



UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO INGENIERÍA EN INFORMÁTICA
INGENIERÍA CIVIL INFORMÁTICA



REDES DE COMUNICACIÓN

Guía Práctica de OMNeT++

Profesora: Rosa Muñoz
Ayudante: Luciano Hidalgo

Santiago, Septiembre de 2011

Tabla de contenido

INTRODUCCIÓN	1
I. INSTALACIÓN DE LA HERRAMIENTA	2
1. REQUISITOS PREVIOS.....	2
2. INSTALANDO OMNeT++.....	2
3. CONFIGURANDO OMNeT++	3
4. VERIFICANDO LA INSTALACIÓN.....	4
II. IDE DE OMNeT++.....	7
1. EJECUCIÓN DEL IDE	7
2. COMPONENTES DEL IDE.....	8
<i>Editor .NED</i>	9
<i>Editor .INI</i>	10
<i>Información adicional</i>	11
III. ENTORNO DE EJECUCIÓN GRÁFICO TKENV.....	14
1. INICIANDO TKENV	14
2. COMPONENTES DE TKENV	15
<i>Ventana Principal</i>	15
<i>Ventana de Módulos</i>	16
IV. EJEMPLO DE PROGRAMA	18
1. ARCHIVOS .NED	19
<i>Aplicación .NED</i>	19
<i>Intermedio .NED</i>	20
<i>Enlace .NED</i>	20
<i>Host .NED</i>	21
<i>Sistema .NED</i>	23
2. ARCHIVOS .CC	25
<i>Aplicación .CC</i>	25
<i>Intermedio .CC</i>	27
<i>Enlace .CC</i>	28
3. ARCHIVO .INI.....	29
V. EJECUTAR EJEMPLO	30
VI. INFORMACIÓN ADICIONAL.....	33
VII. CRÉDITOS	33

Introducción

El presente documento explica el funcionamiento básico y la instalación de la herramienta de simulación de redes, OMNeT++ con la cual se trabaja en laboratorios del curso de Redes de Comunicación de la carrera de Ingeniería Civil Informática en la Universidad de Santiago de Chile, esta guía ha sido desarrollada para el trabajo en el entorno de Windows, tomando como base el Sistema Operativo Windows 7, aunque la herramienta se encuentra disponible para Mac OS X y para las distintas distribuciones de Linux.

OMNeT++ es un simulador que divide el trabajo en módulos, haciendo la analogía de las distintas arquitecturas de capas que se manejan en el tráfico de redes de telecomunicaciones, es útil para simular desde arquitecturas de comunicación básicas, como un sistema de dos estaciones y un canal, hasta sistemas complejos con varios nodos y subredes incluidas. Por otro lado permite simular las distintas capas que permiten la comunicación de redes dentro de una sola estación (computador).

Actualmente la herramienta de desarrollo OMNeT++, se encuentra en su versión 4.1, y en base a ella se ha desarrollado esta guía.

Para mayor información e instrucciones de instalación para otras plataformas descargue el programa en el link que se suministra en el próximo capítulo y revise la carpeta “docs”, en la cual se encuentra la guía de instalación de OMNeT++, el manual de usuario detallado y otros documentos relevantes para quien desee profundizar sus estudios con esta herramienta.

I. Instalación de la Herramienta

OMNeT++ es una herramienta libre y puede ser descargada gratuitamente desde la página oficial del proyecto: <http://www.omnetpp.org/>

1. Requisitos Previos

El único componente de software que se necesita instalar antes de OMNeT++ es el Java Runtime (JRE), todos los otros componentes, incluyendo el compilador de C++ vienen incluidos en el paquete de descarga de OMNeT++.

Descargue Java 5.0 o una versión superior desde <http://www.java.com> e instálela antes de proceder, los componentes de Java se necesitan para el entorno de desarrollo y simulación basado en Eclipse.

2. Instalando OMNeT++

Descargue el código fuente de OMNeT++ desde <http://omnetpp.org>, asegúrese de seleccionar el archivo específico para Windows, se encontrará bajo el nombre `omnetpp-4.1-src-windows.zip`.

El paquete debiera contener todos los componentes necesarios para el funcionamiento e instalación de OMNeT++, entre los archivos se incluye el compilador de C++, un simulador de terminal o consola de Linux y todas las bibliotecas y programas necesarios para trabajar con OMNeT++.

Una vez descargado copie el archivo en el directorio donde desea instalarlo.

IMPORTANTE: Copie el archivo en un directorio cuya ruta completa no contenga ningún espacio, por ejemplo, NO poner OMNeT como subcarpeta de C:\Archivos de Programa, una carpeta donde el programa funcionará correctamente podría ser C:\OMNeT.

Extraiga el archivo en la carpeta deseada, para ello haga clic en el archivo .zip en el Explorador de Windows y seleccione “Extraer Todo” en el menú, también puede usar descompresores específicos como Winrar o 7zip.

Una vez extraído asegúrese de que el nombre del directorio es `omnetpp-4.1` en caso de no ser así, cámbiele el nombre.

Al explorar el contenido del directorio resultante debiera ver directorios llamados doc, images, include, msys, etc, y archivos con el nombre de mingwenv.cmd, configure, Makefile y otros.

3. Configurando OMNeT++

Ejecute el archivo mingwenv.cmd en la carpeta omnetpp-4.1 haciendo doble clic sobre él en el Explorador de Windows. El programa abrirá una consola con el MSYS *bash* shell, en el cual ya se ha configurado directorio raíz la carpeta omnetpp-4.1/, al ejecutarlo se le presentará una pantalla como se muestra a continuación (Figura 1.1.):

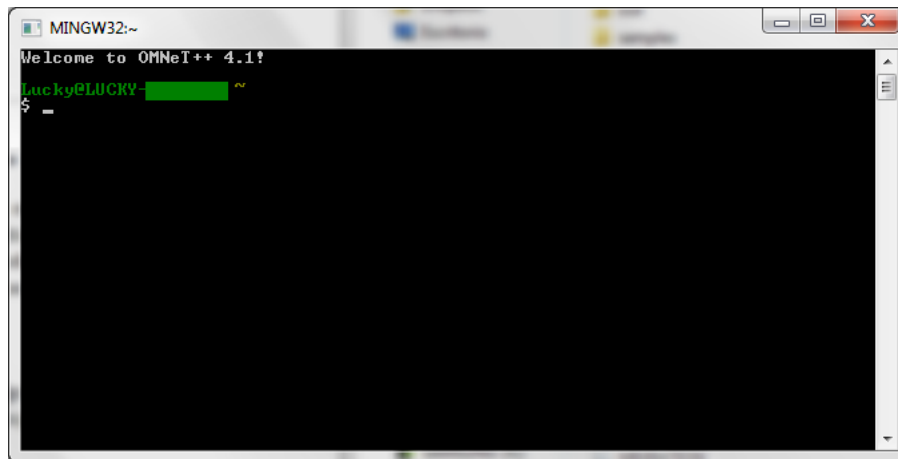


Figura 1.1 - "Consola de OMNeT++"

Primero que todo hay que chequear el contenido del archivo configure.user para asegurarse que contiene la configuración necesaria, en la mayoría de los casos no es necesario cambiar nada, para abrirlo escriba en la consola el siguiente comando:

```
$ notepad configure.user
```

Para compilar y generar los archivos de instalación de OMNeT++, escriba luego:

```
$ ./configure  
$ make
```

El proceso debiera generar todos los archivos necesarios para la utilización del programa.

4. Verificando la Instalación

OMNeT++ incluye en su instalación una carpeta con ejemplos, si usted desea puede probar todos y debieran funcionar correctamente. Como ejemplo de inicio el ejemplo denominado *dyna* puede iniciarse mediante la siguiente secuencia de comandos

```
$ cd samples/dyna  
$ ./dyna
```

Por defecto, los ejemplos funcionarán utilizando el entorno gráfico Tkenv, (el cual se detallará más tarde), del cual tras ejecutar la secuencia anterior se debería visualizar una interfaz gráfica de usuario como se presenta a continuación (Figura 1.2):

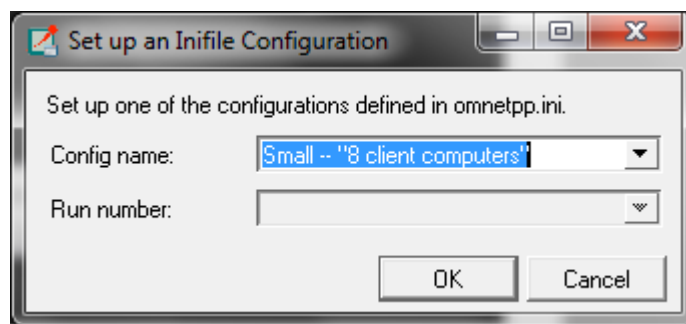


Figura 1.2 – “Diálogo inicial de Dyna en Tkenv”

Seleccione una de las opciones que *Dyna* ofrece, en este caso 8, y pulse el botón OK, debería aparecer la siguiente pantalla (Figura 1.3):

