TRABAJO ESPECIAL DE GRADO

ESTUDIO DE LAS TÉCNICAS DE AHORRO DE ENERGÍA EN DISPOSITIVOS 802.11 MEDIANTE EL SIMULADOR NS-3

Tutor Industrial: Ing. Roberto Sanz Gil
Prof. Guía: Dr. Carlos Moreno

Presentado ante la Ilustre
Universidad Central de Venezuela
por la Br. Borjas M., Reina E.
para optar al Título de
Ingeniero Electricista

Caracas, 2011

CONSTANCIA DE APROBACIÓN

Caracas, 03 de noviembre de 2011

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por la Bachiller Reina E. Borjas M. titulado:

"ESTUDIO DE LAS TECNICAS DE AHORRO DE ENERGIA EN DISPOSITIVOS 802.11 MEDIANTE EL SIMULADOR NS-3"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Título de Ingeniero Electricista en la mención de Comunicaciones, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. Gerlis Caropresse
Jurado

Prof. Zeldivar Bruzual
Jurado

Prof. Carlos Moreno
Prof. Guía

DEDICATORIA

Me gustaría dedicar este trabajo:

A mis padres,

A mi hermano,

A mi novio,

A mi familia,

A mis amigos,

A mi UCV,

Y a mi Dios, que se que escuchas mis oraciones.

Para ustedes, uno de mis mayores logros.

RECONOCIMIENTOS Y AGRADECIMIENTOS

Quisiera agradecer primero a Dios, por darme salud y por darme la fuerza necesaria para luchar en la vida. En ti confío. Gracias.

A mis padres. Gracias por darme la vida. Por ser los mejores padres del mundo. Por estar cuando los necesité. A ustedes que aunque estén lejos, son los primeros en preocuparse por mi educación. Siempre han sido y serán mi mayor orgullo, mi gran modelo a seguir. Los adoro.

A mi hermano por ser mi mayor cómplice en todos mis actos. Sabes que te quiero y que eres mi mejor amigo. Somos de la misma sangre, gracias por existir.

Al Prof. Tomas Fernández por ser un apoyo incondicional en toda mi trayectoria como estudiante en la UNICAN. Gracias.

Al Prof. Carlos Moreno, por ser mi profesor tutor aquí en Venezuela, y además por ser mi amigo. Gracias por creer en mí.

A Maria Auxiliadora, que no sólo has sido una de las personas más especiales que he conocido en mi trayectoria como estudiante. Eres mi amiga, y a ti te agradezco por toda la ayuda y apoyo incondicional que me has brindado durante todos estos años. Gracias, y nunca cambies.

A toda mi familia, que siempre me envían sus oraciones a diario, pidiendo por mi bienestar físico, mental y emocional.

A mis hermanas del alma: Anto, Ari, Lali y Mari, por ser las mejores amigas del mundo. Por ayudarme con sus consejos a crecer como persona y como mujer. Por su eterna compañía en la distancia. Las amo.

A mi novio Alex, por comprenderme, por apoyarme y por quererme. A ti que has decidido compartir tus días a mi lado. A ti por devolverme una sonrisa. A ti por ser el niño de mi vida. Te amo. Gracias por hacerme creer que los cuentos de hadas pueden hacerse realidad.

A mi grupo de baile, mi CDC, por creer en mi, por apoyarme en todo lo que emprendo. Por enseñarme que “si se puede” cuando te lo propones. Por enseñarme que trabajando arduamente, consigues lo que quieras. A ustedes que son el mejor grupo profesional. Los mejores bailadores. Gracias.

A mi Mereketengue, por ser únicos. Siempre serán la firma que llevo en mi corazón. Por siempre serán mis amigos.

A BSD, por su apoyo incondicional en cada paso que he dado durante mi estadía en Santander. “You are the one.”

A mis amigos: Lucy, Andres y Fran. Los amo. Y no imaginan cuanto los extraño.

A mi gente bella de la UCV, Ucevista Hoy, Ucevista Siempre.

A mi nueva familia Santanderina, por no dejarme sola, y brindarme su amistad. Los llevaré siempre en mi corazón.

A la UCV. por enseñarme todo lo necesario para crecer como profesional. Por enseñarme incluso lecciones de vida. Gracias por permitirme sentarme en tus aulas para aprender.

A la UNICAN por recibirme en sus instalaciones durante este año. Por formar parte de mi carrera. Por ayudarme a terminar mi carrera.

Borjas M., Reina E.

ESTUDIO DE LAS TÉCNICAS DE AHORRO DE ENERGÍA EN DISPOSITIVOS 802.11 MEDIANTE EL SIMULADOR NS-3

Tutor Industrial: Prof. Roberto Sanz Gil, Ingeniero y profesor titular de la Universidad de Cantabria. Coordinador en el Departamento de Ingeniería de Comunicaciones.

Palabras claves: Ns-3 Simulator, Ad-Hoc, Ahorro de Energía, Dispositivos 802.11, WiFi. Informática Verde.

Resumen: Se plantea el estudio del comportamiento de las redes de área local inalámbricas, mediante el uso de las aplicaciones del simulador NS-3, para técnicas de ahorro de energía en dispositivos 802.11. Se realiza en el menor tiempo posible simulaciones, obteniendo mediciones de potencia de recepción, de tal manera, de realizar un ajuste en la configuración de los ordenadores para que exista una disminución en la potencia de transmisión, con el menor costo posible, y haciendo un aporte significativo en el desarrollo de la informática verde. La razón de utilizar el simulador NS-3, es para el estudio de técnicas cooperativas, y su posible aplicación futura, en pruebas reales. El entorno de trabajo fue en el Sistema Operativo Ubuntu, utilizando como herramienta el programa Ns-3. En él se define un escenario de dos (2) ordenadores portátiles, con diferentes parámetros a variar. Se concluye que existe una potencia mínima necesaria para alcanzar al receptor, y que ésta va a depender de la distancia a la que se encuentre. A mayor distancia, se necesitará una mayor potencia de transmisión. El ahorro de energía ocurrirá una vez que se obtenga la potencia mínima Tx, asegurando la recepción en el nodo receptor.

CONTENIDO

Pág

LISTAS DE FIGURAS Y GRÁFICAS

FIGURAS Y GRÁFICAS Pág.

LISTAS DE TABLAS

TABLAS **Pág.**

LISTAS DE ACRÓNIMOS

- **AODV:** Ad Hoc On Demand Distance Vector
- **AP:** Access Point(Punto de Acceso)
- **API:** Application Programming Interfase (Interfase de Programación de Aplicaciones)
- **CRC:** Comprobación de Redundancia Cíclica
- **CSMA:** Carrier Sense Multiple Access. (Acceso Múltiple por Detección de Portadora).
- **GNU Gplv2:** General Public License. (Licencia Pública General)
- **GUI:** Graphical User Interface (Interfaz Grafica de Usuario)
- **ICI:** Información de Control de la Interfase. (cabecera)
- **IDU:** Unidad de Datos de la Interfase**.**
- **IEEE:** Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Electricistas y Electrónicos)
- **IPV4:**Internet Protocol Versión 4 (Protocolo de Internet)
- **IPV6:** Internet Protocol Versión 6 (Protocolo De Internet)
- **ISM:** Industrial Scientific Medical(Industria Médica Científica)
- **ISO/CEI:**Comisión Electrotécnica Internacional
- **LAN:** Local Area Network (Red de Area Local)
- **LLC:** Logical Link Control. (Control de Enlace Lógico)
- **MAC:** Médium Access Control (Acceso de Control Al Medio)
- **NS2:** Network Simulator 2. (Simulador de Redes)
- **NS3:** Network Simulator 3. (Simulador de Redes)
- **OLSR:** Optimized Link State Routing Protocol (Mecanismo Estándar de Enrutamiento Pro-Activo)
- **OSI:** Open System Interconnection(Interconexión de Sistema Abierto)
- **PCAP:** Packet Capture (Captura de Paquetes).

- **PCI:** Periphelical Component Interconet. Interconexión de Componentes Periféricos
- **PDA:** Personal Digital Assistent(Asistente Digital Personal)
- **PDU:** Protocol Data Unit (Unidad de Datos de Protocolo)
- **PHY:** Physical(Capa Física)
- **PLCP:** Physical Lawyer Convergence Protocol(Protocolo de Convergencia en la capa física)
- **PMD:** Physical Media Dependent(Medio de Dependencia Físico)
- **PTP:** Picture Transfer Protocol(Protocolo de Transferencia de Imágenes)
- **Qos:** Quality Of Service(Calidad de Servicio)
- **RTS:** Request To Send. (Pedir Permiso para Transmitir)
- **SDU:** Service Data Unit (Unidad de Datos de Servicio)
- **TCP/IP:** Transmission Control Protocol/ Internet Protocol (Protocolo de Control de Transmisión)
- **TIC:** Tecnologías de la Información y de la Comunicación.
- **UDP:** User Datagram Protocol (Protocolo de Intercambio de Paquetes de Datos)
- **Voip:** Voice Over Ip(Voz Sobre Ip)
- **WAN:** Wide Area Network (Red de Área Amplia)
- **WLAN:**Wireless Local Area Network(Red de Área Local Inalámbrica)

INTRODUCCIÓN

Actualmente, las redes de telecomunicación suponen un porcentaje considerable del consumo energético de las TIC (Tecnologías de la Información y de la Comunicación.). Lo que evidentemente, implica un consumo energético a nivel global. [1]

Como las tendencias del uso de las TIC y en el acceso a servicios y contenidos de banda ancha, sigue en continuo aumento, es de suponer que si no se actúa en el momento adecuado y de forma adecuada, no sólo la red correría peligro de saturación sino que además el consumo energético y su equivalente en gases de efecto invernadero podrían dispararse de manera irrevocable. Es necesario entonces, reflexionar sobre las vías para mantener un crecimiento sostenible en la capacidad de transmisión de las redes de comunicación a la vez que se mantiene o reduce el consumo energético.

Hoy en día, existen las llamadas alternativas “Verdes” (Green), las cuales están siendo tomadas en cuenta por las empresas de Telecomunicaciones. Se habla entonces de una “Informática Verde”, cuyo propósito está encaminado, tanto a reducir las emisiones internas como a ayudar a los clientes a reducir las suyas propias. [1]

El plan de actuación a nivel mundial, abarca desde los equipos de acceso a los equipos de red, los códigos de conducta que promueve el regulador y a la propia arquitectura de las redes de comunicación.

Es por ello que el motivo principal del presente proyecto de grado, se basa en estudiar la norma IEEE 802.11 en varias de sus especificaciones (a/b/g). Se pretende evaluar el comportamiento de las redes de área local inalámbricas mediante el simulador de redes NS-3, (Network Simulator) de manera que se pueda simular cómo respondería una configuración arbitraria de una red de este tipo, extrayendo

resultados que permitan actuar posteriormente sobre parámetros de configuración de dispositivos en un escenario real, con ordenadores portátiles, implementando las técnicas necesarias para el ahorro de energía en dispositivos 802.11.

El proyecto se realizó en las instalaciones de la Universidad de Cantabria, en la Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación, ubicada en España, en la Comunidad de Cantabria, Municipio Santander. El Grupo de Ingeniería Telemática, ha trabajado durante varios años con los simuladores de redes, (Ns-2), y mas recientemente con el Ns-3.

El proyecto estuvo coordinado por el Prof. Roberto Sanz Gil, Ingeniero y profesor titular de la Universidad de Cantabria. (Coordinador en el Departamento de Ingeniería de Comunicaciones.) Cabe destacar que dicho proyecto se realizó en el marco del programa de cooperación de intercambio académico, Programa de Becas, "ERASMUS MUNDUS" entre la Universidad de Cantabria y la Universidad Central de Venezuela. El Prof. Carlos Moreno, Ingeniero Electrónico graduado de la Universidad Simón Bolívar, especializado en Telemática, tuvo su participación dentro del proyecto, al colaborar extensivamente en la redacción del tomo, y verificación de parámetros de forma y especificaciones técnicas dentro del área de Telemática, coordinados directamente y vía internet, desde Venezuela.

En el ***capítulo 1***, se explicará un poco el planteamiento del problema, así como también propuesta de solución del mismo a través de los objetivos y una breve descripción del proyecto en si.

En el ***capítulo 2***, se hará reseña al Marco Teórico del trabajo, donde se explican detalladamente los conceptos necesarios de los elementos del proyecto, como lo son las Redes Ad-Hoc, y la pila de protocolos del modelo OSI.

En el ***capítulo 3***, se hará referencia al Marco Metodológico del trabajo, definiendo las variables del proyecto, la población y muestra a estudiar, los instrumentos y procedimientos realizados. Se explicará de manera detallada, las fases

en las que estuvo divido el trabajo, y las limitaciones del mismo. Se hará referencia a los conceptos abstractos utilizados por el simulador para la definición de entidades y su equivalente en el mundo real. También se explicará de manera general, los pasos básicos para realizar una simulación en el Ns-3.

En el ***capítulo 4***, se realizarán los análisis de los resultados obtenidos después de varias simulaciones, y la discusión sobre la información solicitada.

En las conclusiones del proyecto, se mencionarán los principales resultados obtenidos, y los aportes significativos del trabajo realizado. Se expondrán también, algunas líneas futuras de investigación, y las recomendaciones pertinentes que se deben tomar en cuenta a la hora de trabajar con simuladores, basados en lenguajes de programación. Luego en los Anexos, se explicará un poco el código realizado en Ns-3, y el significado de algunas de las líneas de código, para que el lector pueda comprenderlo.

CAPÍTULO I

INTRODUCCIÓN AL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

La situación actual a nivel mundial, con respecto al calentamiento global y cuidado del medio ambiente, han llevado a los expertos, científicos e ingenieros, a buscar soluciones óptimas a problemas diarios que contribuyen a un mejoramiento indudable en el cuidado de nuestro planeta. Como ya se ha mencionado, las TIC, tienen un porcentaje de emisión considerable a nivel mundial de CO_2, y si es posible colaborar, para evitar que esta emisión sea mínima, contribuiremos con las miles de personas que buscan salvar el planeta. El presente trabajo es sólo una pequeña muestra de cómo incluso mediante la conexión de dos ordenadores en modo Ad-Hoc, podemos lograr un ahorro de energía, ya sea mediante pruebas a través de simuladores, o realizando en laboratorios pruebas (escenario real), para la comprobación de los resultados obtenidos a través del escenario teórico planteado, si así se quisiera.

Una de las ventajas de trabajar con simuladores, radica en el hecho de lo fácil y barato que resulta para el desarrollo de las investigaciones científicas. Los sistemas reales son costosos, complejos, y en algunos casos pueden ser hasta inaccesibles para ciertas pruebas. Este es el motivo fundamental del presente trabajo, realizar en el menor tiempo posible, simulaciones, obteniendo mediciones de potencia de recepción, y a través de los resultados obtenidos, realizar un ajuste en la configuración de los ordenadores para que exista un ahorro de energía, con el menor costo posible, y haciendo un aporte significativo en el desarrollo de la informática verde.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL:

- Estudiar el comportamiento de las redes de área local inalámbricas, mediante el uso de las aplicaciones del simulador NS-3, para técnicas de ahorro de energía en dispositivos 802.11.

