son necesarios para la convergencia de la red, son usados en condiciones particulares y no están dentro del alcance de este proyecto.
- Generación de los scripts de configuración para enrutadores CISCO. Cada equipo podrá guardar su configuración, como KivaNS no posee enlaces punto a punto, el usuario deberá especificar el tipo de interfaz y la rata del reloj en caso de ser una interfaz DCE.
- Análisis del impacto de implantación en la universidad. Este impacto será realizado con una muestra de los estudiantes de Telemática del semestre 2007-1, con base en el resultado de estas encuestas se sacaron las conclusiones para este tema.

Como entregables para la universidad se cuenta con:

- Código fuente de la API y de la Interfaz gráfica.
- Manual de Usuario
- Manual Técnico
- Documento final sobre el proyecto realizado.
- Documento de Arquitectura
- Documento de Requisitos del Sistema
- Guía de laboratorio para prácticas con KivaNS

6. Objetivos

6.1 Objetivo General

Adecuar e implantar KivaNS y analizar el impacto de la herramienta como apoyo para el aprendizaje de configuración, diseño y estudio de redes de datos ofrecido en las asignaturas por el área de Telemática de la Universidad EAFIT.

6.2 Objetivos Específicos

- Adicionar a KivaNS un módulo para el manejo de enrutamiento dinámico a través del protocolo RIP.
- Adicionar un módulo a KivaNS que permita generar los scripts de los enrutadores configurados con los comandos de dispositivos CISCO.
- Implantar KivaNS en el laboratorio de Telemática (Aula 18-412) de la Universidad EAFIT.
- Analizar el impacto que la implantación de KivaNS tuvo en el proceso de Generación de Topologías de Red en el departamento de Telemática como parte del proceso pedagógico.

7. Metodología

Metodología

La metodología seguida para la ejecución de este proyecto comprende dos frentes principales, a saber: el componente de desarrollo de software y el componente de análisis de impacto de la herramienta en la enseñanza y aprendizaje de teoría de enrutamiento dinámico en la Universidad EAFIT.

Desarrollo de software

Para el proceso de desarrollo de los protocolos UDP y RIP v1 de la plataforma KivaNS se siguió la metodología de desarrollo RUP[6]. Esta metodología es una implementación de desarrollo en espiral, el cual es un modelo de ciclo de vida de proyectos generalmente de desarrollo de software. Las actividades de este modelo no están fijadas a priori, sino que las siguientes se eligen en función del análisis del riesgo, comenzando por el bucle interior.

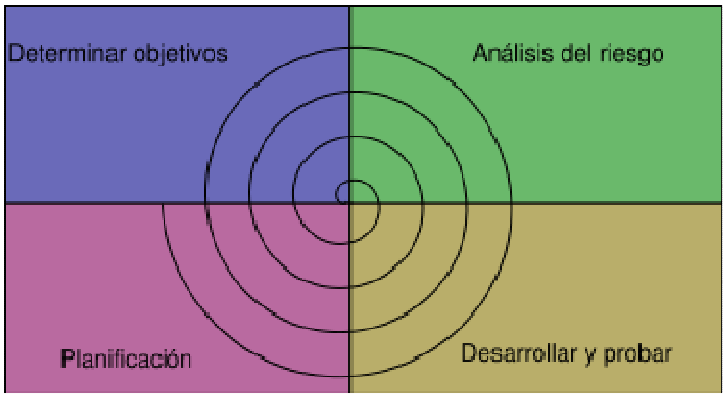


Figura: Actividades del modelo en espiral.

[6] Racional Unified Process, metodología estándar para el análisis, implementación y documentación de sistemas orientados a objetos.

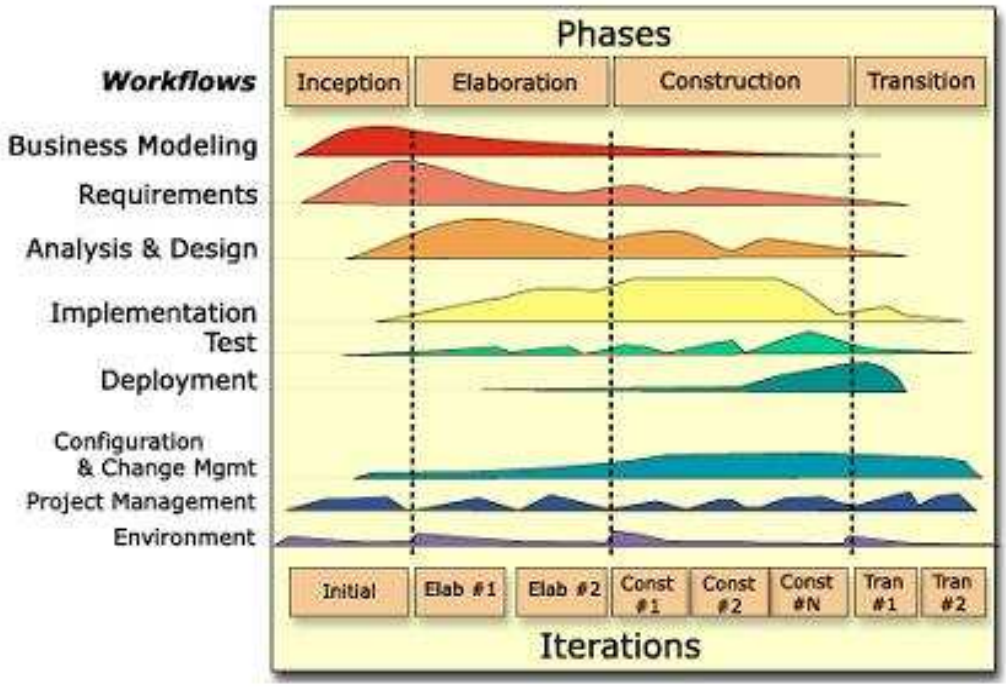


Figura: Fases de RUP
Fuente: Basado en [17]

La figura anterior muestra las diferentes fases de RUP y los diferentes *flujos de trabajo* exponiendo la relación entre ambos.

Para la ejecución de este proyecto, se tuvieron en cuenta los cinco primeros flujos de trabajo, pues estos suplieron todas las necesidades de la implementación completa de dos nuevos protocolos para KivaNS. A continuación se exponen las conclusiones obtenidas en cada una de las fases ejecutadas del *RUP.*

Concepción

Modelado de negocio

En este flujo de trabajo se obtuvo un enfoque global de la necesidad de crear una herramienta libre[7] no orientada a una tecnología particular que

[7] Software libre bajo licencia GPL

permitiera simular el enrutamiento de paquetes en redes de datos y orientada a la enseñanza y aprendizaje particularmente de teoría de enrutamiento dinámico. El primer paso fue entonces ubicar en las herramientas existentes en el mercado para este fin y como resultado final se obtuvo a KivaNS como la herramienta que suplía estas necesidades pero que además podía ser complementada con nuevos módulos. De esta manera se encuentra una base para el trabajo que se desea realizar y como esta suple las necesidades *de negocio* inicialmente planteadas.

La documentación sobre el negocio puede encontrarse en el documento de Arquitectura, adjunto a este trabajo.

Requisitos

Una vez analizado KivaNS a partir de la documentación proveída en el sitio Web oficial del simulador, se pudo determinar cuáles eran los requisitos funcionales y no funcionales de la implementación deseada. Dado que la estructura del simulador está bien definida y define estándares de comportamientos y dado que nuestra implementación se debía ajustar a estos estándares, a partir del descubrimiento de KivaNS como simulador, los requisitos se obtuvieron de manera clara, sin necesidad de elicitarlos demasiado. Para mayor información acerca de los requisitos la ejecución de este proyecto, refiérase al documento de requisitos, el cual es entregado junto a este documento.

Elaboración

Análisis y diseño

En esta etapa del proceso se concentraron las decisiones más importantes del proyecto con respecto al desarrollo de los nuevos componentes de software. Una vez identificados los requisitos funcionales a implementar como componentes nuevos del simulador, se debieron tomar decisiones de arquitectura que se acoplaran perfectamente al diseño y arquitectura propios de KivaNS. Dado que KivaNS es una plataforma bien diseñada y modular desde su concepción inicial, las decisiones de diseño y arquitectura tomadas para la implementación de dos nuevos protocolos y de los demás componentes necesarios para su construcción, no tuvieron incisión alguna en

la API de programación original; de tal modo que los componentes nuevos continúan con el estándar establecido de ser independientes en su implementación mas no en su uso, ya que en algún punto, los niveles nuevos y los ya desarrollados se tienen que relacionar.

Construcción

Implementación y pruebas

La implementación consistió, tal como lo dicta RUP, en la construcción del código fuente que lleva a cabo las características definidas en los requisitos tanto funcionales como no funcionales, siguiendo los lineamientos definidos en la etapa de análisis y diseño.

La construcción de los módulos UDP y RIP v1, los datagramas propios de estos niveles y las clases auxiliares fueron desarrolladas y probadas de manera iterativa, de tal modo que si se desarrollaba una funcionalidad, por sencilla que fuera, se realizaban sus respectivas pruebas unitarias, con el fin de garantizar la calidad de cada componente desarrollado.

Una vez concluye el proceso de construcción y pruebas por parte de los desarrolladores, se inicia la etapa de pruebas documentadas las cuales tienen como objetivo probar el simulador en los casos de pruebas descritos por el plan de pruebas diseñado para el caso. Este plan de pruebas diseña una matriz de situaciones normales y extremas bajo las cuales el simulador KivaNS debe responder adecuadamente, arrojando resultados positivos para poder aprobar cada prueba. La documentación completa de estos casos se puede referenciar en el Anexo IV de este documento.

Transición

Distribución

Dado que KivaNS es una aplicación de escritorio que corre localmente, la distribución no es un proceso tedioso ni complejo. Esta fase se llevó a cabo instalando los componentes necesarios en las máquinas de la universidad en las cuales se efectuarían las pruebas en grupos de estudio objetivo. Tanto en

el manual de usuario como en la documentación de arquitectura y diseño de la aplicación se expone los requerimientos mínimos que debe cumplir la máquina sobre la cual se ejecutará el simulador, no siendo esto un proceso difícil.

Además de la instalación y puesta en marcha del simulador KivaNS con sus nuevas funcionalidades en el laboratorio de Telemática de la Universidad EAFIT, esta etapa de transición también tiene en cuenta soporte y mantenimiento de la aplicación; esta etapa se efectuará en tanto los estudiantes o el público usuario de KivaNS reporte errores que se hayan pasado por alto en la etapa de *testing* de la aplicación.

Es importante anotar que dado que KivaNS es una aplicación de código libre, no es la Universidad EAFIT por medio de los dos desarrolladores de este proyecto quien efectuará el mantenimiento de esta aplicación, sino cualquier posible desarrollador interesado en el proyecto que posea el conocimiento necesario para efectuar la corrección correspondiente.

Análisis de impacto

Para realizar el análisis del impacto del uso de KivaNS en un ambiente académico real, se encuestó a una muestra de la población que atendía al curso de Telemática I, en el cual actualmente se dicta el tema de enrutamiento dinámico básico y para el cual se realizaron las modificaciones ya explicadas al simulador de redes IP KivaNS.

El primer paso entonces consiste en el diseño de las encuestas, de tal modo que puedan obtenerse resultados concretos de las respuestas dadas por lo estudiantes a estas preguntas. Las encuestas se pueden observar en el Anexo II de este documento.

Una vez diseñadas las encuestas y revisadas por el equipo de trabajo, se realiza un laboratorio en las instalaciones físicas de la universidad, concretamente en el laboratorio de Telemática de la Universidad EAFIT. Este laboratorio tiene como objetivo montar una topología de red en KivaNS e invitar a los estudiantes a observar como converge la red mediante el módulo RIPv1 implementado para este proyecto de grado; una vez terminada la sesión práctica se procede a solicitar a los estudiantes que

respondan las encuestas y de este modo tener un insumo de análisis para poder obtener las conclusiones pertinentes.

Finalmente, una vez se tienen las encuestas diligenciadas por el grupo de estudiantes seleccionados para realizar este laboratorio, se procede con un análisis estadístico de las respuestas dadas. Básicamente los estadísticos a obtener son la media y la desviación estándar de cada pregunta y a partir de estos valores poder obtener una conclusión única que responda a la pregunta: ¿Cuál sería el impacto que generaría la implantación de una herramienta de software que permita simular topologías de red y el proceso de enrutamiento entre los dispositivos que la conforman?

8. Cronograma de Trabajo

Referirse al Anexo III.

9. Marco Teórico

Con este marco teórico se busca contextualizar al lector presentando los conceptos más importantes que se definen a lo largo de este proyecto y que garantizan el correcto entendimiento del trabajo expuesto.

9.1 Simulación

Para analizar las herramientas de simulación que se encuentran disponibles en el medio, primero se debe definir que se considera una herramienta de simulación. Se define simulación como "El proceso de diseñar un modelo de un sistema real y llevar a término experiencias con el mismo, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema"[8].

La importancia de los modelos de simulación radica en la seguridad y el costo que ofrecen. En contraste, los sistemas reales pueden ser más costosos, tener riesgos presentes e incluso contar con la limitación de requerir un tiempo considerable para llevarlo a cabo.

[8] Definición de R.E. Shannon.

Tipos de simulación

1. Eventos discretos
   Se refiere a la simulación en la que entre un evento y otro no pasa nada, es decir, el número de eventos es finito y la ocurrencia de los eventos ocurre en espacios de tiempo. Entiéndase evento como un suceso en el tiempo que puede cambiar el estado del sistema.

2. Simulación continua
   Definida para sistemas que cambian constantemente en el tiempo, no sólo en el momento de ocurrencia de un evento particular. Este tipo de sistemas es conveniente simularlo a través de simulación continua, aunque los eventos discretos pueden presentar una aproximación al resultado.

Una herramienta o software de simulación permite:

- Utilización por parte del profesor para ilustrar un procedimiento o proceso concreto.
- Utilización por parte del alumno sin guía del profesor, para tratar de descubrir cómo afectan distintas variables a un procedimiento o proceso.
- Utilización supervisada o guiada por el profesor, con el fin de que el alumno adquiera el suficiente dominio y comprensión de procedimientos y procesos.

9.2 Emulación

A diferencia de la simulación, que pretende imitar el comportamiento de un sistema, la emulación busca modelar de forma precisa el sistema tanto en su parte interna como en la externa; incluyendo su modelo y funcionamiento interno.

Existen tres esquemas básicos que pueden ser usados, incluso juntos, para emular:

1. Interpretación
   Dentro de este esquema el emulador lee instrucción a instrucción desde la memoria, pasando cada una de ellas por las etapas de

captura, decodificación, búsqueda y ejecución, finalmente se realiza la actualización interna (registros y memoria).
La emulación de este tipo es lenta, pero el tiempo de desarrollo es menor.

2. Recompilación
   En esta idea el emulador no intenta simular el comportamiento interno sino traducir el código emulado a código de la máquina.
   La mayor complejidad de esta técnica consiste en la traducción ya que el código original se encuentra en lenguaje ensamblador que puede ser diferente al lenguaje del ensamblador de la máquina en la que se ejecuta.
   - Estática

     En esta técnica la recompilación se hace previa a la ejecución del programa. Si el código cambia durante la ejecución no sería posible usarla.
   - Dinámica

     Diseñada para los casos en que el código está en constante cambio durante la ejecución del programa.

9.3 Arquitectura TCP/IP

Es una arquitectura de red actualmente usada en la gran mayoría de equipos conectados a Internet. TCP/IP posee una gran ventaja sobre las demás arquitecturas puesto que es compatible con cualquier sistema operativo y con cualquier tipo de hardware. No existe un RFC[9] que defina toda la arquitectura, en el ***Anexo I*** se pueden observar los RFC que la componen.

Aunque su nombre viene de los protocolos TCP (Transmisión Control Protocol) e IP (Internet Protocol) en realidad la arquitectura es un conjunto de protocolos que cubren los diferentes niveles del modelo OSI[10], tal como lo muestra la figura 9.1. Aunque el nivel de enlace y el físico están separados en la figura para separar los protocolos, dentro de la arquitectura se conocen

[9] Request For Comment (RFC) son un conjunto de archivos de carácter técnico donde se describen los estándares o recomendaciones de diferentes sistemas, entre los cuales se incluyen los protocolos de Internet.
[10] Modelo OSI – Lineamiento funcional para tareas de comunicaciones, define una Estructura Multinivel, puntos de acceso, dependencia de niveles, encabezados y unidades de información. Aunque no especifica un estándar de comunicación, muchos estándares y protocolos siguen sus lineamientos.

como un solo nivel de acceso al medio. Dentro de los protocolos se encuentra entre otros: UDP, ARP, ICMP, Ethernet, FTP, SMTP, RIP, IGRP.

Figura 9.1. Comparación del Modelo OSI y la Arquitectura TCP/IP
Fuente del Autor

La arquitectura está diseñado para enrutar, actualmente es usado en redes empresariales, campus universitarios e incluso a nivel mundial para conectarse a Internet. Por ello hace parte de la enseñanza en diferentes universidades para el pregrado en Ingeniería de Sistemas.

La Universidad EAFIT posee dos materias básicas y una línea de énfasis en Telemática, en ésta última existe una materia (TCP/IP) dedicada a esta arquitectura. La Universidad de los Andes en su materia Interconectividad

describe como "Se presenta cada uno de los niveles que componen la pila de comunicaciones, haciendo un especial énfasis en la pila TCP/IP"[11]. Otra universidad nacional es la Universidad del Cauca, donde en la materia de Redes 4 de los 7 módulos que la componen se dedica a describir puntualmente la arquitectura TCP/IP[12].

