ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE POSGRADO DE INGENIERÍA Y CIENCIAS

DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ VISUAL EN LA PLATAFORMA WINDOWS PARA FACILITAR EL ANÁLISIS DE SIMULACIONES DE REDES MÓVILES GENERADAS POR NETWORK SIMULATOR

TESIS PREVIA A LA OBTENCIÓN DEL GRADO DE MAGÍSTER (MSc) EN INGENIERÍA ELÉCTRICA, MENCIÓN CONECTIVIDAD Y REDES DE TELECOMUNICACIONES

ÁLVARO MARCELO GUTIÉRREZ BURBANO

DIRECTOR: MSc. MIGUEL HINOJOSA

Quito, mayo 2007

DECLARACIÓN

Yo, Álvaro Marcelo Gutiérrez Burbano, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido en le Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Álvaro Marcelo Gutiérrez Burbano

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Álvaro Marcelo Gutiérrez Burbano, bajo mi supervisión.

MSc. Miguel Hinojosa
DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Quiero expresar mi agradecimiento a las personas que contribuyeron al desarrollo y ejecución de esta tesis.

Al MSc. Miguel Hinojosa por sus criterios y recomendaciones que contribuyeron a un desarrollo eficaz en esta tesis.

A la MSc. María Soledad Jiménez por su visión en el campo de las redes y las telecomunicaciones.

DEDICATORIA

Dedico el trabajo realizado en el desarrollo de esta tesis:

A mí hijo Christian, que con sus preguntas me hacia reflexionar en lo siguiente “no es que los niños no entiendan sino que los padres a veces no sabemos explicar”.

A mi hijo Fernando por su ternura y paciencia que me dedico en todo el tiempo que yo dedique a esta tesis.

A mi esposa Lizeth por su apoyo constante y decidido en todas las actividades que realicé desde el inicio de la Maestría.

A mi padres Segundo y Beatriz por la constancia y paciencia que me supieron transmitir con sus enseñanzas.

A mis hermanos Geovanny, Jairo y Danny quienes supieron darme su apoyo en todas las actividades que he realizado durante toda mi vida.

CONTENIDO

LISTADO DE FIGURAS

RESUMEN

El presente trabajo muestra el diseño e implementación de una interfaz visual para el Network Simulator.

En el primer capítulo se realiza una descripción de las redes móviles y sus protocolos de enrutamiento con sus características, se analiza el modelo de movilidad, componentes de red: nodos, agentes y aplicaciones.

En el segundo capítulo se define la metodología de desarrollo de software para la interfaz visual de simulación de redes, descripción de la interfaz con otros elementos del sistema y el establecimiento de restricciones y limitaciones.

En el tercer capítulo se realiza los pasos del análisis orientado a objetos: definición de los casos de uso, diagramas de los casos de uso, definición del modelo conceptual y glosario.

En el cuarto capítulo se elabora los diagramas de secuencia, diagramas de colaboración y diagramas de estado del sistema. Se elabora el modelo de clases del sistema, el mapeo a entidad relación y diseño de interfaces tanto del modelo de navegación del sistema y el modelo visual de interfaces.

En el quinto capítulo se indica las principales características de Visual Studio .Net, su arquitectura y bibliotecas básicas, así como las ventajas e inconvenientes de esta plataforma. Se muestra estándares de programación y establecimiento de librerías.

En el último capítulo se diseña casos de prueba asociados a los caso de uso según la metodología RUP. Verificación y validación. Además se indicará recomendaciones de software y de hardware.

DESCRIPCIÓN: REDES, SIMULACIÓN, DESARROLLO.

PRESENTACIÓN

Hoy en día las redes móviles ad-hoc están aumentando su relevancia en el área de las comunicaciones inalámbricas. El propósito de este proyecto es mostrar como realizar una simulación de una red móvil usando los diferentes elementos que permite el Network Simulator.

El presente trabajo muestra el diseño e implementación de una interfaz visual para el Network Simulator, con el objetivo de facilitar el análisis de redes móviles generadas por dicho simulador.

Una de las mayores fuentes de investigación en el desarrollo de las telecomunicaciones es la correcta interpretación de las simulaciones. En este contexto surge la herramienta denominada Network Simulator, la cual es capaz de simular muchas situaciones reales de las redes existentes, con un nivel considerable de detalle.

Se indica también como realizar una interfaz entre la aplicación MNB y el Cygwin utilizando la última versión disponible de este emulador.

La aplicación está diseñada con herramientas visuales, fáciles de aprender y manejar, y sobre todo que tienen un enfoque centrado en el usuario.

CAPÍTULO 1

INTRODUCCIÓN A LA SIMULACIÓN DE REDES MÓVILES

1.1.- INTRODUCCIÓN

Este capítulo ofrece una breve descripción de los conceptos básicos y tecnologías asociadas con las comunicaciones móviles y la simulación de redes.

Las Redes sin infraestructura (Ad-Hoc), están formadas por hosts móviles y que pueden estar conectados entre sí arbitrariamente y de manera dinámica. Es decir, no hay ningún elemento fijo y la topología de la red puede adoptar múltiples formas siendo igual de funcional. En este tipo de redes, todos los nodos funcionan como encaminadores (routers) y se ven involucrados tanto en el descubrimiento como en el mantenimiento de rutas. Algunos ejemplos de uso de las redes Ad-Hoc son: Operaciones de emergencia de búsqueda y rescate, convenciones y análisis de datos en terrenos catastróficos. El nodo origen quiere enviar un paquete al nodo destino, pero éste está fuera del alcance de su sistema de transmisión. Es necesario que los nodos intermedios formen parte del juego y retransmitan el paquete desde origen hasta destino.

El análisis de este proyecto se encaminará a las Redes Móviles Ad Hoc (MANET) debido a que se está incrementando su relevancia en el campo de las comunicaciones inalámbricas.

La principal característica de las Redes Móviles Ad Hoc es que todos los dispositivos que forman parte de la red, además de funcionar como terminales finales, realizan también funciones de retransmisión de paquetes típicamente asociadas a routers.

Esta cualidad permite encaminar paquetes a destinos sin cobertura directa a través de otros nodos intermedios que se encuentren en la red. De este modo se

nos ofrece la posibilidad de incrementar de una manera extraordinaria la movilidad y el tamaño de una red de datos inalámbrica. La funcionalidad principal de estas redes es la de crear de una manera rápida y eficaz una red temporal en lugares carentes de una infraestructura de red. Concretamente, las principales aplicaciones para las que estas redes fueron concebidas son operaciones de búsqueda y rescate de emergencias (bomberos), entornos militares (campos de batalla), redes de sensores, creación de vehículos y carreteras inteligentes.

Sin embargo, para que lo anterior sea viable se hace necesaria la inclusión de los protocolos de encaminamiento en la red, que permiten crear las rutas hacia los destinos deseados. Los protocolos tradicionales propios de redes fijas no se adaptan bien a este tipo de entornos tan dinámicos y, por tanto, será necesario el diseño específico de protocolos para proporcionar un comportamiento eficiente a la red.

1.2.- CLASIFICACIÓN DE LOS PROTOCOLOS

Dependiendo de sus características, los protocolos pueden seguir las siguientes clasificaciones:

- **Proactivos vs. Reactivos** (o bajo demanda): ésta es la clasificación más importante. Por una parte, en los protocolos proactivos, periódicamente se envía información de encaminamiento para que en cualquier momento cualquier nodo pueda comunicarse con cualquier otro de la red. Esta característica proporciona una rápida respuesta ante solicitudes de ruta y ofrece un buen comportamiento en situaciones donde la tasa de movilidad es alta. Sin embargo la sobrecarga que se introduce en la red con información de control es alta. Entre estos protocolos se destaca el protocolo **DSDV** (*Destination-Sequenced Distance Vector*). Por otra parte, los protocolos reactivos sólo crean rutas cuando es necesario. Son protocolos bajo demanda donde la sobrecarga es mucho menor, pero los retrasos de establecimiento de rutas son mayores. Se puede nombrar

AODV (*Ad hoc On-Demand Distance Vector*) como protocolo reactivo. Existen algunos protocolos híbridos en los que se mantiene una filosofía proactiva en un ámbito local y reactiva a nivel más global, como el protocolo **ZRP** (*Zone Routing Protocol*).

- **Jerárquico vs. Plano**: en los protocolos jerárquicos, los nodos pertenecen a diferentes niveles y su función en la retransmisión depende del nivel en el que esté. Normalmente las redes se dividen en grupos de nodos llamados *clusters*. En los planos, todos los nodos están al mismo nivel, tienen las mismas funciones y responsabilidades. Uno de los protocolos jerárquicos más destacables es **CGSR** (*Clusterhead GatewaySwitch Routing*).

- **Geográficos vs. No Geográficos**: en los protocolos geográficos, se tiene en cuenta la posición geográfica exacta de cada nodo para realizar los encaminamientos. Su gran inconveniente es la necesidad de dispositivos de posicionamiento global (GPS) todavía no lo suficientemente comunes en todos los dispositivos. Entre estos protocolos se encuentra **DREAM** (*Distance Routing Effect Algorithm for Mobility*).

- **Encaminamiento en el origen vs. Encaminamiento salto a salto**: en los protocolos con encaminamiento en el origen, cada paquete de datos lleva incorporado el camino que debe seguir en la red hasta el destino. Por esto, la necesidad de proceso en los nodos intermedios es prácticamente nula, en cambio se incrementa el volumen de los paquetes. Por el contrario, en el encaminamiento salto a salto, en cada nodo únicamente se decide cuál será el nodo siguiente. El protocolo más conocido y extendido que utiliza encaminamiento en el origen es **DSR** (*Dynamic Source Routing*).

- ***Multipath* vs. *Singlepath***: los protocolos *singlepath* sólo mantienen una ruta única hacia cada destino, en cambio, los *multipath* mantienen varias rutas. El protocolo **TORA** (*Temporally-Ordered Routing Algorithm*) es un ejemplo de protocolo *multipath*.

1.3.- DESCRIPCIÓN DE LOS PROTOCOLOS

Con el fin de ofrecer una base teórica al estudio práctico realizado, se presenta brevemente las características más relevantes de los protocolos analizados en las simulaciones:

1.3.1.- *DESTINATION-SEQUENCED DISTANCE VECTOR* (DSDV)

DSDV significa “Vector Distancia de Destino Secuencial” y mantiene la información global de la topología en forma de tablas en cada nodo. El protocolo DSDV es esencialmente una modificación del algoritmo de encaminamiento Vector Distancia de Bellman-Ford, bien conocido por su utilidad en redes fijas, como por ejemplo en el protocolo RIP.

En este algoritmo (proactivo), los nodos vecinos intercambian periódicamente sus tablas de encaminamiento, cada nodo mantiene una tabla que contiene la distancia más corta y el primer nodo en la ruta hacia cada nodo en la red. Incorpora actualizaciones de la tabla con números de secuencia que se incrementan para prevenir lazos, el problema de cuenta hacia el infinito y para una mayor convergencia.

Como es un protocolo de enrutamiento basado en tablas, todos los destinos están disponibles para cada nodo todo el tiempo. Las tablas se intercambian entre vecinos en intervalos regulares de tiempo con el fin de tener la topología de red actualizada. Las tablas se transmiten también si un nodo observa un cambio significativo en la topología local. Las tablas de actualización son de dos tipos:

a) Tabla de actualización completa (más grande). Utiliza varios paquetes de datos de red, y se usa cuando el nodo observa cambios significativos en la topología o cuando una actualización incremental requiere más de un paquete de datos.

b) Tabla de actualización incremental (más pequeño). Toma un solo paquete de datos de red y se usa cuando un nodo no observa cambios significativos en la topología local.

Las tablas de actualización son iniciadas por un destino, con un nuevo número de secuencia que siempre será mayor que el anterior. Luego de recibir una tabla de actualización un nodo, decide si actualizar sus tablas con la información recibida o mantener su información anterior por un tiempo hasta seleccionar la mejor métrica disponible.

1.3.2. - *TEMPORALLY-ORDERED ROUTING ALGORITHM* (TORA)

TORA significa “Algoritmo de Enrutamiento de Ordenamiento Temporal”. El protocolo TORA fue propuesto por Vincent D. Park y M. Scott Corson. Este algoritmo está diseñado para minimizar la reacción a cambios en la topología de la red. Desde el punto de vista de los autores, un algoritmo bien diseñado para ambientes de redes ad-hoc debe contar con las siguientes propiedades:

- Ejecutarse distribuidamente
- Proporcionar rutas libres de ciclos
- Proporcionar múltiples rutas (para aliviar el congestionamiento)
- Establecer rutas rápidamente (para que puedan ser utilizadas antes de que cambie la topología)
- Minimizar la sobrecarga localizando reacciones algorítmicas a cambios en la topología cuando sea posible (para mantener el ancho de banda e incrementar la escalabilidad)

El funcionamiento del protocolo TORA es de la siguiente manera: cada vez que se requiera una ruta a nodo destino se ejecuta una versión lógica separada del protocolo. El protocolo puede separarse en tres funciones básicas: creación de rutas, manutención de rutas y borrado de rutas.

Para la creación de la ruta de un nodo dado al destino, se requiere el establecimiento de una secuencia de ligas dirigidas que conduzcan el nodo al destino. Esta función sólo se inicia cada vez que un nodo sin ligas dirigidas requiere una ruta a un destino. Así se puede decir que la creación de rutas consiste en asignar direcciones a las ligas de una red o porción de una red. Para lograr esto se construye un grafo acíclico dirigido con raíz en el nodo destino.

La manutención de rutas se refiere a reaccionar a cambios en la topología de la red de forma tal que las rutas hacia el destino puedan ser re-establecidas en un tiempo finito.

Al momento de detectarse una partición de la red, todas las ligas (en la porción de la red que ha quedado particionada desde el destino) deben quedar sin dirección.

El protocolo logra estas tres funciones mediante el uso de tres paquetes de control distintos: de solicitud (QRY), de actualización (UPD), y de limpieza (CLR). Los paquetes QRY son usados para crear rutas, los paquetes UPD son usados tanto para crear rutas como para mantenerlas y los paquetes CLR son utilizados para borrar rutas.

TORA asocia una altura a cada nodo en la red. Todos los mensajes fluyen de nodos con mayor altura hacia nodos con menor altura. Cuando un nodo no tiene algún vecino con una altura menor a la de él mismo, entonces envía un mensaje QRY a toda la red. Este mensaje QRY se propaga por toda la red hasta encontrar un nodo que tenga una ruta al destino deseado o bien, hasta que el paquete llegue al destino. Entonces ese nodo genera un paquete UPD que contiene la altura del nodo. El paquete UPD es enviado a todos los nodos de la red y conforme los nodos vayan recibiendo este paquete, se debe ir actualizando sus propias alturas con valores mayores que el especificado en el mensaje UPD. A su vez cada nodo que recibe un paquete UPD envía su propio mensaje UPD a los demás nodos de la red. Con esta serie de acciones se obtiene un número de ligas dirigidas que van desde el nodo que originó el paquete QRY hasta el nodo destino.

1.3.3.- *DYNAMIC SOURCE ROUTING* (DSR)

DSR significa “Enrutamiento de Origen Dinámico”. El protocolo DSR es un protocolo sobre demanda; diseñado para restringir el consumo de ancho de banda ocasionado por los paquetes de control de redes Ad Hoc, al eliminar los mensajes periódicos de actualización de tablas requeridos por los modelos basados en tablas. La mayor diferencia entre este y otros protocolos sobre demanda, es que este no usa transmisiones periódicas de paquetes, los cuales suelen usarse para informar a un nodo de la presencia de un vecino. En la fase de establecimiento de la ruta, un nodo que desea establecer una ruta hacia un destino inunda la red con paquetes *Request*. El nodo al recibir el paquete *Request* responde al origen con un paquete *Reply* el cual informa la ruta que ha atravesado el paquete *Request*.

Considerando que el nodo origen no tiene una ruta hacia el destino, lo primero que hace es enviar paquetes *Request* hacia la red. Cada nodo al recibir el paquete *Request*, retransmite este paquete hacia sus vecinos, mientras el tiempo de vida del paquete TTL no se ha excedido. Cada paquete *Request* lleva un número de secuencia generado por el nodo origen y almacena la ruta que va cruzando. Un nodo luego de recibir un paquete *Request* revisa el número de secuencia antes de pasarlo para evitar transmitirlo si el paquete esta duplicado.

Además el número de secuencia es usado para prevenir la formación de lazos y evitar múltiples transmisiones del mismo paquete por un nodo intermedio que lo recibe a través de múltiples caminos. De esta manera todos los nodos excepto el destino transmiten el paquete *Request* durante la construcción de la ruta. Un nodo destino luego de recibir el primer paquete *Request*, responde hacia el origen a través del camino inverso que el paquete ha atravesado.

Cuando un nodo intermedio se mueve o deja de funcionar causando una falla en el enlace, se genera un mensaje Error, desde el nodo adyacente al enlace roto, para informar al nodo origen. El nodo origen reinicia el establecimiento de ruta y el

caché de rutas en los nodos intermedios se borra una vez que un paquete Error se ha recibido.
El protocoló DSR utiliza un modelo reactivo que elimina la necesidad de envió de paquetes periódicos a la red con los mensajes de actualización de tablas que requieren los protocolos basados en tablas. Los nodos intermedios utilizan un caché con información de rutas para reducir eficientemente los paquetes de control.

El protocolo DSR presenta la desventaja de que el mecanismo de mantenimiento de rutas no repara localmente una falla en el enlace; además un caché de rutas con información anterior puede conducir a inconsistencias durante la fase de reconstrucción de rutas. El retardo por configuración de la conexión es mayor que el de protocolos basados en tablas.

Aunque el protocolo tiene un buen desempeño en ambientes estáticos o de baja movilidad, el desempeño decae rápidamente si se incrementa la movilidad de los nodos, además de que se halla gran cantidad de paquetes de control debido al mecanismo de descubrimiento de rutas. Estos paquetes de control son directamente proporcionales al largo de la ruta.

1.3.4. - *AD HOC ON-DEMAND DISTANCE VECTOR* (AODV)

AODV significa “Vector Distancia para Redes Ad Hoc”. AODV emplea números de secuencia para identificar las rutas más frecuentes, el nodo origen y los nodos intermedios almacenan la información del siguiente salto, correspondiente a cada flujo para transmisiones de paquetes de datos. En un protocolo de enrutamiento sobre demanda, el nodo origen inunda la red con un paquete *Request*, cuando una ruta no está disponible para un destino dado. El nodo puede obtener múltiples rutas hacia diferentes destinos de un solo paquete *Request*.

La mayor diferencia entre AODV y otros protocolos de enrutamiento sobre demanda, es que éste usa un número de secuencia hacia el destino para determinar una ruta actualizada hacia ese destino. Un nodo actualiza su

información de enrutamiento solamente si el número de secuencia es mayor que el último número almacenado para ese nodo.

El paquete *Request* lleva un identificador de origen, identificador del destino, número de secuencia del origen, número de secuencia del destino, identificador de *broadcast* y tiempo de vida.

1.4.- MODELO DE MOVILIDAD DE NETWORK SIMULATOR

Para realizar las simulaciones se ha utilizado la herramienta de simulación de redes NS-2. Se ha utilizado los protocolos incluidos en la versión 2.31: DSDV, DSR, TORA y AODV.

Esta sección describe el modelo inalámbrico que fue originalmente propuesto como extensión a la movilidad de *Network Simulator* por el grupo *CMU Monarch*.

Esta sección describe:

- La red móvil
- El interior de un nodo móvil,
- Los mecanismos de enrutamiento
- y componentes de red que son usados para construir la estructura de pila de red para un nodo móvil.

Los componentes que se modelan son:

- Canal (*Channel*)
- Interfaz de red (*Network interface*)
- Modelo de propagación de ondas de radio (*Radio propagation model*)
- Protocolos MAC (*MAC protocols*)
- Cola de la Interfaz (*Interface Queue*)
- Capa de enlace (*Link Layer*)

- Modelo ARP (*Address resolution protocol model*)

1.4.1.- INICIALIZACIÓN Y TERMINACIÓN DE NS

Una simulación de NS inicia con:

set ns [new Simulator]

Ésta línea declara una nueva variable denominada “ns”, se la puede denominar con cualquier nombre pero generalmente se usa ese nombre, se instancia un objeto denominado “ns” de la clase “*Simulator*”.

Para obtener archivos de datos de salida con datos de la simulación (archivos de rastro-trace), o archivos usados para visualización (archivos nam), se necesita crearlos usando el comando “*open*”. El comando “*open*” con el atributo “w” indica que se crea el archivo y lo abre para escritura o en el caso de existir el archivo el comando “*open*” sobrescribe el contenido del mismo. El primer atributo de este comando indica el nombre del archivo.

El archivo de trace de datos (.tr) presenta información de los paquetes generados por el Network Simulator en un archivo de texto. De esta manera se puede leer y procesar esta información para obtener información estadística con respecto a la simulación realizada.

El archivo de trace de datos para Nam (.nam) presenta información de los paquetes generados por el Network Simulator en un archivo de texto. De esta manera el utilitario Nam puede leer esta información y mostrar los eventos de la simulación realizada en forma gráfica.

```
#Abrir un archivo trace(archivo con datos)
set tracefile1 [open out.tr w]
$ns trace-all $tracefile1
```

```
#Abrir un archivo de visualizacion NAM
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

La terminación del programa es realizada usando el procedimiento “finish”.