1.2.2 OBJETIVOS ESPECÍFICOS:

- Instalar y comprender el funcionamiento del simulador NS-3, en el sistema operativo Linux, para aplicar la manera en la que se definen los modelos a simular y extraer resultados.
- Analizar los resultados obtenidos para validar la factibilidad del proyecto en un escenario real.
- Estudiar técnicas cooperativas, utilizando el simulador NS-3 para aplicaciones futuras, en pruebas reales.

1.3 BREVE DESCRIPCIÓN DEL PROYECTO

El presente proyecto está basado en la simulación de la conexión de dos ordenadores en modo Ad-Hoc, en cuya función se busca que en la transmisión de datos entre ellos, ocurra un ahorro de energía. La simulación se realizó a través del simulador NS-3.11, el cual se describirá un poco a continuación: [1],[2]

NS-3 es un simulador de eventos discretos para sistemas de Internet. Está diseñado bajo ciertos protocolos con múltiples niveles de abstracción y a su vez, está escrito en varios lenguajes de programación, como C + +.

Es un proyecto que se inició a mediados del 2006, y sigue creciendo a través de un intenso desarrollo de investigación en la materia. Se le conoce como ***Proyecto NSNAM***, debido a las compañías de redes involucradas en el proyecto.

Cuenta con licencias de código abierto GNU (General Public License), modelos de ***scripts*** realizados en ***Python*** y programas hechos en C + +. Además posee una alineación con los sistemas reales (***sockets***, dispositivos de interfaces de controlador), y los paquetes pueden salvarse archivos PCAP, (Packet Capture) en un formato real, lo cual permite que estos puedan ser ejecutados en programas como ***Wireshark***, obteniendo información valiosa para el estudio.

Los fundadores oficiales de este programa, son investigadores tanto de la Universidad de Washington como de la Universidad Tecnológica de Georgia.

Cabe acotar que NS-3 no es una extensión de Ns-2. Los programas de simulación son ejecutables de C + +. Este nuevo simulador añade el concepto de ***Waf***, un ***framework*** basado en ***Python***, cuya analogía es la de un constructor del programa. Se utiliza para configurar, compilar e instalar aplicaciones. Es un sustituto de otras herramientas, como ***Autotools,*** etc. [1]

Trabaja con varios modelos en la capa de enlace: enlace punto a punto, CSMA (enlaces Ethernal), WIFI (enlaces 802.11), etc. En la parte de enrutamiento, admite modelos: Ad-Hoc, como OLSR (Optimized Link State Routing Protocol), AODV (Ad Hoc On Demand Distance Vector), enrutamiento global (ordenadores con enrutamientos fijos), ***Lix routing*** (unión de múltiples enrutamientos en el mismo nodo), etc.

En la capa de aplicaciones, se encuentran las entidades:

a) ***On/Off:*** genera tráfico, alternándolo en períodos activos o inactivos. Puede ser configurado para generar muchos tipos de tráficos. Trabaja con los protocolos UDP o TCP. (User Datagram Protocol o Transmission Control Protocol)
b) ***Packet Sink***: recibe paquetes o conexiones TCP

c) UDP-cliente/servidor: envía paquetes UDP con una secuencia de números.
d) UDP-echo/servidor: envía paquetes UDP sin secuencia de números.
e) Las rutas admitidas para IPV6 o IPV4.

En general, NS-3, posee características útiles. Es flexible, tiene un diseño organizado y una buena integración con el mundo real. Además de tener un buen rendimiento, es uno de los simuladores más rápidos y eficientes (a nivel de memoria) que existen en el mercado europeo actual. Sin embargo, como es un proyecto que aún está en continuo desarrollo, muchos modelos, aún no están disponibles para este simulador, como los GUI (Graphical User Interface), para construir topologías de manera gráfica. Sin embargo, existe una manera de visualizar gráficamente la información obtenida de la simulación, a través del comando ***gnuplot***, después de haber tabulado los resultados obtenidos.

La documentación de NS-3, se puede encontrar fácilmente a través de la web, la cual fue de gran ayuda para la realización de este proyecto:

a) http://www.nsnam.org/docs/release/3.12/tutorial/singlehtml/index.html
b) http://www.nsnam.org/doxygen/index.html
c) http://www.nsnam.org/wiki/index.php/Main_Page

NS-3, soporta plataformas comunes, como Linux, Mac OS, e incluso Windows (Utiliza una máquina virtual, llamada ***Cywing***, la cual no es muy común, ni fácil de usar). Soporta ***desktop*** y ***server*** de 32 bit o 64 bits, y el lenguaje de programación, está basado en la programación orientada a objeto. [3]

Para mayor información sobre la instalación de este simulador en el sistema operativo UBUNTU, Ver Anexo 4.

CAPÍTULO II

MARCO TEÓRICO

2.1 MODELO OSI

Figura 1. Pila del modelo OSI.

Fuente: http://es.wikipedia.org/wiki/modelo_osi [Consulta Mayo, 2011]

El modelo de interconexión de sistemas abiertos, también llamado OSI (en inglés ***open system interconnection***) es el modelo de red descriptivo creado por la Organización Internacional para la Estandarización en el año 1984. Es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones. El modelo se le puede estructurar como una "pila" de protocolos de comunicaciones, como se puede visualizar en la Figura 1.

El modelo especifica el protocolo que debe ser usado en cada capa, y suele hablarse de modelo de referencia ya que es usado como una gran herramienta para la enseñanza de comunicación de redes. Este modelo está dividido en siete capas: [8]

Capa física

Es la que se encarga de las conexiones físicas de la computadora hacia la red, tanto en lo que se refiere al medio físico como a la forma en la que se transmite la información. [8]

Algunas de sus funciones son:

a) Definir las características funcionales de la interfaz (establecimiento, mantenimiento y liberación del enlace físico).
b) Transmitir el flujo de bits a través del medio.
c) Garantizar la conexión (aunque no la fiabilidad de dicha conexión).

Capa de enlace de datos

Esta capa se ocupa del direccionamiento físico, de la topología de la red, del acceso al medio, de la detección de errores, de la distribución ordenada de tramas y del control del flujo. [8]

Capa de red

Se encarga de identificar el enrutamiento existente entre una o más redes. Las unidades de información se denominan paquetes, y se pueden clasificar en protocolos enrutables y protocolos de enrutamiento. [8]

El objetivo de la capa de red es hacer que los datos lleguen desde el origen al destino, aún cuando ambos no estén conectados directamente. En este nivel se realiza el direccionamiento lógico y la determinación de la ruta de los datos hasta su receptor final.

Capa de transporte

Capa encargada de efectuar el transporte de los datos (que se encuentran dentro del paquete) de la máquina origen a la de destino, independizándolo del tipo de red física que se esté utilizando. Sus protocolos son TCP y UDP; el primero orientado a conexión y el otro sin conexión. Trabajan, por lo tanto, con puertos lógicos y junto con la capa de red dan forma a los conocidos como ***Sockets.*** [8]

Capa de sesión

Esta capa es la que se encarga de mantener y controlar el enlace establecido entre dos computadores que están transmitiendo datos de cualquier índole. Se encarga de asegurar que, dada una sesión establecida entre dos máquinas, la misma se pueda efectuar para las operaciones definidas de principio a fin, reanudándolas en caso de interrupción. [8]

Capa de presentación

El objetivo es encargarse de la representación de la información, de manera que aunque distintos equipos puedan tener diferentes representaciones internas de caracteres, los datos lleguen de manera reconocible.

Esta capa también permite cifrar los datos y comprimirlos. Por lo tanto, podría decirse que esta capa actúa como un traductor. [8]

Capa de aplicación

Ofrece a las aplicaciones la posibilidad de acceder a los servicios de las demás capas y define los protocolos que utilizan las aplicaciones para intercambiar datos, como correo electrónico, gestores de bases de datos y servidor de ficheros (FTP), etc. Hay tantos protocolos como aplicaciones distintas y puesto que continuamente se desarrollan nuevas aplicaciones el número de protocolos crece sin parar. El usuario normalmente no interactúa directamente con el nivel de aplicación. Suele interactuar con programas que a su vez interactúan con el nivel de aplicación pero ocultando la complejidad del mismo. [8]

Transferencia de información en el modelo OSI.

El intercambio de información entre dos capas OSI consiste en que cada capa en el sistema fuente le agrega información de control a los datos, y cada capa en el sistema de destino analiza y quita la información de control de los datos como sigue: [8]

Si un ordenador (1) desea enviar datos a otro (2), los datos deben empaquetarse a través de un proceso denominado encapsulamiento, es decir, a medida que los datos se desplazan a través de las capas del modelo OSI, reciben encabezados, información final y otros tipos de información.

La capa de aplicación recibe el mensaje del usuario y le añade una cabecera constituyendo así la PDU (Protocol Data Unit) de la capa de aplicación. La PDU se transfiere a la capa de aplicación del nodo destino, este elimina la cabecera y entrega el mensaje al usuario. [8]

Luego hay que entregar la PDU a la capa de presentación, para ello hay que añadirle la correspondiente cabecera ICI (Información de Control de la Interfase) y transformarla así en una IDU (Unidad de Datos de la Interfase), la cual se transmite a dicha capa.

La capa de presentación recibe la IDU, le quita la cabecera y extrae la información, es decir, la SDU (Service Data Unit), a esta le añade su propia cabecera (PCI) constituyendo así la PDU de la capa de presentación.

Esta PDU es transferida a su vez a la capa de sesión mediante el mismo proceso, repitiéndose así para todas las capas.

Al llegar al nivel físico se envían los datos que son recibidos por la capa física del receptor.

Cada capa del receptor se ocupa de extraer la cabecera, que anteriormente había añadido su capa homóloga, interpretarla y entregar la PDU a la capa superior.

Finalmente llegará a la capa de aplicación la cual entregará el mensaje al usuario. [8]

2.2 WIFI

WIFI es una tecnología de comunicación inalámbrica inicialmente utilizada en redes de área local y luego convertida en un medio para acceder a Internet de banda ancha.

La norma IEEE 802.11 (ISO/CEI 8802-11) es un estándar internacional que describe las características de una red de área local inalámbrica (WLAN). WIFI es el nombre dado inicialmente a esta certificación. De este modo una red WIFI es en realidad una red conforme a la norma 802.11.

De este modo WIFI permite que se comuniquen PCs portátiles, equipos de escritorio, asistentes personales (PDA) e incluso periféricos con una conexión de banda ancha (11 Mbit/s) dentro de un radio de varias decenas de metros al interior (generalmente entre 20 y 50 metros). Al aire libre el alcance puede ser de varias centenas de metros y en condiciones óptimas varias decenas de kilómetros. [10]

WIFI utiliza una banda de frecuencia estrecha llamada ISM de 2,4 a 2,4835 GHz, de tipo compartido, por lo que se tienen interferencias con hornos microondas, transmisores domésticos, las cámaras inalámbricas, etc.

Cabe notar que la potencia emitida por los equipos WIFI (~30 mW) es treinta veces menor que la emitida por los teléfonos móviles (~1 W). [10]

Estructura

La norma 802.11 define las capas del modelo OSI para un enlace inalámbrico utilizando ondas electromagnéticas, así tenemos:

a) La capa física (PHY), la que ofrece 3 tipos de códigos de información;
b) La capa de enlace de datos, constituida de dos subcapas:
    a. El control de enlace lógico (Logical Link Control, o LLC);
    b. El control de acceso al medio (Media Access Control, o MAC).

La capa física define las especificaciones eléctricas y el tipo de señal para la transmisión de datos, mientras que la capa de enlace de datos define la interfaz entre el bus de la máquina y la capa física, especialmente un método de acceso similar al utilizado en el estándar Ethernet y las reglas de comunicación entre diferentes estaciones. [10]

En una red inalámbrica WIFI es posible utilizar cualquier protocolo del mismo modo que en una red Ethernet.

Modos

El modo infraestructura es un modo de funcionamiento que permite conectar ordenadores equipados de una tarjeta de red WIFI por medio de uno o varios puntos de acceso (AP) que actúan como conectores (Ej: Hub/Switch en red cableada). Este modo es empleado especialmente por las empresas. La ventaja de este modo es que garantiza un paso obligado por la AP, lo que permite verificar quien entra a la red. [10]

El modo Ad-Hoc, (que se explicará en detalle mas adelante), tiene la ventaja que elimina materiales suplementarios costosos y es más fácil implementarlo. Gracias a la adición de un programa de enrutamiento dinámico (Ej: OLSR, AODV, etc.) la red crece automáticamente con la conexión de nuevos equipos.

Las normas WIFI

La norma IEEE 802.11 es en realidad la norma inicial, la que ofrecía velocidades de 1 ó 2 Mbit/s. Se tuvieron que realizar revisiones a la norma original para, entre otras cosas, mejorar la velocidad (es el caso de las normas 802.11a, 802.11b y 802.11g, llamadas normas 802.11 físicas), definir los elementos que garanticen una mayor seguridad o una mayor interoperabilidad. [10]

2.3 RED Ad-Hoc

Como se puede ver en la Figura 2, una red Ad-Hoc es una red inalámbrica descentralizada, sin infraestructura. Cada nodo está preparado para reenviar datos a los demás, y la decisión sobre que nodos reenvían los datos, se toma de forma dinámica, en función de la conectividad de la red. Cada nodo actúa simultáneamente como cliente o servidor de la red. También difiere de las redes inalámbricas convencionales en las que un nodo especial, llamado punto de acceso, gestiona las comunicaciones con el resto de nodos. La comunicación se realiza dentro de un área básica de servicios, que es el área en la cual la comunicación es posible, una característica de esta área es que sus límites son difusos. Si el receptor se encontrara fuera del rango de alcance del emisor, no podrá interpretar correctamente los paquetes del envío. Por eso en las redes Ad-Hoc todos los nodos colaboran para enviarse la información enrutando los paquetes de datos salto a salto, son lo que se denominan redes inalámbricas multi-salto. [4]

Las redes Ad-Hoc, son las más adecuadas en aquellas situaciones en las que no puede confiarse en un nodo central y mejora su escalabilidad comparada con las redes inalámbricas tradicionales.

Son también útiles en situaciones de emergencia, como desastres naturales o conflictos bélicos, al requerir muy poca configuración y permitir un despliegue rápido. El protocolo de encaminamiento dinámico permite que entren en funcionamiento en un tiempo muy reducido. [4]

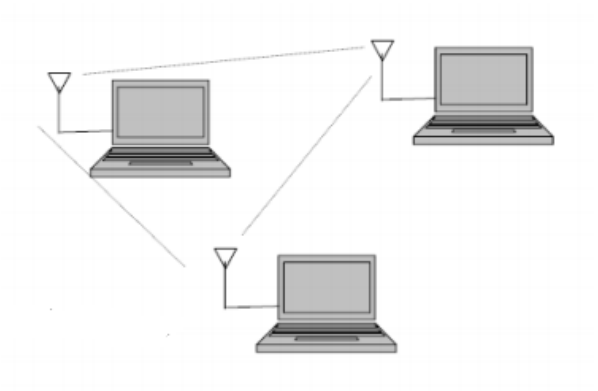


Figura 2. Diagrama de una red Ad-Hoc, con 3 ordenadores.

