SIMULACIÓN Y PARAMETRIZACIÓN DE REDES INALÁMBRICAS (WLAN 802.11b) CON LA HERRAMIENTA SOFTWARE *NETWORK SIMULATOR*

ELKIN TARAZONA VELÁSQUEZ
JAVIER MAURICIO VARGAS FLÓREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2005

SIMULACIÓN Y PARAMETRIZACIÓN DE REDES INALÁMBRICAS (WLAN 802.11b) CON LA HERRAMIENTA SOFTWARE *NETWORK SIMULATOR*

ELKIN TARAZONA VELÁSQUEZ
JAVIER MAURICIO VARGAS FLÓREZ

Este proyecto es presentado como requisito para optar al título de Ingeniero Electrónico

Director
PHD. OSCAR GUALDRÓN GONZÁLEZ
Codirector
MI(c). SAMUEL GONZALO PINZÓN BARRIOS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2005

AGRADECIMIENTOS

Los autores expresan su agradecimiento y reconocimiento a:

Nuestras familias por su apoyo incondicional, en especial nuestros padres quienes con su apoyo ayudaron a convertir esta meta en realidad.

A Dios, quien desde el cielo ilumina nuestro camino.

Doctor Oscar Gualdrón González, director del proyecto y al Ingeniero Samuel Gonzalo Pinzón, codirector del proyecto, por su orientación y colaboración.

La Especialización en Telecomunicaciones.

La Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones

La Universidad Industrial de Santander.

CONTENIDO

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

TITULO: SIMULACIÓN Y PARAMETRIZACIÓN DE REDES INALÁMBRICAS (WLAN 802.11b) CON LA HERRAMIENTA SOFTWARE NETWORK SIMULATOR.*

AUTORES:
ELKIN TARAZONA VELÁSQUEZ
JAVIER MAURICIO VARGAS**

PALABRAS CLAVES:
Simulación, Parametrización, redes, inalambrico, Network Simulator, Linux.

DESCRIPCIÓN:
Manteniendo la tendencia general de las telecomunicaciones hacia los enlaces inalámbricos este trabajo hace énfasis en las simulaciones inalámbricas con la herramienta de simulación NETWORK SIMULATOR basado en ambiente LINUX.

Se buscó la documentación necesaria para describir de manera sencilla y detallada el desarrollo de simulaciones inalámbricas con esta herramienta, revisando proyectos en otras universidades, nuevos desarrollos y avances de sus creadores en materia de este campo de estudio y manuales producidos por diferentes autores, para así no solo determinar los limites y alcances de la herramienta en materia de simulaciones de redes inalámbricas sino generar una documentación específica en materia de simulación de redes inalámbricas bajo el protocolo 802.11b.

Se definieron tres escenarios de observación que fueron: espacio libre, espacio cerrado y multiusuario; esto con el fin de comparar los resultados del simulador en estos ambientes sencillos y reales, definiendo el rango de error de los datos simulados respecto de los datos obtenidos de la realidad. Esto con el fin que futuros estudios basados en esta herramienta tengan en cuenta que el simulador no es del todo congruente con los datos que se miden en la realidad.

Las campañas de medición no solo incluyeron tráfico sino potencia, esto con el fin de determinar los parámetros físicos que modelan las características especiales de cada escenario y así el modelo de radiopropagación escogido para este estudio que describa de manera acorde el ambiente en el que se esta trabajando. Ya hechas las respectivas mediciones y habiendo analizado los datos se definieron los rangos de error correspondientes para cada simulación.

El resultado de este trabajo queda representado por una lista de comandos y una estructura de programación para redes inalámbricas con todos los parámetros no solo físicos sino de simulación que hay que tener en cuenta para utilizar esta herramienta de simulación.

---

* Trabajo de grado.

** Facultad de ingenierías físico-mecánicas escuela de ingeniería eléctrica, electrónica y telecomunicaciones. Director: PHD. Oscar Gualdrón González.

TITLE: SIMULATION AND PARAMETRIZATION OF WIRELESS NETWORKS (WLAN 802.11b) WITH THE NETWORK SIMULATOR SOFTWARE TOOL.[*]

AUTHORS:
ELKIN TARAZONA VELÁSQUEZ
JAVIER MAURICIO VARGAS[**]

KEYWORDS:
Simulation, Parametrization, Network, Wireless, Network Simulator, Linux.

DESCRIPTION:
Maintain the general tend of the telecommunications towards the wireless links, this work done emphasis in the wireless simulations with the simulation tool Network Simulator working in Linux Environment.

We looked for the necessary documentation to describe of single way and detailed the development of wireless simulations with this tool, checking projects in other universities. News development and advances of their creators in subject of field of study and handbooks produced for different authors, not only to fix the limits and reaches of the tool in subject or wireless networks simulations but generate a specific documentation in subject of wireless networks simulations under the 802.11b protocol.

We defined 3 observation places that were: Free space, closed space and multi-user; this with the end of compares the results of the simulator in this single and real environment, defining the error rank if the simulated data in relation to the data obtained in the reality. This with the end that future studies based on this tool bear in mind that the simulator is not of all congruent with the date that are measured in the reality.

The measurement campaign not only include traffic but power, this with the end of decided the physical parameter that model the special characteristics of each place and so the radio propagation model selected for this study that describe of good manner the working environment. When the respective measures had been done and the date had analyzed the error rank were defined corresponding to each simulation.

The result of this work is represented with a list of commands and a structured programming to wireless networks with all the parameters not only physical but of simulation that we must have in mind to use this simulation tool.

---

[*] Tesis of grade.

[**] Faculty of Physical-mechanical Engineering's. Electrical, Electronic and Telecommunications School of Engineering's. Director: PHD. Oscar Gualdrón González.

INTRODUCCIÓN

La tendencia de las telecomunicaciones es ofrecer servicios de acceso de usuario basados en la tecnología inalámbrica, debido a que es una alternativa que ofrece facilidad de manejo, movilidad, versatilidad, bajo costo y sobre todo que ya se cuenta con unos altos estándares de calidad de servicio. Por este motivo las comunicaciones inalámbricas, aunque apenas están evolucionando, prometen ser una tecnología de acceso de amplia aceptación.

En Colombia los sistemas de acceso inalámbrico no han tenido mucha acogida como en otros países, pero la tendencia mundial indica que pronto se instaurarán debido a sus características. Los sistemas inalámbricos en Colombia apenas abarcan transmisión de voz (sistemas celulares) y en algunos casos transmisión de datos punto a punto, pero no existen los sistemas de acceso masivo debido a la poca demanda y cantidad de dispositivos inalámbricos.

NS (Network Simulator) actualmente en la versión 2, permite realizar simulaciones de múltiples tipos de redes (cableadas, inalámbricas y por satélite); para ello utiliza un lenguaje de SCRIPT llamado TCL que nos permite ir generando el modelo, además se dispone de una interfaz gráfica llamada NAM que permite visualizar las simulaciones e incluso crear y editar los modelos a simular. NS es una gran herramienta que puede ayudar en muchos campos a la hora de realizar pruebas o generar nuevos tipos de redes.

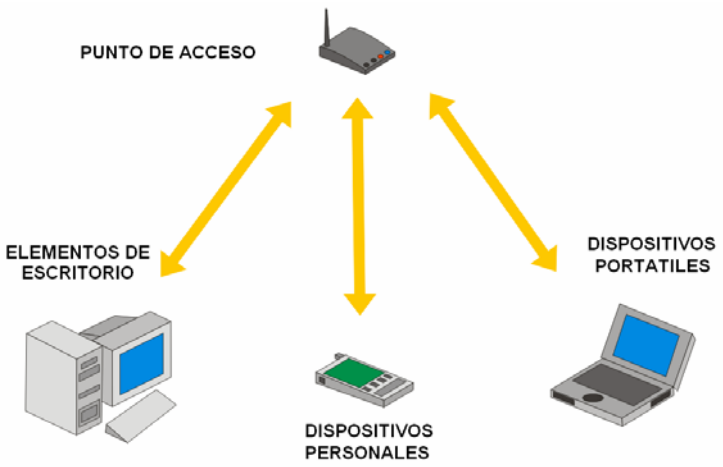
Este trabajo plantea estudiar los sistemas inalámbricos para establecer los principales parámetros de simulación, y así generalizar acerca de los resultados obtenidos por medio de esta herramienta software y poder generar estudios de posibles redes inalámbricas sin necesidad de llegar a construirlas.

1. FUNDAMENTOS SOBRE REDES INALÁMBRICAS

1.1 CONCEPTOS GENERALES

Las WLANs[1] típicamente consisten de dispositivos portátiles (o de escritorio) que se conectan a dispositivos fijos llamados puntos de acceso, figura 1, o entre ellos, figura 2, vía señales de radio o infrarrojo. Las implementaciones de las WLANs abarcan todas las modalidades posibles desde las WPANs (Wireless Personal Area Networks), WMANs (Wireless Metropolitan Area Networks), hasta las WWANs (Wireless Wide Area Networks). Las WPANs son redes inalámbricas de corto alcance, generalmente para uso en interiores a pocos metros, mientras que las redes inalámbricas tipo WAN y MAN consista de torres y antenas que transmiten ondas de radio o usan tecnología de microondas para conectar redes de área local, utilizando enlaces punto-punto y punto-multipunto (Las siguientes figuras muestran las configuraciones inalámbricas mas utilizadas).

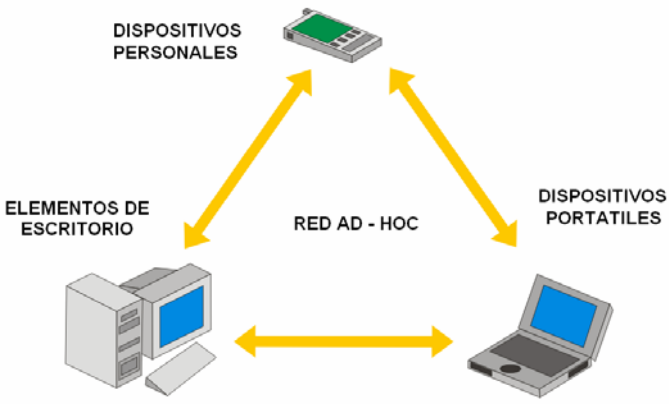
Figura 1. Configuración de una red inalámbrica en modo estructura.



Fuente, autores.

[1] WLAN: Wireless Local Area Network, Red de Área Local Inalámbrica.

Figura 2. Configuración de una red inalámbrica en modo Ad Hoc.



Fuente, autores.

1.1.1 El estándar IEEE 802.11x. Es un conjunto de normas que definen las características físicas y de enlace de datos de las redes inalámbricas en general, dictadas por el IEEE[2] , para establecer un estándar de tecnología inalámbrica en el mercado mundial, dentro del IEEE 802.11x existen varios estándares definidos:

Tabla 1. Resumen de los estándares 802.11x.

ESTANDAR	DESCRIPCION
802.11	Estándar para redes inalámbricas con línea visual.
802.11a	Permite velocidades teóricas máximas de hasta 54 Mbps, apoyándose en la banda de los 5GHz.
802.11b	Proporciona 11 Mbps usando DSSS, conocido comúnmente como Wi-Fi[3]. Es el estándar más utilizado en las comunidades inalámbricas
802.11g	Utiliza la banda de 2,4 GHz, transmitiendo a velocidades teóricas de 54 Mbps.

[2] IEEE: Institute of Electrical and Electronic Engineers.

[3] Wi- Fi: Wireless Fidelity, Término registrado para certificar productos IEEE 802.11b capaces de interoperar con los de otros fabricantes

El estándar IEEE 802.11 define varias tasas de transmisión en la capa física para los diferentes tipos de WLANs, tales como 1, 2, 5.5 y 11 Mbps para 802.11b, y tasas de 6, 9, 18, 24, 36, 48 y 54 Mbps para el 802.11a y 802.11g. A continuación se presenta un cuadro comparativo de los diferentes estándares existentes para conexiones inalámbricas:

Tabla 2. Principales estándares WLAN.

Estándar	Velocidad máxima	Interface de aire	Frecuencia
802.11b	11 Mbps	DSSS[4]	2.4 GHz
802.11a	54 Mbps	OFDM[5]	5.0 GHz
802.11g	54 Mbps	OFDM/DSSS	2.4 GHz
HomeRF2	10 Mbps	FHSS[6]	2.4 GHz
HiperLAN2	54 Mbps	OFDM	5.0 GHz
Bluetooh	1 Mbps	FHSS	2.4 GHz

Tomado de **www.e-advento.com/tecnologia/estandares.php**

1.1.2 Tecnologías. Existen varias tecnologías utilizadas en redes inalámbricas; el empleo de cada una de ellas depende de la aplicación. Cada tecnología tiene sus ventajas y desventajas. Las más importantes en este género son el Infrarrojo (Infrared), Banda Angosta (Narrow band) y el Espectro Extendido (Spread Spectrum). A continuación se presenta un resumen para dar una idea general de estas tecnologías de acceso inalámbrico:

---

[4] DSSS: Direct Sequence Spread Spectrum, tecnología de transmisión WLAN, donde la señal de datos enviada es combinada con una secuencia de bits llamada chip que divide los datos de usuario de acuerdo a un radio expandido.

[5] OFDM: Orthogonal Frequency Division Multiplexing, es un método de modulación digital por medio del cual una señal es partida en varios canales de banda estrecha en diferentes frecuencias.

[6] FHSS: Frequency Hopping Spread Spectrum, utiliza una portadora de banda angosta que cambia la frecuencia en un patrón conocido tanto por el transmisor como por el receptor.

Infrarrojo: Utilizan muy altas frecuencias, justo abajo del espectro de la luz visible para transportar datos no pudiendo penetrar objetos opacos. El infrarrojo directo no es práctico para usuarios móviles pero su uso es prácticamente para conectar dos redes fijas. La tecnología reflectiva [7] no requiere línea de vista pero está limitada a cuartos individuales en zonas relativamente cercanas.

Banda Angosta: Se transmite información en una radio frecuencia específica, con un ancho de banda lo más angosto posible y el receptor filtra todas aquellas frecuencias que no son de su competencia. La desventaja de esta tecnología es el uso amplio de frecuencias, uno para cada usuario, lo cual es impráctico si se tienen muchos usuarios.

Espectro extendido: La gran mayoría de los sistemas inalámbricos emplean la tecnología de Espectro Extendido (Spread Spectrum). Se consume más ancho de banda pero se produce una señal más fuerte, permitiendo la reducción de interferencia entre la señal procesada y otras señales no esenciales o ajenas al sistema de comunicación.

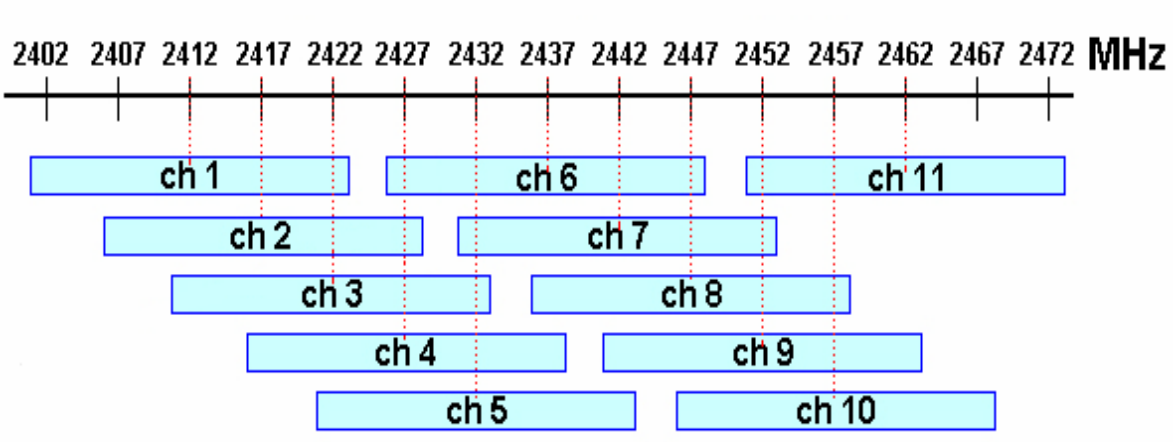
1.2 ESTÁNDAR 802.11b

El estándar de comunicación 802.11b es el sistema de comunicación inalámbrica más extendido entre todas las comunidades inalámbricas del mundo, se diferencia de los otros en la velocidad de transmisión 11Mbps y frecuencia que usa para emitir (2.4GHz). La velocidad máxima del estándar 802.11b actualmente es de 11Mbps en condiciones óptimas (ambiente libre de interferencia y a muy corta distancia.), en condiciones algo más desfavorables 802.11b reduce automáticamente la velocidad de transmisión de la capa física teniéndose tres velocidades 5.5, 2, 1 Mbps.

[7] Tecnología que utiliza la difusión y la reflexión como medo de radiopropagación del infrarrojo.

Se cuenta con 3 canales que no se sobreponen, figura 3, y se puede tener cerca de 32 estaciones por cada Punto de Acceso. El protocolo 802.11b está basado en DSSS (Direct Sequence Spread Spectrum) como técnica de modulación.

Figura 3. Canales inalámbricos 802.11b.



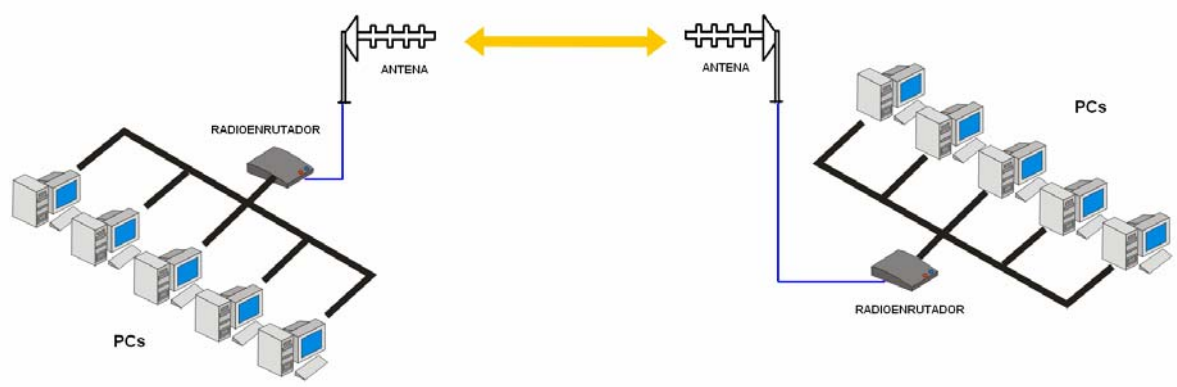
Tomado de **www.wifind.com.ar**

Notar que la asignación de más canales independientes corresponde a los canales 1, 6, y 11; luego 2 y 7, 3 y 8, 4 y 9, y 5 y 10. Es decir que la distancia para que no exista interferencia entre canales es de exactamente 5 canales.

Los productos bajo este estándar (802.11b) no son compatibles con Bluetooth, y aunque se interfieren, el Bluetooth no emite mas allá de 10 a 15 metros por lo tanto no existe problema; el 802.11b tampoco es ínter operable con los de 802.11a ya que utilizan diferentes bandas de frecuencias (2,4 GHz y 5GHz respectivamente), además de tener diferentes velocidades de transmisión (11Mbps y 56Mbps respectivamente). Respecto al 802.11g, el 802.11b trabaja a la misma frecuencia de transmisión (2,4 GHz) con diferentes velocidades (54Mbps y 11Mbps respectivamente).

1.2.1 Aplicaciones de redes inalámbricas 802.11b en exteriores (outdoor). En estas aplicaciones se recomienda que la trayectoria entre el emisor y el receptor esté totalmente libre de obstáculos[8]. En estas aplicaciones los rangos de cobertura pueden llegar a varios kilómetros según la configuración total de la red.

Figura 4. Conexión inalámbrica externa punto a punto.



Fuente, autores.

Dentro de las variables que determinan la cobertura de un Sistema Outdoor se puede mencionar:

- Longitud y tipo de Cable instalado
- Ganancia de la antena del transmisor y receptor.
- Uso de Amplificadores bidireccionales junto a la antena del Nodo.
- Velocidad de procesado y capacidad de buffer.

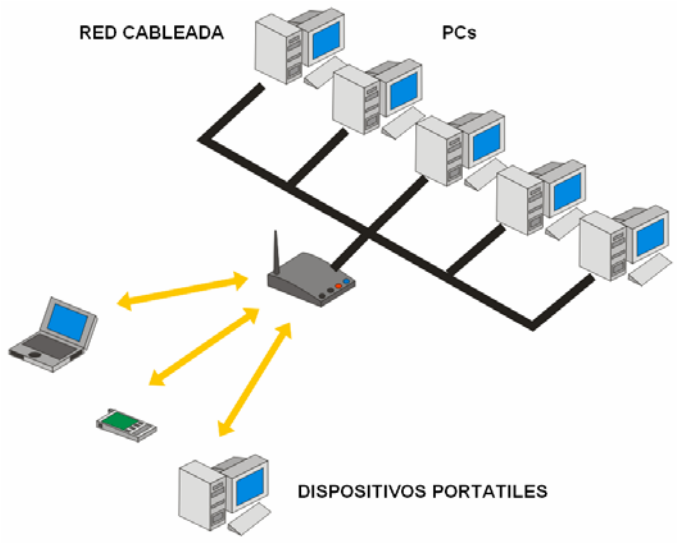
Debido a que el estándar 802.11b es el más difundido a nivel mundial para conexiones inalámbricas de datos, éste goza de una amplia variedad de aplicaciones donde que se pueden mencionar las más importantes: enlaces punto a punto, enlaces punto a multipunto y Servicio de Internet Inalámbrica.

[8] A esta característica se le llama "línea de vista" y consiste en poder ver el emisor desde la ubicación del receptor o viceversa.

- Enlace punto a punto es aquel que permite conectar ya sea dos LANs, PCs a LANs, o enlaces entre PCs con varios kilómetros de distancia.
- El enlace punto a multipunto provee enlaces entre distintos puntos de una sucursal manera sencilla y económica.
- Mediante enlaces inalámbricos bajo el estándar 802.11b también es posible proveer de servicio de Internet inalámbrico a una comunidad, una cooperativa, un edificio, una universidad o sucursales de alguna compañía.

1.2.2 Aplicaciones de redes inalámbricas 802.11b en interiores (indoor). En estas aplicaciones (antena integrada a la Tarjeta de Red Inalámbrica) la distancia entre tarjeta de red y el AP puede llegar a los 300m cuando no existen paredes / obstáculos en la trayectoria. Cuando existen obstáculos en la trayectoria, estas distancias se reducen de acuerdo al tamaño del obstáculo en cuestión. Valores típicos pueden ubicarse dentro los 100m. Las redes LAN inalámbricas son un claro ejemplo de esta aplicación como así también aplicaciones de provisión de servicio de Internet a un conjunto de computadoras a través de una conexión de red cableada.

Figura 5. Conexión inalámbrica típica dentro de una oficina.



Fuente, autores.

En la actualidad las redes de datos LAN de empresas y oficinas crecen con el uso de soluciones inalámbricas de fácil instalación, donde no es necesario ningún tendido de cables y permitiendo libre movilidad de los PCs.

Muchas aplicaciones requieren conexiones móviles de diferentes dispositivos a la red LAN de la empresa. Por medio de una solución inalámbrica los dispositivos pueden moverse libremente desde un área a otra sin perder su conexión a la red LAN, y contratando un servicio de conexión a Internet es posible conectar los dispositivos móviles a la red mundial.

1.2.3 Atenuación e interferencia en redes inalámbricas 802.11b.

Debido a la naturaleza de la tecnología de radio, las señales de radiofrecuencia pueden desvanecerse o bloquearse por materiales medioambientales, la inspección del lugar ayuda a identificar los elementos que afectan negativamente a la señal inalámbrica.

Tabla 3. Atenuación en materiales típicos de construcción para las ondas de radio.

Material	Ejemplos	Atenuación
Madera	Tabiques	Baja
Vidrio	Ventanas	Baja
Amianto	Techos	Baja
Yeso	Paneles interiores	Baja
Ladrillo	Paredes interiores y exteriores	Media
Hojas	Árboles y plantas	Media
Agua	Lluvia/ Niebla	Alta
Cerámica	Tejas	Alta
Papel	Rollos de papel	Alta
Vidrio con alto contenido de plomo	Ventanas	Alta
Metal	Vigas y armarios	**Muy alta**

Tomado de **www.e-advento.com/tecnologia/interfyatenua.php**

No todos los elementos circundantes presentan la misma absorción para las ondas de radio y es por esto que la radio propagación se ve afectada en

función no solo del número de elementos sino en gran parte por el material que los constituye.

Las redes inalámbricas operan en un espectro de frecuencias utilizado por otras tecnologías, pueden existir interferencias que pueden afectar negativamente al rendimiento. Las tecnologías que pueden producir interferencias son el Bluetooth, los hornos microondas, algunos teléfonos inalámbricos y otras redes WLAN.

1.3 MODELOS DE RADIO PROPAGACIÓN

Los modelos de radio propagación son usados para predecir la potencia de la señal recibida. En la capa física de cada nodo inalámbrico hay un umbral de recepción el cual determina cuando un paquete es perdido o recibido correctamente. Si la señal de potencia es menor que el umbral de recepción el paquete es marcado como un error y desechado por la capa MAC.

Existen tres modelos de radio propagación los cuales son descritos a continuación, estos son conocidos como el Modelo de Espacio Libre (Free Space Model), Modelo de Reflexión de dos Rayos a tierra (Two-Ray Ground Reflexion Model) y el Modelo de Sombra (Shadowing Model).

Modelo de espacio libre [15]. El modelo de propagación de espacio libre asume la condición de propagación ideal que solamente hay un camino claro de línea de vista entre el transmisor y el receptor. Friis presentó la siguiente ecuación para calcular la potencia de la señal recibida en espacio libre a una distancia *d* desde el transmisor.

$$\Pr(d) = \frac{Pt * Gt * Gr * \lambda\text{^}2}{(4 * \pi)\text{^}2 * d\text{^}2} \qquad (1)$$

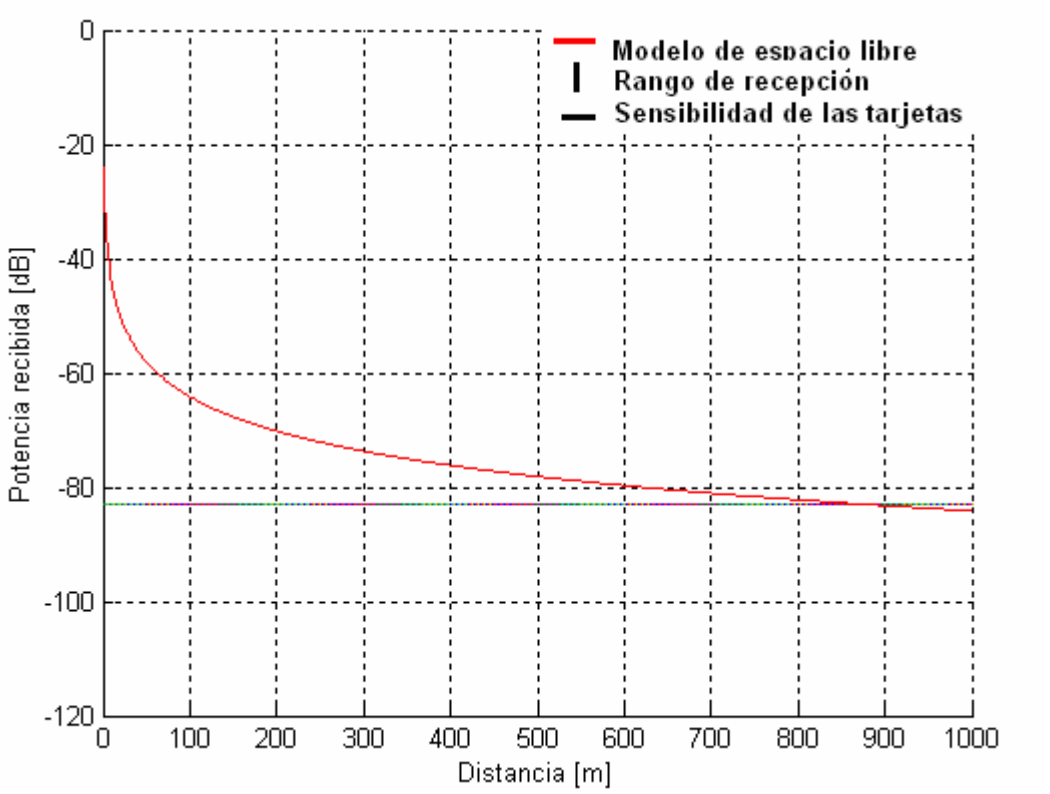
Esta ecuación también puede ser representada de manera logarítmica al trabajar todos los valores de la ecuación 1 en decibeles obteniéndose la siguiente expresión:

$$\Pr(d)[dB] = Pt + Gt + Gr + 32.4 + 20\log(d) + 20\log(f) \qquad (2)$$

Donde *Pt* es la potencia de la señal transmitida, *Gt* y *Gr* son las ganancias de antena del transmisor y receptor respectivamente y f es la frecuencia.

El modelo de espacio libre básicamente representa el rango de comunicación como un círculo alrededor del transmisor. En la figura 6 se representa la ecuación 2 para los valores Pt=15dBm, Gt=1dB y Gr=0dB; si un receptor está dentro del rango delimitado por la línea negra vertical recibe todos los paquetes, de lo contrario los pierde todos.

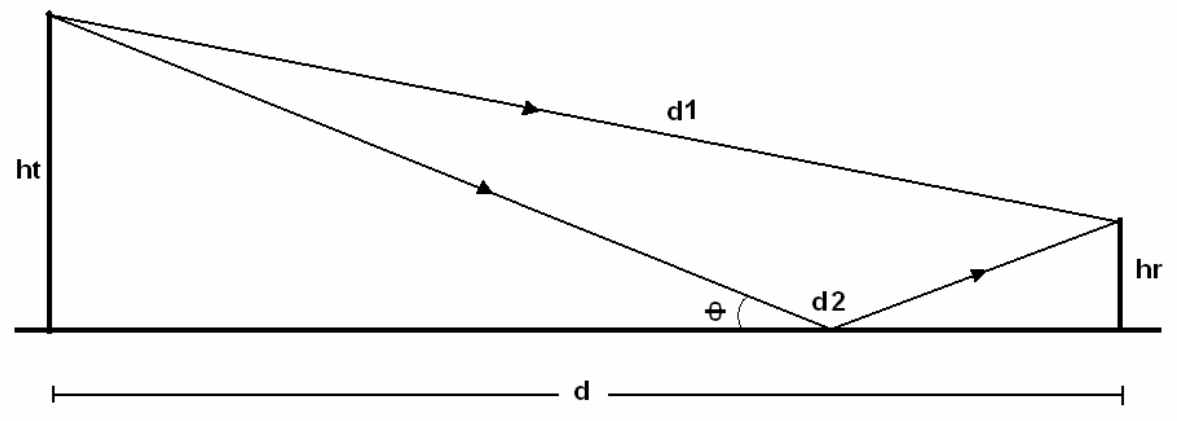
Figura 6. Modelo de propagación en espacio libre.



Modelo de reflexión de dos rayos a tierra [15]. Un simple camino de línea de vista entre dos nodos es raramente el único camino de propagación. El modelo de reflexión de dos rayos a tierra considera el camino directo "d1" y el

camino de reflexión a tierra "d2" de acuerdo con lo que se muestra en la figura 7.

Figura 7. Esquema de propagación de dos rayos a tierra.



Fuente, autores.

Teniendo en cuenta la propagación de los dos rayos por separado y sumando sus amplitudes vectorialmente, al final se obtiene el valor de resultante expresado en la ecuación 3.

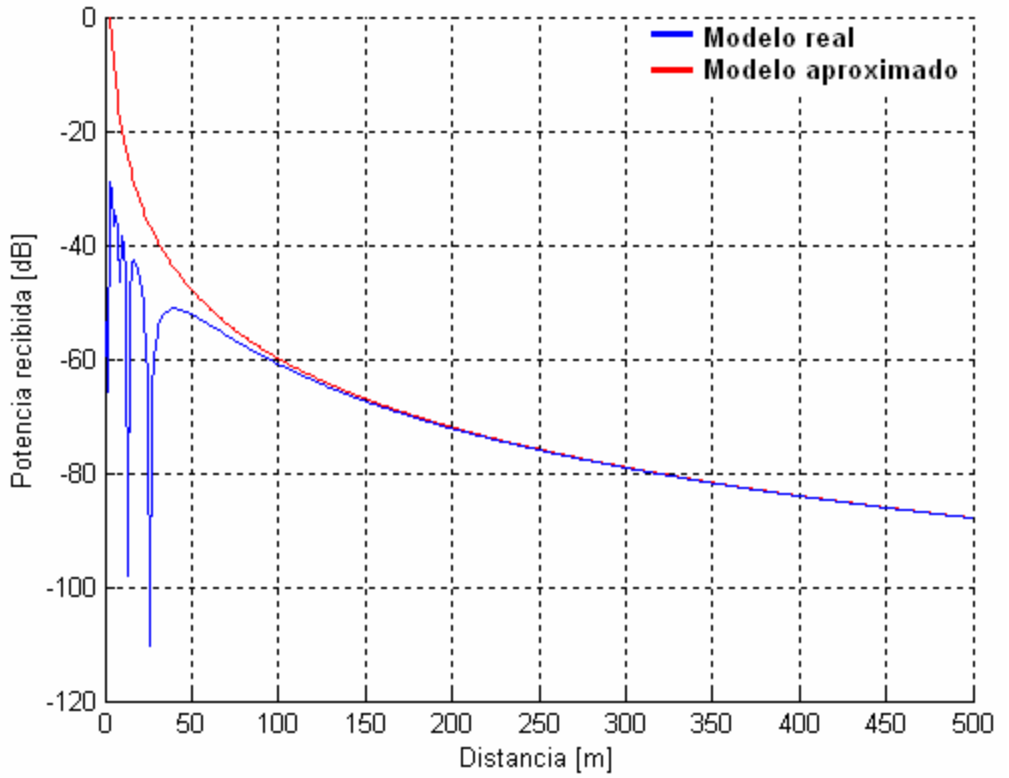
$$\Pr(d)[dB] = Pt + Gt + Gr + 10\log\left[2\left(\frac{\lambda}{4\pi d}\right)^2\left[1-\cos\left(k\frac{2hthr}{d}\right)\right]\right] \quad (3)$$

La anterior ecuación representa con más exactitud las predicciones de potencia a larga distancia pero es una expresión de difícil manejo para cálculos rápidos. Además no ofrece buenos resultados para cortas distancias debido a la oscilación causada por la combinación constructiva y destructiva de los rayos como se observa en la gráfica azul de la figura 8.

Así pues teniendo en cuenta que para largas distancias d>>ht,hr y que para pequeños ángulos cosθ ≈1-θ²/2 esta ecuación se puede simplificar:

$$\Pr(d)[dB] = Pt + Gt + Gr + 10\log\left[\frac{ht^2 hr^2}{d^4}\right] \quad (4)$$

Figura 8. Modelos de propagación de dos rayos a tierra.



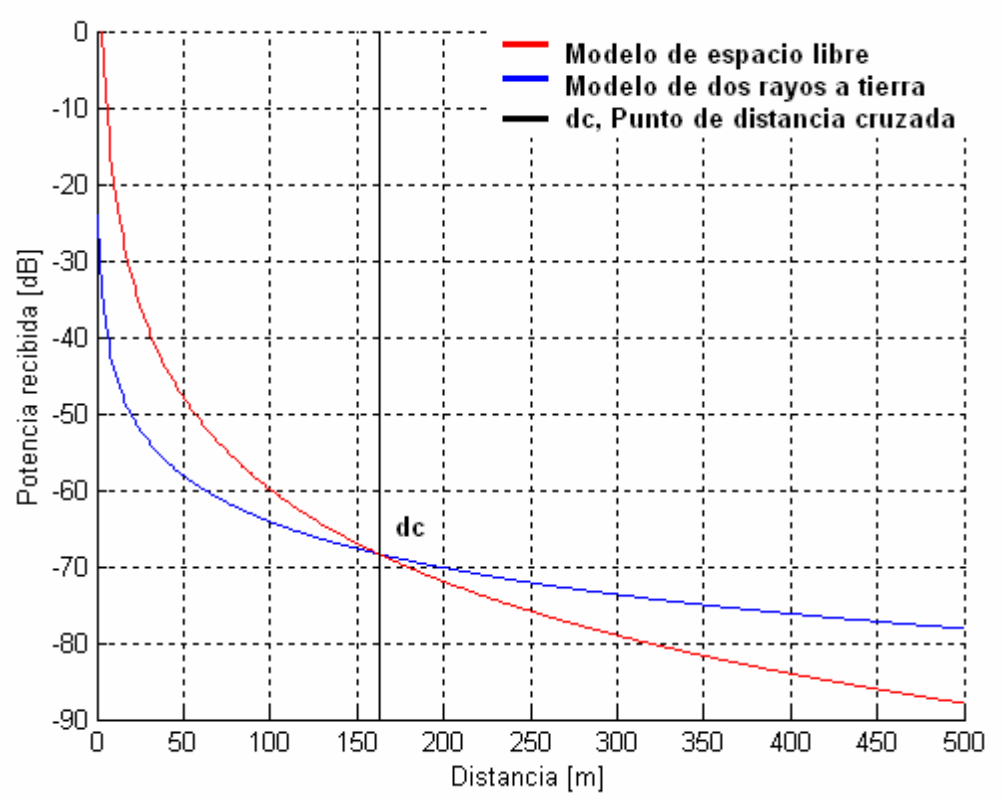
La ecuación 4 es una buena aproximación del modelo de radiopropagación para largas distancias ya que como se observa en la gráfica roja de la figura 8 después de cierta distancia se comporta de manera muy similar al modelo real representado por la gráfica azul de la misma figura.

Para cortas distancias se sigue utilizando el modelo de espacio libre. Por consiguiente, una distancia cruzada (Cross-Over) *dc* es calculada en este modelo. Cuando *d*<*dc*, la ecuación del modelo de espacio libre es utilizada. Cuando *d*>*dc*, la ecuación del modelo de reflexión de dos rayos a tierra es utilizada. En la distancia cruzada, las ecuaciones para espacio libre y reflexión

de dos rayos a tierra dan el mismo resultado. La distancia cruzada puede calcularse por:

$$dc = \frac{4 * \pi * ht * hr}{\lambda} \qquad (5)$$

Figura 9. Distancia cruzada entre los modelos de espacio libre y dos rayos a tierra.



Este modelo de radiopropagación también considera la comunicación como un círculo alrededor del transmisor, para lo cual si el receptor está dentro no presentará pérdidas mientras que si está fuera se perderán todos los paquetes transmitidos.

Modelo de sombra [15]. Los modelos de espacio libre y dos rayos a tierra predicen la potencia recibida como una función determinística de la distancia, Ambas representan el rango de comunicación como un círculo ideal. En realidad la potencia recibida acierta distancia es una variable aleatoria debido a los efectos de propagación de múltiples caminos, los cuales son conocidos

como efectos de desvanecimiento; un modelo más general y ampliamente utilizado es el modelo de sombra.