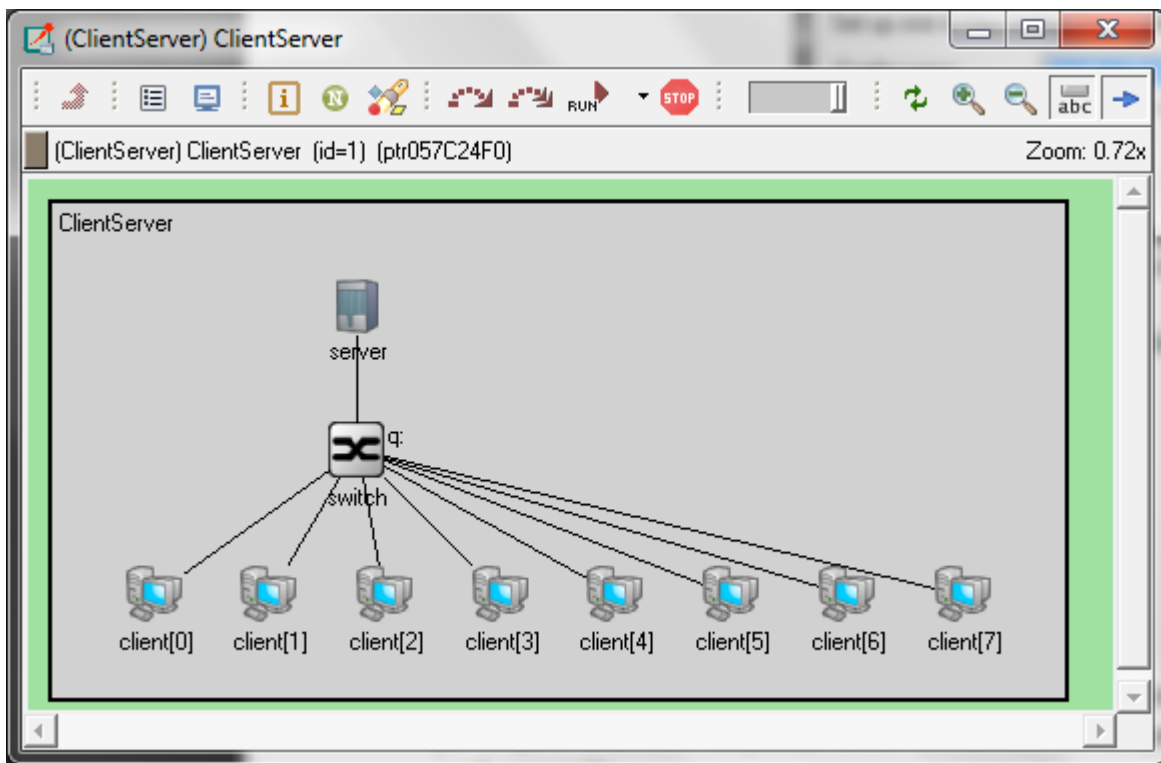


Figura 1.3 – “Pantalla de Simulación de Dyna”

Para iniciar la simulación pulse el botón RUN, de este modo la simulación comenzará a moverse. *Dyna* funciona del siguiente modo:

- Una o más estaciones cliente (client[X]) envían una solicitud de conexión (DYNA_CONN_REQ) al switch del servidor.
- Switch envía las solicitudes en cola al servidor.
- Servidor responde a las solicitudes de conexión (DYNA_CONN_ACK).
- Switch discrimina para que cliente es la respuesta y la envía a la estación correspondiente.
- Cliente se conecta y comienza a enviar tramas de datos (DATA[query]).
- Switch envía la trama al servidor.
- Servidor responde (DATA[result]).
- Switch la envía a la estación correspondiente.
- Cliente envía solicitud de desconexión (DYNA_DISC_REQ).
- Switch envía trama al servidor.
- Servidor genera respuesta (DYNA_DISC_ACK).
- Switch la envía al cliente de destino.
- Se desconecta el cliente de destino.

Mientras se realizan estas operaciones el switch recibe nuevas solicitudes de conexión por parte de las estaciones cliente ociosas, y debe resolverlas siguiendo la misma lógica.

Una vez vista la simulación usted puede cerrar la interfaz Tkenv y el programa mostrará el siguiente diálogo (Figura 1.4):

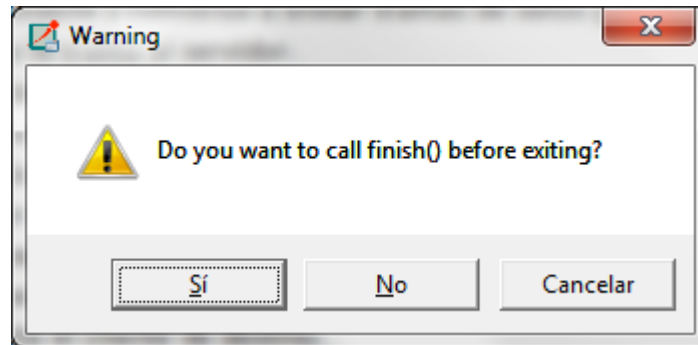


Figura 1.4 – “Diálogo de cierre de Dyna en Tkenv”

Posteriormente puede pasar a cerrar la consola de MINGW32, simplemente tecleando el comando:

```
$ exit
```

Tras esto, pulsando la tecla ENTER la consola se cerrará, terminado el trabajo con OMNeT++.

Puede revisar la demostración del ejemplo *Dyna* siguiendo el siguiente enlace:

<http://www.youtube.com/watch?v=of7UJzpHiKk&hd=1>

II. IDE de OMNeT++

El ejemplo presentado en el capítulo anterior muestra cómo se presenta un programa ya compilado, pero para utilizar el entorno de desarrollo integrado, el procedimiento de ejecución y sus componentes básicos pasarán a describirse a continuación.

1. Ejecución del IDE

Ejecute el archivo mingwenv.cmd en la carpeta omnetpp-4.1 haciendo doble clic sobre él en el Explorador de Windows, se abrirá la consola MINGW32, en ella teclee el comando que se presenta a continuación y pulse la tecla ENTER.

```
$ omnetpp
```

Mediante la ejecución de este comando, aparecerá la pantalla de bienvenida de OMNeT++ (Figura 2.1).



Figura 2.1 – “Pantalla de Bienvenida de OMNeT++ 4”

Posteriormente el programa solicitará que seleccione una carpeta como espacio de trabajo (Figura 2.2), se recomienda seleccionar una carpeta dentro de la instalación del programa para ejecutar los posteriores trabajos realizados con OMNeT directamente desde la consola.

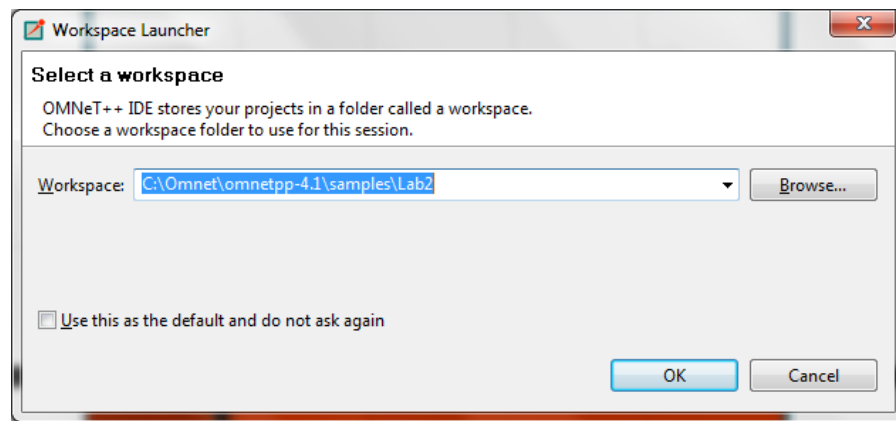


Figura 2.2 – “Selección del Espacio de Trabajo del IDE de OMNeT”

Una vez realizada la operación el IDE de OMNeT se volverá visible para el usuario, esta herramienta ha sido desarrollada en base a Eclipse por lo que para usuarios familiarizados con el programa no debiera resultar difícil el manejo de esta.

IMPORTANTE: Durante la ejecución del IDE, no cierre el terminal MINGW32 o el IDE se cerrará automáticamente y se perderán los datos almacenados en él.

2. Componentes del IDE

Como se señaló anteriormente, el IDE de OMNeT (Figura 2.3), está basado en Eclipse por lo que el manejo de los archivos de código C o C++, funcionan de la misma manera que para otros entornos de desarrollo integrado, sin embargo para simular el tráfico de redes, el IDE de OMNeT introduce otras herramientas, que pasarán a ser detallados a continuación.

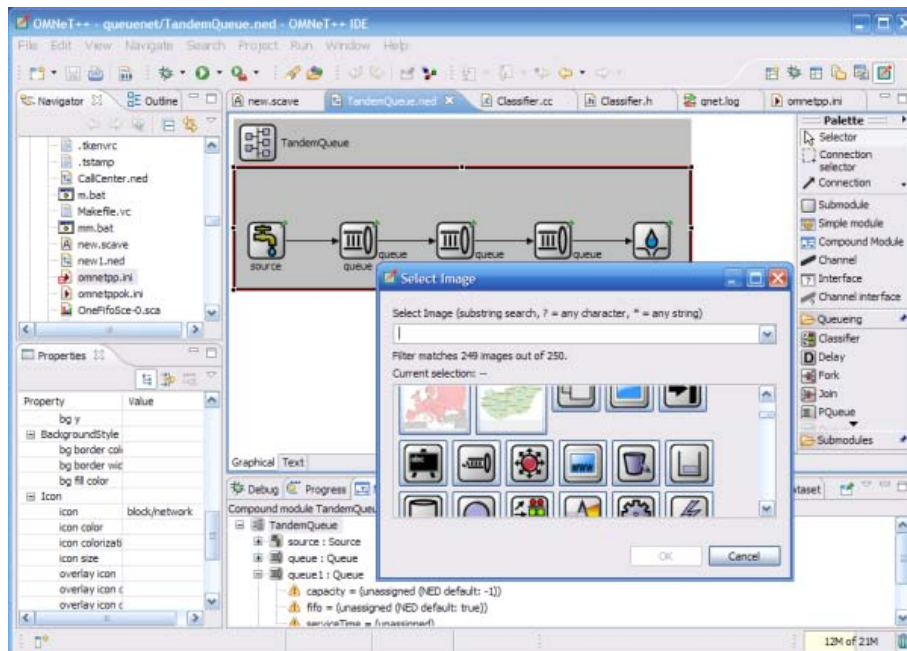


Figura 2.3 – “IDE de OMNeT++”

Editor .NED

Los archivos .NED (Para más información sobre los archivos .NED consulte el manual de OMNeT o el apartado de uso que se presenta más abajo) son archivos en los cuales el usuario puede describir de manera gráfica o escrita la estructura del modelo a simular, el IDE de OMNeT permite editar los archivos de ambas formas y presenta la información mostrando los componentes que conforman la red, permitiendo personalizarlos o editarlos (Figura 2.4).