Fuente: http://eprints.ucm.es/8858/1/microsoft_word-memoria_ssii_final.pdf [Consulta Mayo, 2011]

Hay que acotar que sigue siendo una red inalámbrica, pero no necesita u ***router*** o Access Point. Para poder realizar una conexión en modo Ad-Hoc, se necesita un adaptador o también conocidos como programas de enrutamiento.

Las redes inalámbricas se han extendido rápidamente gracias a su característica principal, que es la movilidad; la posibilidad de no tener la restricción a un puesto fijo permite flexibilidad a la hora de crear la infraestructura de la red. Estas redes se suelen insertar dentro de redes cableadas para tener una estructura híbrida. [4]

Para este tipo de redes la arquitectura se basa en el estándar 802.11 del IEEE, este estándar define una capa LLC (***Logical Link Control***) que logra estandarizar la comunicación con capas superiores, ya que en las capas inferiores (la capa física) existe un número importante de interfaces y protocolos y se necesita un método para poder aislar unas capas de otras. Dentro de la primera capa, la capa física, existe una especificación para cada tipo de medio, y al tener las ondas de radio como medio para la comunicación la capa física se separa en dos subcapas:

a. La capa PLCP: capa que realiza el proceso de convergencia de los ***frames*** MAC sobre el medio.

b. La capa PMD: capa dependiente del medio y se encarga de transmitir el ***frame.***

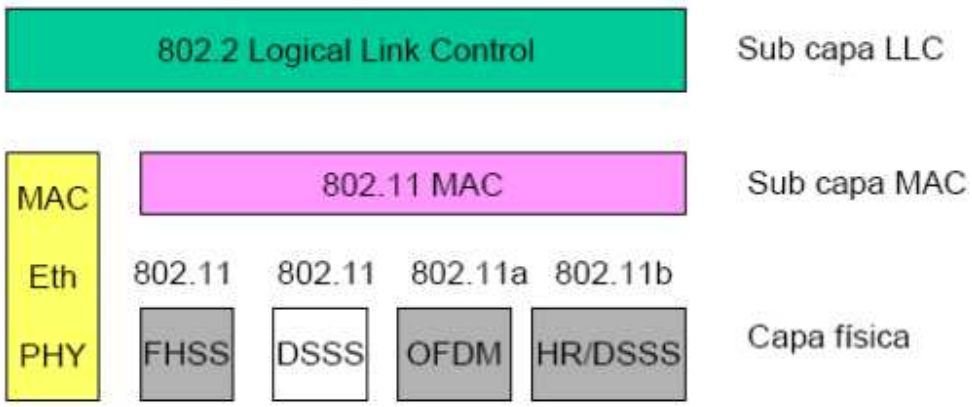


Figura 3. Estructura por capas de una red AD-HOC

Fuente: http://eprints.ucm.es/8858/1/microsoft_word-memoria_ssii_final.pdf [Consulta Mayo, 2011]

A nivel físico, y en el estándar 802.11b, en el cual se trabajará, tenemos que la comunicación se realiza en la Banda de 2,4 Ghz. con velocidad de hasta 11 Mbps.

Debido a las frecuencias en las que trabajamos tenemos el riesgo de interferencias debido a múltiples factores, en particular a las ondas de microondas que funcionan en frecuencias de 2,4 GHz. [5]

Características de una Red Ad-Hoc

1. **Topología dinámica:** los nodos que forman parte de la red son elementos móviles, por eso la topología de la red está en constante cambio. Los enlaces entre los nodos se crean y se destruyan dinámicamente.
2. **Ancho de banda limitado:** el ancho de banda disponible en una interfaz de red inalámbrica es inferior a la de una red cableada, además se ve afectada a interferencia de señales y atenuación propias del medio.
3. **Consumo de energía:** con frecuencia los nodos de la red estarán siendo alimentados por baterías, este hace necesario que el ahorro de energía sea algo importante a tener en cuenta. [5]
4. **Seguridad:** los problemas de trabajar con medio compartido, al que cualquiera puede tener acceso, acentúan los riesgos de seguridad de la red, como protegerse la confidencialidad de los datos, para que no sean utilizados por terceros no autorizados, etc.

Se debe tener en cuenta que el excesivo envío de mensajes de control, disminuye el ancho de banda disponible para el tráfico útil de información, y serán protocolos intensivos en procesamiento para evitar el agotamiento de las fuentes de alimentación de los nodos. Además, se deberán de implementar los mecanismos de seguridad pertinentes. [4] [5]

Figura 4. Arquitectura del Nodo Ad-Hoc

Fuente: http://eprints.ucm.es/8858/1/microsoft_word-memoria_ssii_final.pdf

[Consulta Mayo, 2011]

- **Nivel de Enlace**

Como las redes inalámbricas utilizan un medio compartido que transporta los datos que se emiten, es necesario que los nodos no interfieran en las comunicaciones que puedan estar manteniendo los demás, por eso el control de acceso al medio (MAC) es uno de los aspectos más importantes. Por lo general se usan dos esquemas de coordinación:

a) Centralizado, en el que existe un árbitro que va asignando el turno de palabra a cada emisor, de manera que ningún nodo puede transmitir fuera de su turno.
b) No centralizado. En este esquema cualquier nodo puede transmitir cuando considere oportuno, sin embargo, es susceptible a colisiones, y cada nodo debe ser capaz de recuperase.

Una de las ideas, adecuada al desarrollo de redes Ad-Hoc, es minimizar la potencia de la señal, con la que se transmite, de manera que esta afecte al menor número de nodos posibles, o bien, utilizar antenas direccionales en la dirección del nodo destino.

- **Nivel de Red**

La capa de red es la encargada de formar y enviar los paquetes de datos a sus destinos correctos. El motivo de hacerlo así, no es otro que hacer que la red Ad-Hoc sea independiente de las tecnologías de acceso al medio utilizadas.

Los protocolos ya existentes para trabajar a este nivel, pueden sufrir algunas modificaciones para trabajar con los nodos Ad-Hoc, de esta manera los nodos pueden comunicarse, unos con otros, de una manera transparente para las aplicaciones. (Ver Figura 4.)

En una red de este tipo cada nodo debe comportarse como un ***router***, manteniendo individualmente las rutas a otros nodos. Esto implica que cuando crezca el número de nodos en la red, crecerán el número de rutas que ha de mantener cada nodo, y por consiguiente, las tablas con la información de enrutado hacia otros nodos.

La escalabilidad y la movilidad de estos, se convierten en los grandes problemas del mantenimiento de este tipo de redes, produciendo un aumento de la carga para el mantenimiento de las rutas, consumiendo el ya de por sí escaso ancho de banda.

- **Nivel de Transporte**

El protocolo de transporte UDP ofrece un servicio no confiable y no orientado a conexión de entrega de paquetes. Los mensajes de datos del nivel de aplicación son encapsulados en paquetes UDP y enviados confiando que alcanzarán correctamente su destino, aunque no se garantiza nada.

- **Nivel de Aplicación**

Para un uso comercial, se hace imprescindible que los protocolos usados en este nivel sean los mismos empleados para el mismo propósito en redes fijas; aunque para uso militar o de emergencias es posible que se ideen otros protocolos que atiendan más a las características de una red Ad-Hoc. [4]

2.4 TRANSMISIÓN DE DATOS DIGITALES

Los seres humanos nos desenvolvernos en medios donde tenemos que comunicarnos. Por eso la gran importancia de la transmisión y la recepción de información.[9]

Para nuestro objeto de estudio, haremos referencia sólo a la transmisión de datos digitales, ya que el intercambio de información que se propone, es entre dos ordenadores. Las perturbaciones en la transmisión, que debemos de tener en cuenta al realizar un estudio en esta área, son:

- **Atenuación**

La energía de una señal decae con la distancia, por lo que hay que asegurarse que llegue con la suficiente energía como para ser captada por la circuitería del receptor y además, el ruido debe ser sensiblemente menor que la señal original (para mantener la energía de la señal se utilizan amplificadores o repetidores).

- **Capacidad del canal**

Se llama capacidad del canal a la velocidad a la que se pueden transmitir los datos en un canal de comunicación de datos. La velocidad de los datos es la velocidad expresada en bits por segundo a la que se pueden transmitir los datos. El ancho de banda es aquel ancho de banda de la señal transmitida y que está limitado por el transmisor y por la naturaleza del medio de transmisión (en hertzios). La tasa de errores es la razón a la que ocurren errores. [9]

Para un ancho de banda determinado es aconsejable la mayor velocidad de transmisión posible pero de forma que no se supere la tasa de errores aconsejable. Para señales digitales, hay que codificar más de un bit en cada ciclo, y es posible transmitir más cantidad de información. Por el teorema de Nyquist, al aumentar los niveles de tensión diferenciables en la señal, es posible incrementar la cantidad de información transmitida.

El problema de esta técnica es que el receptor debe de ser capaz de diferenciar más niveles de tensión en la señal recibida, cosa que es dificultada por el ruido. Cuanto mayor es la velocidad de transmisión, mayor es el daño que puede ocasionar el ruido. [9]

Transmisión Sincronía

Este tipo de transmisión se caracteriza porque antes de la transmisión propia de datos, se envían señales para la identificación de lo que va a venir por la línea. Es mucho más eficiente, ya que antes de transmitir cada bit se envía la señal de reloj y en paralelo es síncrona cada vez que se transmite un grupo de bits. [9]

Ventajas de la Transmisión Digital.

La ventaja principal de la transmisión digital es la inmunidad al ruido. Las señales analógicas son más susceptibles que los pulsos digitales a la amplitud no deseada, frecuencia y variaciones de fases.

Se prefieren a los pulsos digitales por su mejor procesamiento y multicanalizaciones que las señales analógicas. Los pulsos digitales pueden guardarse fácilmente, mientras que las señales analógicas no pueden.[9]

Los sistemas digitales utilizan la regeneración de señales, en vez de la amplificación de señales, por lo tanto producen un sistema más resistente al ruido que su contraparte analógica.

Los sistemas digitales están mejores equipados para evaluar un rendimiento de error (por ejemplo, detección y corrección de errores), que los sistemas analógicos.

En el caso de equipos basados en 802.11 a/b/g la tasa máxima de transferencia de datos es aproximadamente la mitad de la velocidad a nivel radio. De este modo un equipo operando a 54 Mbps en radio, ofrecerá alrededor de 22 Megabits de tasa de transferencia de datos y no 54 Mbps. Esto es una limitación del protocolo y aplica para todos los equipos basados en arquitectura WIFI. [9]

CAPÍTULO III

MARCO METODOLÓGICO

- Entorno de trabajo: Sistema Operativo Ubuntu, (deriva de Linux)
- Población: Ordenadores portátiles
- Muestra: Dos (2) ordenadores portátiles.
- Definición de las variables:
    - Potencia de Transmisión Tx.
    - Potencia de Recepción Rx.
    - Distancia de separación entre ambos ordenadores
- Instrumentos para efectuar las pruebas: Simulador NS-3, basado en C++ orientado a objeto.

3.1 Modelo de la Red a Simular

Escenario 1:

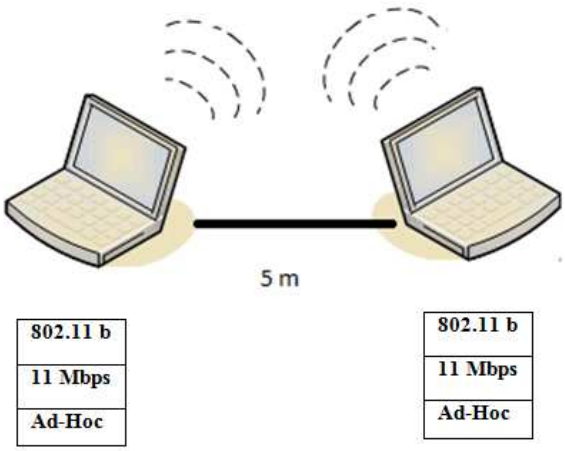


Figura 5. Dos ordenadores fijos conectados en modo AD-HOC

Escenario 2:

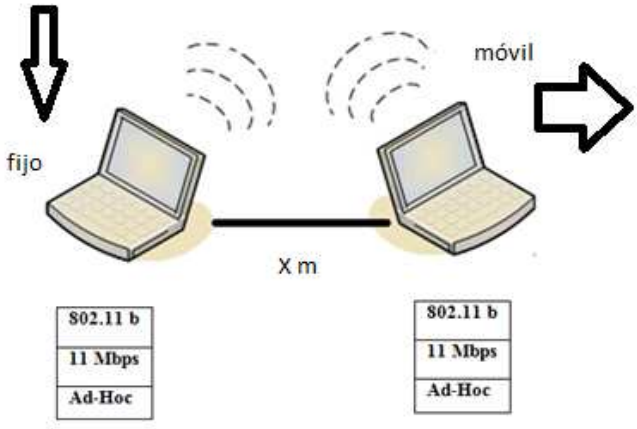


Figura 6. Dos ordenadores, uno fijo, otro móvil, conectados en modo AD-HOC. Distancia variable

En el escenario 1, se puede apreciar el modelo básico de la red, de dos ordenadores conectados en modo Ad-hoc. Es el modelo de prueba, que garantiza que existe una transmisión de datos entre ambos terminales.

En el escenario 2, se plantea el mismo escenario 1, con las variaciones que se van a realizar tanto de potencia de transmisión, como de la distancia que existirá entre ambos ordenadores. Este es el modelo a realizar para cada una de las simulaciones que se realicen mediante el ns-3. Para ello, se realizaron 2 pruebas por separado, cuyos resultados, serán explicados y analizados en el capítulo 4.

3.2 Fases del Proyecto

El presente proyecto estuvo divido en 3 fases:

Fase 1. Estudio Documental

Fase 2. Resolución del Problema. (Pruebas mediante el simulador)

Fase 3. Análisis de Resultados, Propuestas, y Conclusiones.

3.2.1 Fase 1. Estudio Documental.

En esta fase se realizó una investigación exhaustiva sobre el simulador NS-3.11. Se descargó la versión oficial del simulador, en la página web, así como también se realizó el tutorial del mismo para la comprensión de su uso. Se instalaron las librerías necesarias para que el programa funcionara correctamente. Luego se instaló el simulador, y realizando paso a paso los ejemplos de programas a simular (detallados cada uno en el tutorial), se comprendió con mayor énfasis como resolver cada uno de los problemas reales simulados en el NS-3.11.

3.2.2 Fase 2. Resolución del Problema

En esta fase se planteó el problema en forma de esquema, y a través del uso del simulador, se realizó un código capaz de resolverlo. Mediante cierta cantidad de pruebas se obtuvieron resultados que luego fueron tabulados, para encontrar la mínima potencia de recepción entre ambos terminales. Para ello, se fue variando la distancia entre ambos ordenadores, dejando uno fijo, y evidentemente, el otro móvil, con una distancia de separación mínima de 5 m. Este proceso fue aumentando en distancia, y cuidando que hubiese recepción de datos en el terminal receptor. Las primeras pruebas se realizaron con una potencia de Transmisión Tx, por defecto del simulador NS-3, valorada en -80 dBm. Después, ajustando la Potencia de Transmisión Tx, en función de la distancia, obtuvimos la variación de la Potencia de Recepción Rx, asegurando la mínima distancia, y la mínima recepción. Una vez construido el código, mediante el comando. /***waf,*** se generó un archivo PCAP, el cual es indispensable para analizar a través de las tramas de los paquetes, la transmisión de datos, cantidad de data, potencia de recepción, de ruido, etc. Por medio del analizador de protocolo ***Wireshark***, se hizo posible el análisis de los archivos PCAP.