9.4 Routing Information Protocol (RIP)

Es un protocolo de enrutamiento dinámico, tal cual como se conoce hoy fue definido por primera vez en el RFC 1058 [2] por C. Hedrick en Junio de 1988, posteriormente mejorado para la versión dos por G. Malkin en el RFC 2453 [3] en Noviembre de 1998 y su última versión se encuentra recogida en el RFC 2080 [4].

RIP es considerado un protocolo Vector de Distancia, cuya formulación está basada en la ecuación de Bellman [1]. El enrutamiento para los protocolos de vector de distancias determina que un enrutador informe a sus vecinos los cambios en la topología periódicamente o en algunos casos en cuanto se detecte. El cálculo del costo para alcanzar un destino se hace a través de cálculos matemáticos como la métrica. En el caso de RIP la métrica está determinada por el número de saltos que se deben efectuar para llegar al destino, es decir el número de enrutadores por los que pasará un paquete.

El protocolo es simple en cuanto a implementación en contraste con otros protocolos como OSPF o BGP, adicionalmente es un protocolo abierto a diferencia de IGP o EIGRP propietarios de CISCO.

RIP cuenta con tres limitaciones conocidas y descritas por los autores del RFC:

1. La escalabilidad del protocolo es limitada, pues este no permite más de quince saltos, es decir, los dos enrutadores más alejados de la red no pueden alejarse más de 15 saltos, si esto ocurriera no sería posible utilizar RIP en esta red.
2. RIP cuenta con una problemática conocida como ciclos de enrutamiento, que se producen en casos particulares en los que un paquete puede quedar "rondando" por la red porque la convergencia

[11] Obtenido de la página http://sistemas.uniandes.edu.co , que describe los cursos del pregrado.
[12] Información obtenida de la página oficial de la Universidad del Cauca, http://www.unicauca.edu.co/ .

es lenta y puede no detectarse la ausencia de un enlace y seguir enviando a todos los vecinos una ruta que ya no existe. Entre las técnicas para solucionar este inconveniente se encuentra "Conteo a infinito", cuyo funcionamiento se refiere a que si un paquete no alcanza su destino en 15 saltos, se considera que esta "rondando" y por lo tanto se desecha.

3. El protocolo utiliza métricas fijas para comparar rutas alternativas, como se mencionó anteriormente utiliza el conteo de saltos, lo cual implica que este protocolo no es adecuado para escoger rutas que dependan de parámetros en tiempo real como por ejemplo retardos, carga del enlace, ancho de banda.

Adicionalmente la versión 1 del protocolo es *classful*[13] con lo que se tiene el problema de discontinuidad de las redes. Este problema se refiere al caso en el que se tiene una red dividida en subredes y no puede unificarse en una sola ruta porque están distribuidas en interfaces diferentes entre una subred y otra.

Las interfaces de un ordenador o enrutador pueden ser pasivas o activas, una interfaz pasiva es usada cuando un dispositivo no funciona como puerta de enlace pero desea recibir actualizaciones de las rutas. Por otro lado la interfaz activa se encarga de enviar las actualizaciones periódicamente y recibir las enviadas por sus vecinos, manteniendo así su tabla y a sus vecinos actualizados.

La segunda versión del protocolo fue publicada diez años después de la primera, dentro de esta se realizaron ciertas mejoras entre las que se encuentran:

1. Autenticación para la transmisión de información entre vecinos
2. Uso de máscaras de red, por lo que se soporta VLSM.
3. Uso de máscaras de red para determinar el siguiente salto, que permite la utilización de redes discontinuas.
4. Envío de actualizaciones vía multicast a través de la dirección 224.0.0.9.

[13] Classful IP Addresing se refiere a las direcciones de red diferenciadas en clases, donde el primer octeto determina la clase a la que pertenece y esta clase define el número de bits que corresponden al identificador de la red y cuántos a hosts. En la Clase A el primer octeto va desde 0 hasta 127, la Clase B desde 128 hasta 191, la Clase C desde 192 hasta 223, la Clase D desde 224 hasta 239 usadas para comunicaciones Multicast y por último la Clase E, que va desde la 240 hasta la 255.

Finalmente existe una versión que soporta IPv6 y que es conocida como RIPng, que trabaja sobre el puerto 521 de UDP. El formato del mensaje es diferente al detallado dentro de este trabajo, su definición puede verse en el RFC 2080 [4].

Funcionamiento de RIP

Inicialmente un enrutador dentro de su tabla de rutas posee únicamente las referencias a las redes directamente conectadas. El enrutador RIP envía actualizaciones periódicas a sus vecinos cada 30 segundos; para la versión 1 estos mensajes son enviados vía broadcast[14], para la versión 2 pueden ser también multicast[15]. Los enrutadores también tienen la posibilidad de enviar actualizaciones por fuera de la periodicidad cuando se detecta un cambio en la topología.

Si una ruta no es enviada en 180 segundos, se pone una métrica de 16 (infinito) en la tabla de rutas y después de 60 segundos es eliminada. Este cambio también debe ser propagado a través de la red.

Estos mensajes son enviados a través del puerto 520 de UDP (Ver la descripción en el Item UDP del marco teórico). En la figura 9.2 se puede ver el formato de un mensaje RIP. Cada mensaje permite un máximo de 25 rutas, en caso de ser más deben enviarse en otro mensaje.

La convergencia a través de este protocolo puede ser lenta para toda la red, dado que las actualizaciones se realizan cada treinta segundos y se propagan únicamente a los vecinos.

[14] Broadcast o difusión, es una técnica utilizada para enviar información a todos los posibles receptores. En el caso de las redes de comunicación se refiere al "envío de paquetes de datos de manera simultánea a todos los dispositivos de un segmento de red". Definición tomada de http://www.genesisbci.com/genesis/glosario/conceptos.asp .

[15] Muticast es la comunicación que realiza un emisor a varios receptores (no necesariamente todos). Para las redes de datos se refiere al envío simultáneo a un grupo específico de nodos dentro de una red.

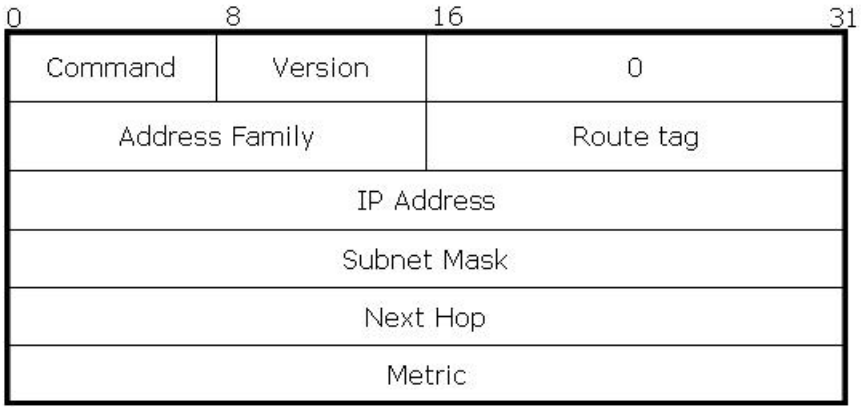


Figura 9.2. Formato de Mensaje RIP
Fuente del Autor

El campo ***command*** determina si es un mensaje de respuesta o de petición. El mensaje de petición solicita a un enrutador toda o parte de su tabla de rutas, el mensaje de respuesta es enviado periódicamente como actualización o como retorno a una petición.

El campo ***version*** se refiere a la versión del protocolo que se está usando; 1 o 2 de acuerdo a la versión de RIP implementada.

Los siguientes dos bytes siempre tienen el valor de cero, puesto que actualmente no son usados.

A partir del campo ***Address Family*** en adelante, los seis campos hacen parte de cada entrada en la tabla de rutas que es enviada a los vecinos de un enrutador. Si es un mensaje de petición, estos campos no están incluidos dentro del mensaje.

El campo ***Address Family*** en particular se refiere al protocolo de direccionamiento utilizado, para el caso de IP el valor es 2.

Route tag provee un método para distinguir rutas aprendidas por RIP y por otros protocolos. Para el caso de RIP v1, este campo permanece en cero.

Los siguientes campos están compuestos por 4 bytes por el tamaño que se define para cada uno de ellos. El primero es ***IP Address*** que contiene la dirección IP de la entrada que se quiere enviar.

Luego viene ***Subnet Mask*** que lleva la máscara de subred para la dirección, éste valor fue implementado para la versión 2 de RIP, puesto que, como se ha mencionado anteriormente, la versión uno no tiene presente este dato y siempre es enviado en cero.

El campo ***Next Hop*** especifica la dirección IP del siguiente salto para a ruta descrita.

Finalmente el campo ***Metric*** indica el número de saltos que actualmente posee la entrada en la tabla de rutas del emisor, este dato le permite al receptor calcular la métrica para determinar si puede ingresarla o no a su tabla de rutas.

A pesar de la edad de RIP y el surgimiento de protocolos de enrutamiento más sofisticados, éste está lejos de ser obsoleto. RIP es un protocolo maduro, estable, ampliamente soportado y fácil de configurar. Su simplicidad encaja bastante bien para ser usado en fragmentos de redes o sistemas autónomos pequeños que no poseen suficientes caminos redundantes para autorizar la sobrecarga de otros protocolos más sofisticados. [16]

9.5 User Datagram Protocol (UDP)

UDP es uno de los protocolos que conforman la capa de transporte de la Arquitectura TCP/IP. Esta capa es la encargada de proporcionar comunicación lógica entre procesos, es decir, simula una conexión física entre hosts. Los servicios proporcionados por los protocolos de la capa de transporte se pueden ver condicionados por el protocolo de la capa de red.

Normalmente dentro de un ordenador corre más de una aplicación al tiempo, cada una de estas aplicaciones es representada mediante un proceso que puede ser identificado dentro del sistema operativo. El protocolo UDP identifica cada aplicación a través de puertos de Protocolo (no del identificador del proceso) que se puede entender como un identificador único representado en un número entero positivo.

Cada mensaje contiene el número del puerto destino y origen. Lo que permite realizar el proceso de Multiplexación y Demultiplexación, este se refiere a la interpretación sobre el mensaje. Una aplicación al enviar un

[16] Fragmento traducido de la página http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/rip.htm .

mensaje lo ubica en un puerto particular, el nivel de transporte se encarga de recoger la información de los distintos puertos y encapsularla con información del origen, lo que se considera Multiplexación. La demultiplexación se refiere al momento en el que UDP recibe un paquete del nivel de red se extrae de la cabecera de IP la dirección IP destino y de la cabecera UDP el puerto de destino, de este modo se entregan los datos a la aplicación identificada con la dupla obtenida. La figura 9.3 presenta una idea más clara de la forma en que se maneja esta comunicación.

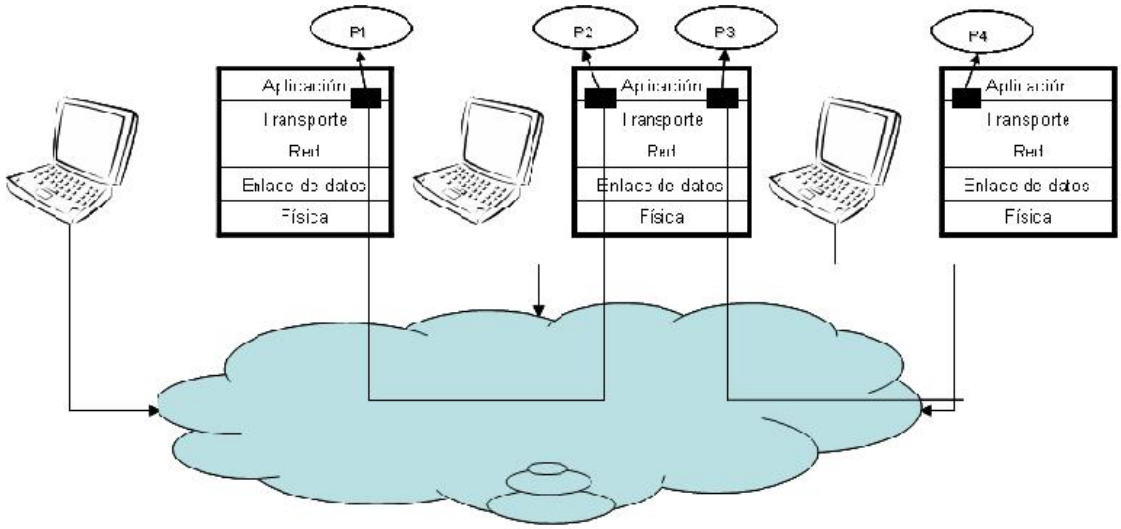


Figura 9.3. Comunicación a través de Puertos de Protocolos
Fuente del Autor

El protocolo UDP es descrito en el RFC 768 [6], dentro de sus características el servicio que presta es de mejor esfuerzo, es decir, no garantiza ordenamiento de los mensajes, entrega, integridad ni velocidad en la entrega de los datos.
Adicionalmente es un protocolo no orientado a la conexión, es decir, no se realiza un establecimiento previo o posterior a la transmisión de los datos. Las implicaciones de esto se ven reflejadas en que no existe capacidad para realizar control de errores ni de flujo.

Sin embargo UDP es efectivo en aplicaciones que no requieran precisión en la entrega de información, como es el caso de las transmisiones de voz, música o video. Como se mencionó en el punto 9.4 UDP también es usado para las transmisiones de mensajes RIP a través de la red.

Es un protocolo rápido en comparación con TCP, puesto que no requiere sincronización, genera poca sobrecarga puesto que su cabecera ocupa lo mínimo posible y es eficiente ya que no tiene control de congestión y por lo tanto usa todo el ancho de banda disponible.

9.6 Análisis estadístico

"El análisis estadístico pretende describir la población con base en información obtenida mediante la observación" [7]. El proceso para realizar esta transformación de datos recolectados a información que permitan obtener conclusiones, sigue un ciclo que inicia con la población y un objetivo de lo que se quiere determinar, luego se realiza una recolección de datos sobre una muestra, estos datos son analizados y se generan estadísticas a partir de ese análisis. Finalmente el análisis de estos resultados estadísticos permite determinar información sobre la población que posiblemente podrá resolver el objetivo inicial.

9.6.1 Recolección de datos

Para realizar un correcto análisis estadístico es importante llevar a cabo una correcta recolección de datos, éste es un proceso complejo que requiere el desarrollo de los siguientes pasos:

1. Definir los objetivos de la investigación o el experimento. Es decir, el factor de determinante que se quiere concluir.
2. Definir la variable y población de interés. Va muy ligado con el punto anterior, comúnmente el objetivo incluye una idea de quién debería ser la población.
3. Definir los esquemas para recolectar y medir datos. Incluye el procedimiento de muestreo, tamaño de la muestra e instrumento de medición de los datos.
4. Determinar las técnicas idóneas para realizar el análisis de datos: descriptivas o inferenciales.

Ahora bien, si es posible listar y observar cada elemento de la población, entonces es posible llevar a cabo un censo. Un censo es una encuesta al 100% de la población. Sin embargo, esta técnica es poco utilizada, muchas veces porque la población es bastante grande lo que conlleva tiempo y costo a los ejecutores del mismo.

En vez de elaborarse un censo, suele realizarse una encuesta muestral. Para ello es necesario elaborar un ***marco muestral***[17].

Para la selección de la muestra se realiza un ***diseño muestral,*** este diseño puede clasificarse en dos categorías:

- Muestras de juicio. La muestra es elegida por el realizador de la encuesta, pues las considera representativas de la población. Estas muestras son elegidas con base en el hecho de que son típicas, es decir, los elementos del marco muestral no tienen características particulares.
- Muestras probabilísticas. Las muestras son seleccionadas con base en la probabilidad, donde cada elemento cuenta con cierta probabilidad de ser elegido.

    - Muestra aleatoria simple

      Una muestra es seleccionada dado que todos los elementos de la población tienen la misma probabilidad de ser elegidos. De este modo todos los elementos del marco muestral son enumerados y aleatoriamente se selecciona la muestra hasta alcanzar el número de muestra necesario.

    - Muestra aleatoria estratificada

      Muestra que se obtiene al estratificar el marco muestral y seleccionar un número fijo de cada estrato por la técnica de muestra aleatoria simple.

    - Muestra aleatoria por conglomerados

      Se obtiene al muestrear algunas, no todas, las subdivisiones posibles que hay dentro de una población.

9.6.2 Diseño del cuestionario o encuesta

En el momento de desarrollar una encuesta se deben tener varios factores presente, entre ellos la longitud de la encuesta juega un papel importante.

[17] El marco muestral es una lista de los elementos que pertenecen a la población de la cual se obtendrá la muestra. Cuando no se tiene conocimiento o no es posible listar toda la población puede usarse un padrón electoral o un directorio telefónico.

Mientras más largo sea el cuestionario, menor será el cociente de respuesta [8], por esta razón las preguntas deben ser cortas y precisas para poner obtener información valiosa.

Las encuestas pueden realizarse en forma personal, telefónica o por correo electrónico. La personal tiene una tasa de respuesta mayor pero es más costosa de realizar.