El modelo de sombra consta de dos partes, la primera es conocida como el modelo de pérdidas por distancia el cual predice la potencia promedio recibida en la distancia *d*. Para hacer el modelado primero que todo *Pr(do)* es calculada de la ecuación 1 de espacio libre:

$$\Pr(do) = \frac{Pt * Gt * Gr * \lambda\text{^}2}{(4 * \pi)\text{^}2 * do\text{^}2 * L} \qquad (6)$$

Con el logaritmo de la división entre la ecuación 1 y la ecuación 7 se obtienen las pérdidas por distancia medidas en dB en función de "β"; por lo tanto se obtiene:

$$\left[\frac{P'r(d)}{\Pr(do)}\right] dB = -10 * \beta * \log\left(\frac{d}{do}\right) \qquad (7)$$

Donde β es llamado el exponente de pérdidas por distancia (Path Loss Exponent), y es empíricamente determinado por medidas en campo. Valores grandes corresponden a más obstrucciones y por consiguiente un rápido decrecimiento en la potencia recibida promedio cuando la distancia comienza a aumentar.

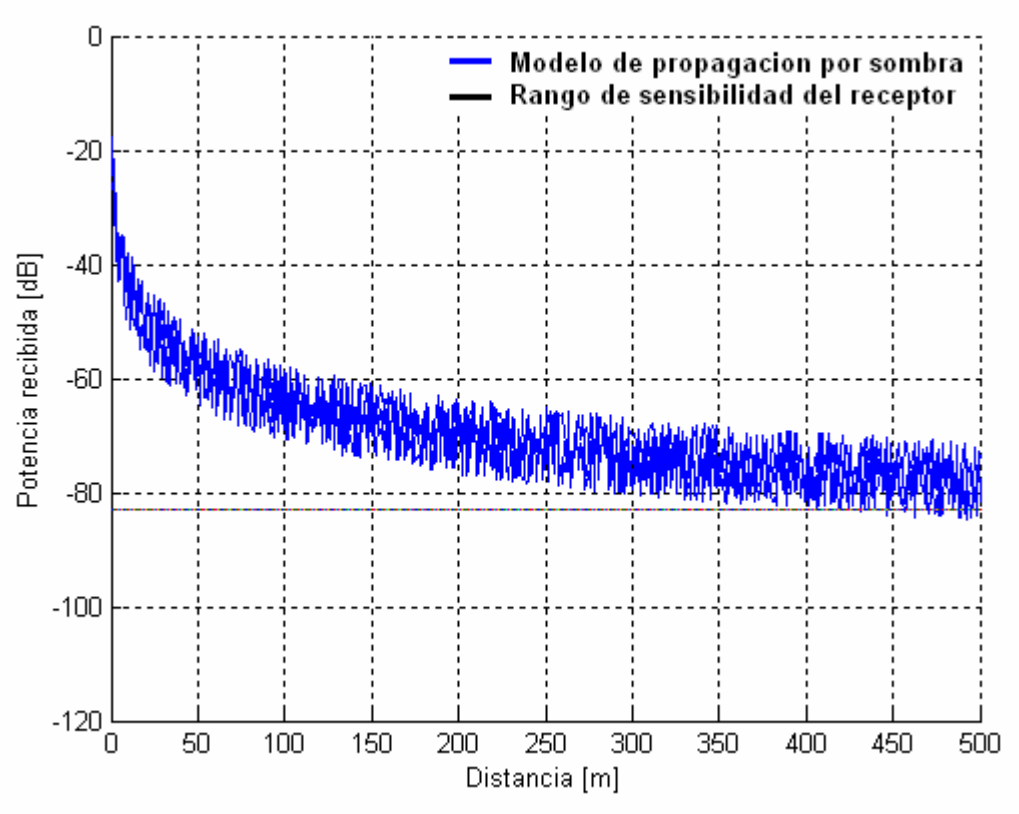
La segunda parte del modelo de sombra refleja la variación de la potencia recibida en cierta distancia la cual es una variable aleatoria de distribución gausiana. El modelo de sombra en conjunto es descrito la siguiente ecuación:

$$\left[\frac{\Pr(d)}{\Pr(do)}\right] dB = -10 * \beta * \log\left(\frac{d}{do}\right) + XdB \qquad (8)$$

Donde X_{dB} es una variable aleatoria gaussiana con media cero y desviación estándar σ_{dB}, la cual es llamada la desviación por sombra y es obtenida

también por medidas en el campo. La figura 10 representa el modelo de propagación de Shadowing con un β=2 y σ=4 de acuerdo a los datos aportados por las tablas 4 y 5 para el espacio libre.

Figura 10. Modelo de propagación por sombra



El modelo de sombra extiende el modelo de círculo ideal a un modelo estadístico, ya que los nodos pueden probablemente comunicarse cuando estén cerca del borde del rango de comunicación. Las tablas siguientes ofrecen ciertos valores para los parámetros β y σ_{dB}:

Tabla 4. Valores típicos del exponente de pérdidas por distancia β.

AMBIENTE		**β**
EXTERIORES	ESPACIO LIBRE	2
	ÁREA URBANA SOMBREADA	2.7 a 5
INTERIORES	LÍNEA DE VISTA	1.6 a 1.8
	OBSTRUIDO	4 a 6

Tomado de *NS MANUAL.*

Tabla 5. Valores típicos de la desviación por sombra σ (dB).

AMBIENTE	**σ(dB)**
EXTERIORES	4 a 12
OFICINA, PARTICIONES FUERTES	7
OFICINA, PARTICIONES SUAVES	9.6
INTERIORES, LINEA DE VISTA	3 a 6
INTERIORES, OBSTRUIDO	6.8

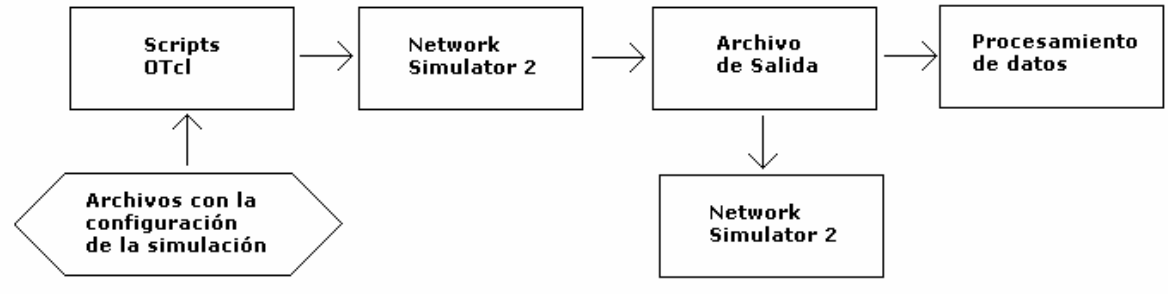
Tomado de *NS MANUAL.*

2. SIMULACIÓN CON NETWORK SIMULATOR

Para realizar una simulación de red inalámbrica en *NS*[9] se deben tener en cuenta aspectos como la definición del escenario, topología de red, pila de protocolos y parámetros específicos de cada protocolo, caracterización de antenas, tipo de modelo de radio propagación, número de nodos y posiciones dentro del escenario, además especificar si tendrán movimiento aleatorio o estarán estáticos, definición de agentes y protocolos[10].

La figura (11) y figura (12) ilustran en forma secuencial la estructura del proceso de simulación y los pasos a seguir en la realización de una simulación inalámbrica en *NS*.

Figura 11. Estructura del proceso de simulación en *NS*.



Fuente, autores.

Como se puede observar, para realizar una simulación en *NS* se debe contar con el script[11] OTcl previamente creado, éste es programado en modo texto y guardado con la extensión ".tcl". Posteriormente se simula con *NS* del cual se

[9] Network Simulator. Para una descripción detalla acerca de este software de simulación ver el anexo A.
[10] Para obtener una detallada lista de comandos disponibles en *NS*, revisar el anexo B
[11] Un script es un archivo de texto que contiene una serie de comandos, parámetros, y expresiones requeridas para realizar una programación.

obtienen los archivos de salida, consistentes en animaciones en *NAM*[12] y archivos de salida con extensión ".tr" que contienen los datos que se quiere sean procesados por el simulador.

2.1 PASOS A SEGUIR EN EL DESARROLLO DE UN SCRIPT DE SIMULACIÓN INALÁMBRICO

2.1.1 Definición de opciones y configuraciones. Cuando se desarrolla un script de simulación, si se desean realizar comentarios acerca de comandos en el programa de modo que no afecten el funcionamiento del script, se utiliza el símbolo de número "#" al inicio de cada línea[13], por ejemplo:

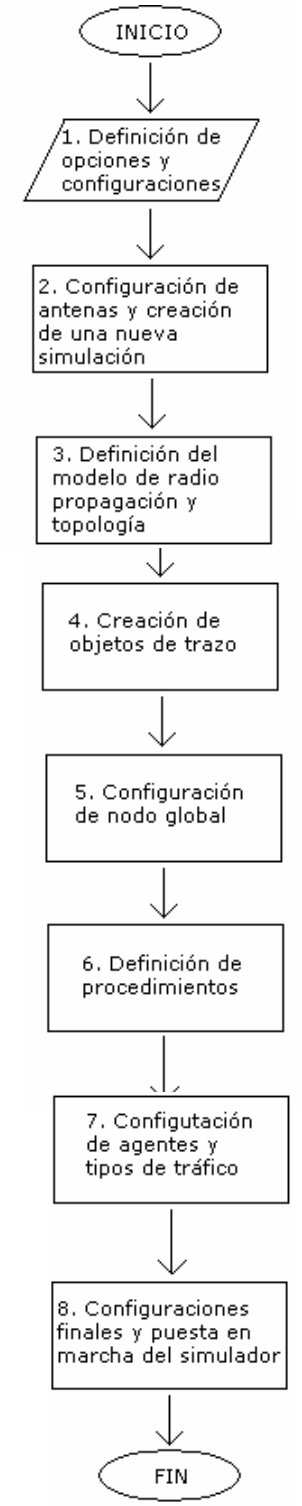
```
# Texto que se desea incluir dentro del script.
```

Seguido de esto se debe establecer la definición de opciones de simulación inalámbrica[14], las cuales se llaman posteriormente como se muestra a continuación:

```
set opt(chan)   Channel/WirelessChannel; # INMODIFICABLE
set opt(netif)  Phy/WirelessPhy        ; # INMODIFICABLE
set opt(mac)    Mac/802_11             ; # INMODIFICABLE
set opt(ifq)    Queue/DropTail/TIPO DE COLA
set opt(ll)          LL
set opt(ant)    Antenna/TIPO DE ANTENA
set opt(adhocRouting)   PROTOCOLO DE ENRUTAMIENTO ADHOC
set opt(x)      X ;# X DIMENSION DE LA TOPOGRAFIA
```

[12] Network Animator, permite visualizar la simulación en modo gráfico.
[13] Al inicio de cada simulación inalámbrica se aconseja explicar en un detallado texto lo que realiza el script.
[14] Algunos de los valores presentados a continuación se dejan establecidos en cierto valor ya que son inmodificables al trabajar con redes inalámbricas que son compatibles con el estándar 802.11b.

Figura 12. Esquema secuencial para simulación en *NS*.



Fuente, autores.

```
set opt(y)      Y ;# Y DIMENSION DE LA TOPOGRAFIA
set opt(ifqlen) Q ;# TAMAÑO MAXIMO DE PAQUETE EN LA INTERFAZ DE
# COLA
set opt(seed)   0.0
set opt(tr)     NOMBRE DE ARCHIVO.tr   ;# ARCHIVO DE TRAZO
set opt(nam)    NOMBRE DE ARCHIVO.nam  ;# ARCHIVO DE TRAZO NAM
set opt(nn)     N              ;# NUMERO DE NODOS SIMULADOS
set opt(stop)   T             ;# TIEMPO DE SIMULACION
```

Donde se configuran valores como el tipo de canal, capa física, subcapa MAC (control de acceso al medio) y enlace lógico, tipo de cola, antena y protocolo de enrutamiento *adhoc*, dimensiones de la topografía, tamaño máximo de paquete de cola, archivos de salida de trazo y *NAM*, número de nodos y tiempo de simulación. Estos valores se seleccionan de la tabla de opciones definida en la tabla 6.

Opcionalmente se pueden establecer las siguientes configuraciones por defecto, ya que no afectan el resultado de la simulación, esto se realiza con el fin de hacer estudios de análisis de puertos:

```
LL set mindelay_           TIEMPO EN SEGUNDOS
LL set delay_              TIEMPO EN SEGUNDOS
LL set bandwidth_          ANCHO DE BANDA EN Mbps

Agent/CBR set sport_       NÚMERO DE PUERTO
Agent/CBR set dport_       NÚMERO DE PUERTO

Agent/LossMonitor set sport_     NÚMERO DE PUERTO
Agent/LossMonitor set dport_     NÚMERO DE PUERTO

Agent/UDP set sport_       NÚMERO DE PUERTO
Agent/UDP set dport_       NÚMERO DE PUERTO
```

Tabla 6. Opciones disponibles para configuración de nodo.

OPCIÓN	VALORES DISPONIBLES	VALOR POR DEFECTO
GENERAL		
TIPO DE DIRECCIONAMIENTO	PLANO, JERARQUICO	PLANO
ORIENTADO A SATÉLITE E INALÁMBRICO		
ENRUTAMIENTO CABLEADO	ON, OFF	OFF
TIPO DE ENLACE LÓGICO	LL, LL/SAT	OFF
TIPO DE MAC	MAC/802.11,MAC/CSMA/CA, MAC/SAT, MAC/SAC/UNSLOTTEDALOHA, MAC/TDMA	OFF
TIPO DE INTERFAZ DE COLA	QUEUE/DROPTAIL, QUEUE/DROPTAIL/PRIQUEUE	OFF
TIPO FÍSICO	PHY/WIRELESSPHY, PHY/SAT	OFF
ORIENTADO A INALÁMBRICO		
ENRUTAMIENTO AD-HOC	DIFFUSION/RATE, DIFFUSION/PROB, DSDV, DSR, FLOODING, OMNIMCAST, AODV, TORA	OFF
TIPO DE PROPAGACIÓN	PROPAGATION/TWORAYGROUND, PROPAGATION/SHADOWING	OFF
EJEMPLAR DE PROPAGACIÓN	PROPAGATION/TWORAYGROUND, PROPAGATION/SHADOWING	OFF
TIPO DE ANTENA	ANNTENA/OMNIANNTENA	OFF
CANAL	CHANNEL/WIRELESSCHANNEL, CHANNEL/SAT	OFF
EJEMPLAR DE TOPOLOGÍA	<ARCHIVO DE TOPOLOGÍA>	OFF
IP MÓVIL	ON, OFF	OFF
MODELO DE ENERGÍA	ENERGYMODEL	OFF
ENERGÍA INICIAL	<VALOR EN JOULES>	OFF
POTENCIA DE RX	<VALOR EN W>	OFF
POTENCIA DE TR	<VALOR EN W>	OFF
POTENCIA DE ESTADO DESOCUPADO	<VALOR EN W>	OFF
TRAZO DE AGENTE	ON, OFF	OFF
TRAZO DE ENRUTAMIENTO	ON, OFF	OFF
TRAZO MAC	ON, OFF	OFF
TRAZO DE MOVIMIENTO	ON, OFF	OFF
ERRPROC	UNIFORMERRORPROC	OFF
TORADEBUG	ON, OFF	OFF
ORIENTADO A SATÉLITE		
TIPO DE NODO SATELITAL	POLAR, GEO, TERMINAL, GEO-REPEATER	OFF
ANCHO DE BANDA DE ENLACE DE BAJADA	<VALOR DE ANCHO DE BANDA>	OFF

Tomado de NS MANUAL.

Estos ajustan los valores de retardos de la subcapa de enlace lógico, puertos origen y destino de los agentes y generadores de tráfico.

El siguiente comando debe ser escrito seguido y obligatoriamente para la configuración de colas y protocolo de enrutamiento:

```
Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1
```

A continuación se deben definir las posiciones de las antenas sobre los nodos en metros y las ganancias de transmisión y recepción de las mismas:

```
Antenna/OmniAntenna set X_ X
Antenna/OmniAntenna set Y_ Y
Antenna/OmniAntenna set Z_ Z
Antenna/OmniAntenna set Gt_ Gt
Antenna/OmniAntenna set Gr_ Gt
```

2.1.2 Configuración de antenas y creación de una nueva simulación. En este paso se configuran las antenas con los parámetros ofrecidos por los fabricantes, se establecen de modo que los valores de potencias y umbrales se ajusten a las tarjetas. Los ajustes realizados son la potencia de transmisión *Pt* en vatios, ancho de banda, el cual es en este caso 11Mbps para *IEEE 802.11b*, frecuencia de transmisión del canal inalámbrico utilizado, en este caso 2.473 GHz la cual es la utilizada por el canal 6, finalmente el umbral de recepción en vatios.

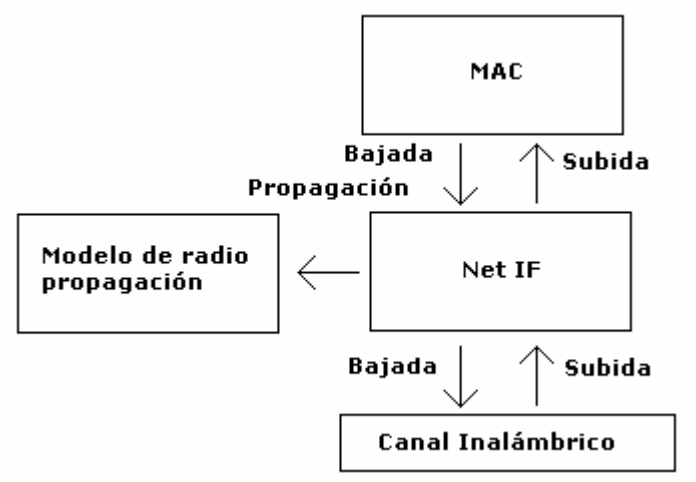
Para la potencia de transmisión, se trabaja con valores nominales en dBm, por lo tanto esta potencia en vatios es calculada por medio de la ecuación:

$$PdBm = 10 * \log\left(\frac{Pw}{10\text{^}(-3)}\right) \quad (1)$$

$$Pw = 10\text{^}(-3) * 10\text{^}\left(\frac{\text{PdBm}}{10}\right) \quad (2)$$

La figura 13 ilustra la estructura de las capas bajas de un nodo móvil en *NS*. Para un nodo móvil, la interfaz de red (NetIF) es un objeto del tipo *WirelessPhy* el cual es conectado al canal inalámbrico a través del apuntador de bajada. El objeto *WirelessPhy* tiene un enlace al modelo de radiopropagación el cual calcula la potencia recibida.

Figura 13. Esquema de un nodo móvil (capas bajas en *NS*).



Fuente, autores.

Para la transmisión de paquetes en *NS*, el canal inalámbrico tiene una lista de objetos *WirelessPhy* que están conectados a este. Cuando un nodo envía un paquete su objeto *WirelessPhy* inserta la potencia de transmisión (*Pt*) dentro del paquete y lo envía al canal inalámbrico.

Una vez recibido el paquete, el canal inalámbrico determina el retardo de propagación, la distancia entre los nodos trasmisor y receptor es dividida entre 300.000 km/s para determinar el retardo de propagación al receptor.

Cuando el paquete es recibido desde el canal inalámbrico, el objeto *WirelessPhy* informa al modelo de propagación para que éste calcule el nivel de potencia recibida de este paquete basado en parámetros tales como la distancia, potencia de transmisión, ganancia de antena transmisora y receptora, etc.

El modelo de propagación utilizado es *Sombra* o *Shadowing* ya que es el modelo más acercado a la realidad de los soportados[15] por *NS*. Teniendo el nivel de potencia recibida de cada paquete el objeto *WirelessPhy* debe decidir cómo reaccionar.

El objeto *WirelessPhy* tiene dos umbrales: El umbral de potencia recibida (*RXthresh*) y el umbral de detección de portadora (*CSThresh*). Si la potencia de recepción *Pr* es mayor que *RXthresh* el objeto *WirelessPhy* envía el paquete a su capa superior, de otro lado, si *Pr* es menor que *RXthresh* y mayor que *CSthresh*, significa que el objeto *WirelessPhy* puede detectar señal pero no puede decodificarla correctamente, lo que se reporta como un error para la capa superior. Finalmente si *Pr* es menor que *CSthresh* significa que no se puede detectar señal y el paquete es descartado.

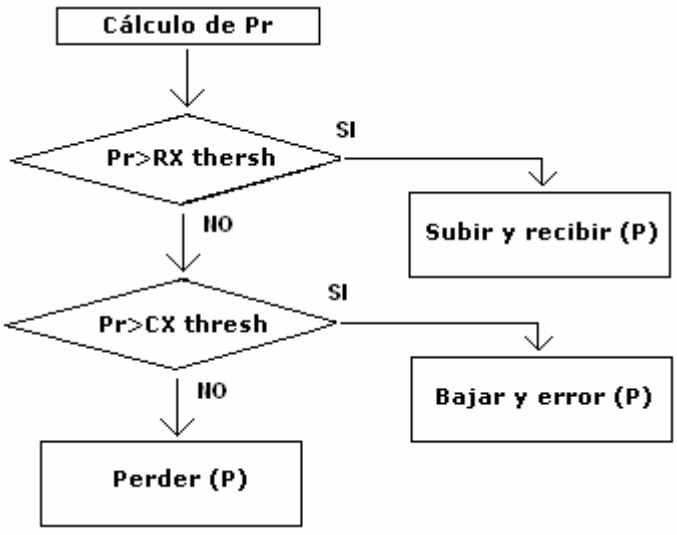
Para determinar el umbral de recepción de portadora *CSthresh*, se tiene que las antenas trabajan un nivel de sensibilidad del receptor en dBm para la tasa de 11 Mbps[16], por lo tanto, para calcular esta potencia en vatios se utiliza la siguiente ecuación:

$$Pw = 10\text{^}(-3) * 10\text{^}\left(\frac{\text{PdBm}}{10}\right) \quad (3)$$

[15] Entre otros modelos de radio propagación se encuentran el de espacio libre y dos rayos a tierra, los cuales se describen en el capítulo 1.

[16] Para el presente proyecto se tiene que los niveles de potencia medidos en los escenarios, incluyendo la toma realizada en el escenario de espacio libre, se encuentran dentro del rango para el cual las tarjetas inalámbricas trabajan a 11Mpbs. La descripción de los escenarios de estudio se presenta en el capítulo 3.

Figura 14. Procedimiento de la decisión WirelessPhy.



Fuente, autores.

Para el umbral de potencia recibida se modelan las tarjetas utilizando la misma ecuación presentada anteriormente.

Los comandos de configuración para las antenas inalámbricas se muestran a continuación:

```
Phy/WirelessPhy set Pt_ POTENCIA DE TRANSMISIÓN EN VATIOS
Phy/WirelessPhy set bandwidth_ 11e6    ; # INMODIFICABLE
Mac/802_11 set dataRate_ 11e6          ; # INMODIFICABLE
Mac/802_11 set basicRate_ 11e6         ; # INMODIFICABLE
Phy/WirelessPhy set freq_ FRECUENCIA DE NÚMERO DE CANAL
Phy/WirelessPhy set CSThresh_ UMBRAL DE DETECCIÓN DE PORTADORA EN
# VATIOS
# DISTANCIA DE REFERENCIA DEL MODELO DE RADIO PROPAGACIÓN
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set RXThresh_ UMBRAL DE RECEPCION EN VATIOS
```

Donde se ajustan valores como la potencia de transmisión de antena, ancho de banda de capa FÍSICA y MAC, estándar de subcapa MAC, frecuencia de transmisión y de canal inalámbrico, potencias de umbrales de recepción y detección de portadora inalámbrica.

Consecutivamente se debe crear una nueva simulación, que es obligatorio para cualquier tipo de simulación en ns:

```
set ns_          [new Simulator]
```

2.1.3 Definición del modelo de radio propagación y topología. Es necesario definir y ajustar el modelo de propagación de canal inalámbrico:

```
set prop   [new Propagation/Shadowing]
$prop set pathlossExp_ β
$prop set std_db_ σ en dBm
$prop set dist0_ do DISTANCIA DE REFERENCIA
$prop seed predef 0
```

Donde se configuran las variables como el modelo de radio propagación a utilizar. En este caso se trabaja con el modelo de sombra o shadowing y este a su vez exige que sean configurados parámetros como el exponente de pérdidas por distancia β, la desviación por sombra σ en dBm y la distancia de referencia *do* en metros.

EL paso siguiente es definir la topografía del terreno utilizada en la simulación, en este caso se configura para terreno plano [17] (ya que los escenarios analizados presentan esta característica) y se ajustan las dimensiones del escenario introducidas anteriormente en la definición de opciones.

[17] Se puede utilizar otra opción de tipo de terreno como por ejemplo escarpado descritas y definidas en ns/dem.cc, .h.

```
set wtopo  [new Topography]
$wtopo load_flatgrid $opt(x) $opt(y)
```

2.1.4 Creación de objetos de trazo. Se deben crear los objetos de trazo para obtener los archivos de salida en *NAM, XGRAPH* o de procesamiento de datos, se configuran para que sean archivos de escritura "w" y *NS* pueda escribir en ellos.

```
set OBJETO DE TRAZO   [open $opt(tr) w]
set OBJETO NAM    [open $opt(nam) w]
```

Los comandos que deben agregarse a continuación son necesarios para la correcta creación y adecuación de los archivos de salida *.nam* y *.tr* nombrados anteriormente, de lo contrario se presenta un mensaje de error al correr la simulación.

```
$ns_ trace-all $OBJETO DE TRAZO
$ns_ namtrace-all-wireless $OBJETO NAM $opt(x) $opt(y)
```

Continuando con el modelo de simulación, opcionalmente se puede utilizar el nuevo formato de archivos de trazo creado para *NS*:

```
$ns_ use-newtrace
```

2.1.5 Configuración de nodo global. Es necesario llamar el objeto *God*[18] junto con la configuración de nodo global, el cual contiene todas las configuraciones globales del escenario inalámbrico así como de los nodos móviles.

[18] "God" es el director general de operaciones (general operations director), y es usado con redes inalámbricas en *NS*-2.

En este punto del script se llaman todos los valores mencionados anteriormente en la definición de opciones y configuración de antenas, para establecerlos en los nodos y escenario de simulación, también se pueden activar las opciones de trazo de agente, MAC y de enrutamiento, simplemente cambiando su estado entre ON y OFF. Esto se realiza con el fin de visualizar cada paquete enviado y recibido en el visualizador NAM. Además de esto, el objeto *God* define cuántos nodos deben ser creados:

```
set god_ [create-god $opt(nn)]

$ns_ node-config  -adhocRouting $opt(adhocRouting) \
            -llType $opt(ll) \
            -macType $opt(mac) \
            -ifqType $opt(ifq) \
            -ifqLen $opt(ifqlen) \
            -antType $opt(ant) \
            -propInstance $prop \
            -phyType $opt(netif) \
            -channelType $opt(chan) \
            -topoInstance $wtopo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
```

Para continuar con la rutina de simulación se crean los nodos y se habilita o deshabilita su movimiento aleatorio, esto es, con un valor de 0 se lleva a cabo la deshabilitación y con un valor de 1 se habilita. Si se activa el movimiento aleatorio se deben especificar las posiciones iniciales y finales de cada nodo a medida que transcurre la simulación, además de definir las velocidades a las cuales se desplazan los nodos.

Luego se definen las posiciones de cada nodo en X, Y y Z. Cuando los nodos se encuentran estáticos, o sea, con movimiento aleatorio desactivado, el número de posiciones debe ser igual al número de nodos creados. Aquí, el número de cada nodo es creado automáticamente, gracias a la utilización del comando de repetición *for* descrito a continuación, pero igualmente se pueden configurar los nodos manualmente. Cabe anotar que para terreno plano la coordenada Z de cada nodo debe ser constante en el transcurso de la simulación si estos poseen movimiento aleatorio.

A continuación se presentan las líneas de comandos descritas anteriormente:

```
for {set i 0} {$i < $opt(nn) } {incr i} {
      set node_($i) [$ns_ node]
# $node_($i) random-motion 0 ;
# $node_($i) topography $wtopo
}

$node_(0) set X_ X
$node_(0) set Y_ Y
$node_(0) set Z_ Z
$node_(1) set X_ X
$node_(1) set Y_ Y
$node_(1) set Z_ Z
```

2.1.6 Definición de procedimientos. Se deben definir los procedimientos para guardar, graficar, realizar procesamiento y cálculo de datos respectivamente.

Un procedimiento para cerrar, editar y graficar los archivos de salida utilizando *NAM*, *XGRAPH* y un programa de edición de texto al final de la simulación, se muestra a continuación:

```
proc NOMBRE DEL PROCEDIMIENTO {} {
      global OBJETO DE TRAZO
      close $OBJETO DE TRAZO
      exec ./xgraph NOMBRE DEL ARCHIVO.tr -geometry RESOLUCUION &
      exec ./nam NOMBRE DEL ARCHIVO.nam &
       exit 0
}
```

El procedimiento expuesto a continuación es eficaz para almacenar datos en los archivos de trazo y gráfico según los cálculos matemáticos que se deseen realizar:

```
proc NOMBRE DEL PROCEDIMIENTO {} {
        global AGENTES OBJETOS DE TRAZO
        set ns [Simulator instance]
       OPERACIONES MATEMÁTICAS QUE SE DESEAN REALIZAR
}
```

2.1.7 Configuración de agentes y tipos de tráfico. Siguiendo el mecanismo de programación, se crean y configuran los agentes emisores y receptores además de los tipos de generación de tráfico.

Los agentes pueden ser del tipo TCP (protocolo de control de transmisión) o UDP (protocolo de datagrama de usuario). Los generadores de tráfico son la forma como los paquetes van a ser enviados, como ejemplos de esto se encuentra CBR (rata de bit constante), FTP (protocolo de transferencia de archivos) o tráfico TELNET (terminal virtual).

```
set AGENTE [new Agent/TIPO DE AGENTE]
$ns_ attach-agent $node_(X) $AGENTE

set TRÁFICO [new Application/Traffic/TIPO DE TRÁFICO]
```

```
$TRÁFICO attach-agent $AGENTE
$ns_ connect $AGENTE TRANSMISOR $AGENTE RECEPTOR
$ns_ at 2.0 "$TRÁFICO start"
$ns_ at 32.0 "$TRÁFICO stop"
```

2.1.8 Configuraciones finales y puesta en marcha del simulador. Finalmente se deben realizar algunas configuraciones y ajustes simples antes de ejecutar la simulación.

En este punto se define el tamaño de los nodos en el visualizador *NAM*, de modo que se pueden ajustar para que sean más o menos visibles en el escenario sin afectar los resultados finales.

```
for {set i 0} {$i < $opt(nn)} {incr i} {

    $ns_ initial_node_pos $node_($i) TAMAÑO DEL NODO EN NAM
}
```

Este comando se encarga de llamar y ejecutar los procedimientos que almacenan y grafican datos luego de unos segundos de simulación.

```
$ns_ at T "NOMBRE DEL PROCEDIMIENTO"
```

Siguiendo los pasos, se debe avisar a los nodos cuando termina la simulación de modo que estos obtengan las configuraciones iniciales.

```
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).000000001 "$node_($i) reset";
}
```

Se debe informar a *NAM* cuando debe parar la simulación gráfica y animada.

```
$ns_ at  $opt(stop)   "$ns_ nam-end-wireless $opt(stop)"
```

Como paso final se utiliza el comando *run* que da a *NS* la orden para que ejecute la simulación.

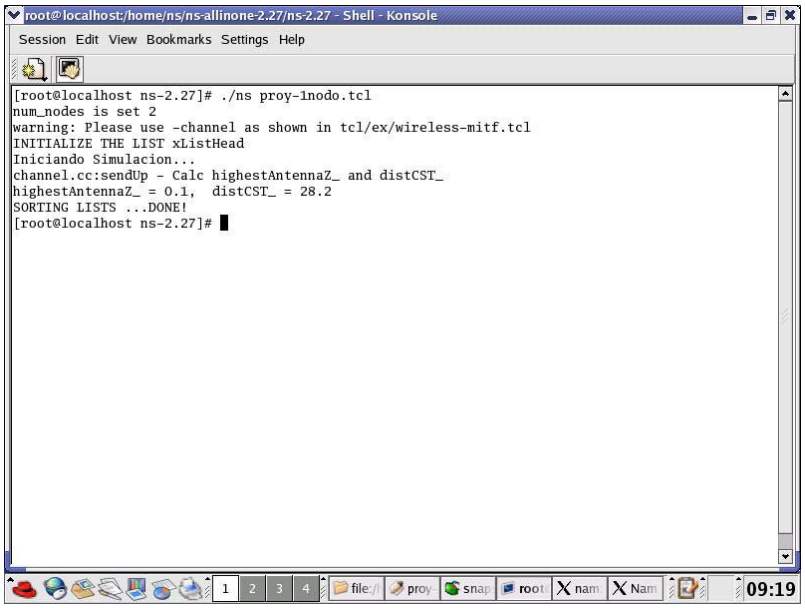
```
$ns_ run
```

Los pasos presentados anteriormente son la base de una simulación de red inalámbrica.

2.2 RESULTADOS GRÁFICOS.

A continuación se presentan resultados gráficos obtenidos luego que una simulación es realizada para uno de los escenarios de simulación[19].

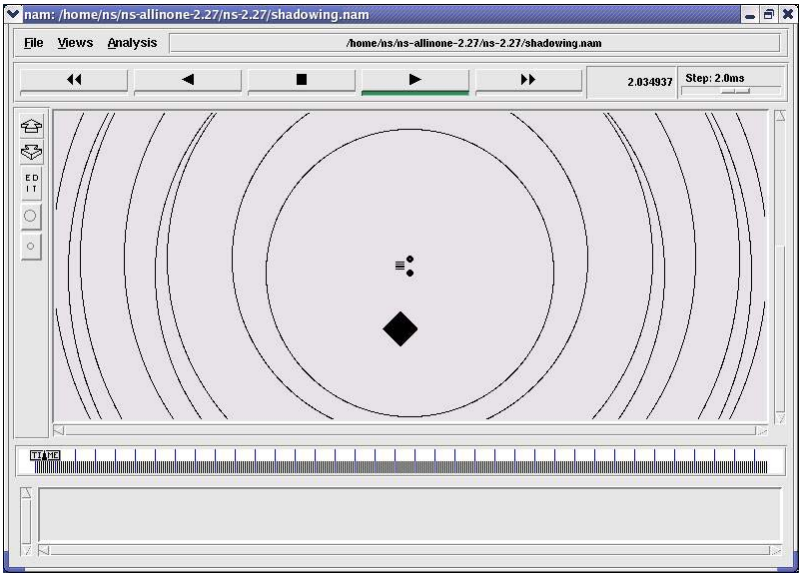
Figura 15. Resultado de la simulación en la ventana de terminal.



Tomado de *NS FUNCIONANDO EN RED HAT LINUX 9.0.*

[19] El script de simulación del cual se obtienen estos resultados se presenta en el anexo C.

Figura 16. Resultado de la simulación en la ventana de *NAM*.

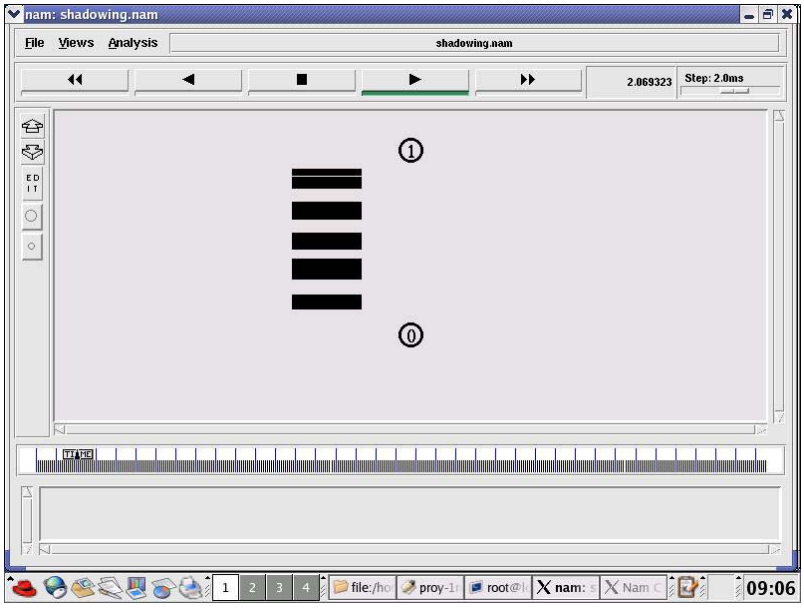


Tomado de *NS FUNCIONANDO EN RED HAT LINUX 9.0.*

En la figura 16, el nodo 0 corresponde al punto de acceso y el nodo 1 a la tarjeta de red inalámbrica receptora. Los círculos concéntricos observados alrededor de cada nodo representan señales de radio dinámicas para animar la simulación. El cuadrado negro corresponde a paquetes perdidos durante la transmisión.

La figura 17 muestra una ampliación de la gráfica anterior, donde el nodo 0 corresponde al punto de acceso y el nodo 1 a la tarjeta de red inalámbrica receptora. Los rectángulos negros que van del nodo 0 hacia el nodo 1 representan la animación dinámica de los paquetes enviados desde el punto de acceso.

Figura 17. Resultado ampliado de la simulación en la ventana de *NAM*.



Tomado de *NS FUNCIONANDO EN RED HAT LINUX 9.0.*

3. DESCRIPCIÓN DE LAS PRUEBAS

Mediante tomas de datos reales se pretende evaluar la calidad de los resultados de la simulación con una serie de pruebas programadas que pueden dar una idea de las diferencias existentes y sus posibles causas.

3.1 DESCRIPCIÓN DE HARDWARE Y SOFTAWARE

Equipo de Cómputo: Se utilizaron 8 equipos Optiplex GX 260 marca DELL Pentium IV de 2,4 GHz, 512 de memoria RAM y 30GB de Disco duro.

Punto de Acceso: El punto de acceso o *Access Point* utilizado en las pruebas fue, DLINK AIRPRO DWL – 2000AP.

Tarjetas Inalámbricas: Se utilizaron tres tarjetas diferentes:

- Orinoco USB Client silver.
- Dlink DWL – 120 USB Wireless.
- 3com 3CRSHEW696

Network Simulator: Se utilizó la versión para Linux NS-2.3.3.a4
MGEN: Se Utilizó la versión 3 del Multigenerador de Tráfico
RED HAT LINUX: se utilizó LINUX versión 9.0 con kernel 2.4.20-8
WINDOWS: la versión es WINDOWS XP PROFESSIONAL con el SP2
MATLAB: la versión MATLAB 6 R12
CLIENT MANAGER: Administrador de dispositivos inalámbricos para tarjeta USB Orinoco

3.2 CONSIDERACIONES GENERALES PARA MEDIR LA TASA DE THROUGHPUT

3.2.1 Throughput. Se define como la velocidad de recepción de paquetes en la cual no se obtienen pérdidas dentro de un rango específico de error. Existen muchas variables que afectan el desempeño de un enlace, pero en este caso se tendrá en cuenta la variabilidad de la medición de throughput respecto de la distancia, y la variabilidad con respecto al número de usuarios instantáneamente conectados a la red, con el fin de observar y evaluar los resultados presentados por la herramienta de simulación y así determinar el alcance y exactitud de las simulaciones en Network Simulator.

