Prácticas de apoyo para la Asignatura
Redes de Datos I

Dr. Victor Rangel Licea
Depto. Ing. en Telecomunicaciones
Facultad de Ingeniería
Universidad Nacional Autónoma de México

Octubre de 2009

AGRADECIMIENTOS

ESTE MATERIAL FUE ELABORADO CON EL APOYO DEL PROYECTO **PAPIME** ***PE103807:*** INTEGRACIÓN DE NUEVAS TECNOLOGÍAS Y ELABORACIÓN DE MATERIAL DIDÁCTICO PAR UN LABORATORIO MULTIDISCIPLINARIO DE REDES DE LA FACULTAD DE INGENIERÍA 2007-2010

SE AGRADECE EL APOYO DE LA DGAPA-UNAM POR LOS RECURSOS PROPORCIONADOS PARA LA ELABORACIÓN DE ESTE MATERIAL.

Contenido

Práctica 1

Procedimiento de recuperación de contraseñas

Este documento describe el procedimiento para recuperar las contraseñas enable y enable secret. Estas contraseñas son utilizadas para acceder al modo privilegiado y configuraciones. La contraseña enable puede ser recuperada pero enable secret no debido a que esta encriptada, esta solo puede ser reemplazada por una nueva.

Por seguridad, para llevar a cabo este procedimiento es necesario tener acceso físico al equipo.

El siguiente procedimiento funciona para los equipos:

Cisco 806	Cisco 4700	Catalyst 2948G-L3
Cisco 827	Cisco AS5x00	Catalyst 4840G
Cisco uBR900	Cisco 6x00	Catalyst 4908G-L3
Cisco 1003	Cisco 7000 (RSP7000)	Catalyst 5500 (RSM)
Cisco 1004	Cisco 7100	Catalyst 8510-CSR
Cisco 1005	Cisco 7200	Catalyst 8510-MSR
Cisco 1400	Cisco 7500	Catalyst 8540-CSR
Cisco 1600	Cisco uBR7100	Catalyst 8540-MSR
Cisco 1700	Cisco uBR7200	Cisco MC3810
Cisco 2600	Cisco uBR10000	Cisco NI-2
Cisco 3600	Cisco 12000	Cisco VG200 Analog Gateway
Cisco 4500	Cisco LS1010	Route Processor Module
Cisco 1800	Cisco 2800	Cisco3800

Procedimiento.

1. Conecta el cable RJ45 a DB9 del puerto de consola del router hacia la PC.Utiliza la siguiente configuración:

    9600 baud rate

    No parity

    8 data bits

    1 stop bit

    No flow control

2. Si aún tienes acceso al router, escribe el comando **show version,** y escribe en una hoja el valor del registro de configuración, usualmente es 0x2102 ó 0x102

3. Si no cuentas con acceso al router (debido a que perdiste tu login), puedes considerar que el valor de registro es 0x2102

4. Utiliza el switch de energía para apagar y volver a encender el router.

5. Dependiendo del software que uses para la comunicación, la plataforma y el sistema operativo, presiona la tecla o la combinación correspondiente durante 60 segundos mientras el router lleva a cabo el proceso de encendido para ponerlo en modo ROMMON

Software	Platform	Operating System	Try This
Hyperterminal	IBM Compatible	Windows XP	Ctrl-Break ó Break
Hyperterminal	IBM Compatible	Windows 2000	Ctrl-Break
Hyperterminal	IBM Compatible	Windows 98	Ctrl-Break
Hyperterminal (version 595160)	IBM Compatible	Windows 95	Ctrl-F6-Break
Kermit	Sun Workstation	UNIX	Ctrl-\l

			Ctrl-\b
MicroPhone Pro	IBM Compatible	Windows	Ctrl-Break
Minicom	IBM Compatible	Linux	Ctrl-a f
ProComm Plus	IBM Compatible	DOS or Windows	Alt-b
SecureCRT	IBM Compatible	Windows	Ctrl-Break
Telix	IBM Compatible	DOS	Ctrl-End
Telnet	N/A	N/A	Ctrl-], then type **send brk**
Telnet to Cisco	IBM Compatible	N/A	Ctrl-]
Teraterm	IBM Compatible	Windows	Alt-b
Terminal	IBM Compatible	Windows	Break
			Ctrl-Break
Tip	Sun Workstation	UNIX	Ctrl-], then Break or Ctrl-c
			~#
VT 100 Emulation	Data General	N/A	F16
Windows NT	IBM Compatible	Windows	Break-F5
			Shift-F5
			Shift-6 Shift-4 Shift-b (^$B)
Z-TERMINAL	Mac	Apple	Command-b
N/A	Break-Out Box	N/A	Connect pin 2 (X-mit) to +V for half a second
	Cisco to aux port	N/A	Control-Shft-6, then b
	IBM Compatible	N/A	Ctrl-Break

6. Escribe **confreg 0x2142** en el prompt *rommon 1>* para bootear desde la memoria Flash sin cargar la configuración.

7. Escribe **reset** en el prompt *rommon 2>* El router se reinicia pero ignora la configuración salvada.

8. Escribe **no** después de cada pregunta de la configuración inicial o **ctrl-c** para saltarse el procedimiento de configuración inicial.

9. Escribe **enable** en el prompt *Router>* Debe aparecer el prompt *Router#*

10. **Importante.** Escribe **configure memory** o **copy startup-config running-config** para copiar la NVRAM en la memoria. No ingreses **configure terminal**

11. Ingresa **write terminal** o **show running-config.** El commando show running-config y write terminal muestran la configuración del router. En esta configuración se pueden ver en las interfaces el comando shutdown, que significa que las interfaces están apagadas, además puedes ver las contraseñas (enable password, enable secret, vty, console passwords) encriptadas o no, según corresponda. Las contraseñas que no están encriptadas se pueden seguir utilizando, las encriptadas tendrán que ser reestablecidas.

12. Escribe **configure terminal** y haz la configuración. El prompt debe verse: **hostname(config)#**

13. Escribe **enable secret ‹password›** para cambiar la contraseña enable secret

14. Escribe **config-register 0x2102,** o el valor que guardaste en el paso 2.

15. Presiona **ctrl.+z** o escribe **end,** para salir del modo de configuración. El prompt debe verse: hostname#

16. Escribe **write memory** o **copy running-config startup-config,** para guarder los cambios

Práctica 2

Configuración de “Router Information Protocol” en Cisco IOS

Introducción

Este documento describe como configurar tres routers en una topología de red simple, a diferencia de los switches.

RIP es un protocolo distance-vector, que emplea el número de saltos como métrica de ruteo. El número máximo de saltos permitido por RIP es de 15, cada routeador RIP transmite, por default, actualizaciones de sus tablas cada 30 segundos, utiliza UDP por el puerto 520 para entregar los paquetes.

Comparado con otros protocolos de ruteo como EIGRP, OSPF o IS-IS, RIP tiene un tiempo de convergencia muy alto, por lo que se considera obsoleto en comparación con estos.

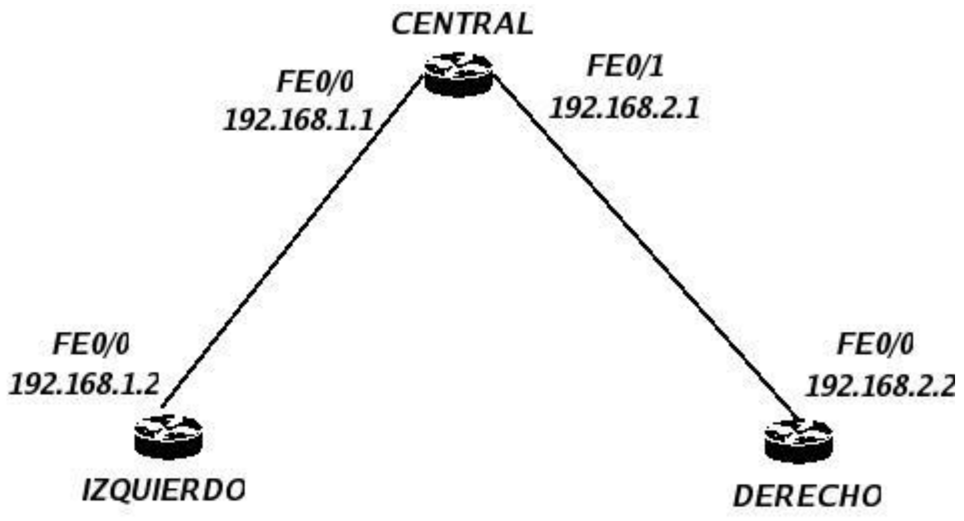
Prerrequisitos

Tener una terminal conectada y configurada por cada uno de los 3 routers.
No tener ninguna configuración en los equipos.

Antes de iniciar cualquier configuración es recomendable que establezcas tu consola en modo síncrono, lo que te servirá en el caso de que el router necesite mostrar información, esta no interrumpa el comando que tu estas ingresando.

	Comando o acción	**Descripción**
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#configure terminal	Entra al modo de configuración global
Paso 3	#line console 0	Entra al modo de configuración de la consola
Paso 4	#logging synchronous	Establece el modo síncrono

Debemos configurar las interfaces como lo muestra la figura:



De tal forma que tengamos:

Router Central:

```
hostname Central
interface FastEthernet0/0
    ip address 192.168.1.1 255.255.255.0
    no shutdown
interface FastEthernet0/1
    ip address 192.168.2.1 255.255.255.0
    no shutdown
```

Router Izquierdo:

```
hostname Izquierdo
interface FastEthernet0/0
    ip address 192.168.1.2 255.255.255.0
    no shutdown
```

Router Derecho:

```
hostname Derecho
interface FastEthernet0/0
    ip address 192.168.2.2 255.255.255.0
    no shutdown
```

También habilitaremos los mensajes del protocolo RIP, para poder ver la información de los paquetes de actualización enviados y recibidos por cada router, esto lo haremos en cada router:

	Comando o acción	**Descripción**
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#debug ip rip	Habilita o deshabilita el debug de RIP

Iniciando RIP

Para el router central:

	Comando o acción	**Descripción**
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#configure terminal	Entra al modo de configuración global
Paso 3	#router rip	Entra al modo de configuración del protocolo RIP
Paso 4	#network segmento-de-red en este caso: #network 192.168.1.0	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 5	#network segmento-de-red en este caso: #network 192.168.2.0	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 6	#end	(Opcional) Regresa el modo de configuración de terminal
Paso 7	#end	(Opcional) Regresa el modo privileged EXEC

Para el router izquierdo

	Comando o acción	**Descripción**
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#configure terminal	Entra al modo de configuración global
Paso 3	#router rip	Entra al modo de configuración del protocolo RIP
Paso 4	#network segmento-de-red en este caso: #network 192.168.1.0	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 5	#end	(Opcional) Regresa el modo de configuración de terminal
Paso 6	#end	(Opcional) Regresa el modo privileged EXEC

Para el router derecho

	Comando o acción	Descripción
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#configure terminal	Entra al modo de configuración global
Paso 3	#router rip	Entra al modo de configuración del protocolo RIP
Paso 4	#network segmento-de-red en este caso: #network 192.168.2.0	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 5	#end	(Opcional) Regresa el modo de configuración de terminal
Paso 6	#end	(Opcional) Regresa el modo privileged EXEC

Una vez hecho esto comenzaras a ver los mensajes RIP viajando entre los routers, a medida que estos mensajes llegan a router, este los agrega a su tabla, siempre y cuando:

El mensaje RIP contiene una red que no este en la tabla actual
El mensaje RIP contiene una red con una mejor métrica (menos saltos) que alguna que ya tenga en su tabla.

Para ver las tablas de ruteo puedes usar el comando:

```
#show ip route
```

En caso de que quieras comprobar la conectividad entre las redes 192.168.1.0 y 192.168.2.0, puedes hacer un ping desde el router izquierdo hacia el derecho, para saber que RIP esta bien configurado debes obtener respuesta.

Práctica 3

VLAN´S

Configuración de HWIC´s

Las interfaces EthernetSwitch HWIC son capa 2 10/100 BaseT, con capacidad de ruteo de capa 3. El trafico entre VLAN's se rutea a través de la plataforma del router y no se lleva a cabo en la tarjeta. Un módulo adicional para proporcionar energía a través de la línea puede agregarse, esto para teléfonos IP.

Agregando VLAN's

Las interfaces EtherSwitch HWIC soportan hasta 15 VLAN's.
En modo privilegiado, sigue los siguientes pasos para configurar una interfaz FastEthernet como capa 2.