Los principales problemas que se pueden obtener durante la realización de una encuesta a una muestra de la población son mencionados a continuación:

- Error de cobertura o sesgo de selección. Resulta por la posible exclusión de elementos del marco muestral, esto genera sesgo en la selección.
- Error de muestreo. Refleja la heterogeneidad o diferencias de oportunidad de muestra a muestra, podría disminuirse tomando muestras de tamaño mayor.

- Error de medición. Inexactitudes en las respuestas debido a la mala formulación de las preguntas, del efecto del entrevistador en el entrevistado o el esfuerzo por parte del encuestado.

10. Desarrollo de la propuesta

10.1 Análisis de la modularidad de KivaNS

Para poder incluir un nuevo protocolo al simulador KivaNS se hizo necesario realizar un análisis detallado de la modularidad de esta API de tal modo que se pudieran identificar claramente los puntos en los cuales debíamos intervenir para poder proveer un nuevo nivel a la pila de comunicaciones que éste ya tenía incluido al momento de obtener los fuentes de Internet.

Este análisis de modularidad se llevó a cabo en dos etapas, una primera etapa de lectura de la documentación de la herramienta, la cual se puede obtener del sitio oficial de KivaNS, y una segunda etapa que consistió en la revisión del código fuente y la comprensión del mismo para poder establecer con seguridad qué se debía hacer en el API y qué componentes nuevos se debían crear para la implementación de RIPv1 en KivaNS.

Básicamente la modularidad la ofrece KivaNS tanto a nivel de Equipos, Redes y Protocolos, pues define (el simulador) unas clases genéricas a partir de las cuales se deben desarrollar los nuevos módulos, con sus respectivos métodos abstractos que deben ser implementados por los componentes nuevos desarrollados. Sólo basta entonces con programar el módulo deseado bajo el estándar de construcción definido por los "esqueletos" de KivaNS y realizar una sencilla modificación en el API de programación en caso de ser un protocolo de la pila de comunicaciones, pues se requiere en algún punto incluir el módulo nuevo en la pila ya existente.

La documentación del API de simulación es supremamente clara y la concordancia con el código fuente es transparente, de tal modo que identificar los puntos a intervenir fue una tarea fácil de realizar; en esta documentación se puede observar cómo en la página once existe un numeral dedicado a la explicación de la pila de comunicaciones, la cual es el componente del simulador encargado de simular el pila de protocolos de la arquitectura TCP/IP con la limitante de tener implementado únicamente hasta el nivel IP. En esta sección (sección 2.4 de la documentación del simulador) se puede ver como los niveles de la pila son independientes entre sí desde el punto de vista de la implementación.

La modularidad original de KivaNS, entonces, soporta fácilmente la inclusión del protocolo RIPv1 debido al diseño del componente *Nivel* implementado en la base del simulador, esta facilidad surge de los parámetros de construcción a seguir por la misma plataforma de trabajo.

10.2 Análisis de la inclusión del protocolo RIPv1

Una vez comprendida la modularidad de KivaNS en cuanto a protocolos, se analizó entonces la adaptabilidad de la plataforma a un nuevo protocolo de enrutamiento dinámico como RIP v1. Fundamentalmente se encontraron las siguientes dificultades:

- RIP es un protocolo que trabaja sobre UDP y KivaNS no tiene implementado un nivel de transporte en su pila de comunicaciones.

- RIP, al utilizar UDP como protocolo de transporte hace uso del concepto de los puertos del Sistema operativo para comunicarse con los otros niveles de la pila de comunicaciones; KivaNS por su parte, al

no contar con un nivel de transporte, no contaba tampoco con la implementación de Puertos en los equipos.

- RIP es un protocolo que envía mensajes de actualización cada 30 segundos, mientras que KivaNS no simula el enrutamiento de paquetes en tiempo real.

Estas dificultades fueron enfrentadas de la siguiente manera:

- Ya que KivaNS es una plataforma en la cual es relativamente fácil introducir un nuevo nivel en la pila de comunicaciones, se decidió implementar entonces el nivel UDP como protocolo de transporte; los motivos de esta decisión fueron los siguientes:

    - UDP es un protocolo no orientado a la conexión, lo que significa que no se encarga de establecer, mantener y finalizar una conexión, no garantiza la correcta entrega de paquetes al destino, ni que la entrega de los mismos suceda en orden; este factor hace de UDP un nivel de fácil desarrollo, pues simplemente se preocupa por obtener el puerto de destino de un data grama y ubicarlo en dicho puerto en el equipo en el que corre.
    - El datagrama UDP simplemente almacena información de los puertos origen y destino, adicionalmente un checksum que es opcional, aunque generalmente en la práctica es usado; en la implementación de este protocolo para KivaNS este campo no es utilizado, pues en este caso es prioritaria la velocidad de comunicación entre los módulos mediante UDP que la garantía de que los mensajes lleguen desde el origen hacia el destino. Por tanto la implementación de este datagrama se podía ensamblar fácilmente usando el esqueleto base obtenido en el sitio Web de KivaNS.
    - Añadir un protocolo a la pila de comunicaciones es una tarea fácil de realizar en el API de simulación, pues solamente hay que modificar la sección de código en los equipos enrutador y Ordenador en la cual se establecen las comunicaciones entre los diferentes módulos de la misma.

- El puerto es una unidad funcional de un nodo o una interfaz a través de la cual los datos pueden entrar o salir de una red de datos [18]. Dicha interfaz puede ser física, o puede ser a nivel software (por Ej.: los

puertos que permiten la transmisión de datos entre diferentes computadoras). En KivaNS la implementación de este concepto es realizable y dado que cada nivel tiene sus propias estructuras de datos que sirven para la comunicación entre niveles, entonces la implementación de un puerto consistiría únicamente en una capa adicional encargada de actuar como interfaz entre estas estructuras de datos. Es decir, si ya cada nivel tiene sus propias estructuras de datos, un puerto entonces se encargará de realizar un puente entre estas estructuras, sin hacer más función que ésta.

- El hecho que KivaNS no pueda simular eventos periódicos constituyó una amenaza para el proyecto pero una vez se profundizó en el análisis de la solución de esta dificultad pudimos concluir que la podíamos convertir en una oportunidad. Esto lo lograríamos ubicando el análisis en la finalidad principal de KivaNS, la cual es la enseñanza de redes de datos y particularmente del enrutamiento de paquetes en redes de arquitectura TCP/IP; si una simulación de RIPv1 simulara un pulso de RIPv1, es decir, aquello que sucede cada 30 segundos en la ejecución del protocolo, estaríamos otorgando al estudiante el control sobre la ejecución del mismo, de tal modo que pueda observar con toda tranquilidad la convergencia de una red en el simulador, manipulando él mismo el tiempo de convergencia. De este modo entonces dedujimos que este factor no representaba inconveniente y se implementó tal como se ha mencionado en este apartado.

10.3 Implementación del Nivel UDP

Como se expuso anteriormente la inclusión entonces de un Nivel de simulación del protocolo UDP se hizo necesaria, pues RIP, si bien es un protocolo de nivel 3, trabaja sobre el puerto 520 UDP y no con un identificador único de protocolo para el nivel de red IP, como se efectúa la comunicación entre protocolos en la arquitectura TCP/IP.

La implementación del nivel UDP tiene las siguientes consideraciones:

- Utiliza la clase padre Nivel para definir sus atributos y comportamientos generales como nivel de la pila de protocolos.

- Un "*NivelUDP" (clase con definición del Nivel UDP)* pertenece a un equipo, es decir, tiene como atributo una variable de tipo Equipo, lo cual le permite acceder a sus atributos mediante sus métodos *getters* o directamente si estos son públicos. Particularmente nos interesa que tenga acceso a los puertos de ese equipo, para poder poner en ellos información que se reciba en el módulo UDP.

- No es una implementación compleja, pues como se mencionó anteriormente, este nivel simplemente se encarga de recibir datagramas UDP, preguntar por su puerto de destino, y proceder entonces a ubicar los datos de ese datagrama en dicho número de puerto.

La siguiente imagen muestra cómo se implementó el Nivel UDP y sus principales características, recién mencionadas.

```
import Redes.*;


/**
 * Módulo IPv4
 */
public class NivelUDP extends Nivel
{

    /**
     * Instante actual
     */
    int instanteActual;

    /**
     * Inicializador de la clase
     */
    static
    {
        // El id en UDP para IPV4 es 0
        LocalizadorRedes.Registrar("UDP","ipv4",0);
    }

    public NivelUDP(Equipo equipo)
    {
        super(equipo);

    }


    public void Procesar(int instante)
    {
        // 0. Actualizacion
        instanteActual=instante;

        // 1. Procesamos las entradas
        ProcesarEntrada(instante);

        // 2. Procesamos las salidas
        ProcesarSalida(instante);
    }
```

Código 10.1: Implementación del Nivel UDP
Fuente del Autor

```
int puerto_origen  = d.getPuertoOrigen();

int puerto_destino = d.getPuertoDestino();

Buffer datos = new Buffer(d.Tam()-d.kLONGITUD_ENCABEZADO);
for(int j=0;j<datos.Tam();j++)
    datos.setByte(j,d.getByte(d.kLONGITUD_ENCABEZADO+j));

Dato datoAux = new Dato(++dato.instante, datos);
datoAux.direccion = dato.direccion;
datoAux.interfaz = dato.interfaz;


//4. Se pasa el dato contenido en el datagrama (payload) al puerto destino
this.equipo.puertos[puerto_destino].ponerDatoParaAplicacion(datoAux);
```

Código 10.2: Pregunta por el puerto de destino y la pone allí el payload del datagrama recibido.
Fuente del Autor

Como componente del diseño de esta implementación del nivel UDP se expone la siguiente imagen la cual expone en el lenguaje universal de modelo (UML) la relación entre lo existente en KivaNS y el componente nuevo implementado. Esta figura (Figura 10.1) refleja la fácil adaptabilidad de un componente nuevo a la plataforma de trabajo que ya estaba desarrollada, pues para adicionar un protocolo nuevo simplemente (en términos de diseño) se debe construir una clase la cual debe heredar de una clase padre que generaliza la mayor parte de los comportamientos de un Nivel.

En el Código 10.3 se muestra como la modificación del código existente de la plataforma de trabajo KivaNS es mínima; simplemente se requirió modificar la asignación de relaciones en la pila de comunicaciones tanto del Ordenador como del Router. En esta sección de código note por favor que simplemente se asigna a nivelUDP como nivel superior del nivelIPv4 y a este como nivel inferior de nivelUDP.

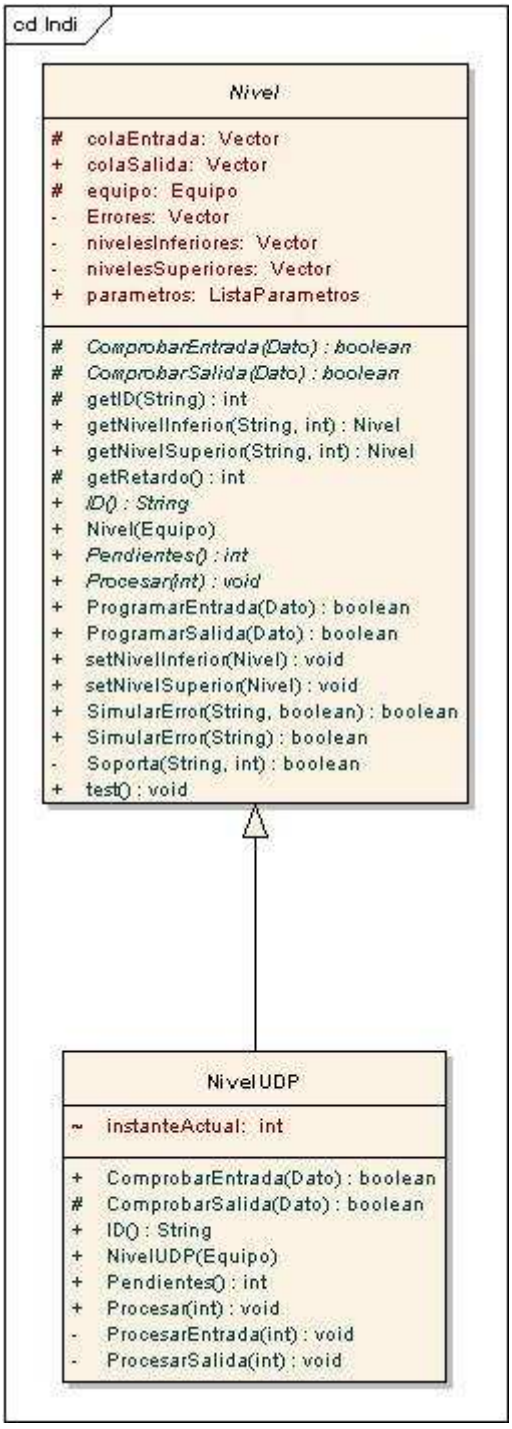


Figura 10.1: Herencia de la clase nueva NivelUDP de la clase existente Nivel.
Fuente del Autor

```
public Router()
{

    // Inicia el arreglo de puertos
    puertos = new Puerto[65536];

    // 1. Definimos los niveles de la pila
    moduloARP=new ModuloARP(this);
    moduloICMP=new ModuloICMP(this);
    nivelIPv4=new NivelIPv4(this,moduloARP,moduloICMP);

    nivelUDP = new NivelUDP(this);


    // 2. Interconectamos los niveles
    moduloICMP.setNivelInferior(nivelIPv4);
    nivelIPv4.setNivelInferior(moduloARP);
    nivelIPv4.setNivelSuperior(moduloICMP);

    nivelIPv4.setNivelSuperior(nivelUDP);
    nivelUDP.setNivelInferior(nivelIPv4);

    // 3. Enlazamos la tabla de rutas
    tablaDeRutas=nivelIPv4.tablaDeRutas;

    // 4. Activamos el IP Forwarding
    nivelIPv4.IPForwarding(true);
}
```

Código 10.3: Inclusión del protocolo UDP en la pila de comunicaciones de KivaNS.
Fuente del Autor.

10.4 Implementación de Puerto

El puerto es una unidad funcional de un nodo o una interfaz a través de la cual los datos pueden entrar o salir de una red de datos [18]. Dicha interfaz puede ser física, o puede ser a nivel software (por Ej.: los puertos que permiten la transmisión de datos entre diferentes computadoras).Un puerto entonces es el encargado de almacenar los datos que llegan a él, para que luego un proceso particular los tome de allí y los procese.

Sin embargo, para aprovechar la modularidad que ofrece KivaNS y que se expuso en el punto *Análisis de la modularidad de KivaNS*, la implementación de puerto no almacena los datos en una estructura de datos propia, sino que

se vale de las estructuras de datos que tiene la clase padre *Nivel* materializadas en sus atributos *colaEntrada* y *colaSalida*. Esto permite que los niveles de aplicación se implementen como clases hijo de la clase *Nivel* aprovechando la generalización que ésta ofrece y evitando la implementación de una nueva clase padre para este TDA.
La clase Puerto entonces tiene dos atributos de tipo *Nivel*, uno que representa el nivel del transporte al que está asociado y el otro que representa el nivel de aplicación que toma datos de él. Los servicios que ofrece un Puerto en KivaNS son, de acuerdo a esto:

a) Pasar los datos de la aplicación relacionada al transporte relacionado.
b) Pasar los datos del transporte relacionado a la aplicación relacionada.

Por pasar se entiende ingresar un *Dato* en la *colaEntrada* o *colaSalida* del nivel al que se esté comunicando los datos.

De este modo, una aplicación no conoce su nivel de transporte, simplemente se preocupa por poner los datos que desea transferir en el puerto asociado, y éste se encargará de pasarlos al nivel de transporte correspondiente. A su vez, el nivel de transporte nunca se entera de qué aplicación particular está tomando los datos del puerto destino, este nivel simplemente se preocupa por ubicar en el Puerto los datos que le llegan.

Así pues, nuestra implementación de Puerto cumple con las características básicas de una interfaz de este tipo, la cual sirve como medio de comunicación entre un protocolo de capa de transporte y un protocolo en la capa de aplicación.

```
/**
 * Métood que ubica un dato en la cola de entrada de la aplicación que está escuch
 * @param dato Dato a pasar a la aplicación
 */
public void ponerDatoParaAplicacion(Dato dato){
    //se programa el dato en la cola de entrada de la aplicacion que escucha este
    boolean t = this.aplicacion.ProgramarEntrada(dato);
}

/**
 * Método que ubica un dato en el protocolo de transporte asociado
 * @param dato Dato a poner en la estructura de datos del nivel de transporte asoc
 */
public void ponerDatoParaTransporte(Dato dato){
    //el puerto marca el dato como procedente de aca mismo
    dato.puerto_origen=this.getNumero();
    //se programa el dato en la cola de salida del nivel de transporte asociado a
    this.nivel_transporte.ProgramarSalida(dato);

}
```

Código 10.4: Servicios prestados por un puerto
Fuente del Autor.

Nótese como los servicios prestados por un puerto, mostrados en el Código 10.4, son tan sencillos como utilizar los métodos *ProgramarEntrada(Dato)* y *ProgramarSalida(Dato)* de la clase *Nivel*, métodos mediante los cuales, efectivamente, se hace la inserción en las estructuras de datos de cada nivel.

Como componente de diseño y arquitectura de esta nueva implementación para KivaNS, se muestra a continuación en UML la relación entre clases existentes y clases implementadas para la construcción de un Puerto el simulador.