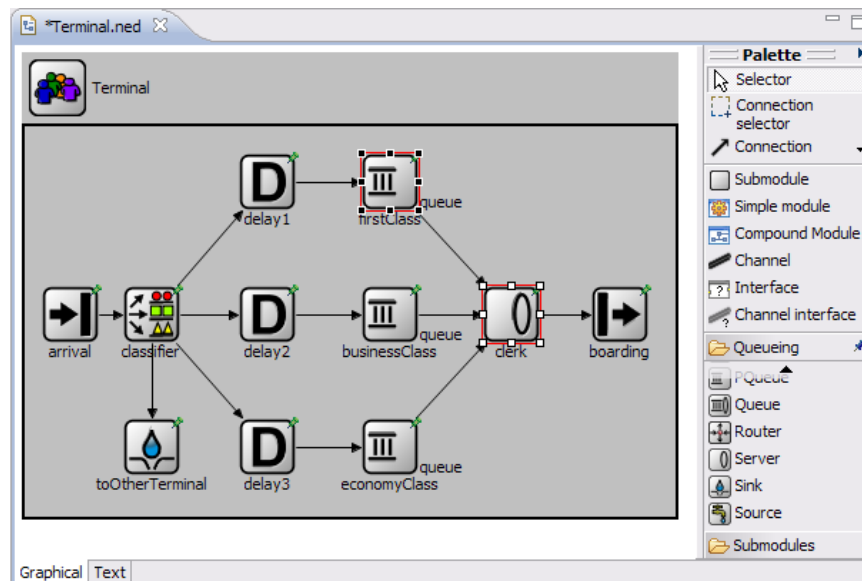


Figura 2.4 - "Editor de archivos .NED"

El uso de este editor es bastante intuitivo, en el panel de la derecha se presentan los elementos que se pueden ir añadiendo a la red, en el panel central se establecen las conexiones entre los componentes y finalmente en la parte inferior se puede cambiar desde el modo gráfico al modo de trabajo en texto (Figura 2.5).

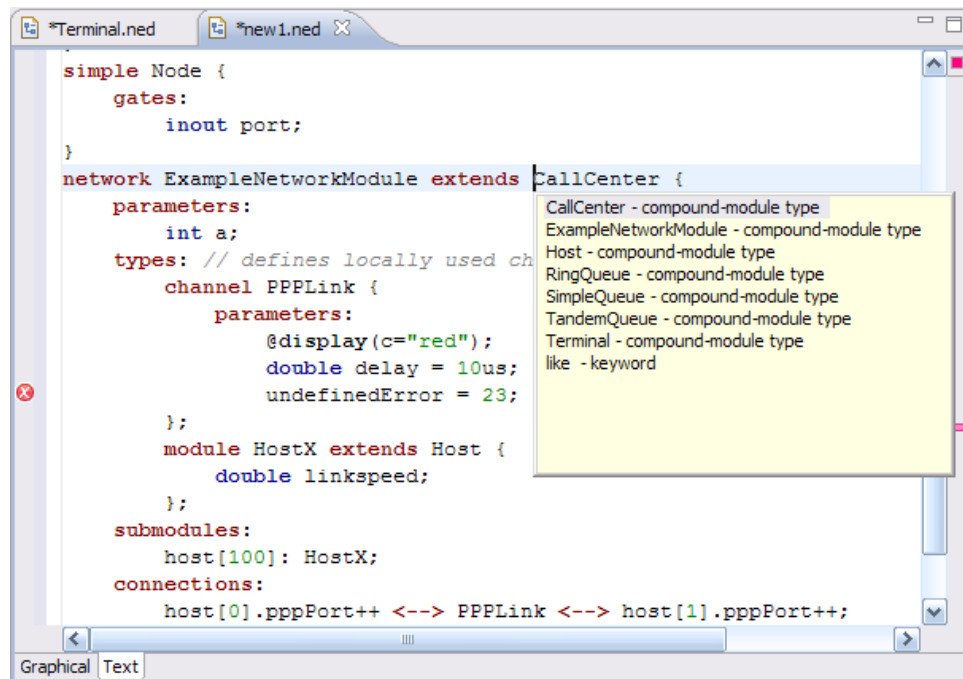


Figura 2.5 – "Editor de archivos .NED en modo texto"

En el modo texto se pueden establecer de forma manual los nodos, la red y los submódulos que la componen, así como las puertas de entrada que representan los puertos de las estaciones conectadas a la red, esta interfaz permite además configurar los parámetros de la red en general y permite la herencia de tipos entre elementos.

Esta interfaz permite dividir la red de forma modular, e ir añadiendo módulos en todas las direcciones posibles, es decir, a partir de una red simple, se pueden añadir gráficamente más nodos, canales e incluso otras redes, por otro lado, estos módulos pueden utilizar otros archivos .NED, y con ello se puede simular la arquitectura de capas de una estación de la red o un conjunto de redes conectados en una red mayor.

Editor .INI

Para configurar la ejecución de los proyectos realizados para OMNeT, se generan archivos .INI los cuales tienen como objetivo establecer los parámetros mediante los cuales se

ejecutará el programa, para configurar estos archivos, que usualmente son de texto plano, OMNeT ofrece un editor especializado que permite configurar todos los parámetros necesarios para la ejecución (Figura 2.6).

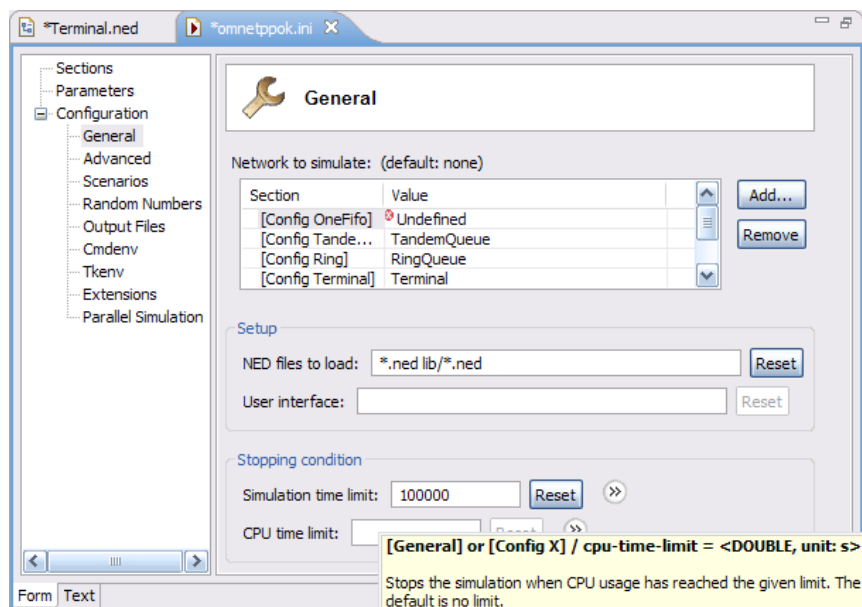


Figura 2.6 – “Editor de Archivos .INI”

La interfaz de configuración de archivos .INI ofrece todos los parámetros por defecto recomendados por los desarrolladores de OMNeT, además proveen de la posibilidad de modificar todos los parámetros posibles y considerar todas las declaraciones posibles en el lenguaje .NED, por último el editor provee una herramienta para la conversión desde archivos .INI de OMNeT 3.X en adelante.

Información adicional

El IDE de OMNeT, en la pantalla principal en la parte inferior provee al usuario de distintas pestañas de configuración e información que permiten revisar diversos aspectos del proyecto que se está desarrollando.

En primer lugar se ofrece la pestaña de herencia de módulos (Figura 2.7), la cual, muestra los componentes que conforman el programa y los organiza en su respectiva posición dentro del esquema de módulos, este módulo es de gran ayuda para visualizar que módulos pueden tener problemas, parámetros sin configurar o estar mal referenciados pues añade las advertencias necesarias para informar al usuario de estas situaciones.

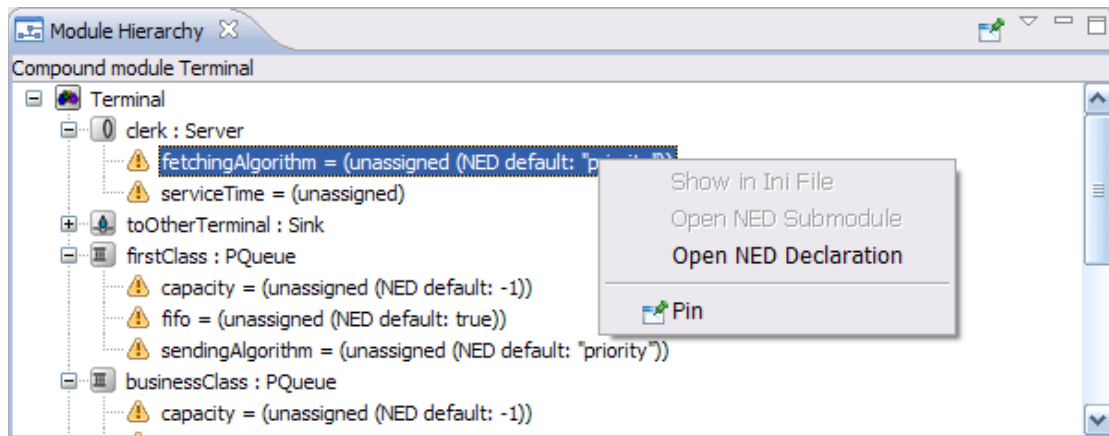


Figura 2.7 – “Pestaña de Herencia de Módulos”

Para configurar parámetros sin asignar el programa ofrece la pestaña de parámetros NED (Figura 2.8), el cual permite añadir los valores necesarios para la ejecución del programa, además de sugerir los valores y tipos por defecto para cada parámetro.