	Comando	**Explicación**
Paso 1	#vlan database	Ingresa al modo de configuración VLAN
Paso 2	#vlan *vlan_id* *ej. #vlan 2*	Agrega una VLAN Ethernet
Paso 3	#exit	Actualiza la base de datos VLAN, la porpaga a través del dominio y regresa al modo EXEC

Comprobando la configuración VLAN

Comando show

En el modo VLAN database ingresa el comando chow para verificar la configuración de la VLAN

```
Router(vlan)#show
  VLAN ISL Id: 1
  Name: default
  Media Type: Ethernet
  VLAN 802.10 Id: 100001
  State: Operational
  MTU: 1500
  Translational Bridged VLAN: 1002
  Translational Bridged VLAN: 1003
  …
Exiting....
router#
router#
```

Comando show vlan-switch

El comando show vlan-switch, en el modo EXEC para verificar la configuración de la VLAN

```
router#show vlan-switch
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
----
1    default                              active     Fa0/1/1, Fa0/1/2, Fa0/1/3,
Fa0/1/4
                                                Fa0/1/5,    Fa0/1/6,    Fa0/1/7,
Fa0/1/8
                                                Fa0/3/0,    Fa0/3/2,    Fa0/3/3,
Fa0/3/4
                                                Fa0/3/5,    Fa0/3/6,    Fa0/3/7,
Fa0/3/8
2    VLAN0002                         active    Fa0/1/0
3    Red_VLAN                         active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
1    enet  100001     1500  -      -      -        -    -        1002   1003
2    enet  100002     1500  -      -      -        -    -        0      0
3    enet  100003     1500  -      -      -        -    -        0      0
1002 fddi  101002     1500  -      -      -        -    -        1      1003
1003 tr    101003     1500  1005   0      -        -    srb      1      1002
1004 fdnet 101004     1500  -      -      1        ibm  -        0      0
1005 trnet 101005     1500  -      -      1        ibm  -        0      0
router#
```

Eliminando una VLAN de la base de datos

No se puede eliminar las VLAN´s por default para los distintos medios: Para Ethernet VLAN 1, para FDDI o TokenRing VLAN's 1002 a 1005.
En modo privilegiado sigue los siguientes pasos:

	Comando	Explicación
Paso 1	#vlan database	Ingresa al modo de configuración VLAN
Paso 2	#no vlan *vlan_id* ej #no vlan 2	Elimina la VLAN
Paso 3	#exit	Actualiza la base de datos VLAN, la propaga a través del dominio y regresa al modo EXEC

Verificando la eliminación:

También se puede utilizar el comando **show** en el modo vlan database:

```
Router(vlan)#show
  VLAN ISL Id: 1
    Name: default
    Media Type: Ethernet
    VLAN 802.10 Id: 100001
    State: Operational
    MTU: 1500
    Translational Bridged VLAN: 1002
    Translational Bridged VLAN: 1003
  VLAN ISL Id: 1002
    Name: fddi-default
    Media Type: FDDI
    VLAN 802.10 Id: 101002
    State: Operational
    MTU: 1500
    Bridge Type: SRB
    Translational Bridged VLAN: 1
    Translational Bridged VLAN: 1003
<output truncated>
Router(vlan)#
```

También se puede utilizar el comando **show vlan-switch brief** en el modo priviliegiado:

```
Router#show vlan-switch brief
VLAN Name                             Status    Ports
---- -------------------------------- --------- ------------------------------
-
1    default                          active    Fa0/1/0, Fa0/1/1, Fa0/1/2
                                                Fa0/1/3, Fa0/1/4, Fa0/1/5
                                                Fa0/1/6, Fa0/1/7, Fa0/1/8
300  VLAN0300                         active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
```

```
1005 trnet-default                    active
Router#
```

Configurando las características opcionales de las interfaces

Cuando configures la velocidad y modo de transmisión de una interface, ten en cuenta:

Si los dos extremos de la línea soportan auto negociación, Cisco recomienda mantener la configuración de auto negociación.
Si una interface soporta auto negociación y la otra no, configura el modo de transmisión y la velocidad en ambas interfaces, no utilices la configuración automática en la interfaz en que la soporta.
Si no soportan la configuración automática ninguna de las interfaces, configura ambas interfaces con la misma velocidad y modo de transmisión.

Modificar la velocidad de la interfaz y el modo de transmisión, produce que la interfaz se reinicie.

Configurando la velocidad

En modo de configuración global:

	Comando	**Explicación**
Paso 1	#interface fastethernet *id* ej. Interface fastethernet0/0/0	Selecciona la interfaz a configurar
Paso 2	#speed |10 |100 |auto	Selecciona la velocidad

En caso de que selecciones la velocidad auto en una interfaz Ethernet 10/100-Mbps, la velocidad y el modo de transmisión se negocian automáticamente.

Configurando el modo de transmisión

En modo de configuración global:

	Comando	**Explicación**
Paso 1	#interface fastethernet *id* ej. Interface fastethernet0/0/0	Selecciona la interfaz a configurar
Paso 2	#duplex [auto | full |half]	Selecciona el modo de transmisión

En caso de que selecciones el modo de transmisión auto en una interfaz Ethernet 10/100-Mbps, la velocidad y el modo de transmisión se negocian automáticamente.

Como vimos en la practica 1 también se puede configurar una descripción para la interfaz. Agregarlo a la práctica.

Configurando la interfaz como acceso de capa 2

Nos sirve para asignar alguna interfaz a una VLAN.
En modo de configuración global:

	Comando	**Explicación**
Paso 1	#interface fastethernet *id* ej. #interface fastethernet0/0/0	Selecciona la interfaz a configurar
Paso 2	#switchport mode access	Configura la interfaz como acceso de capa 2
Paso 3	#switch port access vlan *vlan_num* *ej #switchport access vlan 2*	Especifica la VLAN asociada
Paso 4	#no shutdown	Activa la interfaz
Paso 5	#end	Sale del modo de configuración

Verificando la configuración:

```
Router#show running-config interface fastethernet 0/1/2
Building configuration...
Current configuration: 76 bytes
!
interface FastEthernet0/1/2
  switchport access vlan 3
  no ip address
end
```

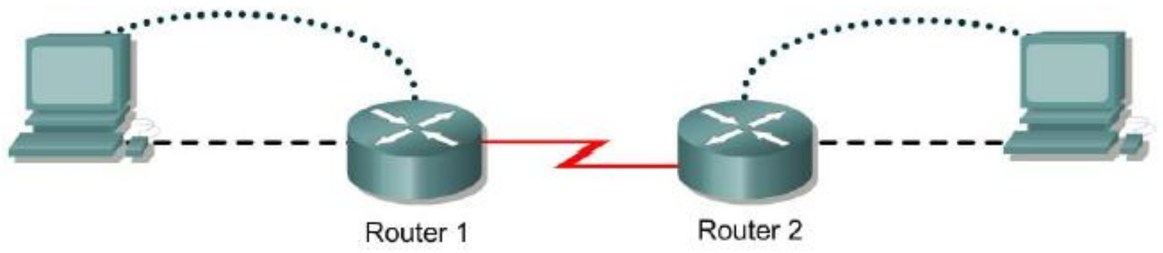
Ejemplo de configuración de una VLAN:

#vlan database
(vlan)#**vlan 1**
(vlan)#**vlan 2**
(vlan)#**exit**
#**configure terminal**
#**interface vlan 1**

```
#ip address 192.168.27.155 255.255.255.0
#no shut
#interface vlan 2
#ip address 192.168.27.158 255.255.255.0
#no shut
#interface FastEthernet0/0/0
#switchport access vlan 1
#interface FastEthernet0/0/1
#switchport access vlan 2
#exit
```

Práctica 4

Pruebas de conectividad – Traceroute



Objetivo:

- Usar el comando **`traceroute`** de Cisco IOS desde el router origen al router destino.
- Usar el comando **`tracert`** de MS-DOS desde la estación origen hasta el router destino.
- Verificar que la capa de red entre origen, destino y cada router que se encuentre en el camino esté funcionando correctamente.
- Recuperar información para evaluar la confiabilidad de la ruta de extremo a extremo.
- Determinar los retardos en cada punto de la ruta y si es posible alcanzar el host.

Información básica / Preparación

El comando **`traceroute,`** abreviado como **`trace`**, es una excelente herramienta para diagnosticar fallas en la ruta que emprende el paquete a través de una internetwork de routers. Puede ayudar a aislar los enlaces y routers problemáticos a lo largo del camino. El comando **`traceroute`** utiliza los paquetes ICMP y el mensaje de error generado por los routers cuando el paquete supera su Tiempo de Existencia (TTL). La versión de Windows de este comando es **`tracert`**.

Con la misma configuración realizada en los ejercicios de ping, y después de haber hecho las pruebas de conectividad en las capas 2 y 3 de dichos ejercicios desarrollar los siguientes puntos:

Iniciar una sesión en el router en el modo usuario.
Inicie una sesión a la petición de entrada de EXEC usuario en "izquierdo".

Descubrir las opciones de trace.

Escriba **`trace`** en la línea de comandos del router y presione INTRO.
¿Cuál fue la respuesta del router? ____

Usar la función ayuda con trace

Introduzca **`trace ?`** en la petición de entrada del router.
¿Cuál fue la respuesta del router? ____

Seguir descubriendo las opciones de trace

Entre en el modo EXEC privilegiado y escriba **`trace ?`**.
¿Cuál fue la respuesta del router? ____

¿Hay alguna diferencia entre los dos resultados de trace? ______________________________

Debería haber una opción adicional <cr>. Esto permite un comando **`traceroute`** extendido en el modo privilegiado. Esta opción no está disponible en el modo EXEC usuario.

Use el comando `traceroute`

Ingrese **`traceroute ip xxx.xxx.xxx.xxx`** donde xxx.xxx.xxx.xxx es la dirección IP del destino final.

Use uno de los routers de un extremo y haga `trace IP` al host del otro extremo. El router responderá de la siguiente manera:

```
izquierdo#traceroute 192.168.16.2
Type escape sequence to abort.
Tracing the route to 192.168.16.2
1 BHM (192.168.15.2) 16 msec 16 msec 16 msec
2 192.168.16.2 16 msec 16 msec 12 msec
izquierdo#
```

Si el resultado no tiene éxito, verifique las configuraciones del router y del host.

Siga usando el comando `traceroute`

Entre en los router y repita el uso del comando **`traceroute`**.

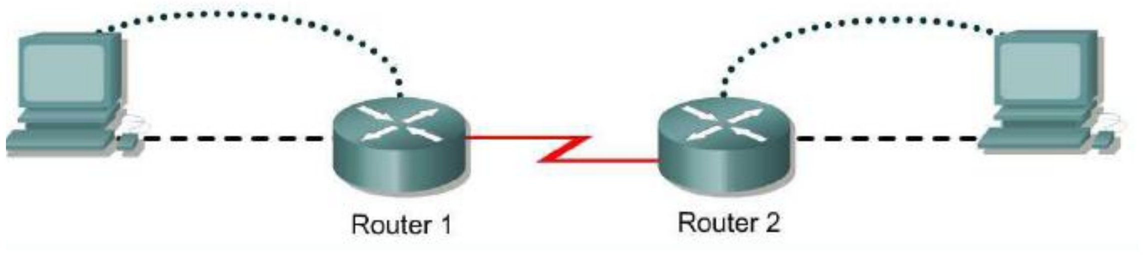
Usar el comando `trace` desde una estación de trabajo

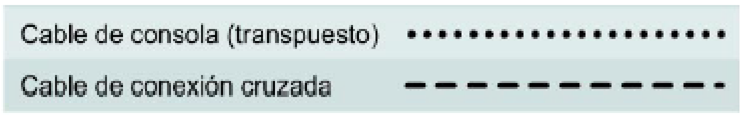
Desde la estación de trabajo de consola

Introduzca **tracert** y la misma dirección IP que se utilizó en la prueba del **trace** en "izquierdo" El primer salto será su gateway por defecto o la interfaz del router más cercana en la LAN a la cual está conectada la estación de trabajo. En la tabla que aparece a continuación, enumere los nombres de host y de las direcciones IP de los routers a través de los cuales se enrutó el paquete ICMP así como cualquier otra entrada que aparezca.