Todos los resultados obtenidos, fueron tabulados y analizados a posterior en la siguiente fase.

3.2.3 Fase 3. Análisis de Resultados, Propuestas, y Conclusiones.

En esta fase final, se analizaron los resultados obtenidos, buscando a que distancia mínima entre ambos terminales, y asegurando la recepción de data, se obtuvo una mínima potencia de transmisión. Se realizaron las conclusiones pertinentes, y se propone como futura líneas de investigación, realizar las pruebas reales pertinentes con ordenadores portátiles, con una configuración arbitraria, (dependiendo de los resultados que se obtengan mediante el uso del simulador) de manera tal de lograr un ahorro de energía en la Transmisión de datos.

3.3 Conceptos previos de las Entidades utilizadas en el Simulador

En esta sección se explicará algunos conceptos que son usados comúnmente en las telecomunicaciones, y que tienen un significado específico en NS-3 .[2] [3]

1. ***Node:***

Es una estructura de datos, que posee diferentes campos. Designado con este nombre, basándose en la teoría de grafos, y debido a que NS-3 es un simulador de red, no un simulador de internet, por tanto, para hacerlo de la manera más general posible, se le da esta designación. Representa el dispositivo de computadoras más simple y sencillo. Representa un componente de una red de telecomunicaciones, un terminal, un usuario final. Se debe pensar que es un ordenador al cual se le añadirán algunas funcionalidades, como aplicaciones, protocolos, plaquetas de periféricos, etc.

Representado por la clase {**Node**}

2. Application

Un usuario, típicamente ejecuta una aplicación que adquiere y utiliza los recursos controlados por el software del sistema para lograr algún objetivo. Así como la aplicación de software se ejecuta en equipos para realizar tareas en el mundo real, las aplicaciones NS-3 se ejecutan en los nodos NS-3 dentro del mundo simulador. Es la abstracción básica para un programa de usuario que genera algún tipo de actividad que se quiere simular. Representado por la clase {**Application**}

Contiene métodos capaces de manejar las representaciones a nivel de usuario, (aplicaciones) en simulaciones. Hay aplicaciones, y para nuestro caso en particular que se utilizarán las aplicaciones ***UdpEchoClientApplication*** y ***UdpEchoServerApplication.*** Está compuesta por un cliente y un servidor, los cuales se establecerán para generar una simulación donde exista intercambio de paquetes.

3. Channel

Entidad que representa un canal de comunicación. Igual que en el mundo real, intenta conectar nodos a la red. Un ***Channel*** puede modelar desde lo más simple como un cable, hasta modelos más complejos como un ***swtich*** de Ethernet, o un espacio tridimensional lleno de obstrucciones como es en el caso de las redes tipo ***wireless***. Representado por la clase {**Channel**}

Ejemplos de funciones de esta clase, están representadas por ***CsmaChanne***l (protocolo para Ethernet), ***PointToPointChannel*** (enlace punto a punto), ***WIFIChannel***.

4. Net Device

Esta entidad se usa cuando se desea conectar ordenadores a una red. Son simples dispositivos de red, los cuales pueden ser: un cable de red, una tarjeta periférica, las cuales se necesitan instalar en el ordenador. En pocas palabras, son dispositivos de red.

Este se instala en un nodo para que se pueda comunicar con otros nodos en la simulación, a través de los canales. Un **Node** debe conectarse a más de un **Channel** vía múltiples **NetDevice.** Representada por la clase {**NetDevice**}.

De igual manera existen funciones dentro de esta clase, como por ejemplo: ***CsmaNetDevice, PointToPointNetDevice, WIFINetDevice***. Como se puede observar, todos están diseñados para trabajar en conjunto con la clase Channel

5. ***Topology Helpers.***

Para tratar de facilitar el trabajo, se creo la clase **TopologyHelpers**, la cual representa tareas creadas con diferentes opciones de conexiones convenientes al programa a simular. Es decir, representa una librería, donde se guardan distintas clases de topología, y en donde cada clase, ya viene creada con sus métodos y funciones específicas. Sólo hay que invocar este método, y todos los atributos, los heredaran los objetos que se creen dentro del programa. (**Programación Orientada a Objeto**)

6. ***Socket de internet***

Se designa como un concepto abstracto por el cual dos programas situados en ordenadores distintos pueden intercambiar cualquier flujo de datos de manera fiable y ordenada.

La palabra ***socket***, nombra una interfaz de programas de aplicación (API) para el ***stack*** de protocolos de internet TCP/IP, provisto usualmente por el sistema operativo. Constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red.

Un socket queda definido por:

a) Par de direcciones IP: identifica computadora origen y computadora de destino
b) Protocolo de Transporte: permite el intercambio de octetos

c) Par de números de puertos, local y remoto: identifica un programa dentro de cada computador.

Los ***sockets*** permiten implementar una arquitectura cliente/servidor. La comunicación la inicia el cliente. El segundo programa, espera que el otro inicie la comunicación. (Programa servidor)

Es un proceso que ocurre en la maquina cliente, y en la maquina servidora que se usa en última instancia para que el programa servidor cliente, lean y escriban la información que será transmitida por las diferentes capas de red.

7. Pcap

Es una interfaz de una aplicación de programación para captura de paquetes. La implementación en Unix es ***libpcap*** y en Windows es ***winpcap***. Pueden ser utilizados por un programa para capturar paquetes que viajan por toda una red, para transmitir los paquetes en la capa de enlace de una red.(captura del paquete). Una vez creado, se pueden utilizar programas como ***Wireshark***, para analizar el tráfico de la red.

- ***Tcpdump:*** herramienta en línea de comandos, cuya función es analizar el tráfico que circula por toda la red. Permite capturar y mostrar en tiempo real los paquetes transmitidos y recibidos en la red a la cual el ordenador esta conectado. Hace uso del **PCAP** para capturar los paquetes que circulan por la red.

8. Seguimiento de la red(Obtención de la información). Tracing System

Se crea un ***trace file*** para cada dispositivo, y sólo guarda los emitidos. Estos los guarda en un archivo **PCAP**, y a través del comando ***tcpdump-i eth0***, obtengo las tramas y la información necesaria.

A través del comando ***std::cout*** se obtiene información de datos, de manera rápida, sin cambiar el código.

El sistema Tracing System en Ns3, se basa en:

a) ***Tracing sources***
b) ***Tracing sink***
c) ***Independientes.***

Este tipo de ***tracing*** se usa para conectar las fuentes al mundo real.

Los ***trace sourses***: son entidades que avisan sobre los eventos que suceden durante la simulación. (Generador de eventos). Proporcionar acceso directo a datos de interés. No son útiles por si solos, es por ello que necesitan de los ***traces sinks***. Estas entidades son las que "consumen" información. La división permite repartir los ***trace sources*** por todo el código y obtener información útil. Los ***traces sink*** están conectados a los ***trace sources***. Y estos luego se conectarán al mundo exterior. Para diferenciarlos de una mejor manera, los ***traces sources***, son variables que mantienen una lista de llamadas (***callback***), los ***traces sink***, son funciones utilizadas como tarjetas de una llamada. El ***trace callback*** representa el valor que retorna de una función.

9. MyApp

Existen tres períodos importantes de acotar con respecto al tiempo en la simulación:

a) ***Configuration time***: esta en vigor cuando se ejecuta la función principal.
b) ***Simulation time:*** período de simulación donde se ejecutan los eventos.
c) ***Tear down time***: período cuando los objetos creados, se quedan en ***standby*** hasta que se les sea invocados.

Un error muy común, es creer que las entidades construidas durante la simulación (dinámicamente) están disponibles durante el tiempo de configuración.

Una aplicación en NS-3, siempre tiene una hora de inicio, y otra hora de parada. Con esto se asegura que existen los nodos o las entidades creadas dentro del programa.

La solución que se encontró, a este problema de tener que colocar momentos de parada e inicio a cada aplicación, y mucho más cuando son muchas aplicaciones, fue crear un simulador de eventos, que se ejecute después de que los objetos dinámicos se creen. Una vez creados, en el tiempo de configuración, se pasan como parámetros dinámicos al sistema, para que este lo use en el tiempo de simulación. Este simulador de eventos es el que se le conoce con el nombre de ***MyApp***, el cual toma el ***Socket*** como parámetro, y que este se cree en el tiempo de configuración. Por ello es importante en esta función, crear el socket mediante la línea de código, ***Ptr <socket> Socket***. Una vez ejecutado, se destruye. Es obligatorio el uso de las aplicaciones ***StarApplication*** y ***StopApplication***, las cuales inicializan o finalizan a ***MyApp.***

10. Ventana de congestión

Primero se crea el ***socket*** y la traza de conexión. Luego se pasa el ***socket*** al constructor de la aplicación, y **MyApp** lo instalará en los nodos.

11. Waf

Una vez que se haya descargado el código fuente en el sistema local, se tendrá que compilar la fuente para producir programas útiles. Además de ser el más conocido, es probablemente el más difícil de utilizar en un sistema muy grande y altamente. El sistema de construcción Waf es el que se utiliza en el proyecto de NS-3. Es una de las nuevas generaciones de Python basado en sistemas de construcción.

Las características más importantes de WAF:

a) Automática para construir: el orden de construcción se calcula a partir de archivos de entrada y salida, entre otros.

b) Dependencias automáticas: las tareas a ejecutar son detectados por los archivos y los comandos ***hash.***

c) Rendimiento: Las tareas se ejecutan en paralelo de forma automática.

d) Flexibilidad: los nuevos comandos y tareas se pueden añadir muy fácilmente a través de la subclasificación

e) Extensibilidad: aunque muchos lenguajes de programación y compiladores son soportados por defecto, muchos otros están disponibles como extensiones IDE de apoyo: Eclipse y Visual Studio generadores de proyectos

f) Documentación: La solicitud se basa en un modelo sólido documentado en el libro Waf y en la documentación de la API

g) Portabilidad de Python: CPython 2,4 a 3,2, IronPython

12. ***PHY: (physical layer***) Del modelo OSI. Una instancia de Phy conecta un dispositivo de la capa de enlace de un medio físico (Mac)

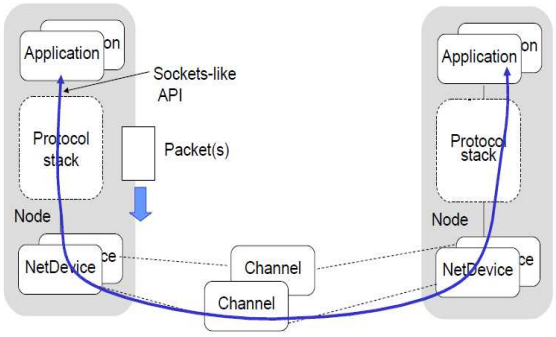


Figura 7. Modelo Básico de una Red en NS-3

Fuente: www.tomh.org/talks/simutools08-keynote-final.ppt. [Consulta Mayo, 2011]

3.4 PASOS BÁSICOS PARA REALIZAR UNA SIMULACIÓN EN NS.3

1. Abrir el editor de texto, en donde se escribirá el código a simular. Ej: ***gedit,***
2. Incluir las librerías (***includes***), las cuales contienen las clases, métodos, acciones necesarias, para ejecutar un programa.
3. Declarar el entorno donde se va a trabajar en NS-3 (***using namespace*** ns3)
4. Utilización de ***Logging***, la cual permite mandar información, mensajes a través de pantalla.
5. Declaración de la función principal, (***main function***, la que se ejecuta primero)
6. Utilización de las topologías, dependiendo de lo que se desee simular. Ej.: creación de nodos, conectar nodos, conectar nodos a red, etc. Las clases ***Node, Channel*** y ***NetDevice***, están ubicados en contenedores, una clase llamada ***Container***, de tal manera que la creación sólo sea la instancia de esa clase creada. Los ***Topology Helper,*** al ser métodos, deben instalarse en los objetos ya creados. Cabe acotar que cada objeto hereda los atributos de las clases de las cuales son instancias.
7. Instalar los protocolos. Protocolos de internet en cada topología creada. En este paso se asocia los dispositivos de los nodos a una dirección **IP**. Por defecto, la dirección **IP** asignada será 10.1.1.1, e ira incrementándose de uno en uno. No se puede generar una misma dirección **IP** ya que esto ocasionaría un error fatal
8. Instalación de las aplicaciones, para poder generar tráfico.
9. Se define el tiempo de la aplicación. En que tiempo empieza, y en que tiempo termina.
10. Otras definiciones, dependiendo de lo que se desee simular.
11. Se define la simulación. se programan los eventos de tal manera que se ejecuten secuencialmente.
12. Se finaliza la simulación.

13. Guardar el código como un documento “nombre_del_programa”.cc, y se guarda dentro de la carpeta *scratch*, la cual se puede encontrar en el directorio de NS-3.11
14. Compilar o construir el programa con. /***waf.***
15. Si no se consigue error en las líneas de código del programa, este se habrá construido con éxito. Si no, se debe corregir lo que se indica en los mensajes de error a través de pantalla.
16. Se ejecuta el programa mediante el comando:

    ./waf - -run scratch/”nombre_del_programa”,

    lo cual indicará que la simulación se realizó con éxito, y la tasa de recepción de bits.
17. Para colocar la data en un archivo de texto:

    ./waf –run nombre_del _programa > destino.dat 2 >&1
18. Para obtener una gráfica con valores en NS-3-11 se utiliza el comando ***gnuplot.***
19. Para poder leer el archivo ***.pcap*** generado, utilizó un analizador de protocolo. Ej.: ***wireshark.”nombre_del_archivo_pcap_.pcap***

3.5 LIMITACIONES

Una de las desventajas de trabajar con NS-3, es que al no tener una buena representación grafica de lo que se está simulando, puede simplificar la vista de interacciones complejas entre entidades. Además, es necesario haber programado en C++ orientado a objeto, o al menos tener un conocimiento previo del tema, en lo que se refiere al área de programación en esta modalidad.

CAPÍTULO IV

ANÁLISIS Y DISCUSIÓN DE RESULTADOS.