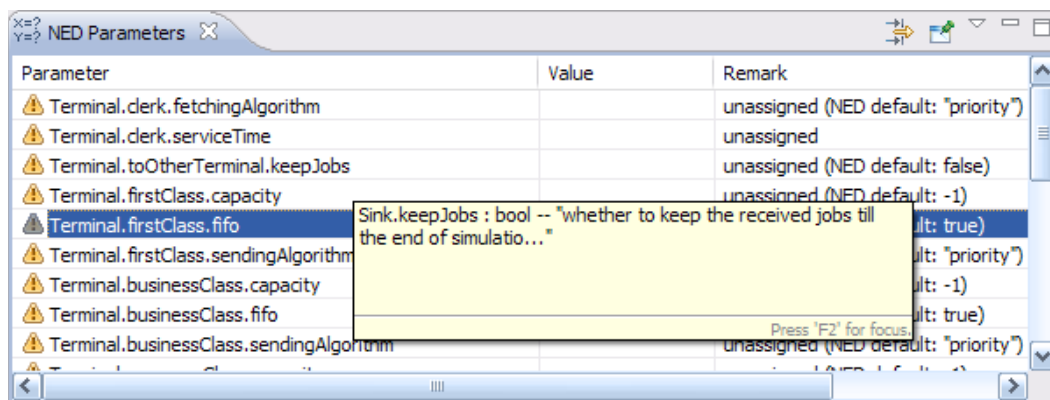
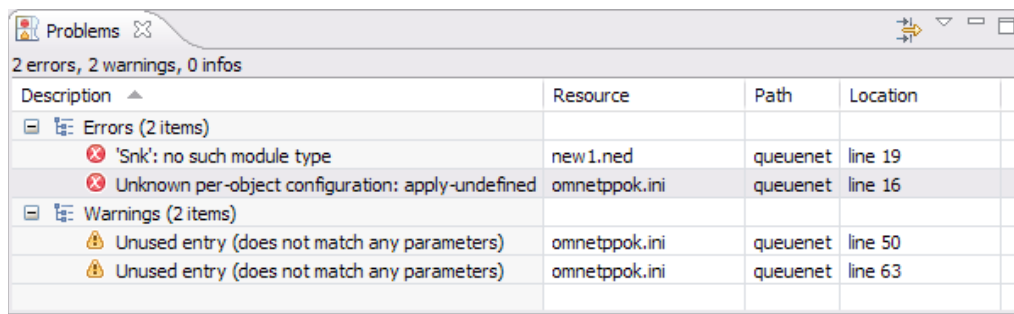


Figura 2.8 – “Pestaña de Parámetros NED”

Para visualizar errores en la compilación del programa el IDE ofrece la pestaña de Problemas (Figura 2.9), la cual muestra los errores y advertencias que se observan en el código del programa, esta información se ofrece del modo usual en que lo hacen estas herramientas, presentando el error, el archivo de ubicación, la ruta del archivo y la ubicación exacta del error dentro del archivo.



The screenshot shows the 'Problems' window in OMNeT++ with the following content:

2 errors, 2 warnings, 0 infos

Description	Resource	Path	Location
Errors (2 items)			
✗ 'Snk': no such module type	new1.ned	queuenet	line 19
✗ Unknown per-object configuration: apply-undefined	omnetppok.ini	queuenet	line 16
Warnings (2 items)			
⚠ Unused entry (does not match any parameters)	omnetppok.ini	queuenet	line 50
⚠ Unused entry (does not match any parameters)	omnetppok.ini	queuenet	line 63

Figura 2.9 – “Pestaña Problemas”

III. Entorno de Ejecución Gráfico Tkenv

Como se presentó anteriormente en el ejemplo de ejecución *Dyna*, OMNeT cuenta con un entorno gráfico de ejecución conocido como Tkenv, el cual permite visualizar el comportamiento de la red generada en tiempo real, entre sus propiedades más importantes se puede destacar:

- Flujo animado de mensajes
- Mensajes programados que se muestran a medida que la simulación avanza
- Distintas ventanas para cada módulo
- Pausar y modificar la velocidad de la simulación.

1. Iniciando Tkenv

Al ejecutar un proyecto de OMNeT, Tkenv automáticamente abrirá al ejecutar este en un computador con la herramienta instalada, en el entorno de Windows el programa no puede ser abierto mediante el uso de doble click por lo que existen dos formas de iniciar el simulador Tkenv:

1. En el IDE de OMNeT, una vez corregidos los errores del proyecto seleccionar el botón RUN, con el ícono triagular en la barra superior del programa (Figura 3.1).

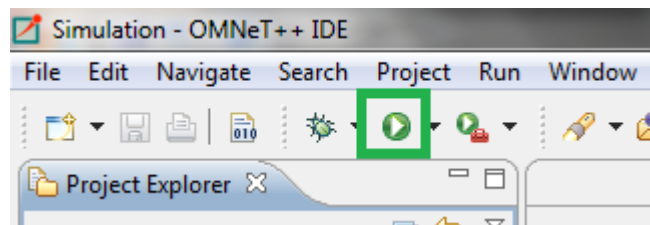


Figura 3.1 – “Botón RUN”

2. Tras generar el ejecutable, (Desde el IDE de OMNeT puede generarse automáticamente en el menú “Project -> Build Project”), salir del IDE, ubicarse en la consola MING32, utilizar el comando `cd` hasta ubicarse en la carpeta del proyecto y teclear el comando que se presenta a continuación.

\$./Nombre_del_Proyecto

De existir un error el programa informará al usuario del error que se produjo, de no encontrar errores, el programa abrirá la simulación ejecutada y presentará las pantallas correspondientes al programa.

2. Componentes de Tkenv

Tras abrir Tkenv el programa presentará el diálogo en el cual se piden los parámetros que el usuario puede ingresar para la simulación (Figura 3.2), tras ejecutar eso se presentan las ventanas de Tkenv que se componen en la ventana principal de eventos y las ventanas de módulos de Tkenv.

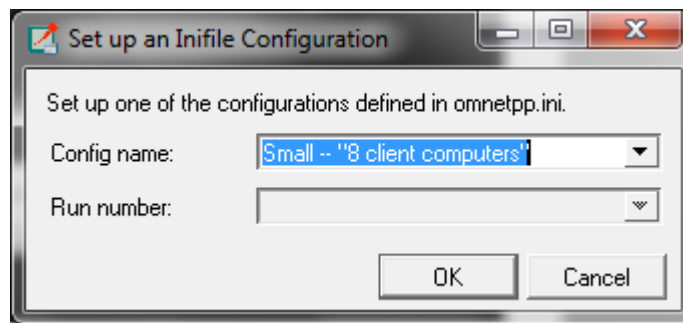


Figura 3.2 – “Diálogo inicial de Tkenv”

Ventana Principal

La ventana principal de Tkenv (Figura 3.3) se caracteriza por no presentar el esquema de módulos, sino que se organiza de forma tal que se puede supervisar el tráfico de la red visualizando los mensajes que se muestran al usuario.

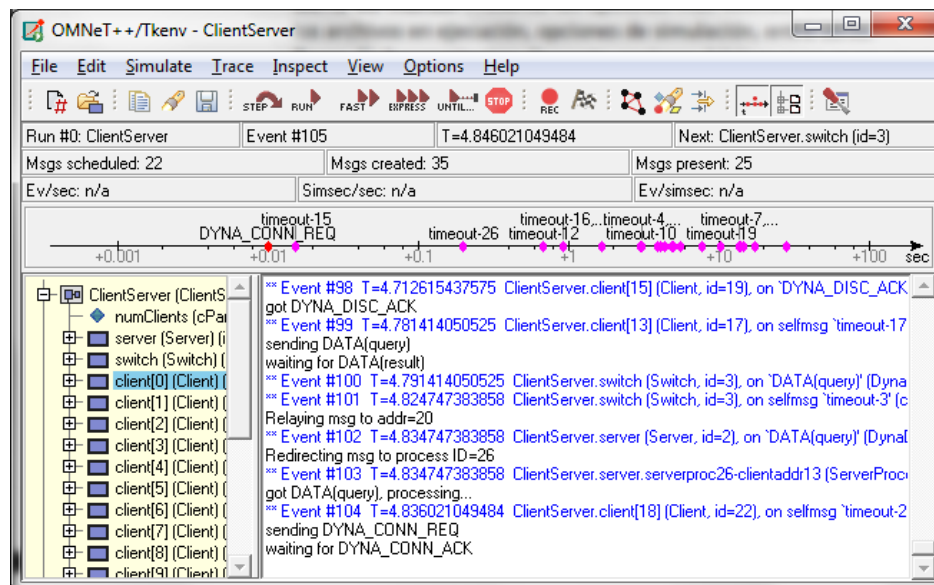


Figura 3.3 – “Ventana Principal de Tkenv”

La ventana se compone de varios componentes que pasan a ser descritos a continuación:

- **Barra de menús:** Presenta las opciones más básicas, tales como cerrar, revisar los archivos en ejecución, opciones de simulación, entre otros.
- **Barra de herramientas:** Presenta opciones básicas para programas de este tipo tales como guardar, copiar, entre otros, pero además presenta los controles de la simulación, desde esta barra el usuario puede ajustar el tipo de simulación, iniciar y pausar la simulación, terminar con esta e incluso grabarla.
- **Barra de estado:** Ubicado justo debajo de la barra de herramientas, presenta las estadísticas que presenta la red, tales como, mensajes creados, mensajes programados, evento en ejecución, eventos por segundo entre otros.
- **Línea temporal:** Presenta los eventos que está por ocurrir en una escala temporal logarítmica, realizando doble click sobre un punto se puede ver la información del evento asociado
- **Árbol de objetos:** Presenta los objetos que actualmente están cargados en el programa, cliqueando en uno se abre el visor del objeto indicando el estado de las variables para ese objeto en ese momento.
- **Área de registro:** Presenta el registro histórico de los eventos que han ocurrido a lo largo de la simulación, también el usuario puede configurar mensajes para que el programa los entregue tras el desarrollo de determinados eventos.