Nombre de Host	Dirección IP

Pruebas de conectividad – Ping




Cable de consola (transpuesto) ••••••••••••••••••••••
Cable de conexión cruzada – – – – – – – – – – –

Router	Nombre del router	Contraseña secret	Contraseña VTY y consola	Protocolo de enrutamiento	Sentencias de RIP
Router1	Izquierdo	practica	cisco	RIP	192.168.14.0 192.168.15.0
Router2	Derecho	practica	cisco	RIP	192.168.15.0 192.168.16.0

Router	Nombre de host IP	Dirección fastethernet	Mascaras de subred
Router1	Izquierdo	192.168.14.1	255.255.255.0
Router2	Derecho	192.168.16.1	255.255.255.0

Objetivo

- Usar el comando **`ping`** para enviar datagramas ICMP al host objetivo.
- Verificar que la capa de red entre el origen y el destino funcione correctamente.
- Recuperar información para evaluar la confiabilidad de la ruta hacia el host.
- Determinar los retardos a lo largo de la ruta y si el host se puede alcanzar o si está en funcionamiento.
- Utilizar el comando **`ping`** extendido para aumentar la cantidad de paquetes.

Información básica / Preparación

El comando **`ping`** es una buena herramienta para diagnosticar fallas de la Capa 1 a 3 del modelo OSI y diagnosticar la conectividad básica de la red. El comando ping envía un paquete ICMP al dispositivo especificado (estación de trabajo, servidor, router o switch) y luego espera la respuesta.

Se puede hacer ping a la dirección IP o el nombre de host. Para hacer ping al nombre de host de un router debe haber una tabla de consulta de host estática en el router o en el servidor DNS para la resolución de nombres a direcciones IP.

Cree una red con un cableado similar al del diagrama anterior. Y realice la configuración que se muestra.

-Configurar los routers 1 y 2

Verifique las configuraciones de los routers ejecutando **`show running-config`** en cada router. Si hay algún error, corríjalo y vuelva a realizar la verificación.

-Mostrar información sobre el mapeo de direcciones del host a la Capa 3

Introduzca **`show host`** en la petición de entrada del router. El router mostrará información acerca del mapeo de direcciones del host a la capa 3 (IP), de qué manera se adquirió esta información y la antigüedad de la entrada.

Haga una lista de los nombres de host y de las direcciones IP de cada uno.

Nombre de host	Dirección IP

Usar el comando `ping`

Introduzca **`ping xxx.xxx.xxx.xxx`** donde xxx.xxx.xxx.xxx representa la dirección IP que aparece en la lista anterior.

Repita el proceso con todas las direcciones IP enumeradas.

El router envía un paquete de Protocolo de Mensajes de Control en Internet (ICMP) para verificar la conexión de hardware y la dirección de la capa de red. El PC está actuando como consola del router, haciendo ping de un router a otro.

¿Las direcciones IP hicieron `ping`? ____

Examinar los resultados del comando ping

Vea el ejemplo del comando **`ping`** generado por un router.

```
lab-b#ping 192.168.3.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echoes to 210.93.105.1, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 68/68/168 ms
```

¿Qué indica el signo de exclamación (!)?
____

¿Qué indica el punto (.)? ____

¿Qué prueba el comando **`ping`** ? ____

Configurar las estaciones de trabajo

La configuración del host conectado al router "izquierdo" es:

Dirección IP	192.168.14.2
Máscara de subred IP	255.255.255.0
Gateway por defecto	192.168.14.1

La configuración del host conectado al router "derecho" es:

Dirección IP	192.168.16.2
Máscara de subred IP	255.255.255.0
Gateway por defecto	192.168.16.1

Hacer ping desde la estación de trabajo

Para verificar que la pila y el gateway por defecto TCP/IP de la estación de trabajo estén configurados y funcionen correctamente, utilice la ventana MS-DOS para hacer ping a los routers introduciendo el comando ping:

```
C:\> ping 192.168.14.1
```

El `ping` deberá responder con resultados exitosos. De lo contrario, verifique las configuraciones del host y del router directamente conectado.

Probar la conectividad de Capa 3

En el Símbolo de sistema, introduzca **`ping`** y la dirección IP de las interfaces de todos los routers. Esto probará la conectividad de Capa 3 entre la estación de trabajo y los routers.

¿El resultado del comando **`ping`** desde la estación de trabajo es el mismo que el del comando

`ping` desde un router?

Desde el host, hacer Telnet al router directamente conectado

Haga Telnet al router conectado. Escriba **`telnet`** y la dirección IP del gateway por defecto del router.(con lo cual tambièn se verifica la conectividad)

```
C:\>telnet 192.168.14.1
```

Cuando aparezca la petición de contraseña, introduzca **cisco**.

Realizar un ping extendido

Entre en el modo EXEC privilegiado. Escriba **`enable`** y luego la contraseña **laboratorio**. Escriba **`ping`** y presione **Intro**. Complete el resto de las peticiones de entrada como aparece a continuación:

```
Protocol [ip]:
Target IP address: 192.168.16.1
Repeat count [5]: 50
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 50, 100-byte ICMP Echos to 192.168.16.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (50/50), round-trip min/avg/max = 32/32/40
ms
GAD#
```

Observe la velocidad de respuesta del ping. ¿Cuál fue el tiempo de respuesta promedio?

Realizar otro ping extendido

Pero ahora varíe el tamaño del paquete y analice el resultado.

Realizar un ping extendido desde el host

Salga de la sesión Telnet y vuelva a la ventana MS-DOS del host. Escriba **`ping`** y presione **Intro**.

¿El ping extendido funciona de la misma manera en el router y en el host? ____________________

En el Símbolo de sistema escriba:

```
C:\>ping 192.168.16.1 -n 25
```

Deberá recibir 25 respuestas del comando.

Práctica 5

Dispositivos Vecinos y escalado de direcciones IP (DHCP)

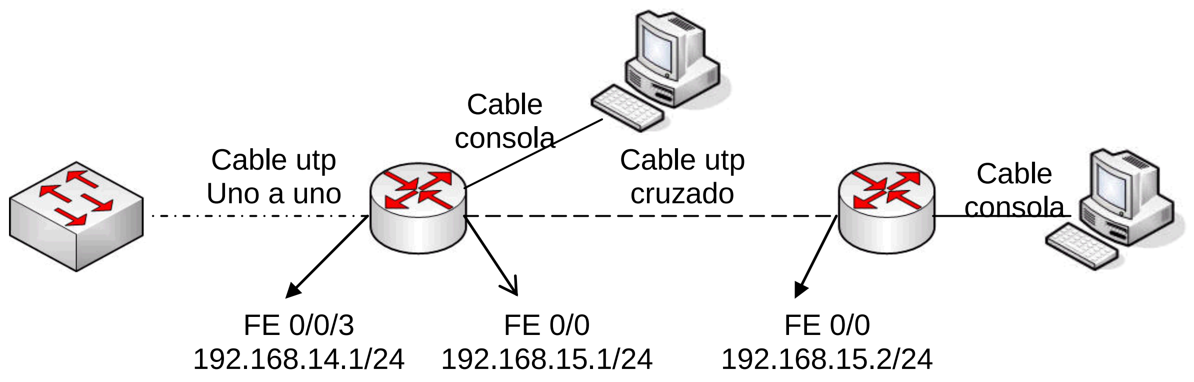
Breve introducción a los sistemas de monitoreo y protocolos de descubrimiento:

El protocolo de descubrimiento de **Cisco (CDP)** es un protocolo de la capa 2 que conecta los medios físicos inferiores y los protocolos de capa de red superiores. CDP se utiliza para obtener información acerca de los dispositivos vecinos. CDP es independiente del medio y del protocolo que se ejecuta en todos los equipos Cisco. CDP es un protocolo patentado nativo de los dispositivos de red de Cisco y que sólo se ejecuta en ellos.

Todo dispositivo configurado para CDP envía mensajes periódicos, conocidos como publicaciones. La versión 2 de CDP es la versión más reciente del protocolo. La versión 1 de CDP está disponible globalmente y de forma predeterminada con Cisco IOS Software Release 10.3 o posterior.

Desarrollo parte 1

a) Arme la siguiente topología



Nota: Es importante que realice el enlace entre routers con un cable cruzado y no usando un dispositivo intermedio ya que el cdp detectará el dispositivo inmediato vecino en la conexión.

Asígnale el nombre que desee a cada router Ejemplo :*"router1"y " router 2"*.

Complemento. Usemos una nueva opción que ofrece Cisco. Coloquemos un mensaje de conexión (banner) en cada uno de nuestros routers. Un mensaje de conexión debe ser una advertencia para no intentar la conexión a menos que se trate de un usuario autorizado.

P A S O S	E J E M P L O
Paso 1. Entre al modo de configuración global con el Comando configure Terminal.	*Router# configure terminal*
Paso 2. Ejecute el comando Banner	*Router(config)#banner motd # mensaje #*
Paso 3. Salga del router para volver a entrar y visualizar el mensaje	*Router(config)#exit* *Router# exit.*

b) Comandos cdp

Por default cdp está habilitado en los dispositivos Cisco. Para comprobar esto ejecute el siguiente comando en modo privilegiado de cada router.

Router1#show cdp * *Nos mostrará información de cdp, anote el número de versión que esta utilizando e inclúyalo en su reporte **

nota: en caso de estar deshabilitado teclee Router1# cdp run para habilitar cdp globalmente.

I.- Comando "*Show cdp neighbors*".

Intercale la ejecución de los comandos entre los routers y el switch para ver la salida de todos los dispositivos.

En modo privilegiado ejecute el siguiente comando que visualiza información acerca de las redes directamente conectadas al router

Router1#show cdp neighbors

De acauerdo a nuestra configuración obtendremos una salida en el Router1 como la que se muestra a continuación:

```
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
          S - Switch, H - Host, I - IGMP, r - Repeater
```

```
Device ID      Local Intrfce   Holdtme   Capability      Platform        Port
ID
Switch2         Fas 0/0/3        156            S I        WS-C2950-2      Fas
0/18
Router2        Fas 0/0          162          R S I       2811           Fas     0/0
```

El campo "capability" nos dice la capacidad que posee cada dispositivo que está conectado de acuerdo a la nomenclatura que utiliza cisco y nos da a conocer en la parte superior de la tabla:
IGMP(Protocolo de gestión de grupo de Internet). Utilizando IGMP el dispositivo puede funcionar como punto de distribución de multidifusión, entregando paquetes únicamente a los segmentos que son miembros del grupo de multidifusión, asegurando trayectos sin bucles y eliminando paquetes de multidifusión duplicados.

II.- Comando "*show cdp interface*".

En modo privilegiado ejecute el comando siguiente para obtener información que CDP utiliza para su publicación y transmisión de la trama de descubrimiento.

Router1#show cdp interface.

Compruebe los estados de sus interfaces.

III.- Comando "*show cdp neighbors detail*"

Ejecute en la línea de comando en modo privilegiado el siguiente comando que nos da información detallada de los equipos vecinos conectados directamente.

Router1#show cdp neighbors detail

Que información extra se incluye en la salida de este comando que antes no teníamos?.

IV.- Comando "debug cdp (packets/ ip / adjacency / events).

El comando debug nos mostrará información cdp específica de algún parámetro que le especifiquemos. Ejecutemos los siguientes comandos en modo privilegiado. (Espere por lo menos dos minutos para ver la salida del comando).

Router1#debug cdp packets *Información relacionada con paquetes cdp*
Router1#debug cdp adjacency *Información relacionada de los vecinos de cdp*
Router1#debug cdp events *Información relacionada de los eventos cdp*
Router1#debug cdp ip *Información relacionada con ip cdp*

Después de examinar la salida, introduzca el comando **undebug all** para detener la actividad de depuración.

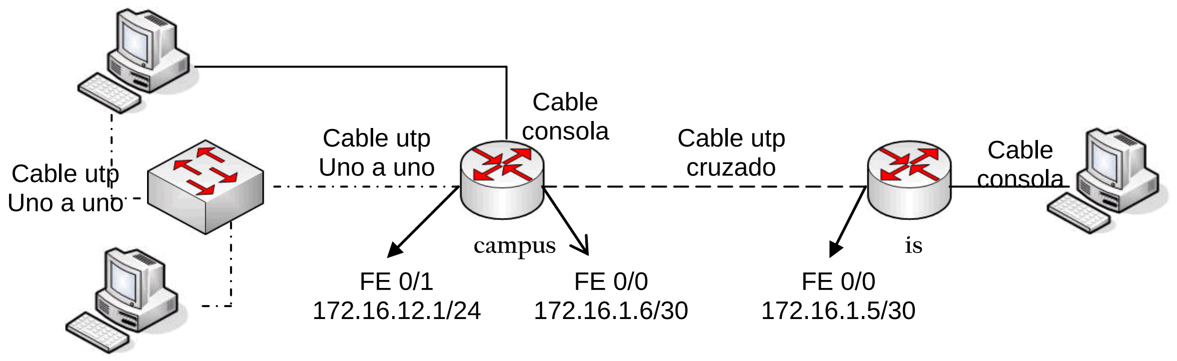
V.- Comando "show cdp traffic"

Para ver el recuento de información cdp que se ha generado en nuestro equipo teclee el siguiente comando en modo privilegiado.