```
#Definir un procedimiento de fin
proc finish {} {
        global ns tracefile1 namfile
        $ns flush-trace
        close $tracefile1
        close $namfile
        exec nam out.nam &
        exit 0
}
```

La palabra “proc” declara un procedimiento en este caso denominado “*finish*” sin argumentos. La palabra “*global*” se usa para usar variables que están declaradas fuera del procedimiento. El método “*flush-trace*” transfiere los datos de los *buffers* a los archivos. El comando tcl “*close*” cierra los archivos. El comando “*exec*” ejecuta un comando, en este caso ejecuta el “nam” pasando de parámetro el archivo “out.nam”. El comando “*exit*” finaliza la aplicación y retorna el valor 0 al sistema.

Al final del programa NS se debe llamar al procedimiento “*finish*” y especificar el tiempo en el que la terminación debe ocurrir:

```
#Especificar tiempo de terminación y procedimiento de fin
$ns at 125.0 "finish"
```

El método “*at*” de Simulator permite planificar eventos en forma explícita. En este caso se llamará a “*finish*" al tiempo 125 sec.

Entonces, la simulación puede empezar con el comando:

```
#Comando para correr simulación
$ns run
```

1.4.2.- DEFINICIÓN DE PARÁMETROS

Cuando se desea definir parámetros que se requieren utilizar en varias partes del código en un script TCL, podemos hacerlo de la siguiente manera:

```
set opt(MobileNodes) 3
set opt(SizeTopgraphyX) 501
set opt(SizeTopgraphyY) 501
```

1.4.3.- EL MODELO INALÁMBRICO BÁSICO:

Esencialmente el modelo inalámbrico consiste de un nodo móvil (*MobileNode*) como punto de partida y de características adicionales de apoyo que permiten simulaciones de *multi-hop ad-hoc networks* y LANs inalámbricas. La clase MobileNode hereda las características de la clase Node. Para detalles de Node refierase a “The NS Manual”, The VINT Project, Capítulo 5.

Un MobileNode es entonces un objeto Node básico con funcionalidades añadidas de un nodo inalámbrico y móvil como:

- la habilidad de moverse dentro de una topología dada,
- habilidad de recibir y transmitir señales a y desde un canal inalámbrico,
- actualización de la posición periódicamente

La mayor diferencia entre ellos, es sin embargo que un MobileNode no está conectado por medio de Links a otros nodos o nodos móviles.

En esta sección describiremos el interior de un MobileNode, sus mecanismos de enrutamiento, los protocolos de enrutamiento dsdv, aodv, tora y dsr, creación de la pila de red permitiendo acceso al canal, breve descripción de cada componente de pila, soporte para trazas y generación de escenario de movimiento/tráfico para simulaciones inalámbricas.

1.4.4.- NODO MÓVIL: CREACIÓN DE TOPOLOGÍA INALÁMBRICA:

La funcionalidad descrita en esta sección puede ser encontrada en:

- ~ns/common/mobilenode.cc
- ~ns/common/mobilenode.h
- ~ns/tcl/lib/ns-mobilenode.tcl
- ~ns/tcl/mobility/dsdv.tcl
- ~ns/tcl/mobility/dsr.tcl
- ~ns/tcl/mobility/tora.tcl

Ejemplos pueden ser encontrados en

- ~ns/tcl/ex/wireless-test.tcl
- ~ns/tcl/ex/wireless.tcl

El primer ejemplo usa topología pequeña de 3 nodos, el segundo ejemplo usa una topología de 50 nodos. Estos ejemplos pueden ejecutarse digitando:

- $ns tcl/ex/wireless-test.tcl
- $ns tcl/ex/wireless.tcl

Los cuatro protocolos de enrutamiento que actualmente están configurados son:

- Destination Sequence Distance Vector (DSDV)

- Dynamic Source Routing (DSR)
- Temporally ordered Routing Algorithm (TORA)
- Adhoc On-demand Distance Vector (AODV)

La antigua forma de crear un MobileNode depende del protocolo usado, y es la siguiente:

set mnode [$opt(rp)-create-mobile-node $id]

donde $opt(rp) define “dsdv”, “aodv”, “tora” o “dsr”.

La nueva API para crear un nodo móvil es la siguiente:

```
$ns_ node-config
        -addressingType <usually flat or hierarchical used for
                        wireless topologies>
        -adhocRouting <adhoc rotuing protocol like DSDV, DSR,
                        TORA, AODV etc>
        -llType <LinkLayer>
        -macType <MAC type like Mac/802_11>
        -propType <Propagation model like
                        Propagation/TwoRayGround>
        -ifqType <interface queue type like
                        Queue/DropTail/PriQueue>
        -ifqLen <interface queue length like 50>
        -phyType <network inteface type like
                        Phy/WirelessPhy>
        -antType <antenna type like Antenna/OmniAntenna>
        -channelType <Channel type like Channel/WirelessChannel>
        -topoInstance <the topography instance>
        -wiredRouting <turning wired routing ON or OFF>
        -mobileIP <setting the flag for mobileIP ON or OFF>
```

-energyModel <EnergyModel type>
-initialEnergy <specified in Joules>
-rxPower <specified in W>
-txPower <specified in W>
-agentTrace <tracing at agent level turned ON or OFF>
-routerTrace <tracing at router level turned ON or OFF>
-macTrace <tracing at mac level turned ON or OFF>
-movementTrace <mobilenode movement logging turned
ON or OFF>

Lo anterior configura un nodo móvil con todos los valores dados: protocolo de enrutamiento ad-hoc, pila de red (consistente de capa de enlace, cola de la interface, capa MAC e interface de red con una antena), canal, topografía, modelo de propagación, manejo de trasas activado o desactivado en diferentes niveles: enrutamiento, mac, agente.

1.4.5.- CREACIÓN DE MOVIMIENTOS DE UN NODO

El nodo móvil esta diseñado para moverse en una topología tri-dimensional. Sin embargo la tercera dimensión no se usa. Esto significa que la posición en Z siempre es igual a 0. Las coordenadas son continuamente ajustadas a los movimientos del nodo. Existen dos mecanismos para inducir movimientos en nodos móviles.

- En el primer método la posición de inicio del nodo y su destino futuro debe ser explícito. Estas directivas son normalmente incluidas en archivo de escenario de movimientos por separado. Se usan las siguientes APIs:
  $node set X_ <x1>
  $node set Y_ <y1>
  $node set Z_ <z1>
  $ns at $time $node setdest <x2> <y2> <speed>

- El segundo método emplea movimiento aleatorio y tiene actualización rutinaria para cambiar la dirección y velocidad del nodo.

Sin importar ninguno de los dos métodos usados para generar movimiento, se necesita definir una topología. Esta debe ser definida antes de crear los nodos. Se usan las siguientes APIs:

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

Los movimientos de los nodos móviles pueden ser anotados usando un procedimiento como el siguiente:

```
proc log-movement {} {
    global logtimer ns_ ns
    set ns $ns_
    source ../mobility/timer.tcl
    Class LogTimer -superclass Timer
    LogTimer instproc timeout {} {
    global opt node_;
    for {set i 0} {$i < $opt(nn)} {incr i} {
    $node_($i) log-movement
    }
    $self sched 0.1
    }
    set logtimer [new LogTimer]
    $logtimer sched 0.1
}
```

1.4.6.- COMPONENTES DE RED EN UN NODO MÓVIL

La pila de red en un nodo móvil consiste de una capa de enlace (LL Link Layer), un módulo ARP conectado a LL, una cola de prioridad de interface, una capa

MAC, una interfaz de red (netIF), todo conectado al canal. Estos componentes de red son creados y unidos en OTcl.

Cada componente es descrito brevemente.

***Link Layer*:** el LL usado en el nodo móvil tiene la misma funcionalidad que el usado en el nodo de una red LAN y que está descrito en el capítulo 14 del Manual de NS. La única diferencia es que el nodo móvil tiene un módulo ARP conectado a el para resolver todos las conversiones de dirección de máquina IP.

***ARP*:** el módulo *Address Resolution Protocol* recibe requerimientos de la capa de enlace. Si ARP tiene la dirección de hardware para el destino, este lo escribe en la cabecera MAC del paquete. En otro caso este lo transmite en un requerimiento ARP y guarda en cache el paquete en forma temporal. Para cada dirección de hardware desconocida, hay un buffer para un solo paquete. En caso de existir paquetes adicionales al mismo destino es enviado a ARP, el paquete en buffer que llego primero es borrado. Una vez que la dirección de hardware de un paquete en el siguiente salto es conocida, el paquete es ingresado en la cola de interfaz.

***Interface Queue*:** la clase PriQueue es implementada como una cola de prioridades, en la cual se da prioridad más alta a paquetes de protocolo de enrutamiento, insertándolos en la cabeza de la cola.

Mac Layer: la función de coordinación distribuida DCF del protocolo Mac IEEE 802.11 ha sido implementada por CMU. Este usa un patrón RTS/CTS/DATA/ACK para todos los paquetes *unicast* y simplemente envía “out DATA” para todos los paquetes *broadcast*.

***Network Interfaces*:** la capa de interfaz de red sirve como una interfaz de hardware la cual es usada por MobileNode para acceder al canal. La interfaz compartida inalámbrica es implementada con la clase Phy/WirelessPhy.

***Radio Propagation Model*:** esta usa atenuación "*Friss-space*" para distancias cercanas y una aproximación a "*Two ray Ground*" para distancias lejanas.

***Antenna*:** una antena omni-direccional que tiene ganancia unitaria es usada por MobileNode.

CAPÍTULO 2

REQUERIMIENTOS

2.1.- DEFINICIÓN DE METOLOGÍA DE DESARROLLO DE SOFTWARE

La metodología que se usará para el desarrollo del software es RUP que significa “Proceso Unificado Rational" por sus siglas en inglés “*Rational Unified Process*”.

El RUP es una filosofía de desarrollo de software que detalla con exactitud la forma de crear aplicaciones, siguiendo pasos, estableciendo estándares, proponiendo métodos de trabajo, y presentando en esquemas como se debe, se puede y se necesita diseñar el software.

En general, en muchas empresas desarrolladoras de software, lo que se pretende es hacer programas sin la necesidad de un proceso desarrollado que permite sostener las aplicaciones realizadas, RUP presenta un modelo de trabajo que toma en cuenta desde el inicio del proyecto, su análisis, diseño, desarrollo, implementación, pruebas, actualizaciones y nuevas versiones.

En analogía con la Ingeniería Civil o la Arquitectura, no se puede construir un edificio, sin hacer un completo estudio de suelos, diseño de estructuras, y toma de riesgos en la construcción, de la misma manera al crear software se debe contemplar muchos aspectos de estudio, diseño y desarrollo que ayudarán ante todo a obtener un producto de calidad sostenible en el tiempo, sin importar que tipo de software se desee crear.

El RUP es resultado de una conjunción de algunos métodos de diseño, es por eso que parte de una evolución de anteriores procesos que han permitido entregar un estándar clave para los desarrolladores

Los marcos de referencia del RUP definen qué debe hacerse y en algunos casos cómo hacerlo, además consiste en ajustar este marco de referencia a la organización y realizar el proceso de apropiación de la metodología y asimilación de las guías de trabajo al interior del equipo del proyecto y del área de informática.

Teniendo en cuenta los aspectos mencionados previamente, La empresa Rational que recientemente fue comprada por IBM, elaboró RUP como marco de referencia para el proceso de desarrollo de software basado en el modelo en espiral.

2.2.- CLIENTES

Analistas con conocimientos de redes móviles ad-hoc.

2.3.- DEFINICIÓN DEL PROBLEMA

Se requiere crear una interfaz que permita al analista definir redes móviles ad-hoc, establecer sus parámetros, invocar al *Network Simulator* y visualizar los resultados de la simulación obtenida.

2.4.- METAS

En términos generales, la meta es lograr un análisis de la simulación de redes móviles en forma más amigable para el analista. Más concretamente, la meta incluye:

- Establecer las condiciones y parámetros que debe cumplir el proceso de simulación de redes móviles ad-hoc en Network Simulator.

- Crear una Interfaz que permita definir la topología de una red móvil ad-hoc, definiendo características de la red, nodos, movimientos y conexiones.

- Generar información de entrada para el Network Simulator en base a la topología de red definida.

- Visualizar las tramas resultantes del proceso de simulación.

2.5.- FUNCIONES DEL SISTEMA

El sistema a desarrollar deberá permitir:

- Establecer parámetros

    - Parámetros de la red
    - Parámetros de los nodos
    - Parámetros de conexión de agentes
    - Parámetros de movimiento

- Definir topología de red

    - Características de la red
    - Número de nodos
    - Características de los nodos
    - Conexiones de los agentes
    - Movimientos de los nodos
    - Guardar información XML

- Generar Entrada a NS

    - Recuperar información XML
    - Generar Archivos de Entrada para NS

- Visualizar Simulación de NS

    - Invocar a NS
    - Recolectar archivos de salida de NS
    - Inspeccionar o Visualizar el proceso de simulación

2.6.- CATEGORIAS DE LAS FUNCIONES

Las funciones deben clasificarse a fin de establecer prioridades entre ellas e identificarlas, que de lo contrario pasarían inadvertidas (pero consumen tiempo y otros recursos). Las categorías son:

CATEGORÍA DE LA FUNCIÓN	SIGNIFICADO
EVIDENTE	Debe realizarse, y el usuario debería saber que se ha realizado.
OCULTA	Deber realizarse, aunque no es visible para los usuarios. Esto se aplica a muchos servicios técnicos subyacentes, como guardar información en un mecanismo persistente de almacenamiento. Las funciones ocultas a menudo se omiten (erróneamente) durante el proceso de obtención de los requerimientos.
SUPERFLUA	Opcionales; su inclusión no repercute significativamente en el costo ni en otras funciones.

2.7.- FUNCIONES BÁSICAS

Las funciones del sistema se categorizan de la siguiente manera:

- **R1. Establecer parámetros**

REF. #	FUNCIÓN	CATEGORÍA
R1.1	Parámetros de la red	OCULTA
R1.2	Parámetros de los nodos	OCULTA
R1.3	Parámetros de conexión de agentes	OCULTA
R1.4	Parámetros de movimiento	OCULTA

- **R2. Definir topología de red**

REF. #	FUNCIÓN	CATEGORÍA
R2.1	Características de la red	EVIDENTE
R2.2	Número de nodos	EVIDENTE
R2.3	Características de los nodos	EVIDENTE
R2.4	Conexiones de los agentes	EVIDENTE
R2.5	Movimientos de los nodos	EVIDENTE
R2.6	Guardar información XML	OCULTA

- **R3. Generar Entrada a NS**

REF. #	FUNCIÓN	CATEGORÍA
R3.1	Recuperar información XML	OCULTA
R3.2	Generar Archivos de Entrada para NS	OCULTA

- **R4. Visualizar Simulación de NS**

REF. #	FUNCIÓN	CATEGORÍA
R4.1	Invocar a NS	OCULTA
R4.2	Recolectar archivos de salida de NS	OCULTA
R4.3	Obtener resultados y visualizar la simulación	EVIDENTE

2.8.- ATRIBUTOS DEL SISTEMA

Los atributos del sistema son sus características o dimensiones; no son funciones. Los atributos del sistema pueden abarcar todas las funciones (por ejemplo la plataforma del sistema operativo) o ser específicos de una función o grupo de funciones.

ATRIBUTO	DETALLES
Metáfora de interfaz	Ventanas orientadas a la metáfora de una forma y cuadros de díalogo.
	Maximiza una navegación fácil con mouse.
Plataforma del Sistema Operativo	Windows XP, Windows 2000.

2.9.- DESCRIPCIÓN DE LA INTERFAZ CON OTROS ELEMENTOS DEL SISTEMA

El proyecto que se va ha desarrollar debe tener una interfaz directa con el Network Simulator.

Debido a que el Network Simulator está desarrollado en la plataforma LINUX, se requiere un software denominado Cygwin que proporciona una API de UNIX en la plataforma Windows.

Cygwin es un ambiente semejante a Linux para ambientes Windows. Consiste de un DLL (cygwin1.dll), el cual actúa como una capa de emulación brindando funcionalidad POSIX (Portable Operating System Interface) de llamadas al sistema y una colección de herramientas, las cuales proporcionan una forma real de ver y sentir el ambiente Linux. El DLL Cygwin trabaja con todas las versiones x86 de Windows desde Windows 95.

Una vez instalado Cygwin, los usuarios tienen acceso a la mayoría de utilitarios estándar de Linux.

2.10.- ESTABLECIMIENTO DE RESTRICCIONES Y LIMITACIONES

El software que se desarrollará en este proyecto interactuará con Network Simulator y permitirá definir topologías de redes móviles ad-hoc en un formato que el Network Simulator puede entender y proceder al proceso de simulación. El Network Simulator luego del proceso de simulación entregará un archivo que refleja el contenido de la simulación. Una vez obtenidos los resultados de esta simulación, se visualizará las tramas resultantes de esta simulación.

Este proyecto analizará la forma de definir topologías generando el formato que sirve de entrada para el Network Simulator. Facilitará el ingreso de datos en una interfaz visual. También analizará la forma de entregar la información de las simulaciones realizadas en el Network Simulator estableciendo este formato como punto de partida para el Visualizador.

Dada la complejidad que los tipos de redes encierran, las diferentes topologías existentes y nuevas que se crean en forma constante, este proyecto se limitará hacia la visualización de las simulaciones de redes móviles ad-hoc que gestiona el Network Simulator.

CAPÍTULO 3
ANÁLISIS

3.1.- INTRODUCCIÓN

3.1.1.- DEFINICIÓN DE CASO DE USO

Una técnica excelente que permite mejorar la comprensión de los requerimientos es la creación de los casos de uso, es decir, descripciones narrativas de los procesos del dominio.

Los casos de uso requieren tener al menos un conocimiento parcial de los requerimientos del sistema.

El caso de uso es un documento narrativo que describe la secuencia de eventos de un actor (agente externo) que utiliza un sistema para completar un proceso. Los casos de uso son historias o casos de utilización de un sistema; no son exactamente los requerimientos ni las especificaciones funcionales, sino que ejemplifican e incluyen tácticamente los requerimientos en las historias que narran.

3.1.2.- TIPOS DE CASOS DE USO

Los casos de uso se clasifican en primarios, secundarios y opcionales. En base a estas designaciones, se clasificarán los conjuntos de casos de uso para establecer prioridades de desarrollo.

Los casos primarios de uso representan los procesos comunes más importantes. Los casos secundarios de uso representan procesos menores o raramente

utilizados. Los casos opcionales de uso representan procesos que pueden no abordarse.

3.1.3.- CURSO NORMAL DE LOS EVENTOS

La sección denominada “Curso Normal de los eventos” en un caso de uso es la parta medular del formato expandido de casos de uso; describe los detalles de la interactividad entre los actores y el sistema. Un aspecto esencial de esta sección es que explica la secuencia más común de los eventos: la historia normal de las actividades y la terminación exitosa de un proceso. No incluye situaciones alternas.

CURSO NORMAL DE LOS EVENTOS	
ACCIÓN DEL ACTOR	**RESPUESTA DEL SISTEMA**
Acciones numeradas de los actores.	Descripciones numeradas de las respuestas del sistema.

3.1.4.- CURSO ALTERNO DE LOS EVENTOS

La última sección, curso alterno de los eventos, describe importantes opciones o excepciones que pueden presentarse en relación con el curso normal. Si son complejas, podemos expandirlas en nuevos casos de uso.

Son alternativas que pueden ocurrir en el número de línea. Descripción de excepciones.

3.1.5.- ACTORES

El actor es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso. Por lo general estimula al sistema con eventos de entrada o recepción de él. Los actores están representados por el papel que

desempeñan en el caso. Conviene escribir su nombre con mayúsculas en la narrativa del caso para facilitar la identificación.

Fig. 3.1. Actor

3.2.- CASOS DE USO

En este proyecto se han definido en los siguientes casos de uso:

3.2.1.- CASO DE USO: DEFINIR TOPOLOGÍA DE RED

Caso de uso:	**Definir topología de red**
Actores:	Analista
Propósito:	Definir la topología de una red móvil ad-hoc.
Resumen:	El analista define en el Sistema Informático, las características de la red móvil ad-hoc, define el número de nodos, las características, conexiones de agentes, movimientos de los nodos y guarda esta información en formato XML.
Tipo:	Primario y esencial
Precondiciones:	El sistema informático debe estar disponible.
Referencias cruzadas:	Funciones: R2
CURSO NORMAL DE LOS EVENTOS	
ACCIÓN DEL ACTOR	**RESPUESTA DEL SISTEMA**
1.- Este caso de uso comienza cuando el analista ingresa al sistema	

con el propósito de definir una topología de red móvil.	
	2.- El sistema informático presenta la pantalla principal y su menú.
3.- El analista selecciona la opción de definir una nueva topología de red.	
	4.- El sistema presenta una pantalla con las características de la topología de red.
5.- El analista ingresa las características de la topología de red.	
	6.- El sistema presenta una pantalla para el número de nodos.
7.- El analista ingresa el número de nodos. Ingresa además las características de cada nodo.	