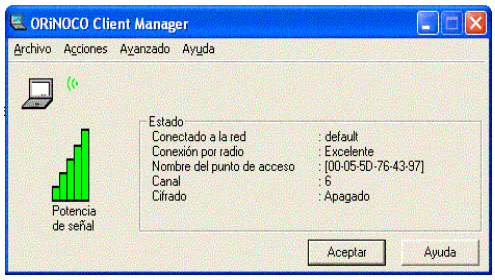
3.2.2 Potencia. La medición de potencia hace parte de la especificación del escenario ya que permite asegurar que éste se encuentra dentro del rango especificado (mayor a -83dBm) para la mayor tasa de transmisión (11Mbps), de acuerdo a datos obtenidos de las especificaciones de los fabricantes consignadas en el Anexo C.

Esta medición es necesaria ya que debido a que el simulador no tiene la capacidad integrada de hacer cambios de velocidad para cada nivel de sensibilidad entonces se hace importante garantizar que dentro de las condiciones del escenario se encuentra la de tener una distribución de potencia estable dentro del rango de sensibilidad de las tarjetas.

Normalmente la medida de la potencia viene especificada por la interfaz propia de instalación de cada tarjeta sobre todo en sistema operativo Windows; en vista que las tarjetas Orinoco serían las más utilizadas durante la pruebas se escogió el sistema de administración de esta tarjeta, llamado CLIENT MANAGER para hacer este tipo de medición bajo el sistema operativo Windows debido a que Linux presenta los resultados de las mediciones esperadas pero no es iterativo a menos que se repita el comando, mientras que en Windows la

interfaz aparte de ser amigable presenta los datos en tiempo real permitiendo así tener un idea más detallada de la medida tomada.

Figura 18. Interfaz de administración de la tarjeta Lucent/Orinoco



Fuente software Client Manager

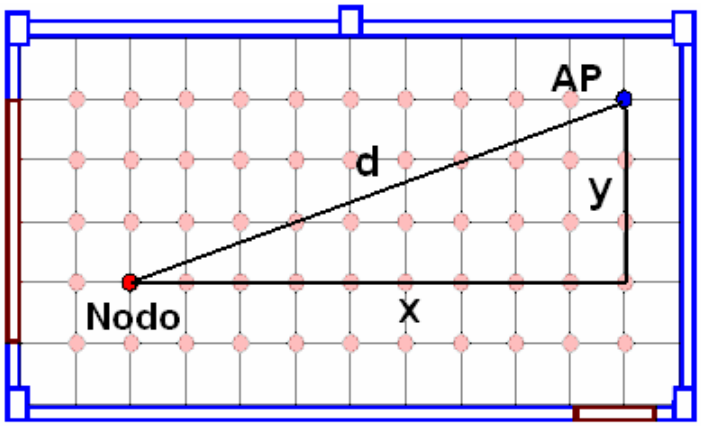
Figura 19. Interfaz de supervisión de la tarjeta Lucent/Orinoco

Fuente software Client Manager

Para las tarjetas D'link y 3com que solo participaban en las medidas multiusuarios se tomaron medidas de potencia mediante el comando "iwconfig" en una ventana "Terminal" de Linux, repitiendo varias veces el comando para obtener varios valores y sacar un promedio, debido a que la prueba de throughput fue hecha en Linux y resultó más conveniente obtener los datos de potencia sin tener que cambiar de sistema operativo. Utilizando las medidas de potencia, se buscan los coeficientes que mejor se adecúen a los datos obtenidos. Cada medida de nivel de potencia en el plano le corresponde una

distancia lineal "d", hasta el Punto de Acceso; teniendo esta distancia y la medida de potencia se genera una gráfica de potencia ya no en función de "X" y "Y" sino en función de "d" que es la distancia.

Figura 20. Transformación de las variables dependientes "X" y "Y" por "d".



Fuente, Autores.

Teniendo ya los datos en función de la distancia "d" como se muestra en la figura 20 para los puntos rojos, se hace una regresión hacia la curva descrita por la siguiente ecuación[20]:

$$\left[\frac{\Pr(d)}{\Pr(do)}\right]dB = -10 * \beta * \log\left(\frac{d}{do}\right) + X(o)dB$$

Esta es una regresión de tipo no lineal y por tanto es necesario hacer varias iteraciones en busca del mejor coeficiente "β". El coeficiente "σ" se obtiene de tomar los mismos datos utilizados para la regresión y hallar la desviación estándar respecto de la curva con el coeficiente "β" hallado. El procesado de los datos de los diferentes niveles de potencia a través del escenario da como resultado los coeficientes que mejor describen la tendencia general de los datos de acuerdo a su nivel de dispersión y atenuación; estos coeficientes son utilizados en las simulaciones con el modelo de radio propagación de Shadowing explicado en el capítulo uno.

[20] Ver sección 1.3.

3.2.3 Tipo y tamaño de paquete. El tipo de paquete es importante definirlo ya que dependiendo de la especificación del tráfico se tendrán unas mediciones más adecuadas del canal; en la capa de transporte del modelo OSI se encuentran los protocolos TCP que están orientados a conexión y del tipo confiable y el UDP que no es orientado a conexión y por lo tanto no se garantiza la llegada, careciendo de secuencias de asignación, control de flujo y acuses de llegada. George Xylomenos y Goerge C. Polyzos, en su documento "TCP and UDP Performance over a Wireless LAN" [28] observan que la medición de tráfico con paquetes UDP es más confiable que la medición bajo el protocolo TCP ya que UDP no necesita acuses de llegada y por lo tanto no depende de la velocidad de procesamiento del ente receptor, además que no presenta desbordamientos de buffer ni esperas de acuses. Así pues para el trabajo realizado en este proyecto se define como tipo de paquete el UDP.

El tamaño es muy importante también a la hora de obtener una medición de throughput debido a que tiene una influencia directa sobre el nivel de tráfico en la red dependiendo del estado físico del canal. Un paquete grande contiene menos encabezados y más datos que una serie de paquetes pequeños mejorando así la tasa de transmisión, pero tiene el problema que tarda mas tiempo en ser transmitido y por lo tanto tarda más en liberar el canal, además si el canal es ruidoso puede hacer que se pierda un paquete con una gran cantidad de información.

En el proyecto realizado por ALVAREZ, J et. al [2] se definió que el mayor tamaño de paquete con el que trabaja un enlace inalámbrico es de 1472 bytes produciendo una mínima cantidad de encabezados y por lo tanto un mayor desempeño; entonces para dar continuación a este estudio y por consideraciones de tráfico se definirá para efectos de este proyecto un tamaño de paquete de la misma dimensión.

3.2.4 Tráfico sobre la red. El programa emisor (ver Anexo C) que ejecuta MGEN controla el envío generando una serie de paquetes a diferentes tasas de velocidad, consignando en un archivo de texto los datos necesarios para este proyecto, tales como paquetes recibidos, hora de la prueba, tasa de envío y la velocidad.

En la tabla siguiente se describen las tasas de envío y la cantidad de paquetes enviados durante el ciclo de 30 segundos. Se observa un decremento de 25 pps en la tasa de envio que corresponde al 5% de la máxima tasa que es 500 pps.

Tabla 7. Generación de paquetes desde el Punto de Acceso.

Tiempo (s)	Tasa de envío (pps)	Cantidad de paquetes
0-30	500	15000
40-70	475	14250
80-110	450	13500
120-150	425	12750
160-190	400	12000
200-230	375	11250
240-270	350	10500
280-310	325	9750
320-350	300	9000
360-390	275	8250
400-430	250	7500
440-470	225	6750
480-510	200	6000
520-550	175	5250
560-590	150	4500
600 630	125	3750
640-670	100	3000
680-710	75	2250
720-750	50	1500
760-790	25	750

Cada ciclo tiene una duración de 30 segundos y esta intermediado por un lapso de 10 segundos que son utilizados por el receptor para hacer algunos cálculos; en total son 20 envíos que duran 790 segundos.

El programa receptor ejecuta tanto el DREC[21] para escuchar los paquetes como el MCALC[22] para hacer algunos cálculos con los datos entregados por DREC; parte de los datos entregados son el número de paquetes recibidos, el retardo, la tasa de paquetes y datos recibidos, los paquetes perdidos y estadísticas de latencia y variación de latencia, pero de todos esos sólo es necesario para propósito de este proyecto el número de paquetes recibidos así que las demás variables se descartaron del archivo de salida.

El DREC funciona de la misma manera que el MGEN en ciclos de 30 segundos intermediados por 10 segundos entre ciclo y ciclo, estos 10 segundos son utilizados por el MCALC para hacer el conteo de paquetes y el respectivo archivo de salida. Tanto el programa de recepción como el programa de emisión deben estar sincronizados para que los ciclos coincidan y no se pierdan paquetes, por lo tanto se hizo necesario la sincronización con un reloj externo el cual fue activado a través de Internet por medio de **clock.redhat.com.**

3.3 METODOLOGÍA

Se escogieron tres escenarios para hacer las pruebas, en cada escenario se realizó una prueba en específico tal como se describe a continuación:

3.3.1 Prueba 1, determinación de la tasa de throughput en espacio libre con un usuario.

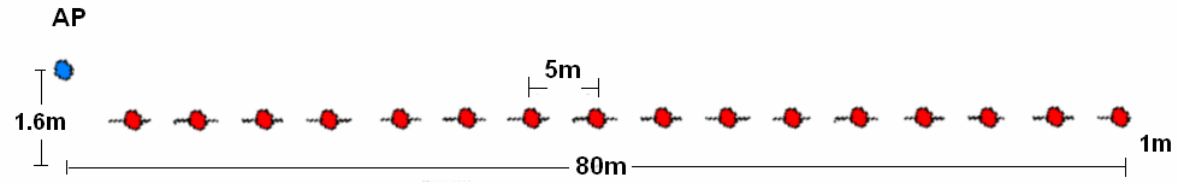
Esta prueba pretende medir la tasa de throughput alcanzada por el enlace inalámbrico en un espacio abierto.

[21] DREC: Programa de recepción de paquetes que se utiliza junto al generador MGEN.
[22] MCALC: Programa de calculo que utiliza DREC para el conteo de paquetes y cálculos internos necesarios.

De acuerdo con la siguiente figura el tráfico es enviado desde la estación generadora (punto azul) donde se encuentra el Punto de Acceso a una altura de 1.60m, hacia las tarjeta inalámbrica (punto rojo) ubicadas a una altura de 1m, alturas que se mantendrán durante todo el estudio.

Las mediciones son tomadas en una línea recta a partir de los 5m con una separación de 5m entre cada prueba tal como se indica en la figura siguiente:

Figura 21. Distribución de las pruebas en el plano.



Fuente, Autores.

Para este caso también son tomadas muestras de potencia en cada posición donde se mide el tráfico con el objeto de obtener los coeficientes característicos que describen el escenario. Las mediciones de potencia se hacen trasladando la tarjeta receptora de marca Lucent Orinoco mencionada en el Anexo D linealmente cada cinco metros, para esto no es necesario hacer emisiones de tráfico ya que el software de administración presenta estos resultados en pantalla sin necesidad de tener datos en el canal. La prueba se realizó bajo el sistema operativo Windows XP con el software de administración y monitoreo que posee la tarjeta.

Los datos son tomados manualmente y consignados en hojas de cálculo junto con la coordenada específica para el posterior análisis de la información. Estos resultados son muy importantes a la hora de implementar una simulación ya que permite asegurar que el receptor se encuentra dentro del rango de

sensibilidad de las tarjetas para la máxima transmisión de enlace de radio que es de 11Mbps; si este no fuese el caso los resultados simulados podrían no ser iguales a los medidos porque el simulador no presenta variaciones automáticas de velocidad.

Al final de esta prueba se obtendrá la distribución del nivel de throughput en función de la distancia para espacio libre junto con los coeficientes "σ" Y "β" mencionados en el capítulo uno que describen las características físicas del lugar.

3.3.2 Prueba 2, determinación de la tasa de throughput en interiores sin obstáculos para un usuario. En esta prueba se pretende medir la tasa de throughput en diferentes puntos dentro del escenario, enviando tráfico desde la estación generadora fija cableada en donde se encuentra el punto de acceso, hacia una estación móvil que posee la tarjeta inalámbrica.

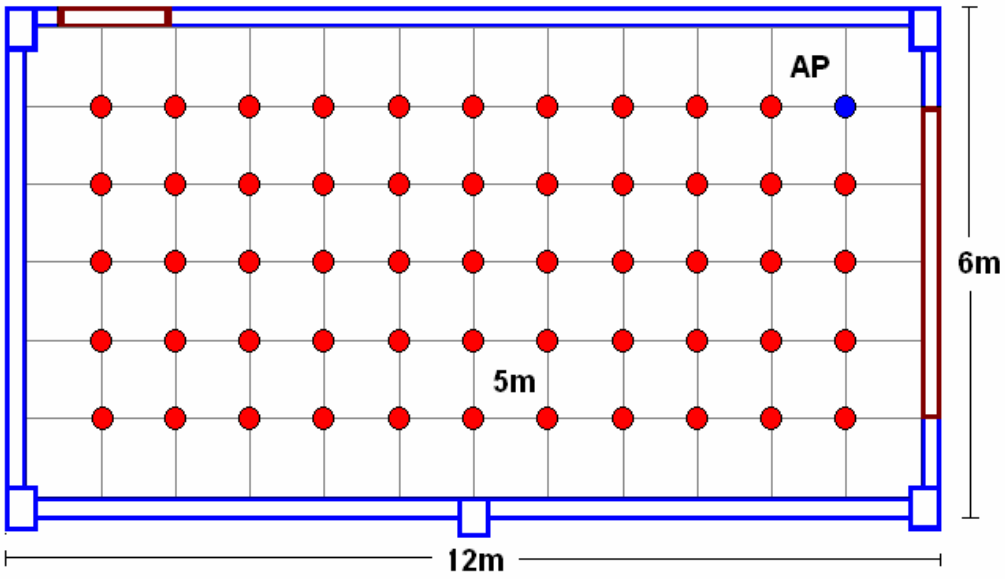
La tarjeta es movilizada por 55 posiciones a través del escenario de acuerdo a los puntos rojos de la figura 22 tomando cada una de las pruebas, para luego procesar los datos y hallar la respectiva medición de throughput.

Paralelamente se irán tomando medidas de potencia trasladando la tarjeta inalámbrica de la misma manera que se hizo para las mediciones de tráfico; para esto no es necesario hacer emisiones de tráfico ya que el software de administración presenta estos resultados en pantalla sin necesidad de tener datos en el canal. Los datos son tomados manualmente y consignados en hojas de cálculo junto con la coordenada específica para el posterior análisis de la información.

Al final de esta prueba se obtendrá la distribución del nivel de throughput en función de la posición para interiores sin obstáculos junto con los coeficientes

"σ" Y "β" mencionados en el capítulo uno que describen las características físicas del lugar.

Figura 22. Configuración de la prueba del Throughput.



Fuente, Autores.

3.3.3 Prueba 3, determinación de la tasa de throughput en interiores para varios usuarios. Esta prueba consiste en hacer variar el número de usuarios que ingresan a la red instantáneamente para observar la distribución de velocidad que presenta la red inalámbrica. Se toma un total de doce muestras por usuario distribuidos de acuerdo a la tabla 8; para las pruebas de un usuario se toman doce muestras, para dos usuarios se toman seis muestras, para las de tres usuarios se toman cuatro muestras y para las pruebas de cuatro usuarios o mas usuarios se toman tres muestras dando un número igual de muestras por cada usuario. Las ubicaciones se pueden observar en la figura 23.

Para esta prueba se utilizaron dos tarjetas Lucent Orinoco, tres tarjetas Dlink y tres tarjetas 3com. Las muestras son consignadas por el Script en un archivo de texto de salida junto con la hora de recepción, datos que luego son

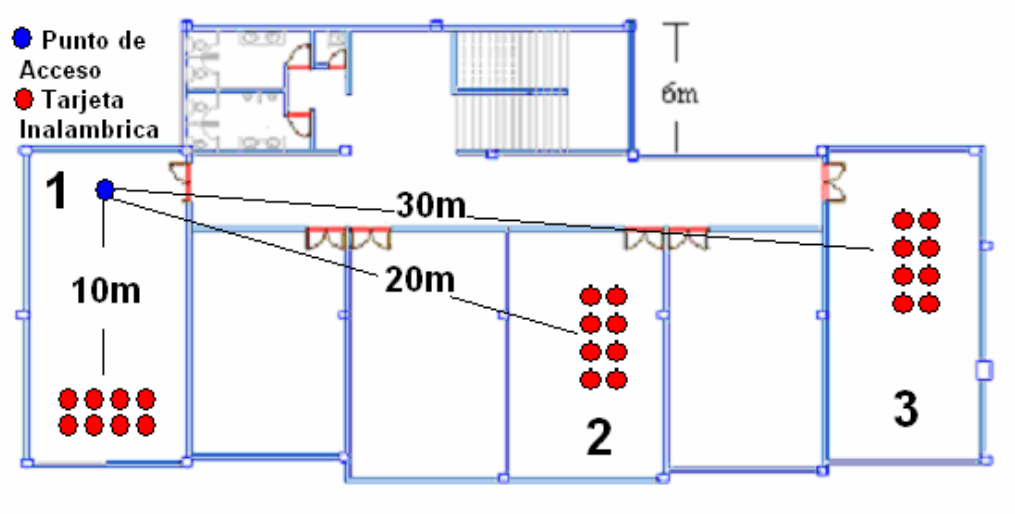
procesados por medio de un programa hecho en MATLAB (Anexo D) que halla el punto de intercepción entre la tasa de emisión y recepción con un porcentaje de error determinado[23], tal como se hizo para las anteriores pruebas de throughput. La siguiente tabla muestra la cantidad de lecturas tomadas para cada cantidad de usuarios en los respectivos escenarios.

Tabla 8. Distribución de pruebas de acuerdo a los escenarios.

Prueba	# Usr	Lecturas
SALA 1	1	12
	2	6
	3	4
	4	3
	5	3
	6	3
	7	3
	8	3
SALA 2	1	12
	2	6
	3	4
	4	3
	5	3
	6	3
	7	3
	8	3
SALA 3	1	12
	2	6
	3	4
	4	3
	5	3
	6	3
	7	3
	8	3

[23] Ver sección 4.1, Nivel de throughput y rangos de error.

Figura 23. Plano de distribución de las pruebas.

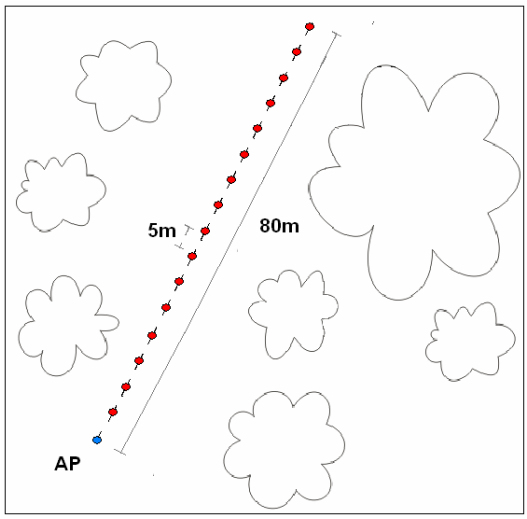


Fuente, Autores

3.4 DESCRIPCIÓN DE LOS ESCENARIOS.

Escenario 1. Este escenario es un espacio abierto en línea recta, el piso es de tierra húmeda cubierto por una fina capa de gramilla, en los laterales se encuentra una distribución de árboles con una distancia promedio de separación de 20m tal como se observa en la figura 24.

Figura 24. Dibujo esquemático del espacio libre utilizado.



Fuente, Autores.

La ubicación del lugar consiste en la parte trasera de los laboratorios de diseño industrial, en la zona arbórea comprendida hasta los linderos de la carretera que recorre la parte trasera de la universidad.

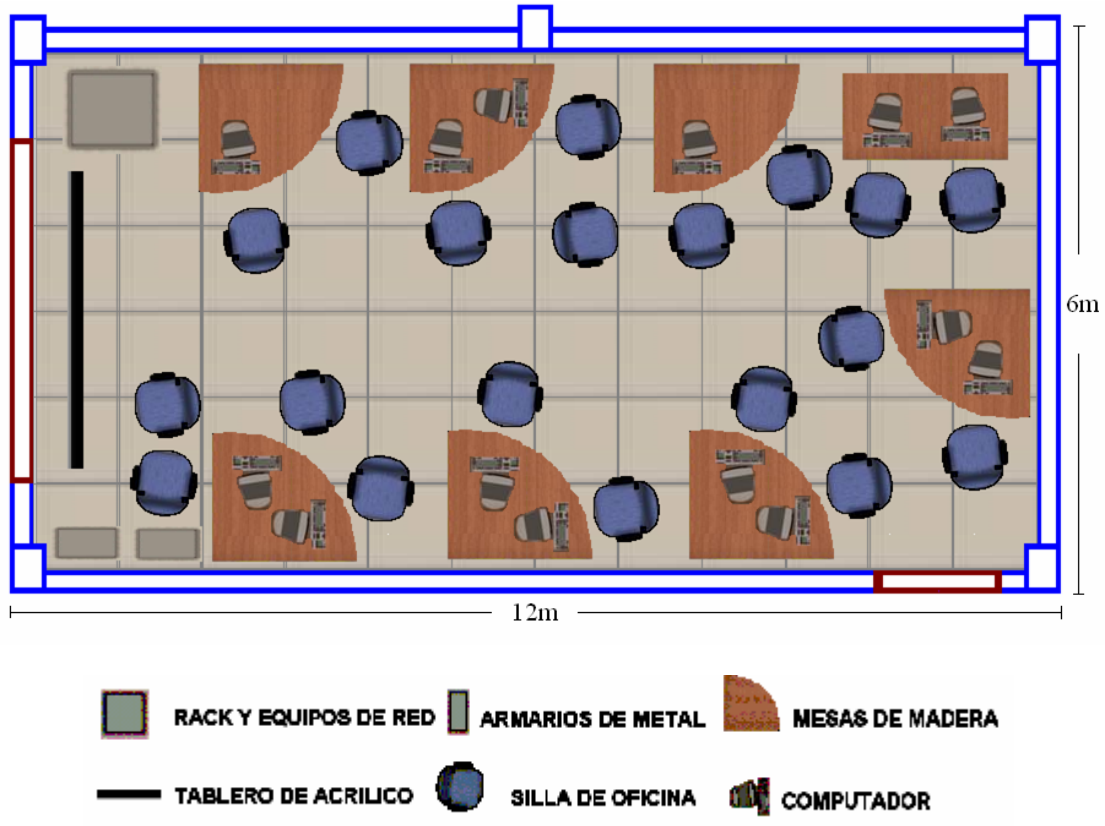
Escenario 2. El escenario se define como una locación de 12mX6m encerrada por paredes de ladrillo recubierto por cemento mezclado con arena, la cual es forrada por una fina capa de yeso pintado con acrílico; el techo se compone de un cielo raso de icopor a un altura de 2.7m, superior a este cielo raso se encuentra un techo de tejas sostenidas por vigas de hierro trenzado descubierto y el piso esta cubierto de baldosa bajo la cual se encuentra la placa de concreto.

Este escenario se encuentra en el segundo piso del edificio de alta tensión de la Universidad Industrial de Santander; la locación es un laboratorio que cuenta con los equipos de cómputo, mesas de madera, armarios de madera y metal, sillas plásticas y demás instrumentos y sistemas de desarrollo propios de cada laboratorio, como se observa en la figura 25.

Escenario 3. Este escenario corresponde a la totalidad del segundo piso del edificio de alta tensión de Ingeniería Eléctrica el cual tiene las mismas especificaciones de construcción que el escenario dos, figura 26.

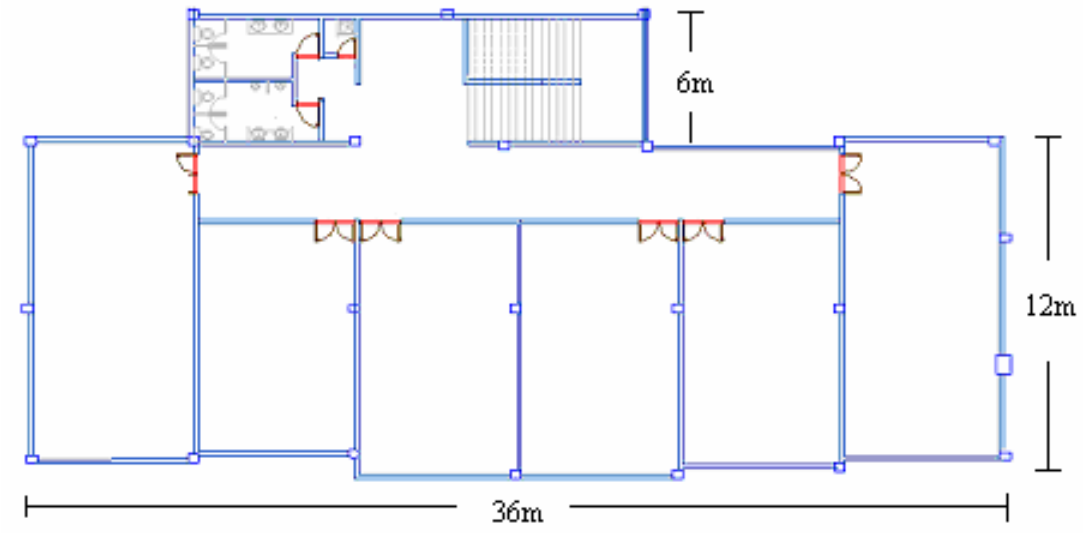
Se escogió este escenario con el fin de tener una idea más general del comportamiento del throughput en una extensión más amplia que el escenario dos, obteniendo un solo valor para todo el escenario y así poder simular sin tener variabilidad por efectos de distancia.

Figura 25. Distribución de mueblería del segundo escenario.



Fuente, Autores.

Figura 26. Escenario de simulación multiusuarios



Fuente, Autores.

4. VALIDACIÓN DEL SIMULADOR

Teniendo ya las bases de simulación y las pruebas realizadas para los diferentes escenarios se hizo una comparación de los resultados para validar los datos del simulador en contraste con ambientes reales.

4.1 RESULTADOS DE LAS PRUEBAS REALES

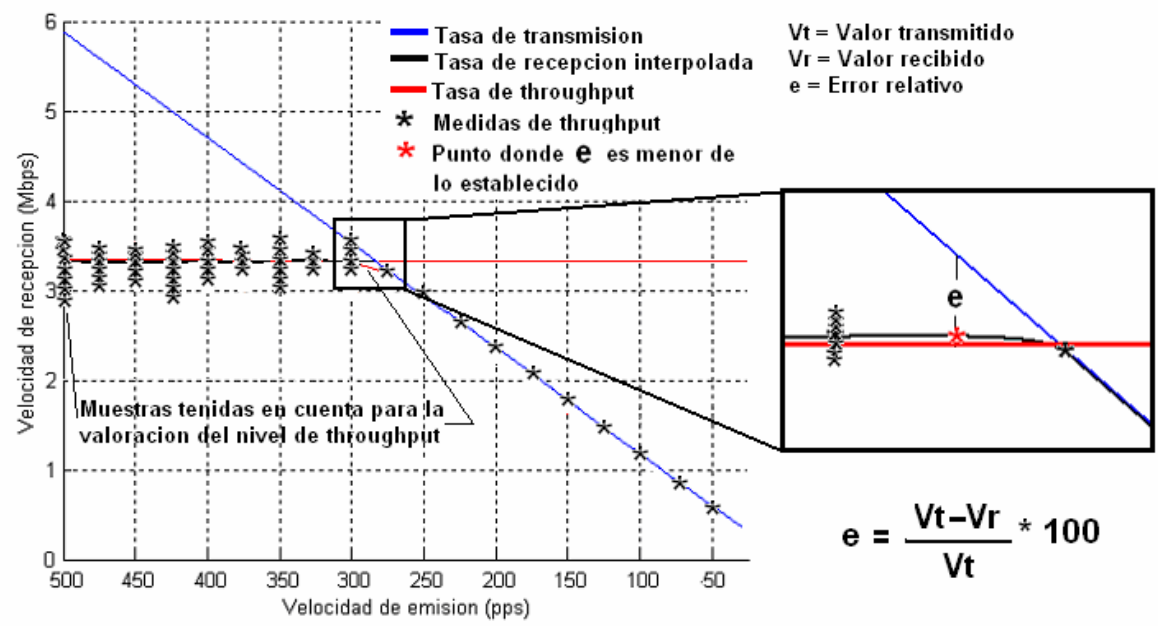
Antes de presentar los resultados es necesario explicar de manera no muy detallada la forma como se calcularon los valores de throughput y el error que se tuvo en cuenta para este cálculo.

Nivel de throughput y rangos de error. Teniendo varias mediciones de recepción de tráfico para cada velocidad, se tomó el promedio de todas estas para obtener una sola gráfica resultante como se observa en los puntos negros de la figura 27. A estos datos resultantes se les realizó una interpolación de 100[24] puntos entre cada muestra, con el fin de obtener una mayor resolución, línea negra; lo mismo se hizo para los datos de emisión de paquetes como se observa para la línea azul de la siguiente figura.

Teniendo las graficas de emisión y recepción interpolada se busca el punto donde la velocidad de emisión y recepción solo se diferencian en un porcentaje de error de acuerdo a la precisión necesaria respecto a los valores de emisión; una vez encontrado ese punto como lo muestra la figura 27, se toman las muestras anteriores de recepción y se calcula un promedio fijándose este como el valor de throughput que es la línea horizontal color roja que se muestra en la figura 27 y en cada gráfica a continuación.

[24] Este nivel de definición para la grafica interpolada se escogió con el ánimo de tener una gráfica lo mas continua posible y que no tuviera tanta carga computacional.

Figura 27. Cálculo del nivel de throughput de acuerdo al error estipulado.



Fuente, Autores.

Durante el procesado de los datos se hicieron variaciones de error entre 10% y 0.1% observándose que no se tenían mayores cambios en los resultados, por lo cual teniendo en cuenta la invariabilidad con respecto al error, se toma como error para este estudio el valor de 5%, además que coincide con el valor de exactitud que tienen los datos tomados, ya que estos se realizan con disminuciones de 5% respecto a la mayor tasa que es 500pps.

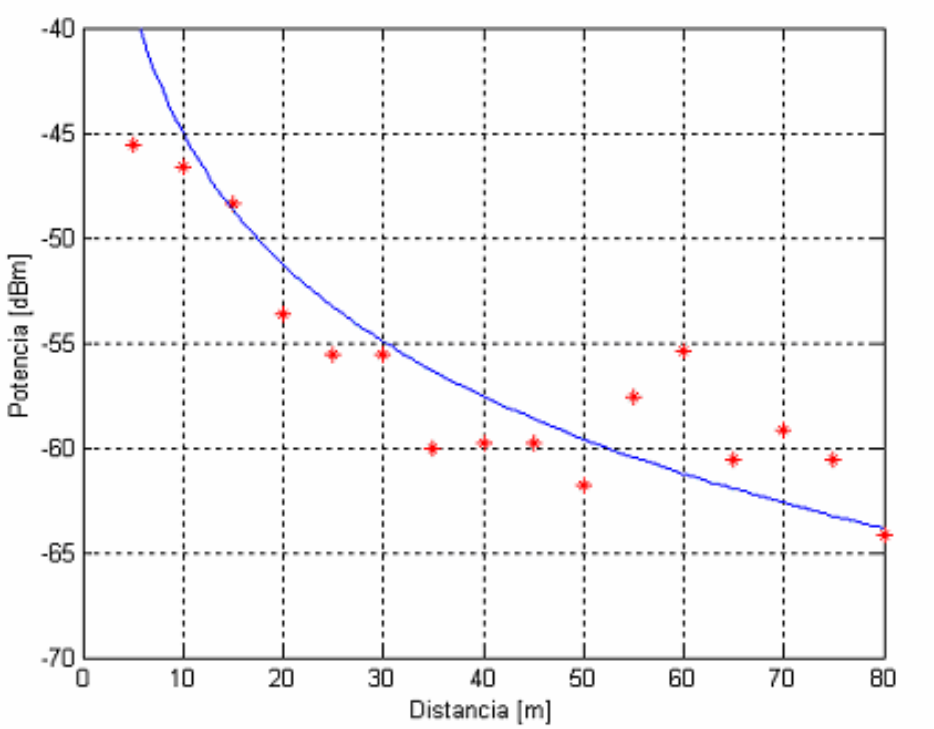
4.1.1 Prueba 1, determinación de la tasa de throughput en espacio libre con un usuario. Las medidas de potencia para los 80m definidos dan como resultado los puntos rojos de la figura 28.

Con estos datos se calcula la curva que mejor se adecúe a esta tendencia tal como se explico en la sección 3.2.2, y se extraen las constantes necesarias para describir el escenario mediante la ecuación de radio propagación; estas

constantes son “β” (Beta) y “σ” (Sigma), de las cuales se explica en la sección 1.3, obteniéndose los siguientes valores:

“B” (BETA)	2.086
“σ” (Sigma),	3.1477

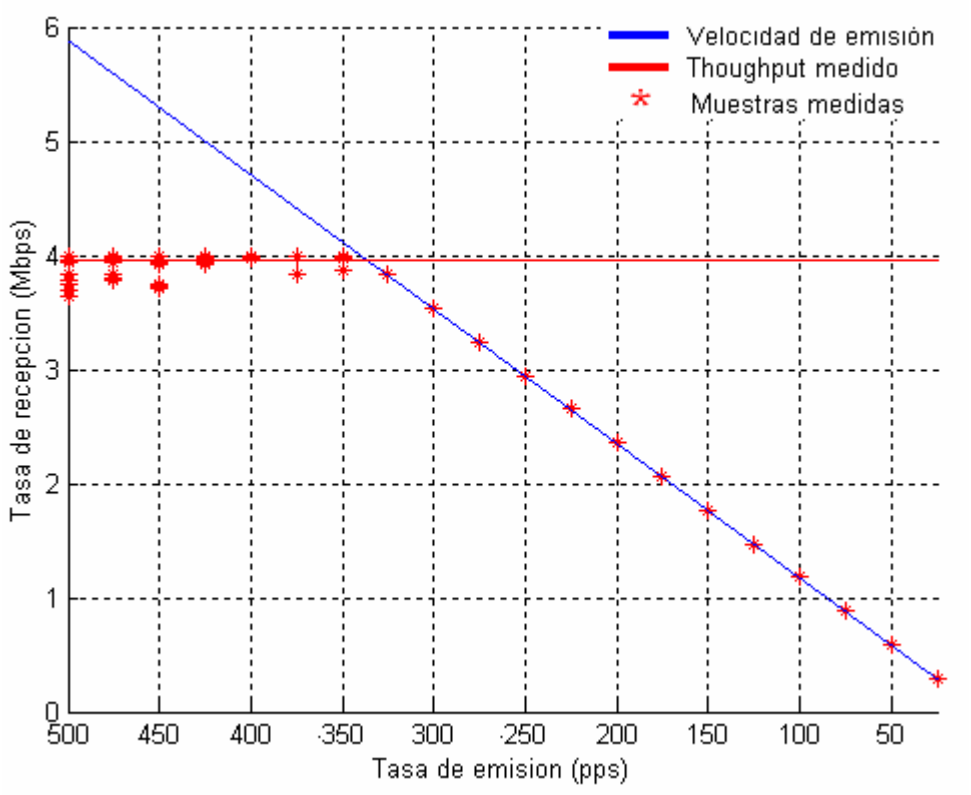
Figura 28. Datos de potencia a través de la distancia.



La curva azul de la figura 28 muestra la gráfica de regresión hasta los 80m de donde se obtuvieron los coeficientes. Se puede observar que estos valores se encuentran dentro del rango de máxima transmisión de paquetes para las tarjetas utilizadas, donde el umbral de recepción de las mismas indica potencias mayores a -83dBm, ver Anexo D.

La prueba de throughput es tomada de acuerdo a las especificaciones planteadas en el capítulo tres y con los mismos patrones de tráfico que se han descrito.

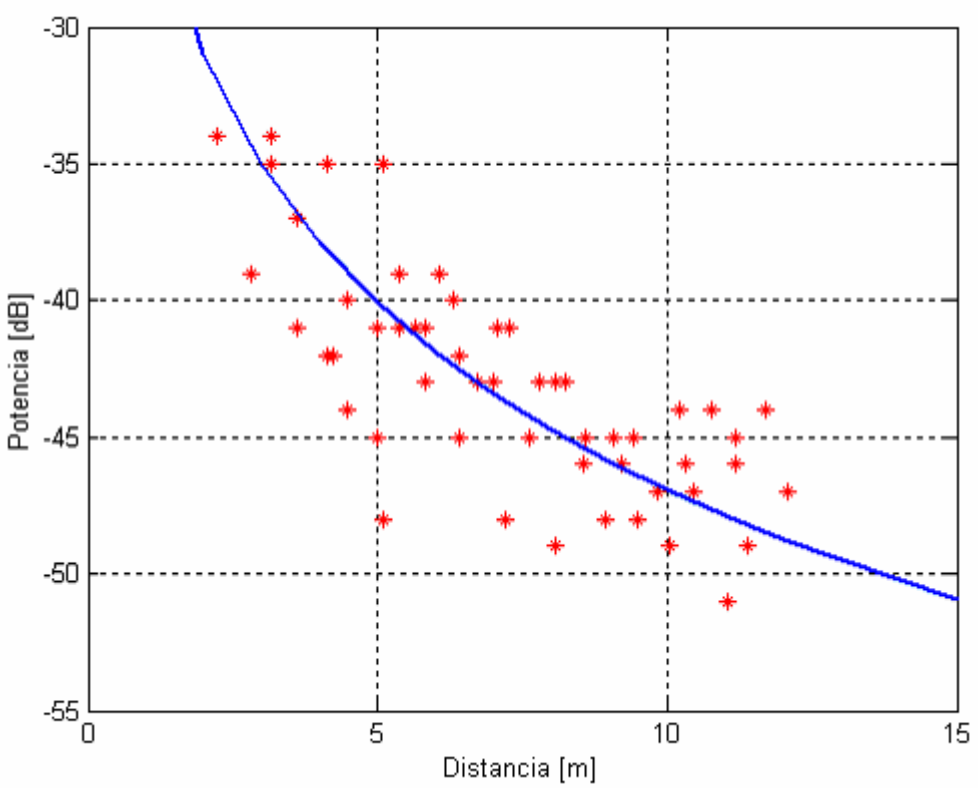
Figura 29. Nivel de throughput en espacio libre.



La figura 29 presenta los datos obtenidos para esta prueba, dejando ver que las medidas tomadas para las diferentes distancias son muy semejantes; esto da la posibilidad de obtener un solo nivel de throughput que para este caso de acuerdo al criterio de nivelación anteriormente descrito tiene un valor de 3.9461 Mbps que corresponde a la línea horizontal color rojo.

4.1.2 Prueba 2, determinación de la tasa de throughput en interiores sin obstáculos para un usuario. En este escenario también son tomadas las respectivas medidas de potencia con el fin de obtener los coeficientes característicos de la ecuación de radiopropagación. Las medidas se tomaron en toda la extensión del escenario dependiendo tanto de una variable de posición horizontal "X" como de una variable vertical "Y", se le aplica la transformación explicada en la sección 3.2.2 para que la potencia quede en función de la variable distancia "d", obteniéndose la siguiente gráfica:

Figura 30. Aproximación a una curva de los datos de potencia tomados.