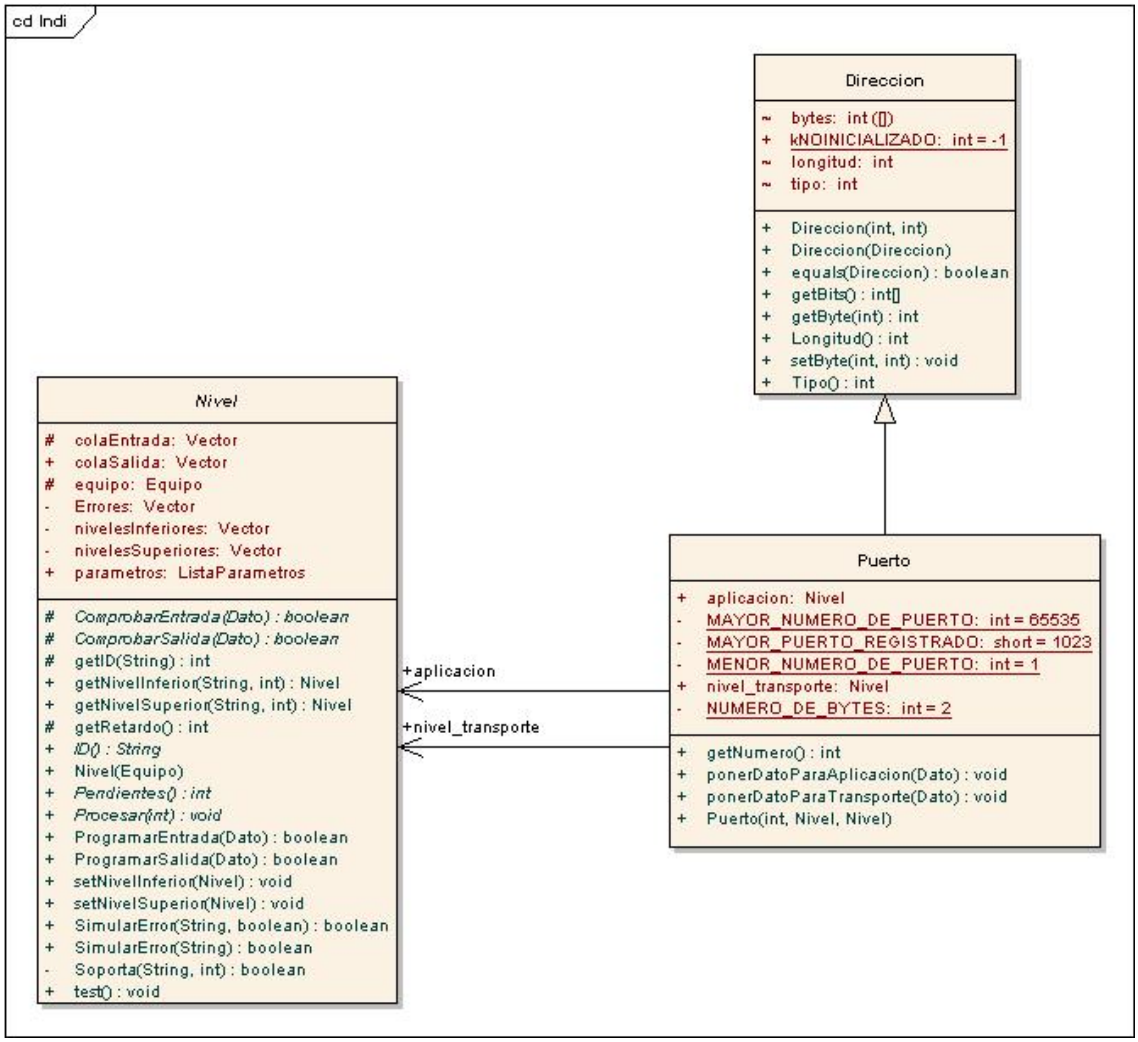


Figura 10.2: Representación en UML de la implementación de Puerto en KivaNS.
Fuente del Autor.

Nótese la relación con Nivel, tanto para el nivel de transporte como para el de aplicación, y la herencia de la clase Dirección. Si bien un Puerto no es una dirección como tal, se construyó como clase hija de Dirección para utilizar las funcionalidades que ofrece ésta sobre su arreglo de enteros. De este modo lo que para una Direccion almacena la dirección en sí, para un Puerto, este campo de la clase almacena el número de puerto y mediante los métodos de la clase dirección se pueden ejecutar operaciones sobre este atributo, tales como, crearla, modificarla etc; de esta manera se evita incurrir en duplicación de código que se generaría al desarrollar estas funcionalidades dentro de la clase Puerto si no fuera hija de la clase Direccion.

10.5 Implementación del Datagrama UDP

Así como existe la modularidad y escalabilidad a nivel de Protocolo (clase padre Nivel), los diferentes tipos de datagramas también se pueden implementar fácilmente en el simulador KivaNS, pues para éstos también existe un estándar de construcción y un conjunto de clases que facilita la implementación de un datagrama nuevo requerido.

Dado que se implementó un NivelUDP para simular el comportamiento del protocolo de transporte no orientado a la conexión UDP, se hizo necesario, evidentemente, la creación de una clase que simulara un Datagrama generado por este nivel. Dado que es un protocolo sencillo, el encabezado de un datagrama UDP contiene información del puerto origen, el puerto destino, la longitud del paquete y una información de suma de verificación que generalmente no es usada, la descripción en la figura 10.3.

+	Bits 0 - 15	16 - 31
0	Puerto origen	Puerto destino
32	Length	Suma de verificacion
64	Datos	

Figura 10.3: Datagrama UDP
Fuente del Autor.

De acuerdo a esto entonces se creó la clase *DatagramaUDP* la cual tiene estos cuatro atributos. Esta clase entonces provee los dos métodos constructores que debe tener cualquier datagrama en KivaNS, los cuales son:

a) Aquél que recibe los datos en un *Buffer*: Se encarga de tomar los datos del buffer de entrada y asignar a partir de estos los valores en las variables de un *DatagramaUDP.*

b) Aquél que recibe los datos puntualmente, es decir, recibe el número de puerto origen, el número de puerto destino, la longitud del datagrama, la suma de verificación y el *payload* del datagrama. Este constructor por el contrario, se encarga de llevar estos datos particulares a un

Buffer para que éste pueda ser enviado como tal a un nivel superior o inferior.

La representación formal de la implementación del DatagramaUDP en el simulador KivaNS mediante UML se muestra en la Figura 10.4; esta figura expone como este Datagrama hereda de la clase Buffer, la cual es la clase padre de todos los datagramas que se deseen crear para KivaNS. Siguiendo este estándar de construcción, se puede manipular un DatagramaUDP bien sea asignando o recuperando datos de éste, mediante los métodos que para esto ofrece la clase Buffer.

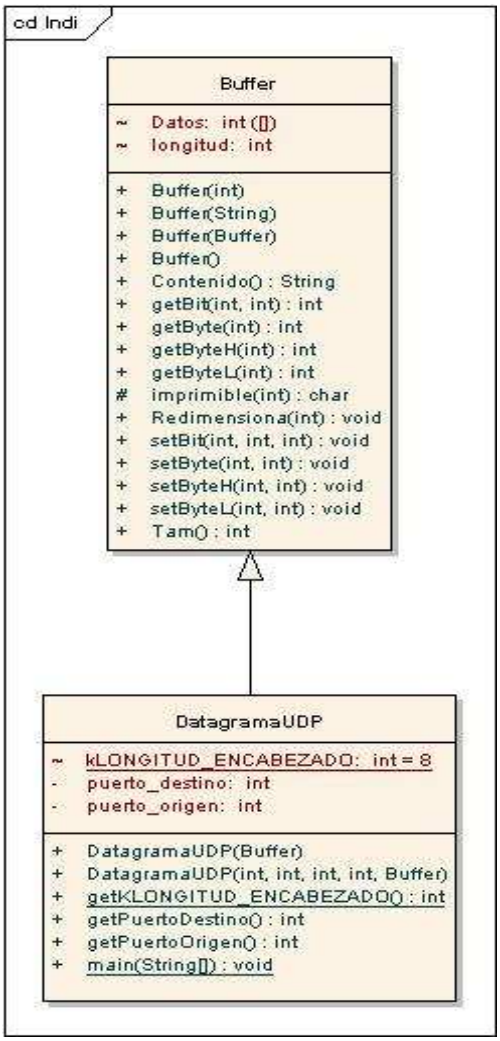


Figura 10.4: Modelado de clases del DatagramaUDP para KivaNS.
Fuente del Autor

10.6 Implementación del Nivel RIPv1

El nivel RIPv1 en la implementación se ubica en la clase *ModuloRIPv1* y contiene toda la lógica funcional del protocolo de enrutamiento dinámico RIP v1. Si bien no es un nivel como tal, pues funcionalmente actúa como un protocolo de la capa de aplicación del estándar OSI (escuchando un puerto particular), ya se ha expuesto porqué se aprovechó la generalización de la clase *Nivel* para abstraer un protocolo como Ripv1.

La siguiente figura muestra la ambigüedad que puede presentar RIPv1 al querer ubicarlo en un nivel determinado dentro de KivaNS, puesto que es un protocolo de enrutamiento que trabaja como una aplicación puesto que utiliza los servicios de UDP y tiene un número de puerto asignado para el envío de mensajes.

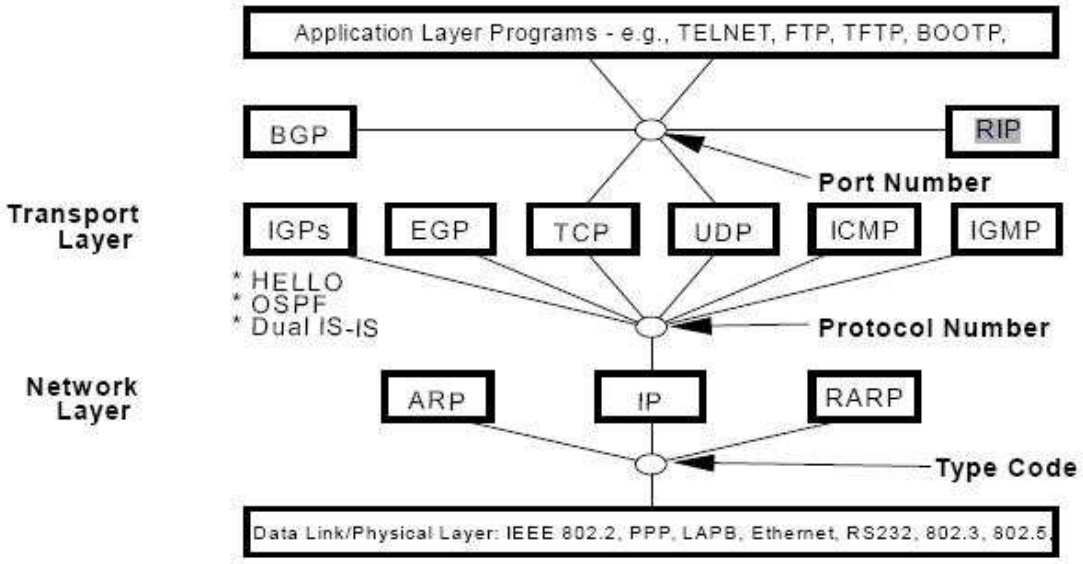


Figura 10.5: Familia de protocolos TCP/IP
Fuente: Tomado de documento de Asignatura TCP/IP Universidad EAFIT Semestre 2004-1

La clase *Nivel* ofrece la posibilidad de ser usada para RIPv1 pues igual permite implementar un protocolo independiente de la capa a la que pertenezca en el pila de protocolos TCP/IP.

De manera general, las principales funcionalidades de este módulo RIPv1 consisten en:

a) Actualización del timeOut de una ruta: básicamente este componente toma decisiones sobre los timeouts de una ruta en la tabla de rutas del equipo al que pertenece, basado en las rutas que llegan en un mensaje RIPv1.

b) Ingreso de Rutas: luego de ejecutarse la actualización de los timeouts, se realiza una verificación de las rutas que han llegado como actualización de los vecinos, y que deben ser ingresadas en la tabla de rutas del equipo que corre RIPv1. Las tablas de rutas de cada equipo son actualizadas de acuerdo a la información obtenida en este punto, lo que permite la convergencia real de la red.

Bien es sabido los problemas que presenta RIPv1 con respecto a no soportar máscara de longitud variable, no aplicar en redes cuyas rutas presenten una métrica mayor a 15 saltos en la tabla de rutas de algún nodo y presentar problemas en redes con ciclos si no se maneja una estrategia adecuada.

Estos tres inconvenientes se abordaron de la siguiente manera:

- **No soportar VLSM:** El VLSM es soportado siempre y cuando no haya discontinuidad de la subred, dado que la máscara no viaja en el mensaje RIP no puede determinarse la red original. De este modo RIPv1 permite tener directamente conectadas una subred y realiza el enrutamiento correctamente, una vez la discontinuidad se pierde, es decir, la subred se encuentra en redes físicamente separadas la red no realiza una convergencia adecuada.
- **No aplicar en redes grandes:** para contrarrestar esta desventaja no existe ninguna estrategia en la implementación del protocolo y si la red es grande (presenta más de 15 saltos) la estrategia radical es no usar RIPv1 como protocolo de encaminamiento dinámico.
- **No soportar ciclos en la topología sin estrategia adecuada:** para contrarrestar esta desventaja, se sugiere implementar una estrategia llamada conteo al infinito mediante las siguientes técnicas: *horizonte dividido*, *envenenamiento de ruta* y *actualización automáticas*. La técnica seleccionada en esta implementación del protocolo es ***triggered updates (actualización automática)*** la cual consiste en que un equipo que detecte una actualización en la métrica de una ruta

envíe automáticamente una actualización a sus vecinos anunciando este cambio. Para KivaNS se implementa esto programando un mensaje de actualización vía broadcast una vez se perciba que una métrica de una ruta se modifica.

Esta implementación de RIPv1 solamente utiliza mensajes de actualización, mas no hace uso de los mensajes de solicitud de actualización. Sin embargo, se han creado los cascarones necesarios para esta implementación tanto del mensaje de petición como en el módulo RIPv1 para el tratamiento de estos mensajes.

En el Código 10.5 y Código 10.6 se muestra como queda abierta la posibilidad de un trabajo futuro que consista en la implementación de la elaboración, envío y tratamiento de mensajes de petición RIPv1

Nótese cómo se realiza la validación al momento de procesar la salida, la cual verifica qué tipo de mensaje se está intentando enviar en RIPv1, una petición o una actualización. El mensaje de todos modos se crea y se envía, sin embargo, como se puede observar en la siguiente sección de código (*Código 6)*, el mensaje de tipo petición no contiene nada en absoluto y por tanto quien lo reciba no hará nada con él.

A continuación se muestra, en la *Figura 10.6* la representación formal en UML de la clase que implementa el módulo RIPv1 de acuerdo a las clases ya existentes del simulador.

Un componente supremamente importante a tener en cuenta es como el módulo RIPv1 no es nivel superior de ningún nivel en la pila así como tampoco existe algún nivel en la pila de comunicaciones que haga las veces de nivel inferior para RIPv1 de manera explícita en KivaNS. Esto se logra gracias a que a partir de la existencia de un protocolo de transporte y de la implementación del concepto de Puerto, el nivel UDP no tiene porqué conocer qué protocolo está por encima suyo sino que simplemente se limita a poner los datos en un puerto indicado y ya está.

Note como en Código 10.3 el moduloRIPv1 nunca es ingresado en la pila de comunicaciones y sin embargo los mensajes llegan hasta allí al momento de correr una simulación RIPv1.