Ventana de Módulos

En condiciones normales Tkenv al iniciarse debe abrir por lo menos dos ventanas, la ventana principal, presentada anteriormente, y la ventana de módulo principal, la cual presenta la red con los módulos que la componen, (Figura 3.4), desde esta ventana se puede iniciar la simulación y además entrega la posibilidad de analizar el comportamiento de los módulos en detalle.

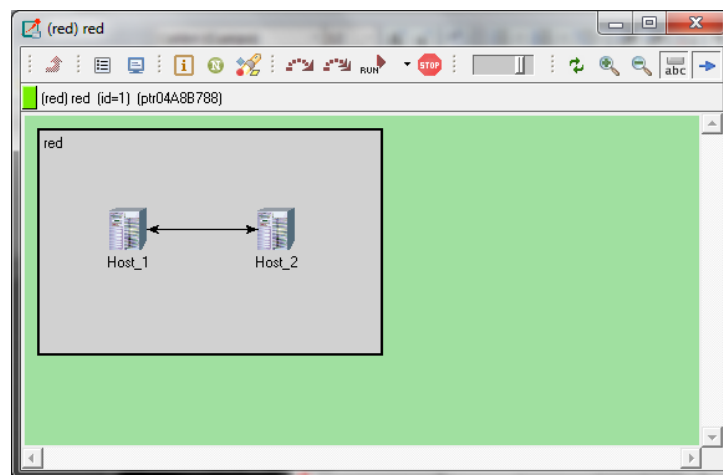


Figura 3.4 – “Ventana Principal de Módulos de Tkenv”

Como se puede observar la ventana principal muestra el diagrama básico de la red con la que se trabaja (Para ver un diagrama más complejo consultar Figura 1.3), y en la parte superior la barra de herramientas similar a la de la ventana principal con algunas herramientas adicionales, tales como el selector de velocidad (ubicado a la derecha del botón STOP), y la posibilidad de realizar zoom en caso de querer ver partes en detalle de redes más complejas, para analizar la estructura de algún módulo en particular tkenv muestra el módulo en una ventana nueva (Figura 3.5) tras realizar doble click sobre el módulo deseado.

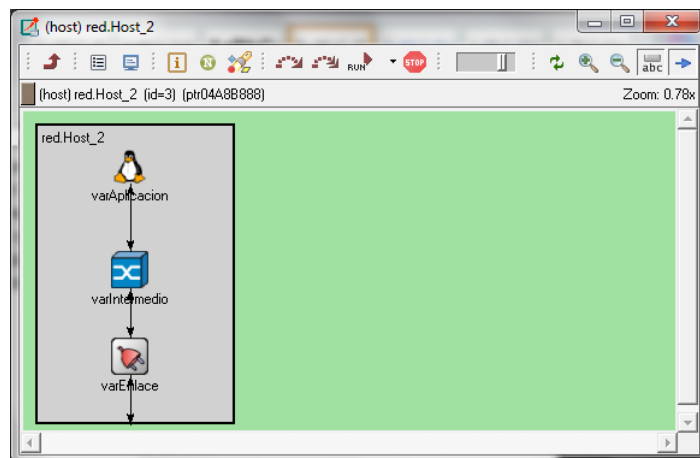


Figura 3.5 – “Ventana de módulo secundario de Tkenv”

La ventana de módulo secundario al ejecutar la simulación también mostrará la respuesta que este genera y como los mensajes recibidos actúan sobre este, se puede seguir avanzando con esta acción hasta, finalmente, llegar a los módulos más básicos de la red, en los cuales solo se muestra las variables que actúan y sus valores (Figura 3.6).

Class	Name	Info
cPar	tamTrama	4
cPar	direccion	2
cGate	desde_abajo	<- varIntermedio.hacia_arriba
cGate	hacia_abajo	-> varIntermedio.desde_arriba

Figura 3.6 – “Ventana de módulo básico de Tkenv”

IV. Ejemplo de Programa

Anteriormente se han presentado los aspectos de ejecución de OMNeT++, el presente capítulo tiene por objetivo presentar un ejemplo en el cual se explica de forma práctica el funcionamiento del simulador y como este pueden ser usado para emular situaciones de tráfico de telecomunicaciones y redes real.

Se presentará detalladamente un ejemplo de una simulación utilizando OMNeT en su desarrollo que simula una red simple con un solo canal de comunicación, y dos estaciones comunicándose entre ellas (Figura 4.1).

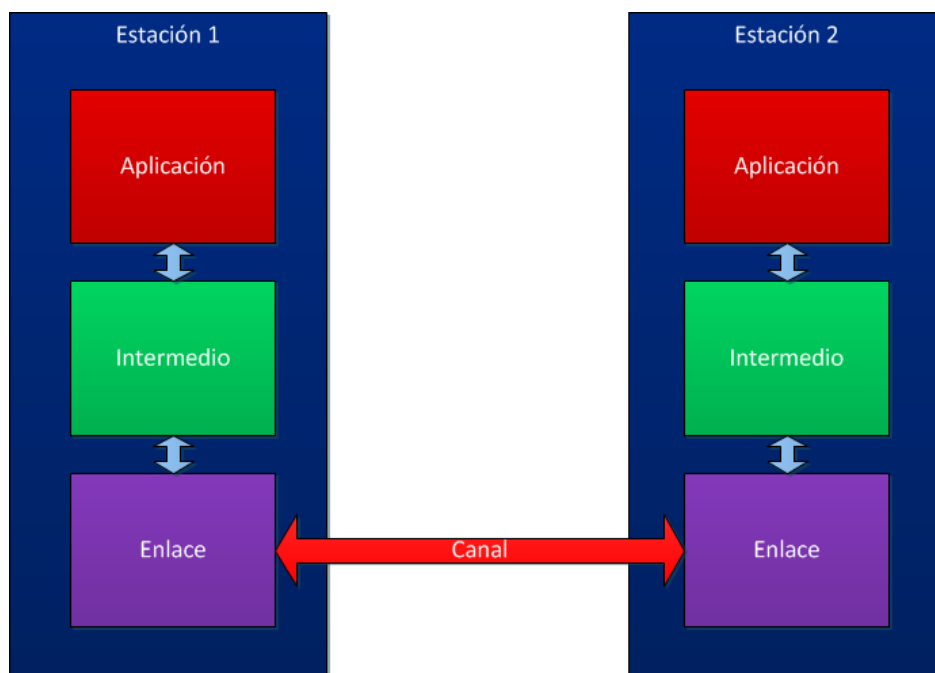


Figura 4.1 – “Esquema de comunicación utilizado en el programa”

Como se observa en la figura anterior, las estaciones dentro poseen 3 capas o niveles, cada una con un propósito específico, las tres capas aquí presentes emulan la arquitectura de capas de los sistemas de redes en los ordenadores, pues tanto el estándar OSI, como el conjunto de protocolos de red TCP/IP trabajan en base a niveles o capas que prestan y consumen servicios de las otras capas.

En el ejemplo se generaron todas las capas y el sistema en general, por lo tanto, se presentan a continuación los archivos fuente generados para ello.

1. Archivos .NED

En primer lugar, se debe genera el esquema de red con el que se trabajará, es decir, las capas, en este ejemplo en particular se ha ido desarrollando desde los módulos más pequeños hacia fuera, es decir, se finalizará mostrando el archivo .NED de la red.

Por lo tanto a continuación se presentan y explican los archivos generados

Aplicación .NED

El archivo aplicacion .NED, define la mecánica de la capa de nivel superior, es decir, aplicación, consta de un módulo simple que se presenta a continuación (Figura 4.2).

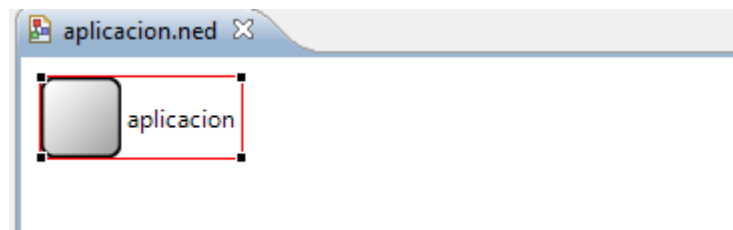


Figura 4.2 – “Módulo Aplicación .NED”

El módulo simplemente está compuesto por una caja, sin embargo la definición de sus parámetros indica como está organizado internamente el módulo, su definición interna se presenta a continuación (Figura 4.3).

```
simple aplicacion
//Parámetros de la aplicación
parameters:
    //Tamaño de la trama
    tamTrama : numeric const,
    //Dirección de Destino
    direccion : numeric const;
//Conexiones permitidas
gates:
    //Recepción de datos desde abajo
    in: desde_abajo;
    //Envío de datos desde abajo
    out: hacia_abajo;
endsimple
```

Figura 4.3 – “Código fuente del módulo aplicación”

Puede apreciarse que la sintaxis del código de .NED es bastante simple e intuitiva, por otro lado el módulo es simple, consta solo de dos parámetros y dos puertas de conexión y que ambas son para comunicarse con el nivel inferior, intermedio, que se presenta a continuación.