Como es de esperarse, se encuentra una distribución decreciente que varía entre -34 dBm y -52 dBm, indicando así que todo el escenario se encuentra dentro del rango de sensibilidad de las tarjetas el cual está por encima del umbral de -83dBm para una velocidad de enlace de radio de 11 Mbps, según lo indican las hojas de especificaciones técnicas presentadas en el Anexo D. Esto es muy importante a la hora de simular un escenario real, ya que se puede asegurar que en el escenario estudiado no existen cambios de velocidad por parte de los equipos debido a variaciones de potencia en la señal, y por tanto el sistema está trabajando a la misma velocidad de enlace.

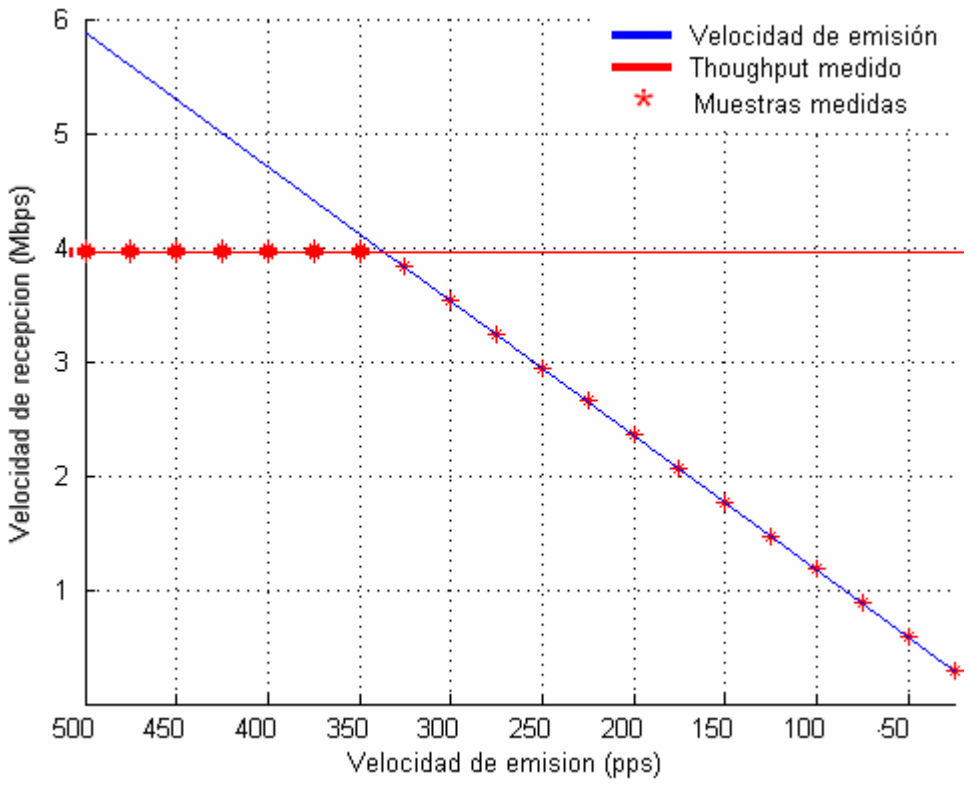
El procesado de los datos tal como se indica en la sección 3.2.2 de los diferentes niveles de potencia a través del escenario da como resultado los coeficientes que mejor describen la tendencia general de acuerdo a su nivel de

dispersión y atenuación, estos coeficientes son utilizados en las simulaciones con el modelo de radio propagación de Shadowing explicado en el capítulo uno.

"B" (BETA)	2.2780
"σ" (Sigma)	2.7412

La prueba de tráfico es realizada de acuerdo a lo descrito en el anterior capítulo y se obtiene la siguiente gráfica de recepción de paquetes respecto de la tasa de transmisión:

Figura 31. Nivel de throughput en espacio cerrado.

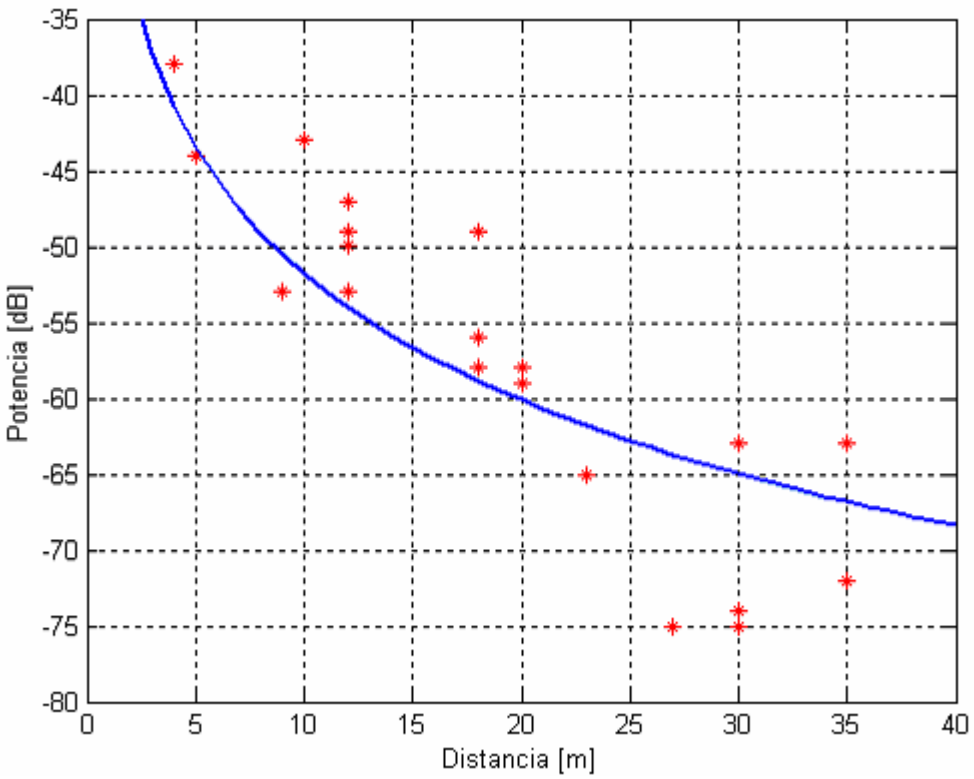


De acuerdo con la figura anterior se puede notar a simple vista que los datos son semejantes pese a ser tomados en distintas posiciones, por lo tanto se

permite establecer un solo valor de throughput para el escenario completo. En este caso, de acuerdo al criterio establecido es de 3.9628 Mbps para todo el escenario, siendo este valor el correspondiente al nivel representado por la línea de color rojo presente en la figura 31.

4.1.3 Prueba 3, determinación de la tasa de throughput en interiores para varios usuarios. Es necesario hallar los coeficientes característicos de este escenario para describir por medio de un modelo de radio propagación la distribución de potencia del escenario en general.

Figura 32. Gráfica de potencia para todo el escenario.



Después de tener las mediciones hechas en toda la extensión del escenario para los niveles de potencia, figura 32, se procede a realizar el mismo proceso que en las anteriores pruebas. Se toman los puntos de prueba y se hace la aproximación a la curva con el mejor coeficiente de correlación que

corresponda a la tendencia general de los datos y finalmente se extraen los coeficientes necesarios para la descripción física de radio propagación del escenario.

"B" (BETA)	2.7640
"σ" (Sigma),	5.7381

Tal como se observó en la descripción hecha en el capítulo tres, este escenario tiene lugares con pocos obstáculos (paredes), como también lugares con varios obstáculos; por tanto, estos coeficientes describen en forma general el escenario pero es muy posible que se presenten variaciones de estos coeficientes al escoger un sitio en particular dentro de este escenario.

La prueba de throughput es realizada en tres sitios diferentes para obtener una caracterización más acorde a la totalidad del escenario y poder elegir un nivel de throughput que describa al escenario en general

Teniendo establecido un valor de 5% para el error se procede a determinar el nivel respectivo de throughput para cada cantidad de usuarios con que se realiza el estudio; este nivel es determinado tal como se ha hecho para los escenarios anteriores.

Las siguientes figuras [25] presentan los datos obtenidos para una cantidad variable de usuarios conectados al mismo tiempo recibiendo tráfico UDP en la red, tal como se especifico en el capítulo tres para las pruebas multiusuario.

Figura 33. Nivel de throughput para un solo usuario.

[25] La curva roja continua mostrada en las siguientes figuras representa la "tasa de recepción promedio interpolada" y no indica que las pruebas se hayan hecho de forma continua en el tiempo.

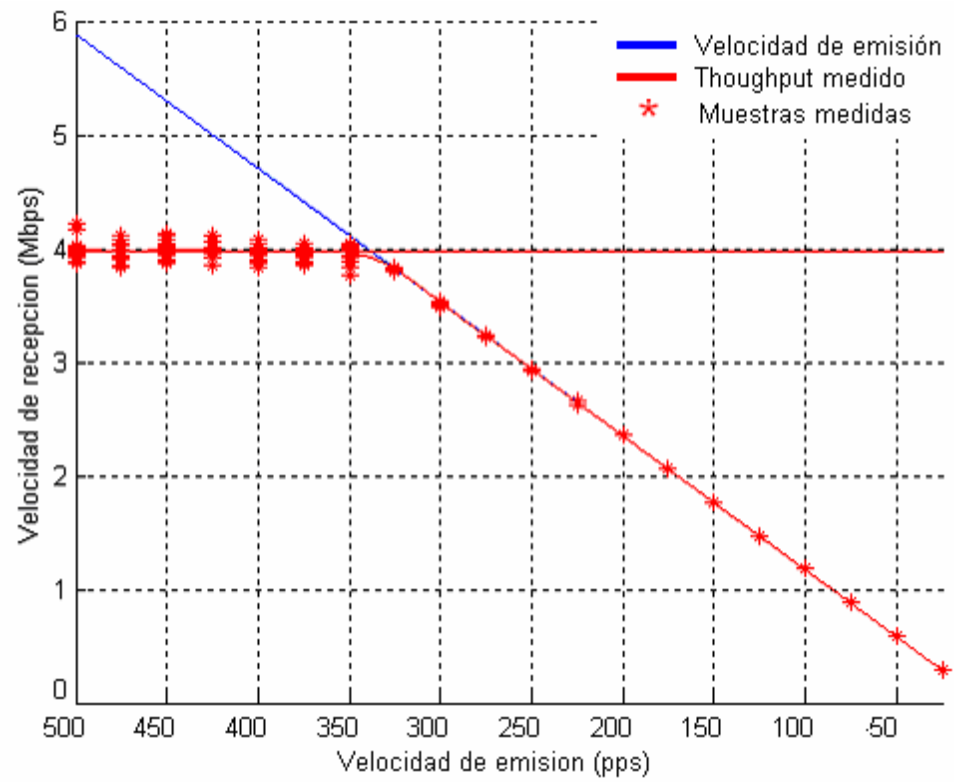


Figura 34. Nivel de throughput para dos usuarios.

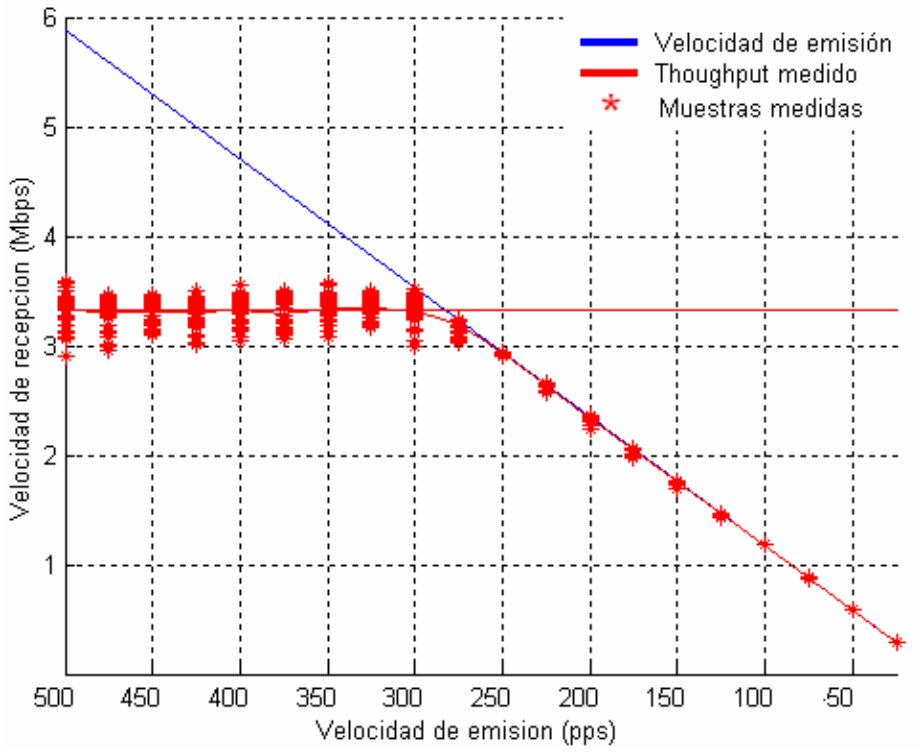


Figura 35. Nivel de throughput para tres usuarios.

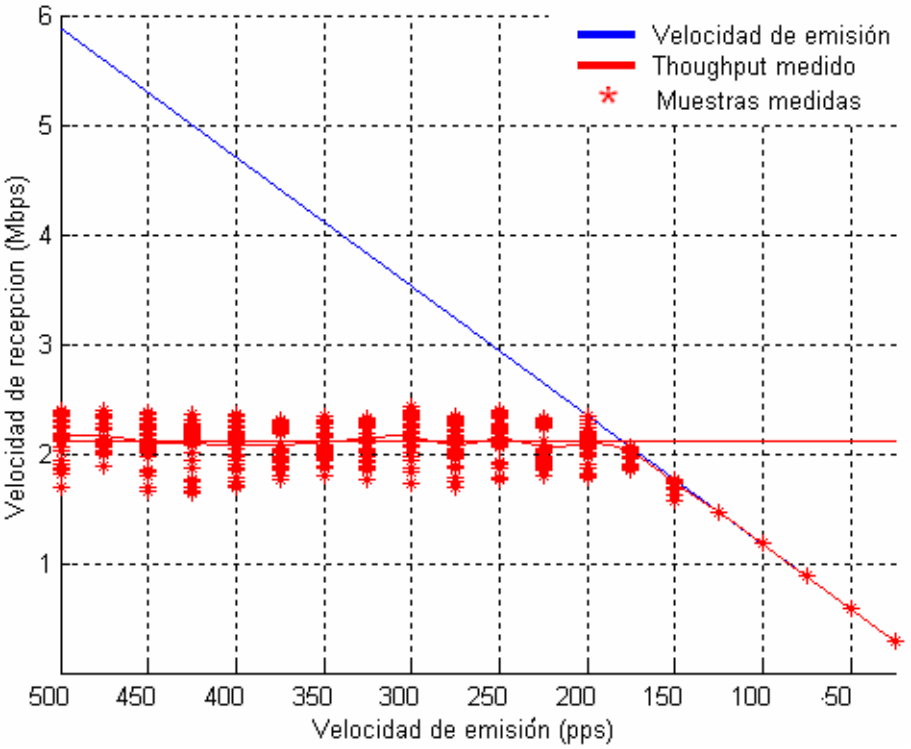


Figura 36. Nivel de throughput para cuatro usuarios.

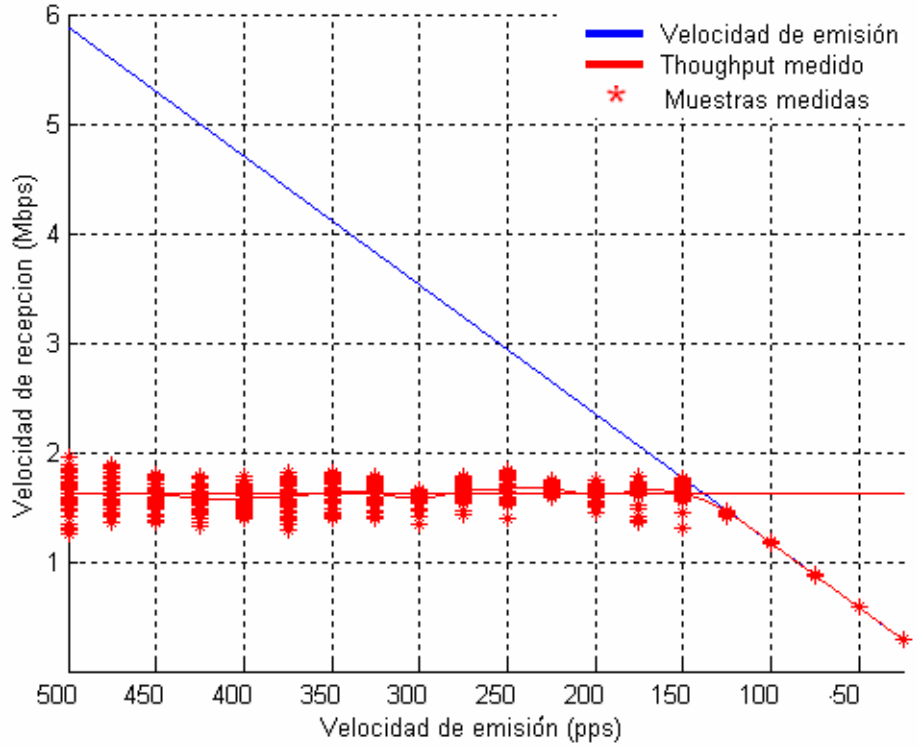


Figura 37. Nivel de throughput para cinco usuarios.

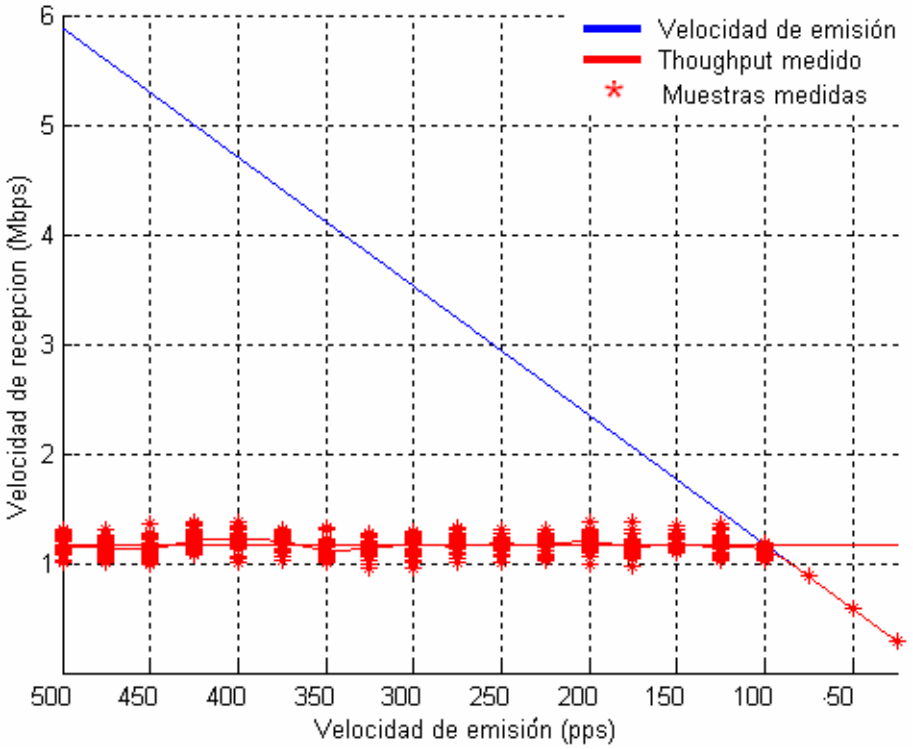


Figura 38. Nivel de throughput para seis usuarios.

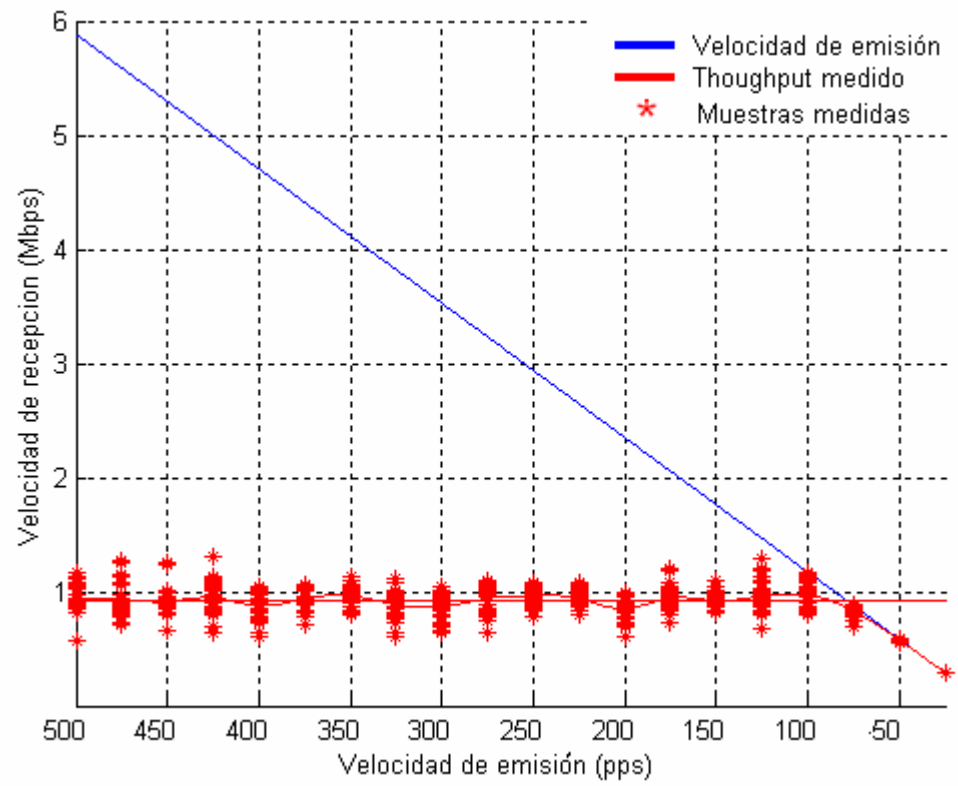


Figura 39. Nivel de throughput para siete usuarios.

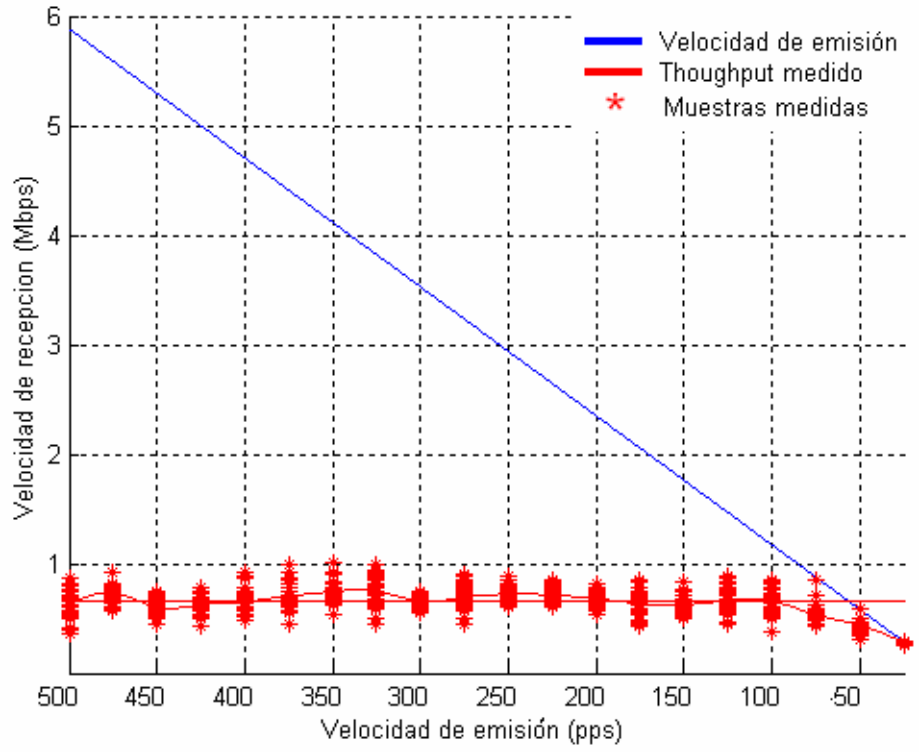


Figura 40. Nivel de throughput para ocho usuarios.

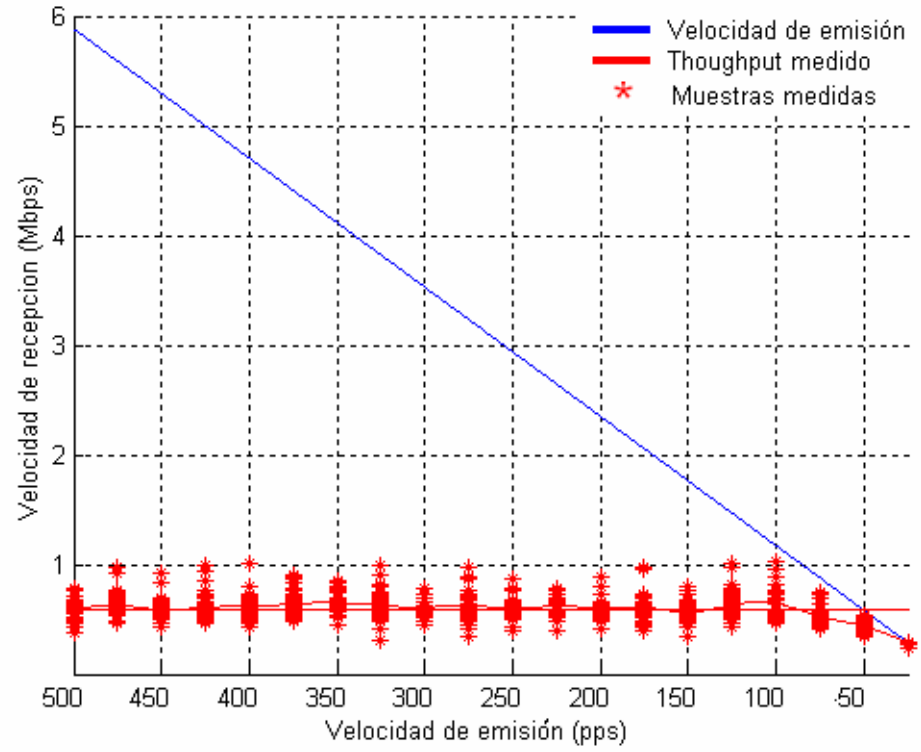


Figura 41. Resumen de los resultados por cantidad de usuarios.

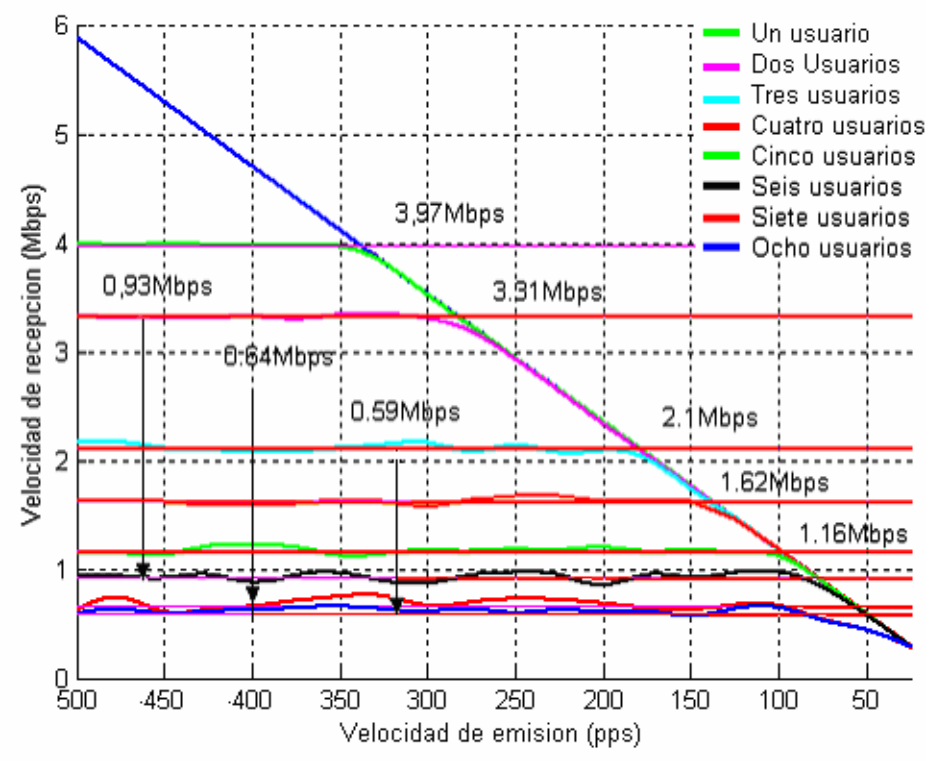
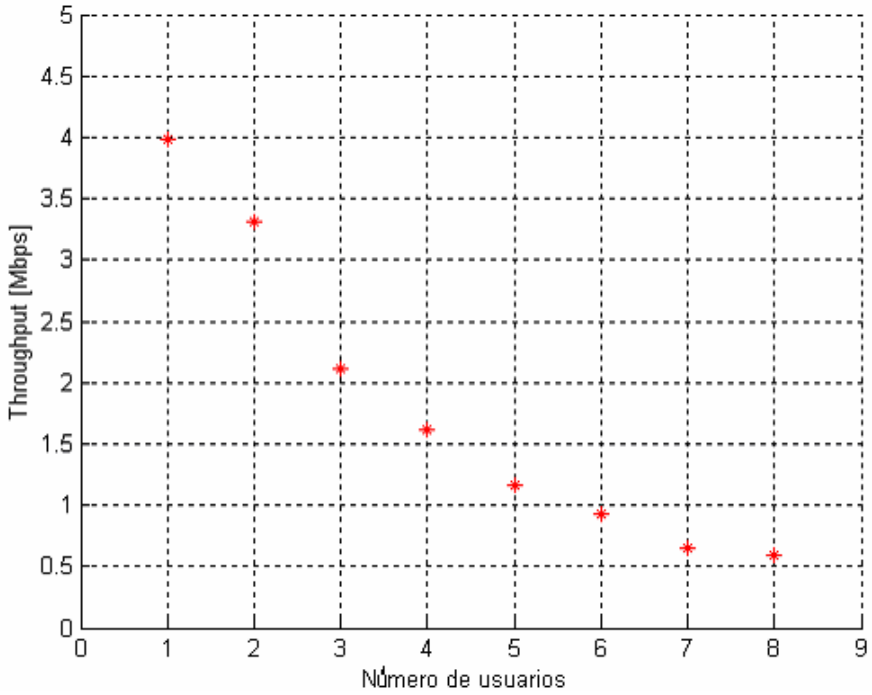


Figura 42. Nivel de throughput para usuarios entre uno y ocho.



La figura 41 muestra en definitiva los valores de throughput finales que se toman para establecer la tendencia del tráfico multiusuario. Para el nivel de error determinado (5%), se tiene en definitiva la gráfica de tendencia del nivel de throughput en función del número de usuarios, figura 42.

De acuerdo con este resultado se busca una curva cuyo coeficiente de correlación sea el mejor por medio de regresiones lineales; Las curvas negra y azul de la figura 43 corresponden a las dos regresiones lineales obtenidas con mayor correlación. Sin embargo estas curvas no muestran una tendencia adecuada ya que la regresión 2 presenta un punto de inflexión lo que hace que la curva se incremente para un número igual o superior a 8 usuarios; la regresión 4 corrige este error de tendencia decreciente para mas de 8 usuarios pero cerca del primer usuario presenta un punto de inflexión que no corresponde a la tendencia que se espera para describir los datos. Por estas razones se procede a realizar una regresión no lineal hacia una exponencial decreciente del tipo:

$$Y = a\left(1 - e^{-bX}\right) + c \quad (1)$$

La curva roja de la figura 43 ilustra el resultado obtenido tomando esta como modelo matemático para la aproximación de los datos estimados, no solo porque posee uno de los mejores coeficientes de correlación según la tabla 9, sino porque tiene las características de una exponencial decreciente que es justo la tendencia que presentan los datos obtenidos de las mediciones.

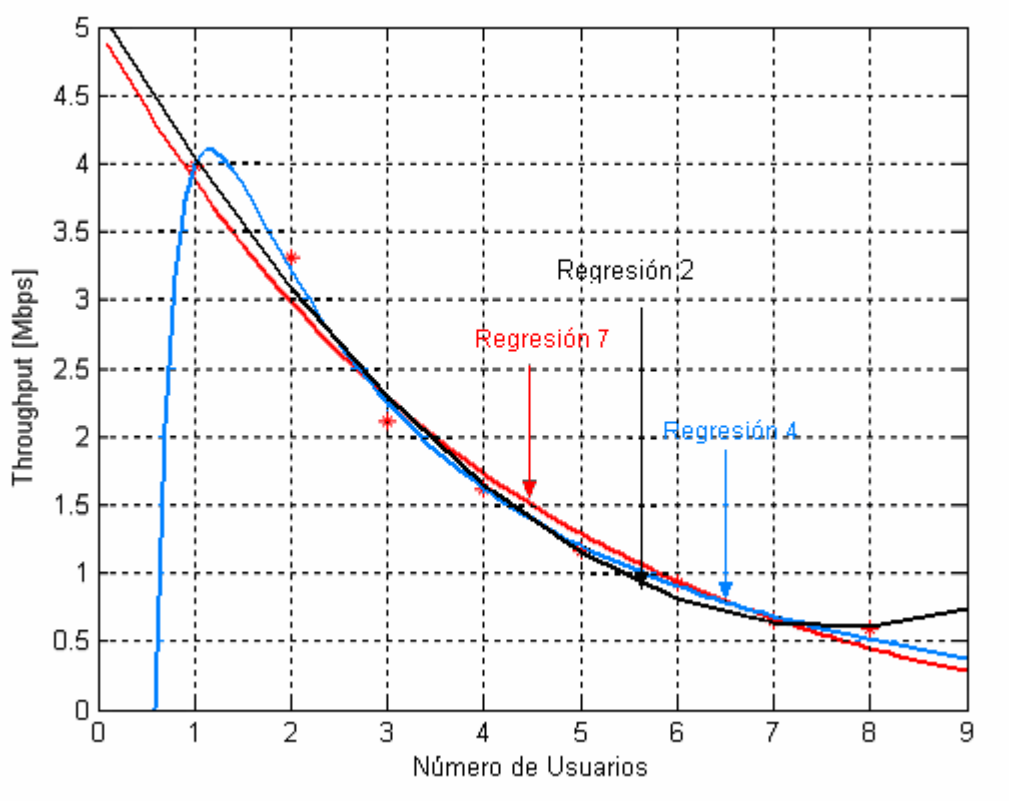
Por lo tanto la ecuación que mejor describe los datos anteriormente recopilados queda expresada de la siguiente manera para los coeficientes a=5.37, b=0.23 y c=-5.

$$Y = -5.3717\left(1 - e^{-0.2346X}\right) - 5 \quad (2)$$

Tabla 9. Coeficiente de determinación y correlación de las curvas.

	INTERPOLACIÓN	DETERMINACIÓN	CORRELACIÓN
1	aX + b	0.9013	0.9494
2	aX² +bX+C	0.9910	0.9955
3	a/X + b	0.8726	0.9341
4	a/X +b/X² + c	0.9967	0.9983
5	a +b/x +ax	0.9648	0.9822
6	a + b/X +cX2	0.9484	0.9738
7	a(1-e(-bX))+c	0.9821	0.9910

Figura 43. Regresiones con mayor correlación.

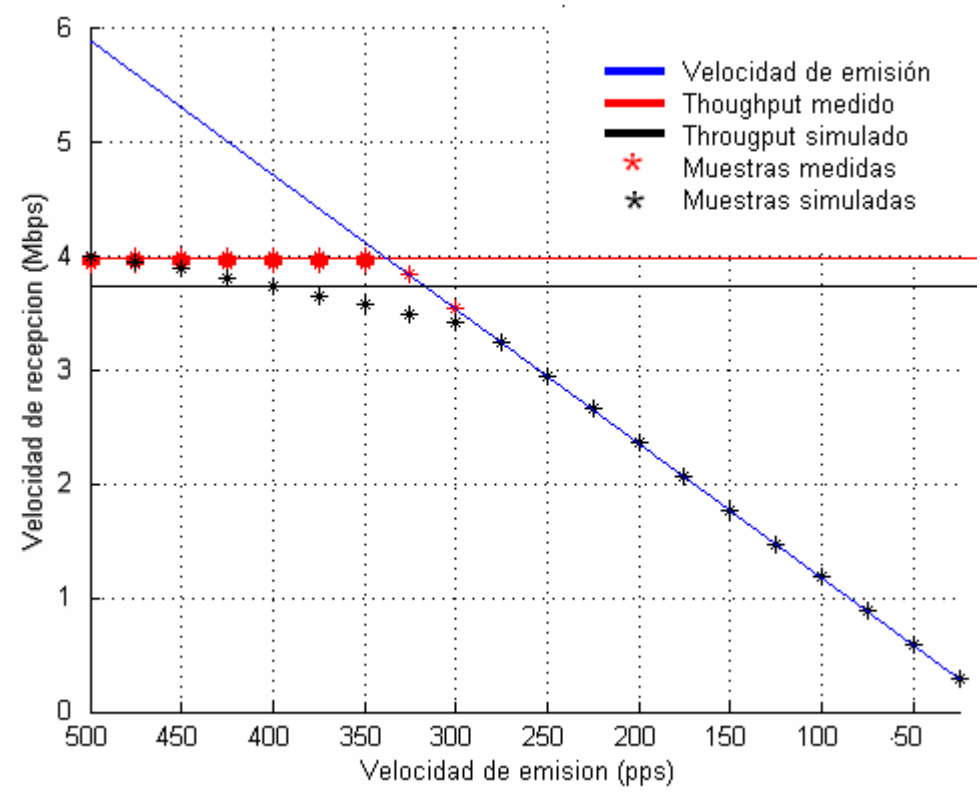


4.2 RESULTADOS DE LAS SIMULACIONES

Basados en los parámetros hallados para los diferentes escenarios (σ y β), con las especificaciones físicas de las tarjetas y con el mismo tráfico para las pruebas reales se procedió a hacer las diferentes simulaciones con el fin de comparar los resultados obtenidos.

4.2.1 Simulación 1, determinación de la tasa de throughput en espacio libre con un usuario. Tal como las pruebas reales, las pruebas simuladas se hicieron llegar hasta 80m, dando como resultado la siguiente gráfica:

Figura 44. Gráfica de recepción de paquetes para los diferentes puntos a través de la línea de prueba.