Roueter1#show cdp traffic
Limpie los contadores antes vistos con el comando *Roueter1# clear cdp counters* y vuelva a ver los contadores para asegurarse que se han limpiado.

Parte 2
DHCP trabaja en modo cliente / servidor, valida los hosts (clientes DHCP) en una red IP para obtener sus configuraciones desde un servidor (servidor DHCP). Esto reduce el trabajo necesario para administrar una red IP. La opción de configuración más significativa que el cliente recibe desde el servidor es su dirección IP.

Desarrollo parte 2
a) Modifique su topología usada hasta ahora como se muestra en la figura siguiente:



Reasigne los nombres de los routers como se muestra en la figura anterior.

Tablas de host: Nos permitirán utilizar nombres para identificar todas las interfaces conectadas a ese router. Entonces utilizaremos esos nombres en lugar de direcciones IP en los comandos que utilizan direcciones IP para identificar una localización.

Agregue a cada router el nombre del router remoto con su dirección IP correspondiente
Con el comando ip host.

campus(config)# ip host isp 172.16.1.5
isp(config)# ip host campus 172.16.1.6
Ahora cada router podrá comprobar su conexión hacia el otro usando el nombre y no la IP.

b) El enrutamiento entre el ISP y el router de campus lo realizaremos mediante una ruta estática entre el ISP y el gateway. Esto se hace con el comando ip route:

ISP(config)# ip route 172.16.12.0 255.255.255.0 172.16.1.6

* Una ruta estática no posee ningún protocolo de encaminamiento y la ruta a seguir por los mensajes la decide el administrador y no el router.

c) Creamos una ruta determinada entre el gateway y el ISP en el router "campus" de la manera siguiente:

campus(config)# ip route 0.0.0.0 0.0.0.0 172.16.1.5

* Con una ruta determinada logramos que todas las direcciones ip que el router no conoce salgan por la ip que asignemos, en éste caso 172.16.1.5

d) Creamos un nombre para el almacén de direcciones llamado "campus" y situamos al router en modo de configuración DHCP especializado.

campus (config)# ip dhcp pool campus

e) Especificamos el número de subred y la máscara del almacén de direcciones DHCP con el comando "network". Escriba todos los comandos que le siguen para terminar la configuración.

Campus (dhcp-config)# network 172.16.12.0 255.255.255.0
Campus (dhcp-config)# default-router 172.16.12.1 ** Aquí especificamos la dirección IP del router predeterminado.*
Campus (dhcp-config)# dns-server 172.16.1.2 ** Esta sería la manera de indicar la dirección ip de un servidor DNS*

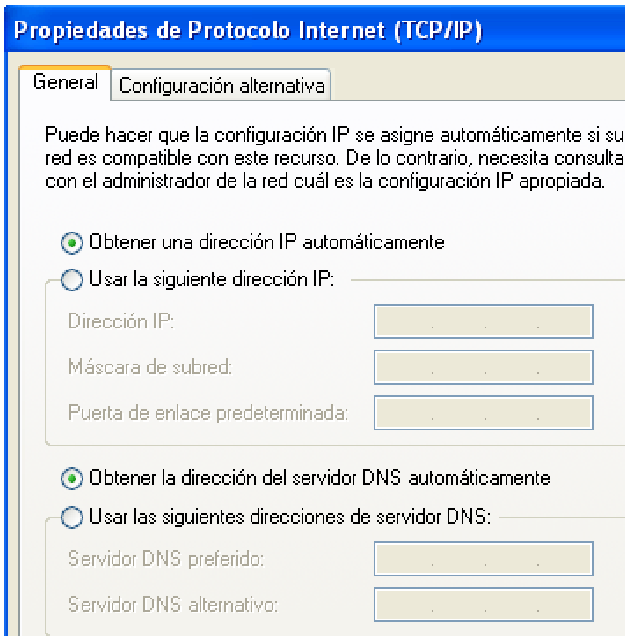
f) Excluyamos algunas direcciones ip que deseamos que no se asignen por dhcp ya que las tenemos en uso. Esto se logra con el comando "excluded-address".

Campus (dhcp-config)# ip dhcp excluded-address 172.16.12.1 172.16.12.10

Ya que excluímos la ip 172.16.12.10 démosle un uso asignándola a un servidor netbios:

Campus (dhcp-config)# netbios-name-server 172.16.12.10

g) Verificamos el funcionamiento DHCP en cada estación de trabajo conectada en el switch. Configure las propiedades TCP/IP de modo que la estación de trabajo obtenga una dirección IP y una dirección dns del servidor DHCP.


Propiedades de Protocolo Internet (TCP/IP)

General | Configuración alternativa

Puede hacer que la configuración IP se asigne automáticamente si su red es compatible con este recurso. De lo contrario, necesita consulta con el administrador de la red cuál es la configuración IP apropiada.

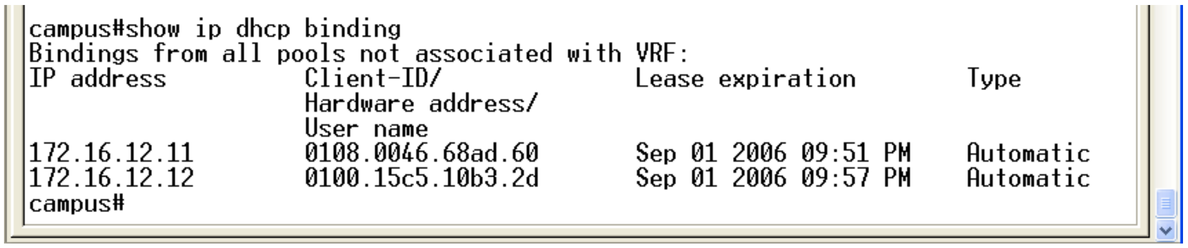
[✓ Obtener una dirección IP automáticamente]
Usar la siguiente dirección IP:
Dirección IP: . . .
Máscara de subred: . . .
Puerta de enlace predeterminada: . . .

[✓ Obtener la dirección del servidor DNS automáticamente]
Usar las siguientes direcciones de servidor DNS:
Servidor DNS preferido: . . .
Servidor DNS alternativo: . . .

Resetee la computadora conectada al switch después de hacer los últimos cambios y verifique la dirección IP asiganada con el comando ipconfig en una ventana de DOS.

Visualice los enlaces para los host desde el router campus, con el comando siguiente:
Campus> show ip dhcp binding

Deverá ver una salida como la que se muestra a continuación:



```
campus#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address          Client-ID/              Lease expiration        Type
                    Hardware address/
                    User name
172.16.12.11        0108.0046.68ad.60       Sep 01 2006 09:51 PM    Automatic
172.16.12.12        0100.15c5.10b3.2d       Sep 01 2006 09:57 PM    Automatic
campus#
```

Por default el DHCP alquila las direcciones IP por 24 horas

Práctica 6

Configuración de NAT, VLAN’s, Ruteo RIP, Servicio de INTERNET

INTRODUCCIÓN

Una VPN es una red virtual que se crea "dentro" de otra red, como por ejemplo Internet. Generalmente las redes privadas se crean en redes públicas, en las que se quiere crear un entorno confidencial y privado. La VPN nos permitirá trabajar como si estuviésemos en la red local, es totalmente transparente para el usuario.

Una vez establecida la conexión de la red privada virtual los datos viajan encriptados de forma que sólo el emisor y el receptor son capaces de leerlos.

Para poder realizar una VPN se necesita un servidor (o host) que espera conexiones entrantes, y uno o varios clientes, que se conectan al servidor para formar la red privada.

Otra aplicación es que podemos establecer conexiones seguras entre otros equipos para acceder a los recursos del otro equipo de forma segura y confidencial, por ejemplo a impresoras, documentos, servidores de base de datos, aplicaciones específicas, etc.

Su funcionamientos es exactamente igual que cualquier otra conexión de red, es decir, dentro de la VPN cada equipo tendrá una IP, todas las conexiones usando esa IP estarán funcionando dentro de la VPN, el usuario simplemente tendrá que usar las IPs de la VPN, y no preocuparse de nada más, el resto ya lo hace el cliente VPN y el servidor VPN.

Prerrequisitos

- Tener una Hyperterminal conectada y configurada por cada uno de los 2 routers.
- No tener ninguna configuración en los equipos.

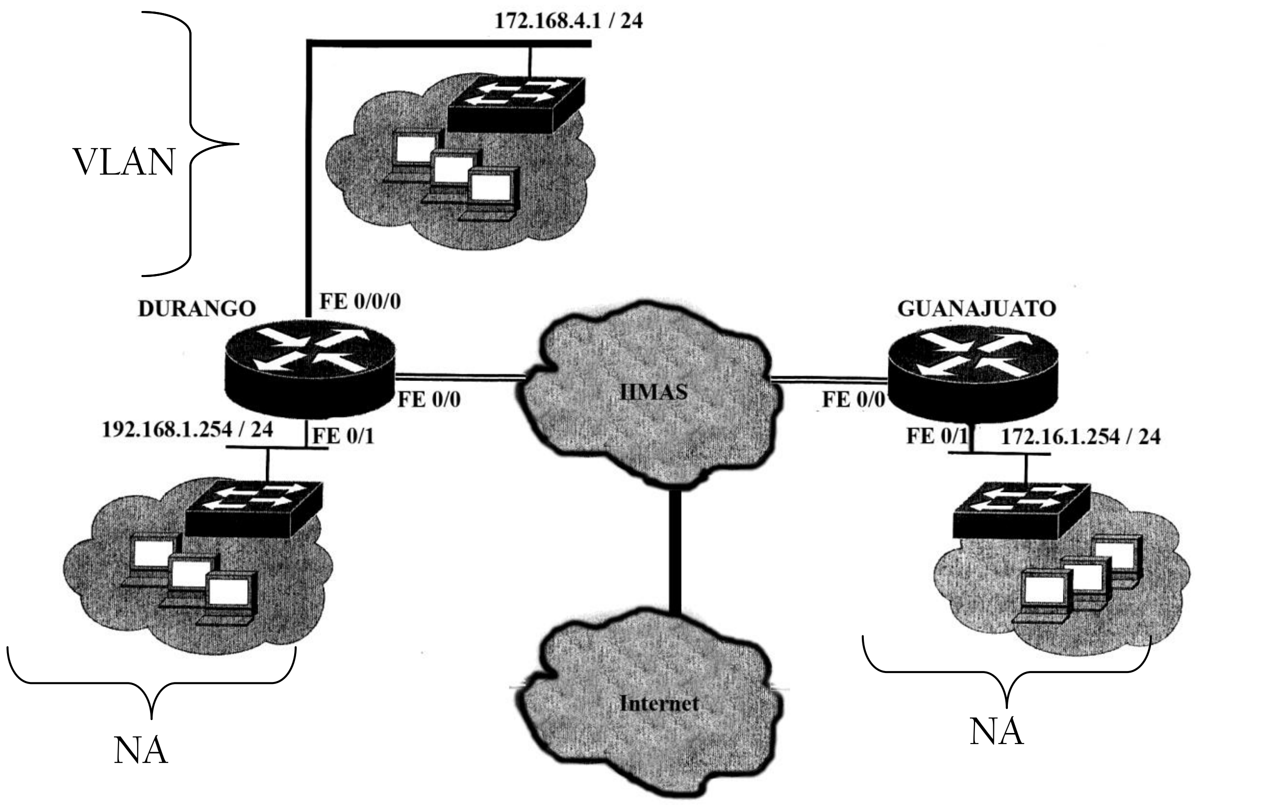
	Comando o acción	Descripción
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#erase startup-config	Elimina la configuración de inicio
Paso 3	#reload	Carga la configuración de inicio

- Tres PC con tarjetas de red Ethernet.
- **Contar con dos direcciones ip de red del IIMAS y el gateway.**
- Es recomendable establecer la consola de cada terminal en modo síncrono. Esto es para que la información que arroje el router no interrumpa algún comando que se ejecute desde la terminal.