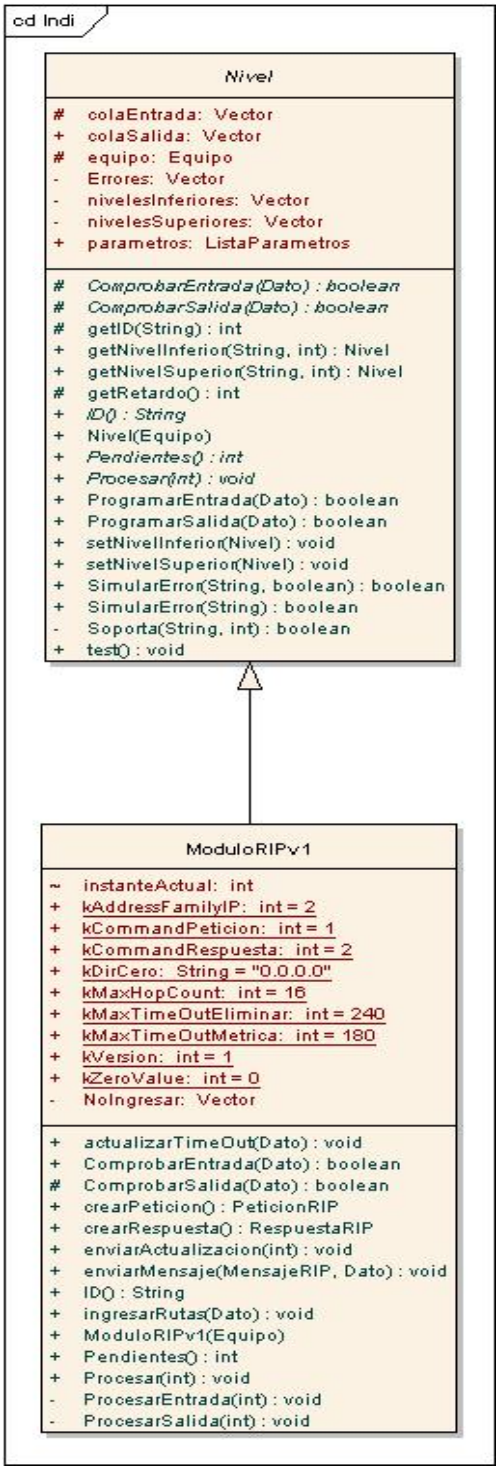


Figura 10.6: Diagrama de clases del módulo RIPv1 para KivaNS.
Fuente del Autor.

```
private void ProcesarSalida(int instante)
{
    Dato dato;

    // 1. Comprobamos todos los datos de salida
    for(int i=0;i<colaSalida.size();i++)
    {
        dato=(Dato)colaSalida.get(i);

        if(dato.instante==instante)
        {
            try
            {
                // Eliminamos el dato de la cola de salida
                colaSalida.remove(i);
                i--;

                MensajeRIP mensaje;

                //Leer el tipo de mensaje a enviar
                if(dato.paquete.getByte(1) == 0){
                    // Crear mensaje - peticion
                    mensaje=crearPeticion();
                }else{
                    // Crear mensaje - respuesta
                    mensaje = crearRespuesta();
                }

                // Enviar el mensaje
                enviarMensaje(mensaje, dato);



            }
            catch(Exception e) {}
        }
    }
}
```

Código 10.5: método ProcesarSalida(int instante) del módulo RIPv1
Fuente del Autor.

```
public PeticionRIP crearPeticion(){
    // queda planteado como trabajo futuro
    PeticionRIP peticion = new PeticionRIP(kCommandPeticion, kVersion);

    return peticion;
}

public RespuestaRIP crearRespuesta(){
    // Respuesta a enviar en un dato
    RespuestaRIP respuesta = new RespuestaRIP(kCommandRespuesta, kVersion);

    // Entrada a ingresar en el mensaje RIP
    DireccionACompartir entrada;

    int metricaEntrada;

    // Recorre toda la tabla de rutas armando una dirección a compartir
    for(int i=0; i<this.equipo.tablaDeRutas.getNumeroEntradas(); i++){
        //obtiene la métrica en la tabla de rutas por cada entrada
        metricaEntrada = this.equipo.tablaDeRutas.getMetrica(i) ;

        // Se enviará una entrada si:
        // 1. la métrica de la entrada es menor a 16
        // 2. la métrica de la entrada es 16 y el timeOut es 180
        if(metricaEntrada < kMaxHopCount ||
         (metricaEntrada == kMaxHopCount && this.equipo.tablaDeRutas.getTimeOut(i)

            // Se construye una direccionACompartir con los valores de la tabla de
            entrada = new DireccionACompartir(kAddressFamilyIP,
                      kZeroValue,
                      this.equipo.tablaDeRutas.getDestino(i),
                      new MascaraIPv4(kDirCero),
                      new DireccionIPv4(kDirCero),
                      this.equipo.tablaDeRutas.getMetrica(i));

            // Se inserta en el mensaje
            respuesta.insertarDireccionACompartir(entrada);
        }
    }


    return respuesta;
}
```

Código 10.6: Diferencias en la creación de una respuesta y una petición RIPv1.
Fuente del Autor.

10.7 Implementación de los Mensajes RIP

RIPv1 según RFC 1058 define dos tipos de mensajes, uno de actualización que se envía cada 30 segundos a todos los vecinos vía broadcast y uno de petición que se envía extemporáneamente cuando un enrutador nuevo llega a la red o recién se le activa el protocolo RIPv1.

Las consideraciones acerca del tipo de mensaje que maneja KivaNS en la implementación de RIPv1 han sido expuestas en la sección *Implementación del Nivel RIPv1*.

Las clases mediante las cuales se implementó estos dos tipos de mensajes son MensajeRIP, PeticionRIP y RespuestaRIP. Para exponer más claramente la relación entre estas clases se muestra en la siguiente figura el diagrama UML.

En el diagrama de clases se puede observar como nuevamente se hace uso de la clase Buffer como abstracción de cualquier datagrama que se desee implementar para KivaNS de tal modo que se pueda utilizar los métodos que esta clase provee para manejar fácilmente los datos de un datagrama. La clase que se relaciona directamente con *Buffer* es MensajeRIP la cual a su vez hace de padre de PeticionRIP y RespuestaRIP. Estos mensajes son independientes de la versión, pues la estructura es la misma según el RFC que define RIP, la diferencia está en que RIP v1 no utiliza ciertos campos mientras que RIP v2 si los utiliza. Esto no debe representar ningún problema a la hora de implementar RIP v2 pues estos mensajes almacenan todos los campos y pueden ser asignados por parámetro, de tal modo que no se tenga que crear un TDA nuevo o modificar los ya creados cuando se desee implementar RP v2.

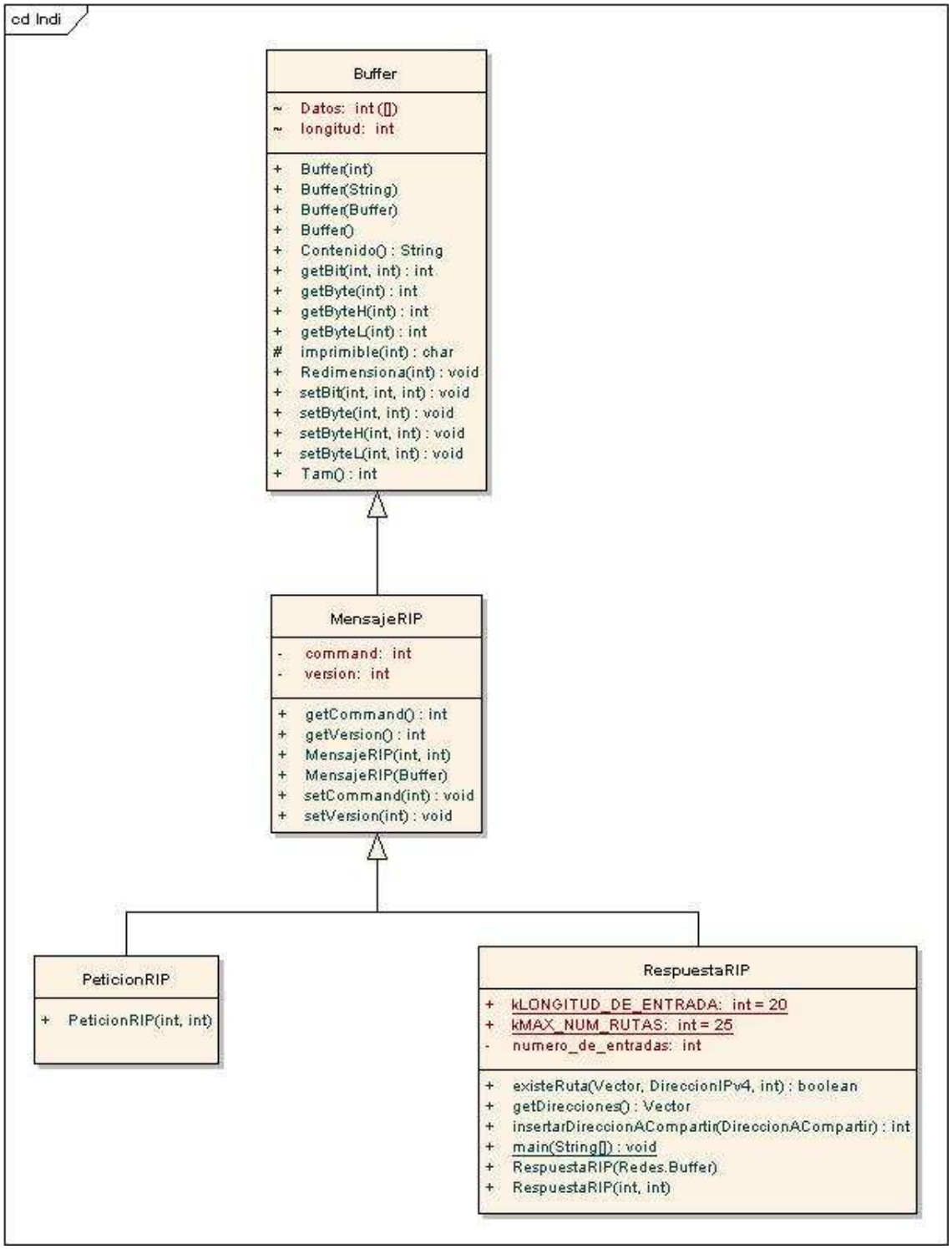


Figura 10.7: Diagrama UML de mensajes RIPv1.
Fuente del Autor.

10.8 Implementación de una DireccionACompartir

Un mensaje RIP de actualización, el cual es el utilizado en la implementación actual de RIPv1, contiene la información de la tabla de rutas del equipo que envía el mensaje. Esta información, como es evidente, es tomada de la tabla de rutas de este equipo y es luego ingresada en un mensaje RIP de actualización para ser enviado por la red en un mensaje IP Broadcast.

Esto no presenta ningún inconveniente desde el punto de vista conceptual, pero al momento de la implementación es mucho más cómodo de manipular una estructura de datos particular para ser enviada en el mensaje de actualización RIPv1; esta comodidad se verá reflejada tanto al momento de armar el mensaje de respuesta RIP como al momento de leerla para procesarla.

Es por este motivo que se creó un tipo de dato abstracto (TDA) particular para este fin llamado *DireccionACompartir* el cual encapsula los datos particulares que contiene cada ruta enviada en un mensaje de respuesta RIP tal como lo muestra la figura 10.8.

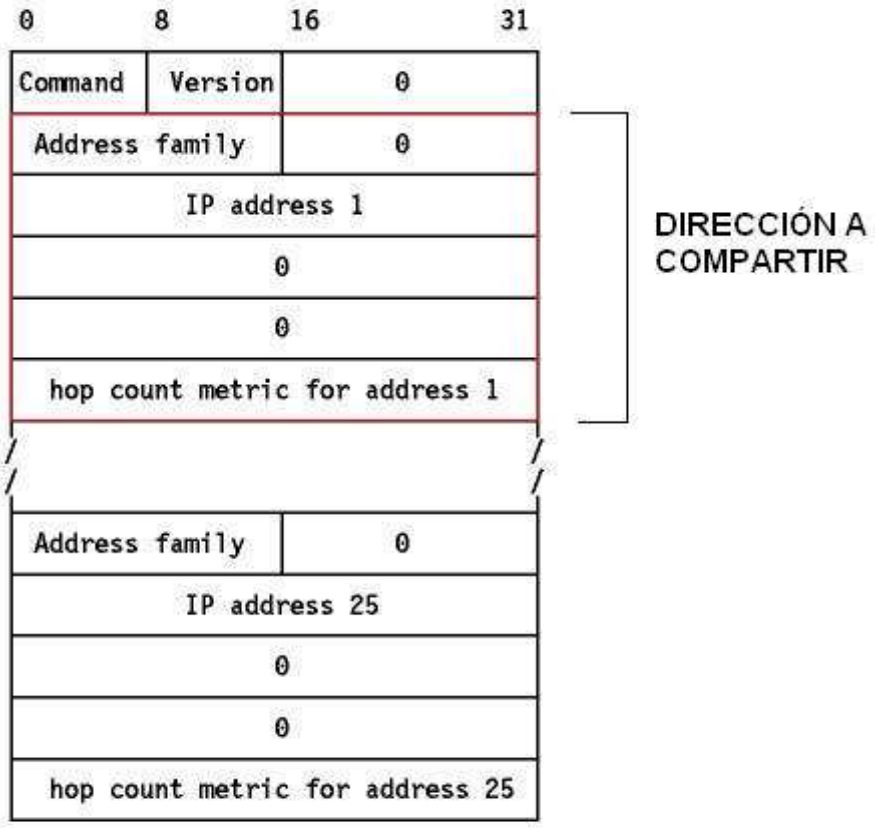
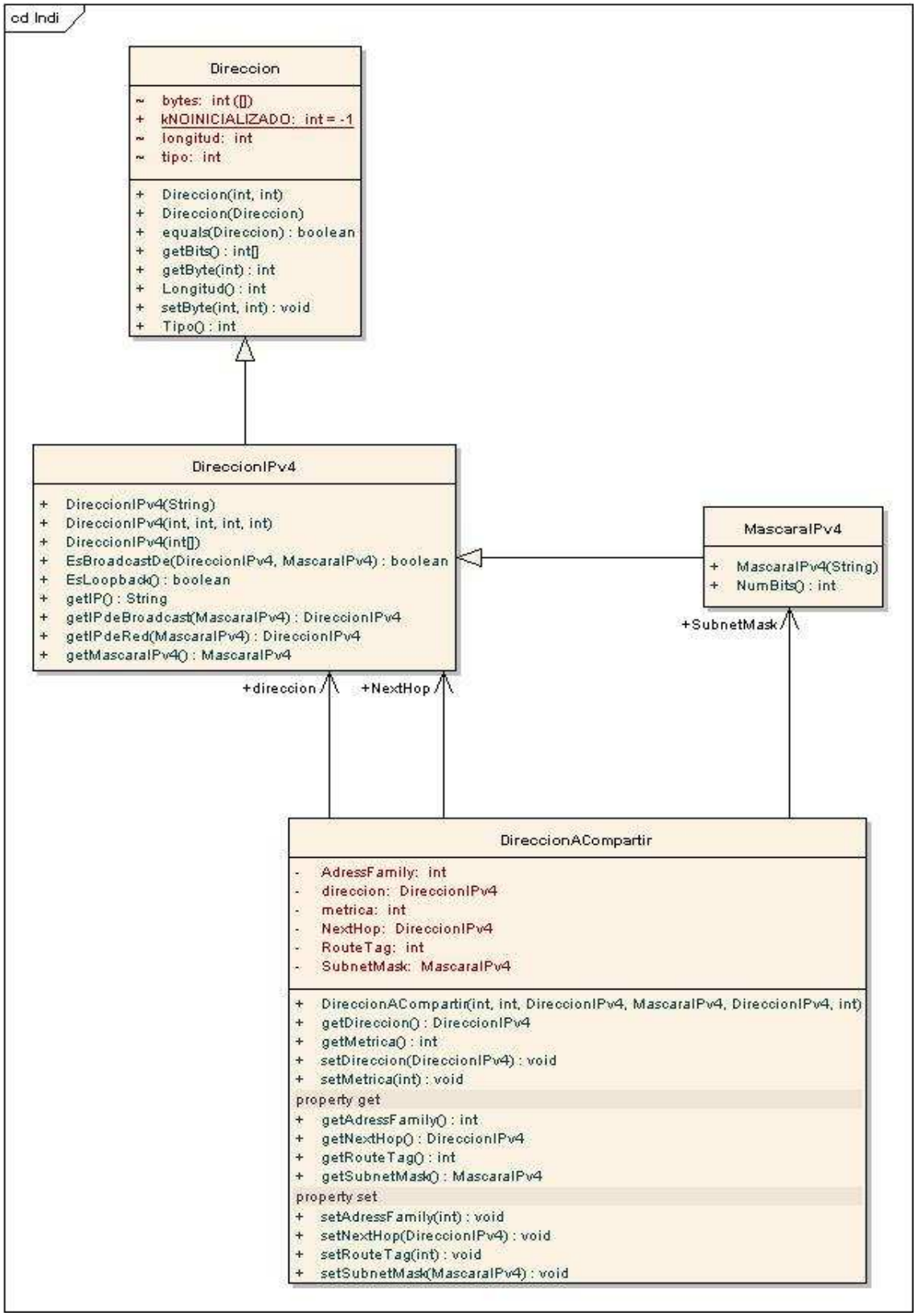


Figura 10.8: Encapsulado de registro en mensaje de respuesta RIP.
Fuente: RFC 1058.

Note que la zona remarcada en color rojo se repite en el mensaje, pues en un solo mensaje se envía información de más de una ruta. La clase *DireccionACompartir* encapsula esta información y provee un método para obtener un *Vector* con las repeticiones de esta zona del mensaje. Esto reduce la lectura a un solo llamado en lugar de estar leyendo el mensaje cada vez y en lugar de delegar el control de esta lectura al módulo RIPv1.

Es importante tener en cuenta que los campos que siempre van en cero en RIPv1 pueden ir llenos en RIPv2; esto ha sido tenido en cuenta y la clase *DireccionACompartir* trata estos campos como valores variables y no como ceros estáticos, de tal modo que para el equipo que trabaje la implementación de RIPv2 en KivaNS le sea de utilidad este trabajo realizado.

Para entender más claramente el encapsulamiento que ofrece la clase *DireccionACompartir*, se puede mirar la imagen siguiente:



Marcela Vélez Pulgarín **mvelezpu@eafit.edu.co**
Juan Pablo Vergara Villarraga **jvergar7@eafit.edu.co**

Figura 10.9: Representación UML de la clase DireccionACompartir.
Fuente del Autor.

10.9 Implementación de los scripts de configuración

Para el almacenamiento de los scripts planteados dentro de los objetivos del trabajo se optó por realizar la modificación exclusivamente en la interfaz gráfica, puesto que esta funcionalidad no hace parte de la simulación.

Se tomó como base para la generación de los scripts los archivos que se pueden generar desde los equipos CISCO para guardar la configuración del enrutador. Estos sólo se habilitaron para los equipos enrutadores en KivaNS y almacenan información de configuración como interfaces conectadas, direccionamiento IP, rata de reloj para interfaces seriales, activación de RIP, entre otros.