En la figura anterior comparando las líneas horizontales roja y negra se puede observar una diferencia en los niveles de throughput obtenidos de medidas reales y medidas de simulación; de acuerdo con la gráfica este bajo nivel es debido a que la velocidad de recepción en simulación no es constante durante

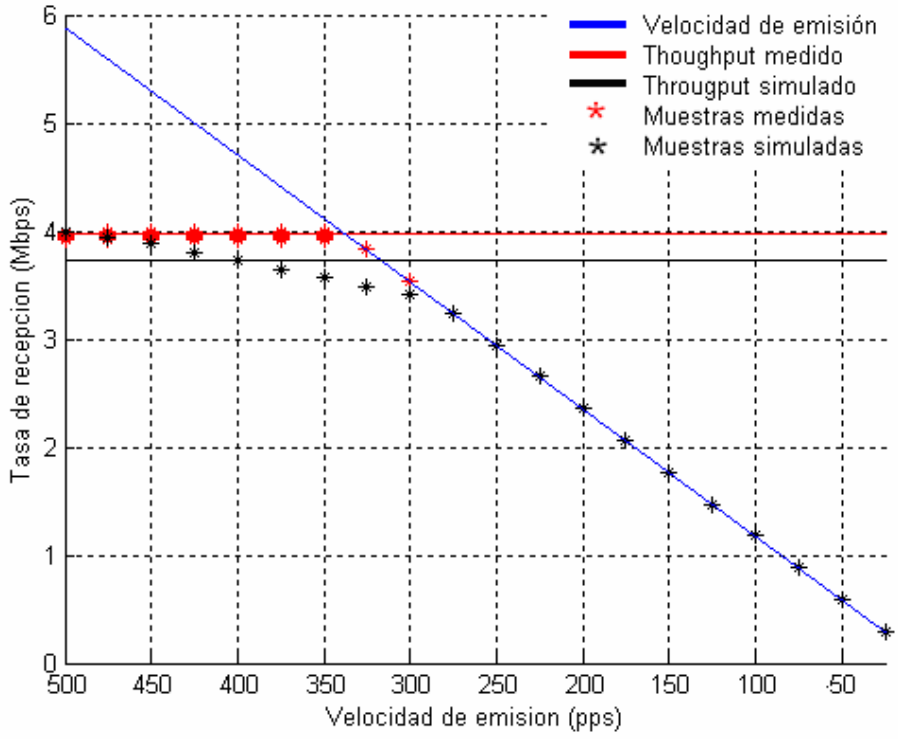
el lapso donde existen pérdidas, se puede notar que aunque las muestras reales y simuladas empiezan con el mismo nivel, las muestras simuladas descienden al disminuir la velocidad de transmisión.

Los valores de las líneas horizontales de la figura 45 junto con el error y la desviación estándar respecto de los datos reales son:

VALOR REAL	3.9757Mbps
VALOR SIMULADO	3.7271Mbps
ERROR	6.2529 %
DESVIACION ESTANDAR	0,163899Mbps

4.2.2 Simulación 2, determinación de la tasa de throughput en interiores sin obstáculos para un usuario.

Figura 45. Gráfica de recepción de paquetes para los diferentes puntos en el escenario.



En la anterior gráfica se presentan las 54 medidas estipuladas para este escenario en el ambiente simulado (puntos negros), notándose una clara coincidencia entre las medidas, esto debido a que dentro del modelo de radio propagación todas las muestras están en el rango de máxima recepción y no existen variaciones significativas. Por lo tanto el valor de throughput estimado puede ser utilizado para describir el escenario en general.

Los valores correspondientes a los niveles de throughput, el error y la desviación entre los datos medidos y los reales son los siguientes:

VALOR REAL	3.9638Mbps
VALOR SIMULADO	3.6470Mbps
ERROR	7.9691%
DESVIACION ESTANDAR	0.200113Mbps

4.2.3 Simulación 3, determinación de la tasa de throughput en interiores para varios usuarios. La simulación para este escenario abarcó tres diferentes distancias (10m, 20m y 30m) igual que en las pruebas reales, con los coeficientes determinados de las pruebas de potencia, con el fin de observar el comportamiento del tráfico a distancias similares a las utilizadas en las pruebas reales, dando como resultado una invariabilidad respecto a la distancia.

Las siguientes figuras presentan los datos por usuarios y no por distancias ya que además de ser invariables con la distancia utilizada para este escenario la idea principal de esta prueba es observar el comportamiento del simulador bajo simulaciones multiusuario.

Figura 46. Gráfica de recepción de paquetes para un usuario.

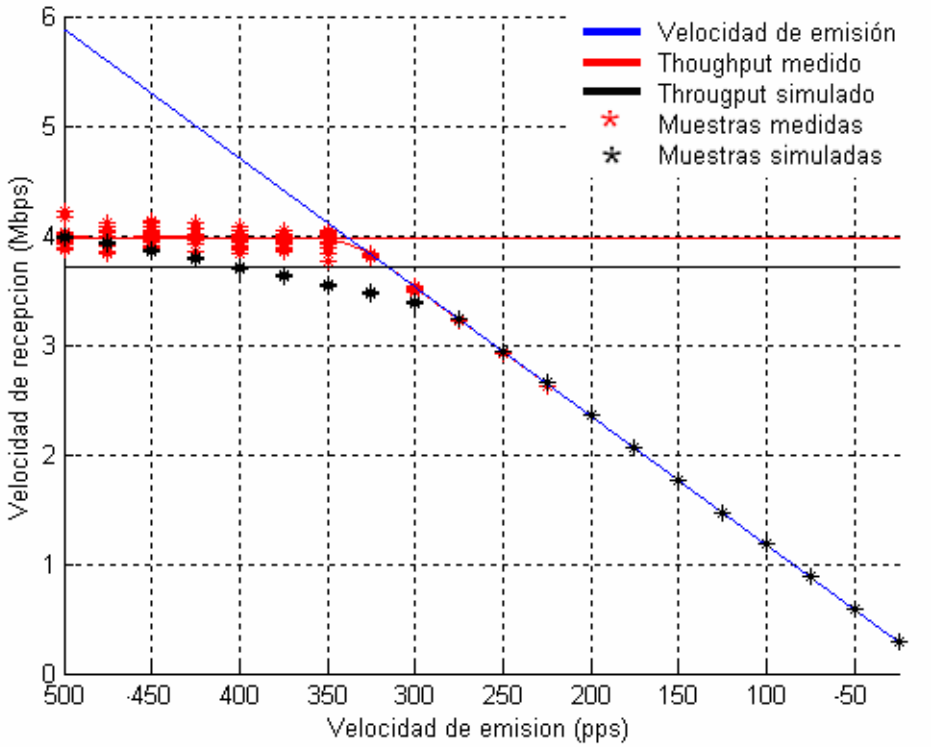


Figura 47. Gráfica de recepción de paquetes para dos usuarios.

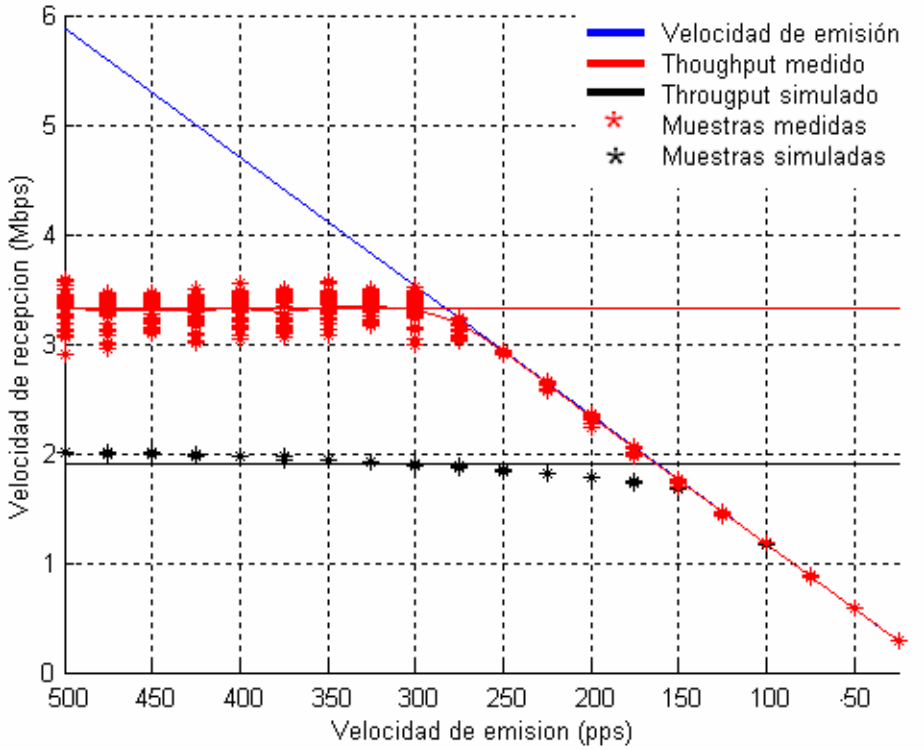


Figura 48. Gráfica de recepción de paquetes para tres usuarios.

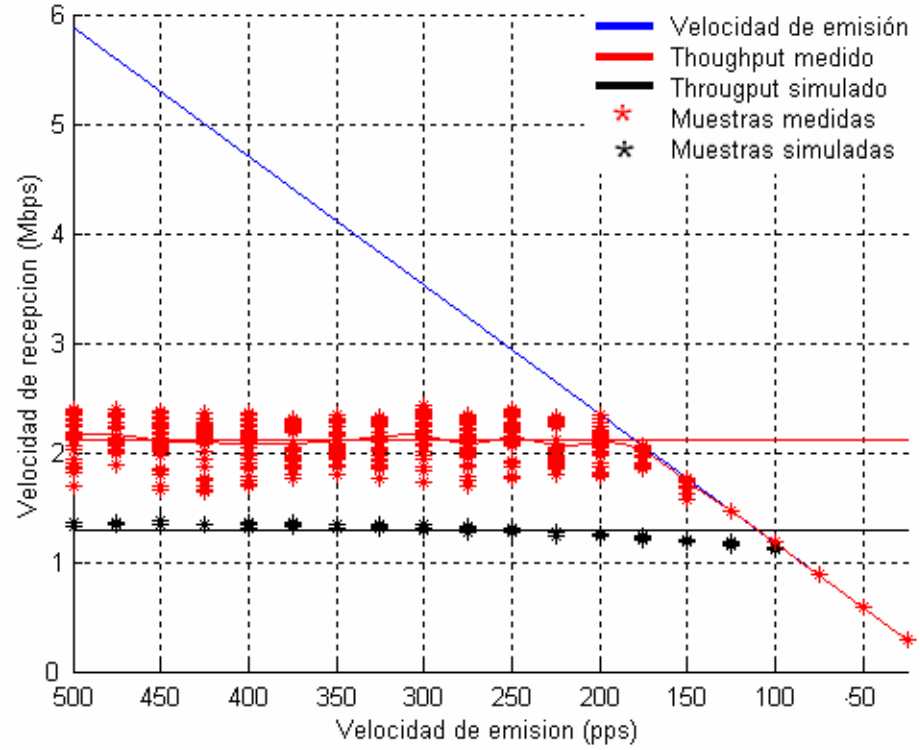


Figura 49. Gráfica de recepción de paquetes para cuatro usuarios.

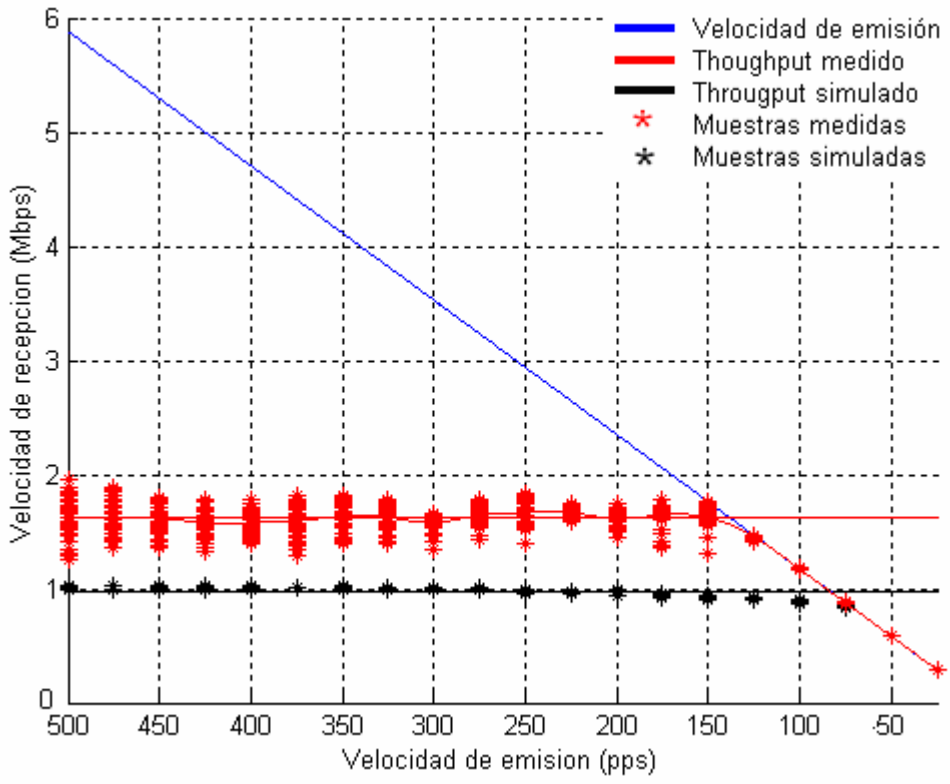


Figura 50. Gráfica de recepción de paquetes para cinco usuarios.

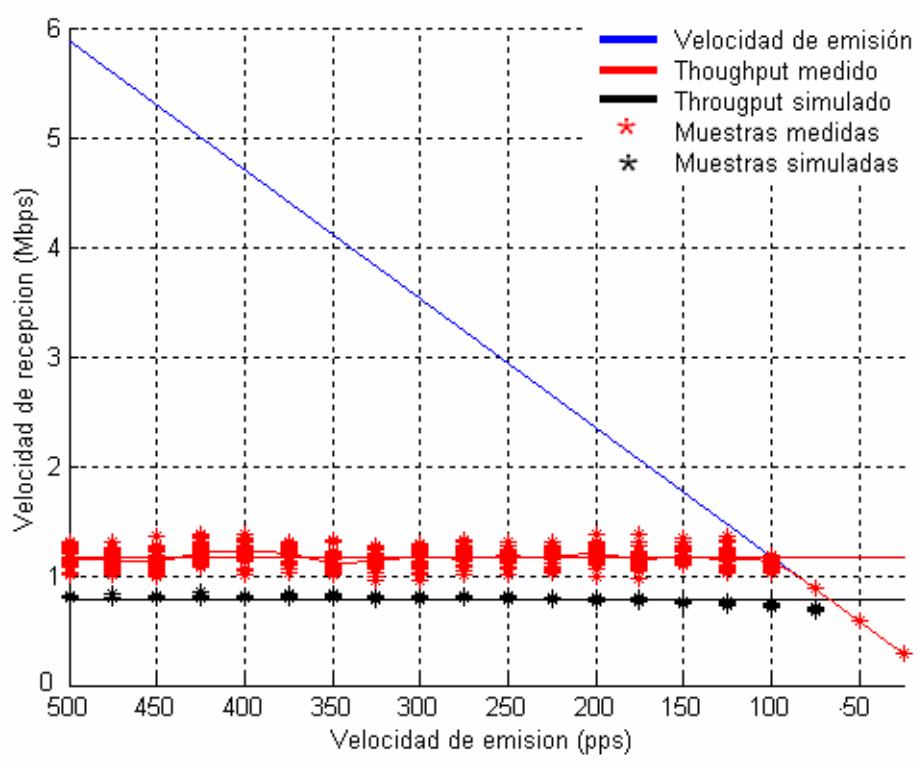


Figura 51. Gráfica de recepción de paquetes para seis usuarios.

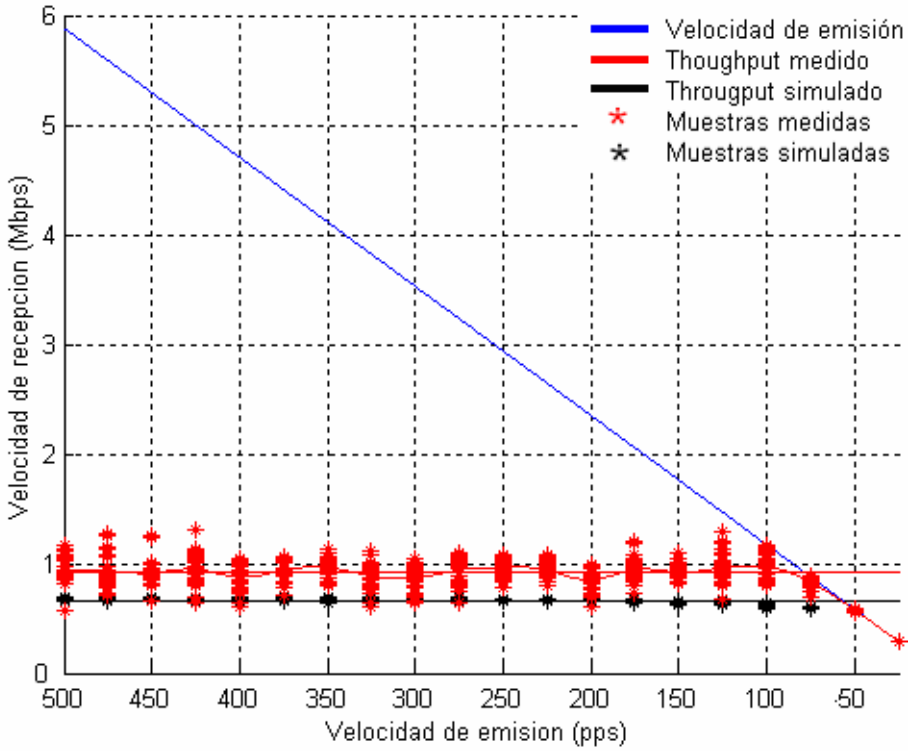


Figura 52. Gráfica de recepción de paquetes para siete usuarios.

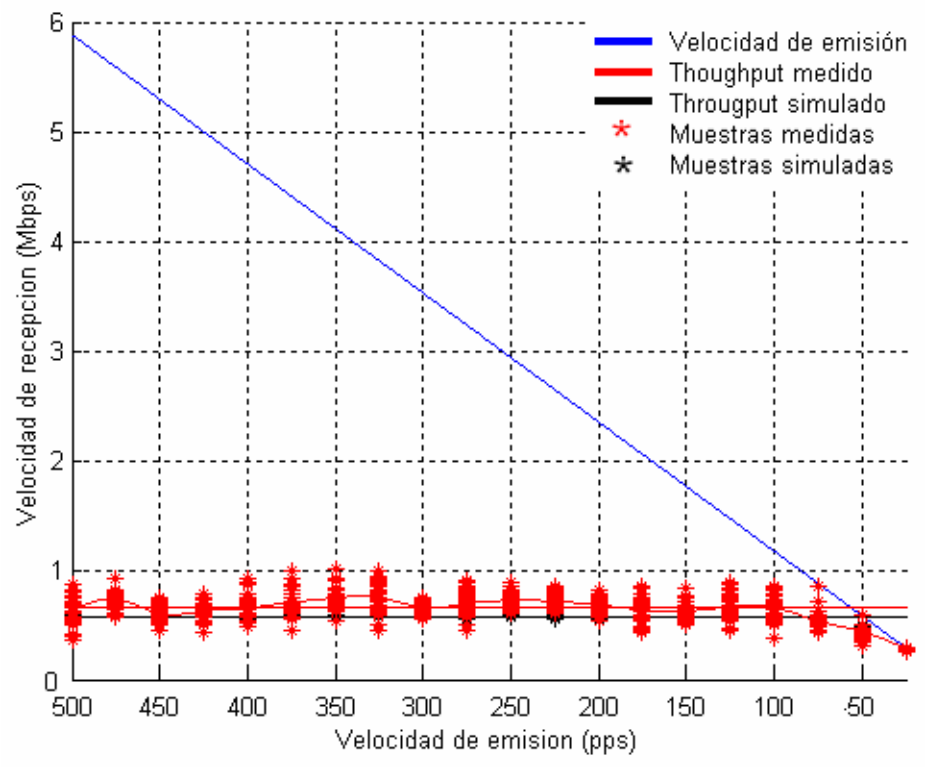


Figura 53. Gráfica de recepción de paquetes para ocho usuarios.

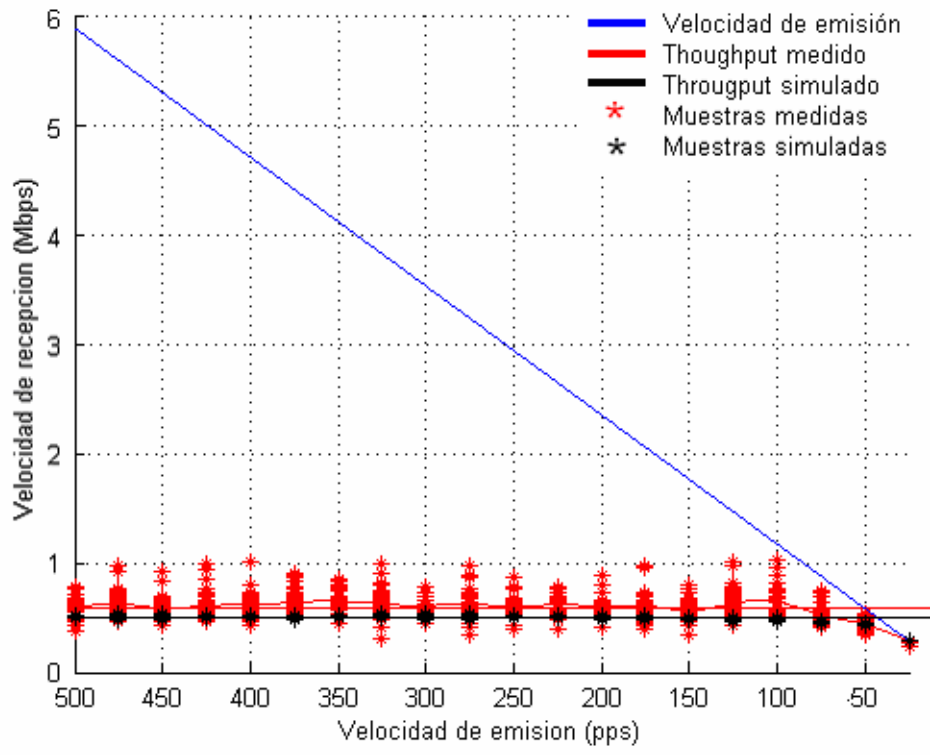
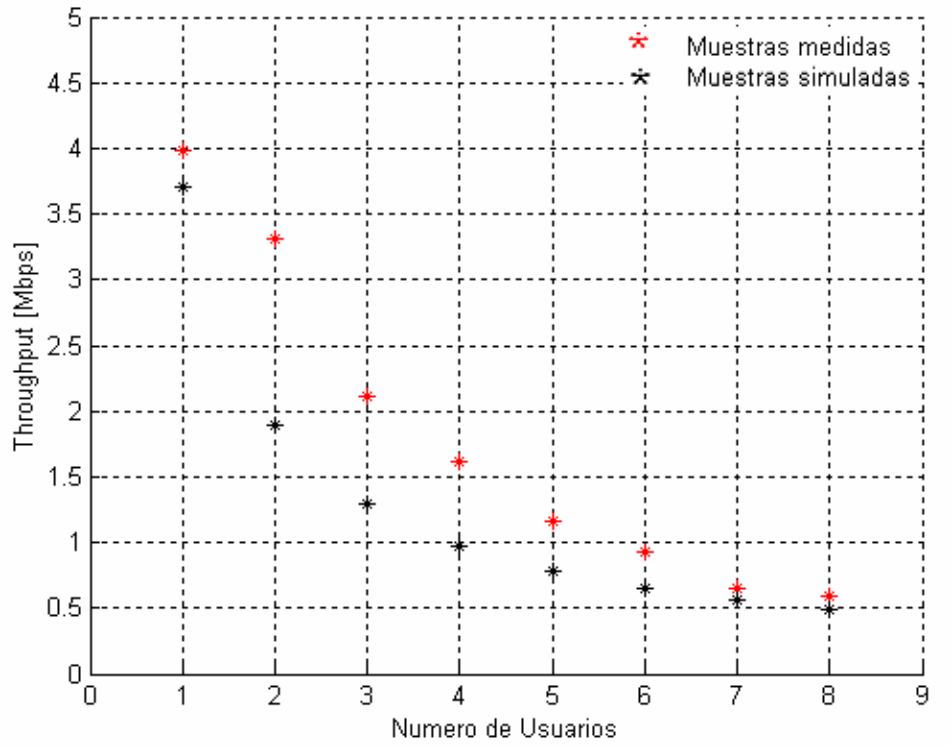


Figura 54. Tasa de throughput en función del número de usuarios.

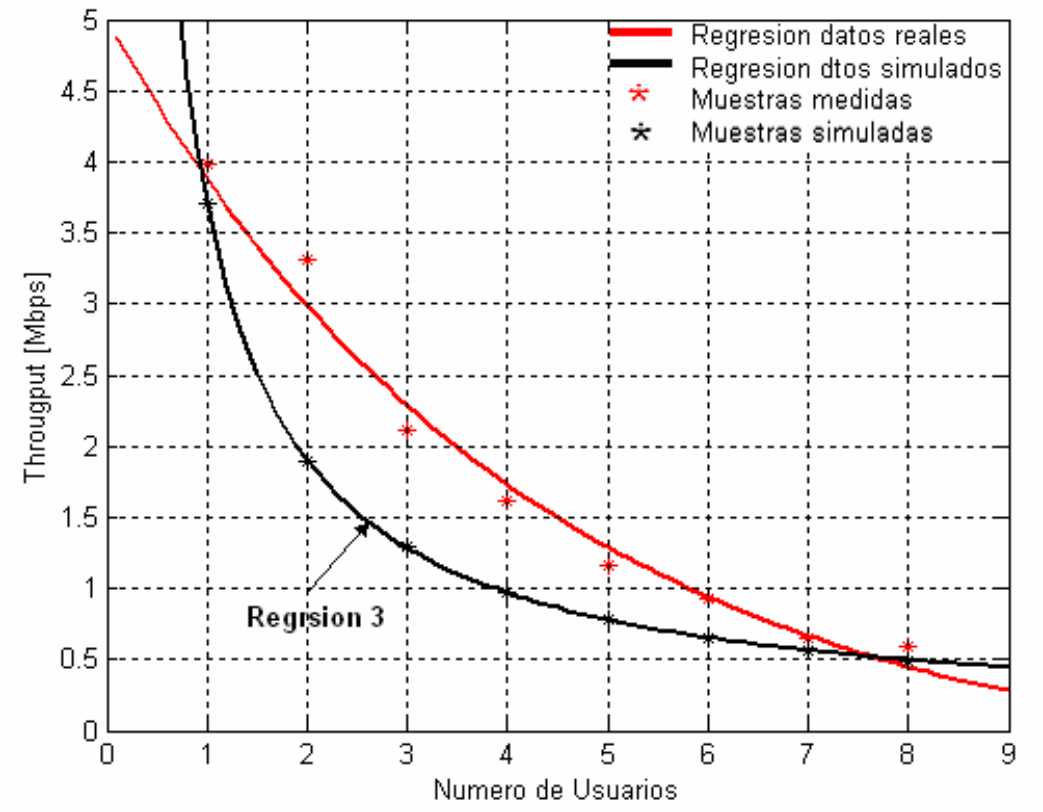


En definitiva se obtiene la gráfica de la figura 55 en donde se tienen los datos de simulación (puntos negros) en contraste con los datos reales (puntos rojos), todos estos datos en función del número de usuarios instantáneamente conectados a la red. Partiendo de esta gráfica se busca la curva que mejor describa las características de estos datos simulados, es decir aquella que posea el mejor coeficiente de correlación que además corresponda a la tendencia de los datos.

Se utiliza el mismo procedimiento que para los datos reales, haciendo una serie de regresiones lineales polinómicas y no lineales buscando el mejor coeficiente de correlación, obteniéndose los resultados consignados en la tabla 10; como se puede observar en la tabla existen varios coeficientes de correlación que tienen el mejor valor que es 1, así que existen varias ecuaciones que podrían modelar los datos. Se escoge la ecuación de la tercera

regresión subrayada en la tabla ya que matemáticamente es la más simple y sencilla, haciéndose la regresión de los datos para obtener la siguiente gráfica:

Figura 55. Regresión de los datos simulados.



Por lo tanto la ecuación que mejor describe los datos anteriormente recopilados queda expresada en la ecuación 2 para a=0.048 y b=3.6735 que es la representada por la línea negra en la anterior gráfica, con una desviación respecto de los datos medidos de 0,699341035 Mbps.

$$Y = \frac{0.0479}{X} + 3.6738 \qquad (3)$$

Tabla 10. Coeficiente de determinación y correlación de las curvas.

	INTERPOLACIÓN	DETERMINACIÓN	CORRELACIÓN
1	aX + b	0.7141	0.8451
2	aX² +bX+C	0.9322	0.9655
3	a/X+b	0.9999	1.0000
4	a/X +b/X² + c	1.0000	1.0000
5	a +b/x +ax	1.0000	1.0000
6	a + b/X +cX2	1.0000	1.0000
7	a(1-e(-bX))+c	0.9599	0.9798

4.2.4 Resumen de errores. La tabla 11 muestra el resumen de los errores presentados a lo largo del capítulo **c**omparando los datos reales y simulados con el fin de dar una idea de la exactitud que puede presentar el simulador; estos valores de error se calculan de la manera como se explicó al principio de este capítulo con los valores promedio de cada prueba.

Tabla 11. Resumen de errores en las pruebas

ESCENARIO	VALOR REAL [Mbps]	VALOR SIMULADO [Mbps]	ERROR [%]	DESVIACIÓN
Espacio libre	3.9757	3.7271	6.2529 %	0,163898897
Espacio cerrado	3.9638	3.6470	7.9691%	0.200113
Un usuario	3.9792	3.7124	6.7048%	0,170597348
Dos usuarios	3.3180	1.8960	42.8571%	1,041108674
Tres usuarios	2.1098	1.2880	38.9516%	0,697220327
Cuatro usuarios	1.6215	0.9743	39.9137%	0,580511132
Cinco usuarios	1.1662	0.7838	32.7903%	0,355710537
Seis usuarios	0.9253	0.6568	29.0176%	0,259905617
Siete usuarios	0.6497	0.5621	13.4831%	0,107818146
Ocho usuarios	0.5952	0.4949	16.8515%	0,108308358

Es notable que el error multiusuario es relativamente grande, ya que según lo visto en la figura 56 es evidente una gran diferencia entre las tendencias de los datos simulados y los datos reales; aunque ambas graficas tiene la característica decreciente se puede observar que los datos simulados caen de forma más rápida que los reales y según las ecuaciones obtenidas, mientras los datos reales corresponden a una ecuación exponencial los datos simulados corresponden a una ecuación polinómica decreciente.

5. CONCLUSIONES

En este capítulo se presentan las ideas finales que se consolidaron durante el trabajo, como producto del análisis de la herramienta para redes inalámbricas 802.11b, bajo las condiciones especificadas.

Después de analizar las propiedades, alcances y limitaciones de la herramienta se obtienen las bases de simulación para redes inalámbricas 802.11b, dando los principales comandos, la estructura y metodología de programación, como también los parámetros generales a tener en cuenta a la hora de simular un escenario. Después de hacer las campañas de medición reales se hace un estudio comparativo de los niveles de tráfico presentados en el simulador contra los niveles de tráfico reales para así determinar las diferencias y similitudes y a partir de las comparaciones hechas se produce el resultado final que son los parámetros necesarios para realizar una simulación basada en un escenario real, tales como "σ" y "β".

Las redes inalámbricas tienen diversos parámetros internos que pueden ser medidos y que tienen influencia directa en el desempeño de una red, tal es el caso del tipo de tarjetas, tipo de antenas, protocolo utilizado, tipo de codificación, tipo de modulación, buffer, etc. Pero también existen parámetros externos que tienen influencia en el desempeño de una red inalámbrica tales como el clima en general, tipo de obstáculos, ubicación, cantidad de usuarios conectados al mismo tiempo etc. Todos estos parámetros se convierten en variables de estudio cuando se quiere representar un enlace de datos inalámbrico, sin embargo tener en cuenta muchas de estas variables es un trabajo tedioso, por lo cual es necesario obviar algunas que no se creen tan indispensables (tipo de ambiente, tipo de hardware, cantidad de obstáculos, etc.) y tratar de estabilizar otras (tamaño de paquete, tipo de paquete, velocidad, escenario, etc.) para así obtener la característica de variabilidad de la red inalámbrica sólo en función de alguna variable en particular.

De acuerdo con la documentación revisada, NS, actualmente en la versión 2, permite trabajar con Redes terrestres, inalámbricas y por satélite con varios algoritmos de enrutado (DV, LS, PIM-DM, PIM-SM, AODV, DSR). Además se puede tener distintas fuentes de tráfico: Web, ftp, telnet, cbr, etc. Y simular fallos como perdidas probabilísticas y deterministas, fallos en la conexión, etc. Junto con las distintas disciplinas de encolado (drop-tail, RED, FQ, SFQ, DRR, etc.) y QoS (calidad de servicio, como por ejemplo InmtServ y Diffserv). Con estas simulaciones es posible observar el flujo del paquete, su encolado y su posible descarte, monitoreando comportamientos del protocolo: comienzo lento de TCP, control de congestión, retransmisión rápida y recuperación, además de trabajar con movimiento de nodos en redes inalámbricas y dar informe de los sucesos más importantes y los estados del protocolo.

Network Simulator trabaja con dos lenguajes de programación, los cuales son C++ y OTcl. Para el presente proyecto se decidió trabajar fundamentalmente con OTcl sin dejar a un lado C++, ya que el objetivo del proyecto es el estudio de redes inalámbricas trabajando en varios escenarios, y se hace poco énfasis en el estudio de protocolos y formatos de paquetes. La programación en OTcl es compleja para principiantes, especialmente porque para el estudio y programación de NS no se encuentra mucha documentación que pueda facilitar el trabajo.

La herramienta NAM, presente en NS, es de gran utilizad, ya que permite la visualización de escenarios de simulación permitiendo observar fácilmente si lo que se ha programado corresponde con lo esperado, ya que la programación se realiza en modo texto. Sin embargo esto tiene sus consecuencias, ya que al utilizar NAM la carga computacional aumenta considerablemente junto con los tiempos de simulación y obtención de resultados. Cabe anotar que esta característica es observable en simulaciones donde el volumen de tráfico presente es alto como ocurre en las simulaciones realizadas para el presente proyecto.

En el caso de este estudio se propusieron tres escenarios en particular para los cuales se dejaron constantes el tipo y tamaño de paquete, variando la posición y la cantidad de usuarios, calculando la tasa de throughput; el throughput es la variable de desempeño que para efectos de este proyecto fue el centro del estudio, ya que es una variable muy apropiada para medir el desempeño de una red porque su valor depende de propiedades del enlace, tales como retardos, procesado de información, velocidad del enlace, interferencia en el enlace, etc.

Para hacer las mediciones de throughput en los diferentes escenarios en función de una sola variable (posición o cantidad de usuarios) es necesario garantizar que durante el proceso de medición no habrá variaciones significativas de otros parámetros externos al estudio tales como disminución de la velocidad del enlace por atenuación o cambios de modulación, para así caracterizar adecuadamente el throughput.

En las campañas de medición al tomar lecturas en los diferentes escenarios escogidos en particular para este estudio se obtuvieron resultados similares, sin importar que esté en espacio libre o dentro de una oficina cerrada con varios obstáculos tales como puertas y paredes, debido a que los rangos de potencia se encuentran dentro del límite establecido para la máxima transmisión de paquetes (11Mbps) y por lo tanto los niveles de recepción de paquetes son los mismos para cualquier posición dentro del escenario.

El simulador no realiza el cambio automático de tasa de transmisión como ocurre en la realidad, donde las tarjetas varían su velocidad de recepción entre 11, 5.5, 2 y 1 Mbps dependiendo de los umbrales de potencia recibida, por lo tanto para obtener resultados comparables es necesario garantizar que no se presentarán cambios de velocidad hechos por el hardware de la tarjeta,

obteniendo las mediciones de throughput solo en función de la posición y la cantidad de usuarios en los diferentes escenarios

Al igual que en la realidad los resultados de simulación no presentaron variabilidad respecto a la distancia debido a que todos los resultados estaban dentro del rango de recepción máximo de las tarjetas, por lo cual todas las medidas en espacio libre y espacio cerrado dieron muy similares.

La ecuación de modelado por sombra (Shadowing) contiene los coeficientes de pérdidas por distancia "β" (Path Loss Exponent) y la desviación por sombra "σ", que son empíricamente determinados por medidas en campo. Valores grandes de "β" corresponden a más obstrucciones y por consiguiente un rápido decrecimiento logarítmico en la potencia, sumado con una variación aleatoria gaussiana con media cero y desviación estándar σ. Esta ecuación es muy compleja de manejar manualmente ya que es un modelo no lineal probabilístico, por lo tanto se debe utilizar sistemas de computo para hallar estos coeficientes por medios iterativos.

Debido al criterio de sensibilidad del simulador, el modelo de propagación para efectos de este proyecto no tiene mucha influencia en la tasa de paquetes perdidos ya que todas las mediciones tomadas están muy por encima del límite en donde se aplica el criterio y por tanto todos los paquetes son aceptados.

Los valores generalizados de "σ" y "β" obtenidos de algunas tablas (tabla 4 y 5) pueden ayudar en una simulación rápida, sin embargo se recomienda hacer las medidas de campo para obtener los coeficientes de la ecuación de radio propagación y verificar si se logra mayor exactitud en la caracterización de los escenarios.

En vista de que el throughput real no presentó mayor variabilidad en función de la distancia se varió la cantidad de usuarios conectados instantáneamente a

la red, observándose un decremento de forma exponencial al adicionar usuarios a la red; este tipo de decremento no fue exactamente el encontrado en los datos simulados ya que se presentó una caída más rápida representada por una ecuación polinómica decreciente, ver sección 4.2.4. Se demuestra así que el simulador maneja la tendencia decreciente para adiciones de nuevos usuarios, pero los resultados son una estimación con un margen relativamente grande de error por diferencias de distribución de ancho de banda; entonces aunque los resultados no son del todo iguales a los medidos, sí califican al NS para hacer simulaciones multiusuario con una buena caracterización de lo que se presenta en la realidad.

Los resultados en general simulados no son exactamente iguales comparados con los datos obtenidos en las mediciones reales, debido a que el simulador, aunque es muy robusto y tiene en cuenta muchas variables del enlace inalámbrico, no permite representar en forma real algunos variables que se comportan de manera aleatoria tales como ruido, interferencia y error; además al hacer las mediciones reales no es posible controlar y estabilizar todo el sistema para que los datos medidos tenga la característica de repetibilidad[26]. Hay que considerar además que las mediciones tienen un error aproximado del 5% debido a que esta fue la variación que se tomo para hacer las mediciones de tráfico, haciendo decrementos del 5% de la tasa total, por lo tanto cualquier resultado estará en un 5% por encima o por debajo del valor real; la técnica para hallar el valor medio de throughput considera también un 5% de error de aproximación para hallar la tasa media de throughput (aunque para variaciones entre 10% y 0.1% el resultado es casi invariable).

También es importante considerar, a la hora de comparar los valores reales con los simulados, que los datos de las mediciones reales no son tenidos en cuenta en forma individual sino en conjunto por escenario, debido a que los

[26] REPETIBILIDAD: Es la característica de una variable medida que indica la invariabilidad de la magnitud tomada en distintos instantes de tiempo bajo las mismas características.

datos a distancias mas alejadas comienzan a experimentar mayor dispersión y por lo tanto los valores individuales se alejan de la tendencia general. Es muy posible que para nuevas mediciones de tráfico se presenten variaciones pequeñas del nivel total de throughput ya que los datos tomados tienen una considerable variabilidad y si no se toman la cantidad de medidas necesarias que demuestren un comportamiento general del enlace se puede presentar niveles de throughput diferentes al medido originalmente, aunque estas pequeñas variaciones no afectan la tendencia general de los datos.