Intermedio .NED

El archivo intermedio.NED al igual que el anterior consta solo de un módulo simple, es decir, una caja, con parámetros, al ser su presentación gráfica idéntica a la del nivel anterior (pues los enlaces entre estas se definen posteriormente), solo se presentará el código fuente de ella (Figura 4.4).

```
simple intermedio
//Conexiones
gates:
    //Hacia el nivel superior
    in: desde_arriba;
    out: hacia_arriba;
    //Hacia el nivel inferior
    in: desde_abajo;
    out: hacia_abajo;
endsimple
```

Figura 4.4 – “Código fuente del módulo intermedio”

Su estructura es parecida al nivel anterior con las salvedades de que este nivel posee cuatro puertas de enlace, dos de comunicación con el nivel superior, para enviar y recibir, y dos con el nivel inferior para el mismo propósito, además como este modulo no espera alojar datos para ningún procedimiento en particular no necesita almacenar los datos de la trama, solo transmitirla. A continuación se analizará el nivel inferior, de enlace.

Enlace .NED

El nivel de enlace tiene como objetivo enviar fuera del host hacia el canal de transmisión, la trama de datos que se desea enviar, por lo tanto, este nivel consta de un módulo simple, que por ser gráficamente igual a los anteriores no se presentará compuesto de varias puertas de comunicación y espacio para la recepción de datos, el código fuente del módulo de enlace se presenta a continuación (Figura 4.5).

```
simple enlace
//Parámetros del nivel de Enlace
parameters:
    //Dirección de destino
    direccion: numeric const,
    //Tamaño de trama
    tamTrama: numeric const;
//Conexiones
gates:
    //Con el nivel inferior
    in: desde_fisico;
    out: hacia_fisico;
    //con el nivel superior
    in: desde_arriba;
    out: hacia_arriba;
endsimple
```

Figura 4.5 – “Código fuente del módulo de enlace”

Como se aprecia, el módulo presenta similitudes con los módulos presentados anteriormente, entre ellos el espacio para almacenar los valores de dirección y el tamaño de la trama, además de las cuatro puertas de enlace para satisfacer la conexión hacia los módulos superiores y el canal de transmisión, a continuación se analizará la estructura del Host que engloba estas tres estructuras para conformar la estación que transmite y recibe mensajes.

Host .NED

El archivo host.NED establece las conexiones entre las distintas capas de este, y realiza el llamado a los módulos para incluirlos en un módulo mayor, aquí los módulos finalmente se vuelven personalizables, y se pueden visualizar las conexiones que los comunican (Figura 4.6).

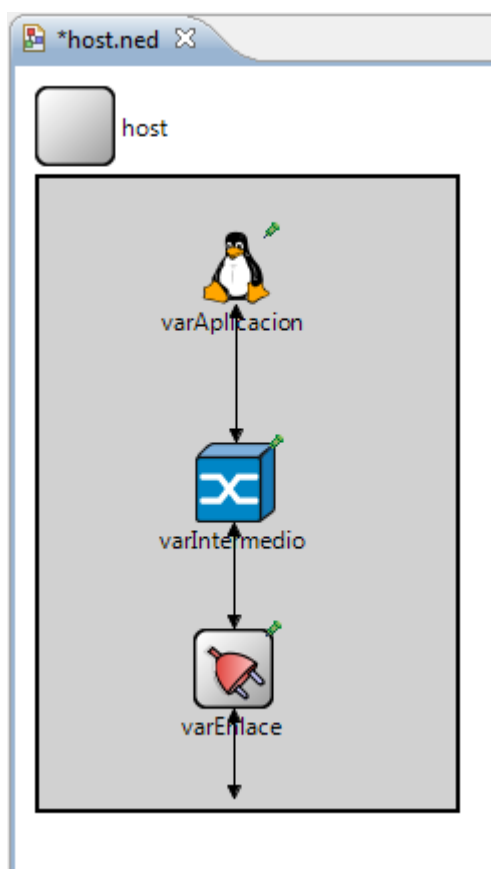


Figura 4.6 – “Representación gráfica de Host.NED”

Además, en este nivel el código fuente empieza a verse más completo, ya no solo representando los parámetros y las puertas de enlace del módulo, sino que se hacen los llamados a los submódulos que componen el módulo, y se establecen las conexiones que comunicarán a estos (Figura 4.7).


```

//Importación de niveles del host
import
    "enlace",
    "intermedio",
    "aplicacion";

//Definición del Host
module host
    //parámetros
    parameters:
        //dirección del host
        direccion: numeric const,
        //tamaño de trama
        tamTrama: numeric const;
    //Puertas de enlace
    gates:
        //puerta de entrada
        in: entrada;
        //puerta de salida
        out: salida;
    //Llamado a Submódulos
    submodules:
        //Llamado al nivel de aplicación
        varAplicacion: aplicacion;
        //parámetros del nivel
        parameters:
            tamTrama = tamTrama,
            direccion = direccion;
        //icono del nivel
        display: "p=98,44;i=abstract/penguin";

        //Llamado al nivel intermedio
        varIntermedio: intermedio;
        //Icono del nivel intermedio
        display: "p=98,152;i=abstract/switch";

        //Llamado al nivel de enlace
        varEnlace: enlace;
        //parámetros del nivel
        parameters:
            tamTrama = tamTrama,
            direccion = direccion;
        //Iconos del nivel
        display: "p=97,245;i=block/plug";
    //Conexiones del host
    connections:
        //Desde aplicacion hacia intermedio
        varAplicacion.hacia_abajo --> varIntermedio.desde_arriba;
        //Desde intermedio a aplicación
        varIntermedio.hacia_arriba --> varAplicacion.desde_abajo;
        //Desde intermedio a enlace
        varIntermedio.hacia_abajo --> varEnlace.desde_arriba;
        //Desde enlace a intermedio

```

```

varEnlace.hacia_arriba --> varIntermedio.desde_abajo;
//Desde enlace a la salida del host
varEnlace.hacia_fisico --> salida;
//Desde la salida del host hacia enlace
entrada --> varEnlace.desde_fisico;
//Parámetros visuales del host
display: "o=,,;b=208,315,,,,";
//Fin de la declaración
endmodule

```

Figura 4.7 – “Código fuente de host.NED”

Sistema .NED

Una vez definido el comportamiento del módulo Host se pasa a la construcción de la red en sí, en el archivo Sistema .NED, en el cual se define la forma que tendrá la red, los módulos participantes y las conexiones que se generan entre los distintos componentes de la red, de este modo queda definida la estructura física de la red a simular y se presenta gráficamente en este archivo (Figura 4.8).

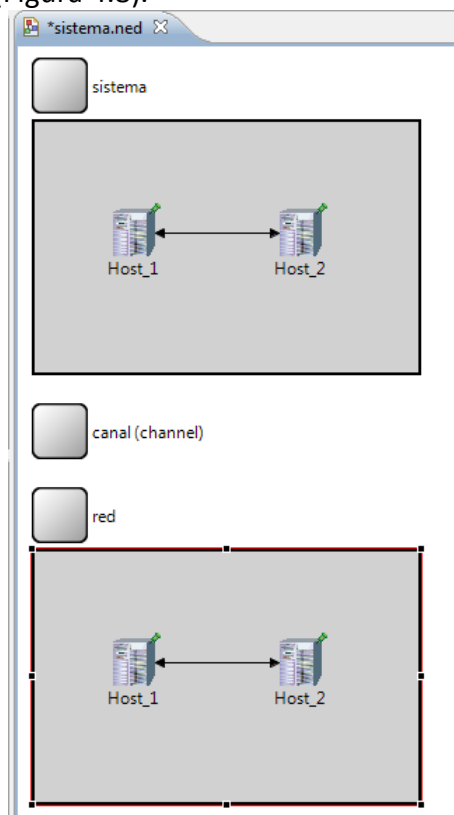


Figura 4.8 – “Representación gráfica de Sistema.NED”

Como se puede apreciar, en el esquema son visibles dos esquemas de red idénticos, uno corresponde al sistema y el otro corresponde a la instanciación de la red, es decir, como se comportará la red en tiempo de ejecución, por otro lado, para evitar el exceso de llamados se

define el canal de comunicación aquí, que como es una clase del tipo canal, para generar una simulación acorde con la realidad se le pueden configurar diversos parámetros de comunicación tales como la tasa de transmisión y el retardo de propagación que se produce en un canal de comunicación en unidades reales, todas estas declaraciones pueden apreciarse en código fuente del sistema que se presenta a continuación (Figura 4.9).

```
//Importación del host
import
    "host";

//Definición del módulo
module sistema
    //Parámetros
    parameters:
        tamTrama: numeric;
    //Submódulos que lo componen
    submodules:
        //Definición del host 1
        Host_1: host;
        //Parámetros
        parameters:
            tamTrama = tamTrama,
            //Se suministra la dirección que le corresponde al host
            direccion = 1;
            //Aspecto gráfico del host
            display: "p=71,79;i=old/server1";
        //Definición del host 2
        Host_2: host;
        //Parámetros
        parameters:
            tamTrama = tamTrama,
            //Se suministra la dirección que le corresponde al host
            direccion = 2;
            //Aspecto gráfico del host
            display: "p=190,79;i=old/server1";
    //Conexiones de la red
    connections:
        //Desde el Host 1 hacia el Host 2
        Host_1.salida --> canal --> Host_2.entrada;
        //Desde el Host 2 hacia el Host 1
        Host_2.salida --> canal --> Host_1.entrada;
    //Aspecto gráfico de la red
    display: "o=,,;b=274,179,yellow,,, ";
//Fin del módulo
endmodule

//Definición del canal de transmisión
channel canal
    //Retardo de propagación
    delay 100ms;
    //Velocidad de datos
    datarate 128000bps;
```

```

endchannel

//Instanciación de la red
network red : sistema
    //Parámetros
    parameters:
        tamTrama = input(4, "Ingrese el largo de la palabra, entero mayor a
cero:");
//Fin de la definición de la red
endnetwork