Durante la generación de este punto, se encontró un inconveniente puesto que KivaNS no tiene implementadas Redes Punto a Punto o nubes WAN, por lo que no era posible determinar si una interfaz era serial o fastethernet. Este problema se cubrió dejando en el estudiante la responsabilidad de seleccionar por cada interfaz si es serial y, en caso de serlo, si hace las veces de DCE[18] y por lo tanto debe configurarse la rata de reloj.

En el laboratorio de Telemática se encuentran actualmente dos series de enrutadores CISCO, la serie 1700 y la 2600. Como existe una mayor cantidad de serie 1700 que de 2600 se determinó que los scripts soportaran la serie 1700. La diferencia (a nivel de comandos de configuración) que poseen es que la serie 2600 cuenta con dos slots para las interfaces, por lo que cada una de ellas se identifica por una dupla que indica el slot en el que se encuentra y el número de la interfaz dentro del mismo, mientras que en la serie 1700 las interfaces se identifican con un solo número.

Los scripts quedan almacenados en la ruta donde están la aplicación en una carpeta llamada "Scripts", en el Anexo V puede observarse un script generado.

10.10 Análisis del impacto

[18] Data Communication Equipment

Para analizar el impacto de la herramienta dentro de la universidad se realizó un laboratorio con algunos estudiantes que cursan Telemática I durante el primer semestre de 2007 y que previamente habían realizado el laboratorio con los dispositivos físicos. Luego de realizado el laboratorio con KivaNS se realizó una encuesta siguiendo los lineamientos expuestos dentro del marco teórico (Anexo II).

El tamaño de la muestra se determinó de 11 de una población de 35 que estaba cursando la materia, lo cual corresponde al 31,4% del total de estudiantes. Se realizó un muestreo aleatorio simple dado que todos los estudiantes tenían la misma probabilidad, por lo tanto con la ayuda de los profesores de la materia se solicitó un grupo de voluntarios y fueron seleccionados aleatoriamente.

Las variables a evaluar para el análisis de este tema fueron:
1. Tiempo de realización del laboratorio.
2. Riesgo de los dispositivos físicos.

Se debe tener en cuenta que la encuesta[19] abordó dos frentes diferentes, a saber: KivaNS como herramienta de software y KivaNS en el desarrollo de laboratorios y prácticas. Si bien se expone la tabulación de los resultados de todas las preguntas, se realizaron gráficos de torta de las 4 preguntas más representativas de cada uno de estos grupos anteriormente especificados.

Respuestas a las preguntas realizadas

Las siguientes son las respuestas dadas por los estudiantes a cada una de las preguntas realizadas, resumidas por la cantidad de respuestas por cada posible respuesta:

KivaNS como herramienta de Software

Número de la pregunta	**1**	**2**	**3**	**4**
En gran medida	5	5	2	1
En buena medida	6	4	7	3
Lo suficiente	0	2	2	5
Poco	0	0	0	1
Prácticamente nada	0	0	0	1

[19] Diríjase a la sección de anexos en la cual se indican las preguntas realizadas a los estudiantes.

Número de la pregunta	**5**	**6**	**7**	**8**	**9**
En gran medida	5	6	8	3	8
En buena medida	5	5	2	7	2
Lo suficiente	1	0	1	1	1
Poco	0	0	0	0	0
Prácticamente nada	0	0	0	0	0

KivaNS como herramienta de apoyo en la docencia

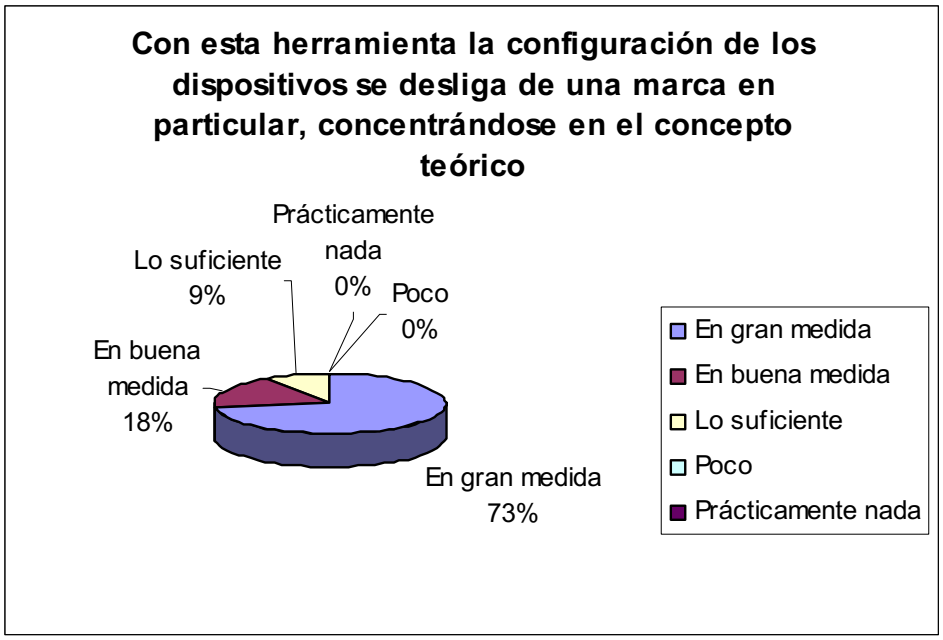
Número de la pregunta	**1**	**2**	**3**	**4**	**5**
En gran medida	5	8	8	7	5
En buena medida	3	2	3	2	5
Lo suficiente	3	1	0	2	1
Poco	0	0	0	0	0
Prácticamente nada	0	0	0	0	0

Número de la pregunta	**6**	**7**
En gran medida	7	6
En buena medida	4	5
Lo suficiente	0	0
Poco	0	0
Prácticamente nada	0	0

Gráficos de preguntas principales

A continuación se muestran gráficamente los resultados de las preguntas más importantes de cada frente y su respectivo análisis.

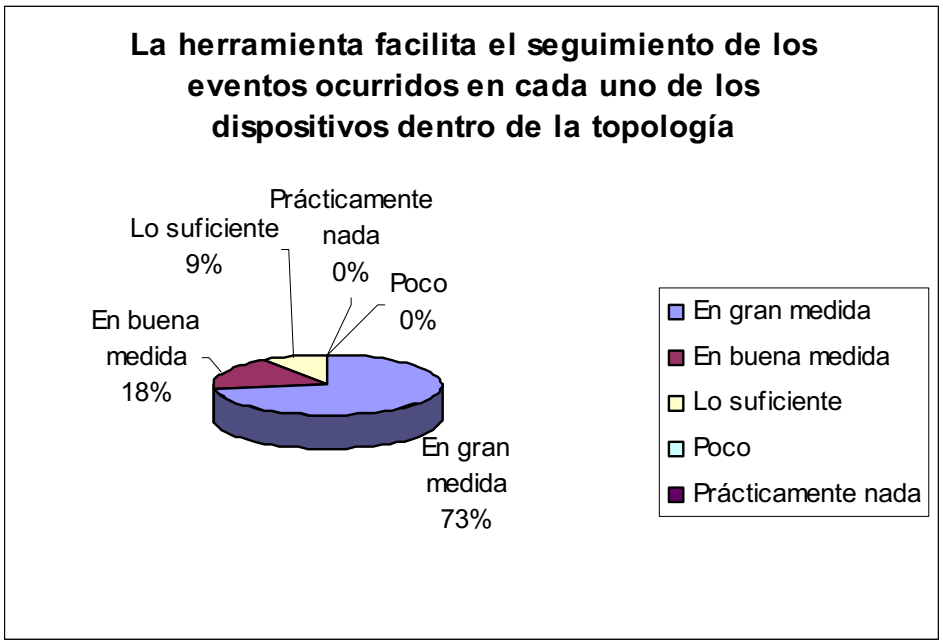
KivaNS como herramienta de Software



A lo largo de este documento se ha querido dejar claro que KivaNS es una aplicación que tiene el propósito de ser libre de marcas que implementen protocolos del estándar OSI o su implementación particular TCP/IP, de tal forma que el estudiante que la utilice se relaciona más directamente con los conceptos que con la implementación particular de estos protocolos. Esta pregunta cuestiona si KivaNS logra ese objetivo; los estudiantes en un 73% de sus respuestas nos indicaron que en gran medida se logra desligar una

herramienta de simulación de una marca particular, acercándolos de manera práctica a los conceptos como tal de la teoría de enrutamiento dinámico.

El 27% restante de las respuestas se encuentra en las categorías *Lo suficiente* y *En buena medida*, evidenciando esto que para la totalidad de los estudiantes encuestados KivaNS constituye una herramienta que logra su objetivo en este aspecto.



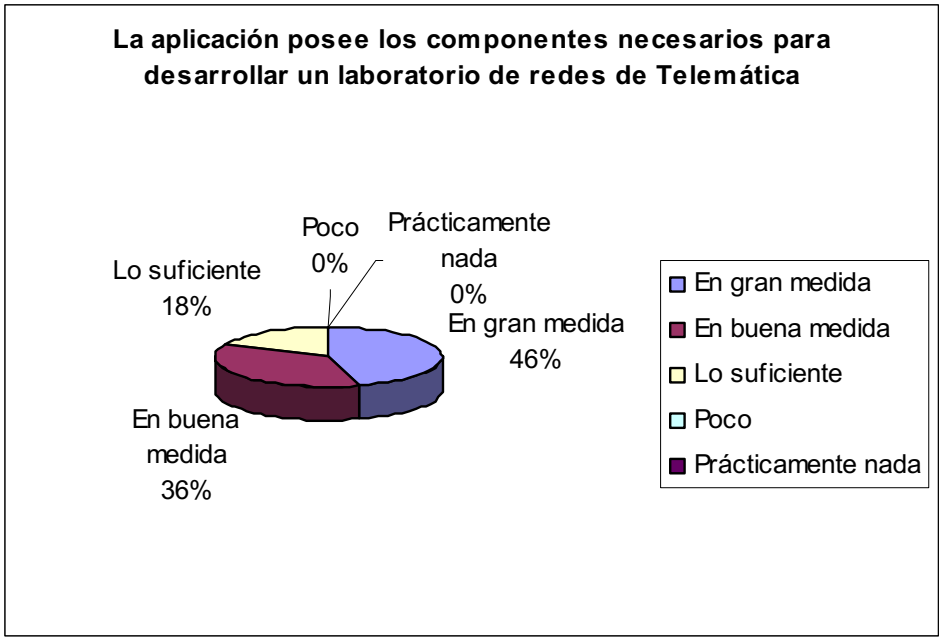
KivaNS en su versión original, ofrece la funcionalidad del rastreo de eventos ocurridos durante una simulación tanto en toda la topología como discriminado por cada equipo en la topología. Una vez se planteó el proyecto de diseñar e implementar un módulo de enrutamiento dinámico, se planteó como requerimiento de este desarrollo que este componente nuevo tendría que integrarse con la plataforma general de KivaNS generando el log de eventos ocurridos a en sus determinados niveles de la arquitectura TCP/IP implementados.

Con los estudiantes se realizó una simulación que les permitiera observar el funcionamiento del nuevo módulo, y en esta actividad se les solicitó que observaran los eventos registrados tanto en la topología como en cada

equipo de la misma y que dieran su percepción de la implementación de este requerimiento mediante la respuesta a esta pregunta.

Como se puede observar, el 73% de los estudiantes está de acuerdo en que el registro de eventos es fácilmente observable y que esta observación permite realizar un seguimiento a los eventos que genera el nuevo módulo RIP V1 junto con el componente del protocolo UDP.

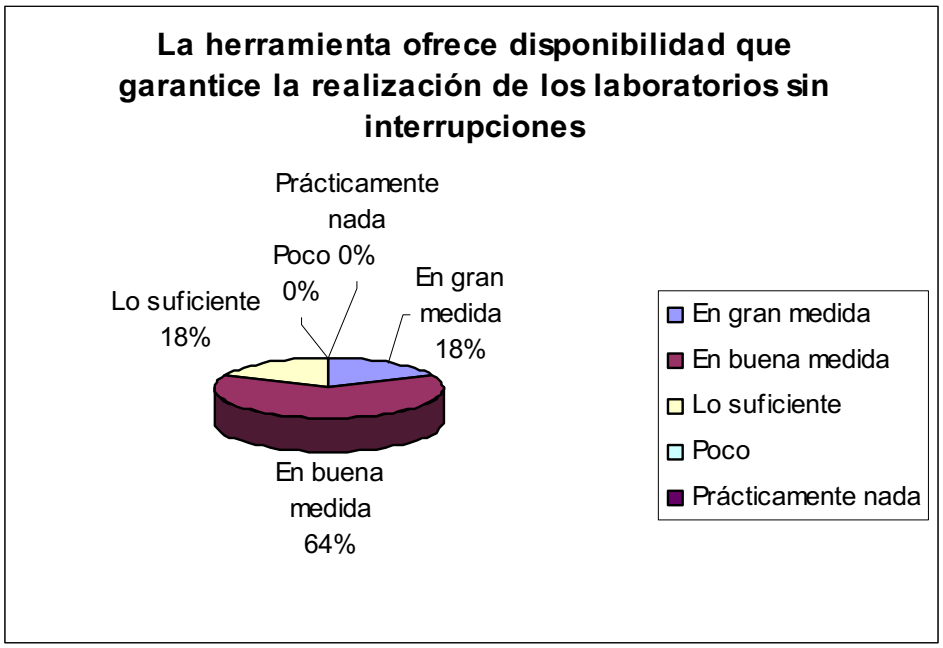
Nuevamente, el 27% restante se encuentra en las categorías *En buena medida* y *Lo suficiente*, lo que nos indica que este requerimiento satisface las necesidades de los usuarios finales y aporta enormemente a la funcionalidad de KivaNS como simulador de redes IP.



KivaNS es un software que ofrece elementos como dispositivos y tipos de enlace de diferentes clases, los cuales permiten simular topologías en diferentes condiciones de conectividad. Esta pregunta está orientada a evaluar la flexibilidad que ofrece KivaNS como simulador de redes IP para realizar un laboratorio de redes, teniendo en cuenta que la herramienta ahora contiene componentes como el nuevo módulo RIP V1.

En esta oportunidad, los estudiantes percibieron la ausencia de muchos componentes que ofrecen otros simuladores como Packet Tracer como

conmutadores de nivel 3 y enlaces WAN de diferentes tipos. Esta percepción por parte de los estudiantes, hace que la categoría *En buena medida* tenga mayor porcentaje que en las preguntas anteriormente expuestas; sin embargo, en general, los estudiantes opinan que los componentes que ofrece KivaNS si son suficientes para elaborar un laboratorio de redes, teniendo en cuenta que ahora cuenta con un módulo de enrutamiento dinámico, el cual hace del simulador una herramienta más fuerte y con mayor número de funcionalidades.

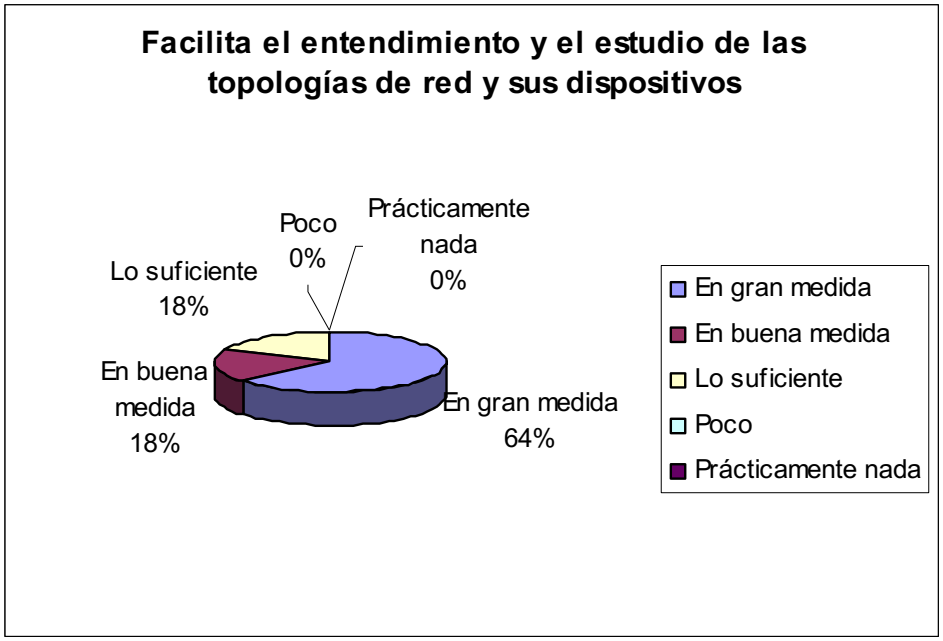


Esta pregunta se orientó al aspecto de calidad, en general, de la aplicación y su nuevo componente; al realizar la pregunta por la garantía de no interrupciones durante la ejecución de una simulación, se quiso cuestionar la percepción de los estudiantes de la ausencia de errores en tiempo de ejecución de la aplicación y el flujo normal de ejecución de este simulador.