Se debe considerar que aunque los datos simulados no son iguales a los reales, estos si se acercan mucho y contienen la tendencia general de los datos reales obtenidos y por lo tanto es posible mediante el simulador hacer estudios de redes inalámbricas y obtener una idea del comportamiento del enlace a medida que hacemos cambiar alguna variable en simulación considerando de todos modos los márgenes de error existente.

6. RECOMENDACIONES

Se recomiendan fuentes como *MARC GREIS' tutorial web pages* (en el sitio Web) las cuales son de gran eficacia para empezar el estudio de *NS*.

El primer paso es empezar el estudio de un simple *script* de simulación que esta disponible en ~ns/tcl/ex/simple.tcl[27].

CARACTERÍSTICAS INDOCUMENTADAS

La documentación de NS está ubicada en el subdirectorio *doc* del código fuente. NS está creciendo e incluyendo nuevos protocolos pero su documentación no crece así. A continuación se presentan los temas que deben ser mejorados:

1. Interfaz al intérprete: No hay documentación actualmente.

2. Bases del Simulador:
    - Las *LAN*[28] necesitan ser actualizadas para soporte de nuevas redes cableadas e inalámbricas.
    - Se requiere más documentación Inalámbrica.
    - Es importante explicar la lista de opciones de cola.

3. Enrutamiento:
    - La presencia de un capítulo sobre protocolos de enrutamiento *ad-hoc* es requerido.

4. Colas:

[27] "~" Indica la ubicación de la carpeta NS en el disco duro dentro del sistema operativo *linux*, esta depende de donde se haya instalado el software *NS*.
[28] LAN = Red de Área Local.

- Es necesaria la documentación de colas.

5. Trafico y Escenarios:

Con el fin de lograr un mayor entendimiento en cuanto a la utilización de NS es aconsejable agregar:

- Una descripción acerca del manejo del simulador para hacer trazos de gráficos en *NAM* y *XGRAPH*.

6. A nivel general es recomendable:

- Incluir el *Marc Greis' tutorial* antes que referirse a este.

A la hora de realizar una simulación de red inalámbrica, se deben caracterizar correctamente las antenas junto con el escenario de simulación. Para caracterizar las antenas se procede a revisar las hojas de datos de especificaciones técnicas ofrecidas por los fabricantes, de donde se obtienen parámetros como la potencia de transmisión, umbrales de potencia de recepción, frecuencias de canal inalámbrico etc. En cuestiones de escenario se debe hacer un estudio de nivel de potencia, para poder encontrar los valores característicos de coeficientes de pérdidas por distancia "*β*" (Path Loss Exponent) y la desviación por sombra "σ" (desviación estándar), para así tener una caracterización del lugar y para obtener resultados acordes a la realidad, además de modelar los dispositivos inalámbricos utilizando valores proporcionados en las hojas de datos de los fabricantes.

Al simular un escenario inalámbrico, se debe tener en cuenta un modelo de propagación, el cual se escoge entre tres definidos en NS, conocidos como el modelo de línea directa, dos rayos a tierra y sombra, es aconsejable utilizar el modelo de sombra si se requiere un modelado de potencia muy parecido a la realidad ya que es un modelo estadístico y no determinístico, el cual es una ecuación que define que la potencia recibida en cierta distancia es la suma logarítmica de una variable de atenuación con una variable aleatoria o

aleatoria debido a los efectos de propagación de múltiples caminos y dispersión por obstáculos.

REFERENCIAS BIBLIOGRÁFICAS

[1] Agilent Technologies: "Mixed Packet Size Throughput". Actualmente no disponible. http://advanced.comms.agilent.com/routertester/member/journal/1MxdPktnSzThroughput.html

[2] ALVAREZ, J; FLOREZ, ADRIANA; TORRES, Y. "DISEÑO E IMPLEMENTACIÓN DE UNA METODOLOGÍA PARA LA EVALUACIÓN DEL DESEMPEÑO DE UNA RED INALÁMBRICA (WLAN 802.11b)", Grupo De Investigación en Conectividad Y Procesado De Señal, UIS 2003.

[3] ATHEROS COMMUNICATIONS. Methodology for testing Wireless LAN Performance [on line]. 2003. Disponible en Internet: http://www.atheros.com/pt/papers.html.

[4] BING, B. "Measured Performance of the IEEE 802.11 Wireless LAN," Local Computer Network, 1999. Conference on (LCN '99). Pág. 34-42. 1999.

[5] DEMIR, T; KOMAR, C; ERSOY, C. "Measuremed Performance of an IEEE 802.11 Wireless LAN," Proceeding of the Fifteenth International Symposium on Computer and Inforamtion Sciences, Istanbul, Turkey. Pág 246-254, Oct 2000.

[6] DUCHAMP, D.; Reynolds, N. "Measured Performance of a Wireless LAN" Local Computer Networks, 1992.

[7] GATES, Mark; TIRULAMA Ajay; FERGUSON Jim; DUGAN, Jon; QIN, Feng; GIBAS, Kevin. IPERF. National Laboratory for Applied Network, Research

National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign. 2003 http://dast.nlanr.net/Projects/Iperf/

[8] INFORMATION NETWORKS DIVISION HEWLETT-PACKARD. Netperf revision 2.1 Company February 15, 1996 http://www.netperf.org/netperf/NetperfPage.html

[9] JACOBSON, Van. Pathchart. LBL's Network Research Group. http://www.caida.org/tools/utilities/others/pathchar/

[10] John Ousterhout. Scripting: Higher-level programming for the 21st century.

[11] JUN, Jabgeun; PEDDABACHAGARI, Pushkin; SICHITIU, Mihail. "Theorical Maximun Throughput of IEEE 802.11 and its aplications".

[12] KAMERMAN, A.; ABEN, G. "Throughput performance of wireless LANs operating at 2.4 and 5 GHz," Personal, Indoor and Mobile Radio Communications, 2000, The 11th IEEE International Symposium on, (PIMRC 2000), Vol. 1, Pág. 190-195, 2000.

[13] KEVIN FALL , KANNAN VARADHAN , The *ns* Manual (formerly *ns* Notes and Documentation), The VINT Project A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, December 13, 2003.

[14] MAEDA, Y; TAKAYA, K; KUWABARA,N. "Experimental Investigation of Propoagation Characteristics of 2.4 GHz ISM-Band Gíreles LAN in Various Indoor Enviroments", IEICE Transactions on Communications, Vol. E82-B, No. 10, Oct 1999.

[15] MARC GREIS' Tutorial for the UCB/LBNL/VINT Network Simulator "ns"

[16] Measurement & Operations Analysis Team from the National Library for Applied Network Research (NLANR): Proyecto de recolección de datos, febrero 2001.http://moat.nlanr.net/Datacube/

[17] MERAT, Frank; LIBERATORE, Vincenzo. "An Analysis of Energy-Efficient Voice Over IP Communication in Wireless Networks", Case Western Reserve University, Marzo 2004

[18] NAVY General Purpose Electronic Test Equipment (GPETE) Program and MARINE CORPS Test Measurement and Diagnostic Equipment (TMDE) Programs.MGEN
http://tang.itd.nrl.navy.mil/5522/mgen/mgen_index.html

[19] NETIQ CORPORATION Chariot Copyright © 1993 - 2003 NetIQ and/or its suppliers, 3553 North First Street, San Jose, CA 95134, U.S.A. All rights reserved http://www.netiq.com/products/chr/default.asp

[20] NETIQ CORPORATION. QCheck. Copyright © 1993 - 2003 NetIQ and/or its suppliers, 3553 North First Street, San Jose, CA 95134, U.S.A. All rights reserved http://www.netiq.com/qcheck/default.asp

[21] PITTSBURGH SUPERCOMPUTING CENTER (PSC), CARNEGIE MELLON UNIVERSITY, UNIVERSITY OF PITTSBURGH. Treno URL: http://www.psc.edu/general/help/assistance.html. Revised: Agosto, 2002.

[22] PRASAD, A.R.; PRASAD, N.R; KAMERMAN, A.; MOELARD, H; EIKELENBOOM, A. "Indoor Wirereless LANs Deployment" Vehicular

Technology Conference Proceeding, 2000. IEEE 51st, (VTC 2000-Spring Tokio.), Vol 2, Pág. 1562-1566, 2000.

[23] RODRGUEZ H. FRANCISCO, "Manual MGEN V6 Multigenerator Toolset", Universidad CARLOS III de Madrid.

[24] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy,John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu,Haobo Yu, and Daniel Zappala. Improving simulation for network research. Technical Report 99-702b, University ofSouthern California, March 1999. (revised September 1999).

[25] SANDIA NATIONAL LABORATORIOS, U.S. DEPARTMENT OF ENERGY. Pchar "Copyright (c) 1995, 1996, 1997, 1998 The Regents of the University of California." http://packages.debian.org/stable/net/pchar

[26] SPURGEON, Charles. Ethernet : The definitive guide. Estados Unidos: O'Reilly & Associates, 2000. ISBN 1-56592-660-9.

[27] TANENBAUN, Andrew S. REDES DE COMPUTADORAS, Tercera Edición, Prentice Hall Hispanoamericana S.A., 1997.

[28] XYLOMENOS, G; POLYZOS, G. C " Internet Protocol Performance Ovewr Networks UIT Gíreles Links," IEEE Network, Vol. 13, Iss 4, Pág. 55-63, 1999.

ANEXO A. Network simulator

1. RESEÑA HISTÓRICA

Network Simulator (NS) se originó como un proyecto denominado "VINT[29]", llevado a cabo bajo la supervisión y colaboración de investigadores enfocados en UC BERKELEY, LBL, USC/ISI y XEROX PARC.

El proyecto fue soportado por la "*Defense Advanced Research Projects Agency"* (*DARPA*) en *LBL*. Dicho simulador es escrito en *C++* y utiliza *OTcl* como una interfaz de comandos y configuración, por otra parte su documentación está disponible en html[30].

2. GENERALIDADES NS-2

NS es un simulador de eventos de libre distribución que trabaja bajo el sistema operativo LINUX, el cual empieza como una variante del REAL Network Simulator en 1989 y ha evolucionado substancialmente durante los últimos años. En 1995 el desarrollo lo llevaba acabo DARPA a través del proyecto VINT de LBL, XEROX PARC, UCB y USC/ISI. Actualmente el desarrollo de NS lo lleva DARPA junto a SAMAN y otros. NS siempre ha contado con contribuciones de muchos otros desarrolladores, incluyendo código inalámbrico de los proyectos CMU Monach y UCB Daedelus y también de Sun Microsystems.

NS utiliza un lenguaje de SCRIPT llamado TCL[31] que permite ir generando el modelo. También dispone de una interfaz gráfica llamada NAM[32] que permite

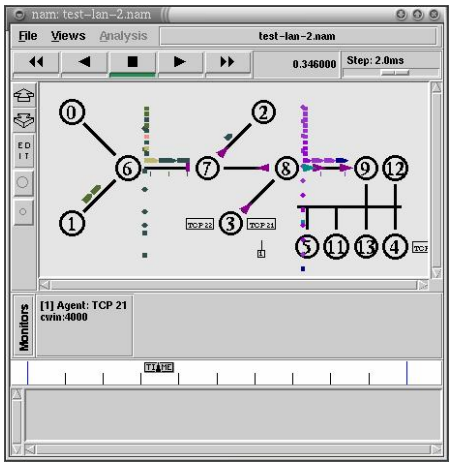
---

[29] Entre sus editores se encuentran KEVIN FALL (kfall@ee.lbl.gov) y KANNAN VARADHAN (kannan@catarina.usc.edu).
[30] Información adicional y formatos *pdf* Ver http://www.isi.edu/nsmam/ns/ns-documentation.
[31] TCL, Lenguaje de herramienta de comando (Tool Command Language), lenguaje de programación que junto con C++ compone el lenguaje de programación de NS.

visualizar las simulaciones e incluso crear y editar los modelos a simular. Ns también posee a XGRAPH[33] el cual ofrece la facilidad de visualizar gráficas sobre la línea de tiempo. NS es una gran herramienta que puede ayudar en muchos campos a la hora de realizar pruebas o generar nuevos tipos de redes.

En las siguientes gráficas se puede observar la interfaz gráfica NAM, la cual de una forma muy amigable al usuario muestra el funcionamiento de la simulación, ya sea redes cableadas (figura 56), o redes inalámbricas (figura 57).

Figura 56. Interfaz NAM para una red cableada.



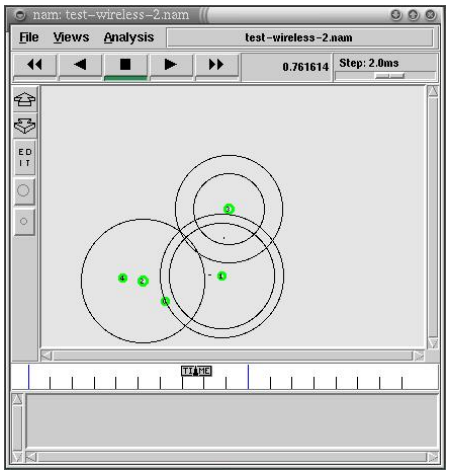
Tomado de *NS FUNCIONANDO EN RED HAT LINUX 9.0.*

NS (Network Simulator) actualmente en la versión 2 permite trabajar con Redes terrestres, inalámbricas y por satélite con varios algoritmos de enrutado (DV, LS, PIM-DM, PIM-SM, AODV, DSR). Además se pueden tener distintas fuentes de tráfico: Web, ftp, telnet, cbr, etc, y simular fallos como perdidas probabilísticas y deterministas, fallos en la conexión, etc, junto con las

[32] NAM (Network Animator), interfaz gráfica que permite la animación de la simulación.
[33] XGRAPH, interfaz que permite visualizar las funciones con respecto al tiempo.

distintas disciplinas de encolado (drop-tail, RED, FQ, SFQ, DRR, etc.) y QoS (calidad de servicio).

Figura 57. Interfaz NAM para una red inalámbrica.



Tomado de *NS FUNCIONANDO EN RED HAT LINUX 9.0.*

Con estas simulaciones es posible observar el flujo de paquetes, su encolado y posible descarte, monitoreando comportamientos del protocolo: comienzo lento de TCP, control de congestión, retransmisión rápida y recuperación, además de trabajar con movimiento de nodos en redes inalámbricas y dar informe de los sucesos más importantes y los estados del protocolo.

3. INTERFAZ AL INTÉRPRETE

Enlace Otcl. *Ns* es un simulador orientado a objetos, escrito en *C++*, con un interpretador *OTcl* como intermediario para el usuario. El simulador soporta una clase de jerarquía en *C++* (Llamada jerarquía compilada) y una clase de jerarquía similar dentro del interprete *OTcl* (llamada jerarquía interpretada).

Estas jerarquías se hallan en estrecha relación una a una; desde la perspectiva del usuario, se visualiza una correspondencia entre la jerarquía interpretada y la jerarquía compilada. La raíz de estos componentes es la clase *TclObject*.

Los usuarios crean un objeto simulador a través del intérprete, estos objetos son inicializados dentro de este y son estrechamente reflejados por un objeto correspondiente en la jerarquía compilada. La clase de jerarquía interpretada es automáticamente establecida a través de métodos definidos en la clase *TclClass*. Los objetos de usuario al instante se presentan a través de métodos definidos en la clase *TclObject*. Es posible observar otras jerarquías en el código *C++* y los *scripts OTcl*, las cuales no son utilizadas en la misma manera que *TclObject*.

4. CONCEPTO GENERAL

NS usa dos lenguajes debido a que el simulador posee dos tipos de actividades diferentes por realizar. De una parte se encuentran las simulaciones detalladas de protocolos que requieren sistemas de lenguajes de programación que puedan manejar eficientemente los bits, paquetes, cabeceras e implementar algoritmos que operen sobre varias series de datos. Para estas tareas la velocidad a la que el computador procesa una simulación, o sea que dure poco tiempo, es más importante que el tiempo que lleva editar una simulación, encontrar defectos, arreglar defectos, recompilar y volver a correr la simulación.

Por otro lado, una gran parte de investigación de la red involucra un poco la variación de parámetros y configuraciones, o la exploración rápida de un número de escenarios. En estos casos el tiempo de interacción (cambiar el modelo y correr de nuevo la simulación) posee mayor relevancia, ya que la configuración corre una vez (al inicio de la simulación), y el tiempo con que opera la simulación es de mínima trascendencia.

NS maneja ambas necesidades con dos lenguajes, *C++* y *OTcl*. *C++* es rápido para correr pero lento para cambiar, constituyéndose adecuado para la implementación detallada de protocolos. *OTcl* corre más lento pero puede ser cambiado rápido e interactivamente, haciéndose ideal para simulación de configuraciones, lo que conduce a que el presente proyecto se centre esencialmente en el manejo de lenguaje *OTcl*, pero no se deja de lado la programación en *C++* ya que habrá necesidad de cambiar y programar algunas aplicaciones para el manejo de datos.

NS (via *tclcl*) provee una unión para hacer que los objetos y variables aparezcan en ambos lenguajes[34].

Con el propósito de conocer el lenguaje que debe ser usado para cierto fin se recomienda:

1. Usar OTcl:
    - Para lograr configurar, arreglar y empaquetar una vez.
    - Cuando se quiera hacer lo que se desee manipulando objetos en C++.

2. Usar C++:
    - Si se desea hacer algo que requiera en un flujo de paquetes el procesamiento de cada uno de estos.
    - Si se tiene que cambiar el comportamiento de una clase C++ existente en la forma que fue proporcionada.

Por ejemplo los *Links* (Enlaces) son objetos *OTcl* que introducen retrasos, encolado y posibles módulos de pérdidas. Si el trabajo puede ser hecho con

[34] Para mas información acerca de la idea de escritura de lenguajes y partir lenguajes de programación vea el articulo Ousterhout's en IEEE Computer [10]. Para mas información acerca de partir niveles de programación para simulaciones de red, vea el NS paper.

estas características, es aceptable. Si por el contrario lo que se quiere es hacer algo mas profundo (Una disciplina especial de colas o modelos de pérdidas), entonces se necesita un nuevo objeto *C++*.

5. CÓDIGO GENERAL

Se utiliza el término Intérprete para hacer referencia al *OTcl*, el código para lograr una interfaz con el intérprete se localiza en el directorio *Tclcl*, el resto del código simulador se encuentra en el directorio *ns-2*[35].

Hay un número de clases definidas en *~tclcl/*. Se hace énfasis solo en las seis que son usadas en NS. La clase *Tcl* contiene los métodos que el código *C++* utiliza para acceder al intérprete, por otro lado el *TclObject* constituye la base para todos los objetos del simulador que son reflejados en la jerarquía compilada.

Además *TclClass* define la jerarquía de clase interpretada y los métodos para permitir al usuario iniciar *TclObjects*. *TclCommand* es empleada para definir comandos simples de interpretación global. La clase *EmbeddedTcl* contiene los métodos para cargar comandos incorporados de alto nivel que hacen las configuraciones de simulación fáciles. Por último *InstVar* contiene métodos con el fin de acceder a variables miembros *C++* como los ejemplares *OTcl*[36].

Luego de dar a conocer de manera general ciertas características y aspectos relevantes sobre las clases dedicadas al manejo de *NS*, se expone de forma detalla la profundización con respecto a este tema:

---

[35] Se utiliza la notación ~Tclcl/<Archvio> para referirse a un archivo particular en el directorio Tcl. Similarmente, se utiliza la notación ~ns/<Archivo> para referirse a un archivo particular en el directorio ns-2.
[36] Los procedimientos y funciones descritas pueden ser encontradas en ~Tclcl/Tcl.{cc, h}, ~Tclcl/Tcl2.cc, ~Tclcl/tcl-object.tcl, y ~Tclcl/tracedvar.{cc, h}. El archivo ~Tclcl/tcl2c++.c es usado en la construcción de NS.

5.1 La Clase Tcl. La clase *Tcl* (Class Tcl) encapsula el ejemplar actual del intérprete *OTcl* (OTcl Interpreter) y provee los métodos para acceder y comunicarse con este. El programador *NS* (Ns programmer) es escrito en C++, la clase *Tcl* proporciona métodos para los siguientes propósitos:

- Obtener una referencia del ejemplar *Tcl* (Tcl instance).
- Invocar procedimientos *OTcl* hacia el intérprete.
- Recuperar o devolver resultados al intérprete.
- Reportar situaciones de error y su salida en una manera uniforme.
- Almacenar y buscar objetos *Tcl* "*TclObjects*".
- Adquirir acceso directo en relación al intérprete.

Adicional a lo anterior se da la presencia de la función *Hash* dentro del intérprete, con base en la cual *Ns* almacena una referencia para cada objeto Tcl (*TclObject*) de la jerarquía compilada en una tabla *hash* lo que le permite un acceso rápido a los objetos y usar el nombre del *TclObject* como una llave para entrar, buscar o borrar en la tabla *hash.*

5.2 La clase TclObject.

Definición. La clase TclObject es la base fundamental para otras en la jerarquía compilada e interpretada. Cada objeto en dicha clase es creado por el usuario dentro del intérprete. Un objeto reflejado equivalente es creado en la jerarquía compilada. Los dos objetos están cercanamente asociados el uno al otro. La clase TclClass, descrita en la próxima sección, contiene los mecanismos que realizan este reflejo.

En lo posterior del documento elaborado, frecuentemente se hace alusión a un objeto como un *TclOject*. Por esto, se enuncia a un objeto particular el cual esta en la clase *TclObject* o en su derivada.

Al observar la necesidad de una explicación sobre si este objeto se ubica dentro del intérprete o existe la presencia de un objeto a nivel del código

compilado, se procede a su elaboración. En tal caso es posible manejar las abreviaciones "objeto interpretado" y "objeto compilado" para distinguir las dos.

Ejemplo de configuración de un TclObject. El siguiente ejemplo ilustra la configuración de un agente SRM (class Agent/SRM/Adaptive).

```
set srm [new Agent/SRM/Adaptive]
$srm set packetSize_ 1024
$srm traffic-source $s0
```

Por convención en *NS*, la clase Agent/SRM/Adaptive es derivada de Agent/SRM, que constituye una subclase de Agent, la cual representa una división de *TclOject*.

La correspondiente clase de jerarquía compilada es el agente ASRM, generado de SRMAgent, que representa un producto de Agent, resultado del *TclObject* respectivamente.

La primera línea del ejemplo expuesto con anterioridad muestra como un *TclObject* es creado o destruido, la línea siguiente configura una variable frontera y finalmente, se ilustra el objeto interpretado invocando un método *C*++ como si ellos fueran un procedimiento ejemplar.

Creando y Destruyendo TclObjects. El usuario puede crear un nuevo TclObject o eliminarlo, usando los procedimientos new{} y delete{}, estos son definidos en ~tclcl/tcl-object.tcl. Ellos pueden ser usados para crear y destruir objetos en todas las clases, incluyendo TclObjects (Las clases Simulator, Node, Link o rtObject, no son derivadas de TclObject, por tanto los objetos en ellas no son TclObject, sin embargo el Simulator, Node, Link o route Object pueden ser iniciados usando el procedimiento new en ns).

Usando *new{}* el usuario crea un *TclObject* interpretado. El interprete ejecutará el constructor para ese objeto, *init{}*, pasando a este algunos argumentos proporcionados por el usuario. NS es responsable de crear automáticamente el objeto compilado. El objeto reflejado obtenido es creado por la clase base constructora *TclObject*. Por consiguiente, el constructor para el nuevo T*clObject* debe llamar a la clase constructora primero. *New{}* devuelve un manejador para el objeto, ese entonces puede ser usado para futuras operaciones sobre ese objeto.

La operación de eliminación destruye el objeto interpretado y el correspondiente objeto reflejado.

El objeto destructor debe llamar a su clase pariente Destructora. El *TclObject* destructor invoca el procedimiento ejemplar *delete-shadow*, que solicita al método equivalente compilado para eliminar el objeto reflejado. El intérprete mismo demolerá el objeto interpretado.

Variables vinculantes. En muchos casos, el acceso a variables miembro compiladas es restringido por su código y el acceso a variables miembro interpretadas es igualmente limitado para abrir vía al código correspondiente; Sin embargo, es posible establecer vinculaciones bidireccionales puesto que ambas; la variable miembro interpretada y la variable miembro compilada acceden a los mismos datos, y cambiando el valor de cualquier variable se da la transformación del valor de la correspondiente variable pareja a el mismo valor.

La vinculación es establecida por el constructor compilado cuando ese objeto es inicializado; Este es automáticamente accesible por el objeto interpretado como una variable ejemplar. *Ns* soporta cinco tipos diferentes de datos: reales, variables estimadas de ancho de banda, variables estimadas de tiempo,

enteros y booleanos. La sintaxis de cómo estos valores pueden ser especificados en *OTcl* es diferente para cada tipo de variable.

- Variables estimadas reales y enteras son especificadas en la forma "normal". Por ejemplo:

$object set realvar 1.2e3
$object set intvar 12

- El ancho de banda es especificado como un valor real, opcionalmente sufijo por una 'k' o 'K' lo que significa kilo-cantidades, o 'm' o 'M' lo que significa Mega-cantidades. Un sufijo opcional de 'B' indica que la cantidad expresada esta en Bytes por segundo. El ancho de banda por defecto esta expresado en bits por segundo. Por ejemplo, todos los siguientes son equivalentes:

$object set bwvar 1.5m
$object set bwvar 1.5mb
$object set bwvar 1500k
$object set bwvar 1500kb
$object set bwvar .1875MB
$object set bwvar 187.5kB
$object set bwvar 1.5e6

- El tiempo es especificado como un valor real, opcionalmente sufijo por una 'm' para expresar el tiempo en mili-segundos, 'n' para expresar el tiempo en nano-segundos, o 'p' para expresar el tiempo en pico-segundos. El tiempo por defecto esta expresado en segundos. Por ejemplo, todos los siguientes son equivalentes:

$object set timevar 1500m

$object set timevar 1.5
$object set timevar 1.5e9n
$object set timevar 1500e9p

Notese que se puede con seguridad agregar una *s* para reflejar el tiempo en unidades de segundos. *Ns* ignorará cualquier otra cosa con especificación numérica real, o seguido con 'm', 'n' o 'p'.

- Las cantidades booleanas pueden ser expresadas como un entero o como una 'T' o 't' para indicar verdadero. Caracteres posteriores luego de la primera letra son ignorados. Si el valor no es un entero, ni un valor verdadero, entonces es asumido como falso. Por ejemplo:

$object set boolvar t *;# ajustado a verdadero*
$object set boolvar true
$object set boolvar 1 *;# o algún valor diferente de cero*

$object set boolvar false *;# ajustado a falso*
$object set boolvar junk
$object set boolvar 0

Por ejemplo, si se definen las siguientes variables del agente *ASRMAgent* como:

Agent/SRM/Adaptive set pdistance_ 15.0
Agent/SRM set pdistance_ 10.0
Agent/SRM set lastSent_ 8.345m
Agent set ctrlLimit_ 1.44M
Agent/SRM/Adaptive set running_ f

Por lo tanto, cada nuevo objeto *Agent/SRM/Adaptive* tendrá el valor *pdistance_* establecido en *15.0*, *lastSent_* lo posee en *8.345m* desde la configuración de la variable, *ctrlLimit_* lo tiene en *1.44M* y *running está* determinado en *falso*.
Se debe conocer que *ns* garantiza que el valor actual de la variable tanto en el objeto interpretado como en el objeto compilado y serán idénticos todo el tiempo, sin embargo si se da la existencia de métodos y otras variables del objeto compilado que cambien el valor de esta variable, ellos deben ser explícitamente invocados cuando el valor de esta variable sea cambiado.

5.3 La Clase TclClass. Esta clase compilada (*class TclClass*) es una clase virtual. Las clases derivadas de esta clase base proveen dos funciones:

- Construir la clase de jerarquía interpretada para reflejar la clase de jerarquía compilada.
-Proveer métodos para inicializar nuevos *TclObject*.

Cada una de las clases derivadas es asociada con una clase compilada particular en la clase de jerarquía compilada y puede inicializar nuevos objetos en la clase asociada.

5.4 La Clase TclCommand. Esta clase (class TclCommand) provee los mecanismos para que *NS* exporte comandos simples al intérprete que pueden ser ejecutados dentro de un contexto global por el intérprete. Hay dos funciones definidas en *~ns/misc.cc*: *ns-random* y *ns-version*.

La clase *VersionCommand* define el comando *ns-version*. Este no toma argumentos y devuelve la versión actual de *ns*:

```
% ns-version ;# Devuelve la versión actual de ns
2.0a12
```

La clase *RandomCommand* define el comando *ns-random*. Sin argumentos, *ns-random* devuelve un entero uniformemente distribuido en el intervalo [0, 2^{31}-1].

Cuando se especifica un argumento, se toma éste para decidir, si este argumento es cero "*0*" el comando usa un valor heurístico, pero si este argumento es diferente de cero "*0*" el generador de números aleatorios lo asume como el número que debe generar.

% ns-random ;# *Genera un número aleatorio*
2078917053
% ns-random 0 ;# *Se configura heurísticamente*
858190129
% ns-random 23786 ;#*Se configura un valor específico*
23786

5.5 La Clase Embeddedtcl. *Ns* permite el desarrollo de funcionalidades ya sea a través del código compilado o el código interpretado que es evaluado en el inicio, por ejemplo el script *~tclcl/tcl-object.tcl* o los scripts en *~ns/tcl/lib*. La carga y evaluación de los scripts se realiza mediante objetos en la clase *EmbeddedTcl*. La forma más fácil de extender *NS* es agregar código OTcl a *~tclcl/tcl-object.tcl* o a través de scripts en el directorio *~ns/tcl/lib*, debe notarse que en el segundo caso *NS* origina *~ns/tcl/lib/ns-lib.tcl* automáticamente y por consiguiente el programador debe agregar un par de líneas a este archivo para que sus scripts sean también originados por *NS* automáticamente al inicio.

Tres puntos de relevancia en cuanto al código *EmbeddedTcl* son los descritos a continuación: Primero, si el código tiene un error que es encontrado durante la evaluación, entonces *NS* no correrá. Segundo, el usuario puede explícitamente sobrescribir algo del código en el script, se puede re-obtener el script de

entrada luego de hacer sus propios cambios. Finalmente, luego de agregar los scripts a *~ns/tcl/lib/ns-lib.tcl* y cada vez que el usuario cambie el script se debe recompilar *NS* para que los cambios tengan efecto. En muchos casos el usuario puede originar su script para sobrescribir el código incrustado (Embedded).

5.6 Clase Instvar. Esta clase define los métodos y mecanismos para vincular una variable miembro *C++* en el objeto reflejado compilado con una variable ejemplar específica *OTcl* en el objeto interpretado equivalente. La vinculación es establecida tal que el valor de la variable puede ser configurado o accedido de cualquier forma, ya sea dentro del intérprete o dentro del código compilado todo el tiempo.

Hay cinco variables clase ejemplares: *InstVarReal*, *InstVarTime*, *InstVarBandwidth*, *InstVarInt* y *InstVarBool* que corresponden a vinculaciones para variables de valores real, tiempo, ancho de banda, enteros y booleanos respectivamente.

ANEXO B. Comandos network simulator

A continuación se explica una breve descripción de algunos comandos generales en *NS*[37]:

- Si se quiere agregar un comentario dentro del script para identificar un conjunto de líneas se utiliza el comando:

#Texto que se quiere introducir

- El Preámbulo para iniciar una simulación es:

set ns_ [new Simulator]
Este comando crea un ejemplar del objeto simulador.

- Es posible obtener la noción del tiempo actual del programador por medio de:

set now [$ns_ now]

Debido a que el programador mantiene el rastro del tiempo en una simulación.

- Para parar o pausar el programador es viable:

$ns_ halt

- Con el fin de iniciar el programador se utiliza:

[37] Tomado de Kevin Fall et. al [27] y Marc Greis' et. al [28].

$ns_ run

- Al querer programar un evento <*event*> (El cual es normalmente una pieza de código) para que sea ejecutado en el tiempo especificado es preciso apoyarse en <*time*>:

$ns_ at <time> <event>

- Llevar a cabo la cancelación de un evento es posible por medio de:

$ns_ cancel <event>

En efecto, el evento es removido de la lista del programador para ejecutar los eventos.

- La creación de un objeto de trazo posee como mecanismo o comando a:

$ns_ create-trace <type> <file> <src> <dst> <optional arg: op>

Este crea un objeto de trazo del tipo <*type*> entre los objetos fuente <*src*> y el destino <*dst*> y adjunta el objeto de trazo al archivo <*file*> para escribir la traza de salida. Si *op* es definido como "*NAM*", este crea archivos de trazo *NAM*, de otro lado si *op* no es definido, los archivos de trazo NS son creados como su valor por defecto.

- El lanzamiento de los objetos de trazo escritos en el buffer se realiza por medio de:

$ns_ flush-trace

- Volcar o deshacerse de información como nodos, componentes de nodo, enlaces, etc. Creados para una simulación dada, es acertado a través de:

$ns_ gen-map

- Para decirle a *ns* que ejecute un evento en el tiempo actual se usa:

$ns_ at-now <args>

Este comando es en efecto como el comando "$ns_ at $now $args"

- Con el propósito de especificarle a *NS* el tipo de programador (Scheduler) usado para una simulación se recurre a:

$ns_ use-scheduler <type>

Los diferentes tipos de programador (Scheduler) disponibles son: *List*, *Calendar*, *Heap* y *RealTime*. El tipo por defecto es *Calendar*.

- La programación encaminada a que un evento sea ejecutado luego de un lapso de tiempo se logra por medio de:

$ns_ after <delay> <event>
Donde *<event>* es el evento y *<delay>* el lapso de tiempo.

- En los propósitos de supresión de fallos de memorias (Debugging) es indispensable contar con:

$ns_ clearMemTrace

- Este comando es capaz de devolver un verdadero, si el simulador ha ejecutado y una afirmación falsa si esto no es posible:

$ns_ is-started

- Este comando se utiliza para volcar o deshacerse de eventos en cola en el programador mientras el programador está parado o interrumpido:

$ns_ dumpq

- Con el objeto de configurar el formato de paquete del simulador es fundamental utilizar:

$ns_ create_packetformat

- Al desear crear y obtener un nodo ejemplar lo más factible es recurrir a:

$ns_ node [<hier_addr>]

Si <*hier_addr*> es dado, se asigna al nodo el direccionamiento <*hier_addr*>. <*hier_addr*> debe ser usado solamente cuando el direccionamiento jerárquico es activado ya sea por medio de *set-address-format hierarchical{}* o *node-config -addressType hierarchical{}*.

- Un comando útil que permite configurar los nodos lo constituye:

$ns_ node-config -<config-parameter> <optional-val>

Los diferentes parámetros de configuración <*config-parameter*> son, tipo de direccionamiento, diferentes tipos de componentes de stack de red, si, tracing o rastreo será encendido o no, la bandera de mobileIP está encendida o no, el modelo de energía será utilizado o no, etc. Una opción de reset puede ser

usada para ajustar la configuración del nodo a sus valores por defecto el cual crea un nodo simple.

- Para obtener el número de identificación *id* del nodo es viable abrirse camino con:

$node id

- Obtener la dirección del nodo es posible a través de:

$node node-addr

En caso de direccionamiento plano, la dirección del nodo es la misma que su número *id*. A nivel de direccionamiento jerárquico, la dirección de nodo es devuelta en forma de string.

- Cuando la finalidad consiste en iniciar todos los agentes adjuntos a un nodo es viable recurrir a :

$node reset

- Si se pretende obtener el manejador del agente en el puerto especificado acertado es valerse de:

$node agent <port_num>

Si no es encontrado un agente en el puerto especificado, un string nulo devuelto.

- Obtener el punto de entrada para el nodo es posible a través de:

$node entry

Adicionalmente se tiene que este es el primer objeto que maneja paquetes recibidos en este nodo.

- Para adjuntar un agente a un nodo:

$node attach <agent> <optional:port_num>

Este comando adjunta el agente <agent> a este nodo. Si no se especifica un número de puerto <optional:port_num>, el nodo destina un número de puerto y vincula el agente a este. Así, una vez el agente es adjuntado, recibe paquetes destinados para este host (nodo) y puerto.

- Con el objetivo de separar un agente de un nodo hay que dirigirse hacia:

$node detach <agent> <null_agent>

Este comando es el dual del comando de adjuntar descrito anteriormente. Se encarga de separar el agente de este nodo e instalar un agente Null al puerto que el agente fue adjuntando.

Esto es hecho para manejar el tránsito de paquetes que pueden ser destinados al agente separado. Los paquetes que van destinados a este agente son recibidos por el agente Null.

- En vista de obtener una lista de nodos vecinos es posible recurrir a :

$node neighbors

Además de los comandos descritos anteriormente existen otros con diversas funciones e igual utilidad:

- *$node add-neighbor <neighbor_node>:* Permite agregar un nodo a la lista de vecinos mantenida por el nodo.

- *$node add-route <destination_id> <target>:* Sirve para adicionar una ruta de enrutamiento unicast a un nodo.

Este es usado en enrutamiento unicast para poblar el clasificador. <*target*> es un objeto *Tcl*, que puede ser la entrada de un de multiplexor (puerto de multiplexor en el nodo) en caso de que <*destination_id*> sea el mismo que su *id* de nodo. De otro lado este es usualmente la cabeza del enlace para ese destino.

- *$node alloc-port <null_agent>:* Colabora en la obtención del número siguiente de puerto disponible.

- *$node incr-rtgtable-size:* Aumenta el tamaño de la tabla de enrutamiento cada vez que una entrada de enrutamiento es agregada a los clasificadores.

La variable ejemplar *rtsize_* es usada para mantener el rastro del tamaño de la tabla de enrutamiento en cada nodo.

- La creación de un archivo de trazos NAM se hace por medio de:

set nf [open SALIDA.NAM w]
$ns NAMtrace-all $nf

Donde *SALIDA.NAM* es el nombre del archivo que contiene los datos utilizados por *NAM* para graficar. La *W* indica escribir en el archivo.

- En cuanto a la elaboración de un archivo de traza XGRAPH[38]:

set f [open SALIDA.tr w]
$ns trace-all $f

Donde *SALIDA.tr* es el nombre del archivo que contiene los datos utilizados por *XGRAPH* para graficar. La *W* indica escribir en el archivo.

- La creación de un nodo se realiza por parte de:

set n0 [$ns node]

Donde *n0* es el nombre del nodo.

- El siguiente comando se utiliza en la creación de un enlace de datos:

$ns Tipo-de-enlace $n0 $n1 Bw Retraso Tipo-de-cola

Donde se puede seleccionar varios *Tipo-de-enlace*, como por ejemplo simplex, duplex o full-duplex; *$n0* y *$n1* son los nodos que están conectados por medio de este enlace; *Bw* define el ancho de banda del enlace en Bits por segundo; *Retraso* indica tiempo que tardan los datos en atravesar dicho enlace en segundos y finalmente *Tipo-de-cola* es la cola del enlace como puede ser del tipo DropTail, Red o SFQ (Stochastic fair queueing).

[38] "*XGRAPH*" (Permite visualizar gráficos de flujo de datos en ns).