	Comando o acción	Descripción
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#configure terminal	Entra al modo de configuración global
Paso 3	#line console 0	Entra al modo de configuración de la consola
Paso 4	#logging synchronous	Establece el modo síncrono

- Tres PC con tarjetas de red Ethernet.

Configuración de los routers



Debemos configurar las router como se muestra en la figura, para esto debemos establecer las direcciones IP de los puertos de cada router. Para realizar esto, las direcciones IP de los puertos hacia el IIMAS deben contener las direcciones que debió investigar o pedir en el departamento de computo del IIMAS. Para los otros puertos pueden ser las que desee.

En cada router sólo es necesario utilizar dos puertos, uno conectado a la red del IIMAS y otro conectado a un switch para que se puedan conectar varias computadoras a éste. De cada router, la interfaz "FastEthernet 0/0" se va a conectar a la red del IIMAS; y la interfaz "FastEthernet 0/1" se va a conectar a un switch.

Para el router DURANGO:

```
    hostname DURANGO
    interface FastEthernet0/0
        ip address "de la red del IIMAS"
        no shutdown
        description Enlace Ethernet de Durango hacia el IIMAS
    interface FastEthernet0/1
        ip address 192.168.1.254 255.255.255.0
        no shutdown
        description Enlace Ethernet hacia el Switch 1
    interface FastEthernet 0/0/0
        no shutdown
        description Enlace Ethernet hacia el Swich 3
```

Para el router Guanajuato:

```
    hostname GUANAJUATO
    interface FastEthernet0/0
        ip address "de la red del IIMAS"
        no shutdown
        description Enlace Ethernet de Guanajuato hacia el IIMAS
    interface FastEthernet0/1
        ip address 172.16.1.254 255.255.255.0
        no shutdown
        description Enlace Ethernet hacia el Switch 2
```

Habilite el protocolo RIP como lo visto en la práctica 3.

Para el router DURANGO:

	Comando o acción	**Descripción**
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#configure terminal	Entra al modo de configuración global
Paso 3	#router rip	Entra al modo de configuración del protocolo RIP
Paso 4	#network 192.168.1.0	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 5	#network 172.168.4.0	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 6	#network “segmento red del IIMAS”	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 7	#exit	Regresa el modo de configuración de terminal.
Paso 8	#wr	Guarda la configuración actual.

Para el router GUANAJUATO:

	Comando o acción	**Descripción**
Paso 1	#enable	Habilita el modo privileged EXEC

Paso 2	#configure terminal	Entra al modo de configuración global
Paso 3	#router rip	Entra al modo de configuración del protocolo RIP
Paso 4	#network 172.16.1.0	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 5	#network "segmento de red del IIMAS"	Indica a RIP cual es el segmento de red conectado directamente al router
Paso 6	#exit	Regresa el modo de configuración de terminal
Paso 7	#wr	Guarda la configuración actual.

Configuración de VLAN

Ahora, configuraremos la VLAN para el router DURANGO.

	Comando	**Explicación**
Paso 1	>enable	Entra al modo de acceso privilegiado
Paso 2	#vlan database	Ingresa a la base de datos de las vlan
Paso 3	(vlan)#vlan 22	Agrega una vlan
Paso 4	#exit	Actualiza la base de datos de vlan
Paso 5	#configure terminal	Ingresa al modo de configuración global
Paso 6	#interface fastEthernet 0/0/0	Ingresa al modo de configuración de interface.
Paso 7	(config-if)#switchport mode access	Configure como modo de acceso switcheable
Paso 8	(config-if)#switchport access vlan 22	Indica qué es lo que va aswitchear
Paso 9	(config-if)#no shutdown	Levanta el estatus de la interface
Paso 10	#exit	Salida de la configuración de interface.
Paso 11	#interface vlan 22	Entra al modo de configuración de interface.
Paso 12	(config-if)#ip address 172.168.4.1 255.255.255.0	Se ingresa la dirección IP de la VLAN 22
Paso 13	(config-if)#no shutdown	Levanta el estatus de la interface VLAN 22
Paso 14	(config-if)#exit	Salida de la configuración de interface
Paso 15	(config)#exit	Salida a la configuración global.
Paso 16	#wr	Guarda la configuración actual.

Configuración de las NAT

Como la interfaz FastEthernet 0/0 es la que se conecta a la red del IIMAS entonces se configura como red externa de la nat (ip nat outside), y como la interfaz FastEthernet 0/1 es la que pertenece a la red privada entonces se configura como red interna de la nat (ip nat inside). Para configurar las NAT de cada router, sigua los pasos que a continuación se muestran.

Para el router DURANGO:

	Comando	Explicación
Paso 1	>enable	Modo de acceso privilegiado.
Paso 2	#configure terminal	Configuración global.
Paso 3	(config)#interface fastEthernet 0/0	Configuración de interface.
Paso 4	(config-if)#ip nat outside	Declaración de salida del NAT.
Paso 5	(config-if)#exit	Salida de configuración de interface.
Paso 1	(config)#interface fastEthernet 0/1	Configuración de interface.
Paso 7	(config-if)#ip nat inside	Declaración de salida del NAT.
Paso 8	(config-if)#exit	Salida de configuración de interface.
Paso 1	(config)#interface fastEthernet 0/0	Configuración de interface.
Paso 10	(config-if)#ip nat pool testpool "dirección del IIMAS para el router DURANGO" "dirección de IIMAS" netmask 255.255.255.0	Se configura el conjunto de direcciones para salir por la nat, en este caso sólo es el puerto del router DURANGO.
Paso 11	(config-if)#exit	Salida del modo de configuración de interface.
Paso 12	(config)#ip nat inside source list 1 interface fastEthernet 0/0 overload	Se declara la fuente de la NAT en determinada interface.
Paso 13	(config)#access-list 1 permit any	Permite el acceso a cualquier equipo.
Paso 14	(config)#access-list 1 permit 192.168.1.0 0.0.0.255	Permite el acceso a cualquier PC con dirección del segmento de red permitida.
Paso 15	(config)#ip default-gateway **"dirección del IIMAS"**	Se declara la dirección del gateway.
Paso 16	(config)#ip route 0.0.0.0 0.0.0.0 FastEthernet 0/0	Ruta estática para la interface 0/0
Paso 17	(config)#ip route 0.0.0.0 0.0.0.0 "dirección del gateway"	Ruta estática para cualquier máquina a través del gateway.
Paso 18	(config)#exit	Salida de la configuración globla.
Paso 19	#wr	Guarda la configuración actual

Para el router GUANAJUATO:

	Comando	Explicación
Paso 1	>enable	Modo de acceso privilegiado.
Paso 2	#configure terminal	Configuración global.
Paso 3	(config)#interface fastEthernet 0/0	Configuración de interface.
Paso 4	(config-if)#ip nat outside	Declaración de salida del NAT.
Paso 5	(config-if)#exit	Salida de configuración de interface.
Paso 1	(config)#interface fastEthernet 0/1	Configuración de interface.
Paso 7	(config-if)#ip nat inside	Declaración de salida del NAT.
Paso 8	(config-if)#exit	Salida de configuración de interface.

Paso 1	(config)#interface fastEthernet 0/0	Configuración de interface.
Paso 10	(config-if)#ip nat pool testpool "dirección del IIMAS para el router GUANAJUATO" "dirección del IIMAS para el router GUANAJUATO" netmask 255.255.255.0	Se configura el conjunto de direcciones para salir por la nat, en este caso sólo es el puerto del router Guanajuato.
Paso 11	(config-if)#exit	Salida del modo de configuración de interface.
Paso 12	(config)#ip nat inside source list 1 interface fastEthernet 0/0 overload	Se declara la fuente de la NAT en determinada interface.
Paso 13	(config)#access-list 1 permit any	Permite el acceso a cualquier equipo.
Paso 14	(config)#access-list 1 permit 172.16.1.0 0.0.0.255	Permite el acceso a cualquier PC con dirección del segmento de red permitida.
Paso 15	(config)#ip default-gateway **"dirección del IIMAS"**	Se declara la dirección del gateway.
Paso 16	(config)#ip route 0.0.0.0 0.0.0.0 FastEthernet 0/0	Ruta estática para la interface 0/0
Paso 17	(config)#ip route 0.0.0.0 0.0.0.0 "dirección del gateway"	Ruta estática para cualquier máquina a través del gateway.
Paso 18	(config)#exit	Salida de la configuración globla.
Paso 19	#wr	Guarda la configuración actual

Con estos comandos, la red 192.168.1.0 y la red 172.16.1.0 tienen salida a Internet, además cualquier máquina en estas redes, pueden ver a los dos routers y tienen acceso a internet. Pero no se pueden comunicar entre máquinas de la red 192.168.1.0 con alguna de la 172.16.1.0 o viceversa pues son redes privadas. También las máquinas conectadas a la red 172.168.4.0 puede ver a las máquinas de la red 192.168.1.0 y los puertos de los router GUANAJUATO y DURANGO, pero no a las máquinas de la red 172.16.1.0, pues es de nueva cuenta una red privada para la 172.168.4.0.

Puede observar las tablas de ruteo de los router pormedio del comando "#show ip route", también los detalles rápidos de las interfaces con "#show ip interface brief", además del comando "tracert A.B.C.D".

Se recomienda que realice una Virtual Terminal Lines (VTL), para un acceso remoto a la consola del router. Para cualquier router realice los siguientes pasos.

	Comando o acción	**Descripción**
Paso 1	#enable	Habilita el modo privileged EXEC
Paso 2	#configure terminal	Entra al modo de configuración global
Paso 3	#line vty numero-de-linea	Inicia el modo de configuración de alguna linea
Paso 4	#password contraseña	Establece la contraseña para una linea

Paso 5	#login	Habilita la autenticación para la linea
Paso 6	#end	Regresa el modo privileged EXEC
Paso 8	Desde otro dispositivo de la red intenta abrir una conexión telnet hacia el router	Verificar remotamente que se puede acceder al router

Desde otra máquina de red, intente abrir una conexión telnet hacia el router configurado.

Reporte las configuraciones finales de los router, así como los pings de los router y los de las tres PC's conectadas a cada red de los router.

Consideraciones.

Debe tener cuidado con la conexión física de los dispositivos hacia la red del IIMAS, debe tener en cuenta cuando se usan cables de configuración cruzada o de configuración lineal. Además de que si no cuenta con las direcciones IP del IIMAS, no podrá realizar esta práctica.
Debe conocer las configuraciones de los cables y si no cuenta con ellos, deberá fabricar los necesarios para conectar los dispositivos de red para esta práctica.

Práctica 7

Captura y análisis de paquetes en Ethernet

Hardware :

PC con tarjeta ethernet
Cable de Red

Software:

Windows / Linux
Internet
WinDump 3.9.3/ Tcpdump
WinPcap 3.1
Ethereal 0.99.0

Configuración:

Para Windows

Instalar WinPcap 3.1
Descargar y ejecutar WinDump
Descargar e Instalar Ethereal 0.99.0

Para Linux es análogo

Objetivo:

Capturar tráfico de una red ethernet para analizar y descomponer la información que viaja a través de la red.

Desarrollo:

Utilización de Tcpdump para almacenar datos en un archivo

#tcpdump -n –i eth0

-n indica a Tcpdump que no resuelva las direcciones IP generando nombres de dominio, y los número de puertos generando nombres de servicio.

-i <interfaz> indica a Tcpdump qué interfaz debe observar.

-s <longitud> indica a Tcpdump qué parte del paquete debe registrar, admite valor 0 para utilizar la longitud necesaria para capturar paquetes completos.

-w <capfile.lpc> Se envía el resultado de Tcpdump al archivo especificado. (La extensión lpc indica que pertenece a un archivo con formato libcap)

Nota: Para saber desde que interfaz estarás escuchando utilizando en Linux puedes utilizar *ifconfig* y en Windows *windump –D*

Ejemplo para Windows
:
```
C:\Documents and Settings>windump -D
1.\Device\NPF_GenericDialupAdapter (Generic dialup adapter)
2.\Device\NPF_{A9D636B0-6026-42FD-92AE-707BA360D5DF}  (Broadcom  440x  10/100
Integrated Controller (Microsoft's Packet Scheduler) )
```

En este caso la sintáxis para windows sería: **>windump -n –i 2**

Captura de datos en un archivo:

#tcpdump –n –i eth0 –s 1515 –w eth0.lpc

Cuando se ejecuta Tcpdump este seguirá escribiendo en la ubicación señalada hasta q se desborde la partición, por tanto se debe tener cuidado con el tiempo de ejecución.