Con respecto a este aspecto, los estudiantes opinaron que *En buena medida* se pueden realizar laboratorios sin interrupciones utilizando KivaNS como simulador de apoyo; un 18% se ubica tanto en la categoría *En gran medida* como en *Lo suficiente* lo que evidencia que si bien fue fácil y fluida la ejecución de la simulación realizada con los estudiantes, ésta no obtuvo el mayor grado de fluidez. Esto se puede atribuir a que los estudiantes manejaban por primera vez la herramienta y como es normal, se presentan

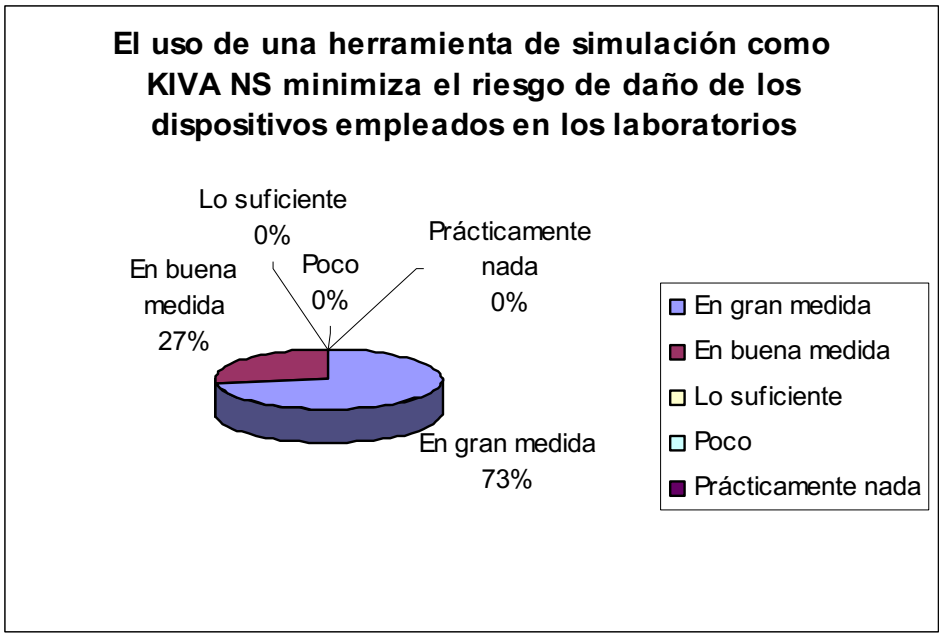
inquietudes acerca de la operación en la aplicación, lo que pudo reflejarse en sus respuestas. Sin embargo, consideramos que un 64% en la categoría *En buena medida* es un porcentaje altamente aceptable para esta aplicación que no es comercial.

KivaNS en el desarrollo de laboratorios y prácticas



Esta pregunta es la principal de este frente pues valida con los estudiantes de Telemática I si la herramienta como tal les facilita la compresión de las topologías de red y sus dispositivos en sus diferentes aspectos. KivaNS como objetivo principal tiene el participar de manera positiva en el proceso de enseñanza de diferentes aspectos de topologías de redes IP y los resultados de esta pregunta materializan el hecho de haberlo logrado. Un 64% de los estudiantes está de acuerdo con que en gran medida la herramienta les ayuda a la comprensión de conceptos de las topologías de redes IP, y con el registro del detalle de los eventos ocurridos en cada simulación ayuda a comprender de una manera clara lo que sucede cuando se efectúa el envío de un paquete IP desde una máquina hacia otra en una red.

El 36% restante se reparte en un 18% para la categoría *En buena medida* y el otro 18% se ubica en *Lo suficiente*, lo que deja ver que el impacto en la comprensión de conceptos de topologías y envíos de paquetes, incluyendo conceptos de enrutamiento dinámico, es fuertemente apoyado por herramientas como KivaNS.

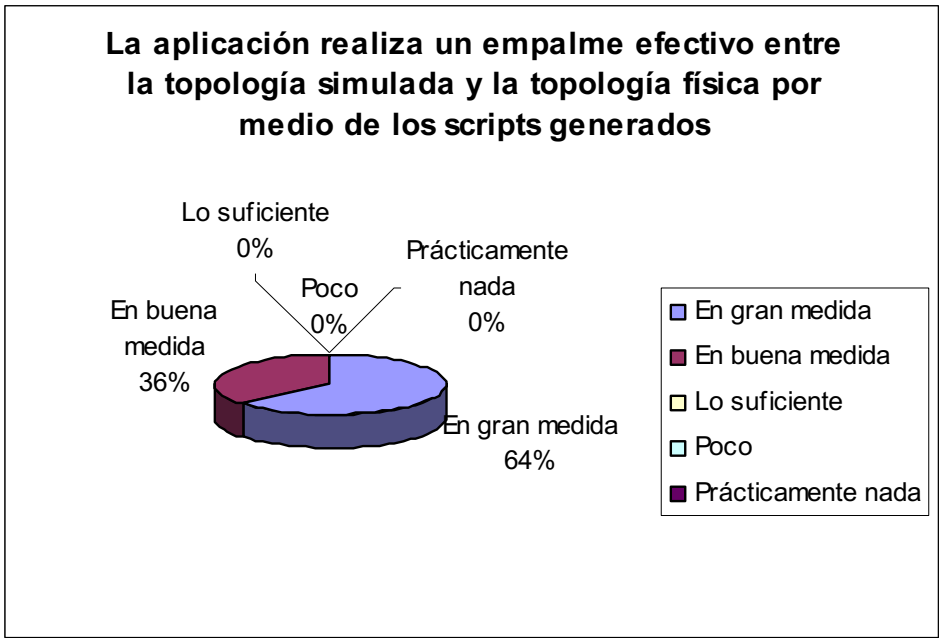


Esta pregunta apunta hacia una de las motivaciones del fortalecimiento de la plataforma KivaNS con un módulo de enrutamiento dinámico y su implantación en el laboratorio de Telemática de la universidad EAFIT, pues como bien se puede leer en el apartado *Justificación* de este documento, el riesgo de dañar los equipos del laboratorio debido a su mal uso por desconocimiento de su manejo es alto, sobretodo en manos de estudiantes que comienzan a estudiar estos conceptos; este riesgo es alto debido a que muchas veces se procede de manera inadecuada con los comandos y con la configuración deseada para un laboratorio en los primeros laboratorios de la materia, en los cuales los estudiantes aún son novatos en el manejo de los enrutadores, conmutadores, etc.

Si se realiza una primera aproximación a la configuración de una red real en un simulador como KivaNS, el riesgo de daño de los equipo en laboratorios posteriores se disminuye *En gran medida* de acuerdo al 73% de los estudiantes, mientras que el 27% restante opina que se disminuiría *En buena medida*.

Esta disminución se debe a que los estudiantes, luego de realizar algún par de ejercicios en una herramienta de simulación como KivaNS, se sienten más familiarizados tanto con la configuración de una red diseñada por ellos mismos, como con los conceptos teóricos aplicados para dicho diseño; una

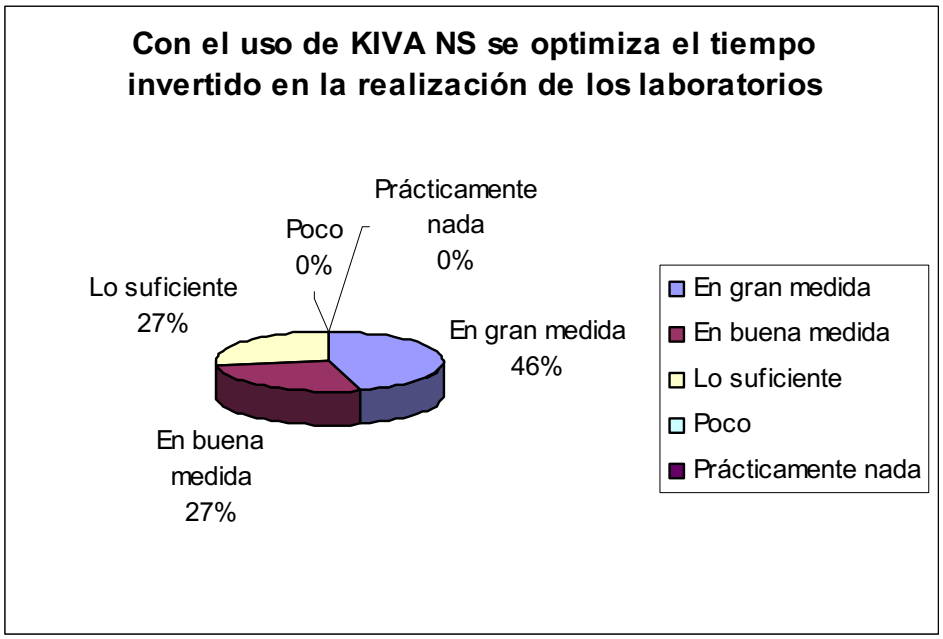
vez se realizan estos ejercicios y su correcta comprensión, la disminución del riesgo en los equipos reales se disminuiría y el aprovechamiento de las prácticas con equipos reales sería mayor.



Como vínculo entre una herramienta de simulación y los equipos reales del laboratorio, KivaNS ofrece la opción de exportar la configuración de los equipos del simulador a archivos utilizables para configurar equipos CISCO reales. Esta pregunta entonces se orienta a validar la utilidad de este empalme entre la simulación y equipos reales mediante los scripts generados por el simulador.

Los estudiantes en un 64% piensan que este empalme se realiza con satisfacción *En gran medida* y el 36% restante de los estudiantes piensan que *En buena medida* KivaNS empalma lo realizado en una simulación con configuraciones de equipos reales.

Este componente de KivaNS hace que el simulador no se quede en la sola simulación de una topología y de unos envíos de paquetes en la misma, sino que ofrece al estudiante la oportunidad de llevar cada simulación a equipos reales y observar cómo suceden las cosas que observó en el simulador, en una situación real.



Finalmente, y con un enfoque holístico de KivaNS como herramienta de apoyo en la educación, se realizó esta pregunta, la cual otorga información valiosa acerca de la realización de prácticas con simuladores de topologías de red, en este caso KivaNS, previas a la realización de laboratorios con equipos reales y su impacto en estas últimas.

El 46% de los estudiantes indicaron que realizar estas prácticas previas se vería reflejado *En gran medida* en el tiempo que toma una práctica de laboratorio con equipos reales; nuevamente, se indica que esta conclusión tiene que ver con el fortalecimiento que ofrece KivaNS en la comprensión de conceptos teóricos de topologías de red, entre ellos el enrutamiento dinámico, el cual se reflejaría en la rápida y mejor comprensión de eventos ocurridos en una topología real.

El resto de los estudiantes pensaron, en una proporción del 27% que la realización de prácticas previas con simuladores como KivaNS se reflejarían tanto *En buena medida* como *Lo suficiente* en el tiempo de realización de prácticas sobre equipos reales.

11. Conclusiones

De acuerdo a los resultados de las encuestas realizadas a los estudiantes para determinar el impacto de la utilización de una herramienta de software como KivaNS, se concluye que un acercamiento inicial a una configuración de redes mediante simuladores de este tipo es considerado por los estudiantes como positivo en aspectos tales como: el nivel de comprensión y el tiempo de ejecución en prácticas realizadas sobre topologías reales y para la comprensión de conceptos de enrutamiento dinámico, desligados de tecnologías particulares que las implementen.

Ahora bien, mediante la utilización de KivaNS como simulador de topologías de red, concluimos que esta herramienta es en gran medida aceptada y apoya el proceso de aprendizaje especialmente en la materia TCP/IP, debido a la facilidad que ofrece para la comprensión de conceptos teóricos de enrutamiento dinámico mediante el protocolo RIP V1 y demás eventos ocurridos tanto en la pila de protocolos de la arquitectura TCP/IP como en el nivel físico en redes Ethernet.

Esta percepción positiva de KivaNS también evidenció una disminución del riesgo que existe actualmente de posibles daños de los enrutadores y conmutadores del laboratorio de Telemática de la Universidad Eafit, ya que al tener un primer contacto con la configuración de topologías de red mediante simuladores se logra una aproximación a lo que son las prácticas de laboratorio, sin intervenir directamente sobre los equipos físicos y observando con gran fidelidad el comportamiento de los paquetes que transitan en la red configurada.

Acerca de KivaNS como herramienta de Software, obtenemos como conclusión que las buenas prácticas de diseño y construcción de aplicaciones, especialmente en el tema de Arquitectura y Patrones, no son exclusivas de últimas tendencias en Ingeniería de Software; KivaNS es una muestra clara de un patrón Modelo Vista Controlador (MVC) perfectamente aplicado a una aplicación stand-alone que se encuentra lejos de aplicaciones en contextos empresariales.

Acerca del protocolo RIP V1, concluimos que su validez como protocolo de enrutamiento radica en la necesidad de la solución de enrutamiento solicitada, dada ésta en términos de la complejidad de la topología sobre la cual se quiera implantar dicha solución. RIP V1 está en capacidad de proveer soluciones óptimas de enrutamiento en la medida en que las condiciones de la red como el tamaño y la presencia de bucles entre los enrutadores (o hosts en caso de ejecutar RIP) sean adecuados para su implantación; esto hace que protocolos más complejos y elaborados como OSPF o RIP V2 no sean los únicos que se tengan en cuenta como posibles soluciones de enrutamiento dinámico.

Finalmente concluimos que herramientas con licenciamiento GPL, como KivaNS, representan una oportunidad de crecimiento bilateral, tanto para la plataforma de simulación, como para la Universidad Eafit, en este caso; pues mientras la primera se ve beneficiada en su robustecimiento funcional, la Institución se está proveyendo a si misma una herramienta de gran aporte en el proceso de aprendizaje y enseñanza de conceptos básicos de redes de datos, sin dejar de lado, el prestigio que representa que sus estudiantes sean quienes implementen estas nuevas funcionalidades.

12. Trabajos Futuros

Dada la amplia gama de protocolos y tipos de redes que se encuentran actualmente en el mercado e incluso las nuevas tecnologías que se presentan diariamente, la aplicación esta abierta para incluir más módulos que sigan apoyando los diferentes temas que se enseñan en las materias de la línea de telemática. Entre ellas se recomiendan:

- Desarrollo del protocolo TCP, incluyendo la generación de la PDU de este nivel, el trabajo con ventanas deslizantes y la sincronización entre equipos finales que en este momento no se realiza en la aplicación.
- Desarrollo de aplicaciones sobre nivel UDP, como TFTP
- Desarrollo de aplicaciones sobre el nivel TCP como Web, SMTP
- IPV6.
- WLAN
- Otros protocolos de enrutamiento como OSPF, RIP v2.
- Implementar DHCP
- Implementar el manejo de VLAN's en los conmutadores, etc.

13. Referencias Bibliográficas

[1] Bellman, R. E., "Dynamic Programming", Princeton University Press, Princeton, N.J., 1957.

[2] http://www.faqs.org/rfcs/rfc1058.html

[3] http://www.faqs.org/rfcs/rfc2453.html

[4] http://www.faqs.org/rfcs/rfc2080.html

[5] http://home.ubalt.edu/ntsbarsh/simulation/sim.htm
Universidad de Baltimore, trabajo sobre software de simulación.

[6] http://www.faqs.org/rfcs/rfc768.html

[7] R. Jonson; Patricia Kuby, Estadística elemental: Lo esencial, 2004.

[8] Estadísticas de Berenson y Levin. Sistema de Gestión de los resultados y el impacto, FIDA, Enero de 2005

[9] http://disclab.ua.es/kiva/
Página oficial de Kiva

[10] Steidley, Carl; Bachnak, Rafia. Software and Hardware Development for a Virtual Laboratory. Department of Computing and Mathematical Sciences, Texas A&M University, 2005.

[11] Kele, Bruce; De Horta, Ain Y.; Box, Ivona. VELNET (Virtual Environment for Learning Networking). University of Western Sydney, Australia, 2004.

[12] Baumgartnert, Florian; Braun, Torsten; Kurt, Eveline; Weyland, Attila. Virtual Routers: A Tool for Networking Research and Education. Universität Bern, July 2003.

[13] Zorzona Martínez, Eduardo. Aprendizaje con Simuladores. Aplicación a las redes de comunicaciones.

[14] http://nsl.csie.nctu.edu.tw/NCTUnsReferences/capitulo4.pdf
Comparación de herramientas de simulación.

[15] http://www.boson.com/
Página Oficial de Boson: NetSim.

[16] http://www.opnet.com/
Página Oficial de OPNET.

[17] Grady Booch, Robert Martin y James Newkirk. Object Oriented Analysis and Design with Applications, 2a edición, Addison-Wesley, 1998

[18] Huidobro Moya, José Manuel. Redes y Servicios de Telecomunicaciones.