4.1 Resultado de la compilación de los códigos en Ns-3

Luego de haber realizado los 2 códigos en Ns-3, para la resolución del problema, ambos códigos, fueron compilados y ejecutados por la consola de Ubuntu. Para que el código fuese reconocido por el simulador, se guardó como un archivo *.cc.* Mediante la siguiente línea de código:

./waf - -run/scratch/proyecto

se compila y se ejecuta el archivo. A través de ***./waf –run nombre_del _programa > destino.dat 2 >&1*** se envió la data a un archivo de texto para poder visualizarlo mejor. El archivo, contenía la siguiente información:

```
Waf: Entering directory `/home/quenny/Downloads/ns-allinone-3.11/NS-3.11/build'
Waf: Leaving directory `/home/quenny/Downloads/ns-allinone-3.11/NS-3.11/build'
'build' finished successfully (0.725s)                    (1)

NODE POSITIONS
Node 0 is at (0, 0, 0)
Node 1 is at (5, 0, 0)                                    (2)

IP ADDRESSES ASSIGNMENT
STA0: 10.1.1.1
STA1: 10.1.1.2                                            (3)

Create On-Off UDP App

Creating capture file: proyecto-0-0.pcap
Creating capture file: proyecto-1-0.pcap                  (4)

STATS (Thput includes IP & UDP headers)
Flow 1 (10.1.1.2 -> 10.1.1.1)
  Tx Bytes (includes IP & UDP headers):   1142472
  Rx Bytes (includes IP & UDP headers):   260896
  Throughput: 0.234513 Mbps                               (5)

Modules built:
aodv              applications          bridge    template
click             config-store          core      topology-read
csma              csma-layout           dsdv      visualizer
emu               energy                flow-monitor      tools
internet          lte                   mesh              wifi          (6)
```

En **(1)** Se puede observar un mensaje que indica que el código ha sido compilado con éxito, y por lo tanto procede a su ejecución.

En **(2)** se puede observar la creación de los 2 nodos, y la posición tipo coordenadas, en unidades de metro, a la que se encuentran ubicados en el espacio cada uno de ellos. La diferencia en las coordenadas X, es la separación a tomar en cuenta entre cada ordenador.

En **(3)** se visualiza la de la dirección **IP** asignada a cada nodo.

En **(4)** se puede observar que han sido creados los archivos .**pcap** correspondientes a casa nodo, para su posterior estudio a través del analizador de protocolos.

En **(5)** se puede observar la cantidad de datos enviados por Tx y la cantidad de datos recibidos por Rx. en unidades de bytes. Así como también se puede visualizar el ***troughtput,*** que es el rendimiento final de la conexión realizada.

En **(6)** se pueden observar los módulos construidos al momento de la compilación del programa, y que ya fueron ejecutados.

4.2 Análisis de Tablas

Una vez generado el archivo .***pcap***, mediante el analizador de Protocolos ***Wireshark***, se pudo obtener valores de potencia recibida, potencia de ruido, y tasa de recepción. Los valores obtenidos se presentan mediante tablas, a continuación:

- **Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión Fija (-80 dBm), Distancia variable.**

Distancia (m)	Potencia (dBm)	Potencia de Ruido (dBm)	Tasa de Transmisión (Bytes)
5	-53	-101	1142472
10	-61	-101	1142472
15	-66	-101	1142472
20	-69	-101	1142472
30	-74	-101	1142472
40	-78	-101	1142472
45	-80	-101	1142472
50	-81	-101	1142472
55	-82	-101	1142472
60	-83	-101	1142472
65	-84	-101	1142472
70	-85	-101	1141420
75	-85	-101	1133004
80	-87	-101	737452
85	-88	-101	148332
90	-88	-101	2104

- Modo de conexión: Ad-Hoc.
- Distancia inicial de separación: 5 m
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

En esta tabla se puede observar que a medida que aumenta la distancia, como es de esperarse, la potencia de recepción disminuye, por tanto será necesaria una mayor potencia de transmisión. Las condiciones de propagación, las determina el programador, y en este caso, se consideró en el espacio libre. La librería que crea estas condiciones, estaba ya creada dentro de Ns-3.

Después de obtener estos resultados, el paso siguiente fue modificar la potencia de Transmisión, para que sea la mínima, conservando la cobertura en el otro terminal, para el mismo rango de distancias. Los resultados obtenidos se presentan a continuación:

- **Potencia de Transmisión: 90 dBm**

Distancia (m)	Potencia RX (dBm)	Ruido (dBm)	Tasa (bytes)
5	20	-101	1142472
10	12	-101	1142472
15	7	-101	1142472
20	4	-101	1142472
30	-1	-101	1142472
40	-5	-101	1142472
45	-7	-101	1142472
50	-8	-101	1142472
55	-9	-101	1142472
60	-10	-101	1142472
65	-11	-101	1142472
70	-12	-101	1142472
75	-13	-101	1142472
80	-14	-101	1142472
85	-15	-101	1142472
90	-16	-101	1142472

Tabla 2. Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión 90 dBm, Distancia variable.

- Modo de conexión: Ad-Hoc.
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

En esta tabla de igual manera se puede observar que a medida que aumenta la distancia, disminuye la potencia de recepción. El valor 90 dBm, se utilizó como valor didáctico, para ver el funcionamiento del simulador, y que tipo de resultados se obtienen a través de las simulaciones. Evidentemente no es una transmisión óptima ni típica en este tipo de redes inalámbricas, ya que el consumo de energía es muy grande, y hasta cierto punto, es ilógico manejar este tipo de resultados en un escenario real, pero funcionó como un ejemplo para el entendimiento de los resultados obtenidos a través del simulador.

- **Potencia de Transmisión: 75 dBm**

Distancia (m)	Potencia RX (dBm)	Ruido (dBm)	Tasa (bytes)
5	5	-101	1142472
10	-3	-101	1142472
15	-33	-101	1142472
20	-11	-101	1142472
30	-16	-101	1142472
40	-20	-101	1142472
45	-22	-101	1142472
50	-23	-101	1142472
55	-24	-101	1142472
60	-25	-101	1142472
65	-26	-101	1142472
70	-27	-101	1142472
75	-28	-101	1142472
80	-29	-101	1142472
85	-30	-101	1142472
90	-30	-101	1142472

Tabla 3. Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión 75 dBm, Distancia variable.

- Modo de conexión: Ad-Hoc.
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

Como se puede ver en esta tabla, ocurre igual que las tabla 2. El consumo de energía es muy grande, y sólo se utilizó como método de aprendizaje.

- **Potencia de Transmisión: 50 dBm**

Distancia (m)	Potencia RX (dBm)	Ruido (dBm)	Tasa (bytes)
5	-20	-101	1142472
10	-28	-101	1142472
15	-33	-101	1142472
20	-36	-101	1142472
30	-41	-101	1142472
40	-45	-101	1142472
45	-47	-101	1142472
50	-48	-101	1142472
55	-49	-101	1142472
60	-50	-101	1142472
65	-50	-101	1142472
70	-52	-101	1142472
75	-53	-101	1142472
80	-54	-101	1142472
85	-55	-101	1142472
90	-55	-101	1142472

Tabla 4. Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión 50 dBm, Distancia variable.

- Modo de conexión: Ad-Hoc.
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

De igual manera, se puede observar que no es una potencia de transmisión óptima ni lógica, sólo se utilizó de manera didáctica.

- **Potencia de Transmisión: 25 dBm**

Distancia (m)	Potencia RX (dBm)	Ruido (dBm)	Tasa (bytes)
5	-45	-101	1142472
10	-53	-101	1142472
15	-58	-101	1142472
20	-61	-101	1142472
30	-66	-101	823716
40	-70	-101	181996
45	-72	-101	53652
50	Indeterminada	0	0
55	Indeterminada	0	0
60	Indeterminada	0	0
65	Indeterminada	0	0
70	Indeterminada	0	0
75	Indeterminada	0	0
80	Indeterminada	0	0
85	Indeterminada	0	0
90	Indeterminada	0	0

Tabla 5. Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión 25 dBm, Distancia variable.

- Modo de conexión: Ad-Hoc.
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

En esta tabla, se puede observar, que a 45 m, garantizo todavía recepción en el Nodo 2, pero con poca cantidad de datos recibidos. A 20 m de separación, además de garantizar la recepción total de los datos, existe una disminución de la potencia inicial de transmisión. A partir de este valor, se puede observar que obtengo valores que se adaptan mejor a la realidad. En los casos, donde la potencia de recepción, es indeterminada, se debe a que como el nodo receptor, ya se encuentra a una distancia donde no recibe datos, (dentro del espacio del simulador), el simulador no genera un archivo ***.pcap,*** por tanto, el analizador de protocolo no tiene nada que descifrar, y no tendremos un valor de potencia de recepción que reflejar.

- **Potencia de Transmisión: 15 dBm**

Distancia (m)	**Potencia RX (dBm)**	**Ruido (dBm)**	**Tasa (bytes)**
5	-55	-101	1142472
10	-63	-101	1142472
15	-68	-101	764804
20	-71	-101	122032
30	Indeterminada	0	0
40	Indeterminada	0	0
45	Indeterminada	0	0
50	Indeterminada	0	0
55	Indeterminada	0	0
60	Indeterminada	0	0
65	Indeterminada	0	0
70	Indeterminada	0	0
75	Indeterminada	0	0
90	Indeterminada	0	0

Tabla 6. Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión 15 dBm, Distancia variable.

- Modo de conexión: Ad-Hoc.
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

- **Potencia de Transmisión: 10 dBm**

Distancia (m)	Potencia RX (dBm)	Ruido (dBm)	Tasa (bytes)
5	-60	-101	1142472
10	-68	-101	503908
13	-71	-101	118876
20	Indeterminada	0	0
30	Indeterminada	0	0
40	Indeterminada	0	0
45	Indeterminada	0	0
50	Indeterminada	0	0
55	Indeterminada	0	0
60	Indeterminada	0	0
65	Indeterminada	0	0
70	Indeterminada	0	0
75	Indeterminada	0	0
80	Indeterminada	0	0
85	Indeterminada	0	0
90	Indeterminada	0	0

Tabla 7. Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión 10 dBm, Distancia variable.

- Modo de conexión: Ad-Hoc.
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

- **Potencia de Transmisión: 5 dBm**

Distancia (m)	Potencia RX (dBm)	Ruido (dBm)	Tasa (bytes)
5	-65	-101	457845
9	-67	-101	32440
15	Indeterminada	0	0
20	Indeterminada	0	0
30	Indeterminada	0	0
40	Indeterminada	0	0
45	Indeterminada	0	0
50	Indeterminada	0	0
55	Indeterminada	0	0
60	Indeterminada	0	0
65	Indeterminada	0	0
70	Indeterminada	0	0
75	Indeterminada	0	0
80	Indeterminada	0	0
85	Indeterminada	0	0
90	Indeterminada	0	0

Tabla 8. Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión 5 dBm, Distancia variable.

- Modo de conexión: Ad-Hoc.
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

- **Potencia de Transmisión: 2 dBm**

Distancia (m)	Potencia RX (dBm)	Ruido (dBm)	Tasa (bytes)
5	-68	-101	260896
7	-72	-101	17698
15	Indeterminada	0	0
20	Indeterminada	0	0
30	Indeterminada	0	0
40	Indeterminada	0	0
45	Indeterminada	0	0
50	Indeterminada	0	0
55	Indeterminada	0	0
60	Indeterminada	0	0
65	Indeterminada	0	0
70	Indeterminada	0	0
75	Indeterminada	0	0
80	Indeterminada	0	0
85	Indeterminada	0	0
90	Indeterminada	0	0

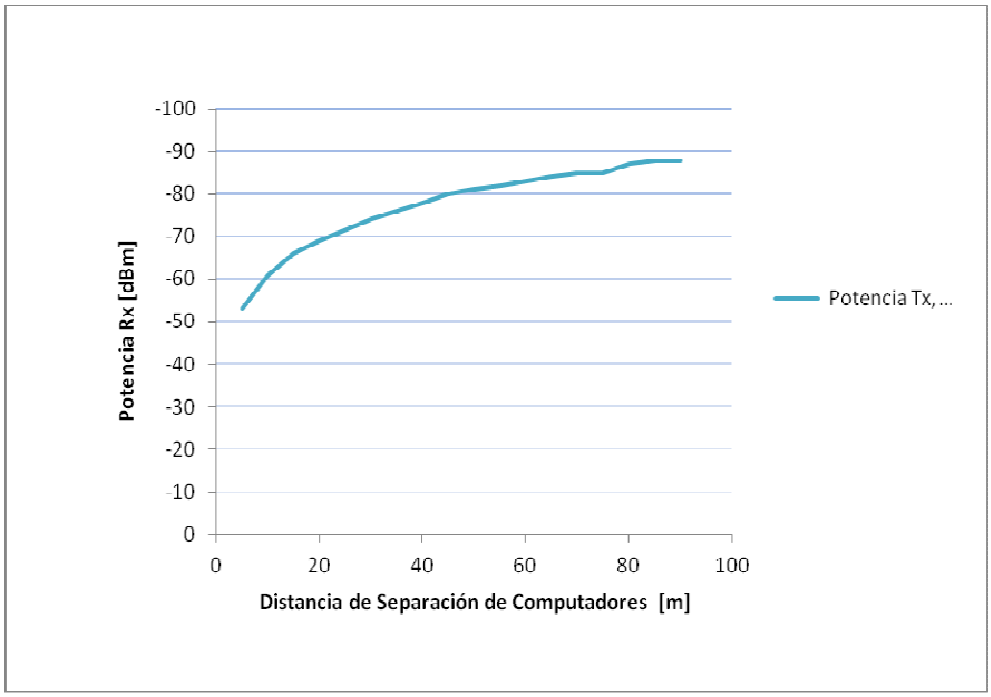
Tabla 9. Resultados de la Potencia de Recepción en el Nodo 2, a Potencia de Transmisión 2 dBm, Distancia variable.

- Modo de conexión: Ad-Hoc.
- Movilidad del Nodo 1: 1 m/seg.
- Data Rate: 11 Mb/seg

Como se puede observar en las tablas 6, 7, 8, y 9, al ir disminuyendo la potencia de transmisión, y a su vez la distancia de separación entre los nodos, la potencia de recepción se mantiene estable a una distancia en particular. Para estos resultados en concreto, la potencia de recepción promedio es de -60 dBm, y la distancia de separación, entre ambos terminales, varía entre 5 y 10 m. Todo va a

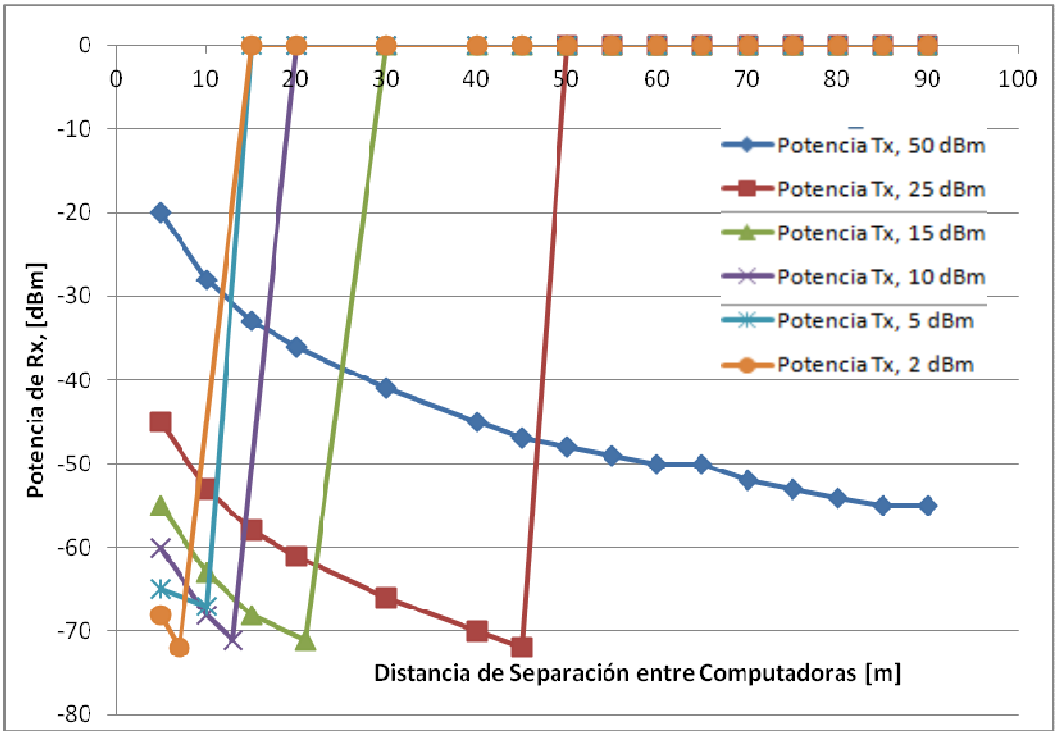
depender de la potencia de transmisión, y de cuantas simulaciones se realicen variando dicho parámetro. Se puede notar también además, que para todas las simulaciones, la potencia de ruido es de -101 dBm, lo cual no causó mayor problema en las simulaciones, ya que son transmisiones digitales, y en teoría éstas, son inmunes al ruido.