```

Figura 4.9 – “Código fuente de Sistema .NED”

2. Archivos .CC

Tras haber definido el esquema de la red se deben definir el comportamiento de esta, para ello OMNeT++ soporta archivos .CC, es decir, se pueden definir funciones y procedimientos en el lenguaje de programación C++, y hacer uso de las librerías que OMNeT++ ofrece para comunicarse con el resto del sistema, para ello solo basta definir el comportamiento de los niveles inferiores (aplicación, intermedio, enlace) en su respectivo archivo y se tendrá el definido el comportamiento de la red. A continuación se presentan estos archivos y se describen a grandes rasgos.

Aplicación .CC

El archivo aplicación indica los procedimientos que realizará la capa de aplicación a la hora de simular, al ser la capa de aplicación el nivel superior del programa, este debe generar los mensajes hacia el otro host, enviarlos al nivel inferior, obtener la información entrante desde el nivel intermedio y realizar algo con ella, a continuación se presenta el código fuente del nivel de aplicación debidamente comentado (Figura 4.10).

```

/*
 * En éste módulo se generan palabras de información, la cual viaja
 * a los modulos inferiores para ser enviada al otro Host a través del canal
 */

#include <string.h>
//librería de OMNeT++
#include <omnetpp.h>
#include <cstdlib>
#include <iostream>

using namespace std;

//Nombre de la clase y tipo de la librería omnetpp.h
class aplicacion : public cSimpleModule
{
//Métodos de la clase

```

```

public:
    //Inicializar módulo
    virtual void initialize();
    //Manejador de mensajes
    virtual void handleMessage(cMessage *msg);
    //Generador de palabras
    virtual void generaPalabraInfo();
};

//Entero que define quien está enviando y quien está recibiendo
int turno=0;

//Definición de las funciones de aplicación
Define_Module(aplicacion);

//Inicializar
void aplicacion::initialize()
{
    //Si es mi turno
    if(turno==0)
    {
        //Generar palabra para envío
        generaPalabraInfo();
        //Entregar el turno
        turno=1;
    }
}

//Manejar mensajes
void aplicacion::handleMessage(cMessage *msg)
{
    //Si el mensaje ha llegado
    if (msg->arrivedOn("desde_abajo"))
    {
        //Borrar el mensaje
        delete msg; //cuando llega un mensaje solo se descarta
    }
    //Generar un mensaje de respuesta
    generaPalabraInfo();
}

//Generador de palabras
void aplicacion::generaPalabraInfo()
{
    //Se entrega la dirección
    int direccion = par("direccion");
    //Se entrega el tamaño
    int tamT = par("tamTrama");
    //Puntero al mensaje
    char *mens;

    //Reserva de memoria para el mensaje
    mens = (char*)malloc(sizeof(char)*tamT);

```

```

//Se crea el mensaje solo con valores 0
strcpy(mens, "0");
//Se concatenan ceros al mensaje
for(int i=1;i<tamT;i++)
{
    strcat(mens, "0");
}
//Se crea el mensaje
cMessage *palabra = new cMessage(mens);
//Se envia el mensaje por la puerta determinada
send(palabra, "hacia_abajo");
//Se informa al usuario de que se envió la palabra
ev<<"Host "<<direccion<<" - LA PALABRA QUE SE ENVIO DESDE APLICACION ES:
"<<mens;
}

```

Figura 4.10 – “Código Fuente de Aplicacion.CC”

Intermedio.CC

El archivo intermedio .CC, es aquel que controlará el comportamiento de esta capa en ambos host durante el proceso de comunicación, su único objetivo es revisar el mensaje, ver desde donde viene y enviarlo hacia donde debe ir, si viene desde abajo, es decir desde enlace enviarlo hacia la capa de aplicación y viceversa, a continuación se presenta el código fuente de esta capa (Figura 4.11).

```

/*
 * Archivo que controla el nivel intermedio
 */

#include <string.h>
#include <omnetpp.h>

//Nombre de la clase y tipo
class intermedio : public cSimpleModule
{
    //Métodos
protected:
    //Procesador de mensajes desde la capa superior
    virtual void processMsgFromHigherLayer(cMessage *packet);
    //Procesador de mensajes desde la capa inferior
    virtual void processMsgFromLowerLayer(cMessage *packet);
    //Receptor de mensajes
    virtual void handleMessage(cMessage *msg);
};

//Definición de funciones
Define_Module( intermedio );

//Manejador de mensajes
void intermedio::handleMessage(cMessage *msg)
{
    //Si el mensaje viene desde abajo

```

```

    if (msg->arrivedOn("desde_abajo"))
        //Enviar al procesador desde abajo
        processMsgFromLowerLayer(msg);
    //Sino, el mensaje viene desde arriba
    else
        //Procesar el mensaje desde arriba
        processMsgFromHigherLayer(msg);
}

//Procesador desde arriba
void intermedio::processMsgFromHigherLayer(cMessage *packet)
{
    //Envia el paquete hacia la capa de enlace
    send(packet, "hacia_abajo");
}

//Procesador desde abajo
void intermedio::processMsgFromLowerLayer(cMessage *packet)
{
    //Envía el paquete hacia la capa de aplicación
    send(packet, "hacia_arriba");
}

```

Figura 4.11 – “Código fuente de intermedio.CC”

Enlace.CC

Enlace.CC funciona de forma análoga al archivo anterior, pues su tarea es recibir los mensajes, identificar si vienen desde los niveles superiores del mismo Host o desde el otro Host y enviarlo hacia su destino final, el código fuente del nivel de enlace se presenta a continuación (Figura 4.12).

```

/ * Envía las palabras desde la capa intermedia al otro host
 * y recibe las palabras que llegan desde el otro host y las envía al nivel
superior
 */

#include <string.h>
#include <omnetpp.h>

//Nombre y tipo de la clase
class enlace : public cSimpleModule
{
    //Métodos
protected:
    //Procesador de mensaje desde Intermedio
    virtual void processMsgFromHigherLayer(cMessage *dato);
    //Procesador de Mensaje desde el otro host
    virtual void processMsgFromLowerLayer(cMessage *packet);
    virtual void handleMessage(cMessage *msg);
};

```

```

//Definición de los métodos
Define_Module( enlace );

//Manejador de mensajes
void enlace::handleMessage(cMessage *msg)
{
    //Si el mensaje llega desde el otro Host
    if (msg->arrivedOn( "desde_fisico" ))
    {
        //Procesarlo como si viene desde abajo
        processMsgFromLowerLayer(msg);
    }
    //Sino, el mensaje viene desde intermedio
    else
    {
        //Procesarlo como si viene desde arriba
        processMsgFromHigherLayer(msg);
    }
}

//Si la palabra llega desde intermedio enviar al otro host
void enlace::processMsgFromHigherLayer(cMessage *dato)
{
    //Enviar el paquete a través del canal
    send(dato, "hacia_fisico");
}

//Si la palabra llega desde el otro host enviar hacia intermedio
void enlace::processMsgFromLowerLayer(cMessage *packet)
{
    //Enviar el paquete hacia el nivel intermedio
    send(packet, "hacia_arriba");
}

```

Figura 4.12 – “Código fuente de Enlace.CC”

3. Archivo .INI

El archivo .INI presenta la configuración que utilizará la red en tiempo de ejecución, en este archivo se pueden configurar los diversos parámetros de la simulación, tales como el tiempo límite de la simulación, la precisión de tiempos de la simulación, la red a simular, entre otros, como este es un ejemplo simple no requirió una configuración elaborada, únicamente se especificó la red que se simularía y la apariencia final del archivo fue la que se presenta a continuación (Figura 4.13).

```

[General]
network = red

```

Figura 4.13 – “Apariencia de Omnetpp.ini”

V. Ejecutar Ejemplo

Finalmente para ejecutar el programa únicamente se debe ir a la carpeta del proyecto en el IDE de OMNeT++, y posicionarse con el botón derecho sobre ella, para seleccionar la opción “Build Project” (Figura 5.1) y se generará automáticamente el archivo ejecutable con el nombre del proyecto

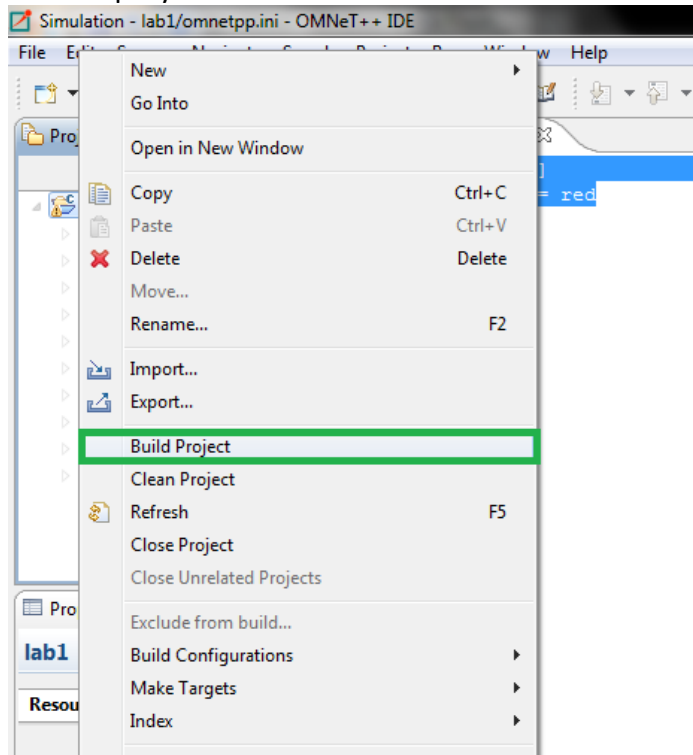


Figura 5.1 – “Comando que genera el proyecto”

Posteriormente se podrá visualizar el archivo .Exe del proyecto en el IDE, y para ejecutarlo pulse el botón de “RUN” dentro del IDE, (Figura 5.2).