	8.- El sistema presenta una pantalla para definición de conexiones entre agentes.
9.- El analista define las conexiones entre agentes.	
	10.- El sistema presenta una pantalla para definición de movimientos de los nodos.
11.- El analista define los movimientos de los nodos.	
12.- El analista indica que desea grabar esta definición.	
	13.- El sistema graba esta definición en un archivo XML.

3.2.2.- CASO DE USO: GENERAR ENTRADA PARA NS

Caso de uso:	**Generar entrada para NS**
Actores:	Analista, Sistema NS
Propósito:	Llamar a la interfaz de NS
Resumen:	Una vez definida la topología de red móvil, se necesita recuperar esta información, generar archivos de entrada para NS.
Tipo:	Primario y esencial
Precondiciones:	El sistema informático debe estar disponible. Caso de uso: Definir topología de red.
Referencias cruzadas:	Funciones: R3

CURSO NORMAL DE LOS EVENTOS	
ACCIÓN DEL ACTOR	**RESPUESTA DEL SISTEMA**
1.- Este caso de uso comienza cuando el analista ha definido la topología de la red móvil, y necesita generar el proceso de simulación con NS.	
	2.- El sistema informático pide el nombre del archivo XML.
3.- El analista selecciona el archivo XML.	
	4.- El sistema genera archivos que sirven como entrada de información para NS.
	5.- El sistema informa que la generación de la simulación se ha realizado correctamente.

3.2.3.- CASO DE USO: VISUALIZAR SIMULACIÓN DE NS

Caso de uso:	**Visualizar simulación de NS**
Actores:	Analista
Propósito:	Mostrar al Analista la simulación efectuada por el NS.
Resumen:	Una vez definida la topología de red móvil, y generado los archivos de entrada para NS, el sistema informático invoca al NS para luego visualizar la información de la simulación de la red.
Tipo:	Primario y esencial
Precondiciones:	El sistema informático debe estar disponible. Caso de uso: Definir topología de red. Caso de uso: Generar entrada para NS.
Referencias cruzadas:	Funciones: R4

CURSO NORMAL DE LOS EVENTOS	
ACCIÓN DEL ACTOR	**RESPUESTA DEL SISTEMA**
1.- Este caso de uso comienza cuando el analista ha definido la topología de la red móvil, y el NS ha generado información de salida y necesita visualizar la simulación con NS.	
	2.- El sistema invoca al NS para que realice el proceso de simulación con la información de entrada.
	3.- NS genera archivos con información de salida de la simulación.
	4.- El sistema informático recolecta los archivos de salida que produjo el NS con información de la salida.

	5.- El sistema muestra los resultados de la simulación efectuada.
	6.- El sistema informático muestra el proceso de simulación de la red móvil.
	7.- El sistema informa que la simulación se ha realizado correctamente.

3.3.- DIAGRAMAS DE LOS CASOS DE USO

En este proyecto se han definido los siguientes diagramas de casos de uso:

3.3.1.- DIAGRAMA DE CASO DE USO: DEFINIR TOPOLOGÍA DE RED

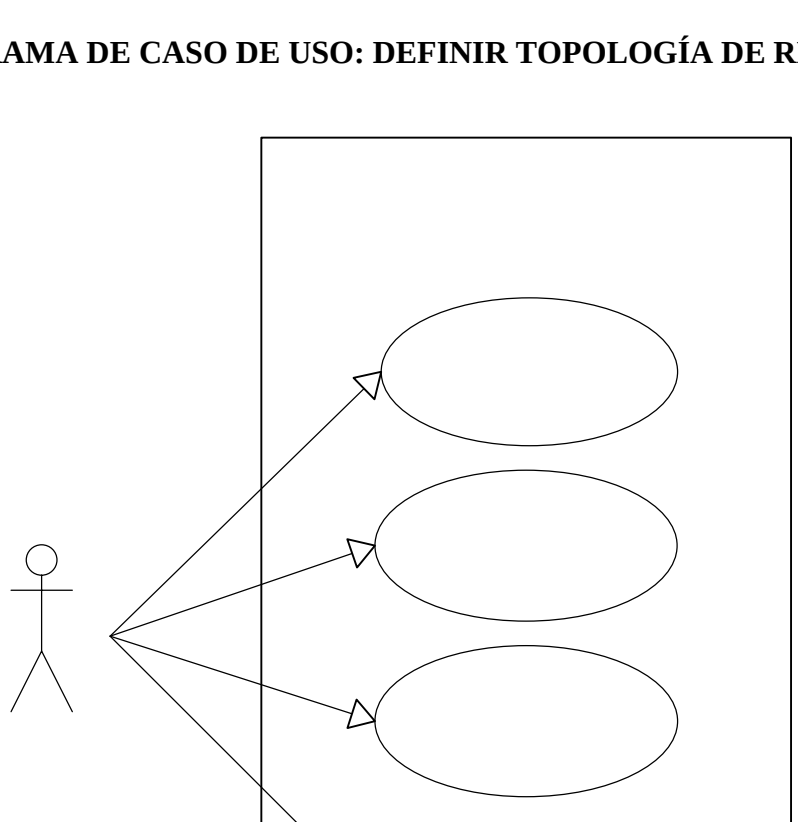

Fig. 3.2. Caso de Uso: Definir Topología de Red

3.3.2.- DIAGRAMA DE CASO DE USO: GENERAR ENTRADA PARA NS

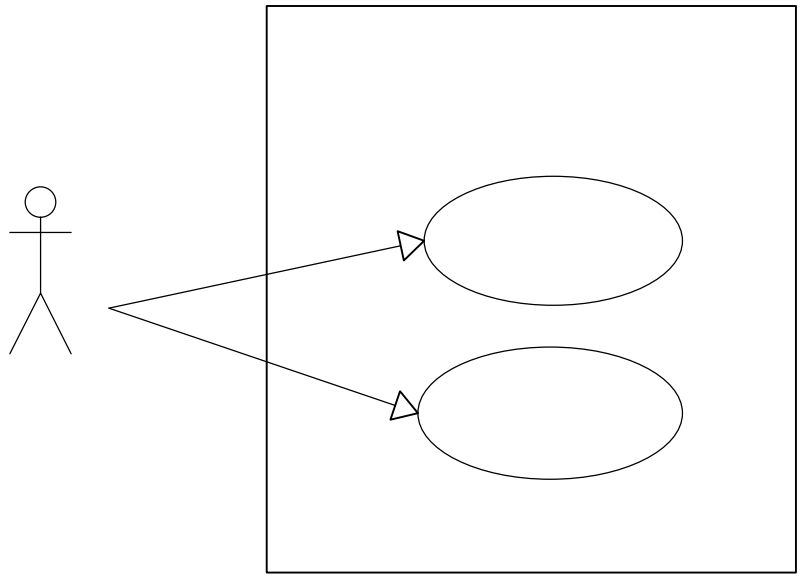

Fig. 3.3. Caso de Uso: Generar Entrada para NS

3.3.3.- DIAGRAMA DE CASO DE USO: VISUALIZAR SIMULACIÓN DE NS

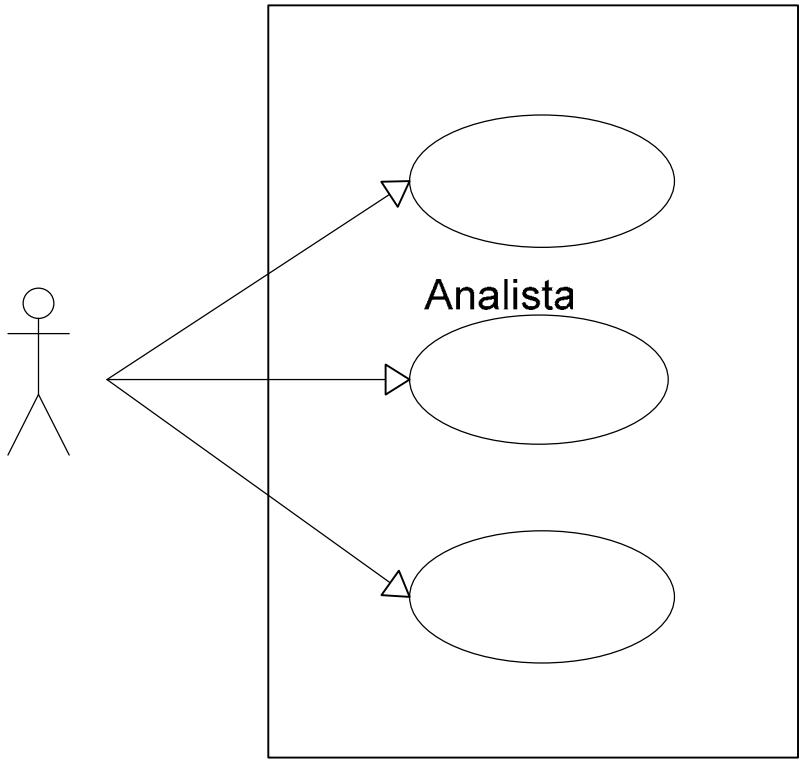

Fig. 3.4. Caso de Uso: Visualizar Simulación de NS

3.4.- MODELO CONCEPTUAL

Un modelo conceptual explica los conceptos significativos en un dominio del problema.

Identificar objetos o conceptos constituye la esencia del análisis orientado a objetos, y el esfuerzo se compensa con los resultados conseguidos durante la fase de diseño e implementación.

Una cualidad esencial que debe ofrecer un modelo conceptual es que representa cosas del mundo real, no componentes del software.

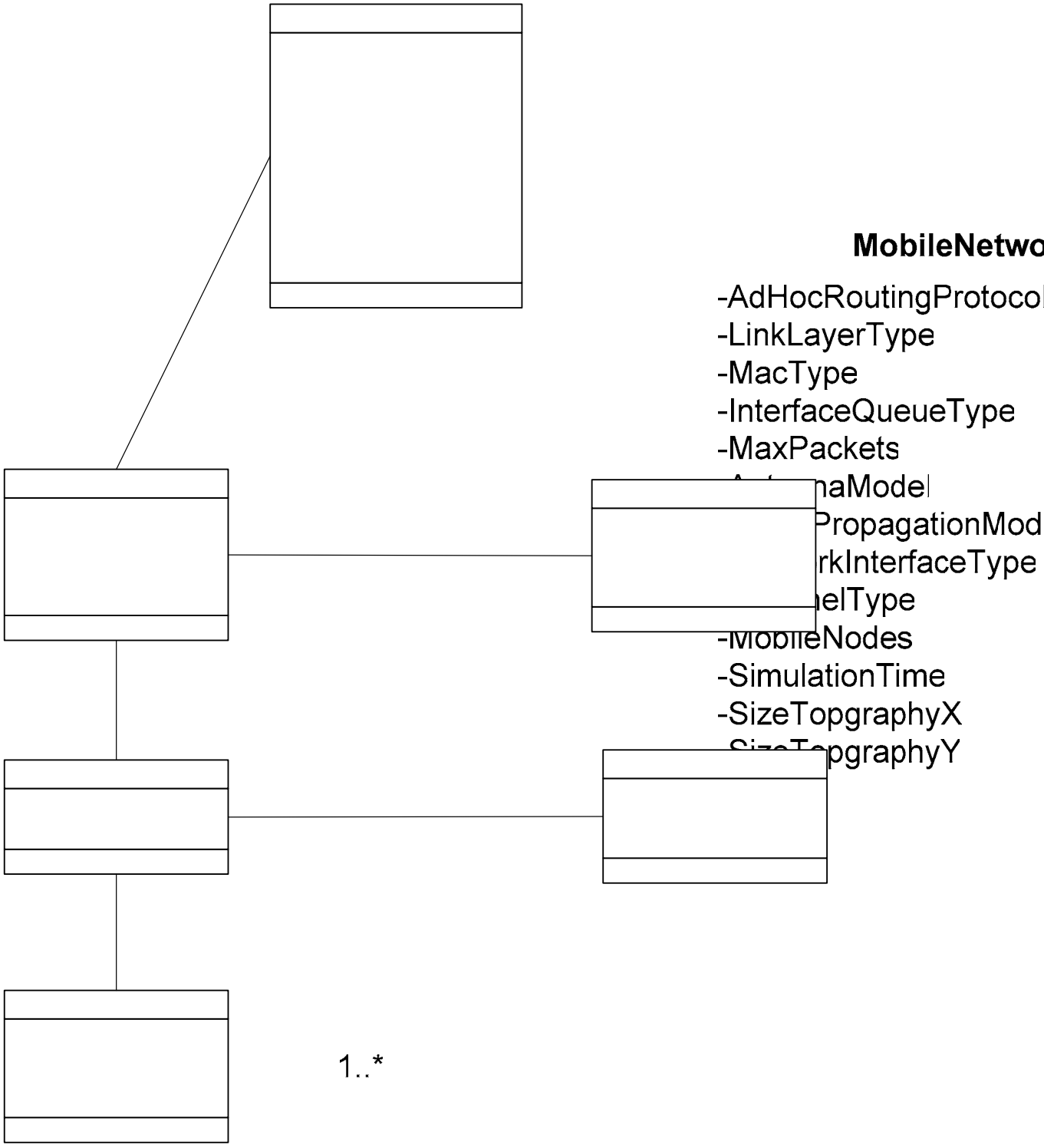

Fig. 3.5. Modelo Conceptual

3.5.- GLOSARIO

Término	Significado
NS	Network Simulator. Simulador de redes del Proyecto VINT. Una investigación de UC Berkeley, LBL, USC/ISI y Xeros Parc.
MNB	Mobile Network Builder. Constructor de Redes Móviles. Proyecto resultado de este proyecto.
CYGWIN	Cygwin es un ambiente semejante a Linux para ambientes Windows. Consiste de un DLL (cygwin1.dll), el cual actúa como una capa de emulación brindando funcionalidad POSIX de llamadas al sistema y una colección de herramientas, las cuales proporcionan una forma real de ver y sentir el ambiente Linux.
POSIX	Portable Operating System Interface. Interfaz de sistema operativo portable.
RUP	Rational Unified Process. Proceso Unificado Rational. Filosofía de desarrollo de software.

Parámetros de la red móvil (MobileNetworkParameters)

Columna	Significado	Ejemplo
AdHocRoutingProtocol	Protocolo de enrutamiento ad-hoc	DSDV, Destination-Sequenced Distance Vector. DSR, Dynamic Source Routing. TORA, Temporally-Ordered Routing Algorithm. AODV, Ad hoc On-Demand Distance Vector.
LinkLayerType	Tipo de capa de enlace	LL, LL/Sat
MacType	Tipo de Mac	Mac/802_11, Mac/Simple, Mac/Csma/Ca, Mac/Sat, Mac/Sat/UnslottedAloha, Mac/Tdma
InterfaceQueueType	Tipo de cola de interfaz	Queue/DropTail/PriQueue, Queue/DropTail,

		CMUPriQueue
MaxPackets	Número máximo de paquetes en la cola de interfaz	50
AntennaModel	Modelo de antena	Antenna/OmniAntenna
RadioPropagationModel	Modelo de radiopropagación	Propagation/TwoRayGround, Propagation/FreeSpace, Propagation/Shadowing, Propagation/ShadowingVis
NetworkInterfaceType	Tipo de interfaz de red	Phy/WirelessPhy, Phy/Sat NetIf/SharedMedia
ChannelType	Tipo de canal	Channel/WirelessChannel, Channel/Sat
MobileNodes	Número de nodos móviles	2
SimulationTime	Tiempo total de la simulación	100
SizeTopgraphyX	Tamaño en X de la topografía	500
SizeTopgraphyY	Tamaño en Y de la topografía	500

Nodos (Nodes)

Columna	**Significado**	**Ejemplo**
CodeNode	Código del nodo	1
NameNode	Nombre del nodo	Node1
InitialX	Posición inicial en X	100
InitialY	Posición inicial en Y	100
InitialZ	Posición inicial en Z	0
Size	Tamaño del nodo para el visualizador	20

Agentes (Agents)

Columna	**Significado**	**Ejemplo**
CodeAgent	Código de agente	1
CodeNode	Código del nodo	1
Agent	Tipo de agente	Agent/TCP Agent/UDP

		Agent/TCPSink Agent/TCP/FullTcp Agent/LossMonitor Agent/Null Agent/DumbAgent Agent/TCP/Reno Agent/TCP/Vegas Agent/TCP/Sack1 Agent/TCP/Fack

Aplicaciones (Applications)

Columna	**Significado**	**Ejemplo**
CodeApplication	Código de aplicación	1
CodeAgent	Código de agente	1
CodeNode	Código del nodo	1
Application	Tipo de aplicación	Application/FTP Application/Telnet Application/Traffic/CBR Application/Traffic/RealAudio Application/Traffic/Exponential Application/Traffic/Pareto
ApplicationStart	Tiempo de inicio de la aplicación	10

Conexiones entre Agentes (AgentConnections)

Columna	**Significado**	**Ejemplo**
CodeAgentSource	Código de agente origen	1
CodeNodeSource	Código de nodo origen	1
CodeAgentTarget	Código de agente destino	2
CodeNodeTarget	Código de nodo destino	2

Movimientos de Nodos (NodeMovements)

Columna	Significado	Ejemplo
CodeNode	Código de nodo	1
Time	Tiempo de inicio del moviiento	10
DestinationX	Destino en X	400
DestinationY	Destino en Y	400
Speed	Velocidad	100

CAPÍTULO 4

DISEÑO

4.1.- DIAGRAMAS DE SECUENCIA

4.1.1.- DIAGRAMA DE SECUENCIA: DEFINIR TOPOLOGÍA DE RED

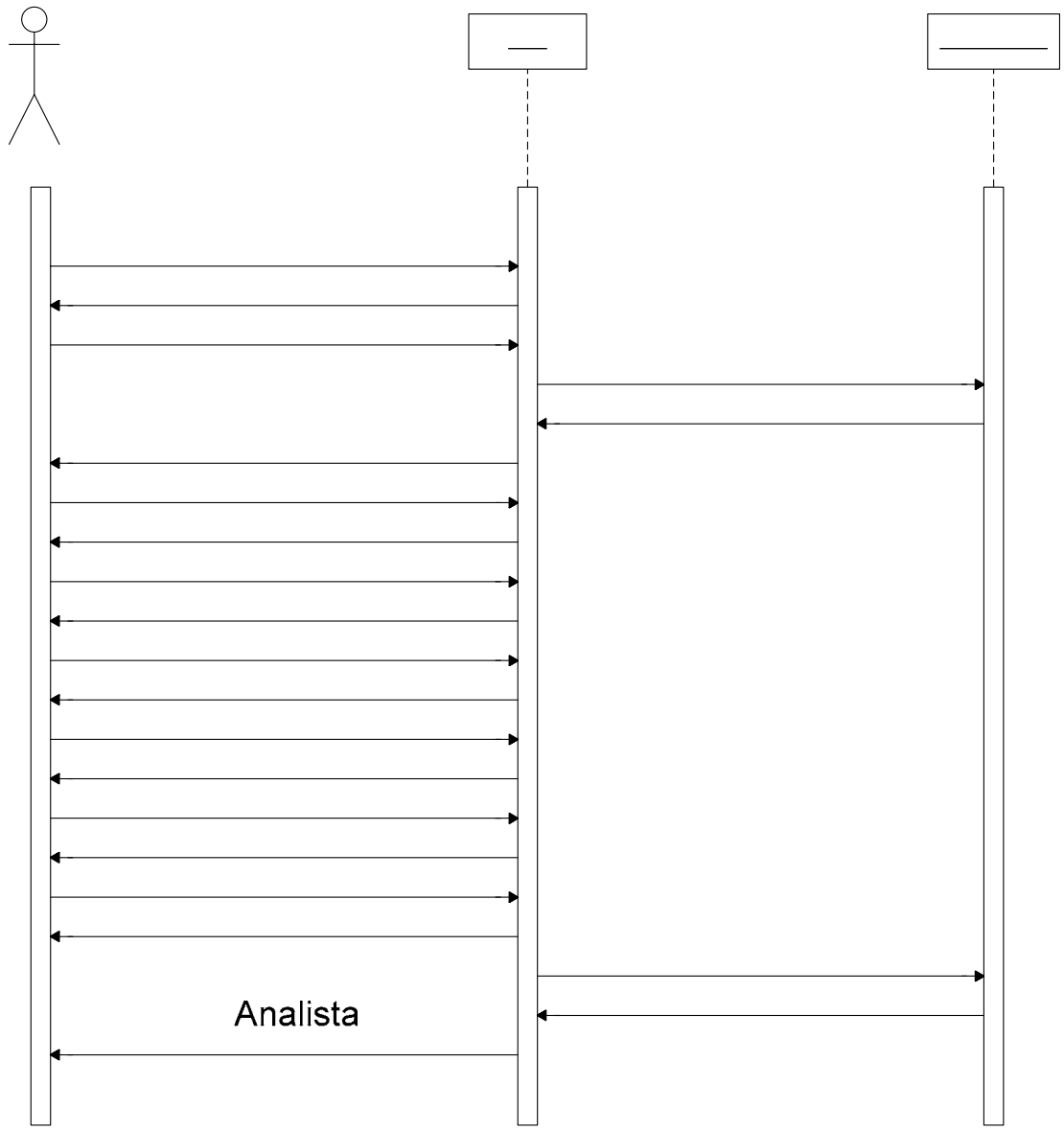

Fig. 4.1. Diagrama de Secuencia: Definir Topología de Red

Ingresar al Sistema

Presentar Pantalla Principal y Menú

Seleccionar Opción "Nueva Red Móvil"

MN

4.1.2.- DIAGRAMA DE SECUENCIA: GENERAR ENTRADA PARA NS

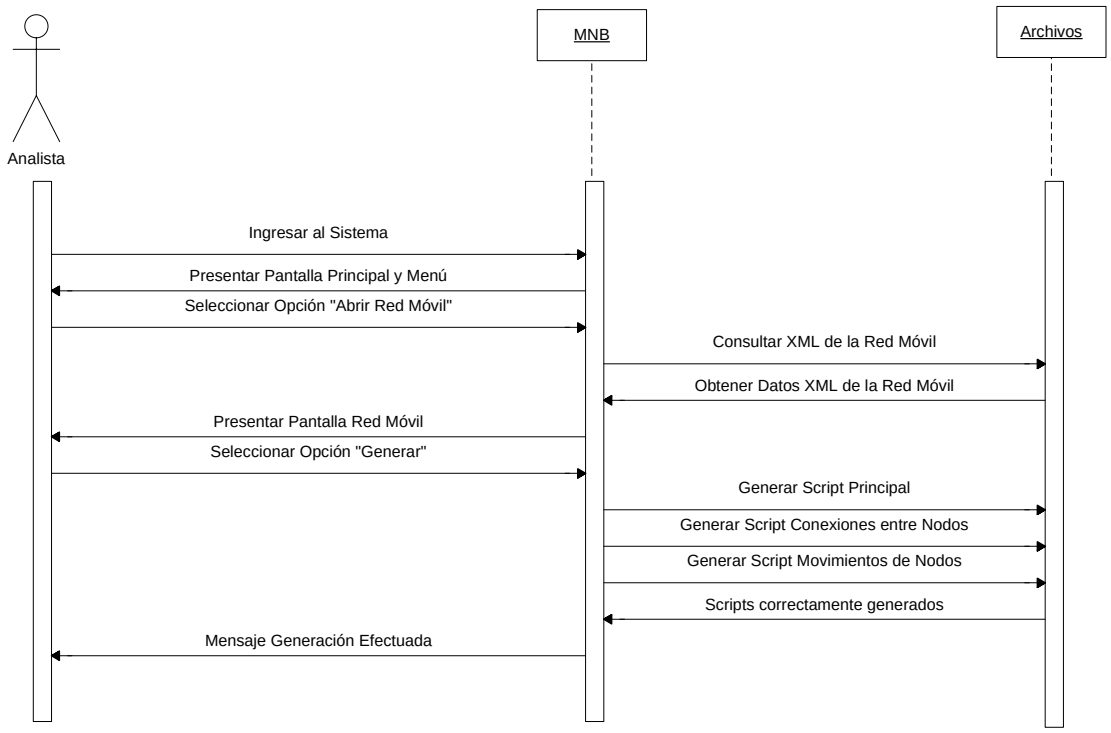

Fig. 4.2. Diagrama de Secuencia: Generar Entrada para NS

4.1.3.- DIAGRAMA DE SECUENCIA: VISUALIZAR SIMULACIÓN DE NS

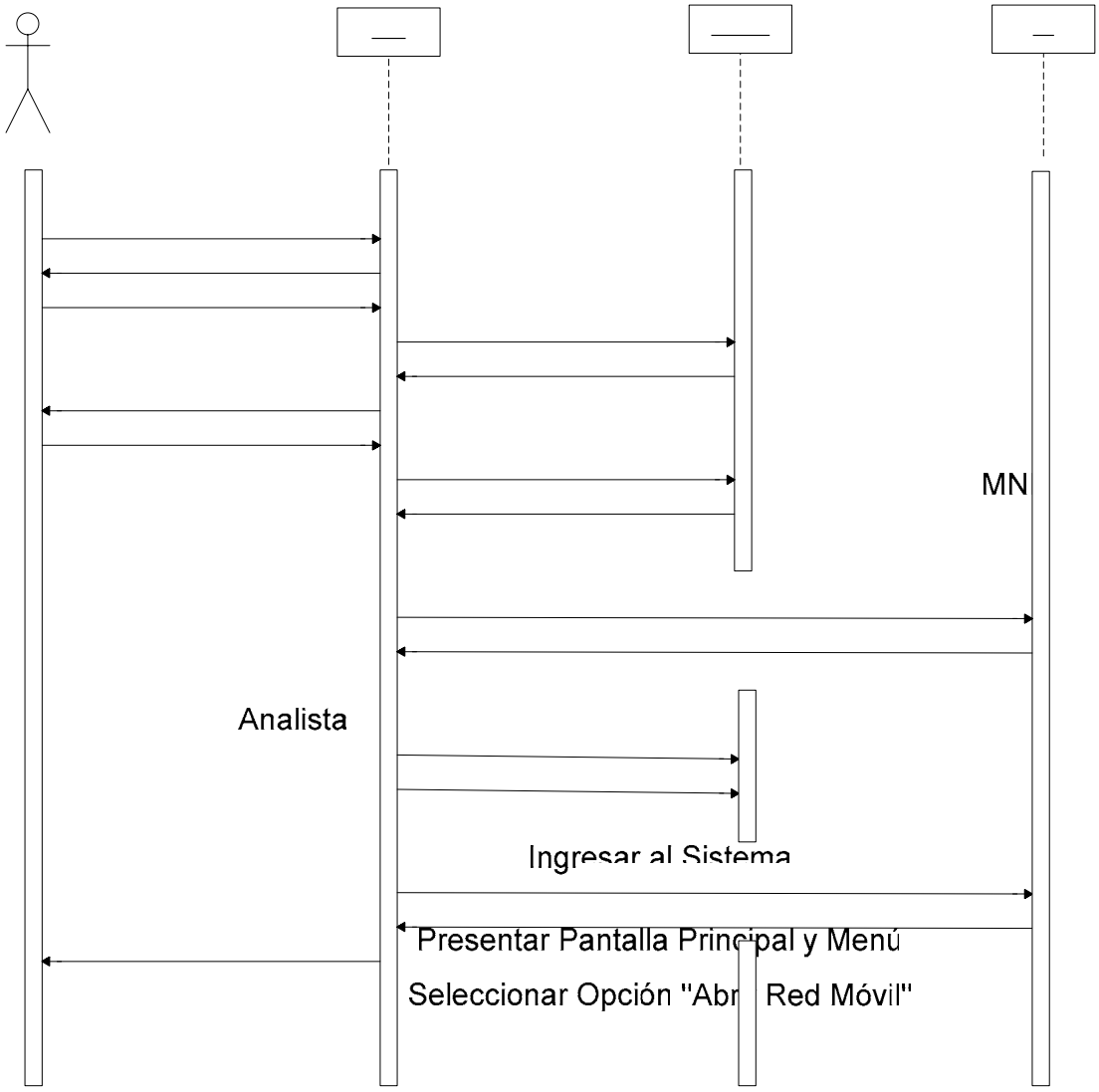

Fig. 4.3. Diagrama de Secuencia: Visualizar Simulación de NS

4.2.- DIAGRAMAS DE COLABORACIÓN

4.2.1.- DIAGRAMA DE COLABORACIÓN: DEFINIR TOPOLOGÍA DE RED

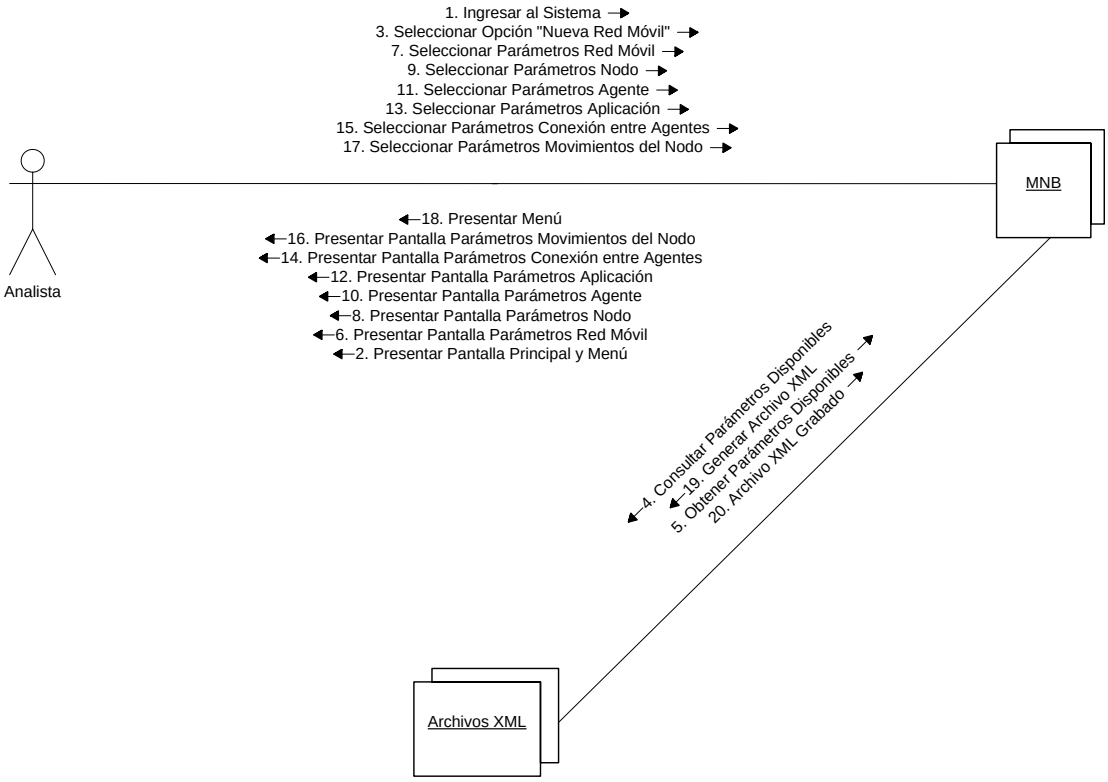

Fig. 4.4. Diagrama de Colaboración: Definir Topología de Red

4.2.2.- DIAGRAMA DE COLABORACIÓN: GENERAR ENTRADA PARA NS

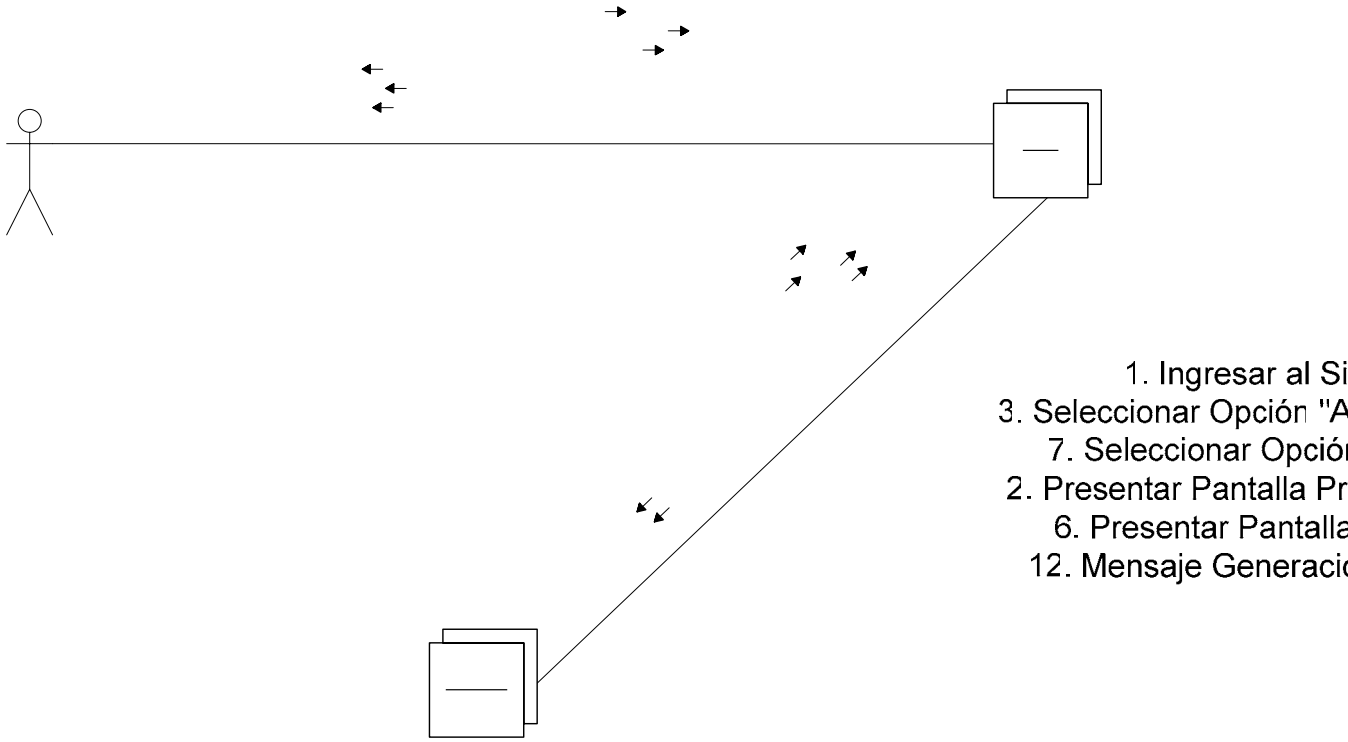

Fig. 4.5. Diagrama de Colaboración: Generar Entrada para NS

Analista

Archivos

4.2.3.- DIAGRAMA DE COLABORACIÓN: VISUALIZAR SIMULACIÓN DE NS

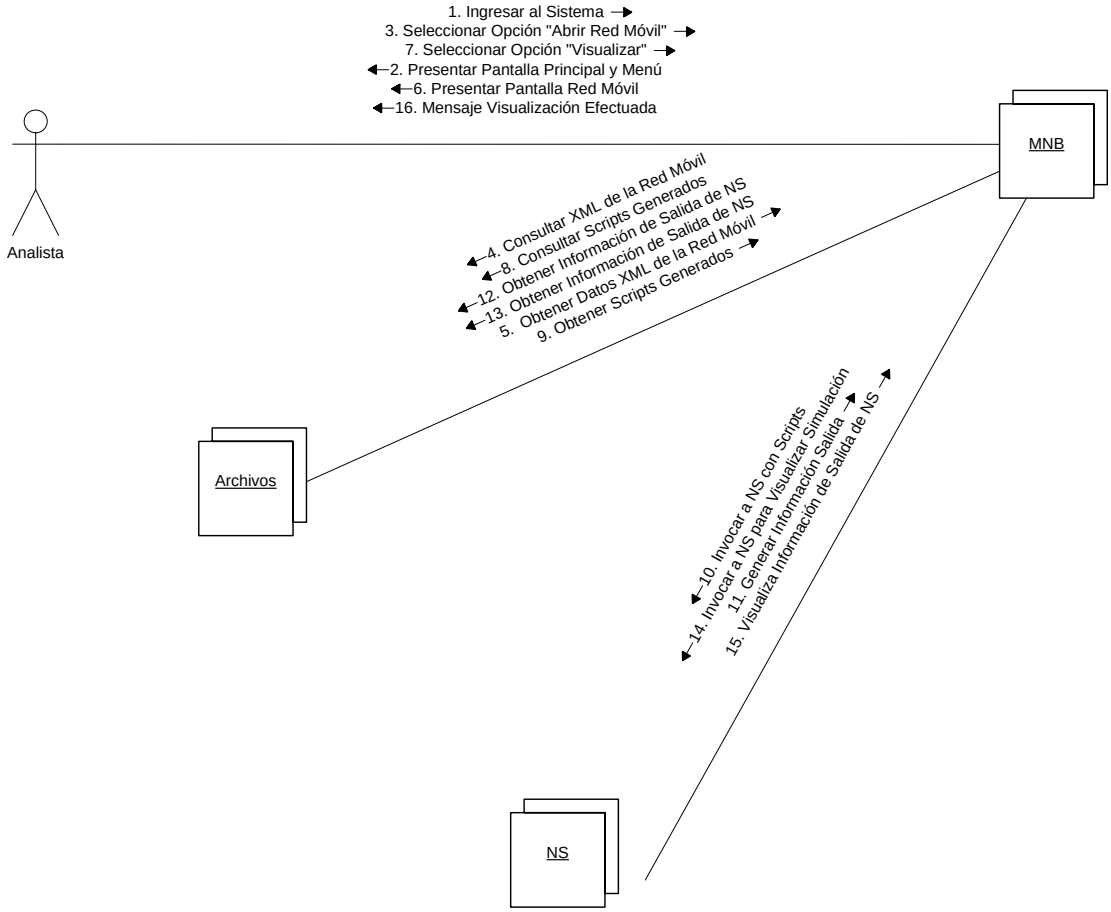

Fig. 4.6. Diagrama de Colaboración: Visualizar Simulación de NS

4.3.- DIAGRAMA DE ESTADOS DE LA RED MÓVIL

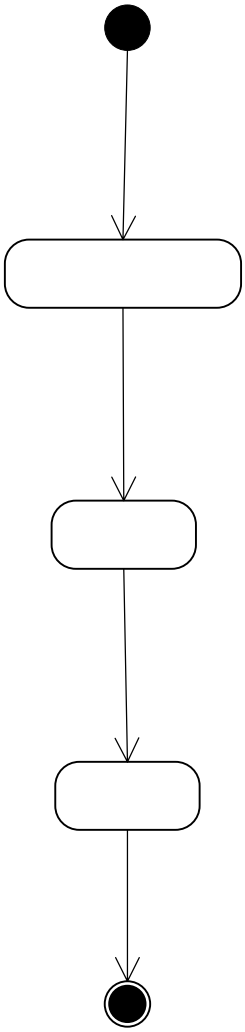

Fig. 4.7. Diagrama de Estado de la Red Móvil

4.4- MODELO DE CLASES DEL SISTEMA

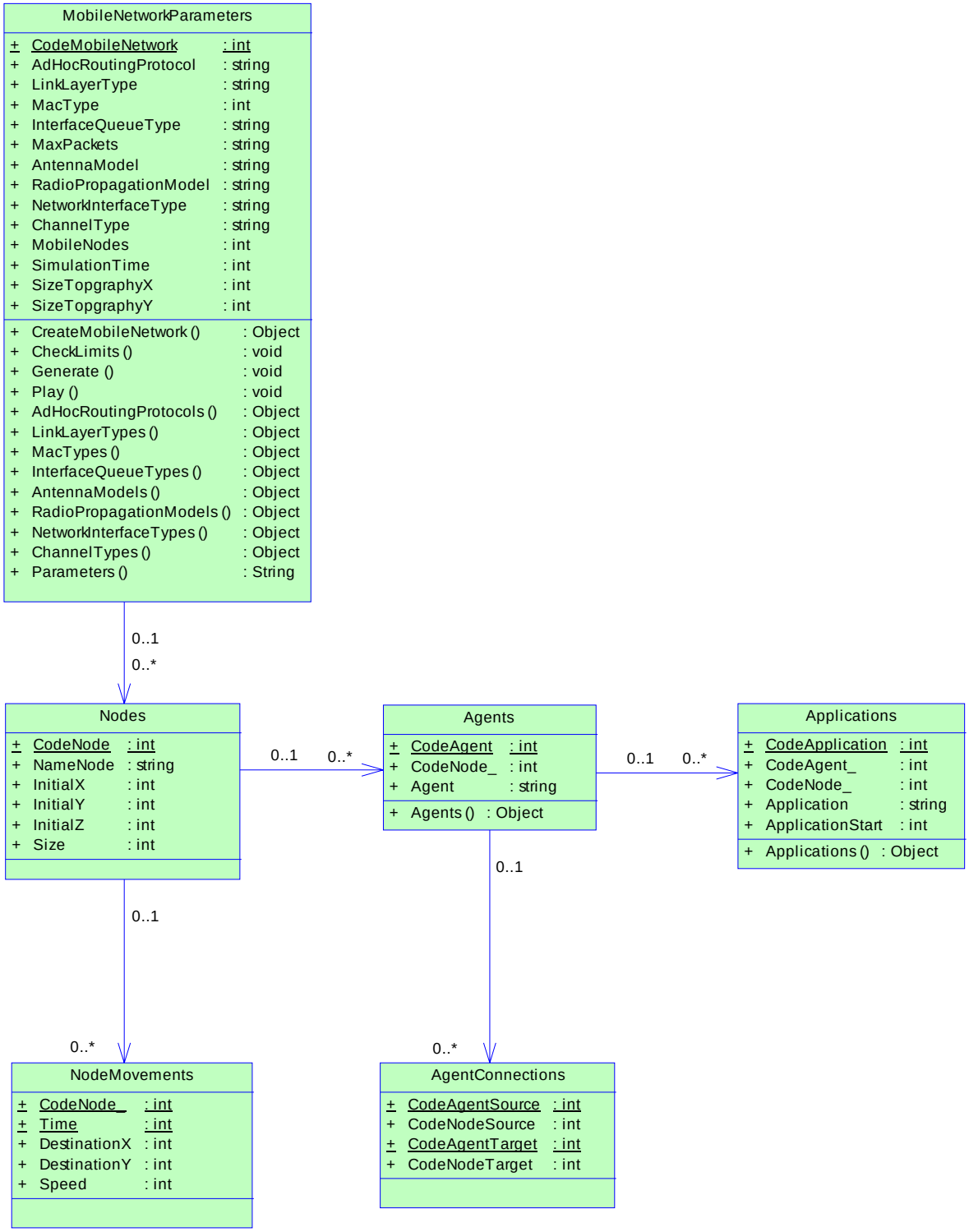

Fig. 4.8. Modelo de Clases del Sistema

4.5- MAPEO A ENTIDAD RELACIÓN (DIAGRAMA ENTIDAD RELACIÓN)

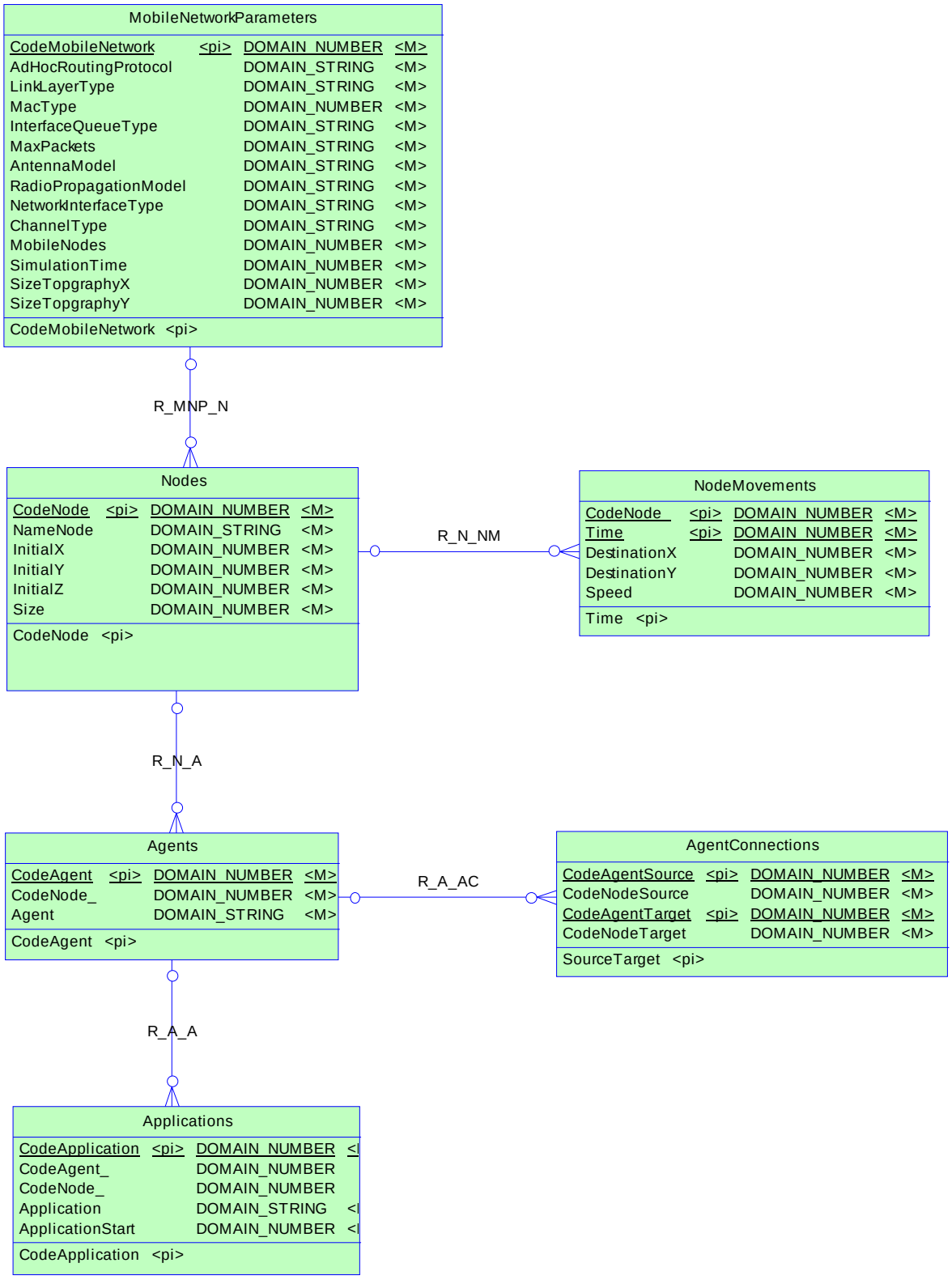

Fig. 4.9. Mapeo a Entidad Relación

4.6- DISEÑO DE INTERFACES

4.6.1.- MODELO DE NAVEGACIÓN DEL SISTEMA

4.6.1.1.- Menú Principal

Archivo	Herramientas	Ventana	Ayuda

Archivo	Opciones de menú para abrir, crear, guardar modelos de simulación de redes móviles en archivos XML.
Herramientas	Herramientas disponibles para graficar un modelo de simulación de redes móviles como: nodos, agentes, aplicaciones, conexiones, etc. También se encuentran herramientas para generar y ejecutar la simulación de la red móvil.
Ventana	Opciones para manipular las ventanas de modelos de simulación.
Ayuda	Opciones para solicitar ayuda sobre un determinado tema.

4.6.1.2.- Menú "Archivo"

Archivo	Herramientas	Ventana	Ayuda
Nuevo			
Abrir			
Guardar			
Guardar como			
Cerrar			
Salir			

Nuevo	Permite crear una nueva ventana para configurar un nuevo modelo de red móvil.
Abrir	Permite leer un archivo XML, previamente grabada por MNB, permitiendo modificar la red móvil ya definida.
Guardar	Permite guardar la configuración de la red móvil en un archivo XML.
Guardar como	Permite guardar la configuración de la red móvil en un archivo XML con un nombre diferente.
Cerrar	Permite cerrar la ventana del modelo de red móvil activo.
Salir	Permite salir de la aplicación informática MNB.

4.6.1.3.- Menú “Herramientas”

Archivo	Herramientas	Ventana	Ayuda
	Puntero		
	Nodo		
	Agente		
	Aplicación		
	Conexión		
	Eliminar		
	Propiedades Red Móvil		
	Generar Red Móvil		
	Ejecutar Red Móvil		

Puntero	Permite quitar la seleccionar de cualquier elemento de red móvil que previamente se haya seleccionado en la ventana de configuración de red móvil.
Nodo	Permite seleccionar el elemento de red “Nodo” para poder añadir un “Nodo” a la configuración de red móvil en la ventana activa. El siguiente paso es dar “clic” en la posición

	que se necesita añadir el nodo en la ventana activa.
Agente	Permite seleccionar el elemento de red “Agente” para poder añadir un “Agente” a la configuración de red móvil en la ventana activa. El siguiente paso es dar “clic” en el nodo donde se necesita añadir el agente. El tipo de agente en forma inicial será el indicado en el combo “Agente”.
Aplicación	Permite seleccionar el elemento de red “Aplicación” para poder añadir una “Aplicación” a la configuración de red móvil en la ventana activa. El siguiente paso es dar “clic” en el agente donde se necesita añadir la aplicación. El tipo de aplicación en forma inicial será el indicado en el combo “Aplicación”.
Conexión	Permite seleccionar el elemento de red “Conexión” para poder añadir una “Conexión” entre agentes a la configuración de red móvil en la ventana activa. El siguiente paso es dar “clic” en el agente que se necesita tener como origen de la conexión. Luego del paso anterior en un menú deberá escoger el agente destino.
Eliminar	Permite eliminar un elemento de red en la configuración de red móvil en la ventana activa. El siguiente paso es dar “clic” en el elemento de red que se necesita eliminar como por ejemplo “Nodo”, “Agente” y “Aplicación”.
Propiedades de red móvil	Permite presentar una pantalla con información de la configuración de la red móvil.
Generar red móvil	Permite generar scripts TCL con referencia al modelo de red móvil de la ventana activa. Estos scripts se generar en el misma carpeta donde esta localizado el archivo .xml.
Ejecutar red móvil	Permite invocar a la interfaz con Network Simulator y enviar como entrada los datos de los scripts TCL generados. Luego de esperar que NS genere la salida del proceso de simulación se procede mostrar los resultados de la simulación y ha visualizar dicho proceso.

4.6.1.4.- Menú “Ventana”

Archivo	Herramientas	Ventana	Ayuda
		Cascada	
		Horizontal	
		Vertical	
		Organizar Íconos	

Cascada	Permite organizar las ventanas de los modelos de simulación de red móvil en forma de cascada.
Horizontal	Permite organizar las ventanas de los modelos de simulación de red móvil en forma horizontal.
Vertical	Permite organizar las ventanas de los modelos de simulación de red móvil en forma de vertical.
Organizar Iconos	Permite organizar los íconos de las ventanas de los modelos de simulación de red móvil que se encuentran minimizadas.

4.6.1.5.- Menú “Ayuda”

Archivo	Herramientas	Ventana	Ayuda
			Manual del Usuario
			Manual del Programador
			Guía de Instalación de Cygwin
			Guía de Instalación de NS
			Guía del Modelo de Movilidad de NS
			Guía de Instalación de MNB
			Acerca de …

Manual del Usuario	Permite presentar la información del Manual del Usuario de la aplicación informática MNB, que se

	encuentra disponible en formato pdf.
Manual del Programador	Permite presentar la información del Manual del Programador de la aplicación informática MNB, que se encuentra disponible en formato pdf.
Guía de instalación de Cygwin	Permite presentar la Guía de Instalación de Cygwin, que se encuentra disponible en formato pdf.
Guía de instalación de NS	Permite presentar la Guía de Instalación de Network Simulator, que se encuentra disponible en formato pdf.
Guía del Modelo de Movilidad de NS	Permite presentar la Guía del Modelo de Movilidad de Network Simulator, que se encuentra disponible en formato pdf.
Guía de Instalación de MNB	Permite presentar la Guía de Instalación de la Aplicación Informática Mobile Network Builder, que se encuentra disponible en formato pdf.

4.6.2- MODELO VISUAL DE INTERFACES

4.6.2.1.- Secuencia Lógica De Interfaces

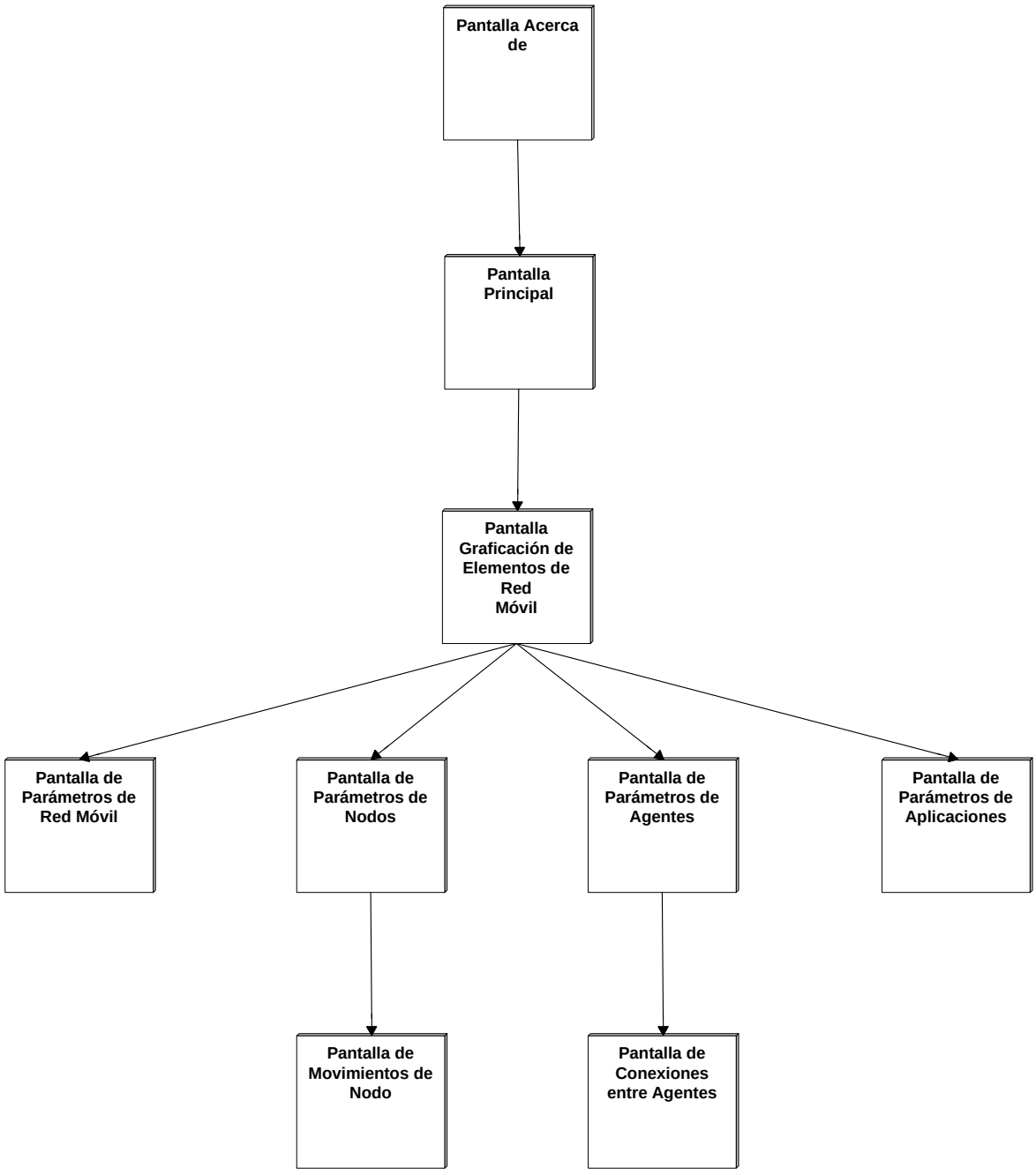

Fig. 4.10. Secuencia Lógica de Pantallas

4.6.2.2.- Pantalla Principal

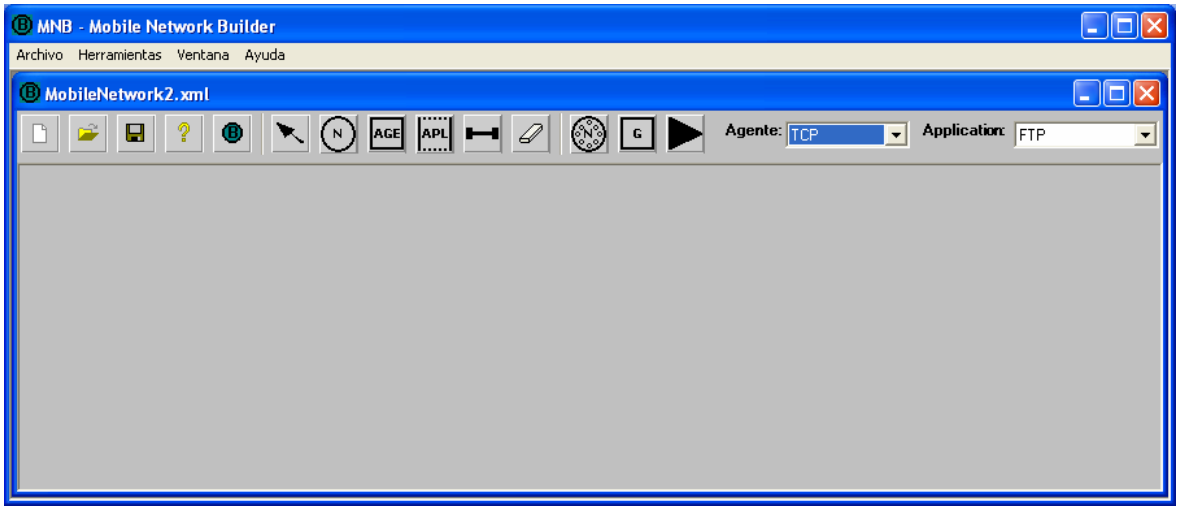

Fig. 4.11. Pantalla Principal

4.6.2.3.- Pantalla De Graficación De Elementos De Red Móvil

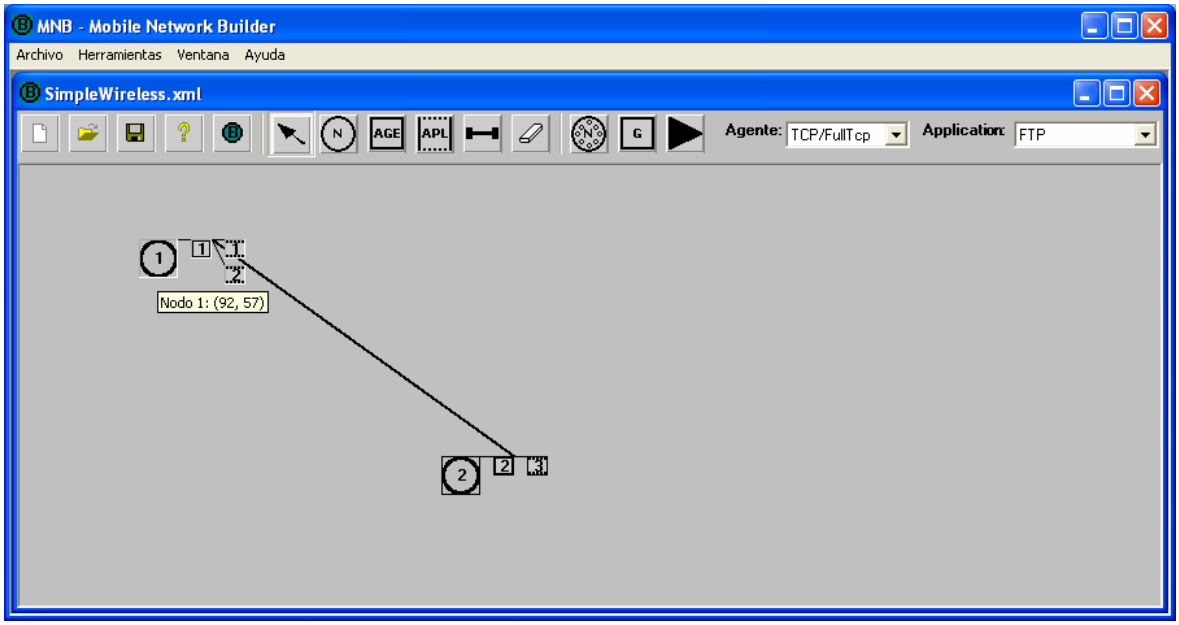