4.3 Comparación de la Potencia de Recepción



Gráfica 1. Variación de la Potencia de Recepción Rx, en función de la Distancia de separación de los ordenadores. A Potencia de Transmisión fija igual a -80dBm

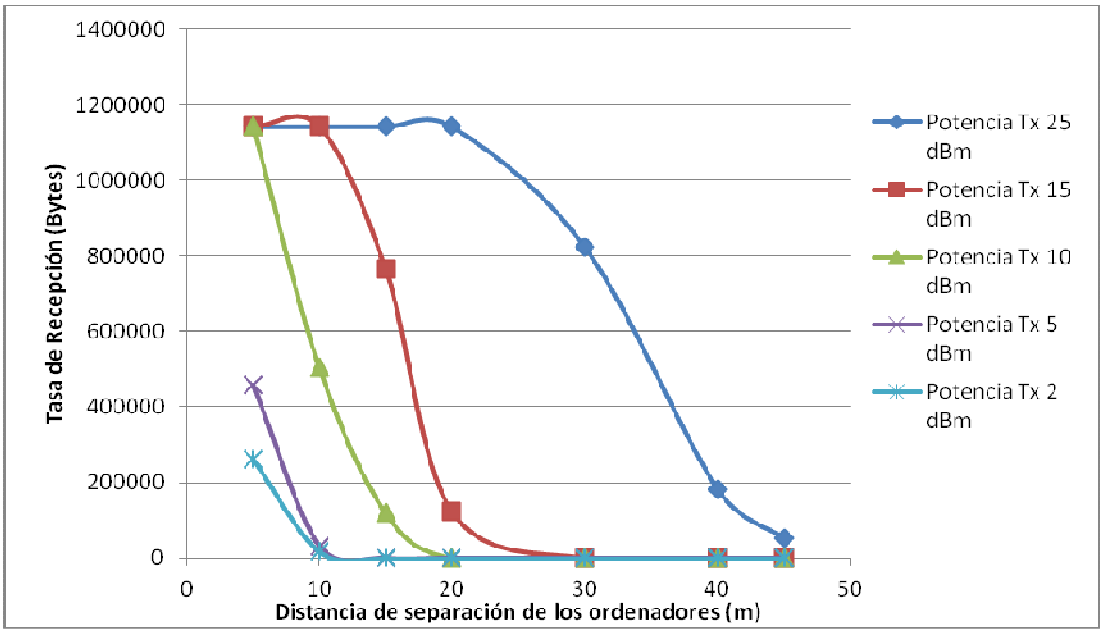
En la gráfica 1, se puede apreciar que a medida que aumenta la distancia entre ambos ordenadores, disminuye la potencia de recepción. Lo cual es de esperarse, ya que al trabajar con tecnología WiFi, existirá un alcance determinado de recepción en el terminal receptor. Lo que nos lleva a concluir, que la simulación se ajusta bastante a lo que ocurre en la realidad cuando se trabaja con este tipo de redes.



Gráfica 2. Variación de la Potencia de Recepción Rx, en función de la Distancia de separación de los ordenadores para diversos valores de Potencia de Transmisión Tx.

En la gráfica 2, se puede apreciar la variación de la potencia de recepción, a medida que se ajustaba la potencia de trasmisión, y a su vez la distancia entre ambas computadoras. Además se aprecia que existirá un punto en el que no existirá, recepción. Esta será la distancia máxima que deberá existir entre los terminales, para cada caso en particular. A su vez, ésta será la potencia mínima de transmisión necesaria para lograr un ahorro de energía, asegurando potencia de recepción en el nodo receptor.

De igual manera se puede concluir que para un valor promedio de separación entre ordenadores (entre 25m y 30m), la potencia mínima óptima, en la cual obtengo un ahorro de energía, es de 25 dBm. con la cual garantizo recepción total de los datos.



Gráfica 3. Variación de la Tasa de Recepción, en función de la Distancia de separación de los ordenadores para diversos valores de Potencia de Transmisión Tx.

En la Gráfica 3, se puede observar que a medida que se iba aumentando la distancia se separación entre ordenadores, la tasa de recepción, fue disminuyendo, lo cual era lo esperado ya que si la potencia de recepción disminuye, menor cantidad de datos, recibirá el nodo receptor.

CONCLUSIONES

Mediante los resultados obtenidos se puede concluir que existe una potencia mínima necesaria para alcanzar al receptor, y que ésta va a depender de la distancia a la que se encuentre. A mayor distancia, se necesitará una mayor potencia de transmisión. El ahorro de energía ocurrirá una vez que se obtenga la potencia mínima, asegurando la recepción en el nodo receptor.

Como futura línea de investigación, se propone realizar pruebas reales pertinentes con ordenadores portátiles, con una configuración arbitraria, tomando en cuenta parámetros como la potencia de transmisión, y la distancia entre ellos, las cuales al variarlas, se pueda tener una disminución en el consumo energético.

De igual manera se propone también, crear un mecanismo capaz de conocer la distancia a la que se encuentre el receptor del transmisor, y que automáticamente se ajuste la potencia de transmisión a la mínima necesaria para la recepción.

Se puede concluir también, que el uso del simulador Ns-3, es bastante fiable, ya que los resultados obtenidos se ajustan a la realidad, tomando en cuenta las tablas sobre las potencias de transmisión típicas y óptimas, dependiendo de la banda de frecuencia en la que se esté trabajando. (Ver Anexo 2). Las simulaciones que se realicen mediante Ns-3, son rápidas, y proporcionan la información necesaria que el usuario desee. El nivel de abstracción de información, es completamente manejado por el usuario. Además, tiene la ventaja que se pueden generar archivos, que sean compatibles con otros programas, bien sean editores de texto, o analizadores de protocolos.

RECOMENDACIONES

Se recomienda que al trabajar con este tipo de simuladores, sea necesario tener un conocimiento previo y bastante amplio sobre programación orientada a objeto, así como también el conocimiento del idioma inglés, ya que tanto la documentación, tutorial, comandos, e incluso los grupos de ayuda creados para la colaboración de información, son manejados en ingles. El hecho de incluirse dentro de estos grupos desarrolladores, fue de vital ayuda para la resolución de dudas, o problemas de sintaxis, dentro del código realizado en Ns-3.

Se recomienda también, haber trabajado en el sistema operativo Ubuntu, y el manejo de comandos vía consola.

Además, se recomienda realizar con cautela el tutorial recomendado en la página de principal de Ns-3, ya que fue de gran ayuda para la realización de la Fase 1 del presente trabajo. (Ver en Referencias Bibliográficas [3])

Por último, se recomienda trabajar con el analizador de protocolos ***Wireshark***, ya que es de fácil manejo, y la interfaz gráfica es bastante amigable para el usuario.

REFERENCIAS BIBLIOGRÁFICAS

[1] University of Technology, Helsinki Institute of Physics, Haruki, I., Carsten, Bormann. “Energy-efficient adaptive interface activation for delay/disruption tolerant networks” http://dl.acm.org/citation.cfm [Consulta Noviembre, 2010]

[2] University of Washington, Henderson, T. “Simutools Conference, ns-3 tutorial”, marzo 2008. www.tomh.org/talks/simutools08-keynote-final.ppt. [Consulta Noviembre, 2010]

[3] University of Washington, Henderson, T. “Ns-3 tutorial” junio 2006, http://www.nsnam.org/docs/release/3.12/tutorial/singlehtml/index.html_tutorial__ns3. [Consulta Noviembre, 2010]

[4] Cano Fernández, R., Fernández Ortega, D. De la Osa Martínez, Mª E. “Estudio de las necesidades en las Redes Ad-Hoc y creación de un protocolo de enrutamiento”. 2005-2006 http://eprints.ucm.es/8858/1/microsoft_word-memoria_ssii_final.pdf. [Consulta Febrero, 2011]

[5] Morán, L., González, N., “La informática verde. Las redes de telecomunicaciones en verde.” Septiembre, 2010. http://www.telefonica.com. [Consulta Febrero, 2011]

[6] “Communication Networks Laboratory. The University ok Kansas EECS 780. Egemen, K., Kaya, C., Sterbenz, J. “Introduction to Network Simulation with ns-3” septiembre, 2010. http://www.nsnam.org/doxygen/index.html [Consulta Noviembre, 2010]

[7] Sociedad Horizontal. “Optimización de parámetros WiFi.” Febrero, 25 de 2010. http://my.opera.com/danitool/blog/2010/02/25/optimiza-wifi [Consulta Febrero, 2011]

[8] Universidad de Buenos Aires, Facultad de Ingeniería. “Modelo de referencia OSI” Feldgen, M. 2011. http://materias.fi.uba.ar/7574/s1apuntes/s1_modelo_osi.pdf. [Consulta Febrero, 2011]

[9] Couchill, L. 1998. “Sistemas de comunicaciones digitales y analógicos” Quinta Edición, Prentice Hall. México.

[10] Wi-Fi Alliance, 2011. http://www.wi-fi.org/discover_and_learn.php [Consulta Febrero, 2011]

GLOSARIO DE TÉRMINOS

- Autotools: es un conjunto de herramientas producido por el proyecto GNU. Estas herramientas están diseñadas para ayudar a crear paquetes de código fuente portable a varios sistemas Unix. El GNU build system forma parte de GNU toolchain y se usa mucho para desarrollar software libre.
- Código Fuente: es un conjunto de líneas de texto de un programa informático (o software), que representan las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento. El código fuente de un programa está escrito por un programador en algún lenguaje de programación
- Código Objeto: se llama así al código que resulta de la compilación del código fuente.
- Comando: es una instrucción u orden que el usuario proporciona a un sistema informático, desde la línea de comandos o desde una llamada de programación. Puede ser interno (contenido en el propio intérprete) o externo (contenido en un archivo ejecutable).
- Compilar: proceso de traducción de un código fuente (escrito en un lenguaje de programación de alto nivel) a lenguaje máquina (código objeto) para que pueda ser ejecutado por la computadora. Las computadoras sólo entienden el lenguaje máquina. La aplicación o la herramienta encargada de la traducción se llama compilador o constructor.
- Componentes de registro: Los componentes de un registro se denominan campos. Cada campo tiene un nombre llamado identificador de Campo, que es algún identificador elegido por el programador cuando se declara el tipo de registro y algún tipo que se especifica cuando se declara el tipo de dato record. Los componentes pueden contener archivos, grupos de archivos, directorios, claves del registro de Windows, accesos directos y otro tipo de datos. El usuario final no interactúa directamente con los componentes.

- Consola: es un programa informático que simula el funcionamiento de un terminal de computadora en cualquier dispositivo de visualización. Incorporan características tales como control de procesos, redirección de entrada/salida, listado y lectura de ficheros, protección, comunicaciones y un lenguaje de órdenes para escribir programas por lotes o (scripts o guiones). Al ingresar las órdenes en el emulador, un intérprete de comandos analiza la secuencia de caracteres ingresada y, si la sintaxis de la orden es correcta, la ejecuta, recurriendo para ello a las funciones que ofrece el sistema operativo o el programa que representa, bien sea un gestor de banco de datos, una sesión de FTP, etc.
- Tasa de Transmisión de Datos (Data rate): es un promedio del número de bits, caracteres o bloques, que se transfieren entre dos dispositivos, por una unidad de tiempo
- Desktop: el entorno de Escritorio o simplemente Escritorio, existe con el objetivo de crear un ambiente de trabajo o esparcimiento cómodo, amigable, funcional, liviano o pesado, sencillo o complejo en el cual los programas de uso cotidiano puedan ejecutarse, y administrarse
- Enrutamiento: es la función de buscar un camino entre todos los posibles en una red de paquetes cuyas topologías poseen una gran conectividad. Dado que se trata de encontrar la mejor ruta posible, lo primero será definir qué se entiende por mejor ruta y en consecuencia cuál es la métrica que se debe utilizar para medirla.
- Fragmentación: es la memoria que queda desperdiciada al usar los métodos de gestión de memoria. ocurre cuando el sistema operativo no asigna suficiente espacio contiguo para almacenar un archivo completo como una unidad, sino que, en cambio, pone partes de él en huecos entre otros archivos (usualmente estos huecos existen porque antes contuvieron un archivo que posteriormente fue borrado por el sistema operativo, o porque éste en primer lugar asignó demasiado espacio para un archivo). Los archivos más grandes y el mayor número de archivos también contribuyen a la fragmentación y en

consecuencia a la pérdida de rendimiento. La defragmentación intenta aliviar estos problemas.

- Trama (***Frame***): es variable en tamaño y puede ser tan largo como miles de bytes o más.
- Frameworks: infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio
- Funciones: en programación, una función es un grupo de instrucciones con un objetivo en particular y que se ejecuta al ser llamada desde otra función o procedimiento. Una función puede llamarse múltiples veces e incluso llamarse a sí misma (función recurrente). Las funciones pueden recibir datos desde afuera al ser llamadas a través de los parámetros y deben entregar un resultado. Se diferencian de los procedimientos porque estos no devuelven un resultado.
- Grafos: un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.
- Hub: Un concentrador o hub es un dispositivo que permite centralizar el cableado de una red y poder ampliarla. Esto significa que dicho dispositivo recibe una señal y repite esta señal emitiéndola por sus diferentes puertos. Los concentradores no logran dirigir el tráfico que llega a través de ellos, y cualquier paquete de entrada es transmitido a otro puerto (que no sea el puerto de entrada).
- Implementación: instalación y puesta en marcha, en una computadora, de un sistema de explotación o de un conjunto de programas de utilidad, destinados a usuarios.