Para ver el tráfico en pantalla al mismo tiempo que se escribe en archivo (sólo válido en linux):

#tcpdump –n –i eth0 –s 1515 –l | tee outfile.txt

El modificador –l infica a tcpdump que debe crear un búfer de líneas para la salida y al mandar la salida a través de un tubo a la utilidad tee se envía el resultado a la pantalla y al archivo de texto outfile.txt

Para terminar la captura oprime ctrl-C.

Utilización de tcpdump para leer datos de contenido completo almacenados

Una vez que la herramienta Tcpdump ha capturado lso paquetes, podemos utilizarla para leer los archivos de seguimiento y ver lo que contienen. Emplearemos el modificador –r junto con el nombre del archivo capturado, con objeto de visualizar su contenido.

#tcpdump –n –r eth0.lpc -c 4

El modificador c es para especificar que solo deben mostrarse 4 paquetes. La alternativa es mandar los resultados a more o a less (solo en linux) a través de un tubo, para mostrar sólo una pantalla de resultados de cada vez:

#tcpdump –n –r eth0.lpc | more
ó
>windump -n -r 1.lpc | more
Primero se explicará las convenciones de salida con el protocolo ICMP de tcpdump. La siguiente tabla explica los campos del primer paquete:

```
C:\Documents and Settings >windump -n -r 1.lpc -c4 icmp
reading from file 1.lpc, link-type EN10MB (Ethernet)
22:19:25.757988 IP 217.141.249.1 > 172.28.6.249: ICMP host 80.104.151.159 unreachable,
length 36
```

Valor	Explicación
15:20:04.78092	Marcación de la hora
172.27.20.4	IP de origen
>	indicador de la dirección
IP de destino	
icmp:echo request	Tipo de mensaje ICMP

El tráfico UDP también aparece, se interpreta de la siguiente forma:

#tcpdump –n –r sf1.lpc –c2 udp

Valor	Explicación
15:20:21.78092	Marcación de la hora
172.27.20.4	IP de origen
41197	Puerto de origen
>	indicador de la direccion
192.68.60.5	IP de destino
udp 300	Longitud del datagrama en bytes

El tráfico TCP es más complicado que ICMP o que UDP

#tcpdump –n –r em0.lpc –c 8 tcp

La siguiente tabla desglosa los valores que hay en el paquete y da un descripción de cada uno de los campos, tal y como lo interpreta tcpdump.

Valor	Explicación
15:20:21.78092	Marcación de la hora
172.27.20.4	IP de origen
41197	Puerto de origen
>	indicador de la direccion
192.68.60.5	IP de destino

S	Indicador SYN de TCP activado
2345677536	ISN, número inicial de secuencia de TCP
2345677536	Número de secuencia del proximo byte de datos de la aplicación que espera TCP
(0)	Contador de datos de aplicación en este segmento
win 5840	Tamaño de ventana de TCP en bytes
mss 1460	La opción MSS (tamaño de segmento máximo) de TCP tiene un valor de 1460 bytes
sackOK	Acuse de recibo selectivo admitido por la computadora de origen (es otra opción TCP)
timestamp 25027249 0	Valor de marcación de hora y configuración del echo de respuesta de marcación de hora
nop	significa “inoperate” se incluye para rellenar correctamente la sección de opciones de TCP
wscale 0	El emisor admite el escalado de ventanas de TCP; el multiplicador actual es 0
(DF)	Especifica “no fragmentar” para este paquete

Para observar el encabezado del nivel de enlace se emplea el modificador –e

#tcpdump –n –r em0.lpc –e –c1 tcp

En este caso se pueden observar las direcciones MAC de origen y destino.

Otro modificador es –X que da lugar a que a la salida se muestre el paquete en notación hexadecimal a la izquierda y en notación ASCII a la derecha.

ETHEREAL

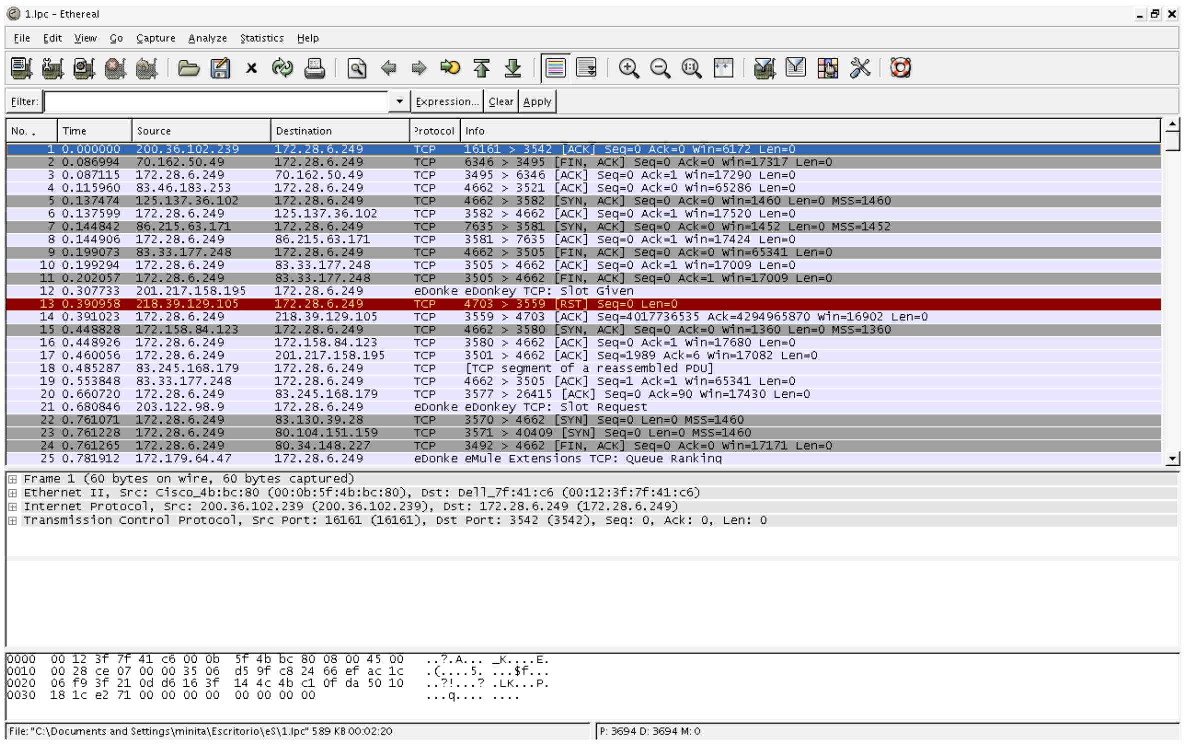
Analizar tráfico desde la interfaz gráfica con Ethereal.

Ethereal puede capturar paquetes en tiempo real empleando la ventana Capture Options

Menu Capture → Start

Ethereal también puede abrir un archivo de seguimiento empleando la secuencia Archivo→ Abrir.

Vamos a abrir el archivo capturado.



```
No. .  Time      Source           Destination      Protocol Info
    1 0.000000  200.36.102.239   172.28.6.249     TCP      16161 > 3542 [ACK] Seq=0 Ack=0 Win=6172 Len=0
    2 0.086994  70.162.50.49     172.28.6.249     TCP      6346 > 3495 [FIN, ACK] Seq=0 Ack=0 Win=17317 Len=0
    3 0.087115  172.28.6.249     70.162.50.49     TCP      3495 > 6346 [ACK] Seq=0 Ack=1 Win=17290 Len=0
    4 0.115960  83.46.183.253    172.28.6.249     TCP      4662 > 3521 [ACK] Seq=0 Ack=0 Win=65286 Len=0
    5 0.137474  125.137.36.102   172.28.6.249     TCP      4662 > 3582 [SYN, ACK] Seq=0 Ack=0 Win=1460 Len=0 MSS=1460
    6 0.137599  172.28.6.249     125.137.36.102   TCP      3582 > 4662 [ACK] Seq=0 Ack=1 Win=17520 Len=0
    7 0.144842  86.215.63.171    172.28.6.249     TCP      7635 > 3581 [SYN, ACK] Seq=0 Ack=0 Win=1452 Len=0 MSS=1452
    8 0.144906  172.28.6.249     86.215.63.171    TCP      3581 > 7635 [ACK] Seq=0 Ack=1 Win=17424 Len=0
    9 0.199073  83.33.177.248    172.28.6.249     TCP      4662 > 3505 [FIN, ACK] Seq=0 Ack=0 Win=65341 Len=0
   10 0.199294  172.28.6.249     83.33.177.248    TCP      3505 > 4662 [ACK] Seq=0 Ack=1 Win=17009 Len=0
   11 0.202057  172.28.6.249     83.33.177.248    TCP      3505 > 4662 [FIN, ACK] Seq=0 Ack=1 Win=17009 Len=0
   12 0.307733  201.217.158.195  172.28.6.249     eDonke   eDonkey TCP: Slot Given
   13 0.390958  218.39.129.105   172.28.6.249     TCP      4703 > 3559 [RST] Seq=0 Len=0
   14 0.391023  172.28.6.249     218.39.129.105   TCP      3559 > 4703 [ACK] Seq=4017736535 Ack=4294965870 Win=16902 Len=0
   15 0.448828  172.158.84.123   172.28.6.249     TCP      4662 > 3580 [SYN, ACK] Seq=0 Ack=0 Win=1360 Len=0 MSS=1360
   16 0.448926  172.28.6.249     172.158.84.123   TCP      3580 > 4662 [ACK] Seq=0 Ack=1 Win=17680 Len=0
   17 0.460056  172.28.6.249     201.217.158.195  TCP      3501 > 4662 [ACK] Seq=1989 Ack=6 Win=17082 Len=0
   18 0.485287  83.245.168.179   172.28.6.249     TCP      [TCP segment of a reassembled PDU]
   19 0.553848  83.33.177.248    172.28.6.249     TCP      4662 > 3505 [ACK] Seq=1 Ack=1 Win=65341 Len=0
   20 0.660720  172.28.6.249     83.245.168.179   TCP      3577 > 26415 [ACK] Seq=0 Ack=90 Win=17430 Len=0
   21 0.680846  203.122.98.9     172.28.6.249     eDonke   eDonkey TCP: Slot Request
   22 0.761071  172.28.6.249     83.130.39.28     TCP      3570 > 4662 [SYN] Seq=0 Len=0 MSS=1460
   23 0.761228  172.28.6.249     80.104.151.159   TCP      3571 > 40409 [SYN] Seq=0 Len=0 MSS=1460
   24 0.761265  172.28.6.249     80.34.148.227    TCP      3492 > 4662 [FIN, ACK] Seq=0 Ack=0 Win=17171 Len=0
   25 0.781912  172.179.64.47    172.28.6.249     eDonke   eMule Extensions TCP: Queue Ranking
```

```
⊞ Frame 1 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Cisco_4b:bc:80 (00:0b:5f:4b:bc:80), Dst: Dell_7f:41:c6 (00:12:3f:7f:41:c6)
⊞ Internet Protocol, Src: 200.36.102.239 (200.36.102.239), Dst: 172.28.6.249 (172.28.6.249)
⊞ Transmission Control Protocol, Src Port: 16161 (16161), Dst Port: 3542 (3542), Seq: 0, Ack: 0, Len: 0
```

```
0000  00 12 3f 7f 41 c6 00 0b  5f 4b bc 80 08 00 45 00   ..?.A... _K....E.
0010  00 28 ce 07 00 00 35 06  d5 9f c8 24 66 ef ac 1c   .(....5. ...$f...
0020  06 f9 3f 21 0d d6 16 3f  14 4c 4b c1 0f da 50 10   ..?!...? .LK...P.
0030  18 1c e2 71 00 00 00 00  00 00 00 00               ...q.... ....
```

```
File: "C:\Documents and Settings\minita\Escritorio\eS\1.lpc" 589 KB 00:02:20     P: 3694 D: 3694 M: 0
```