Fig. 4.12. Pantalla de Graficación de Elementos de Red Móvil

4.6.2.4.- Pantalla De Parámetros De Red Móvil

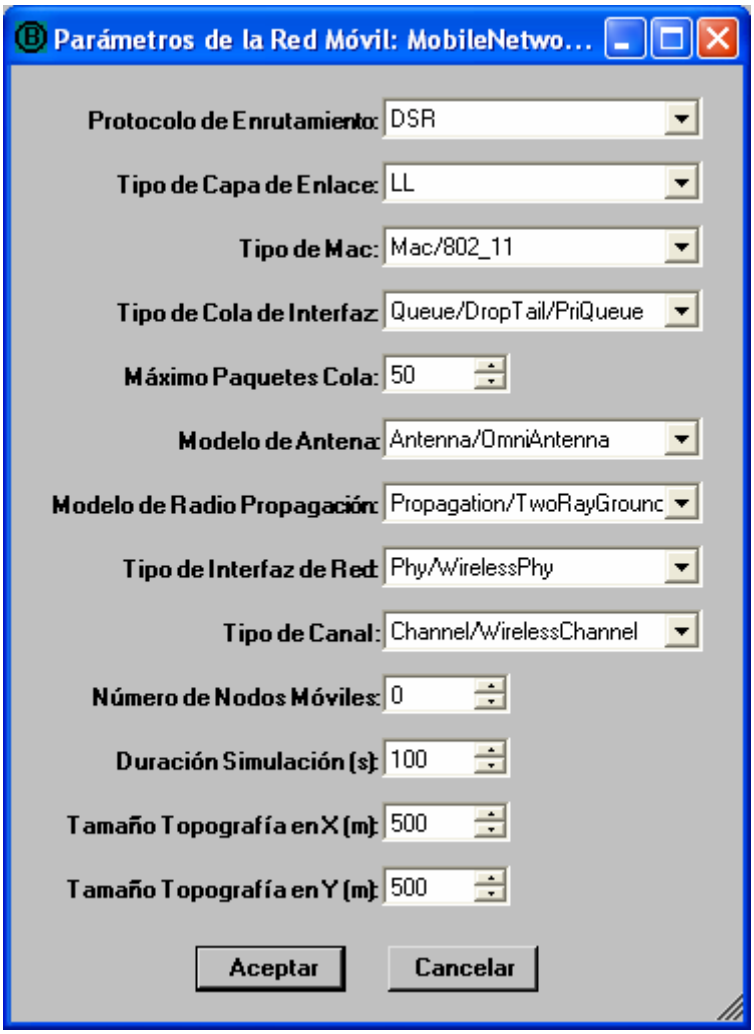

Fig. 4.13. Pantalla de Parámetros de Red Móvil

4.6.2.5.- Pantalla De Parámetros De Nodos

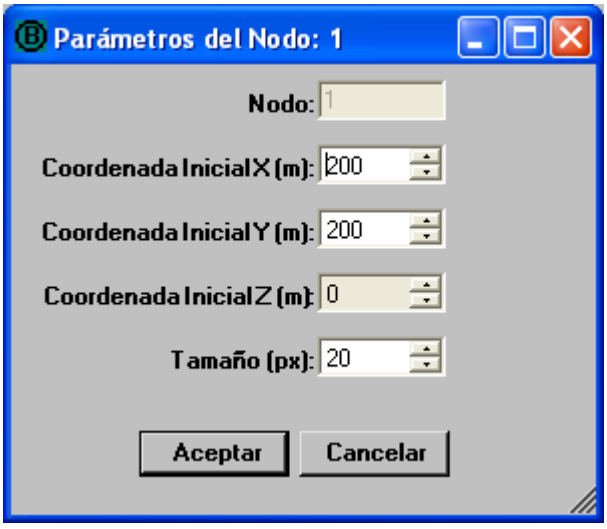

Fig. 4.14. Pantalla de Parámetros de Nodos

4.6.2.6.- Pantalla De Parámetros De Agentes

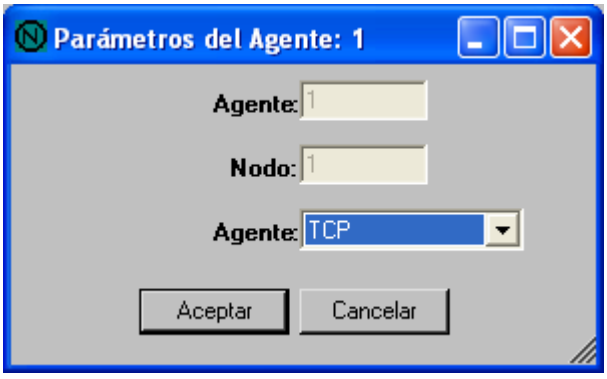

Fig. 4.15. Pantalla de Parámetros de Agentes

4.6.2.7.- Pantalla De Parámetros De Aplicaciones

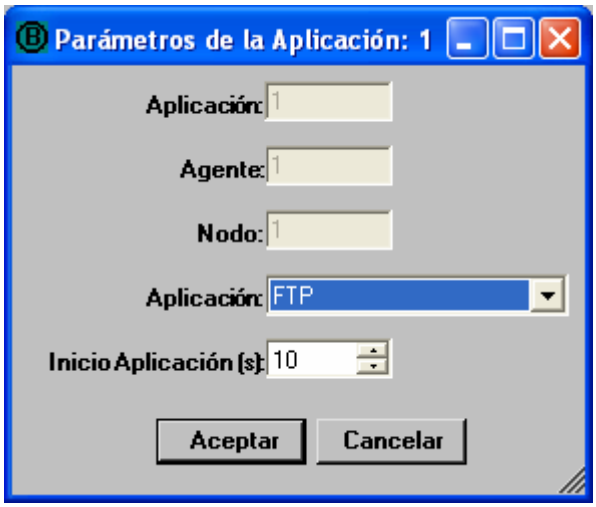

Fig. 4.16. Pantalla de Parámetros de Aplicaciones

4.6.2.8.- Pantalla De Conexiones Entre Agentes

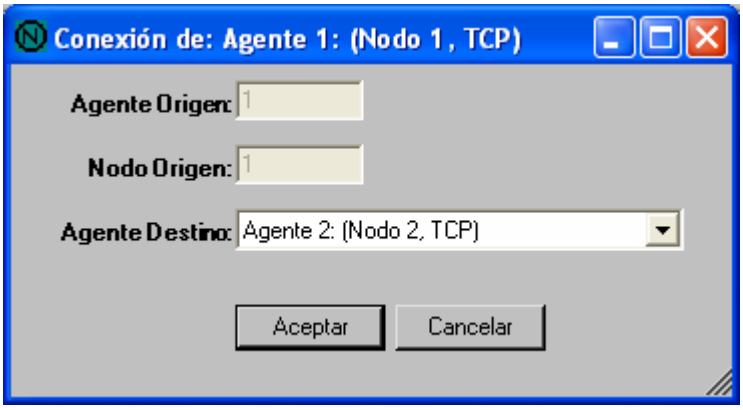

Fig. 4.17. Pantalla de Conexiones entre Agentes

4.6.2.9.- Pantalla De Movimientos De Nodo

Fig. 4.18. Pantalla de Movimientos de Nodo

CAPÍTULO 5
IMPLEMENTACIÓN

5.1.- HERRAMIENTA DE PROGRAMACIÓN

5.1.1.- ELECCIÓN DE LA HERRAMIENTA DE PROGRAMACIÓN

La herramienta de programación elegida es Visual Studio .Net debido a que es la herramienta comercial de programación orientada a objetos más difundida y debido a la amplia gama de controles que pueden usarse para mejorar la imagen de las aplicaciones que se puedan crear.

Visual Studio .Net es un conjunto de lenguajes orientados a objetos, acorde a la tecnología actual que está diseñado para programar aplicaciones en dos modalidades: Windows Forms (Aplicaciones solo para Windows) y Web Forms (Aplicaciones para el Web) y es escalable a posibles nuevas plataformas.

El lenguaje elegido es C# debido a que garantiza un completo control sobre el código y conversión de datos.

5.1.2.- ARQUITECTURA BÁSICA DE LA PLATAFORMA .NET

La nueva tecnología de Microsoft ofrece soluciones a los problemas de programación actuales, como son la administración de código, programación de componentes gráficos o la programación para Internet. Para aprovechar al máximo las características de .Net es necesario entender la arquitectura básica en la que está implementada esta tecnología y así beneficiarse de todas las características que ofrece esta nueva plataforma.

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de

aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- Lenguajes de compilación
- Biblioteca de clases de .Net
- CLR (Common Language Runtime)

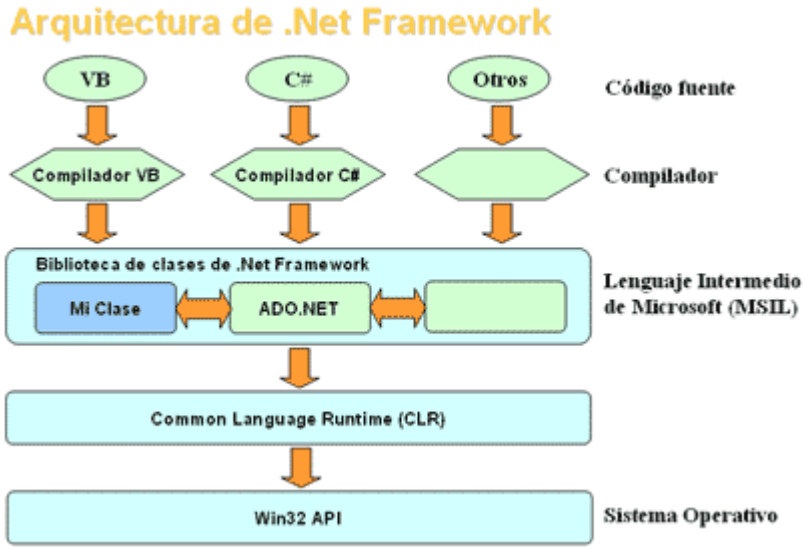

Fig. 5.1. Arquitectura de .Net Framework

Actualmente, el Framework de .Net es una plataforma no incluida en la mayoría de sistemas operativos distribuidos por Microsoft, por lo que es necesaria su instalación previa a la ejecución de programas creados mediante .Net. El Framework se puede descargar gratuitamente desde la web oficial de Microsoft.

.Net Framework soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes. Existen más de 30 lenguajes adaptados a .Net, desde los más conocidos como C# (C Sharp), Visual Basic o C++ hasta otros lenguajes menos conocidos como Perl o Cobol.

5.1.3.- COMMON LANGUAGE RUNTIME (CLR)

El CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de

ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32.

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL, Microsoft Intermediate Lenguaje). Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear código MSIL compatible con el CLR.

De esta forma, indistintamente de la herramienta de desarrollo utilizada y del lenguaje elegido, el código generado es siempre el mismo, ya que el MSIL es el único lenguaje que entiende directamente el CLR. Este código es transparente al desarrollo de la aplicación ya que lo genera automáticamente el compilador.

Sin embargo, el código generado en MSIL no es código máquina y por tanto no puede ejecutarse directamente. Se necesita un segundo paso en el que una herramienta denominada compilador JIT (Just-In-Time) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora.

De esta forma se consigue con .Net cierta independencia de la plataforma, ya que cada plataforma puede tener su compilador JIT y crear su propio código máquina a partir del código MSIL.

La compilación JIT la realiza el CLR a medida que se invocan los métodos en el programa y, el código ejecutable obtenido, se almacena en la memoria caché de la computadora, siendo recompilado sólo cuando se produce algún cambio en el código fuente.

5.1.4.- BIBLIOTECA DE CLASES DE .NET

Cuando se está programando una aplicación, muchas veces se necesitan realizar acciones como manipulación de archivos, acceso a datos, conocer el estado del sistema, implementar seguridad, etc. El Framework organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

Para ello, el Framework posee un sistema de tipos universal, denominado Common Type System (CTS). Este sistema permite que el programador pueda interactuar los tipos que se incluyen en el propio Framework (biblioteca de clases de .Net) con los creados por él mismo (clases). De esta forma se aprovechan las ventajas propias de la programación orientada a objetos, como la herencia de clases predefinidas para crear nuevas clases, o el polimorfismo de clases para modificar o ampliar funcionalidades de clases ya existentes.

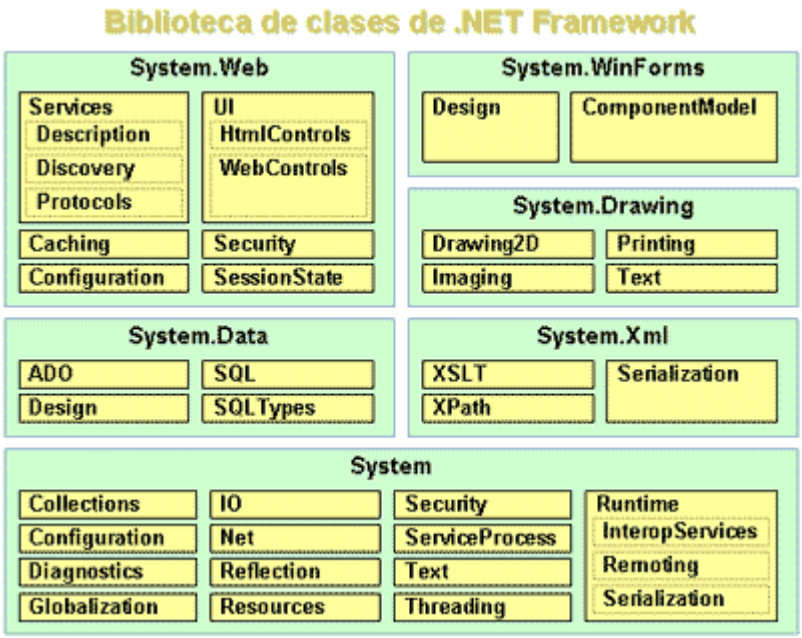

Fig. 5.2. Biblioteca de clases de .Net Framework

La biblioteca de clases de .Net Framework incluye, entre otros, tres componentes clave:

- ASP.NET para construir aplicaciones y servicios Web.
- Windows Forms para desarrollar interfaces de usuario.
- ADO.NET para conectar las aplicaciones a bases de datos.

La forma de organizar la biblioteca de clases de .Net dentro del código es a través de los espacios de nombres (namespaces), donde cada clase está organizada en espacios de nombres según su funcionalidad. Por ejemplo, para manejar ficheros se utiliza el espacio de nombres System.IO y si lo que se quiere es obtener información de una fuente de datos se utilizará el espacio de nombres System.Data.

La principal ventaja de los espacios de nombres de .Net es que de esta forma se tiene toda la bliblioteca de clases de .Net centralizada bajo el mismo espacio de nombres (System). Además, desde cualquier lenguaje se usa la misma sintaxis de invocación, ya que a todos los lenguajes se aplica la misma biblioteca de clases.

5.1.5.- ENSAMBLADOS

Uno de los mayores problemas de las aplicaciones actuales es que en muchos casos tienen que tratar con diferentes archivos binarios (DLL´s), elementos de registro, conectividad abierta a bases de datos (ODBC), etc.

Para solucionarlo el Framework de .Net maneja un nuevo concepto denominado ensamblado. Los ensamblados son ficheros con forma de EXE o DLL que contienen toda la funcionalidad de la aplicación de forma encapsulada. Por tanto la solución al problema puede ser tan fácil como copiar todos los ensamblados en el directorio de la aplicación.

Con los ensamblados ya no es necesario registrar los componentes de la aplicación. Esto se debe a que los ensamblados almacenan dentro de si mismos

toda la información necesaria en lo que se denomina el manifiesto del ensamblado. El manifiesto recoge todos los métodos y propiedades en forma de meta-datos junto con otra información descriptiva, como permisos, dependencias, etc.

Para gestionar el uso que hacen la aplicaciones de los ensamblados .Net utiliza la llamada caché global de ensamblados (GAC, Global Assembly Cache). Así, .Net Framework puede albergar en el GAC los ensamblados que puedan ser usados por varias aplicaciones e incluso distintas versiones de un mismo ensamblado, algo que no era posible con el anterior modelo COM.

5.1.6.- ALGUNAS DE LAS VENTAJAS DE LA PLATAFORMA .NET.

A continuación se resumen las ventajas más importantes que proporciona .Net Framework:

- **Código administrado**: El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.

- **Interoperabilidad multilenguaje**: El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (MSIL).

- **Compilación just-in-time**: El compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.

- **Garbage collector**: El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura (garbage collector). El CLR detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma el programador no tiene por que liberar la memoria de forma explícita aunque también sea posible hacerlo manualmente (mediante el método dispose) liberamos el objeto para que el recolector de basura lo elimine de memoria.

- **Seguridad de acceso al código**: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.

- **Despliegue**: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones.

5.1.7.- INCONVENIENTES DE LA PLATAFORMA .NET

Procesos como la recolección de basura de .Net o la administración de código introducen factores de sobrecarga que repercuten en la demanda de más requisitos del sistema.

El código administrado proporciona una mayor velocidad de desarrollo y mayor seguridad de que el código sea bueno. En contrapartida el consumo de recursos

durante la ejecución es mucho mayor, aunque con los procesadores actuales esto cada vez es menos inconveniente.

El nivel de administración del código dependerá en gran medida del lenguaje que utilicemos para programar. Por ejemplo, mientras que Visual Basic .Net es un lenguaje totalmente administrado, C Sharp permite la administración de código de forma manual, siendo por defecto también un lenguaje administrado. Mientras que C++ es un lenguaje no administrado en el que se tiene un control mucho mayor del uso de la memoria que hace la aplicación.

5.1.8.- INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS, UTILIZADA EN LA TECNOLOGÍA .NET.

La programación orientada a objetos es una evolución de la programación procedimental basada en funciones. La POO nos permite agrupar secciones de código con funcionalidades comunes.

Una de las principales desventajas de la programación procedimental basada en funciones es su construcción, cuando una aplicación bajo este tipo de programación crece, la modificación del código se hace muy trabajosa y difícil debido a que el cambio de una sola línea en una función, puede acarrear la modificación de muchas otras líneas de código pertenecientes a otras funciones que estén relacionadas.

Con la programación orientada a objetos se pretende agrupar el código encapsulándolo y haciéndolo independiente, de manera que una modificación debida al crecimiento de la aplicación solo afecte a unas pocas líneas.

La organización de una aplicación en POO se realiza mediante estructuras de código, también llamados objetos. Estos objetos contienen una serie de procedimientos e información destinados a resolver un grupo de tareas con un denominador común. Un procedimiento que esté situado en un objeto no podrá ser usado por otro procedimiento perteneciente a otro objeto, si no es bajo una serie de reglas. Los datos que mantenga el objeto, permanecerán aislados del exterior y sólo se podrá acceder a ellos siguiendo ciertas normas.

El objetivo de POO es catalogar y diferenciar el código, en base a estructuras jerárquicas dependientes, al estilo de un árbol genealógico.

Los objetos se crean a partir de una serie de especificaciones o normas que definen como va a ser el objeto, esto es lo que en POO se conoce como una clase.

Las clases definen la estructura que van a tener los objetos que se creen a partir de ella, indicando que propiedades y métodos tendrán los objetos.

Las propiedades definen los datos o información del objeto, permitiendo modificar o consultar su estado, mientras que los métodos son las rutinas que definen el comportamiento del objeto. Es necesario tener muy clara cual es la diferencia entre un objeto y una clase, a este respecto podemos decir que una clase constituye la representación abstracta de algo mientras que un objeto constituye la representación concreta de lo que la clase define. Imaginemos los planos de una casa diseñados por un arquitecto, en ellos encontramos el esquema de la casa, las medidas, los materiales etc. Una vez construida la casa podremos comprobar que cumple todo lo que los planos determinaban, de esta manera podemos comparar los planos de la casa con las clases en POO, y la casa en si con un objeto creado a partir de una clase. Se debe destacar también que con los mismos planos se pueden crear muchas casas iguales, lo mismo ocurre en POO, a partir de una clase se pueden crear muchos objetos iguales.

La creación de un objeto a partir de una clase se conoce como instanciación de un objeto.

5.2.- ESTÁNDARES DE PROGRAMACIÓN

Nomenclatura

Con la utilización de esta nomenclatura se intenta crear un código fuente preciso, legible y conciso. Se presenta a continuación un ejemplo de nomenclatura que puede servir para los requerimientos de desarrollo previstos.

Métodos

Los nombres de los métodos (Métodos, Funciones, Clases) deben ser mnemotécnicos (fáciles de memorizar) y seguir las mismas pautas de escritura que las de los identificadores de los nombres de las variables. Es conveniente que el primer nombre de un método sea un verbo en infinitivo.

La primera letra del identificador se escribe en mayúscula y las siguientes en minúsculas. En el caso de nombres compuestos se deben utilizar mayúsculas al comienzo de cada palabra para facilitar su lectura. No se deben usar tildes ni guiones. Ejemplos: estudiante(); estudianteNota(); getBackground(); initialize(); getTelescopicOrientation();void materiaEstudiante( ).

Variables

Las variables se deben declarar únicamente al principio de los bloques de código. Un bloque de código es cualquier código encerrado entre llaves “{..}”.

El nombre de la variable, debe dar una idea de lo que se almacena en la misma. La primera letra del identificador se escribe en minúscula. En el caso de nombres compuestos se deben utilizar mayúsculas al comienzo de cada uno para

facilitar su lectura. Ejemplos: fechaActual (variable local de tipo fecha), mensaje (variable pública de tipo cadena)

Si desean utilizar nombres cortos para variables temporales puede utilizar: “i, j, k, m, n” para enteros y “c, d, e” para caracteres

Constantes

Sus nombres deben ser todo con mayúsculas y los subguiones se deben utilizar para separar palabras, Ejemplos: MAXIMO ‘Constante de tipo entero’, NOMBRE_MATERIA ‘Constante de tipo string’.

Clases

Los nombres de las clases tienen la siguiente estructura:

```
class PrefijoIdentificador
{
        <Sentencias>;
}
```

- Prefijo: Identifica el tipo de clase. La primera letra del prefijo se escribe en mayúscula y las siguientes en minúsculas.

- Identificador: Identifica la clase. La primera letra del identificador se escribe en mayúscula y las siguientes en minúsculas. En el caso de nombres compuestos se deben utilizar mayúsculas al comienzo de cada palabra para facilitar su lectura, no se deben usar tildes ni guiones.

Código Fuente

Las normas de diseño de código fuente pretenden estandarizar la estructura lógica y el estilo de programación de las aplicaciones.

Escritura de los programas fuentes

Se debe velar por la claridad de los fuentes, tanto al visualizarlos como al imprimirlos, lo cual nos conduce al uso de comentarios, tabuladores y a no sobrepasar los 80 caracteres por línea.

Tabuladores

Los tabuladores deben tener un tamaño de cuatro caracteres; deben utilizarse para resaltar las estructuras lógicas y sentencias complejas (que ocupen más de una línea). También se debe utilizar un tabulador utilizando sangrías en cada sentencia de una rutina con respecto a su cabeza. Ejemplo:

```
private static synchronized horkingLongMethodName(int anArg,
 Object anotherArg, String yetAnotherArg,
 Object andStillAnother) { ... }
```

Uso de comentarios

El compilador de C# reconoce y elimina los espacios en blanco, tabuladores, retornos de carro y comentarios durante el análisis del código fuente. Los comentarios se pueden presentar en dos formatos distintos:

- // comentario de una línea
- /* comentario
  De múltiples líneas
  */

5.3.- ESTABLECIMIENTO DE LIBRERIAS

Dentro de la solución de Visual Studio .Net para el desarrollo de este proyecto denominado MNB (Mobile Network Builder), se han identificado los siguientes proyectos:

PROYECTO	TIPO	DESCRIPCIÓN
Com.Comun	Library C#	Librería de Visual Studio .Net escrita en lenguaje C#. Librería que dispone de objetos padre

		para el manejo de ventanas en una aplicación windows. Contiene clases heredadas de System.Windows.Forms.
Com.Simulator.Comun	Library C#	Librería de Visual Studio .Net escrita en lenguaje C#. Librería que dispone de objetos para manejar los datos de la aplicación del simulador. Hacen referencia a las clases System.Data para manejar los arhivos XML en este caso. Además tiene objetos para manejar el comportamiento de la red móvil.
Com.Simulator	WinExe C#	Aplicación ejecutable de Visual Studio .Net escrita en lenguaje C#. Esta aplicación hace referencia a la librería Com.Comun para manejo de ventanas y a la librería Com.Simulator.Comun para manejar los datos del simulador. Además define los objetos principales para el manejo del simulador.

5.4.- DESCRIPCION DE CLASES.

DESCRIPCION DE CLASES DEL PROYECTO “Com.Comun”.

PROYECTO	CLASE	DESCRIPCIÓN
Com.Comun	clsComunMain	Contiene la clase padre principal de inicio del Proyecto.
	frmComun	Define la clase común para todas las formas de la aplicación. Se hereda de: System.Windows.Forms.Form.
	frmComunMDI	Define la clase común para todas las formas windows tipo MDI.
	frmComunModal	Define la clase común para todas las formas windows tipo modal.
	frmComunSheet	Define la clase común para todas las formas windows tipo hijo que van a funcionar dentro de una forma MDI.

DESCRIPCION DE CLASES DEL PROYECTO “Com.Simulator.Comun”.

PROYECTO	CLASE	DESCRIPCIÓN
Com.Simulator.Comun	clsAgentParameters	Contiene los métodos para obtener datos con respecto a los parámetros de los Agentes.
	clsApplicationParameters	Contiene los métodos para obtener datos con respecto a los parámetros de las Aplicaciones.
	clsMobileNetwork	Contiene los métodos para procesar lo que tiene que ver con la red móvil, pero sin tener que ver con controles gráficos. Tiene métodos para crear la red móvil para generar los scripts TCL y ejecutar la simulación.
	clsMobileNetworkParameters	Contiene los métodos para obtener datos con respecto a los parámetros de la red móvil.
	clsNodeParameters	Contiene los métodos para obtener datos con respecto a los parámetros de los nodos.
	clsNS	Contiene los métodos para obtener datos con respecto a las variables de NS que se generarán en los scripts a través de un archivo XML.

DESCRIPCION DE CLASES DEL PROYECTO “Com.Simulator\Class”.

PROYECTO	CLASE	DESCRIPCIÓN
Com.Simulator\Class	AgentMobileNetwork	Abstrae el comportamiento de un agente en una red móvil. Tiene propiedades y métodos para ser usados en la interfaz gráfica.
	ApplicationMobileNetwork	Abstrae el comportamiento de una aplicación en una red móvil. Tiene propiedades y métodos para ser usados en la interfaz

		gráfica.
	NodeMobileNetwork	Abstrae el comportamiento de un nodo en una red móvil. Tiene propiedades y métodos para ser usados en la interfaz gráfica.

DESCRIPCION DE CLASES DEL PROYECTO “Com.Simulator”.

PROYECTO	CLASE	DESCRIPCIÓN
Com.Simulator	clsSimulatorMain	Contiene la clase principal de inicio del Proyecto. Hace referencia a Com.Comun.
	frmAbout	Contiene la forma que muestra al usuario, información acerca de quienes colaboraron con el presente trabajo.
	frmParamAgent	Contiene la forma que muestra al usuario, los parámetros de los Agentes de la red móvil para que el usuario los pueda modificar.
	frmParamAgentConnection	Contiene la forma que muestra al usuario, los parámetros de la conexión entre Agentes de la red móvil para que el usuario los pueda modificar.
	frmParamApplication	Contiene la forma que muestra al usuario, los parámetros de la aplicación de la red móvil para que el usuario los pueda modificar.
	frmParamMobileNetWork	Contiene la forma que muestra al usuario, los parámetros de la red móvil para que el usuario los pueda modificar.
	frmParamNode	Contiene la forma que muestra al usuario, los parámetros de los nodos de la red móvil para que el usuario los pueda modificar.
	frmParamNodeMovements	Contiene la forma que muestra al usuario, los

		parámetros de los movimientos de un nodo de la red móvil para que el usuario los pueda modificar.
	frmSimulatorMDI	Contiene la forma principal tipo MDI que muestra al usuario las formas hijo que corresponden a la simulación de una red móvil individual.
	frmSimulatorMobileNetwork	Contiene la forma que se muestra en la ventana MDI principal de la aplicación MNB. Contiene toda la funcionalidad para que se defina una red móvil individual. Esta forma invoca a todas las formas de parámetros que se describieron anteriormente.

CAPÍTULO 6

PRUEBAS

6.1.- CASOS DE PRUEBA DE FUNCIONALIDAD

6.1.1.- CASO DE USO 1. DEFINIR TOPOLOGÍA DE RED

6.1.1.1.- Caso de Prueba “Ingreso de Parámetros de la Red Móvil”.

Descripción de Caso de Prueba	Ingrese a la aplicación MNB. Seleccione en el menú principal la opción Archivo/Nuevo. Seleccione en el menú principal la opción Herramientas/Propiedades Red Móvil. En la ventana de parámetros de la red móvil ingrese toda la información requerida: Protocolo de Enrutamiento Ad-Hoc. Tipo de Capa de Enlace. Tipo de Mac. Tipo de Cola de Interfaz. Máximo Paquetes Cola. Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal. Número de Nodos Móviles. Duración Simulación. Tamaño de la Topografía en X. Tamaño de la Topografía en Y.

	Presione “Aceptar”. Seleccione en el menú principal la opción Archivo/Guardar. Verifique con algún editor XML que la información ha sido guardada correctamente.

Prueba #1.

Descripción General	Ingresar Parámetros Red Móvil
Datos de Entrada	Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola:50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 0. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500.
Resultados Esperados	Los parámetros de la red móvil se registran adecuadamente
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil.
Fecha de Ejecución	4 de Diciembre, 2006
Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	No se registra información en los parámetros: Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal.

Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	Iguales a los resultados esperados
Conclusión 2:	La prueba fue realizada con éxito
Total de Repeticiones:	2

Prueba #2

Descripción General	Ingresar Parámetros Red Móvil
Datos de Entrada	Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 0. Duración Simulación: 50. Tamaño de la Topografía en X: 100. Tamaño de la Topografía en Y: 100.
Resultados Esperados	Los parámetros de la red móvil se registran adecuadamente
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil.
Fecha de Ejecución	4 de Diciembre, 2006
Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	Se repite la información en los parámetros: Tipo de Interfaz de Red. Tipo de Canal.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	Iguales a los resultados esperados
Conclusión 2:	La prueba fue realizada con éxito
Total de Repeticiones:	2

Prueba #3

Descripción General	Ingresar Parámetros Red Móvil
Datos de Entrada	Protocolo de Enrutamiento Ad-Hoc: TORA. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/Simple. Tipo de Cola de Interfaz: CMUPriQueue. Máximo Paquetes Cola:50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/ShadowingVis. Tipo de Interfaz de Red: NetIf/SharedMedia. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 0. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500.
Resultados Esperados	Los parámetros de la red móvil se registran adecuadamente
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil.
Fecha de Ejecución	4 de Diciembre, 2006
Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	Se repite la información en los parámetros: Tamaño de la Topografía en X. Tamaño de la Topografía en Y.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	Iguales a los resultados esperados
Conclusión 2:	La prueba fue realizada con éxito
Total de Repeticiones:	2

6.1.1.2.- Caso de Prueba “Ingresar Nodos en la Red Móvil”.

Descripción de Caso de Prueba	Ingrese a la aplicación MNB. Seleccione en el menú principal la opción Archivo/Nuevo.

	Seleccione en el menú principal la opción Herramientas/Propiedades Red Móvil. En la ventana de parámetros de la red móvil ingrese toda la información requerida: Protocolo de Enrutamiento Ad-Hoc. Tipo de Capa de Enlace. Tipo de Mac. Tipo de Cola de Interfaz. Máximo Paquetes Cola. Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal. Número de Nodos Móviles. Duración Simulación. Tamaño de la Topografía en X. Tamaño de la Topografía en Y. Presione “Aceptar”. Definimos los nodos requeridos y sus posiciones iniciales. Seleccione en el menú principal la opción Archivo/Guardar. Verifique con algún editor XML que la información ha sido guardada correctamente.

Prueba #4.

Descripción General	Ingresar Nodos Red Móvil
Datos de Entrada	Propiedades de la red móvil: Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz:

	Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 3. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500. Se definen 3 nodos de la siguiente manera: Nodo1: Posición X: 200. Posición Y: 100. Tamaño: 20. Nodo2: Posición X: 300. Posición Y: 300. Tamaño: 20. Nodo3: Posición X: 100. Posición Y: 300. Tamaño: 20.
Resultados Esperados	Los parámetros de la red móvil y de los nodos se registran adecuadamente
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil y sus nodos.
Fecha de Ejecución	4 de Diciembre, 2006
Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	No se registra información en el Nodo 3.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	Iguales a los resultados esperados

Conclusión 2:	La prueba fue realizada con éxito
Total de Repeticiones:	2

Prueba #5.

Descripción General	Ingresar Nodos Red Móvil
Datos de Entrada	Propiedades de la red móvil: Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 3. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500. Se definen 3 nodos de la siguiente manera: Nodo1: Posición X: 200. Posición Y: 100. Tamaño: 20. Nodo2: Posición X: 300. Posición Y: 300. Tamaño: 20. Nodo3: Posición X: 100. Posición Y: 300. Tamaño: 20.

	Nodo4: Posición X: 400. Posición Y: 100. Tamaño: 25. Nodo5: Posición X: 500. Posición Y: 300. Tamaño: 20.
Resultados Esperados	Los parámetros de la red móvil y de los nodos se registran adecuadamente
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil y sus nodos.
Fecha de Ejecución	4 de Diciembre, 2006
Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	Iguales a los resultados esperados
Conclusión 1:	La prueba fue realizada con éxito
Total de Repeticiones:	1

6.1.1.3.- Caso de Prueba "Definir Agentes en los Nodos".

Descripción de Caso de Prueba	Ingrese a la aplicación MNB. Seleccione en el menú principal la opción Archivo/Nuevo. Seleccione en el menú principal la opción Herramientas/Propiedades Red Móvil. En la ventana de parámetros de la red móvil ingrese toda la información requerida: Protocolo de Enrutamiento Ad-Hoc. Tipo de Capa de Enlace. Tipo de Mac.

	Tipo de Cola de Interfaz. Máximo Paquetes Cola. Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal. Número de Nodos Móviles. Duración Simulación. Tamaño de la Topografía en X. Tamaño de la Topografía en Y. Presione “Aceptar”. Definimos los nodos requeridos y sus posiciones iniciales. Definimos agentes para cada nodo. Seleccione en el menú principal la opción Archivo/Guardar. Verifique con algún editor XML que la información ha sido guardada correctamente.

Prueba #6.

Descripción General	Ingresar Nodos Red Móvil
Datos de Entrada	Propiedades de la red móvil: Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 3. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500.

	Se definen 3 nodos de la siguiente manera: Nodo1: Posición X: 200. Posición Y: 100. Tamaño: 20. Nodo2: Posición X: 300. Posición Y: 300. Tamaño: 20. Nodo3: Posición X: 100. Posición Y: 300. Tamaño: 20. Se definen los siguientes agentes: Agente 1: Nodo: 1 Agente: TCP Agente 2: Nodo: 1 Agente: UDP Agente 3: Nodo: 2 Agente: TCP/Reno Agente 4: Nodo: 3 Agente: TCP
Resultados Esperados	Los parámetros de la red móvil, de los nodos y de los agentes se registran adecuadamente
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil, nodos y agentes.
Fecha de Ejecución	4 de Diciembre, 2006

Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	No se registra información en el Nodo 3.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	Iguales a los resultados esperados
Conclusión 2:	La prueba fue realizada con éxito
Total de Repeticiones:	2

6.1.1.4.- Caso de Prueba "Definir Aplicaciones en los Agentes".

Descripción de Caso de Prueba	Ingrese a la aplicación MNB. Seleccione en el menú principal la opción Archivo/Nuevo. Seleccione en el menú principal la opción Herramientas/Propiedades Red Móvil. En la ventana de parámetros de la red móvil ingrese toda la información requerida: Protocolo de Enrutamiento Ad-Hoc. Tipo de Capa de Enlace. Tipo de Mac. Tipo de Cola de Interfaz. Máximo Paquetes Cola. Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal. Número de Nodos Móviles. Duración Simulación. Tamaño de la Topografía en X. Tamaño de la Topografía en Y. Presione "Aceptar". Definimos los nodos requeridos y sus posiciones iniciales.

	Definimos un agente para cada nodo. Definimos una aplicación para cada agente. Seleccione en el menú principal la opción Archivo/Guardar. Verifique con algún editor XML que la información ha sido guardada correctamente.

Prueba #7.

Descripción General	Ingresar Nodos Red Móvil
Datos de Entrada	Propiedades de la red móvil: Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 3. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500. Se definen 3 nodos de la siguiente manera: Nodo1: Posición X: 200. Posición Y: 100. Tamaño: 20. Nodo2: Posición X: 300. Posición Y: 300. Tamaño: 20.

	Nodo3: Posición X: 100. Posición Y: 300. Tamaño: 20. Se definen los siguientes agentes: Agente 1: Nodo: 1 Agente: TCP Agente 2: Nodo: 2 Agente: UDP Agente 3: Nodo: 3 Agente: TCP Se definen las siguientes aplicaciones: Aplicación 1: Agente: 1 Aplicación: FTP Aplicación 2: Agente: 2 Aplicación: Traffic/CBR Aplicación 3: Agente: 3 Aplicación: Telnet
Resultados Esperados	Los parámetros de la red móvil, de los nodos, de los agentes y de las aplicaciones se registran adecuadamente
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil, nodos, agentes y aplicaciones.
Fecha de Ejecución	5 de Enero, 2007

Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	No se registra información en la Aplicación 2.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	No se registra información en la Aplicación 3.
Conclusión 2:	La prueba debe repetirse
Resultados Obtenidos 3	Iguales a los resultados esperados
Conclusión 3:	La prueba fue realizada con éxito
Total de Repeticiones:	3

6.1.1.5.- Caso de Prueba "Definir Movimientos en los Nodos".

Descripción de Caso de Prueba	Ingrese a la aplicación MNB. Seleccione en el menú principal la opción Archivo/Nuevo. Seleccione en el menú principal la opción Herramientas/Propiedades Red Móvil. En la ventana de parámetros de la red móvil ingrese toda la información requerida: Protocolo de Enrutamiento Ad-Hoc. Tipo de Capa de Enlace. Tipo de Mac. Tipo de Cola de Interfaz. Máximo Paquetes Cola. Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal. Número de Nodos Móviles. Duración Simulación. Tamaño de la Topografía en X. Tamaño de la Topografía en Y.

	Presione “Aceptar”. Definimos los nodos requeridos y sus posiciones iniciales. Definimos movimientos para cada nodo. Seleccione en el menú principal la opción Archivo/Guardar. Verifique con algún editor XML que la información ha sido guardada correctamente.

Prueba #8.

Descripción General	Ingresar Nodos Red Móvil
Datos de Entrada	Propiedades de la red móvil: Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 3. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500. Se definen 3 nodos de la siguiente manera: **Nodo1:** Posición X: 200. Posición Y: 100. Tamaño: 20. **Nodo2:** Posición X: 300. Posición Y: 300.

	Tamaño: 20. **Nodo3:** Posición X: 100. Posición Y: 300. Tamaño: 20. Se definen los siguientes movimientos: **Nodo 1:** Tiempo: 0 Destino X: 200 Destino Y: 100 Velocidad: 20 Tiempo: 10 Destino X: 300 Destino Y: 100 Velocidad: 20 **Nodo 2:** Tiempo: 0 Destino X: 300 Destino Y: 300 Velocidad: 20 Tiempo: 10 Destino X: 400 Destino Y: 300 Velocidad: 20 **Nodo 3:** Tiempo: 0 Destino X: 100 Destino Y: 300 Velocidad: 20 Tiempo: 10 Destino X: 200 Destino Y: 300 Velocidad: 20
Resultados Esperados	Los parámetros de la red móvil, de los nodos, y parámetros de movimientos de los nodos se registran adecuadamente
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil, nodos y movimientos.

Fecha de Ejecución	5 de Enero, 2007
Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	No se registra información del segundo movimiento en el Nodo 1.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	Iguales a los resultados esperados
Conclusión 2:	La prueba fue realizada con éxito
Total de Repeticiones:	2

6.1.2.- CASO DE USO 2. GENERAR ENTRADA PARA NS

6.1.2.1.- Caso de Prueba "Generación de bloques de código en script principal".

Descripción de Caso de Prueba	Ingrese a la aplicación MNB. Seleccione en el menú principal la opción Archivo/Nuevo. Seleccione en el menú principal la opción Herramientas/Propiedades Red Móvil. En la ventana de parámetros de la red móvil ingrese toda la información requerida: Protocolo de Enrutamiento Ad-Hoc. Tipo de Capa de Enlace. Tipo de Mac. Tipo de Cola de Interfaz. Máximo Paquetes Cola. Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal.

	Número de Nodos Móviles. Duración Simulación. Tamaño de la Topografía en X. Tamaño de la Topografía en Y. Presione “Aceptar”. Definimos los nodos requeridos y sus posiciones iniciales. Seleccione en el menú principal la opción Archivo/Guardar. Seleccione en el menú principal la opción Herramientas/Generar Red Móvil. Verificar que archivo de script principal contenga todos los bloques de scripts tcl.

Prueba #9.

Descripción General	Generar Red Móvil
Datos de Entrada	Propiedades de la red móvil: Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 3. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500. Se definen 3 nodos de la siguiente manera: **Nodo1:** Posición X: 200. Posición Y: 100.

	Tamaño: 20. **Nodo2:** Posición X: 300. Posición Y: 300. Tamaño: 20. **Nodo3:** Posición X: 100. Posición Y: 300. Tamaño: 20. **Nodo4:** Posición X: 400. Posición Y: 100. Tamaño: 25. **Nodo5:** Posición X: 500. Posición Y: 300. Tamaño: 20.
Resultados Esperados	El script principal .tcl muestra los siguientes bloques: # Definición de constantes para cada nodo. # Inicializar Variables Globales. # Inicializar Objeto Topography. # Configurar el canal. # Crear la Red Móvil. # Configurar el nodo. # Crear los nodos. # Definir atributos para NAM. # Conexiones entre Agentes. # Movimientos de Nodos. # Indicar cuando la simulación debe finalizar. # proc finish {}. # Iniciar Simulación.
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil y sus nodos.
Fecha de Ejecución	4 de Diciembre, 2006
Responsable de la Prueba	Usuario Analista.

Resultados Obtenidos 1	No se registra el bloque: # Configurar el canal. por lo que al invocar con posterioridad al Network Simulator presenta errores de compliación.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	No se registra el bloque: # Configurar el nodo. por lo que al invocar con posterioridad al Network Simulator presenta errores de compliación.
Conclusión 2:	La prueba debe repetirse
Resultados Obtenidos 3	Iguales a los resultados esperados
Conclusión 3:	La prueba fue realizada con éxito
Total de Repeticiones:	3

6.1.2.2.- Caso de Prueba "Generación de bloques de código en script de movimientos".

Descripción de Caso de Prueba	Ingrese a la aplicación MNB. Seleccione en el menú principal la opción Archivo/Nuevo. Seleccione en el menú principal la opción Herramientas/Propiedades Red Móvil. En la ventana de parámetros de la red móvil ingrese toda la información requerida: Protocolo de Enrutamiento Ad-Hoc. Tipo de Capa de Enlace. Tipo de Mac. Tipo de Cola de Interfaz. Máximo Paquetes Cola. Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal. Número de Nodos Móviles.

	Duración Simulación. Tamaño de la Topografía en X. Tamaño de la Topografía en Y. Presione “Aceptar”. Definimos los nodos requeridos y sus posiciones iniciales. Definimos movimientos para cada nodo. Seleccione en el menú principal la opción Archivo/Guardar. Seleccione en el menú principal la opción Herramientas/Generar Red Móvil. Verificar que archivo de script de movimientos contenga todos los bloques de scripts tcl.

Prueba #10.

Descripción General	Generar Red Móvil
Datos de Entrada	Propiedades de la red móvil: Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel. Número de Nodos Móviles: 3. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500. Se definen 3 nodos de la siguiente manera: Nodo1: Posición X: 200. Posición Y: 100.

	Tamaño: 20. **Nodo2:** Posición X: 300. Posición Y: 300. Tamaño: 20. **Nodo3:** Posición X: 100. Posición Y: 300. Tamaño: 20. Se definen los siguientes movimientos: **Nodo 1:** Tiempo: 0 Destino X: 200 Destino Y: 100 Velocidad: 20 Tiempo: 10 Destino X: 300 Destino Y: 100 Velocidad: 20 **Nodo 2:** Tiempo: 0 Destino X: 300 Destino Y: 300 Velocidad: 20 Tiempo: 10 Destino X: 400 Destino Y: 300 Velocidad: 20 **Nodo 3:** Tiempo: 0 Destino X: 100 Destino Y: 300 Velocidad: 20 Tiempo: 10 Destino X: 200 Destino Y: 300 Velocidad: 20
Resultados Esperados	El script de movimientos .tcl muestra los siguientes

	bloques: # Posición inicial. # Movimientos por tiempo.
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil, nodos y movimientos.
Fecha de Ejecución	5 de Enero, 2007
Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	No se registra información del bloque: # Posición inicial.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	No se registra movimientos del nodo 3 en el bloque: # Movimientos por tiempo.
Conclusión 2:	La prueba debe repetirse
Resultados Obtenidos 3	Iguales a los resultados esperados
Conclusión 3:	La prueba fue realizada con éxito
Total de Repeticiones:	3

6.1.3.- CASO DE USO 3. VISUALIZAR SIMULACIÓN DE NS

6.1.3.1.- Caso de Prueba "Ejecución de archivo de procesamiento por lotes para invocar al visualizador de NS".

Descripción de Caso de Prueba	Ingrese a la aplicación MNB. Seleccione en el menú principal la opción Archivo/Nuevo. Seleccione en el menú principal la opción Herramientas/Propiedades Red Móvil. En la ventana de parámetros de la red móvil ingrese toda la información requerida:

	Protocolo de Enrutamiento Ad-Hoc. Tipo de Capa de Enlace. Tipo de Mac. Tipo de Cola de Interfaz. Máximo Paquetes Cola. Modelo de Antena. Modelo de Radio Propagación. Tipo de Interfaz de Red. Tipo de Canal. Número de Nodos Móviles. Duración Simulación. Tamaño de la Topografía en X. Tamaño de la Topografía en Y. Presione “Aceptar”. Definimos los nodos requeridos y sus posiciones iniciales. Seleccione en el menú principal la opción Archivo/Guardar. Seleccione en el menú principal la opción Herramientas/Ejecutar Red Móvil. Verificar que se invoca al visualizador de NS.

Prueba #11.

Descripción General	Visualizar Red Móvil
Datos de Entrada	Propiedades de la red móvil: Protocolo de Enrutamiento Ad-Hoc: DSR. Tipo de Capa de Enlace: LL. Tipo de Mac: Mac/802_11. Tipo de Cola de Interfaz: Queue/DropTail/PriQueue. Máximo Paquetes Cola: 50. Modelo de Antena: Antenna/OmniAntenna. Modelo de Radio Propagación: Propagation/TwoRayGround. Tipo de Interfaz de Red: Phy/WirelessPhy. Tipo de Canal: Channel/WirelessChannel.

	Número de Nodos Móviles: 3. Duración Simulación: 100. Tamaño de la Topografía en X: 500. Tamaño de la Topografía en Y: 500. Se definen 3 nodos de la siguiente manera: **Nodo1:** Posición X: 200. Posición Y: 100. Tamaño: 20. **Nodo2:** Posición X: 300. Posición Y: 300. Tamaño: 20. **Nodo3:** Posición X: 100. Posición Y: 300. Tamaño: 20. **Nodo4:** Posición X: 400. Posición Y: 100. Tamaño: 25. **Nodo5:** Posición X: 500. Posición Y: 300. Tamaño: 20.
Resultados Esperados	Se visualiza el proceso de simulación.
Condiciones para esta prueba	Ingresar todos los parámetros de la red móvil y sus nodos.
Fecha de Ejecución	4 de Diciembre, 2006
Responsable de la Prueba	Usuario Analista.
Resultados Obtenidos 1	No encuentra archivo .tcl en la carpeta esperada.
Conclusión 1:	La prueba debe repetirse
Resultados Obtenidos 2	La visualización no se realiza debido a que no se carga X-Windows.
Conclusión 2:	La prueba debe repetirse

Resultados Obtenidos 3	Iguales a los resultados esperados
Conclusión 3:	La prueba fue realizada con éxito
Total de Repeticiones:	3

6.2.- RECOMENDACIONES DE SOFTWARE Y DE HARDAWARE

6.2.1. DE LA PLATAFORMA DE SOFTWARE.

Para Cygwin:

Actualmente el dll de Cygwin trabaja con todas las recientes versiones comerciales x86 de 32 bits y de 64 bits de los sistemas operativos de Microsoft Windows, con la excepción de Windows CE.

Se debe tomar en cuenta que el soporte oficial de Cygwin para Windows 95, Windows 98 y Windows ME será discontinuado con la siguiente versión mayor de Cygwin (1.7.0). Actualmente se tiene la versión 1.5.24.

Para NS:

Las recomendaciones de NS son las mismas que para Cygwin.

Para MNB:

Las mismas recomendaciones que para Microsoft Framework 2.0.

Para Microsoft Framework 2.0.

El Framework .Net 2.0 SDK corre en:
Windows Server 2003

Microsoft Windows 2000 (Service Pack 2 recommended)
Windows XP Professional and Home
Windows 98
Nota: Windows Millennium Edition and Windows NT 4.0 Terminal Server no son soportados.

6.2.2. DE LA PLATAFORMA DE HARDWARE.

Se requiere en general:
Computador x86 con procesador Pentium III.
Memoria RAM: 512 MB.
Disco Duro: 200 MB Libres.
Se recomienda Monitor de 17 pul.

CAPÍTULO 7

CONCLUSIONES Y RECOMENDACIONES

7.1.- CONCLUSIONES

- Este proyecto se puede tomar como base si se requiere realizar un análisis de simulaciones de redes satelitales generadas por Network Simulator, ya que se tiene identificada la forma de trabajo de NS, la generación de scripts tcl y la visualización de la simulación.

- Los casos de prueba permiten representar las interacciones entre actores y la aplicación informática, además permite representar los valores de prueba implicados y los resultados esperados.

- El RUP es un método de trabajo a nivel informático que facilita el diseño de aplicaciones, tomando en cuenta una vista panorámica de la situación real del problema a resolver, es sistemático y de amplia relación con las variantes que en cada aplicación se desee implementar.

- El análisis de los requerimientos es una de las fases más importantes en el desarrollo de software donde deben estar claramente identificados todos los participantes del proyecto y sus usuarios.

- Los diagramas de casos de uso permiten visualizar de una manera panorámica el problema a resolver, contextualizando por medios gráficos el entorno de trabajo con mayor facilidad.

- Las herramientas que se usen en el desarrollo de la aplicación deben tener una gran apertura a las ideas creativas de los programadores, además deben facilitar la escritura de instrucciones en poco código.

- Las pruebas de funcionalidad son un requerimiento básico para medir el desempeño de la aplicación informática y posteriormente hacer ajustes claves que la optimicen.

7.2.- RECOMENDACIONES

- Si se requiere usar nuevos tipos de agentes para la capa 4 del modelo TCP/IP (Transporte), se puede programar un nuevo agente, para lo cual se debe bajar la última versión del código fuente disponible en http://www.isi.edu/nsnam/. La programación se la realiza en TCL y OTCL, debe aumentar el código para estos nuevos agentes y compilar el NS con las herramientas propias del NS.

- El mismo caso, si se requiere usar nuevos tipos de aplicaciones para la capa 5 del modelo TCP/IP (Aplicación), se puede programar una nueva aplicación, para lo cual se debe bajar la última versión del código fuente disponible en http://www.isi.edu/nsnam/. La programación se la realiza en TCL y OTCL, debe aumentar el código para estas nuevas aplicaciones y compilar el NS con las herramientas propias del NS.

- De la misma manera si usted quiere aumentar un nuevo protocolo de enrutamieno ad-hoc para redes móviles aparte del DSDV, AODV, TORA y DSR, puede programar en TCL y OTCL e incrementar el código fuente del NS.

- A pesar de las múltiples formas de diseñar software en la actualidad, la mejor forma de contextualizar es el RUP, por lo cual se recomienda su utilización.

- El nivel de detalle dentro de los diagramas de casos de uso, de secuencia y de colaboración deben ser lo más detallados y con el máximo nivel de profundidad posible para que se pueda entender el funcionamiento claro y preciso de la aplicación a realizar.

- Al terminar los diagramas de casos de uso, de secuencia y de colaboración de debe tener una nueva revisión completa, de modo que no se escapen detalles importantes, y es necesario hacerlo así cada vez que se termine una fase de revisión de dichos diagramas.

- Visual Studio .Net es una plataforma de programación y diseño de última generación, permite desarrollar aplicaciones de manera visual y gráfica. Soporta diseño para Windows y para Web con componentes claves de estructura de datos. Puede conectarse a diferentes motores de bases de datos haciendo transparente la interfaz al usuario. La escritura de código es más rápida y al presentar paneles de control hace que la interfaz sea más amigable. Se recomienda su uso y constante actualización de conocimientos respecto a la herramienta.

REFERENCIAS BIBLIOGRAFICAS

1. Booch, Rumbaugh, Jacobson. (1999). El Lenguaje Unificado de Modelado. Addison Wesley. Estados Unidos.

2. Cetina Victor. (2006). Tutorial de Protocolos de Ruteo para Redes Inalámbricas Móviles Ad-Hoc. Facultad de Matemáticas Universidad Autónoma de Yucatán, Mérida.

3. Larman Craig. (2005). UML Y PATRONES Introducción al análisis y diseño orientado a objetos. Prentice Hall.

4. Larman, Craig. (2004). Agile and iterative development: a manager's guide. Mc Graw Hill, Estados Unidos. Fifth edition.

5. León Antonio, Subiela Roberto. (2006). Simulación de Protocolos de Encaminamiento en Redes Móviles Ad Hoc con NS-2. Universidad Politécnica de Valencia. España.

6. McConnell, Steve. (1996). Desarrollo y Gestión de Proyectos Informáticos. McGraw Hill. Estados Unidos. Segunda Edición.

7. Ortuño Miguel. (2006). Encaminamiento en Redes Ad Hoc. Universidad Rey Juan Carlos. España. http://gsyc.escet.urjc.es/. Grupo de Sistemas y Comunicaciones Departamento de Informática, Estadística y Telemática Universidad Rey Juan Carlos.

8. Perkins, C. E. y Bhagwat, P. (1994). Highly Dynamic Destination-Sequenced Distance-Vector Routign (DSDV) for Mobile Computers. Comp Común.

9. Rational Software Corporation. (2001). Principles of Managing Iterative Development. Rational Corporation. First Edition.

10. Red Hat, Inc. (2003). Cygwin User´s Guide Red Hat.

11. The VINT Project. (2003). The NS Manual.

12. Cygwin Project. (2000). http://www.cygwin.com/.

13. Cygwin Project. (2000). http://www.isi.edu/nsnam/ns/.

14. Microsoft Corporation. (2006). http://msdn2.microsoft.com/en-us/netframework/aa569264.aspx.

ANEXOS

ANEXO A: GUIA DE INSTALACIÓN DE CYGWIN.

1.- QUÉ ES CYGWIN

Cygwin es un ambiente semejante a Linux para ambientes Windows. Consiste de un DLL (cygwin1.dll), el cual actúa como una capa de emulación brindando funcionalidad POSIX (Portable Operating System Interface) de llamadas al sistema y una colección de herramientas, las cuales proporcionan una forma real de ver y sentir el ambiente Linux. El DLL Cygwin trabaja con todas las versiones x86 de Windows desde Windows 95.

Una vez instalado Cygwin, los usuarios tienen acceso a la mayoría de utilitarios estándar de Linux.

2.- DESCARGAR INSTALADOR

- Descargar de: http://www.cygwin.com/setup.exe
- Guardar en un directorio local, por ejemplo “C:/CygIns”.

3.- DESCARGAR PAQUETES A UN DIRECTORIO LOCAL

- Ejecutar “setup.exe” de un directorio local. Debe ser la versión 2.457.2.2 o superior.
- La primera pantalla nos indica la versión del Instalador. Presionamos “Siguiente”.
- Indicamos la fuente de descarga, en este caso: “Descarga sin instalación”.
- Seleccionamos el directorio local donde descargar los archivos de instalación: “C:/CygIns”.
- Especificamos la forma de conexión al Internet adecuada.
- El programa buscara automáticamente los sitos posibles de descarga; podemos utilizar: ftp://ftp.cise.ufl.edu.

- El programa descargará la lista de archivos de instalación del sitio especificado y mostrara las opciones de instalación.
- Si usted tiene unos 300 MB de disco puede escoger descargar todo, para lo cual de clic en “Default” para cambiar a “Install” en la categoría “All”.
- Debe seleccionar por lo menos los siguientes paquetes según la versión de NS que necesite:

	NS 2.27	**NS 2.29**
CYGWIN	CYGWIN 1.5.12(0.116/4/2) 2004-11-10 08:34	CYGWIN 1.5.18(0.132/4/2) 2005-07-02 20:30
CYGWIN SETUP	2.457.2.2	2.510.2.2
PAQ. MININOS	Base gawk gzip tar Devel gcc gcc-g++ make Interpreters perl Libs w32api Utils patch Removed Packages XFree86-base XFree86-bin XFree86-prog XFree86-lib	Base gawk gzip tar Devel gcc gcc-g++ make Interpreters perl Libs w32api Utils patch X11 xorg-x11-base xorg-x11-bin xorg-x11-bin-dlls xorg-x11-devel

	XFree86-etc	xorg-x11-libs-data xorg-x11-etc

- Debe dar click en “Default” para cambiar a “Install” en cada categoría.
- Damos “Siguiente” y el programa descargará los archivos de instalación pedidos en el directorio indicado.
- Aparecerá un mensaje de “Descarga Completa”. Si no aparece este mensaje, no se descargaron todos los archivos, inténtelo de nuevo ejecutando “setup.exe” y siga los mismos pasos de esta sección.

4.- INSTALAR CYGWIN

- Ingresar como usuario: “Administrador”, a Windows XP, Windows 2000.
- Poner las siguientes variables de entorno:
  - DISPLAY = 127.0.0.1:0.0
  - PATH = %PATH%;C:\cygwin\bin;C:\cygwin\usr\X11R6\bin
- Ejecutar “setup.exe” de un directorio local.
- A la primera pantalla presionamos “Siguiente”.
- Indicamos la fuente de descarga, en este caso: “Instalar desde directorio local”.
- Seleccionamos el directorio local que será la raíz de ”Cygwin”, generalmente es “C:\cygwin”, si lo cambia es aconsejable que el nombre no tenga más de ocho caracteres, no contenga espacios en blanco ni caracteres especiales.
- Seleccionamos además UNIX text type.
- Seleccionamos el directorio local donde descargamos los archivos de instalación: “C:/CygIns”. (Apartado Anterior).
- El programa buscara automáticamente los archivos descargados y presentará las opciones de instalación por Categoría.
- Debe dar clic en “Default” para cambiar a “Install” en “All” al inicio de las categorías.

- Debe seleccionar por lo menos los siguientes paquetes según la versión de NS que necesite:

	NS 2.27	**NS 2.29**
CYGWIN	CYGWIN 1.5.12(0.116/4/2) 2004-11-10 08:34	CYGWIN 1.5.18(0.132/4/2) 2005-07-02 20:30
CYGWIN SETUP	2.457.2.2	2.510.2.2
PAQ. MININOS	Base gawk gzip tar Devel gcc gcc-g++ make Interpreters perl Libs w32api Utils patch Removed Packages XFree86-base XFree86-bin XFree86-prog XFree86-lib XFree86-etc	Base gawk gzip tar Devel gcc gcc-g++ make Interpreters perl Libs w32api Utils patch X11 xorg-x11-base xorg-x11-bin xorg-x11-bin-dlls xorg-x11-devel xorg-x11-libs-data xorg-x11-etc

- Damos clic en “Siguiente” y el programa instalará los archivos pedidos y realizará opciones de post-instalación

- Aparecerá un mensaje de “Instalación Completa”. Si no aparece este mensaje, no se instalaron todos los archivos, inténtelo de nuevo ejecutando “setup.exe” y siga los mismos pasos de esta sección.
- Se crean acceso directos en el menú y en el escritorio.

5.- NOTAS

- No es conveniente tener un nombre de dominio o nombre grupo de trabajo igual al nombre de usuario. En este caso debe aumentar una entrada en forma manual al archivo “/etc/passwd”.
- Al ejecutar “Cygwin.bat”, se debe crear el directorio “/home/userxxx”, donde “userxxx” es el usuario Windows, de no ser así tiene algun problema con el usuario de Windows.
- Es aconsejable que los nombres usuarios de Windows no tengan espacios en blanco ni caracteres especiales.
- Es aconsejable que el directorio de instalación de Cygwin no tengan espacios en blanco ni caracteres especiales.

6.- COMANDOS

- Información. Nombre del Kernel.

  uname -s

  CYGWIN_NT-5.1

- Información. Nombre de la Máquina.

  uname -n

  SERVER

- Información. Release del Kernel.

  uname -r

  1.5.12(0.116/4/2)

- Información. Versión del Kernel.

  uname -n

  2004-11-10 08:34

ANEXO B: GUIA DE INSTALACIÓN DE NS.

NS es un simulador de eventos discreto destinado a la investigación de redes. NS proporciona soporte sustancial para simulaciones TCP, simulaciones de enrutamiento y protocolos multicast sobre redes cableadas e inalámbricas.

NS comenzó como una variante del simulador de red REAL en 1989 y ha evolucionado sustancialmente en los últimos años. En 1995 el desarrollo de NS fue apoyado por DARPA a través del proyecto VINT de LBL, Xerox Parc, UCB and USC/ISI. Actualmente el desarrollo de NS es apoyado por DARPA a través del proyecto SAMAN y es apoyado por NSF a través del proyecto CONSER. Ambos proyectos trabajan en colaboración con otras investigaciones incluyendo ACIRI.

NS siempre ha incluido contribuciones importantes de otros investigadores incluyendo código perteneciente a los proyectos UCB Daedelus y CMU Monarch y también de Sun Microsystems.

1.- COMPROBACIÓN DE CYGWIN

- Es recomendable seguir las instrucciones mostradas en:

  http://www.sims.berkeley.edu/~christin/ns-cygwin.shtml
- Necesita cygwin-1.3.12 o posterior.

  uname -r

  1.5.12(0.116/4/2)
- Asegúrese que cygwin este instalado con UNIX text type, ya que DOS text type no ha sido probado totalmente. Ejecute el comando:

  mount | grep textmode

  si no retorna nada, debe estar instalado en forma correcta, si retorna algo esta en DOS text type y es recomendable reinstalar Cygwin poniendo esta opción.

- Asegúrese de que los siguientes paquetes se encuentren instalados:

NS 2.27	**NS 2.29**
Base	Base
gawk	gawk
gzip	gzip
tar	tar
Devel	Devel
gcc	gcc
gcc-g++	gcc-g++
make	make
Interpreters	Interpreters
perl	perl
Libs	Libs
w32api	w32api
Utils	Utils
patch	patch
Removed Packages	X11
XFree86-base	xorg-x11-base
XFree86-bin	xorg-x11-bin
XFree86-prog	xorg-x11-bin-dlls
XFree86-lib	xorg-x11-devel
XFree86-etc	xorg-x11-libs-data
	xorg-x11-etc

Ejecute el siguiente comando:

```
cygcheck -c XFree86-bin  | grep XFree86-bin
cygcheck -c xorg-x11-base  | grep xorg-x11-base
```

Si no retorna nada, debe instalar este paquete.

- Ejecute: /usr/X11R6/bin/startxwin.bat desde cygwin.

2.- DESCARGAR ARCHIVOS DE NS:

NS 2.27	NS 2.29
• ns-allinone-2.27.tar.tar descárgelo de: http://www.isi.edu/nsnam/dist/ns-allinone-2.27.tar.gz. • install.patch debido a un bug en el instalador, descárguelo de: http://www.sims.berkeley.edu/~christin/software/install.patch.	• ns-allinone-2.29.tar.tar

3.- INSTALACIÓN DE NS 2.27:

NS 2.27	NS 2.29
Copie los arhivos: • ns-allinone-2.27.tar.tar • install-patch al directorio: “/usr”.	Copie los arhivos: • ns-allinone-2.29.tar.tar al directorio: “/usr”.
Ejecute: /usr/X11R6/bin/startxwin.bat desde cygwin.	Ejecute: /usr/X11R6/bin/startxwin.bat desde cygwin.
Ejecute los siguientes comandos en forma separada: cd /usr gzip -d -c ns-allinone-2.27.tar.tar | tar xvf – cd ns-allinone-2.27 patch -p0 < ../install.patch	Ejecute los siguientes comandos en forma separada: cd /usr gzip -d -c ns-allinone-2.29.tar.tar | tar xvf – cd ns-allinone-2.29
Ahora puede ejecutar la instalación con el commando: ./install	Ahora puede ejecutar la instalación con el commando: ./install

Actualice las variables de medio ambiente: export NS_HOME=/usr/ns-allinone-2.27 export PATH=$NS_HOME/tcl8.4.5/unix:$NS_HOME/tk8.4.5/unix:$NS_HOME/bin:$PATH export LD_LIBRARY_PATH=$NS_HOME/tcl8.4.5/unix:$NS_HOME/tk8.4.5/unix:$NS_HOME/otcl-1.8:$NS_HOME/lib:$LD_LIBRARY_PATH export TCL_LIBRARY=$NS_HOME/tcl8.4.5/library:$TCL_LIBRARY	Actualice las variables de medio ambiente: export NS_HOME=/usr/ns-allinone-2.29 export PATH=$NS_HOME/tcl8.4.11/unix:$NS_HOME/tk8.4.11/unix:$NS_HOME/bin:$PATH export LD_LIBRARY_PATH=$NS_HOME/tcl8.4.11/unix:$NS_HOME/tk8.4.11/unix:$NS_HOME/otcl-1.11:$NS_HOME/lib:$LD_LIBRARY_PATH export TCL_LIBRARY=$NS_HOME/tcl8.4.11/library:$TCL_LIBRARY
Para evitar tipear estos comandos los podemos añadir al archivo “/home/userxxx/.bashrc”.	Para evitar tipear estos comandos los podemos añadir al archivo “/home/userxxx/.bashrc”.

ANEXO C: GUÍA DEL MODELO DE MOVILIDAD DE NETWORK SIMULATOR

1. DEFINICIÓN DE CONSTANTES

Generalmente para definir constantes se utiliza una sola variable denominada “val”, la cual es un arreglo de valores. La asignación de valores se la realiza generalmente al inicio del script tcl.

Para indicar que se realiza un comentario en el script tcl, se escribe el carácter “#”, y se puede escribir un texto como comentario hasta el final de la línea.