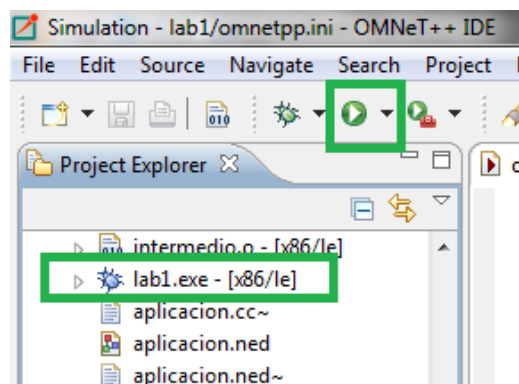


Figura 5.2 – “Ejecutable generado y botón de RUN”

Además puede ejecutar directamente el proyecto en la consola MINGW32 ubicándose en la carpeta del programa, en este ejemplo “samples/lab1” mediante el comando “cd” y posteriormente ejecutando la siguiente línea de comandos:

```
$ ./lab1
```

Donde lo que se añade luego del “./” es el nombre del proyecto.

Tras realizar esta acción Tkenv comienza su ejecución y le presenta las ventanas del programa (Figura 5.3 y 5.4).

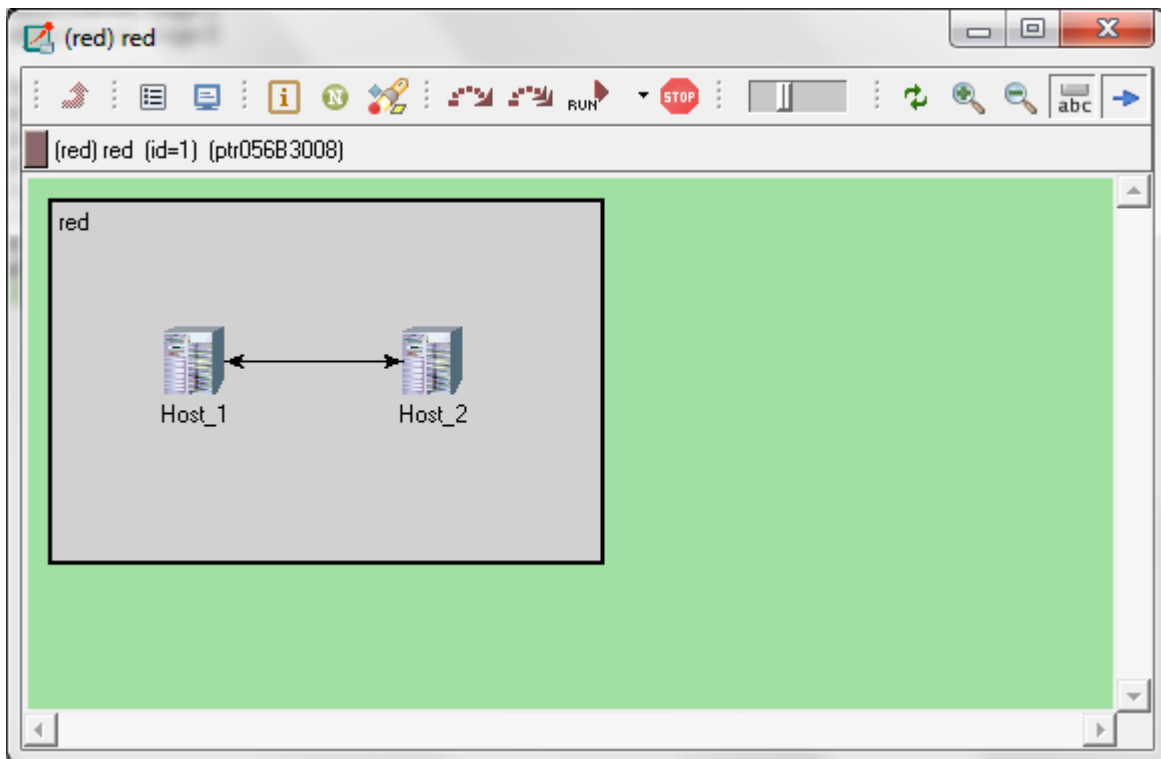


Figura 5.3 – “Pantalla de red de Tkenv”

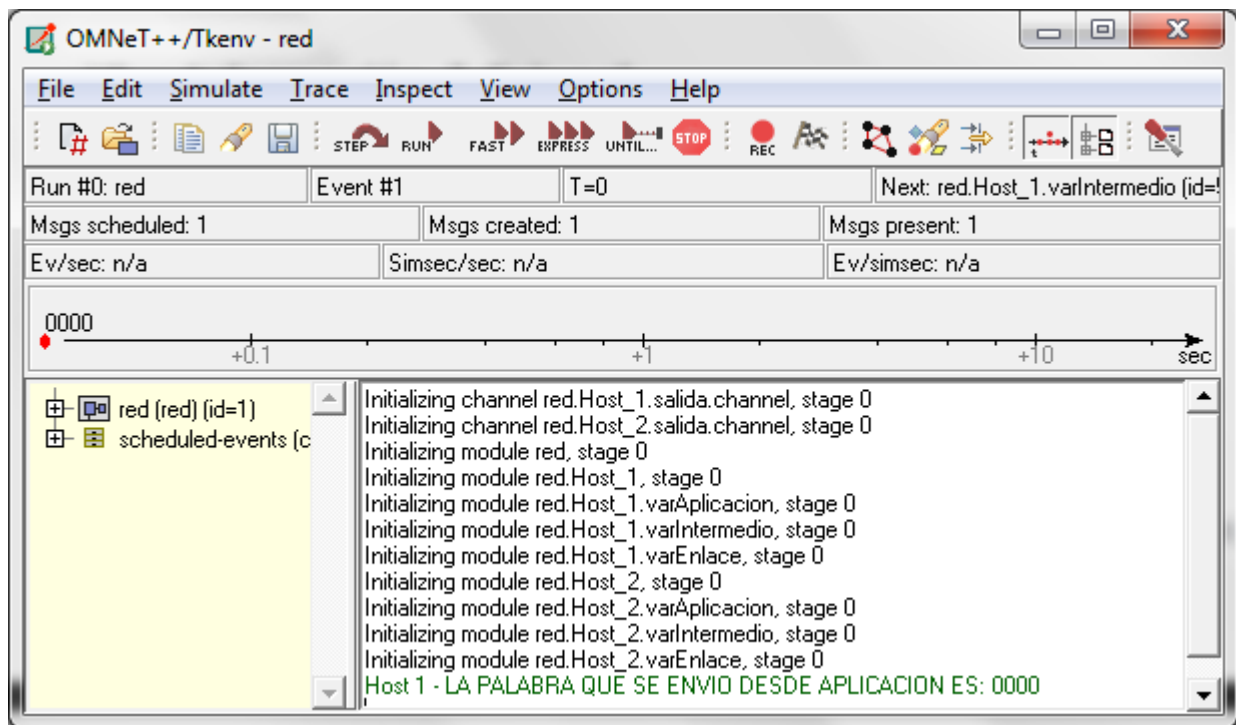


Figura 5.4 – “Pantalla principal de Tkenv”

Puede iniciar la simulación haciendo click en el botón “RUN” ubicado en ambas ventanas, tras esto la simulación comenzará y podrá ver la animación en la ventana de red, si desea visualizar el detalle de los Host, basta con hacer doble click sobre ellos en la pantalla de red, la demostración de la simulación completa puede revisarse en el siguiente enlace.

www.youtube.com/watch?v=Hqlr_TfwTag

Aquí se presenta la demostración completa de la ejecución de este ejemplo.

VI. Información Adicional

Para mayor información revise la documentación en la carpeta docs de Omnet4.1, para más ejemplos de simulaciones en Tkenv, consulte los siguientes enlaces:

- Para el ejemplo de enrutamiento de paquetes a través de una red compleja:
http://www.youtube.com/watch?v=nm1_2fi98Oc
Este ejemplo corresponde a la ejecución del programa “routing” ubicado en la carpeta de ejemplos “samples” de OMNeT
- Para el ejemplo de una red que utilice el protocolo Aloha revise el siguiente enlace:
<http://www.youtube.com/watch?v=eBpQuCYrpHI&hd=1>
Este ejemplo corresponde a la ejecución del programa “aloha” ubicado en la carpeta de ejemplos “samples” de OMNeT.

VII. Créditos

Se agradece sinceramente a Alan Olivares, Alumno de Ingeniería Civil Informática de la Universidad de Santiago de Chile por el desarrollo del código fuente del ejemplo presentado aquí realizado en el año 2008, que se adjunta junto con el presente documento.