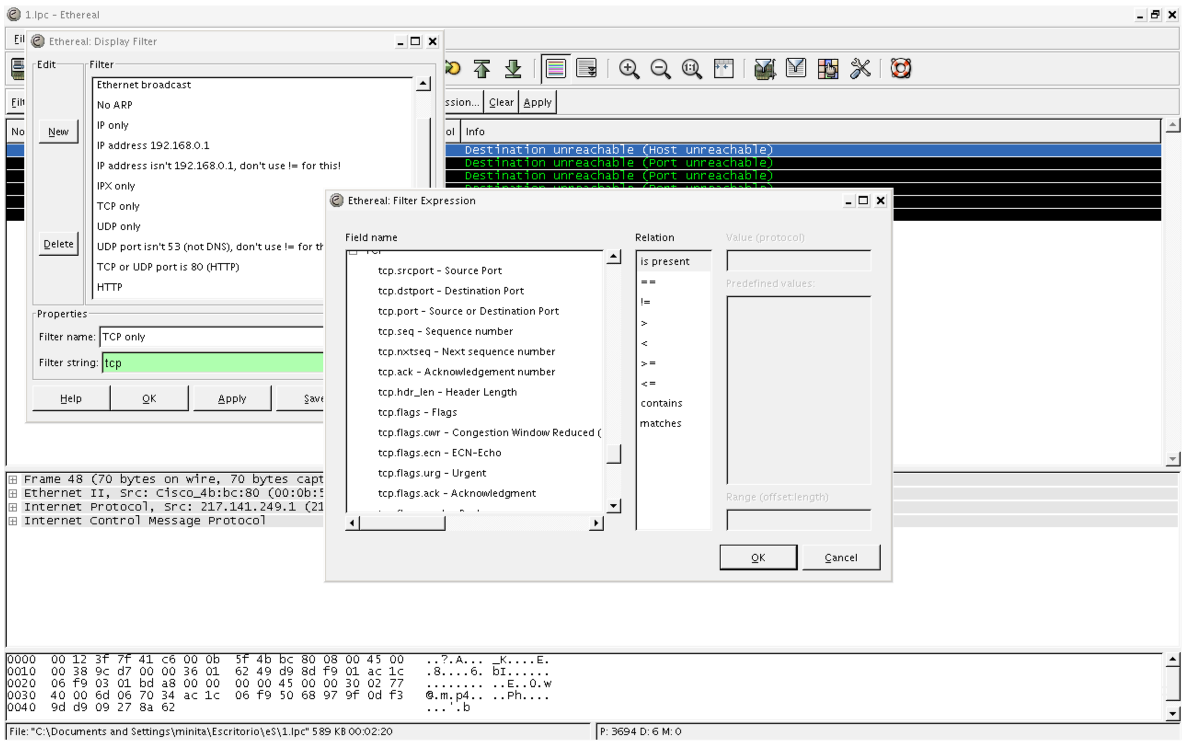
Una vez desplegado el archivo, es preciso utilizar una sintaxis de filtro diferentes. Ethereal es más enfocado al análisis, ya que es más fácil utilizar desde una Terminal remota programas que se ejecuten desde Terminal.

Por lo general se aplican filtros, con el fin de obtener la información a analizar:

Ejemplos de filtros en Ethereal

ip.src = = 10.10.10.3	Tráfico entrante o saliente de la dirección 10.10.10.3
icmp = = 0	Respuestas del eco ICMP
udp.dstport = = 9325	Puerto de destino 9325 UDP
tcp.flags.urg = = 1	El indicador URG de TCP esta activado

Para comprender el vasto número de filtros de Ethereal que estan disponibles, Analize → Display Filtres → Expression



Ethernet broadcast
No ARP
IP only
IP address 192.168.0.1
IP address isn't 192.168.0.1, don't use != for this!
IPX only
TCP only
UDP only
UDP port isn't 53 (not DNS), don't use != for th
TCP or UDP port is 80 (HTTP)
HTTP

Filter name: TCP only
Filter string: tcp

```
Destination unreachable (Host unreachable)
Destination unreachable (Port unreachable)
Destination unreachable (Port unreachable)
```

tcp.srcport - Source Port
tcp.dstport - Destination Port
tcp.port - Source or Destination Port
tcp.seq - Sequence number
tcp.nxtseq - Next sequence number
tcp.ack - Acknowledgement number
tcp.hdr_len - Header Length
tcp.flags - Flags
tcp.flags.cwr - Congestion Window Reduced (
tcp.flags.ecn - ECN-Echo
tcp.flags.urg - Urgent
tcp.flags.ack - Acknowledgment

is present
==
!=
>
<
>=
<=
contains
matches

```
⊞ Frame 48 (70 bytes on wire, 70 bytes capt
⊞ Ethernet II, Src: Cisco_4b:bc:80 (00:0b:5
⊞ Internet Protocol, Src: 217.141.249.1 (21
⊞ Internet Control Message Protocol
```

```
0000   00 12 3f 7f 41 c6 00 0b  5f 4b bc 80 08 00 45 00   ..?.A... _K....E.
0010   00 38 9c d7 00 00 36 01  62 49 d9 8d f9 01 ac 1c   .8....6. bI......
0020   06 f9 03 01 bd a8 00 00  00 00 45 00 00 30 02 77   ........ ..E..0.w
0030   40 00 6d 06 70 34 ac 1c  06 f9 50 68 97 9f 0d f3   @.m.p4.. ..Ph....
0040   9d d9 09 27 8a 62                                  ...'.b
```

File: "C:\Documents and Settings\minita\Escritorio\eS\1.lpc" 589 KB 00:02:20
P: 3694 D: 6 M: 0

Una vez que se encuentren los filtros y cuales son las opciones para utilizarlo, se puede ir directamente a: Edit → Find Packet

Por ejemplo: tcp.port = = 20

Si se desea utilizar toda la información de debe utilizar la opción clear antes de utilizar un filtro.

Ethereal es especialmente útil para visualizar todos los datos pertinentes de un determinado paquete.

Por último Ethereal también permite visualizar la lista de conversaciones TCP.

Statistics → Conversations

Address A	Address B	Packets	Bytes	Packets A->B	Bytes A->B	Pakets A<-B	Bytes A<--B
AppleCom_51:35:42	Broadcast	1	136	1	136	0	0

Ejercicios:

1. Realiza un ejercicio en la plataforma que elijas, en el que utilices los comandos descritos en este documento. Tanto para Tcpdump como Ethereal.

2. Identifica dentro de la información del archivo la conexión inicial de tres vías de TCP, y trata de describir los componentes de estos paquetes.

3. Utilizando **windump -?** ó **man tcpdump** elige un parámetro que no se mencione en el documento, utilízalo y da una breve explicación para que sirve.

4. Prueba al menos 3 filtros diferentes con Ethereal de la información capturada.

5. Da un ejemplo de un paquete de Voz o de un paquete de mensajero.

Práctica 8

SOCKETS EN JAVA

Los sockets son un sistema de comunicación entre procesos de diferentes máquinas de una red. Un socket es un punto de comunicación por el cual un proceso puede emitir o recibir información.
Fueron popularizados por Berckley Software Distribution, de la universidad norteamericana de Berkley. Los sockets han de ser capaces de utilizar el protocolo de streams TCP (Transfer Contro Protocol) y el de datagramas UDP (User Datagram Protocol).
Utilizan una serie de primitivas para establecer el punto de comunicación, para conectarse a una máquina remota en un determinado puerto que esté disponible, para escuchar en él, para leer o escribir y publicar información en él, y finalmente para desconectarse.
Con todas primitivas se puede crear un sistema de diálogo muy completo.

Tipos de sockets:
TCP sockets (En Java implementados en la clase Socket): Ofrecen una comunicación fiable y libre de errores, garantizando que los mensajes llegarán ordenados, sin duplicados y sin pérdidas. Antes de comenzar la transmisión necesitan una fase previa de establecimiento de la conexión.
UDP sockets (En Java implementados en la clase DatagramSocket): Ofrecen un servicio no fiable de comunicación. Los paquetes pueden llegar duplicados, desordenados, o pueden perderse sin llegar a su destino.
La comunicación es mucho más rápida que en los TCP Sockets, y no necesitan una fase de establecimiento de conexión.

Cliente / Servidor:
Normalmente la comunicación entre dos ordenadores se realiza mediante la filosofía Cliente-Servidor. El usuario Cliente desea obtener unos servicios de la máquina remota (Servidor). Para ello, el Servidor proporciona un puerto de

comunicaciones donde deben conectarse todos los Clientes que deseen obtener dicho servicio. Estableciendo un socket en la máquina local (Cliente) y otro en la máquina remota (Servidor) , y comunicándolos entre sí por el puerto proporcionado conseguimos establecer la vía de comunicación entre dos ordenadores interconectados por una red de comunicación.
El cliente es normalmente la parte que inicia la comunicación entre dos máquinas.
El servidor es normalmente la parte que acepta las peticiones de conexión.
Un cliente puede crear un socket para iniciar la comunicación con el servidor en el momento que desee.
El servidor debe estar siempre preparado para escuchar las posibles peticiones de conexión de los clientes.

TCP SOCKETS

La clase Socket
Constructores de sockets:

protected Socket()   Crea un socket sin ningún tipo de conexión
public Socket(String host,
            int port) throws UnknownHostException,
IOException   Crea un stream socket y lo conecta a un puerto de una máquina remota.
host: Nombre de la máquina remota.
port: Puerto de la máquina remota.
UnknownHostException, IOException: Si ocurre un error durante la conexión
public Socket(InetAddress address,
            int port) throws IOException   Crea un stream socket y lo conecta a un puerto de una dirección IP.
address: Dirección IP de la máquina remota.
port: Puerto de la máquina remota.
IOException: Si ocurre un error durante la conexión
public Socket(String host,
            int port,
            InetAddress localAddr,
            int localPort) throws IOException   Crea un socket y lo conecta a un puerto de una máquina remota. Además, el socket hará un bind() a la máquina y puerto local (enlazará a ambos).

host: Nombre de la máquina remota.
port: Puerto de la máquina remota.
localAddr: Dirección IP de la máquina local.
localPort: Puerto de la máquina local.
IOException: Si ocurre un error durante la conexión
public Socket(InetAddress address,
int port,
InetAddress localAddr,
int localPort) throws IOException Crea un socket y lo conecta a un puerto de una dirección IP. Además, el socket hará un bind() a la máquina y puerto local (enlazará a ambos).
address: Dirección IP de la máquina remota.
port: Puerto de la máquina remota.
localAddr: Dirección IP de la máquina local.
localPort: Puerto de la máquina local.
IOException: Si ocurre un error durante la conexión

Métodos:
public InetAddress getInetAddress() Devuelve la dirección IP a la cual está conectado el socket.
public InetAddress getLocalAddress() Devuelve la dirección local a la cual el socket está enlazado.
public int getPort() Devuelve el puerto remoto al cual está conectado el socket.
public int getLocalPort() Devuelve el puerto local al cual está enlazado el socket.
public InputStream getInputStream() throws IOException Devuelve un stream de entrada (lectura de bytes) para el socket.
IOException: Si ocurre un error durante la creación del stream de entrada.
public OutputStream getOutputStream() throws
IOException Devuelve un stream de salida (escritura de bytes) para el socket.
IOException: Si ocurre un error durante la creación del stream de salida.
public synchronized void close() throws IOException Cierra el socket.
IOException: Si ocurre un error al cerrar el socket.
public String toString() Convierte el socket en un string.
Excepciones:

BindException: Se produce cuando intentas construir un objeto Socket o ServerSocket en un puerto local que está en uso; o si no tienes suficientes privilegios para construirlo.
ConnectException: Se produce cuando una conexión es rechazada por el host remoto. Normalmente porque el host está ocupado o no está escuchando por ese puerto.
NoRouteToHostException: Indica que a la conexión se le ha acabado el tiempo de establecimiento de la conexión.
SocketException: Si ocurre un error en el socket.
UnknownHostException: Si no encuentra la maquina con la cual queremos establecer la conexión.
A continuación se presenta un programa que establece un pequeño diálogo entre un programa servidor y sus clientes, que intercambiarán cadenas de información.
Programa Servidor
El programa servidor se instala en un puerto determinado, a la espera de conexiones, a las que tratará mediante un segundo socket.
Cada vez que se presenta un cliente, le saluda con una frase "Hola cliente N".
Este servidor sólo atenderá hasta tres clientes, y después finalizará su ejecución, pero es habitual utilizar bucles infinitos (while(true)) en los servidores, para que atiendan llamadas continuamente.
Tras atender cuatro clientes, el servidor deja de ofrecer su servicio:
Utiliza un objeto de la clase ServerSocket (skServidor), que sirve para esperar las conexiones en un puerto determinado (PUERTO), y un objeto de la clase Socket (skCliente) que sirve para gestionar una conexión con cada cliente (línea 7).
Mediante un bucle for y la variable numCli se restringe el número de clientes a tres, con lo que cada vez que en el puerto de este servidor aparezca un cliente, se atiende y se incrementa el contador (línea 9).
Para atender a los clientes se utiliza la primitiva accept() de la clase ServerSocket, que es una rutina que crea un nuevo Socket (skCliente) para atender a un cliente que se ha conectado a ese servidor (línea 10).
Se asocia al socket creado (skCliente) un flujo (flujo) de salida DataOutputStream de escritura secuencial, en el que se escribe el mensaje a enviar al cliente (línea 13).
El tratamiento de las excepciones es muy reducido en nuestro ejemplo, tan solo se captura e imprime el mensaje que incluye la excepción mediante getMessage() (línea 18).

```
1.	import java.io.* ;
```

```
2.    import java.net.* ;
3.    class Servidor {
4.    static final int PUERTO=5000;
5.    public Servidor( ) {
6.    try {
7.    ServerSocket skServidor = new ServerSocket( PUERTO );
8.    System.out.println("Escucho el puerto " + PUERTO );
9.    for ( int numCli = 0; numCli < 3; numCli++; ) {
10.   Socket skCliente = skServidor.accept(); // Crea objeto
11.   System.out.println("Sirvo al cliente " + numCli);
12.   OutputStream aux = skCliente.getOutputStream();
13.   DataOutputStream flujo= new DataOutputStream( aux );
14.   flujo.writeUTF( "Hola cliente " + numCli );
15.   skCliente.close();
16.   }
17.   System.out.println("Demasiados clientes por hoy");
18.   } catch( Exception e ) {
19.   System.out.println( e.getMessage() );
20.   }
21.   }
22.   public static void main( String[] arg ) {
23.   new Servidor();
24.   }
25.   }
```

Programa Cliente

El programa cliente se conecta a un servidor indicando el nombre de la máquina y el número puerto (tipo de servicio que solicita) en el que el servidor está instalado.