```
# ======================================================================
# Definición de constantes para cada nodo.
# ======================================================================
set val(AdHocRoutingProtocol) DSDV                      # Protocolo de enrutamiento ad-hoc
set val(LinkLayerType) LL                               # Tipo de capa de enlace
set val(MacType) Mac/802_11                             # Tipo de Mac
set val(InterfaceQueueType) Queue/DropTail/PriQueue # Tipo de cola de  interfaz
set val(MaxPackets) 50                  # Número máximo de paquetes en la cola de interfaz
set val(AntennaModel) Antenna/OmniAntenna               # Modelo de antena
set val(RadioPropagationModel) Propagation/TwoRayGround    # Modelo de radiopropagación
set val(NetworkInterfaceType) Phy/WirelessPhy           # Tipo de interfaz de red
set val(ChannelType) Channel/WirelessChannel            # Tipo de canal
set val(MobileNodes) 2                                  # Número de nodos móviles
set val(SimulationTime) 150                             # Tiempo total de la simulación
set val(SizeTopgraphyX) 500                             # Tamaño en X de la topografía
set val(SizeTopgraphyY) 500                             # Tamaño en Y de la topografía

set val(traceFile) "SimpleWireless.tr"                  #Nombre para el archivo de traza
set val(traceNAMFile) "SimpleWireless.nam"              #Nombre para el archivo de traza nam
set val(traceConFile) "SimpleWireless_con.tcl"          #Nombre para el script de conexiones
set val(traceMovFile) "SimpleWireless_mov.tcl"          #Nombre para el script de movimientos
```

2. INICIALIZAR VARIABLES GLOBALES

Es necesario inicializar algunas variables globales, las cuales se indican a continuación:

```
# ======================================================================
# Inicializar Variables Globales.
# ======================================================================
```

```
set ns [new Simulator]
set tracefd [open $val(traceFile) w]
$ns trace-all $tracefd
set namtrace [open $val(traceNAMFile) w]
$ns namtrace-all $namtrace
$ns namtrace-all-wireless $namtrace $val(SizeTopgraphyX) $val(SizeTopgraphyY)
$ns use-newtrace
```

La variables “ns”, nos indica que se ha creado una instancia del simulador. Es la variable más importante ya que es la raíz de la configuración de la red móvil como veremos mas adelante.

La variables “tracefd”, nos indica el descriptor de un archivo que lo usaremos para guardar las trazas de seguimiento de la simulación.

El método “trace-all”, nos asocia a la instancia del simulador el descriptor de la traza.

La variable “namtrace”, nos indica el descriptor de archivo que lo usaremos para guardar las trazas de seguimiento de la simulación para el visualizador nam.

El método “namtrace-all”, nos asocia a la instancia del simulador el descriptor de la traza del visualizador nam.

El método “namtrace-all-wireless”, nos asocia a la instancia del simulador el descriptor de la traza del visualizador nam. Se utiliza para ser más explicito e indicar que la simulación indicada es de un modelo inalámbrico, por lo cual se indica también el tamaño de la topografía en “x” y también en “y”.

El método “use-newtrace”, indica al simulador que use el nuevo formato de traza para el visualizador nam.

3. INICIALIZAR OBJETO TOPOGRAFÍA

Es necesario crear un objeto topografía. Para indicar las coordenadas en “x” y en “y” en donde el modelo de movilidad puede desplazarse con sus nodos, se usa el método “load_flatgrid” de la variable de tipografía.

```
# ======================================================================
# Inicializar Objeto Topography.
# ======================================================================
set varTopography [new Topography]
$varTopography load_flatgrid $val(SizeTopgraphyX) $val(SizeTopgraphyY)
```

4. CONFIGURAR EL CANAL

Es necesario crear un objeto tipo de canal. Aunque el modelo de movilidad de NS solo maneja un tipo de canal para todos los nodos de la red móvil. Se utiliza este objeto para en un futuro pasar la instancia de un nuevo canal el momento de crear un nodo de la red móvil.

```
# ======================================================================
# Configurar el canal.
# ======================================================================
set varChannel [new $val(ChannelType)]
```

5.- ARREGLO DE COLORES

En el modelo de la red, es posible utilizar colores para identificar nodos de la red móvil en forma individual en el visualizador nam. Para lo cual es necesario especificar los colores que se pueden usar a la instancia del simulador “ns”.

```
# ======================================================================
# Arreglo de Colores.
# ======================================================================
$ns color 0 blue
$ns color 1 red
$ns color 2 chocolate
$ns color 3 red
$ns color 4 brown
$ns color 5 tan
$ns color 6 gold
$ns color 7 black
```

6.- CREAR LA RED MÓVIL

Para crea el modelo de movilidad en Network Simulator se invoca al método “create-god”, el cual crea este modelo de movilidad con el número de nodos móviles indicados en el parámetro de este método.

```
# ==================================================================
# Crear la Red Móvil.
# ==================================================================
create-god $val(MobileNodes)
```

7.- CONFIGURAR EL NODO

Antes de crear un nodo es necesario configurar los parámetros con los cuales se va a crear el nodo móvil dentro de nuestra red. Para esto se utiliza el método “node-config” de la instancia de la simulación “ns”. En este método se indican todos los parámetros necesarios para crea el nodo con el método “node”, que veremos más adelante.

```
# ==================================================================
# Configurar el nodo.
# ==================================================================
$ns node-config \
                -adhocRouting $val(AdHocRoutingProtocol) \
                -llType $val(LinkLayerType) \
                -macType $val(MacType) \
                -ifqType $val(InterfaceQueueType) \
                -ifqLen $val(MaxPackets) \
                -antType $val(AntennaModel) \
                -propType $val(RadioPropagationModel) \
                -phyType $val(NetworkInterfaceType) \
                -channel $varChannel \
                -topoInstance $varTopography \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace ON
```

Los parámetros del método “node-config” son:

Columna	**Significado**	**Ejemplo**
adhocRouting	Protocolo de enrutamiento ad-hoc	DSDV, Destination-Sequenced Distance Vector. DSR, Dynamic Source Routing. TORA, Temporally-Ordered Routing Algorithm. AODV, Ad hoc On-Demand Distance Vector.
llType	Tipo de capa de enlace	LL, LL/Sat
macType	Tipo de Mac	Mac/802_11, Mac/Simple, Mac/Csma/Ca, Mac/Sat, Mac/Sat/UnslottedAloha, Mac/Tdma
ifqType	Tipo de cola de interfaz	Queue/DropTail/PriQueue, Queue/DropTail, CMUPriQueue
ifqLen	Número máximo de paquetes en la cola de interfaz	50
antType	Modelo de antena	Antenna/OmniAntenna
propType	Modelo de radiopropagación	Propagation/TwoRayGround, Propagation/FreeSpace, Propagation/Shadowing, Propagation/ShadowingVis
phyType	Tipo de interfaz de red	Phy/WirelessPhy, Phy/Sat NetIf/SharedMedia
channel	Instancia del canal a usar	
topoInstance	Instancia de la topografía a usar	
agentTrace	Indicador para que en la generación de trazas incluya información de agentes	ON / OFF

routerTrace	Indicador para que en la generación de trazas incluya información de enrutamiento	ON / OFF
macTrace	Indicador para que en la generación de trazas incluya información de la capa mac	ON / OFF
movementTrace	Indicador para que en la generación de trazas incluya información de movimiento de los nodos	ON / OFF

8.- CREAR LOS NODOS

Una vez invocado al método “node-config” del simulador, ya se puede crear nodos invocando al método “node” del simulador. Es conveniente organizar todos los nodos en un mismo nombre de variable, para lo cual se utiliza un arreglo denominado “node”. Se invoca al método “random-motion” con parámetro “0” para indicar que el movimiento del nodo no se realiza en base a un modelo aleatorio sino que va a ser explícito.

```
# ==========================================================
# Crear los nodos.
# ==========================================================
set node(1) [$ns node]
$node(1) random-motion 0

set node(2) [$ns node]
$node(2) random-motion 0
```

9.- DEFINIR ATRIBUTOS PARA NAM

Si requiere visualizar el proceso de simulación, usted puede manejar algunos atributos para una mejor visualización. Estos atributos no intervienen en el proceso de generación de paquetes sino solo en la visualización.

```
# ==========================================================
# Definir atributos para NAM.
# ==========================================================
$ns initial_node_pos $node(1) 20
$ns at 0.0 "$node(1) label Nodo1"
$node(1) color "black"
$node(1) shape "circle"

$ns initial_node_pos $node(2) 20
$ns at 0.0 "$node(2) label Nodo2"
$node(2) color "black"
$node(2) shape "circle"
```

Se utiliza el método “inicial_node_pos” del simulador para especificar el tamaño del nodo en el gráfico para el visualizador.

Se utiliza el método “label” del nodo para asignar una etiqueta al nodo en la visualización.

Se utiliza el método “at” del simulador para indicar que debe ejecutar una acción en un tiempo específico. El tiempo se especifica en el primer parámetro y la acción en el segundo parámetro.

10.- CONEXIONES ENTRE AGENTES

Es útil guardar el script de conexiones entre agentes en un archivo (script) aparte, ya que estas conexiones se las puede reutilizar en otros ejemplos de redes y solo se invocan indicando el nombre del archivo.

```
# ==========================================================
# Conexiones entre Agentes.
# ==========================================================
puts "Cargando Patron de Conexiones..."
source $val(traceConFile)
```

El método “source” indica que se tome en cuenta el script que esta indicado en el parámetro “archivo”.

11.- MOVIMIENTOS DE NODOS

Es útil guardar el script de movimientos de los nodos en un archivo (script) aparte, ya que estos movimientos se los puede reutilizar en otros ejemplos de redes y solo se invocan indicando el nombre del archivo.

```
# ==========================================================
# Movimientos de Nodos.
# ==========================================================
puts "Cargando Patron de Movimientos..."
source $val(traceMovFile)
```

El método “source” indica que se tome en cuenta el script que esta indicado en el parámetro “archivo”.

12.- INDICAR CUANDO LA SIMULACIÓN DEBE FINALIZAR

Se debe indicar cuando la simulación debe terminar e indicar de este suceso a los nodos. Es útil escribir un método denominado “finísh” para indicar algunos aspectos que se deben realizar para terminar la simulación.

```
# ==========================================================
# Indicar cuando la simulación debe finalizar.
# ==========================================================
$ns at $val(SimulationTime).000000001 "$node(1) reset"
$ns at $val(SimulationTime).000000001 "$node(2) reset"
$ns at $val(SimulationTime).000000001 "finish"
$ns at $val(SimulationTime).000000001 "puts \"Simulacion Terminada...\" ; $ns halt"
```

Se utiliza la variable “val” con el índice “SimulationTime” para indicar el tiempo de finalización.

Se utiliza el método “reset” del nodo para indicar que el trabajo del nodo ha terminado.

Se utiliza el método “at” del simulador para indicar cuando se debe ejecutar una acción.

Se utiliza el método “finísh” para aspectos que se deben realizar al terminar la simulación.

13.- MÉTODO DE FINALIZACÓN DE LA SIMULACIÓN

Es útil escribir un método denominado “finísh” para indicar algunos aspectos que se deben realizar para terminar la simulación.

```
# ================================================================
# proc finish {}.
# ================================================================
proc finish {} {
   global ns tracefd namtrace

   $ns flush-trace
   close $tracefd
   close $namtrace
   puts "Simulacion Terminada..."
   $ns halt
}
```

La instrucción “global” indica que se va a realizar referencia a variables globales definidas fuera de este método.

El método “flush-trace” del simulador indica que la información de trazas pendiente del buffer debe escribirse en el archivo respectivo.

El método “close” indica que se va a cerrar un archivo, terminado la escritura en este archivo.

El método “halt” del simulador indica que se termina el proceso de simulación.

14.- INICIAR SIMULACIÓN

Se debe indicar explícitamente que el proceso de simulación debe ejecutarse.

```
# ================================================================
# Iniciar Simulación.
# ================================================================
puts "Simulation Iniciada..."
$ns run
```

El método “run” del simulador indica que el proceso de simulación debe iniciar con su ejecución.

15.- AGENTES

Se debe crear los agentes (capa 4 del modelo TCP/IP “Transporte”) e indicar a que nodos están asociados. Es conveniente organizar todos los agentes en un mismo nombre de variable, para lo cual se utiliza un arreglo denominado “varAgent”.

```
# ================================================================
# Agentes.
# ================================================================
set varAgent(1) [new Agent/TCP]
$ns attach-agent $node(1) $varAgent(1)

set varAgent(2) [new Agent/TCPSink]
$ns attach-agent $node(2) $varAgent(2)
```

La instrucción “new” crea en este caso un agente de tipo “TCP” en un caso y en el otro un agente de tipo “TCPSink”.

El método “attach-agent” del simulador asocia el nodo descrito en el primer parámetro con el agente descrito en el segundo parámetro.

16.- APLICACIONES

Se debe crear las aplicaciones (capa 5 del modelo TCP/IP “Aplicación”) e indicar a que agentes están asociados. Es conveniente organizar todas las aplicaciones en un mismo nombre de variable, para lo cual se utiliza un arreglo denominado “varApplication”.

```
# ================================================================
# Aplicaciones.
# ================================================================
set varApplication(1) [new Application/FTP]
```

```
$varApplication(1) attach-agent $varAgent(1)
$ns at 10 "$varApplication(1) start"
```

La instrucción “new” crea en este caso una aplicación de tipo “FTP”.

El método “attach-agent” de la aplicación asocia el agente descrito en el primer parámetro.

17.- CONEXIONES ENTRE AGENTES

Se debe conectar los agentes (capa 4 del modelo TCP/IP “Transporte”).

```
# ================================================================
# Conexiones Agentes.
# ================================================================
$ns connect $varAgent(1) $varAgent(2)
```

El método “connect” del simulador conecta el agente descrito en el primer parámetro con el agente descrito en el segundo parámetro.

18.- POSICIÓN INICIAL DE LOS NODOS

Se debe indicar la posición inicial de los nodos en la topografía de la red móvil.

```
# ================================================================
# Posición inicial.
# ================================================================
$node(1) set X_ 5
$node(1) set Y_ 2
$node(1) set Z_ 0
$node(2) set X_ 390
$node(2) set Y_ 385
$node(2) set Z_ 0
```

El método “set” del nodo asigna un valor indicado en el segundo parámetro al atributo indicado en el primer parámetro.

“X_” denota la posición del nodo en las coordenadas de las “X”.

“Y_” denota la posición del nodo en las coordenadas de las “Y”.

“Z_” denota la posición del nodo en las coordenadas de las “Z”.

19.- MOVIMIENTOS DE LOS NODOS POR TIEMPO

Se debe indicar los movimientos que realizan los nodos a través del tiempo.

```
# ============================================================
# Movimientos por tiempo.
# ============================================================
$ns at 10 "$node(1) setdest 20 18 1"
$ns at 100 "$node(2) setdest 490 480 15"
$ns at 50 "$node(2) setdest 25 20 15"
```

El método “setdest” del nodo indica un cambio de movimiento del nodo hacia la posición “x” indicada en el primer parámetro, posición “y” indicada en el segundo parámetro y con una velocidad indicada en el tercer parámetro.