- Interfaz: es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar.
- Lenguaje de Programación: es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.
- Enrutamiento Lix(***Lix routing***): unión de múltiples enrutamientos en el mismo nodo.
- Métodos: un método es una subrutina asociada exclusivamente a una clase (llamados métodos de clase o métodos estáticos) o a un objeto (llamados métodos de instancia). Análogamente a los procedimientos en los lenguajes imperativos, un método consiste generalmente de una serie de sentencias para llevar a cabo una acción, un juego de parámetros de entrada que regularán dicha acción y o, posiblemente, un valor de salida (o valor de retorno) de algún tipo.
- Encabezados(***Overheads***): es la información digital transferida a través de la interfaz funcional entre un usuario y un sistema de telecomunicaciones, o entre las unidades funcionales dentro de un sistema de telecomunicaciones, con el propósito de dirigir o de controlar la transferencia de información de usuario o la detección y corrección de errores.
- Paquete: (***Packet***) Todo tipo de información que es transferida por internet está dividida en paquetes pequeños de información. Cada paquete posee una estructura y tamaño diferente dependiendo del protocolo que lo utilice.

- Parámetro: también conocido como argumento, es una variable que puede ser recibida por una subrutina (también llamada procedimiento, función o rutina), como idea general, se presenta como un algoritmo que forma parte del algoritmo principal, el cual permite resolver una tarea específica
- Programación Orientado a Objeto: La programación orientada a objetos o POO (OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento
- Router: traducido en español como encaminador, enrutador, direccionador o ruteador, es un dispositivo de hardware usado para la interconexión de redes informáticas que permite asegurar el direccionamiento de paquetes de datos entre ellas o determinar la mejor ruta que deben tomar. Opera en la capa tres del modelo OSI.
- Scripts: programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Se crean y controlan con etiquetas del lenguaje HTML, aunque se pueden manipular gráficamente con herramientas diseñadores de páginas.
- Servidor (***Server***): computadora que, formando parte de una red, realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final.
- Sockets: constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

- Caudal (***Throughput***): cantidad de datos real, que son transmitidos hacia o desde algún punto de la red. Es un rendimiento final de una conexión. Volumen de datos que una conexión brinda como resultante de la suma de su capacidad, y la resta de los ***overheads*** que reducen su rendimiento.
- Tráfico: es la cantidad de datos enviados y recibidos por los visitantes de un sitio web. Esta es una gran proporción del tráfico de internet. El tráfico web es determinado por el número de visitantes y de páginas que visitan

ANEXO

[ANEXO 1]

- **Códigos para la simulación en NS-3.**

1. **Código: Potencia Tx fija, por defecto (-80 dBm):**

```
/* Trabajo de Fin de Carrera */
```

// Librerías incluidas necesarias para ejecutar los comandos del programa a simular//

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/flow-monitor-module.h"
```

```
using namespace ns3;
```
// **Declaración del entorno donde se trabaja NS-3, implementado en C++. Después de esta declaración no es necesario escribir ns3:: después de cada línea de código.//**

```
NS_LOG_COMPONENT_DEFINE ( "TrabajoFinCarrera" )
```
// **Declaración que permite obtener información, a través de los mensajes por pantalla.//**

//**Aplicación la cual toma el *socket* como parámetro para crear la configuración en el tiempo del programa//**

```
class MyApp : public Application
{  public:
  MyApp ();
  virtual ~MyApp();
 void Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t nPackets,
DataRate dataRate);
```

```
private:
 virtual void StartApplication (void);
 virtual void StopApplication (void);
 void ScheduleTx (void);
 void SendPacket (void);
 Ptr<Socket>     m_socket;
 Address         m_peer;
 uint32_t        m_packetSize;
 uint32_t        m_nPackets;
 DataRate        m_dataRate;
 EventId         m_sendEvent;
 bool            m_running;
 uint32_t        m_packetsSent;
};
```

//Inicialización de las variables//

```
MyApp::MyApp ()
 : m_socket (0),
   m_peer (),
   m_packetSize (0),
   m_nPackets (0),
   m_dataRate (0),
   m_sendEvent (),
   m_running (false),
   m_packetsSent (0)
{ }
MyApp::~MyApp()
{  m_socket = 0; }
```

// **Constructor y Destructor del App**//

```
void
MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t
nPackets, DataRate dataRate)
{
  m_socket = socket;
  m_peer = address;
  m_packetSize = packetSize;
  m_nPackets = nPackets;
  m_dataRate = dataRate;
}
```

//**Inicia la función**//

```
void
MyApp::StartApplication (void)
{
  m_running = true;
  m_packetsSent = 0;
  m_socket->Bind ();
  m_socket->Connect (m_peer);
  SendPacket ();
}
void
MyApp::StopApplication (void)
{  m_running = false;
  if (m_sendEvent.IsRunning ())
    {
      Simulator::Cancel (m_sendEvent);
```

```
   }
  if (m_socket)
   {  m_socket->Close ();  } }
//Activa el envío de paquetes//
void
MyApp::SendPacket (void)
{
  Ptr<Packet> packet = Create<Packet> (m_packetSize);
  m_socket->Send (packet);


  if (++m_packetsSent < m_nPackets)
    {
     ScheduleTx ();  }}
void
MyApp::ScheduleTx (void)
{
  if (m_running)
    {
     Time tNext (Seconds (m_packetSize * 8 / static_cast<double> (m_dataRate.GetBitRate
()))));
     m_sendEvent = Simulator::Schedule (tNext, &MyApp::SendPacket, this);  }}


// Declaración de la función principal del programa, es la función que se ejecuta
primero//
int main (int argc, char *argv[])
{
  uint32_t nWifi = 2;  //Condiciones iniciales y para habilitar o deshabilitar los
componentes de registro//
```

```
  bool echo = false;

  bool wifilogs=false;

  uint32_t packetSize = 1024; // bytes

  uint32_t numPackets = 1000;

  uint32_t dataRate = 1000000; // bps

  std::string phyMode ("DsssRate11Mbps");

  double timeSim = 10.0; // segundos

  uint16_t i;
```

//**Evaluación de los parámetros**//

```
  CommandLine cmd;

  cmd.AddValue ("nWifi", "Number of wifi STA devices (default 2) [>1]", nWifi);

  cmd.AddValue ("echo", "Tell echo applications to log if true (default 1) [0 | 1]"  ,echo);

  cmd.AddValue ("wifilogs", "Turn on all WifiNetDevice log components (default 0) [0 | 1]",
wifilogs);

  cmd.AddValue ("packetSize", "size of application packet sent (default 1024) [bytes]",
packetSize);

  cmd.AddValue ("numPackets", "Number of packets to be transmitted (default 1000)",
numPackets);

  cmd.AddValue ("dataRate", "Data rate for packet generation (default 1000000) [bps]",
dataRate);

  cmd.AddValue ("phyMode", "Wifi Phy mode (default DsssRate11Mbps)", phyMode);

  cmd.AddValue ("timeSim", "Time to simulate and generate traffic (default 10) [s]",
timeSim);

  cmd.Parse (argc,argv);
```

// **Fragmentación de paquetes, de tamaño 2200 bytes**//

```
  Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold",
StringValue ("2200"));

  Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
("2200" ));
```

```
 Config::SetDefault ( "ns3::WifiRemoteStationManager::NonUnicastMode", StringValue
(phyMode));

 if (echo)

  {

    LogComponentEnable  ("PacketSink", LOG_LEVEL_INFO);    }
```

//CREACIÓN DE LOS NODOS//

```
NodeContainer wifiStaNodes;

wifiStaNodes.Create (nWifi);
```

// CREACIÓN DEL CHANNEL//

```
 YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();   //PHY//

 phy.Set ("RxGain", DoubleValue (0) );

 phy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
//FixedRssLossModel//

 YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();   //CHANNEL//

 channel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");

 phy.SetChannel (channel.Create ());
```

//CONEXIÓN WIFI//

```
 WifiHelper wifi = WifiHelper::Default ();

 if (wifilogs)

 {

  wifi.EnableLogComponents ();  // Activa los mensajes para Wifi//

 }

 wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
```

```
//CONFIGURACIÓN DE LA MAC //
 NqosWifiMacHelper mac = NqosWifiMacHelper::Default ();
 mac.SetType ( "ns3::AdhocWifiMac"); // Coloca los ordenadores en modo Ad-Hoc//

// PARA CONTROLAR LA TASA DE BITS//

 wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                               "DataMode" , StringValue(phyMode),
                               "ControlMode", StringValue(phyMode));
// CONFIGURACIÓN DISPOSITIVOS//
 NetDeviceContainer staDevices;
 staDevices = wifi.Install (phy, mac, wifiStaNodes);

// MOVILIDAD//
 MobilityHelper mobility;
 NS_LOG_INFO ("Installing the mobility model");

 mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                "MinX", DoubleValue (0.0),
                                "MinY", DoubleValue (0.0),
                                "DeltaX", DoubleValue (5.0),
                                "DeltaY", DoubleValue (10.0),
                                "GridWidth", UintegerValue (3),
                                "LayoutType", StringValue ("RowFirst"));

 Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
 double distance = 0.0;
```

```
 for (i = 0;i <= wifiStaNodes.GetN();i++)
 {
  positionAlloc->Add(Vector(distance,0.0,0.0));
  distance += 5.0; }  //Separación de los ordenadores//
 mobility.SetMobilityModel ("ns3::ConstantVelocityMobilityModel");  //Velocidad de movilidad constante//
 mobility.SetPositionAllocator(positionAlloc);
 mobility.Install (wifiStaNodes);

//Para el nodo 1//
 Ptr<ConstantVelocityMobilityModel> m0 = wifiStaNodes.Get(1)->GetObject<ConstantVelocityMobilityModel>();
 m0->SetVelocity(Vector(1.0,0.0,0.0));

// Impresión de los NODOS //

 std::cout << "\nNODE POSITIONS" << std::endl; // Obtener información de datos//
 NodeContainer const & n = NodeContainer::GetGlobal ();
 uint16_t j=0;
 for (NodeContainer::Iterator i = n.Begin (); i != n.End (); ++i)
  {
   Ptr<Node> node = *i;
   Ptr<MobilityModel> mob = node->GetObject<MobilityModel> ();
   if (! mob) continue;
   Vector pos = mob->GetPosition ();
   std::cout << "Node " << (j++) << " is at (" << pos.x << ", " << pos.y << ", " << pos.z << ")\n";
  }
```

// **CONFIGURACIÓN DEL IPv4**//

```
 InternetStackHelper stack;
 stack.Install (wifiStaNodes);
 Ipv4AddressHelper address;
 address.SetBase ( "10.1.1.0", "255.255.255.0");
```

// **CONFIGURACIÓN DE LAS INTERFACES**//

```
Ipv4InterfaceContainer staInterfaces;
 staInterfaces = address.Assign (staDevices);
  std::cout << "\nIP ADDRESSES ASSIGNMENT" << std::endl; // Impresión de Direcciones IP //
 for (i=0;i<nWifi;i++)
 {
   std::ostringstream oss;
   staInterfaces.GetAddress(i).Print (oss);
   std::cout << "STA" << i << ": " << oss.str() << std::endl;
 }
```

// **Crea un *packet sink* para recibir estos paquetes en n0 (10.1.1.1)**//

```
 uint16_t port = 50000;
```

// **Cambiar el valor por defecto de tamaño de máxima segmentación (*Maximum Segment Size*) de 536 a 1460 bytes**

```
 Config::SetDefault ( "ns3::TcpSocket::SegmentSize", UintegerValue (1460));
```

//**ON-OFF APP , para generar tráfico**//

```
  NS_LOG_INFO ("Create On-Off UDP App");
   std::cout << "\nCreate On-Off UDP App" << std::endl;
```

```
// CLIENTE (fuente TX)//

// Configura los paquetes generador de tráfico, que genera un paquete cada pocos segundos//

OnOffHelper onOff ("ns3::UdpSocketFactory",

                InetSocketAddress (staInterfaces.GetAddress (0), port));

onOff.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));

onOff.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));

onOff.SetAttribute ("DataRate", DataRateValue (DataRate (dataRate)));

onOff.SetAttribute ("PacketSize", UintegerValue (packetSize));

onOff.SetAttribute ("MaxBytes", UintegerValue (0));  // 0 means no limit

ApplicationContainer source;

source = onOff.Install (wifiStaNodes.Get (1));

source.Start (Seconds (1.1));

source.Stop (Seconds (timeSim));

// SERVIDOR (sink)

// Crea un paquete receptor para recibir paquetes en modo AP o nodo n[nWifi-1] (10.1.1.nWifi)//

PacketSinkHelper sink ("ns3::UdpSocketFactory",

                InetSocketAddress (staInterfaces.GetAddress (0), port));

source = sink.Install (wifiStaNodes.Get (0));

source.Start (Seconds (0.5));

source.Stop (Seconds (timeSim));


Simulator::Stop (Seconds (timeSim));

//Creación de los archivos PCAP, Tracing//

phy.EnablePcap ("proyecto", staDevices, false);

std::cout << "\nCreating capture file: proyecto-0-0.pcap";

std::cout << "\nCreating capture file: proyecto-1-0.pcap" << std::endl;
```

```
// Instala FlowMonitor en todos los nodos//

FlowMonitorHelper flowmon;

Ptr<FlowMonitor> monitor = flowmon.InstallAll();


//Inicia la Simulación//

Simulator::Run ();


// Imprime parámetros estáticos//

std::cout << "\nSTATS (Thput includes IP & UDP headers)" << std::endl;

monitor->CheckForLostPackets ();

Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());

std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i !=
stats.end (); ++i)

  {

    Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);

    std::cout << "Flow " << i->first << " (" << t.sourceAddress << " -> " <<
t.destinationAddress << ")\n";

    std::cout << "  Tx Bytes (includes IP & UDP headers):   " << i->second.txBytes << "\n";

    std::cout << "  Rx Bytes (includes IP & UDP headers):   " << i->second.rxBytes << "\n";

    std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / (timeSim - 1.1) / 1000.0 /
1000.0  << " Mbps\n";

  }

Simulator::Destroy ();

return 0; }
```

Código: Variación de la Potencia de Transmisión TX.

Al código anterior, sólo se le agregaron las siguientes líneas de código:

```
//CONSUMO DE ENERGÍA ASIGNACIÓN DE POTENCIA TRANSMITIDA//
 phy.Set ("TxPowerStart", DoubleValue(90.0)); //Valor de la potencia a variar//
 phy.Set ("TxPowerEnd", DoubleValue(90.0));
 phy.Set ("TxPowerLevels", UintegerValue(1));
 phy.Set ("TxGain", DoubleValue(0) );
 phy.Set ("RxGain", DoubleValue (0) );
// Para cambiar el umbral de detección de la energía para definir el alcance del radio//
 phy.Set ("EnergyDetectionThreshold", DoubleValue(-71.9842));
 phy.Set ("CcaMode1Threshold", DoubleValue(-74.9842));
```

Para poder realizar la variación de la potencia, solo había que ir cambiar el valor numérico, en la línea de código**:** ***phy.Set*** *("TxPowerStart", DoubleValue(X));* siendo X, el valor en unidades dBm.

[ANEXO 2]

- **Potencias típicas en dBm**

dBm Nivel	Potencia	Notas
80 dBm	100 kW	potencia típica de transmisión de una estación de radio FM con un rango de 30-40 millas
60 dBm	1 kW = 1000 W	Radiación típica combinada de RF de un horno de microondas
40 dBm	10 W	Potencia entregada a las antenas de telefonía móvil
36 dBm	4 W	Salida típica de potencia para una banda de radio ciudadana (27 MHz) en muchos países
33 dBm	2 W	Máxima salida de potencia para un teléfono celular UMTS/3G (teléfono de potencia clase 1) Máxima salida de potencia para un teléfono celular GSM850/900
30 dBm	1 W = 1000 mW	Fuga RF típica de un horno de microondas Máxima salida de potencia para un teléfono celular GSM1800/1900
27 dBm	500 mW	Potencia típica de transmisión de un teléfono celular Máxima salida de potencia para un teléfono celular UMTS/3G (teléfono de potencia clase 2)

26 dBm	400 mW	
25 dBm	316 mW	
24 dBm	250 mW	Máxima salida de potencia para un teléfono celular UMTS/3G (teléfono de potencia clase 3)
23 dBm	200 mW	
22 dBm	160 mW	
21 dBm	125 mW	Máxima salida de potencia para un teléfono celular UMTS/3G (teléfono de potencia clase 4)
20 dBm	100 mW	Bluetooth Estándar Clase 1 , cobertura de 100 m (máxima potencia de salidad para un transmisor FM). Potencia típica de un router inalámbrico WiFi.
15 dBm	32 mW	Potencia típica de de transmisión de WiFi en laptops.
10 dBm	10 mW	
6 dBm	4.0 mW	
5 dBm	3.2 mW	
4 dBm	2.5 mW	Estándar Bluetooth clase 2, cobertura de 10 m
3 dBm	2.0 mW	(=1.9952623 mW)
2 dBm	1.6 mW	
1 dBm	1.3 mW	
0 dBm	1.0 mW = 1000 µW	Estándar Bluetooth Clase 3, cobertura de 1 m.
-1 dBm	794 µW	
-3 dBm	501 µW	
-5 dBm	316 µW	
-10 dBm	100 µW	Potencia de señal típica de recepción de una red inalámbrica WiFi(−10 a −30 dBm)
-20 dBm	10 µW	
-30	1.0 µW =	

dBm	1000 nW	
-40 dBm	100 nW	
-50 dBm	10 nW	
-60 dBm	1.0 nW = 1000 pW	
-70 dBm	100 pW	Rango típico (-60 a -80 dBm) de potencia de señal inalámbrica (802.11x) recibida por una red
-80 dBm	10 pW	
-111 dBm	0.008 pW = 8 fW	Thermal noise floor Para banda comercial GPS single channel signal bandwidth (2 MHz)
-127.5 dBm	0.178 fW = 178 aW	Potencia típica recibida de un satélite GPS
-174 dBm	0.004 aW	Ruido térmico para un ancho de banda de 1 Hz a temperatura ambiente.
-194 dBm	0.00004 aW	Ruido térmico para un ancho de banda de 1 Hz en el espacio exterior. (4 kelvin)
-∞ dBm	0 W	La potencia cero no se expresa correctamente en dBm (su valor es menos infinito)

Fuente: **http://es.wikipedia.org/wiki/dbm** [Consulta Septiembre, 2010]

[ANEXO 3]

PARÁMETROS WIFI

A continuación se expondrán los parámetros del estándar 801.11 b/g que servirán para optimizar enlaces WIFI, además de otros parámetros que existen aunque no influyen en el rendimiento. WIFI 802.11 b/g dispone de dos modos: [7]

A) **Modo b**: la velocidad de conexión en datos WIFI en este modo es como máximo de 11 megabits, aunque en la capa **TCP/IP,** se obtendrá el máximo de 6 megabits que vendrían a ser unos 600 Kb/s. Este modo es más estable y en las especificaciones de los aparatos, dispone de mayor sensibilidad y mayor ganancia

B) **Modo g:** en este modo se puede obtener un máximo de transferencia WIFI de 54 megabits, aunque en la capa **TCP/IP** se puede alcanzar un máximo de 34 megabits, lo que equivaldría a tener una conexión de unos 3 Mib/s. Opera bien cuando las distancias son cortas y con pocos obstáculos o interferencias.

Parámetros

A) ***Preamble*** (preámbulo): define la longitud del bloque de comprobación de redundancia cíclica (**CRC**) para la comunicación entre el punto de acceso y los clientes inalámbricos. En aquellas redes en las que haya demasiado tráfico, se recomienda seleccionar un preámbulo ***short*** (corto), mientras que para las redes con menor tráfico es aconsejable optar por un preámbulo ***long*** (largo).

B) ***Beacon interval*** (intervalo de baliza) es el tiempo transcurrido entre transmisiones de balizas. La baliza es una señal enviada por el cliente inalámbrico o ***router*** para indicar a la red que aún está activo. El valor debe establecerse entre 1 y 1.000 milisegundos

C) ***Dtim interval*** (intervalo) es el tiempo que transcurre entre los envíos de mensajes a los clientes de la red. ***Dtim*** significa ***delivery traffic indication message*** (mensaje indicativo de tráfico de entrega) y se trata de un mensaje remitido a clientes de la red con funciones de ahorro de energía que les informa de que, para recibir cierta información, deben estar activos. El valor predeterminado del intervalo dtim es 1. Son mensajes que permiten usar el ***standby*** para ahorrar energía, útil en portátiles, y no afectan al rendimiento de transmisión de datos.

D) ***Rts threshold*** (umbral rts o de petición de envío) es el tiempo que aguardará el punto de acceso antes de mandar una petición de envío (**RTS**) al cliente. Los mensajes **RTS** avisan al ordenador, red o servidor de que el cliente está intentando mandar datos y acceder con privilegios al ordenador o red durante la transmisión o recepción de los datos. Si un cliente experimenta dificultades al transmitir los datos a un ordenador, una red o un servidor, deberá disminuir el umbral. Este parámetro sólo afecta al rendimiento.

E) ***Fragmentation threshold*** (umbral de fragmentación) es el nivel máximo que alcanzará el punto de acceso al enviar la información en paquetes antes de que estos se fragmenten. Si le cuesta mandar información probablemente se deba al tráfico en la red y a la colisión de los datos transmitidos. Esto se puede solucionar dividiendo la información en fragmentos. Cuanto más bajo sea el umbral de fragmentación, menor será el paquete antes de que se divida en fragmentos. Si establece el máximo (2.346), la fragmentación quedará prácticamente deshabilitada. Este valor siempre debe ser menor que el **RTS.**

F) ***Multicast rate:*** velocidad máxima permitida para los paquete multidifusión, es aconsejable mantener este valor lo más bajo posible ya que la multidifusión puede saturar las redes WIFI con facilidad.

g) ***Wmm (wi-fi multimedia):*** esta función activa la denominada ***quality of service*** (calidad del servicio, **QoS**) para aplicaciones multimedia, tales como voz a través de **IP** (**VoIP**) y vídeo.

ARQUITECTURA CLIENTE/SERVIDOR

Es una relación entre procesos, ejecutándose en máquinas separadas. Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores. Es una red donde se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores). Es de fácil mantenimiento ya que al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio. Este tipo de red puede utilizarse conjuntamente en caso de que se este utilizando en una red mixta.

El cliente y el servidor interactúan por un mecanismo de intercambio de mensajes.

Cliente: inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo o través de redes LAN o WAN. Es un consumidor de servicios. Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo). Espera y recibe las respuestas del servidor. Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Servidor: cualquier recurso de computador dedicado a responder a los requerimientos del cliente. El servidor es un proveedor de servicios. Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente. Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).No es frecuente que interactúen directamente con los usuarios finales.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. [9]

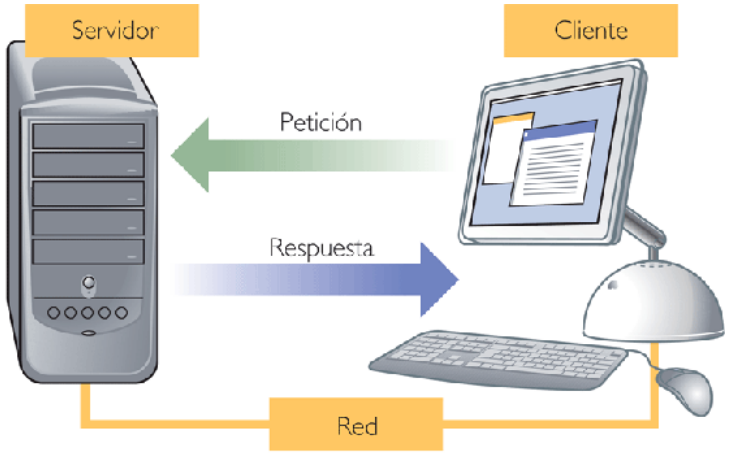


Figura 8. Ejemplo gráfico de la arquitectura cliente servidor.

Fuente: http://eprints.ucm.es/8858/1/microsoft_word-memoria_ssii_final.pdf [Consulta Mayo, 2011]

ELEMENTOS DE LA ARQUITECTURA CLIENTE/SERVIDOR

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico. Con el objetivo de definir y delimitar el modelo de referencia de una arquitectura Cliente/Servidor, se debe identificar los componentes de dicha arquitectura, considerando que toda aplicación de un sistema de información está caracterizada por tres componentes básicos: [9]

a) Presentación/Captación de Información
b) Procesos
c) Almacenamiento de la Información

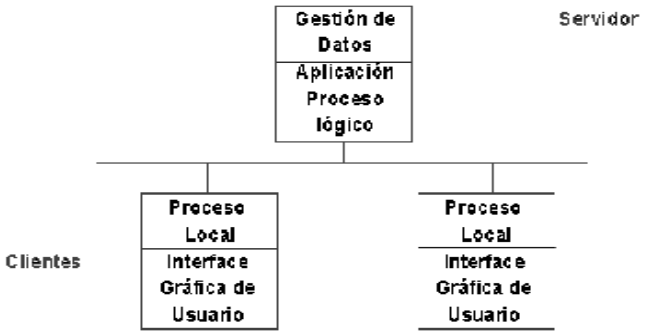


Figura 9. Aplicaciones Cliente/Servidor

Fuente: http://aprendiendo-oracle.blogspot.com/2010/07/arquitectura-cliente-servidor_18.html [Consulta Mayo, 2011]

Y se integran en una arquitectura Cliente/Servidor en base a los elementos que caracterizan dicha arquitectura, es decir:

a) Puestos de Trabajo
b) Comunicaciones
c) Servidores

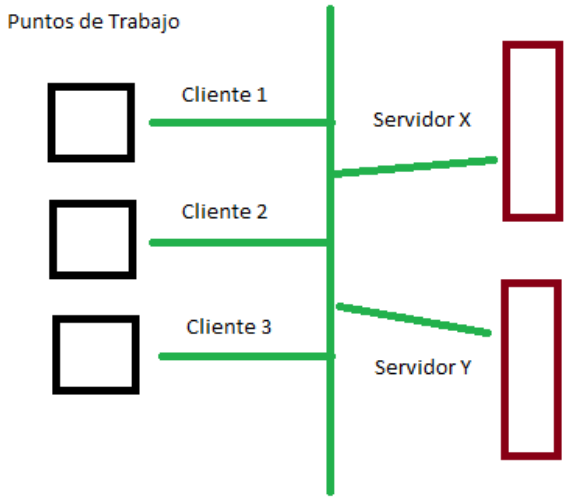


Figura 10. Elementos de Estructura Cliente Servidor.

[ANEXO 4]

Manual Instructivo para la Instalación del ns-3 en el Sistema Operativo UBUNTUS

Preliminares:

1. Tener instalado el Sistema Operativo Ubuntu, en un computador. Para mayor información sobre instalación de este sistema operativo: http://sliceoflinux.com/2010/10/10/instalar-ubuntu-10-10-paso-a-paso/
2. Descargar ns-3.11, desde la página oficial de Ns-3: www.nsnam.org
3. El core (núcleo) de Ns-3 requiere gcc / g + + instalación de 3,4 o mayor, y Python 2.4 o superior. Las diferentes aplicaciones de Ns-3, requieren un soporte adicional.

Una vez realizados estos dos primeros pasos antes de la instalación, es importante instalar las aplicaciones y librerías que Ns-3 necesita (algunas opcionales), pero se recomienda instalar todas las que se pueda. Las instrucciones de instalación y dependencias con otras librerías, se pueden ver en la página siguiente: www.nsnam.org/getting_started.html.

Después de instalar las librerías y aplicaciones necesarias, se debe instalar la siguiente lista de paquetes adicionales, vía consola de UBUNTU, para darle soporte a las aplicaciones de Ns-3.lo siguiente:

- Requerimientos mínimos para C++, para poder ejecutar programas ns-3.

```
sudo apt-get install gcc g++ python
```

- Requerimientos mínimos para Python, para poder ejecutar programas ns-3.minimal

```
sudo apt-get install gcc g++ python python-dev
```

- Debugging:

```
sudo apt-get install gdb valgrind
```

- GNU Scientific Library (GSL) soporte para errores de modelos WiFi

```
sudo apt-get install gsl-bin libgsl0-dev libgsl0ldbl
```

- The Network Simulation Cradle (nsc)

```
sudo apt-get install flex bison
```

- Para instalar gcc-3.4 para algunos protocolos de Network Simulation Cradle (nsc):

```
sudo apt-get install g++-3.4 gcc-3.4
```

- Para leer paquete pcap

```
sudo apt-get install tcpdump
```

- Base de Datos para valores estadísticos del framework

```
sudo apt-get install sqlite sqlite3 libsqlite3-dev
```

- Configuracion del sistema GTK

```
sudo apt-get install libgtk2.0-0 libgtk2.0-dev
```

- Para experimentar con maquinas virtuales

```
sudo apt-get install vtun lxc
```

- Soporte para check-style.py estilo del código del programa

```
sudo apt-get install uncrustify
```

- Documentación: Doxygen documentation:

```
sudo apt-get install doxygen graphviz imagemagick
sudo apt-get install texlive texlive-pdf texlive-latex-extra texlive-generic-extra texlive-generic-recommended
```

- Tutorial y manual de ns-3

```
sudo apt-get install texinfo dia texlive texlive-pdf texlive-latex-extra texlive-extra-utils texlive-generic-recommended texi2html
```

Una vez instaladas todas las librerías, se compila el código de ns-3, para poder instalarlo. Hay que incluirse dentro de las carpetas: Downloads/ns-3-11, y se debe copiar el siguiente código: Código para compilar:

- **bunzip2 ns-allinone-3.11.tar.bz2** (se van desencriptando cada uno de los archivos contenidos dentro de estas carpeta)
- **tar xvf ns-allinone-3.9.tar**
- **cd ns-allinone-3.11** (se incluye dentro de la carpeta donde se ejecutan los programas.)

Después para compilar este código, se utiliza el siguiente comando: **./build.py.** Una vez instalado empezar con el tutorial aquí: www.nsnam.org/docs/release/tutorial.html

Comandos de UBUNTUS utilizados para la instalación del ns-3

Para instalar comandos:

- **sudo**: clave de administración, derecho de administrador. Es la clave de inicio para entrar en el sistema operativo.
- **apt-get install**: para instalar.
- **cd**: incluirse en una carpeta
- **cd ..** : salir de la carpeta actual
- **ls**: muestra el directorio
- **gedit** :ver archivo de texto con editor de texto.