- La creación de un agente y su adjudicación a un nodo se lleva a cabo por:

set Nombre[new Agent/Tipo]
$ns attach-agent $n0 $Nombre

Donde *Nombre* es el nombre que se desea colocar al agente, *Tipo* representa al agente que se desea como por ejemplo TCP, UDP, TCPSINK, NULL o LOSSMONITOR, los cuales se diferencian en que TCP es confiable y orientado a conexión, lo cual indica el envío de acuses de recibo y la utilización de ventanas deslizantes, mientras UDP no es orientado a conexión confiable, lo que indica que transmite paquetes sin importar si estos se perdieron o llegaron bien a su destino.

TCPSINK es un receptor de tráfico el cual envía acuses de recibo (Especial para TCP) mientras NULL no lo hace (Especial para UDP), el agente NULL se utiliza como un receptor de tráfico para paquetes perdidos o un destino para paquetes que no son contados o grabados, LOSSMONITOR se utiliza para calcular el ancho de banda ya que este agente mantiene un registro del número de paquetes que ha recibido, finalmente *$no* es el nombre del nodo al cual se desea adjuntar este agente.

- Para crear un generador de tráfico y adjuntarlo a un agente se dirige la atención hacia:

set Nombre [new Application/Tipo]
$Nombre attach-agent $Agente

Donde *Nombre* es el nombre que se desea asignar al generador de tráfico, *Tipo* es el tipo de tráfico como por ejemplo FTP (Protocolo de Transferencia de Archivos) o CBR (Rata de Bit Constante) y finalmente *Agente* es el nombre del agente al cual se quiere adjuntar el generador de tráfico.

- Si se requiere crear un objeto simulador se escribe el comando:

set ns [new Simulator]

- Abrir un archivo y escribir los datos para que sean utilizados por el *NAM* se usa el comando:

set nf [open out.NAM w]
$ns NAMtrace-all $nf

La primera línea abre el archivo *out.NAM* para escribir y darle este al archivo manejador *nf*. En la segunda línea se le dice al objeto simulador que lo que esta creado arriba se utiliza para crear el archivo relacionado con el *NAM*.

- Para crear un procedimiento de *finish* que cierre los archivos de trazo y gráficos e inicie *NAM* y *XGRAPH* se utiliza el comando:

```
proc finish {} {
global ns nf f

#Se cierran los archivos de salida
$ns flush-trace
close $nf
close $f

#Se llama  a NAM y XGRAPH para que desplieguen los
#resutados
exec NAM out.NAM & exec XGRAPH out0.tr -geometry 800x400 &
exit 0
}
```

Se puede observar que al crear el procedimiento de *finish* se tienen que llamar las variables globales y los archivos manejadores donde se guardan los datos de los archivos de salida de trazos y gráficos, en este ejemplo son las variables *ns*, *nf* las que manejan la parte de *NAM* y *f* para la parte de XGRAPH.

Luego de esto se tiene que cerrar los archivos mencionados anteriormente para ejecutar NAM y XGRAPH, en este ejemplo se cierra el archivo de trazo NAM llamado *nf* y el archivo de trazo XGRAPH llamado *f*.

Finalmente se ejecutan los programas NAM y XGRAPH para desplegar los resultados, donde *out.nam* y *out0.tr* son los correspondientes archivos NAM y XGRAPH de salida, es de notar que cuando se ejecuta XGRAPH hay que definir el tamaño de la ventana, preferiblemente de la misma manera que esta configurado en el computador, en este ejemplo se deja definida en un tamaño de ventana de 800*600 píxeles.

- Para indicarle al simulador que ejecute el procedimiento de *finish* luego de por ejemplo 5 segundos de simulación se escribe el comando:

$ns at 5.0 "finish"

- Para llevar a cabo la iniciación de una simulación:

$ns run

- Con el propósito de crear un agente *UDP* y adjuntarlo a un correspondiente nodo (En este caso el nodo *n0*), ya que en *NS* el tráfico siempre debe ser enviado de un agente a otro, se invoca a:

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

- Desarrollar la creación de una fuente de tráfico *CBR* (Rata de bit constante) y adjuntarla a el agente *UDP* (En este caso el agente *UDP0*) es posible por medio de:

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

Donde en este ejemplo se ajusta el tamaño del paquete en 500 bytes y cada paquete será enviado cada 0.005 segundos, entonces se tiene 1/0.005= 200 paquetes enviados por segundo, además el tamaño de paquete de 500 bytes es igual a 500*8=4.000 bits=1 paquete, lo que representa un ancho de banda de:

*BW=200paquetes*segundo*4.000bits/paquete=800.000bits*segundo=0.8 Mbps.*

- El agente *NULL* actúa como un receptor de tráfico, su creación y asociación al nodo *n1* se realiza con:

set null0 [new Agent/Null]
$ns attach-agent $n1 $null0

- Para conectar dos agentes (Por ejemplo el agente *UDP*0 con el agente *NULL0*) se utiliza el comando:

$ns connect $udp0 $null0

- Programar una agente (En este caso el *CBR0*) para que envié (En este caso en 0.5 segundos) y pare de enviar datos (En este caso en 4.5 segundos) se hace a través del comando:

$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

- Organizar y darle localización a los trazados de los enlaces en NAM de modo que se vean mas organizados, se manejan por ejemplo los comandos:

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

Donde en la primera línea se le expresa a *NS* que el enlace dúplex gráficamente entre el nodo *n0* y *n1* se va a localizar orientado a la derecha y hacia abajo lo que se traduce en una diagonal descendiente, la segunda línea quiere decir que el enlace entre el nodo *n1* y *n2* esta orientado a la derecha y arriba lo que se traduce en una diagonal ascendente y finalmente la tercera línea expresa que el enlace entre el nodo *n2* y *n3* esta orientado hacia la derecha lo que se traduce una línea horizontal.

Diferentes opciones pueden ser utilizadas para las orientaciones de un enlace como por ejemplo, derecha, izquierda, arriba, abajo y combinaciones de estas orientaciones (Right, left, up, down).

- Llevar a cabo la marcación e identificación de flujos de datos para por ejemplo asignarle colores al tráfico para poder distinguir un tipo de paquete de otro (En este ejemplo se identifican dos tipos de paquetes de dos agentes *UDP*) es posible con el comando:

$udp0 set class_ 1
$udp1 set class_ 2

Donde por ejemplo la *clase 1* identifica al flujo de datos de *UDP0* y la *clase 2* identifica al flujo de datos de *UDP2*, debe tenerse en cuenta que esto es posible con los agentes mas no con el tipo de tráfico.

- Para identificar con colores cada tipo de tráfico, por ejemplo los tipos de tráfico anteriores (Uno rojo y uno azul) se utiliza el comando:

$ns color 1 Blue
$ns color 2 Red

Donde a la *clase 1* se le asigna el color azul y a la *clase 2* el color rojo, debe tenerse en cuenta que estas líneas de comando hay que preferiblemente ubicarlas al principio, luego de crear el objeto simulador.

- Si se proyecta monitorizar la cola de un enlace entre dos nodos y poder verlo gráficamente en NAM se utiliza el comando:

$ns duplex-link-op $n2 $n3 queuePos 0.5

Donde *n2* y *n3* son los nodos que están conectados por medio de este enlace y el tiempo *0.5* es el intervalo de tiempo en el cual se monitoriza la cola del enlace, se debe observar que el gráfico de paquetes en cola y pérdidas de paquetes se encontrará en el primer nodo que se coloque en el comando.

- Es posible utilizar comandos de repetición *for* o lazos *for* para facilitar la creación por ejemplo de varios nodos o enlaces en un solo comando, por ejemplo en el siguiente comando se crean 8 nodos:

for {set i 0} {$i < 8} {incr i} {
set n($i) [$ns node]
}

Se puede entender fácilmente la primera línea de este comando de repetición *for* ya que tiene un aspecto muy similar a cualquier comando de programación, indica que para *i* desde *0* hasta *i* menor que *8* tome incrementos de *1*, la segunda línea crea los nodos *n (i)*, entonces los nodos van a quedar numerados de *0* a *7* en este ejemplo.

- Al crear enlaces como se muestra a continuación donde por ejemplo se conectan los *8* nodos originados anteriormente de modo que surge una red en anillo, donde el nodo *0* se conecta al nodo *1*, el *1* con el *2* y así sucesivamente, cerrando el anillo con la conexión entre el nodo *7* con el nodo *0,* es posible la necesidad de usar:

for {set i 0} {$i < 8} {incr i} {
$ns duplex-link $n($i) $n([expr ($i+1)%8]) 1Mb 10ms DropTail
}

Donde se crean los *8* enlaces *duplex* de ancho de banda *1Mbps* y retardo de *10ms* con tipo de cola *Droptail*, en la segunda línea el comando $*n([expr ($i+1)%8])* pide la evaluación la expresión *($i+1)%8,* lo que quiere decir que en cada ciclo se suma uno a el valor actual de *i* y el módulo operador *%8* permite conectar el último nodo o sea el *7* con el nodo *0* para cerrar el anillo.

- *Ns* adicionalmente permite introducir caídas o fallos del enlace al realizar una simulación, por ejemplo si se pretende que un enlace se caiga en un determinado tiempo y vuelva a subirse en un tiempo específico para observar que sucede o como se enruta el tráfico, se puede utilizar el siguiente comando como se muestra a continuación:

$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)

Donde en la primera línea se le ordena al simulador que en el tiempo igual a *1* segundo se caiga el enlace entre el nodo *1* y el nodo *2*, y en la segunda línea se sube este mismo enlace en el tiempo igual a *2* segundos.

- Es viable implementar en NS distintos protocolos de enrutamiento para que *NS* se encargue de enrutar los paquetes, el siguiente comando permite la selección de un protocolo de enrutamiento unicast:

$ns rtproto protocolo $n1 $n2 $n3

Donde *protocolo* es el protocolo de enrutamiento que se va a utilizar en la simulación y *$n1 $n2 $n3…* constituyen los nodos a los cuales se les aplica el protocolo de enrutamiento elegido ya que se pueden utilizar diferentes protocolos de enrutamiento en un escenario de simulación, pero al usar el mismo protocolo de enrutamiento en el escenario de simulación no es necesario colocar los nodos, o sea se deja el comando solo hasta *protocolo ($ns rtproto protocolo)* y NS automáticamente comprende que el protocolo de enrutamiento es el mismo para todo el escenario.

Los protocolos de enrutamiento unicast acertados para la utilización con este comando son: *Static* el cual activa el enrutamiento estático, este es el enrutamiento por defecto, o sea que si no se especifica el protocolo de enrutamiento se supondrá que es estático; *Session* activa el enrutamiento por sesión, *DV* activa el enrutamiento por vector distancia y *LS* activa el enrutamiento por estado del enlace.

- Un comando que puede agregar fuentes y generadores de tráfico a los nodos fácilmente cuando se exige crear varios generadores y

adjuntarlos a los nodos puede ser como el que se muestra a continuación en donde se genera un procedimiento permisible a la creación varios agentes y fuentes de tráfico, también adjunta los agentes a los nodos y las fuentes de tráfico a los agentes, y finalmente conecta los agentes que se van a comunicar.

En este ejemplo se crean agentes fuente *UDP* además de fuentes generadoras de tráfico del tipo *EXPONENCIAL*; los agentes se adjuntan a los nodos los cuales deben ser creados con anterioridad, para luego adjuntar las fuentes de tráfico a los agentes y finalmente conectar los agentes que se van a comunicar, en este caso los agentes *UDP* con los agentes *SINK* que deben ser creados con anterioridad.

```
proc attach-expoo-traffic { node sink size burst idle rate } {
#Para obtener  un ejemplar del simulador
set ns [Simulator instance]
#Para crear  un agente UDP y  adjuntarlo al nodo
set source [new Agent/UDP]
$ns attach-agent $node $source
#Crear una fuente de tráfico exponencial y
#Ajustar sus parámetros de configuración
set traffic [new Application/Traffic/Exponential]
$traffic set packet-size $size
$traffic set burst-time $burst
$traffic set idle-time $idle
$traffic set rate $rate
#Se adjunta  la fuente de tráfico al generador o
#agente de tráfico
$traffic attach-agent $source
#Se conectan  los agentes que se van a comunicar, en
#este caso el agente UDP con el agente que recibe
```

#el tráfico, en este caso un agente SINK
#previamente creado
$ns connect $source $sink
return $traffic
}

Este toma 6 argumentos: un nodo (*node*), un trafico *sink* previamente creado (*sink*) que recibe el tráfico, el tamaño del paquete para la fuente de tráfico (*size*), los tiempos de envío (*burst*) y de no envío (*idle*) para la distribución exponencial y la tasa pico (*rate*).

Como se logra observar en el ejemplo el procedimiento crea un agente fuente de tráfico y adjunta este al nodo, luego se crea un objeto generador de tráfico del tipo exponencial, se ajustan sus parámetros de configuración y se adjunta al agente fuente de tráfico, finalmente el agente fuente y el agente destino (*SINK*) son conectados.

El procedimiento entrega un manejador que puede ser utilizado fácilmente como se muestra en comandos a continuación. Este procedimiento muestra cómo se pueden agilizar tareas como adjuntar una fuente de tráfico a varios nodos.

- Este comando que permite complementar al anteriormente utilizado, o sea por medio de cual se crean los agentes generadores de tráfico utilizando los parámetros definidos anteriormente; se muestra a continuación la formación de 3 agentes generadores de tráfico:

set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]
set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]
set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]

Es viable usar un procedimiento para adjuntar las fuentes de tráfico con diferentes tasas pico a los nodos *n0*, *n1* y *n2* además de conectar estos a los tres agentes receptores de tráfico o agentes *sink* como son *sink0*, *sink1* y *sink2* los cuales deben haber sido creados anteriormente, se observa que se define un tamaño de paquete de *200 bytes*, un tiempo de envió de datos de *2 segundos*, un tiempo de no envió de datos de *1 segundo* y las ratas pico en *100k*, *200k* y *300k* respectivamente para cada nodo.

- Para crear un procedimiento que permita escribir o grabar los datos que se pretenden graficar con XGRAPH en los archivos de salida, se utiliza un procedimiento como el que se muestra a continuación donde se escriben datos en los archivos *f0* y *f1* previamente creados y estos corresponden a la medida del ancho de banda en *Mbps* de los enlaces teniendo en cuenta los bytes recibidos por los agentes *sink* previamente creados, este procedimiento se configura para que se repita cada *0.5 segundos* y es un buen ejemplo de explicación de cálculo de ancho de banda de enlaces:

```
proc record {} {
global sink0 sink1 f0 f1
#SE obtiene  un ejemplar del simulador
set ns [Simulator instance]
#Se configura el tiempo luego del cual el
#procedimiento debe ser llamado nuevamente.
set time 0.5
#Se calcula cuantos bytes han sido recibidos por
#los agentes sink cada 0.5
set bw0 [$sink0 set bytes_]
set bw1 [$sink1 set bytes_]
#Se obtiene el tiempo actual de simulación
set now [$ns now]
```

#Se calcula el ancho de banda en Mbps y
#Se escribe este valor en los archivos de salida
#junto con el tiempo actual de simulación.
*puts $f0 "$now [expr $bw0/$time*8/1000000]"*
*puts $f1 "$now [expr $bw1/$time*8/1000000]"*
#Se reinician los valores de bytes_ en los agentes
#de trafico sink para volver a grabar pasados los
#0.5 segundos
$sink0 set bytes_ 0
$sink1 set bytes_ 0
#Se reprograma el procedimiento
$ns at [expr $now+$time] "record"
}

El procedimiento lee el número de bytes recibidos en los receptores de tráfico, en este caso los 2 agentes *sink*, son enviados desde los correspondientes agentes fuente previamente creados, se debe tener en cuenta que al principio del comando se colocan los nombres de los agentes que reciben el tráfico, seguido de los archivos de salida en orden como se desee guardar los datos, en este ejemplo el archivo de salida *f0* se relaciona con *sink0* y *sink1* con *f1*.

Se debe crear un ejemplar del simulador y configurar el tiempo en el cual el procedimiento será llamado o repetido, en este caso cada *0.5 segundos*, seguido de esto se procede a configurar los archivos que van a almacenar el numero de bytes que reciben los agentes *sink*, llamados en este ejemplo *bw0* y *bw1*.

Luego se obtiene el tiempo actual de simulación para crear el conjunto de puntos de tiempo contra ancho de banda y se almacena este conjunto de datos en los archivos *f0* y *f1*, es indispensable observar que el conjunto de datos se graba como (*$now, [expr $bw0/$time*8/1000000]*), donde el primer valor

corresponde al tiempo actual de simulación y el segundo corresponde a evaluar la expresión *$bw0/$time*8/1000000* donde *$bw0* corresponde al número de bytes almacenados en la variable *bw0* previamente creada, *$time* es la variable tiempo configurada en *0.5 segundos* la cual fue previamente creada, el *8* significa que se pasan los bytes almacenados cada *0.5 segundos* a bits y finalmente el *1000000* sirve para expresar las cantidades en Megas.

Analizando lo anterior es posible observar que esta operación calcula el ancho de banda ya que cada 0.5 segundos se calcula el número de *Mbps* recibidos y se relaciona con un punto de tiempo; para hacer esto se debe reiniciar los valores almacenados en las variables *sink* cada vez que sea llamado el procedimiento. Finalmente se reprograma el procedimiento cada *0.5 segundos* con la expresión *expr $now+$time* la cual reprograma el procedimiento de grabado cada *0.5 segundos*.

- Para decirle a *NS* que inicie un procedimiento de grabado de datos como el que realizó anteriormente se utiliza el comando:

$ns at tiempo "record"

Donde *tiempo* el es punto numérico de tiempo en el cual se pretende comenzar a almacenar datos.

ANEXO C. Componentes de ns y presentación de un script de simulación

Para realizar una simulación de red inalámbrica en *NS* se deben tener en cuenta aspectos como la definición del escenario, topología de red, pila de protocolos y parámetros específicos de cada protocolo, caracterización de antenas, tipo de modelo de radio propagación, numero de nodos y posiciones dentro del escenario, además especificar si tendrán movimiento aleatorio o estarán estáticos, definición de agentes y protocolos.

1. INICIACIÓN DEL SIMULADOR

El simulador es descrito por la clase simuladora *Tcl class;* esta provee una serie de interfaces para configurar una simulación y escoger el tipo de evento a programar usado para manejar la simulación. Un script de simulación generalmente comienza creando un ejemplar de esta clase y llamando varios métodos para crear nodos, topologías y configurar otros aspectos de simulación.

2. NODOS Y ENVÍO DE PAQUETES

2.1 BASES DE NODO. Para realizar una simulación se deben crear los nodos, aquellos que representan los dispositivos transmisores y receptores de paquetes, además, a estos se asignan los agentes, los cuales son los encargados de asignar los protocolos a utilizar como por ejemplo TCP, UDP o un agente de recepción de tráfico como SINK. Finalmente a cada agente se le especifica el tipo de tráfico que va a generar, como por ejemplo CBR, FTP, etc.

La primitiva base para crear un nodo es:

```
set ns [new Simulator]
$ns node
```

Este nodo pertenece a la clase *OTcl*, sin embargo varios de los componentes del nodo son *TclObjects*. La estructura típica de un nodo unicast consiste de dos *TclObjects*: Una dirección del clasificador (*classifer_*) y un puerto clasificador (*dmux_*). La función de estos clasificadores es distribuir paquetes entrantes a los agentes correctos o enlaces de salida.

Todos los nodos contienen al menos los siguientes componentes:

- Una dirección o *id_*, monotónicamente incrementándose en 1 (Desde su valor inicial 0).
- Una lista de vecinos (neighbor_).
- Una lista de agentes (agent_).
- Un identificador de tipo de nodo (nodetype_).
- Un módulo de enrutamiento.

Los nodos en *NS* son construidos por defecto para simulaciones *unicast*, si se quiere activar simulaciones *multicast* se debe utilizar la opción "-multicast on".

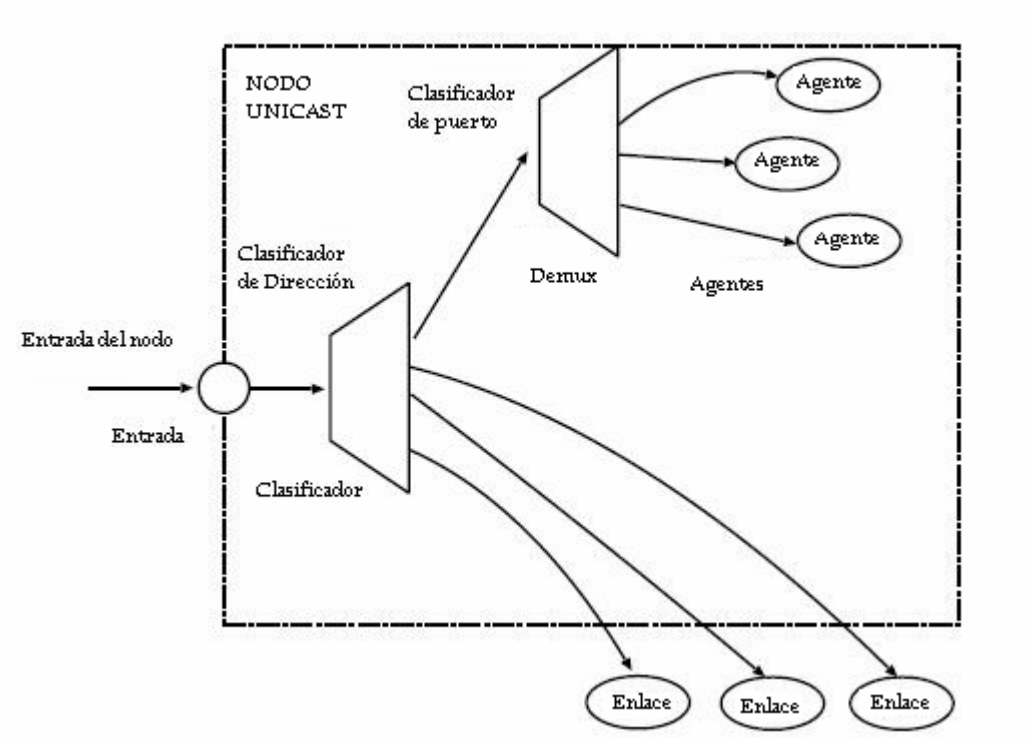
```
set ns [new Simulator -multicast on]
```

Cuando una simulación usa enrutamiento multicast el bit más alto de la dirección indica si la dirección es una dirección multicast o unicast. Si el bit es 0 la dirección representa una dirección unicast, de lo contrario la dirección representa un multicast.

2.2 MÉTODOS Y CONFIGURACIÓN DE NODOS

Los procedimientos para configurar un nodo individual son clasificados en:

Figura 58. Estructura de un nodo Unicast. Observe que *entrada* es una variable en lugar de un objeto real.



Tomado de *NS MANUAL.*

- Funciones de control
- Direcciones y administración de número de puerto, funciones de enrutamiento unicast.
- Administración de agentes
- Agregar vecinos

2.3 INTERFAZ Y CONFIGURACIÓN DE NODOS

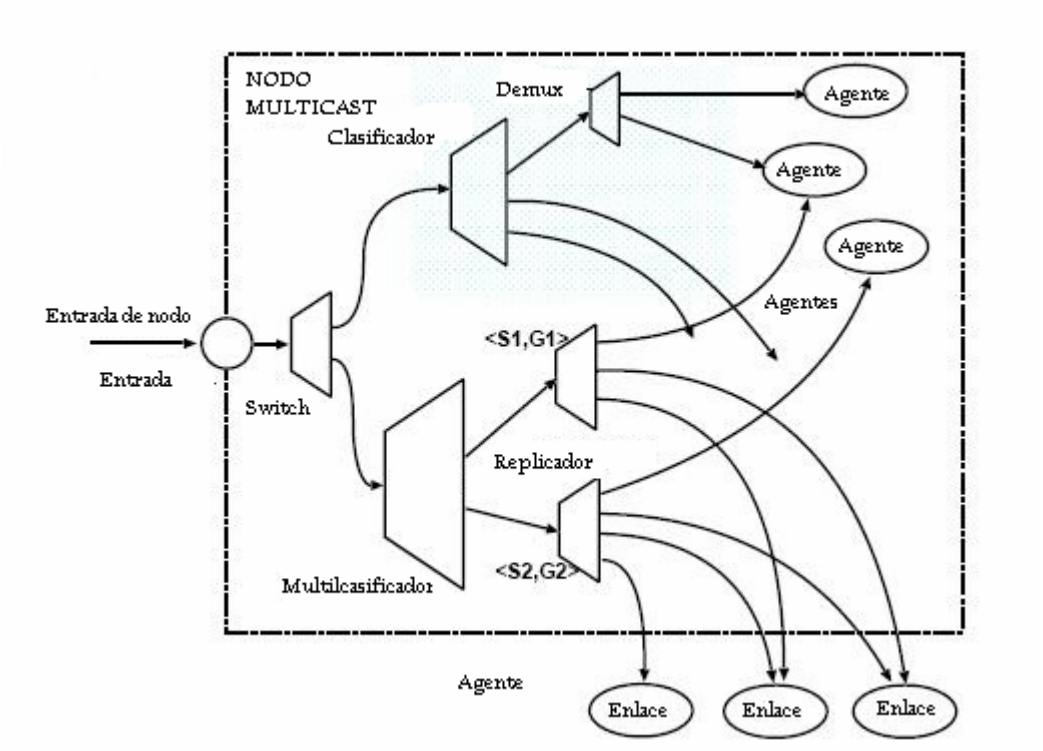
Los nodos API son utilizados para redes inalámbricas y simulaciones satelitales sin embargo se trabaja similarmente como los nodos cableados.

Para crear un nodo móvil capacitado para comunicación inalámbrica se cambia

la configuración por defecto, por ejemplo:

$ns node-config -adhocRouting dsdv

Figura 59. Estructura de un nodo Multicast.



Tomado de *NS MANUAL*.

La configuración de la interfaz de nodo consiste de dos partes, la primera parte trata con la configuración del nodo y la segunda crea nodos del tipo especificado como se visualizo anteriormente.

La configuración del nodo representa la definición de las diferentes características del nodo antes de que sea creado. Ellas pueden ser del tipo de estructura de direccionamiento usado en la simulación, definición de los componentes de red para los nodos móviles, activación o desactivación de opciones de trazo en niveles *Agent/Router/MAC*, seleccionar el tipo de

protocolo de enrutamiento *adhoc* para nodos inalámbricos o la definición de su modelo de energía.

Como un ejemplo de configuración de nodo móvil para un escenario inalámbrico, se asume que el nodo móvil corre *AODV* como su protocolo de enrutamiento *adhoc* en una topología jerárquica, también se decide activar el *tracing* en el agente y en el nivel de ruteo solamente. Aquí se asume una topología que ha sido inicializada con *"set topo [new Topography]"*. El comando de configuración del nodo debe parecerse al que se muestra a continuación:

```
$ns_ node-config -addressType hierarchical \
                 -adhocRouting AODV \
                 -llType LL \
                 -macType Mac/802_11 \
                 -ifqType Queue/DropTail/PriQueue \
                 -ifqLen 50 \
                 -antType Antenna/OmniAntenna \
                 -propType Propagation/TwoRayGround \
                 -phyType Phy/WirelessPhy \
                 -topologyInstance $topo \
                 -channel Channel/WirelessChannel \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF \
                 -movementTrace OFF \
```

Los valores por defecto para todas las opciones de arriba son NULL (Nulas) excepto –addressingType (Tipo de direccionamiento), su valor por defecto es plano (Flat). La opción –reset puede ser usada para iniciar todos los parámetros de configuración de nodo a sus valores por defecto.

Los comandos de configuración pueden ser configurados en líneas separadas cuando solo se desea cambiar algunas y en cuanto a las otras es posible mantenerlas en su valor por defecto, por ejemplo:

```
$ns_ node-config -addressingType hier
$ns_ node-config -macTrace ON
```

Simplemente las opciones que necesitan ser cambiadas son llamadas, por ejemplo, luego de configurar nodos móviles *AODV* (Y luego de crear nodos móviles *AODV*) se pueden configurar nodos estación base (Base-station) en la siguiente forma:

```
$ns_ node-config -wiredRouting ON
```

La mayoría de las características para estaciones base y nodos móviles son las mismas, excepto que los nodos estación base están capacitados para enrutamiento cableado o inalámbrico mientras los nodos móviles carecen de esta facultad. De esta forma se puede cambiar la configuración del nodo solamente cuando es requerido.

Todos los nodos creados luego de que un comando de configuración es dado, tendrán las mismas propiedades a menos que una parte o todo el comando de configuración del nodo sean ejecutados con diferentes valores de parámetros. Todos los valores de parámetros permanecen sin cambios a menos que ellos sean explícitamente cambiados. Luego de la creación de las estaciones base y los nodos móviles *AODV*, si se quiere crear nodos simples, se usa el siguiente comando de configuración de nodo:

```
$ns_ node-config –reset
```

Este configurará todos los valores de los parámetros a su configuración por defecto los cuales básicamente definen configuraciones de un nodo simple. Este tipo de configuración de nodo es orientado hacia nodos inalámbricos o satelitales.

2.4 EL CLASIFICADOR. La función de un nodo al recibir un paquete es examinar el campo del paquete, usualmente su dirección destino y en ocasiones su dirección fuente. Esto lo hace con el fin de determinar el objeto de interfaz de salida que es el siguiente receptor del paquete.

En *NS*, esta tarea es realizada por un simple objeto clasificador (*classifier*). Existen múltiples objetos clasificadores, que miran una parte específica del paquete y lo envían a través del nodo. Un nodo en *NS* usa diferentes tipos de clasificadores para diferentes propósitos. Entre los clasificadores de objetos en *NS* se encuentran clasificadores de dirección (*address classifiers*), clasificadores de multicast (*multicast classifiers*), clasificador de múltiples caminos (*multipath classifier*), clasificador hash (*hash classifier*) y finalmente el replicador, repetidor o duplicador (*the replicator*).

Un clasificador provee la forma de coincidir un paquete con algunos criterios lógicos basados en resultados de coincidencia. Cada clasificador contiene una tabla de objetos de simulación indicados por un número de ranura (*slot*). El trabajo de un clasificador es determinar el número de slot asociado con un paquete recibido y enviar este paquete al objeto referenciado por este slot particular.

Clasificadores De Dirección. Un clasificador de dirección es usado para soportar el envío de paquetes unicast. Este aplica un procedimiento ingenioso de desplazamiento y operación de máscara a una dirección destino de un paquete para producir un número de slot. El número de slot es devuelto desde el método *classify()*.

Clasificadores De Multicast. El clasificador multicast se encarga de clasificar paquetes de acuerdo a ambas direcciones, dirección fuente y direcciones destino (Grupo). Este mantiene una tabla que mapea pares fuente/grupo con números de slot. Cuando un paquete que llega al clasificador contiene un fuente/grupo desconocido este invoca un procedimiento OTcl *Node::new-group{}* para agregar una entrada a su tabla.

Este procedimiento OTcl puede usar el método *set-hash* para agregar nuevas entradas fuente, grupo, slot a la tabla del clasificador. El clasificador aplica una función *hash* a ambos el paquete fuente y la dirección destino, la función *hash* devuelve el número de slot indicado en la tabla para obtener el objeto apropiado.

Clasificador De Múltiples Caminos. Este objeto es diseñado para soportar envíos a través de múltiples rutas de igual costo, donde el nodo tiene múltiples rutas de igual costo al mismo destino y nos gustaría usar todas simultáneamente. Este objeto no revisa ningún campo en el paquete.

Clasificador Hash. Este objeto es usado para clasificar un paquete como un miembro de un flujo particular. El clasificador hash usa una tabla hash internamente para asignar paquetes a flujos. Estos objetos son usados donde el nivel de información de flujo es requerido. Varios tipos de flujos son disponibles, en particular los paquetes pueden ser asignados a flujos basados en ID de flujo, direcciones destino, direcciones fuente/destino o la combinación de direcciones fuente/destino mas ID de flujo. Los campos accedidos por el clasificador hash son limitados a la cabecera ip: src(), dst(), flowid().

El clasificador hash recibe paquetes, clasifica estos de acuerdo a su criterio de flujo y recobra el slot clasificador indicando el siguiente nodo que debería recibir el paquete. El clasificador hash incluye una variable ejemplar *default_* indicando cual slot es el que será usado para paquetes que no coinciden

ninguno de los criterios por flujo. El *default_* puede ser configurado opcionalmente.

Replicador O Duplicador. El replicador se diferencia de los demás clasificadores descritos con anterioridad, en este no se usa la función de clasificación. Este simplemente usa el clasificador como una tabla de *n* slots, sobrecarga el método *recv()* para producir *n* copias de un paquete que son entregados a todos los objetos referenciados en la tabla.

Para soportar envío de paquetes multicast, un clasificador que recibe un paquete multicast de la fuente *S* destinado al grupo de computadores *G* utiliza una función hash *h(S,G)* para dar un número de slot de la tabla de objetos. En envío multicast, el paquete debe ser copiado por cada uno de los enlaces principales a los nodos suscritos G menos uno. La producción de copias adicionales del paquete es realizado por la clase replicadora.

Esta clase realmente no clasifica paquetes, se encarga por el contrario de replicar un paquete, uno por cada entrada en su tabla y entregar la copia a cada uno de los nodos listados en la tabla. La última entrada en la tabla obtiene el paquete original.

2.5 MÓDULO DE ENRUTAMIENTO Y ORGANIZACIÓN DEL CLASIFICADOR

Como se puede observar, un nodo en *NS* esta compuesto de varios clasificadores. El nodo simple (unicast) contiene solamente un clasificador de dirección y un clasificador de puerto como se muestra en la figura 58. Cuando se extiende la funcionalidad del nodo, más clasificadores son agregados dentro del nodo base como por ejemplo el nodo multicast en la figura 59. Como mas bloques de funciones son agregados y cada uno de estos bloques requiere sus propios clasificadores, se convierte necesario para el nodo proveer una interfaz

uniforme para organizar estos clasificadores y puentear estos clasificadores a los bloques de computación de rutas.

Módulo De Enrutamiento. Cada implementación de enrutamiento en *NS* esta compuesto por tres bloques de funciones:

- *Agente de enrutamiento*, el cual intercambia paquetes de enrutamiento con sus vecinos.
- *Ruta lógica*, usa la información recogida por los agentes de enrutamiento (O la base de datos topológica global en el caso de enrutamiento estático) para realizar el cálculo de ruta actual.
- *Clasificadores*, ubicados dentro de un nodo. Ellos usan las tablas de enrutamiento calculadas para realizar el envío de paquetes.

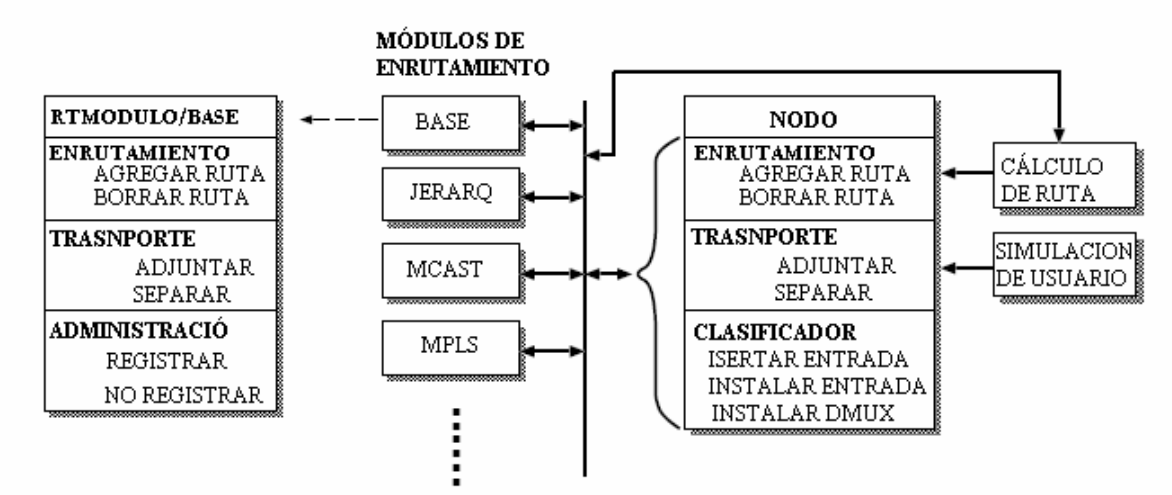
Cuando se implementa un nuevo protocolo de enrutamiento, no necesariamente se implementan todos los tres bloques anteriores. Por ejemplo, cuando se implementa un protocolo de estado del enlace, se implementa un agente de enrutamiento que intercambia información en la manera de estado del enlace y una ruta lógica que aplica el algoritmo de *Dijkstra* sobre la base de datos topológica resultante. Este puede usar los mismos clasificadores que otros protocolos de enrutamiento unicast.

Cuando una nueva implementación de protocolo de enrutamiento incluye más de una función de bloques, especialmente cuando esta contiene sus propios clasificadores, es deseable tener otro objeto el cual se llama *módulo de enrutamiento*, que administra todos estos bloques de funciones y los interfaza con los nodos para organizar los clasificadores.

Los módulos de enrutamiento pueden tener relación directa con los bloques de cálculo de rutas, por ejemplo, rutas lógicas y/o agentes de enrutamiento. Sin embargo, el cálculo de rutas puede no instalar sus rutas directamente a través

de un módulo de enrutamiento, porque allí pueden existir otros módulos que están interesados en aprender acerca de las nuevas rutas.

Figura 60. Relación funcional entre los objetos: Interacción entre nodos, módulos de enrutamiento y rutas.



Tomado de *NS MANUAL*.

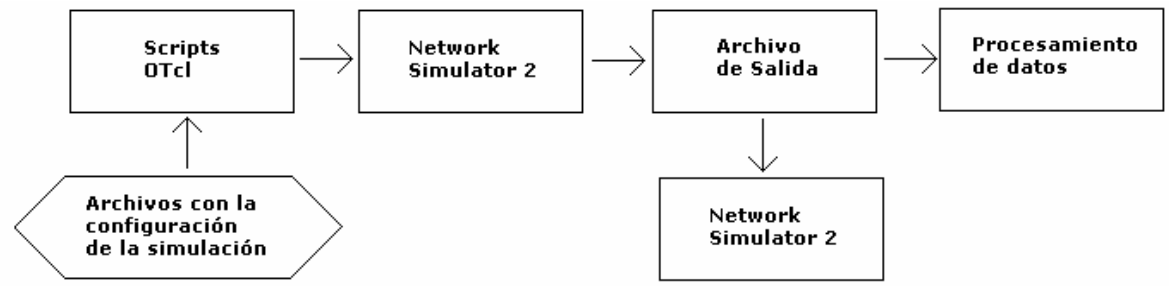
Actualmente, existen seis módulos de enrutamiento implementados en *ns*:

- *RtModule/Base*: Interfaz a protocolos de enrutamiento unicast. Provee funcionalidades básicas para agregar/borrar rutas y adjuntar/quitar agentes.
- *RtModule/Mcast*: Interfaz a protocolos de enrutamiento multicast. Cuyo propósito es establecer clasificadores multicast.
- *RtModule/Hier*: Enrutamiento jerárquico. Envuelve administración de clasificadores jerárquicos y instalación de rutas. Puede ser combinado con otros protocolos de enrutamiento como por ejemplo enrutamiento ad hoc.
- *RtModule/Manual*: Enrutamiento manual.