14. Anexos

V RFC relacionados con la Arquitectura TCP/IP

Número de RFC	Título
768	Protocolo de datagramas de usuario (UDP, User Datagram Protocol)
783	Protocolo trivial de transferencia de archivos (TFTP, Trivial File Transfer Protocol)
791	Protocolo Internet (IP, Internet Protocol)
792	Protocolo de mensajes de control de Internet (ICMP, Internet Control Message Protocol)
793	Protocolo de control de transporte (TCP, Transmission Control Protocol)
816	Aislamiento y recuperación de errores (Fault Isolation and Recovery)
826	Protocolo de resolución de direcciones (ARP, Address Resolution Protocol)
854	Protocolo Telnet (TELNET)
862	Protocolo Eco (ECHO)
863	Protocolo Descartar (DISCARD)
864	Protocolo Generador de caracteres (CHARGEN)
865	Protocolo Cita del día (QUOTE)
867	Protocolo Día (DAYTIME)
894	IP sobre Ethernet (IP over Ethernet)
919	Difusión de datagramas de Internet (Broadcasting Internet Datagrams)
922	Difusión de datagramas de Internet en presencia de subredes (Broadcasting Internet Datagrams in the Presence of Subnets)
950	Procedimiento de subredes estándar de Internet (Internet Standard Subnetting Procedure)
959	Protocolo de transferencia de archivos (FTP, File Transfer Protocol)
1001	Estándar de protocolo para un servicio NetBIOS en un transporte TCP/UDP: conceptos y métodos

Número de RFC	Título
	(Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods)
1002	Estándar de protocolo para un servicio NetBIOS en un transporte TCP/UDP: especificaciones detalladas (Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications)
1009	Requisitos para puertas de enlace de Internet (Requirements for Internet Gateways)
1034	Nombres de dominio: conceptos y servicios
1035	Nombres de dominio: implementación y especificación
1042	IP sobre Token Ring (IP over Token Ring)
1055	Procedimientos no estándar para la transmisión de datagramas IP a través de líneas serie: SLIP (A Nonstandard for Transmission of IP Datagrams Over Serial Lines: SLIP)
1065	Estructura e identificación de información de administración para Internet basada en TCP/IP (Structure and Identification of Management Information for TCP/IP-based Internets)
1112	Protocolo de administración de grupos de Internet (IGMP, Internet Group Management Protocol)
1122	Requisitos para hosts de Internet: niveles de comunicación
1123	Requisitos para hosts de Internet: aplicación y compatibilidad
1144	Compresión de encabezados TCP/IP para vínculos serie de baja velocidad (Compressing TCP/IP Headers for Low-Speed Serial Links)
1157	Protocolo simple de administración de redes (SNMP, Simple Network Management Protocol)
1179	Protocolo demonio de impresora de líneas (Line Printer Daemon Protocol)
1188	IP sobre FDDI (IP over FDDI)
1191	Detector de ruta de acceso MTU (Path MTU Discovery)
1201	IP sobre ARCNET (IP over ARCNET)
1256	Mensajes de descubrimiento de enrutadores ICMP (ICMP Router Discovery Messages)
1323	Extensiones de TCP para obtener un alto rendimiento (TCP Extensions for High Performance)
1332	Protocolo de control del protocolo Internet PPP (IPCP, PPP Internet Protocol Control Protocol)

Número de RFC	Título
1518	Arquitectura para asignación de direcciones IP con CIDR (An Architecture for IP Address Allocation with CIDR)
1519	Enrutamiento entre dominios sin clase (CIDR, Classless Inter-Domain Routing): estrategia para asignar y agregar direcciones (An Address Assignment and Aggregation Strategy)
1534	Interacción entre DHCP y BOOTP (Interoperation Between DHCP and BOOTP)
1542	Aclaraciones y extensiones del protocolo Bootstrap (Clarifications and Extensions for the Bootstrap Protocol)
1552	Protocolo de control de intercambio de paquetes entre redes en PPP (IPXCP, PPP Internetwork Packet Exchange Control Protocol)
1661	Protocolo punto a punto (PPP)
1662	PPP en tramas HDLC (PPP in HDLC-like Framing)
1748	MIB IEEE 802.5 con SMIv2 (IEEE 802.5 MIB using SMIv2)
1749	MIB de enrutamiento de origen de emisora IEEE 802.5 con SMIv2 (IEEE 802.5 Station Source Routing MIB using SMIv2)
1812	Requisitos para enrutadores IP versión 4 (Requirements for IP Version 4 Routers)
1828	Autenticación de IP mediante MD5 con clave (IP Authentication using Keyed MD5)
1829	Transformación DES-CBC de ESP (ESP DES-CBC Transform)
1851	Transformación Triple DES-CBC de ESP (ESP Triple DES-CBC Transform)
1852	Autenticación de IP mediante SHA con claves (IP Authentication using Keyed SHA)
1878	Tabla de subredes de longitud variable para IPv4 (Variable Length Subnet Table For IPv4)
1886	Extensiones DNS para admitir IP Versión 6 (DNS Extensions to Support IP Version 6)
1994	Protocolo de autenticación por desafío mutuo (CHAP, Challenge Handshake Authentication Protocol) en PPP
1995	Transferencia incremental de zona en DNS (Incremental Zone Transfer in DNS)
1996	Un mecanismo para la notificación en DNS de los cambios de zona (A Mechanism for Prompt DNS

Número de RFC	Título
	Notification of Zone Changes)
2018	Opciones de confirmación selectiva de TCP (TCP Selective Acknowledgment Options)
2085	Autenticación IP de HMAC-MD5 con prevención de repetición (HMAC-MD5 IP Authentication with Replay Prevention)
2104	HMAC: Hash con claves para la autenticación de mensajes (Keyed Hashing for Message Authentication)
2131	Protocolo de configuración dinámica de host (DHCP, Dynamic Host Configuration Protocol)
2136	Actualizaciones dinámicas en el Sistema de nombres de dominio (DNS UPDATE) (Dynamic Updates in the Domain Name System)
2181	Aclaraciones respecto a la especificación DNS (Clarifications to the DNS Specification)
2236	Protocolo de administración de grupos de Internet, versión 2 (Internet Group Management Protocol, Version 2)
2308	Almacenamiento en caché de las consultas DNS (DNS NCACHE) negativas (Negative Caching of DNS Queries)
2401	Arquitectura de seguridad para el Protocolo Internet (Security Architecture for the Internet Protocol)
2402	Encabezado de autenticación IP (IP Authentication Header)
2406	Carga de seguridad encapsuladora de IP (ESP, Encapsulating Security Payload)
2581	Control de la congestión en TCP (TCP Congestion Control)

V Encuesta a Estudiantes

VLABS
Juan Pablo Vergara y Marcela Vélez

El trabajo de grado realizado por nosotros, pretende ajustar una herramienta para la simulación de topologías de red que permita servir de apoyo a los estudiantes en el estudio de esta área. El objetivo de esta encuesta es analizar el impacto de la herramienta en los estudiantes que cursan actualmente la materia de Telemática I para determinar que tan efectiva podría ser como apoyo dentro de las diferentes materias del área de Telemática.

Marcela Vélez Pulgarín mvelezpu@eafit.edu.co
Juan Pablo Vergara Villarraga jvergar7@eafit.edu.co

KivaNS permite crear topologías de red y configurar los diferentes dispositivos con que cuenta para realizar así, simulaciones pertinentes de los objetos físicos con los que cuenta el laboratorio. Adicionalmente al finalizar la configuración de los enrutadores, ésta puede ser almacenada en archivos de texto para posteriormente configurar los dispositivos CISCO reales que se encuentran en el laboratorio.

Califique de 1 a 5 los siguientes puntos teniendo en cuanto que cinco es la calificación máxima. 5 → En gran medida. 4 → En buena medida. 3 → Lo suficiente. 2 → Poco. 1 → Prácticamente nada.

- **KivaNS como herramienta de software**

	5	4	3	2	1
La interfaz gráfica que ofrece la aplicación es amigable.					
La aplicación posee los componentes necesarios para desarrollar un laboratorio de redes de Telemática.					
La herramienta ofrece disponibilidad que garantice la realización de los laboratorios sin interrupciones.					
Se cuenta con una ayuda que permita resolver dudas sobre el uso de la herramienta.					
La herramienta permite realizar simulaciones claras y reales sobre el entorno físico que se va a montar.					
La portabilidad de la herramienta ayuda a disminuir la dependencia de un lugar físico para el montaje de una topología.					
Con esta herramienta la configuración de los dispositivos se desliga de una marca en particular, concentrándose en el concepto teórico.					
Permite simular dentro de una topología un cambio en algún dispositivo de forma efectiva.					
La herramienta facilita el seguimiento de los eventos ocurridos en cada uno de los dispositivos dentro de la topología					

- **KivaNS en el desarrollo de laboratorios y prácticas**

	5	4	3	2	1
Con el uso de KivaNS se optimiza el tiempo invertido en la realización de los laboratorios.					
La generación de los scripts de configuración de los equipos ayuda al montaje del laboratorio real.					
El uso de una herramienta de simulación como KivaNS minimiza el riesgo de daño de los dispositivos empleados en los laboratorios.					
Facilita el entendimiento y el estudio de las topologías de red y sus dispositivos.					
Con KivaNS se potencializa la educación no presencial para que estudiantes de diversos entornos puedan simular topologías sin la necesidad de los dispositivos físicos.					
La aplicación realiza un empalme efectivo entre la topología simulada y la topología física por medio de los scripts generados.					
Permite analizar y estudiar la convergencia de la red a través de un protocolo de enrutamiento dinámico tal como RIP v1.					

- **Observaciones y comentarios adicionales**

_____
_____
_____
_____
_____
_____

Marcela Vélez Pulgarín **mvelezpu@eafit.edu.co**
Juan Pablo Vergara Villarraga **jvergar7@eafit.edu.co**

V Cronograma de trabajo

CRONOGRAMA

Fecha de comienzo del proyecto: lun 17/07/06
Fecha de fin del proyecto: sáb 24/03/07

Tareas con asignaciones

Id	Nombre de tarea	Duración	Comienzo	Fin
0	**TRABAJO**		sáb 22/07/06	sáb 24/03/07
1	**Iniciación**	5,63 días	sáb 22/07/06	dom 06/08/06
2	Conformación del equipo de trabajo	7 horas	sáb 22/07/06	sáb 22/07/06
3	Analisis de estado del arte	10 horas	sáb 22/07/06	sáb 29/07/06
4	Realización del estudio de viabilidad	6 horas	sáb 29/07/06	sáb 29/07/06
5	Acta de iniciación	2 horas	sáb 29/07/06	dom 30/07/06
6	Realización del anteproyecto	20 horas	dom 30/07/06	dom 06/08/06
7	Iniciación terminada	0 días	dom 06/08/06	dom 06/08/06
8	**Planeación**	0,88 días	sáb 26/08/06	sáb 26/08/06
9	Generación del WBS Detallado	3 horas	sáb 26/08/06	sáb 26/08/06
10	Realización del Presupuesto detallatado	2 horas	sáb 26/08/06	sáb 26/08/06
11	Realización del documento de análisis de riesgos	2 horas	sáb 26/08/06	sáb 26/08/06
12	Planeación terminada	0 días	sáb 26/08/06	sáb 26/08/06
13	**Ejecución**	59,5 días	sáb 26/08/06	sáb 24/03/07
14	Levantamiento de requisitos del sistema	15 horas	sáb 26/08/06	sáb 02/09/06
15	Realización del documento de casos de uso	15 horas	sáb 02/09/06	sáb 09/09/06

16	**Modelamiento UML**	3,5 días	sáb 09/09/06	sáb 23/09/06
17	Diagrama de objetos	4 horas	sáb 09/09/06	dom 10/09/06
18	Diagrama de clases	4 horas	dom 10/09/06	dom 10/09/06
19	Diagrama de colaboración	4 horas	dom 10/09/06	sáb 16/09/06
20	Diagramas de secuencias	4 horas	sáb 16/09/06	sáb 16/09/06
21	Diagramas de actividades	4 horas	sáb 16/09/06	dom 17/09/06
22	Diagramas de componentes	4 horas	dom 17/09/06	dom 17/09/06
23	Diagramas de despliegue	4 horas	dom 17/09/06	sáb 23/09/06
24	Modelamiento terminado	0 días	sáb 23/09/06	sáb 23/09/06
25	**Desarrollo del sistema**	44,5 días	sáb 23/09/06	sáb 24/02/07
26	**Módulo UDP**	3,75 días	sáb 23/09/06	dom 01/10/06
27	Desarrollo de manejo de puertos	20 horas	sáb 23/09/06	sáb 30/09/06
28	Nivel UDP y Datagrama UDP	5 horas	sáb 30/09/06	dom 01/10/06
29	Comunicación del nivel	5 horas	dom 01/10/06	dom 01/10/06
30	Módulo UDP teminado	0 días	dom 01/10/06	dom 01/10/06
31	**Módulo RIP v1**	8,13 días	dom 01/10/06	dom 29/10/06
32	Desarrollo del Nivel RIP simulador	40 horas	dom 01/10/06	sáb 21/10/06
33	Desarrollo nivel RIPv1 interfaz gráfica	10 horas	sáb 21/10/06	sáb 28/10/06
34	Pruebas y ajustes RIP v1	15 horas	sáb 28/10/06	dom 29/10/06
35	Módulo RIP v1 terminado	0 días	dom 29/10/06	dom 29/10/06
36	**Módulo nueva configuración de dispositivos**	8,13 días	dom 01/10/06	dom 29/10/06

37	Desarrollo del módulo configuración de dispositivos	40 horas	dom 01/10/06	sáb 21/10/06
38	Pruebas módulo configuración de dispositivos	10 horas	sáb 21/10/06	sáb 28/10/06
39	Ajustes módulo configuración de dispositivos	15 horas	sáb 28/10/06	dom 29/10/06
40	Configuración de dispositivos terminado	0 días	dom 29/10/06	dom 29/10/06
41	**Módulo pruebas a las topologías**	8,13 días	sáb 04/11/06	sáb 02/12/06
42	Desarrollo del módulo pruebas a las topologías	40 horas	sáb 04/11/06	sáb 18/11/06
43	Pruebas módulo pruebas a las topologías	10 horas	dom 19/11/06	sáb 25/11/06
44	Ajustes módulo pruebas a las topologías	15 horas	sáb 25/11/06	sáb 02/12/06
45	Pruebas a las topologías terminado	0 días	sáb 02/12/06	sáb 02/12/06
46	**Módulo generación de scripts**	32,63 días	sáb 04/11/06	sáb 24/02/07
47	Desarrollo del módulo generación de scripts	45 horas	sáb 04/11/06	dom 19/11/06
48	Pruebas módulo generación de scripts	12 días	dom 19/11/06	dom 31/12/06
49	Ajustes módulo generación de scripts	15 días	dom 31/12/06	sáb 24/02/07
50	Generación de scripts terminado	0 días	sáb 24/02/07	sáb 24/02/07
51	Pruebas integrales al sistema	20 horas	sáb 24/02/07	dom 04/03/07
52	**Realización de Manuales**	1,5 días	dom 04/03/07	sáb 10/03/07

53	Manual técnico	6 horas	dom 04/03/07	dom 04/03/07
54	Manual de usuario	6 horas	dom 04/03/07	sáb 10/03/07
55	Realización de Manuales terminado	0 días	sáb 10/03/07	sáb 10/03/07
56	Instalación en la universidad en el laboratorio de telemática	10 horas	sáb 10/03/07	dom 11/03/07
57	Realización de encuestas de satisfacción	20 horas	dom 11/03/07	sáb 24/03/07
58	Realización de documento final	30 horas	dom 25/03/07	sáb 02/07/07
59	Control	4 horas	sáb 22/08/06	sáb 22/08/06
60	Cierre	0 horas	sáb 24/08/06	sáb 24/08/06

IV. Casos de Prueba Ejecutados

Los siguientes son los casos de prueba realizados a la aplicación.

Id	Tipo prueba	Caso de uso	Pantalla	Escenario / caso de prueba	Ejecutado	Éxito	Fracaso	Observaciones
1	Funcional	Converger mediante RIP v1	Principal de Kiva NS	Converger mediante RIP v1 en una red sin ciclos - Mediante el botón de la barra de herramientas	x	x		
2	Funcional	Converger mediante RIP v1	Principal de Kiva NS	Converger mediante RIP v1 en una red con ciclos - Mediante el botón de la barra de herramientas	x	x		
3	Funcional	Converger mediante RIP v1	Principal de Kiva NS	No converger mediante RIP v1 en una red con mas de 16 saltos - Mediante el botón de la barra de herramientas	x	x		
4	Funcional	Converger mediante RIP v1	Principal de Kiva NS	"Tumbar" un enlace y verificar que triggered updates maneje correctamente la situación - Mediante el botón de la barra de herramientas	x	x		
5	Funcional	Converger mediante RIP v1	Principal de Kiva NS	Converger mediante RIP v1 en una red sin ciclos - Mediante la opción del menú	x	x		

6	Funcional	Converger mediante RIP v1	Principal de Kiva NS	Converger mediante RIP v1 en una red con ciclos - Mediante la opción del menú	x	x		
7	Funcional	Converger mediante RIP v1	Principal de Kiva NS	No converger mediante RIP v1 en una red con mas de 16 saltos	x	x		
8	Funcional	Converger mediante RIP v1	Principal de Kiva NS	"Tumbar" un enlace y verificar que triggered updates maneje correctamente la situación - Mediante la opción del menú	x	x		
9	Funcional	Mostrar resultados de simulación	Principal de Kiva NS	Mostrar eventos de simulación	x	x		
10	Funcional	Generar Scripts de configuración de Router	Principal de Kiva NS	Generar Scripts de configuración de Router Cisco	x	x		
11	Funcional	Generar Scripts de configuración de Router	Principal de Kiva NS	Probar scripts de configuración en Router Cisco	x	x		
12	Funcional	Generar Scripts de configuración de Router	Principal de Kiva NS	Generar Script para un DTE y para un DCE	x	x		

V. Script generado por KivaNS. Medellin.txt

```
Current Configuration:
!
hostname Medellin
!
!
!
!
!
!
interface Serial0
 ip address 200.12.180.2
```

```
 speed auto
!
interface Serial1
 ip address 200.12.182.2
 speed auto
 clockrate 64000
!
interface FastEthernet0
 ip address 11.0.0.1
 speed auto
!
!
router rip
 network 200.12.180.0
 network 200.12.182.0
 network 11.0.0.0
!
!
line con 0
line aux 0
line vty 0 4
!
no scheduler allocate
end
```