- *RtModule/VC*: Usa clasificadores virtuales.
- *RtModule/MPLS*: Implementa funcionalidades MPLS.

2.6 PROCESO DE SIMULACIÓN. Para realizar una simulación se deben llevar a cabo los pasos que se muestran a continuación:

Figura 61. Estructura del proceso de situación en *NS*.



Fuente, autor.

Como se puede observar para realizar una simulación en *NS* se debe contar con el script OTcl previamente creado, éste es programado en modo texto y guardado con la extensión ".tcl". <Posteriormente se simula con *NS* del cual se obtienen los archivos de salida, consistentes en animaciones en *NAM* y archivos de salida con extensión ".tr" que contienen los datos que se quiere que sean procesados por el simulador.

2.7 PRESENTACIÓN DE UN SCRIPT SIMPLE DE SIMULACIÓN. El presente script es uno de los ejemplares utilizados para simular uno de los 3 escenarios[39] de estudio presentes en el actual proyecto. En el transcurso del script se explica brevemente cada una de las partes que lo componen.

[39] Los escenarios de estudio para el presente proyecto se encuentran descritos en el capítulo 3.

```
#
# SCRIPT DE SIMULACIÓN DE ESCENARIO DE DOS NODOS INALÁMBRICOS
# (PUNTO DE ACCESO TRANSMISOR Y COMPUTADORA CON TARJETA DE RED
# INALAMBRIA LUCENT ORINOCO RECEPTORA) UBICADOS DENTRO DEL
# LABORATORIO DE REDES DE COMPUTADORAS DE LA ESCUELA DE
# INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
# DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER.
#
#
# ================================================================
# DEFINICIÓN DE OPCIONES
# ================================================================
#

set opt(chan)   Channel/WirelessChannel
set opt(netif)  Phy/WirelessPhy
set opt(mac)    Mac/802_11
set opt(ifq)    Queue/DropTail/PriQueue
set opt(ll)          LL
set opt(ant)    Antenna/OmniAntenna
set opt(adhocRouting)   DSDV
set opt(x)      40    ;# X DIMENSIÓN DE LA TOPOGRAFÍA
set opt(y)      40   ;# Y DIMENSIÓN DE LA TOPOGRAFÍA
set opt(ifqlen) 50   ;# TAMAÑO MÁXIMO DE PAQUETE EN LA INTERFAZ DE
# COLA
set opt(seed)   0.0
set opt(tr)     shadowing.tr    ;# ARCHIVO DE TRAZO
set opt(nam     shadowing.nam   ;# ARCHIVO DE TRAZO NAM
set opt(nn)     2               ;# NÚMERO DE NODOS SIMULADOS
set opt(stop)   40.0            ;# TIEMPO DE SIMULACIÓN

# ================================================================
# OTRAS CONFIGURACIONES POR DEFECTO
```

```
LL set mindelay_      50us
LL set delay_         25us
LL set bandwidth_          0     ;# NO USADO

Agent/CBR set sport_       0
Agent/CBR set dport_       0

Agent/LossMonitor set sport_     0
Agent/LossMonitor set dport_     0

Agent/UDP set sport_       0
Agent/UDP set dport_       0

Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

#
# GANANCIA UNITARIA, ANTENA OMNI-DIRECCIONAL.
# SE CONFIGURAN LAS ANTENAS PARA QUE SEAN CENTRADAS EN EL NODO
# CON UNA ALTURA DE 1.0 METRO.
#

Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.0
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

#
# CONFIGURACIÓN PARA ANTENAS LUCENT ORINICO 802.11b PC-CARD A
# 11Mbps CON 25 METROS DE RANGO. SE INICIALIZA LA INTERFAZ DE
# MEDIO COMPARTIDO CON PARÁMETROS PARA HACER QUE TRABAJE A 2.4 GHz
```

```
# CON INTERFAZ DE RADIO DSSS. Pt_ ES LA SEÑAL DE POTENCIA
# TRANSMITIDA EN VATIOS. EL MODELO DE PROPAGACIÓN Y Pt_ DETERMINA
# LA POTENCIA DE LA SEÑAL RECIBIDA DE CADA PAQUETE. EL PAQUETE NO
# PUEDE SER CORRECTAMENTE RECIBIDO SI LA POTENCIA RECIBIDA ES
# MENOR QUE RXThresh_.
#

Phy/WirelessPhy set Pt_ 0.031622777
Phy/WirelessPhy set bandwidth_ 11e6
Mac/802_11 set dataRate_ 11e6
Mac/802_11 set basicRate_ 11e6
Phy/WirelessPhy set freq_ 2.437e9
#CANAL-6, 2.437GHz
Phy/WirelessPhy set CSThresh_ 5.011872e-12
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set RXThresh_ 5.011872e-12

# ======================================================================
# PROGRAMA PRINCIPAL
# ======================================================================

#
# INICIALIZACIÓN DE VARIABLES GLOBALES
#

#
# CREACIÓN DE UN EJEMPLAR DE SIMULACIÓN
#

set ns_         [new Simulator]

# CONFIGURACIÓN DEL CANAL INALÁMBRICO
# set wchan     [new $opt(chan)]
```

```
#
# DEFINICIÓN DEL MODELO DE PROPAGACIÓN. pathlossExp_ ES EL
# EXPONENTE DE PÉRDIDAS POR ESPACIO path-loss exponent,
# PARA PREDECIR LA POTENCIA MEDIA RECIBIDA. std_db_ ES LA
# DESVIACIÓN POR SOMBRA (dB), REFLEJANDO CUANTO CAMBIAN LAS
# PROPIEDADES DE PROPAGACIÓN DENTRO DEL AMBIENTE. dist0_ ES UNA
# DISTANCIA DE REFERENCIA ACERCADA, USUALMENTE 1.0 m.
#

set prop   [new Propagation/Shadowing]
$prop set pathlossExp_ 2.7640
$prop set std_db_ 5.7381
$prop set dist0_ 1.0
$prop seed predef 0

#
# DEFINICIÓN DE TOPOLOGÍA
#

set wtopo  [new Topography]
$wtopo load_flatgrid $opt(x) $opt(y)

#
# CREACIÓN DE OBJETOS DE TRAZO PARA NS Y NAM
#

set tracefd     [open $opt(tr) w]
set namtrace    [open $opt(nam) w]


$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
```

```
#
# USO DEL NUEVO FORMATO DE ARCHIVOS DE TRAZO.
#

$ns_ use-newtrace

#
# CREACIÓN DEL GOD
# SE DEFINE CUANTOS NODOS DEBEN SER CREADOS
#

set god_ [create-god $opt(nn)]

#
# CONFIGURACIÓN DE NODO GLOBAL
#

$ns_ node-config  -adhocRouting $opt(adhocRouting) \
            -llType $opt(ll) \
            -macType $opt(mac) \
            -ifqType $opt(ifq) \
            -ifqLen $opt(ifqlen) \
            -antType $opt(ant) \
            -propInstance $prop \
            -phyType $opt(netif) \
            -channelType $opt(chan) \
            -topoInstance $wtopo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
```

```
#
# CREACIÓN DEL NÚMERO ESPECIFICADO DE NODOS [$opt(nn)]. LUEGO SE
# ADJUNTAN ESTOS AL CANAL.
#

for {set i 0} {$i < $opt(nn) } {incr i} {
     set node_($i) [$ns_ node]
#    $node_($i) random-motion 0 ;# SE DESHABILITA MOVIMIENTO
#                                    RANDÓMICO
#    $node_($i) topography $wtopo
}

#
# DEFINICIÓN DE LAS POSICIONES DE LOS NODOS
#

$node_(0) set X_ 20.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.6
$node_(1) set X_ 20.0
$node_(1) set Y_ 11.0
$node_(1) set Z_ 0.0

#
# DEFINICIÓN DE UN PROCEDIMIENTO DE 'finish'
#

proc finish {} {
     global tracefd
     # SE CIERRAN LOS ARCHIVOS DE SALIDA
     close $tracefd

     # SE LLAMA A NAM PARA DESPLEGAR LOS RESULTADOS
```

```
      exec ./nam shadowing.nam &
         exit 0
}

#
# SE DEFINE UN PROCEDIMIENTO EL CUAL PERIÓDICAMENTE GRABA EL ANCHO
# DE BANDA RECIBIDO POR EL AGENTE DE TRÁFICO sink0 Y ESCRIBE ESTE
# DENTRO DEL ARCHIVO DE SALIDA tracefd.
#

proc record {} {
         global sink0 tracefd
# SE OBTIENE UN EJEMPLAR DEL SIMULADOR
      set ns [Simulator instance]
# SE CONFIGURA EL TIEMPO LUEGO DEL CUAL EL PROCEDIMIENTO DEBE
# SER LLAMADO NUEVAMENTE
         set time 30.0
# CUANTOS BYTES HAN SIDO RECIBIDOS POR EL AGENTE DE TRAFICO sink?
      set bw0 [$sink0 set bytes_]
# SE OBTIENE EL TIEMPO ACTUAL
         set now [$ns now]
# SE CALCULAN LOS PAQUETES RECIBIDOS Y SE ESCRIBE ESTE EN EL
# ARCHIVO DE SALIDA
         puts $tracefd "$now [expr ($bw0-492)/1512+1]"
# SE RESETEA EL VALOR DE bytes_ EN EL AGENTE DE TRAFICO sink
         $sink0 set bytes_ 0
# SE RE-PROGRAMA EL PROCEDIMIENTO
         $ns at [expr $now+$time] "record"
}

#
# SE CONFIGURAN LOS AGENTES Y TIPOS DE GENERADORES DE TRÁFICO
# EMISORES Y RECEPTORES
```

```
#
# SE CREA UN AGENTE DE TRÁFICO UDP LLAMADO udp0 Y SE ADJUNTA AL
# NODO node_(0)
#

set udp0 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp0

#
# SE CREA UN AGENTE RECEPTOR DE TRÁFICO DEL TIPO LossMonitor
# LLAMADO sink0 EL CUAL MONITOREA LOS PAQUETES PERDIDOS Y SE
# ADJUNTA AL NODO node_(1)
#

set sink0 [new Agent/LossMonitor]
$ns_ attach-agent $node_(1) $sink0

#
# SE CREA UN GENERADOR DE TRÁFICO CBR (RATA DE BIT CONSTANTE) CON
# TAMAÑO DE PAQUETE DE 1472 Bytes, SE CONFIGURA PARA QUE ENVIÉ 500
# PAQUETES POR SEGUNDO DURANTE UN TIEMPO DE 30 SEGUNDOS, EL CUAL
# CUENTA DESDE 2 s Y FINALIZA EN 32 s, POR CONSIGUIENTE ENVÍA
# UN TOTAL DE 15000 PAQUETES. ESTE SE ADJUNTA AL AGENTE udp0
#
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1472
$cbr0 set interval_ "[expr 1.0/500.0]"
$cbr0 set random_ 0
$cbr0 set maxpkts_ 15000
$cbr0 attach-agent $udp0
$ns_ connect $udp0 $sink0
$ns_ at 2.0 "$cbr0 start"
$ns_ at 32.0 "$cbr0 stop"
```

```
#
# SE DEFINE LA POSICIÓN INICIAL EN NAM
#

for {set i 0} {$i < $opt(nn)} {incr i} {

# 1 DEFINE EL TAMAÑO DEL NODO EN NAM, ESTE DEBE AJUSTARSE DE
# ACUERDO AL ESCENARIO. LA FUNCIÓN DEBE SER LLAMADA LUEGO QUE EL
# MODELO DE MOVILIDAD ES DEFINIDO

    $ns_ initial_node_pos $node_($i) 1
}

#
# SE INICIA MONITOREANDO EL ANCHO DE BANDA RECIBIDO LLAMANDO AL
# PROCEDIMIENTO record A LOS 2 s
#

$ns_ at 2.0 "record"

#
# SE INFORMA A LOS NODOS CUANDO TERMINA LA SIMULACIÓN
#

for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).000000001 "$node_($i) reset";
}

$ns_ at  38.999999999 "finish"

#
# SE INFORMA A NAM EL TIEMPO DE PARADA DE LA SIMULACIÓN
#
```

```
$ns_ at  $opt(stop)   "$ns_ nam-end-wireless $opt(stop)"

$ns_ at  $opt(stop).000000001 "puts \"NS EXITING...\" ; $ns_ halt"

puts "Iniciando Simulacion..."

$ns_ run
```

ANEXO D. Consideraciones generales

1 SOFTWARE

Debido a que este proyecto es un estudio de una herramienta software de simulación, se hace necesario mostrar las demás herramientas que dieron soporte para el funcionamiento adecuado de las simulaciones.

Instalación y puesta en marcha del simulador network simulator en ambiente linux. La herramienta de simulación Network Simulator versión 2 "*Ns-2*" se utiliza bajo sistema operativo Linux, aunque también se encuentra la versión Windows; Se decide utilizar Linux ya que es un sistema operativo más robusto y el software es mas completo en este entorno.

Para comenzar a instalar la herramienta se debe entrar en modo *root* e introducir el password de *root* para que Linux de los permisos de instalación de software. En el presente proyecto se trabaja con el sistema operativo Linux Red Hat 9.0. Luego de esto se debe crear una carpeta en */home*, preferiblemente llamada *NS*, donde se guardan los archivos de la herramienta. Si se tiene la herramienta *ns-allinone-2.27.tar.gz* en copia dura como en un cd-room se procede a copiarla en la carpeta *ns*, la cual fue creada anteriormente. Si no se tiene en copia dura se puede descargar gratuitamente desde Internet en la siguiente url: http://www.isi.edu/nsnam/dist/, donde se puede escoger desde la versión mas antigua hasta la mas actualizada del simulador, en este caso se elige trabajar con la última versión "2.27" y el nombre del archivo es *ns-allinone-2.27.tar.gz*. Además, el tamaño del archivo es de 54Mbytes. Desde esa url se puede descargar tanto el paquete completo del simulador como paquetes independientes como NS (versiones 1 y 2), XGRAPH, NAM, etc. Igualmente se procede a guardar el paquete en la carpeta *NS* anteriormente creada.

Luego de copiar el archivo en la carpeta *NS* se procede a abrir una ventana de "Terminal" la cual permite mediante series de comandos darle órdenes a Linux para que realice acciones como configurar, instalar, abrir y cerrar programas. En esta ventana se accede a la carpeta *NS* donde se encuentra el paquete *ns-allinone-2.27.tar.gz* para proceder a instalarlo.

Para acceder a esta carpeta se introducen los siguientes comandos en la ventana Terminal:

```
[root@localhost root]# cd /home
[root@localhost home]# cd ns
```

Luego de esto se puede utilizar el comando *ls* para observar que archivos se encuentran contenidos en esta carpeta, pero como ya se conoce de antemano que el paquete se encuentra ubicado allí no hay ningún problema, la verificación se puede realizar a gusto del usuario.

Se debe observar que el archivo como tal se encuentra empaquetado y comprimido utilizando los formatos .tar (tar) y .gz (gunzip) para que no ocupe tanto espacio, por lo tanto se debe proceder a desempaquetarlo y descomprimirlo para luego proceder a instalarlo.

Para descomprimir y desempaquetar el archivo se utilizan los siguientes comandos:

```
[root@localhost ns]# gunzip -df ns-allinone-2.27.tar.gz
[root@localhost ns]#

[root@localhost ns]# tar -xf ns-allinone-2.27.tar
[root@localhost ns]#
```

Luego de realizar estos pasos el paquete crea una capeta llamada *ns-allinone-2.27* donde se encuentran los archivos de instalación del software.

Para instalar el software se debe entrar a la carpeta mencionada anteriormente para ejecutar el archivo *install*, utilizando para eso las siglas "*./*". Para realizar esto se utilizan los siguientes comandos:

```
[root@localhost ns]# cd ns-allinone-2.27
[root@localhost ns]#./install
```

Luego de esto Linux comienza a instalar los archivos correspondientes mostrando en la pantalla cada archivo y paso que realiza durante la instalación hasta que esta termina. Se debe notar que el tiempo que dura la instalación depende del tipo de arquitectura que se esté utilizando, en nuestro caso se utiliza la arquitectura de un computador marca DELL con procesador Intel Pentium 4 a 2.4Ghz y 512Mbytes de memoria RAM en la cual la instalación toma mas o menos 10 minutos. Al terminar la instalación se observa en la pantalla la información de algunos consejos útiles.

Antes de proceder a ejecutar los scripts de simulación se debe configurar algunas pautas que no se comentan cuando se instala ns, ya que se encontraron mediante la practica al observar que las simulaciones no corren *XGRAPH* ni *NAM*, para esto se realizan estos pasos: Se deben copiar y pegar los archivos ejecutables *NAM* y *XGRAPH* a la carpeta *ns-2.27*. Estos archivos se encuentran en las correspondientes carpetas *NAM* y *XGRAPH* que a su vez se encuentran dentro de la carpeta ns-allinone-2.27.

Luego de esto se puede ejecutar cualquier script que se realice posteriormente.

Los scripts desarrollados por los usuarios en *NS* tienen la extensión .tcl, estos scripts se programan en un editor de texto como *Kate* en Red Hat Linux dando la extensión de tipo de archivo *.tcl* al guardar para que *NS* pueda identificarlos.

Para correr un script desarrollado en *NS*, se debe ingresar a la carpeta *ns-2.27* y ejecutar el archivo *ns* ubicado dentro de esta, seguido de un espacio y la dirección de ubicación donde se encuentra el script *.tcl* que se desea ejecutar. Se debe recordar que para correr un archivo ejecutable en Linux se debe introducir el comando "*./*" seguido del archivo en cuestión. Los pasos a seguir para correr un archivo en *NS* desde una ventana de Terminal son los siguientes:

```
[root@localhost root]# cd /home
[root@localhost home]# cd ns
[root@localhost ns]# cd ns-allinone-2.27
[root@localhost ns-allinone-2.27]# cd ns-2.27
[root@localhost ns-2.27]# ./ns /home/UBICACIÓN Y NOMBRE DEL SCRIPT.TCL
```

Se recomienda ubicar los scripts de simulación dentro de esta carpeta (*ns-2.27*) ya que es más fácil ejecutarlos desde allí; Solo es utilizar el siguiente comando cuando se esta ubicado dentro de la carpeta *ns-2.27*:

```
[root@localhost ns-2.27]# ./ns NOMBRE DEL SCRIPT.TCL
```

Además se recomienda que los nombres de las carpetas y el script estén compuestos de una sola palabra. Igualmente los archivos ejecutables *NAM* y *XGRAPH* se pueden correr de la misma manera.

NS almacena los archivos de salida de la simulación como por ejemplo salidas *.NAM* (animación de red) y *.tr* (trazo) dentro de la carpeta *ns-2.27*.

MGEN. El MGEN es un compuesto de herramientas que permiten tomar medidas de rendimiento de una red IP utilizando flujos de paquetes TCP/UDP. La utilidad "MGEN" genera patrones de tráfico en tiempo real de tal forma que la red se puede someter a diferentes tipos de carga. Es posible crear archivos de configuración (scripts) que simulan distintos tipos de flujos para aplicaciones TCP/UPD/IP unicast y multicast.

Los paquetes pueden ser recibidos utilizando la herramienta "DREC" que posee la capacidad de unirse/abandonar dinámicamente grupos multicast y que se encarga de generar un fichero de log con marcas de tiempo, que posteriormente puede ser analizado con otra herramienta para obtener estadísticas tales como tasa de pérdidas, retardo en la comunicación, throughput, etc.

La herramienta ha sido migrada a IPv6 y sólo está soportada para Linux y FreeBSD, con los parches de USAGI y KAME para IPv6.

Resumiendo, el paquete se distribuye con las siguientes herramientas:
MGEN: (Multi-GENerator) genera patrones de tráfico en tiempo real para destinos unicast/multicast según un script de configuración o por línea de comandos. Se pueden especificar tamaño de paquetes y velocidades de transmisión para flujos individuales.
DREC: (Dynamic-RECeiver) recibe y almacena el tráfico generado por la herramienta Mgen.
MCALC: (Multi-CALCulator) Examina el archivo de log creado por Drec y calcula estadísticas por flujo recibido.

Procedimiento de generación de tráfico. Para generar un tráfico cambiante se utilizaron archivos o programas en SHELL con extensión .sh tanto para el MGEN como para el DREC. Estos archivos contenían todo el proceso de generación y recepción del tráfico variable.

Generación. Para generar tráfico se digito un archivo de texto y se compiló como un archivo shell de extensión .sh para manejar las variaciones de velocidad que se pretendían hacer para cada prueba. Este shell contenía dentro la generación del script correspondiente para ejecutar el MGEN. Las siguientes líneas corresponden al shell para el manejo del envío de paquetes:

```
#!/bin/bash
puerto1=2557
echo "HORA DE INICIO HH"; read hora
echo "MINUTOS MM";  read minutos;
num=1;
puerto2=$puerto1+2000;
segundos=00;
diferencia=0;
tamao=1472
tasa=502
diferencia=$((tasa/20));
tasa1=$tasa;
echo " " > /home/Resultados/OP/prueba
        while test $num -le 20
        do
    tasa=$tasa1;
    echo "HORA: $hora:$minutos:$segundos"
    echo "HORA: $hora:$minutos:$segundos" >>
/home/Resultados/OP/prueba
    echo "TAMAnO DE PAQUETE: $tamao" >> /home/Resultados/OP/prueba
    echo "TASA: $tasa" >> /home/Resultados/OP/prueba
    echo "PORT $puerto2" > /home/mscript
    echo "001000 1 ON 192.168.45.201:$puerto1 PERIODIC $tasa
$tamao" >> /home/mscript
    echo "031000 1 OFF" >> /home/mscript
    echo "032000 EXIT " >> /home/mscript
    /home/MGENv3/MGEN/./mgen -i eth0 -S $hora:$minutos:$segundos
/home/mscript >> /home/Resultados/OP/prueba
    tasa1=$((tasa-diferencia));
    segundos=$((segundos+40));
        if test $segundos -ge 60
        then
          minutos=$((minutos+1));
              segundos=$((segundos-60));
              if test $minutos -ge 60
                    then
              hora=$((hora+1));
```

```
                minutos=0;
                        if test $hora -ge 24
                        then
                        hora=0;
                        fi
                fi
        fi
     num=$((num+1));
        done
num=1;
grep 'HORA:' /home/Resultados/OP/prueba >/home/Resultados/OP/buffer.txt
cut -b 7-15 /home/Resultados/OP/buffer.txt >/home/Resultados/hora.txt
grep 'TAMAnO DE PAQUETE:' /home/Resultados/OP/prueba
>/home/Resultados/OP/buffer.txt
cut -b 20-24 /home/Resultados/OP/buffer.txt >/home/Resultados/tamano.txt
grep 'TASA:' /home/Resultados/OP/prueba >/home/Resultados/OP/buffer.txt
cut -b 7-11 /home/Resultados/OP/buffer.txt >/home/Resultados/tasa.txt
grep 'MGEN: Packets Tx' /home/Resultados/OP/prueba
>/home/Resultados/OP/buffer.txt
cut -b 31-36 /home/Resultados/OP/buffer.txt >/home/Resultados/PkTx.txt
```

Este shell define entre sus líneas los puertos de envío y recepción de datos, la hora de comienzo de la prueba, el tamaño y la tasa de transferencia; esto se define internamente ya que para las pruebas se hicieron que fueran constantes. A partir de esos datos se comienza a hacer las variaciones de velocidad por medio de un bucle repetitivo de 20 ciclos.

En cada ciclo este shell genera un script pequeño con los datos correspondientes a hora, tamaño de paquete y tasa de transferencia para ejecutar variablemente el MGEN; los resultados son almacenados en un archivo de texto temporal.

Al terminar los 20 ciclos, el shell toma el archivo de texto temporal de resultados del MGEN y le hace un proceso de selección de los datos necesarios, aislando varios archivos con los correspondientes datos de hora, tamaño de paquete, tasa y número de paquetes transmitidos.

Recepción. El shell receptor ejecuta tanto el DREC para escuchar los paquetes como el MCALC para hacer algunos cálculos, y entregar resultados tales como el número de paquetes recibidos, el retardo, la tasa de paquetes y datos recibidos, los paquetes perdidos y estadísticas de latencia y variación de latencia o jitter, pero de todos esos sólo es necesario para propósito de este proyecto el número de paquetes recibidos así que las demás variables se descartaron del archivo de salida aunque de todos modos el MCALC las entrega como resultado. Las líneas correspondientes al shell de recepción son las siguientes:

```
#!/bin/bash
echo "COORDENADA "; read cod_posicion;
echo "HORA DE INICIO HH"; read hora
echo "MINUTOS MM";  read minutos;
num=1;
diferencia=0;
segundos=00;
tasa=500;
tamao=1472;
diferencia=$((tasa/20));
tasa1=$tasa;
echo " " > /home/Resultados/OP/prueba$cod_posicion
        while test $num -le 20
        do
     tasa=$tasa1;
     echo "$num"
     echo "HORA: $hora:$minutos:$segundos"
     echo "COORDENADA: $cod_posicion" >>
/home/Resultados/OP/prueba$cod_posicion
     echo "HORA: $hora:$minutos:$segundos" >>
/home/Resultados/OP/prueba$cod_posicion
     echo "TAMAnO DE PAQUETE: $tamao" >>
/home/Resultados/OP/prueba$cod_posicion
     echo "TASA: $tasa" >> /home/Resultados/OP/prueba$cod_posicion
     echo "032000   EXIT" > /home/rscript
     /home/MGENv3/MGEN/./drec -p 2557 -i eth0 -S
$hora:$minutos:$segundos /home/rscript /home/Resultados/OP/logfile
     /home/MGENv3/MGEN/./mcalc /home/Resultados/OP/logfile >>
/home/Resultados/OP/prueba$cod_posicion
     echo "  " >> /home/Resultados/OP/prueba$cod_posicion
     tasa1=$((tasa-diferencia));
```

```
    segundos=$((segundos+40));
      if test $segundos -ge 60
      then
      minutos=$((minutos+1));
      segundos=$((segundos-60));
            if test $minutos -ge 60
            then
            hora=$((hora+1));
                  minutos=0;
                  if test $hora -ge 24
                  then
                  hora=0;
                  fi
            fi
      fi
    num=$((num+1));
      done
num=1;
grep 'COORDENADA:' /home/Resultados/OP/prueba$cod_posicion
>/home/Resultados/OP/buffer.txt
cut -b 13-18 /home/Resultados/OP/buffer.txt
>/home/Resultados/OP/coordenada.txt
grep 'HORA:' /home/Resultados/OP/prueba$cod_posicion
>/home/Resultados/OP/buffer.txt
cut -b 7-15 /home/Resultados/OP/buffer.txt >/home/Resultados/OP/hora.txt
grep 'TAMAnO DE PAQUETE:' /home/Resultados/OP/prueba$cod_posicion
>/home/Resultados/OP/buffer.txt
cut -b 20-24 /home/Resultados/OP/buffer.txt
>/home/Resultados/OP/tamano.txt
grep 'TASA:' /home/Resultados/OP/prueba$cod_posicion
>/home/Resultados/OP/buffer.txt
cut -b 7-11 /home/Resultados/OP/buffer.txt >/home/Resultados/OP/tasa.txt
grep 'MCALC: Total packets received'
/home/Resultados/OP/prueba$cod_posicion >/home/Resultados/OP/buffer.txt
cut -b 35-40 /home/Resultados/OP/buffer.txt >/home/Resultados/OP/PkRx.txt
paste -d' ' /home/Resultados/OP/coordenada.txt
/home/Resultados/OP/hora.txt /home/Resultados/OP/tamano.txt
/home/Resultados/OP/tasa.txt /home/Resultados/OP/PkRx.txt
>>/home/Resultados/TOTAL.txt
paste -d' ' /home/Resultados/OP/coordenada.txt
>>/home/Resultados/coordenada.txt
paste -d' ' /home/Resultados/OP/hora.txt >>/home/Resultados/hora.txt
paste -d' ' /home/Resultados/OP/tamano.txt
>>/home/Resultados/tamano.txt
paste -d' ' /home/Resultados/OP/tasa.txt >>/home/Resultados/tasa.txt
paste -d' ' /home/Resultados/OP/PkRx.txt >>/home/Resultados/PkRx.txt
```

Este shell se utiliza en la máquina receptora de tráfico y sus líneas contienen entradas de datos por parte del usuario tal como la coordenada del lugar a tomar la prueba y la hora de inicio que debe ser la misma que la hora de inicio de generación por parte del shell que emite los paquetes.

Al igual que el shell generador, éste para cada prueba ejecuta un ciclo de 20 repeticiones en las cuales genera el script necesario para la ejecución del DREC, pero además va guardando los resultados en unos archivos temporales de texto; al final toma los datos que se necesitan para analizar y los almacena en un archivo de texto que guarda registro tanto de la prueba actual como de las pruebas anteriores de diferentes coordenadas.

2. HARDWARE

Equipo de Cómputo. Se utilizaron 2 equipos Optiplex GX 260 marca Dell[40], Sus características se especifican a continuación:

- Fabricante: Dell Computer corporation
- modelos : Dell Optiplex Gx 260
- Procesador: Pentium 4 de 2.4 Ghz
- Memoria RAM: 512 MB
- Disco duro: 20MB
- Sistema operativos: linux red hat 9.0

Punto de Acceso. El punto de acceso o *Access Point* utilizado en las pruebas fue, DLINK AIRPRO DWL – 2000AP el cual cuenta con modo de acceso dual, es decir que puede conectar una red 802.11a conectándose a 54 Mbps y una red 802.11b, conectándose a 11Mbps o 22Mbps, además este dispositivo mantiene una conexión transparente con los dispositivos, brindando así el

servicio de *roaming*. También construye un puente entre una red cableada y una red inalámbrica por lo cual tiene un puerto Ethernet 10/100.

Para conexión a 54 Mbps la frecuencia utilizada por el punto de acceso es de 5 Ghz y para conexión 11 Mbps y 22 Mbps la frecuencia es 2.4 Ghz.

- Soporta multiples Acces Point
- Compatible con 802.11a y 802.11b
- Potencia de transmisión ajustable
- Escalamiento de Tasa dinámica
- Rango de 900 *feet* (dependiendo del ambiente)
- Soporta encriptación de 64/128/152/256 bits
- Consistente con las regulaciones regionales de frecuencia
- Modulación DSSS (*Direct Secuence Spread Spectrum*)
- Soporta modulación PBCC (*Packet Binary Convolution Code*)
- Utiliza ODFM (*Ortogonal División Frecuency Multiplexing*)
- Fácil configuración basada en la Web
- Nivel de seguridad de usuario

Tabla 12. Características generales del punto de acceso.

Standards	IEEE 802.11ª IEEE 802.11b IEEE 802.3 and IEEE 802.3u
Ports	10/100 Base-T Ethernet RJ-45 UTP Power – 5.0V DC 2.5A
Network Management	Web – Based Interface SNMP Management
Network architecture	Support infrastructure mode (Comunications to wired

	via Acces Point with Roaming)
Diagnostic LED	Power 100M Link/Act 10M Link/Act 11ª WLAN 11b WLAN
Range	Indoors-up to 328 feet (100 meters) Outdoor-up to 1312 feet (400meters)
Temperature	Operating 0ºC to 40ºC (32ºF to 104ºF) Storing -25ºC to 65ºC (-77ºF to140ºF)
Humidity	5% - 95% non-condensing
EMI/Safety	IEE 802.11ª -EMC: EN 301 489-1 And -17 EN 60950 DFS/TPC: 301 893 Draft IEEE 802.11b –EMC: EN 300 328, EN 300 826, EN 60950
Operation Voltage	3.3 V+/- - 10%
Physical Dimensions	L = 23.5 cm W = 15.9 cm H= 3.8 cm

802.11ª Specifications	
Data Rates	6, 9, 12, 18, 24, 36, 48, 54 Mbps
Data Security	64, 128, 154-bit WEP (Wired Equivalent Privacy) Encryption Acces Control List based on MAC address
Antena Type	5bBi dipole antenna with diversity
Avaible Channels	Subject to local regulatory restrictions
Frecuency Range	5.150 – 5.825 GHz
Modulation Technology	Ortogonal Frecuency Division Multiplexing (OFDM)
Modulation Techniques	BPSK

	QPSK 16 QAM 64 QAM
802.11b Specifications	
Data Rates	1, 2, 5.5, 11, 22 Mbps
Data Security	64, 128, 256- bit WEP (Wired equivalent privacy) Encryption
Antena Type	2dBi Antenna with diversity
Avaible Channels	Subject to local regulatory restrictions
Frecuency Range	2.4 2.4835 GHz
Modulation Technology	Direct Sequence Spread spectrum
Modulation Techniques	CCK DQPSK DBSK

Tarjetas Inalámbricas

Lucent Orinoco USB Client silver. Se utilizaron dos tarjetas Lucent Tecnologies Orinoco USB que poseen las siguientes características:

- Conecta un computador en un grupo de trabajo punto a punto o infraestructura
- Certificado por la WECA (Wireless Ethernet Compatibility Alliance) lo cual quiere decir que tiene compatibilidad con otros vendedores de productos WLAN
- Completa compatibilidad con otras LAN basada en DSSS (Direct Secuence Spread Spectrum)
- Compatible con USB rev. 1.1
- Mecanismo de selección de tasa automática de transmisión en el rango de 11,5.5,2,1 Mbit/s

- Selección de la frecuencia del canal (2.4Ghz)
- Roaming sobre múltiples canales
- Manejo de potencia
- Encriptación de datos WEP (Wired Equivalent Privacy)
- 128 bit RC4 encryption (Gold)

Tabla 13. Especificaciones físicas.

Dimensions	(LxWxH)	63 x 89 x 145 mm
Weight	170 gram	
Cable length	100 cm	
Temperature & Humidity		
Operation	0° to 40° C[1]	
Transit	-20° to 75° C	maximum humidity 95% (no condensation allowed)
Storage	-20° to 75° C	

1. Although the USB Client may still operate in the range of -20° to 70°C, operation outside the range of 0° to 40° C may no longer be according to specifications.

Tabla 14. Características de potencia.

Doze Mode	10 mA
Receive Mode	245 mA (Nominal)
Transmit Mode	360 mA (Nominal)
Power Supply	5 V

Tabla 15. Características de red.

Compatibility	■ IEEE 802.11 Standard for Wireless LANS (DSSS) ■ Wi-Fi (Wireless Fidelity) certified by the Wireless Ethernet Compatibility Alliance (WECA). ■ Universal Serial Bus Revision 1.1. specification
Host Operating System	Microsoft Windows® 98, ME and 2000: ■ NDIS5 Miniport Driver
Media Access Protocol	CSMA/CA (Collision Avoidance) with Acknowledgment (ACK)

Tabla 16. Características de radio.

R-F Frequency Band	**2.4 GHz** (2400-2500 MHz)	
Supported sub-channels	1	2412
	2	2417
	3	2422
	4	2427
	5	2432
	6	2437
	7	2442
	8	2447
	9	2452
	10	**2457**
	11	2462
Modulation Technique	Direct Sequence Spread Spectrum CCK 11 & 5.5 Mb/s, DQPSK for 2 Mb/s and DBPSK for 1 Mb/s	
Spreading	11-chip Barker Sequence	
Bit Error Rate (BER)	Better than 10^{-5}	
Nominal Output Power	15 dBm	
Encryption	64-bit Wired Equivalent Privacy (WEP) - Silver 128-bit (RC4) - Gold	

Range / Transmit Rate	**11 Mb/s**	**5.5 Mb/s**	**2 Mb/s**	**1 Mb/s**
Open Office Environment	160 m (525 ft.)	270 m (885 ft.)	400 m (1300 ft.)	550 m (1750 ft.)
Semi-Open Office Environment	50 m (165 ft.)	70 m (230 ft.)	90 m (300 ft.)	115 m (375 ft.)
Closed Office	25 m (80 ft.)	35 m (115 ft.)	40 m (130 ft.)	50 m (165 ft.)

Receiver Sensitivity	-83 dBm	-87 dBm	-91 dBm	-94 dBm
Delay Spread (FER of <1%)	65 ns	225 ns	400 ns	500 ns

Dlink DWL – 120 USB Wireless. Se utilizaron 3 tarjetas Dlink air DWL-120. Las características generales de este producto son las siguientes:

Data Security

- 64/128-bit WEP (Wired Equivalent Privacy) Encryption

Data Rate & Modulation

- 11 Mbps: CCK
- 5.5 Mbps: CCK
- 2 Mbps: DQPSK
- 1 Mbps: DBPSK

Range

- Indoors – per cell, up to 230 feet
- Outdoors – per cell, up to 984 feet

Diagnostic LED

- Power

Media Access Control

- CSMA/CA with ACK

Current Consumption

- 350mA

Operating Voltage

- 5.0V ± 5%

Transmit Power

- 13dBm @ Nominal Temp Range

Receive Sensitivity

Nominal Temp Range

- 11 Mbps 10-5 BER @-80 dBm, minimum

Network Architecture

- Supports Ad-Hoc Mode (Peer-to-Peer without Access Point) or Infrastructure Mode (Communications to wired networks via Access Points with Roaming)

3Com USB 3CRSHEW696. Se utilizaron 3 tarjetas 3com, las características de estas tarjetas son las siguientes:

Tabla 17. Características generales.

Feature	Benefit
Reliable	
Dynamic rate shifting	Speeds dynamically shift between 11, 5.5, 2, and 1 Mbps, in busy conditions, to achieve the fastest possible connections.
Range	Up to 100 m (328 ft) on 3CRSHPW196 and 3CRSHPW796, and up to 300 m (984 ft) on 3CRSHPW796; for indoor use only.*
Compatible	
Wi-Fi certified	Ensures interoperability with all Wi-Fi certified products from other vendors.
Secure	
Encryption	Uses 40/64-bit and 104/128-bit WEP encryption, helping to keep all of your wireless transmissions private.

Especificaciones:

Standards Conformance
Wi-Fi certified
IEEE 802.11b
USB 1.1 (for USB adapter only)

Frequency Range
Channels 1-13: 2.401-2.483 GHz
Channels 10-13: 2.451-2.483 GHz
Channels 5-7: 2.421-2.453 GHz

Data Rates
1, 2, 5.5, and 11 Mbps (supports dynamic rate shifting)

Computer Slot Type
3CRSHPW196 PC Card: Type II 16-bit PC Card
3CRSHPW796 PC Card: Type II 32-bit PC Card
3CRSHEW696: USB Port

Operating Voltage
3.0V-3.6V

Security
40/64-bit and 104/128-bit WEP encryption

Drivers/Supported Operating Systems
NDIS 5: Windows 2000, Me, 98 SE
NDIS 5.1: Windows XP

Safety and Electromagnetic Conformance
Safety: UL/CSA 60950, EN 60950
Radio: FCC Part 15.247, RSS-210, EN 300 328-2
EMC: FCC Part 15 Subpart B, EN 301 489-17
SAR: FCC OET Bulletin 65, RSS-102, prEN 50371

Physical Dimensions
3CRSHPW796: 113.3 mm x 54 mm x 5 mm
3CRSHPW196 (retracted): 85.6 mm x 54 mm x 5 mm
3CRSHPW196 (extended): 105.2 mm x 54 mm x 5 mm
3CRSHEW696: 100 mm x 71 mm x 20 mm

Environmental Operating Ranges
Temperature: 0 to 50° C
Humidity: 10 to 95%

Status LEDs
PC Cards
Power=solid on
Activity=fast blink

USB Adapter
Power=solid on
Activity=blink

Warranty
One-year hardware warranty