Una vez conectado, lee una cadena del servidor y la escribe en la pantalla:

En primer lugar se crea el socket denominado skCliente, al que se le especifican el nombre de host (HOST) y el número de puerto (PORT) en este ejemplo constantes (linea8).

Luego se asocia el flujo de datos de dicho socket (obtenido mediante getInputStream)) (linea 9), que es asociado a un flujo (flujo) DataInputStream de lectura secuencial (linea 10). De dicho flujo capturamos una cadena (readUTF()), y la imprimimos por pantalla (System.out) (linea 11).

El socket se cierra, una vez finalizadas las operaciones, mediante el método close()(linea 12).

Debe observarse que se realiza una gestión de excepción para capturar los posibles fallos tanto de los flujos de datos como del socket (linea 11).

```
1.      import java.io.*;
2.      import java.net.*;
3.      class Cliente {
4.        static final String HOST = "localhost";
5.        static final int PUERTO=5000;
6.       public Cliente( ) {
7.          try{
8.            Socket skCliente = new Socket( HOST , Puerto );
9.            InputStream aux = skCliente.getInputStream();
10.           DataInputStream flujo = new DataInputStream( aux );
11.           System.out.println( flujo.readUTF() );
12.           skCliente.close();
13.         } catch( Exception e ) {
14.           System.out.println( e.getMessage() );
15.         }
16.       }
17.       public static void main( String[] arg ) {
18.         new Cliente();
19.       }
20.   }
```

Ejecución

Aunque la ejecución de los sockets está diseñada para trabajar con ordenadores en red, también se puede probar el correcto funcionamiento de un programa de sockets en una misma máquina.

Para ellos se ha de colocar el servidor en una ventana, obteniendo lo siguiente:

```
>java Servidor
Escucho el puerto 5000
```

En otra ventana se lanza varias veces el programa cliente, obteniendo:

```
>java Cliente
Hola cliente 1
>java cliente
Hola cliente 2
>java cliente
Hola cliente 3
>java cliente
connection refused: no further information
```

Mientras tanto en la ventana del servidor se ha impreso:

```
Sirvo al cliente 1
Sirvo al cliente 2
Sirvo al cliente 3
Demasiados clientes por hoy
```

Cuando se lanza el cuarto cliente, el servidor ya ha cortado la conexión, con lo que se lanza una excepción.

FTP

Es uno de los diversos protocolos de la red Internet, concretamente significa File Transfer Protocol (Protocolo de Transferencia de Ficheros) y es el ideal para transferir grandes bloques de datos por la red. Su comportamiento está definido por la recomendación RFC 959.

Por defecto utiliza los puertos 20 y 21. El puerto 20 es el utilizado para el flujo de datos entre el cliente y el servidor y el puerto 21 para el flujo de control, es decir, para enviar las órdenes del cliente al servidor. Mientras se transfieren datos a través del flujo de datos, el flujo de control permanece en espera. Esto puede causar problemas en el caso de transferencias de datos muy grandes realizadas a través de cortafuegos que interrumpan sesiones después de periodos largos en espera. El archivo puede que se haya transferido con éxito, pero el cortafuegos puede desconectar la sesión de control, por lo que se genera un error.

Ejercicio

A continuación se muestra un programa RemoteFileServer.java que implementa un servidor ftp en el puerto 3000 permitiendo que se descargue un archivo de texto.

Programa Servidor

```
import java.io.*;
import java.net.*;
public class RemoteFileServer
{
	protected int listenPort = 3000;

	public static void main(String[] args)
	{
		RemoteFileServer server = new RemoteFileServer();
		server.acceptConnections();
	}
	public void acceptConnections()
	{
		try
		{
			ServerSocket server = new ServerSocket(listenPort);
			Socket incomingConnection = null;
			while (true)
			{
				incomingConnection = server.accept();
				handleConnection(incomingConnection);
			}
		}
		catch (BindException e)
		{
			System.out.println("Unable to bind to port " + listenPort);
		}
		catch (IOException e)
		{
			System.out.println("Unable to instantiate a ServerSocket on
port: " + listenPort);
		}
	}

	public void handleConnection(Socket incomingConnection)
	{
		try
		{
			OutputStream outputToSocket =
incomingConnection.getOutputStream();
			InputStream inputFromSocket =
incomingConnection.getInputStream();
			BufferedReader streamReader = new BufferedReader(new
inputStreamReader(inputFromSocket));
			FileReader fileReader = new FileReader(new
File(streamReader.readLine()));
			BufferedReader bufferedFileReader = new
BufferedReader(fileReader);
```

```
                        PrintWriter             streamWriter            =           new
PrintWriter(incomingConnection.getOutputStream());
                        String line = null;
                        while ((line = bufferedFileReader.readLine()) != null)
                        {
                                streamWriter.println(line + "server");
                        }
                        fileReader.close();
                        streamWriter.close();
                        streamReader.close();
                }
                catch (Exception e)
                {
                        System.out.println("Error handling a client: " + e);
                }
        }


}
```

El siguiente programa contiene el código del cliente que debe conectarse al servidor y descargar el archivo:
ejemplo:
Java RemoteFileClient practica.txt

El archivo practica.txt está en el servidor y se debe pasar al cliente.

Completa el siguiente archivo creando o modificando la clase getFile para que transfiera el archivo del servidor al cliente

Programa cliente

```
import java.io.*;
import java.net.*;
public class RemoteFileClient
{
        protected String hostIp;
        protected int hostPort;
        protected BufferedReader socketReader;
```

```
	protected PrintWriter socketWriter;


	public static void main(String[] args)
	{
		RemoteFileClient remoteFileClient = new
RemoteFileClient("127.0.0.1", 3000);
		remoteFileClient.setUpConnection();
		String fileContents = remoteFileClient.getFile(args[0]);
		remoteFileClient.tearDownConnection();
		System.out.println(fileContents);
	}

	public void setUpConnection()
	{
	try
		{
			Socket client = new Socket(hostIp, hostPort);
			socketReader = new BufferedReader(new
InputStreamReader(client.getInputStream()));
			socketWriter = new
PrintWriter(client.getOutputStream());
		}
	catch (UnknownHostException e)
		{
			System.out.println("Error setting up socket connection:
unknown host at " + hostIp );
		}
	catch (IOException e)
		{
			System.out.println("Error setting up socket connection: " +
e);
		}
	}

	public String getFile(String fileNameToGet)
	{
        StringBuffer fileLines = new StringBuffer();
        try
                {
                . // leer el archivo del servidor
                .// Escribir el archivo en el cliente
                .
                }

        }
        catch (IOException e)
                {
                        .
                        .       // Excepción por si no encuentra el archivo
                        .
                }
        return fileLines.toString();
        }

        public void tearDownConnection()
        {
        try
                {
                        socketWriter.close();
                        socketReader.close();
                }
        catch (IOException e)
                {
                        System.out.println("Error tearing down socket connection:
" + e);
                }
        }

        public RemoteFileClient(String aHostIp, int aHostPort)
        {
                hostIp = aHostIp;
                hostPort = aHostPort;
        }

}
```

Práctica 9

CONFIGURACIÓN DE UNA NAT CON VLAN POR MEDIO DE UNA INTERFAZ GRÁFICA

INTRODUCCIÓN

Router y Administrador del dispositivo de seguridad de Cisco (SDM) es una herramienta de gestión de dispositivos de fácil utilización que le permite configurar las funciones de seguridad del IOS de Cisco y las conexiones de la red a través de una interfaz gráfica de usuario intuitiva basada en la web. Esta Guía de inicio rápido muestra cómo conectar su equipo al router y comenzar a utilizar el SDM.

SDM se ejecuta con Internet Explorer versión 5.5 o superior y con Netscape 7.1, en un equipo que opere con Microsoft Windows XP, Windows 2000, Windows 2003, Windows ME, Windows NT 4.0 (con Service Pack 4), o Windows 98. SDM admite el plugin de Java versión 1.4.2_05 y superior.

Objetivos:

- Aprender como entrar al modo gráfico
- Aprender a configurar el router con la interfaz
- Aprender a usar algunas herramientas del modo gráfico

Para realizar la configuración necesitaremos:

- ✓ 2 Routers
- ✓ Una hyperterminal conectada a una PC y a un router
- ✓ Una PC donde podamos configurar la IP de la tarjeta de red Ethernet
- ✓ Tener los 2 router sin ninguna configuración.

Procedimiento:

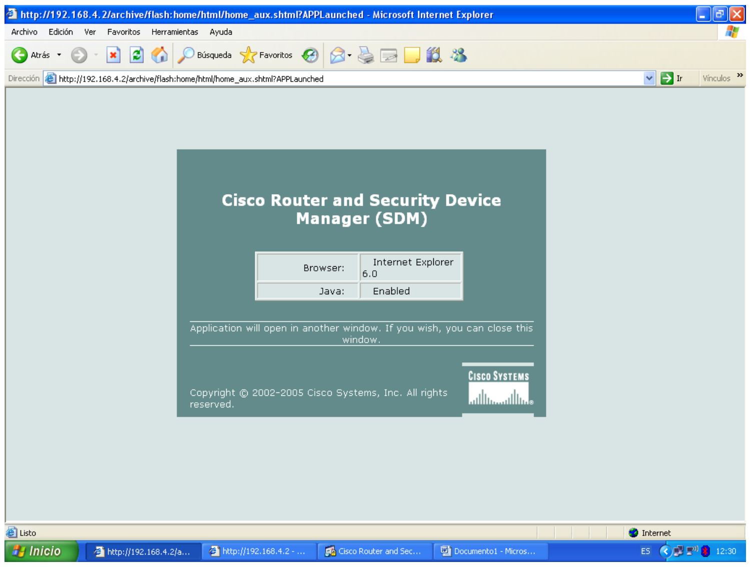
Por medio de la hyperterminal entrar y borrar la configuración actual como se ha realizado a lo largo de las practicas anteriores realcionadas, posteriormente asignaremos una IP a la interfaz ethernet 0/0 (esta puede ser cualquiera en este caso usamos la 192.168.4.2).
Repetimos el proceso anterior para el router2 (usamos la ip 192.168.5.5)

-- Para ver otra forma de conectarse al router en modo grafico verificar:

http://www.exio.com/en/US/products/sw/secursw/ps5318/products_quick_start09186a008043fd03.html ---

Conectemos la interfaz 0/0 del router con la tarjeta de red de nuestra PC, esta PC la debemos configurar con una IP del mismo segmento que el del router.

Ya realizado lo anterior abrimos un navegador de Internet y accesamos a la IP del router (192.168.4.2), donde aparecera la siguiente ventana:


http://192.168.4.2/archive/flash:home/html/home_aux.shtml?APPLaunched - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección http://192.168.4.2/archive/flash:home/html/home_aux.shtml?APPLaunched
Ir Vínculos
Listo
Internet
Inicio
http://192.168.4.2/a...
http://192.168.4.2 - ...
Cisco Router and Sec...
Documento1 - Micros...
ES 12:30

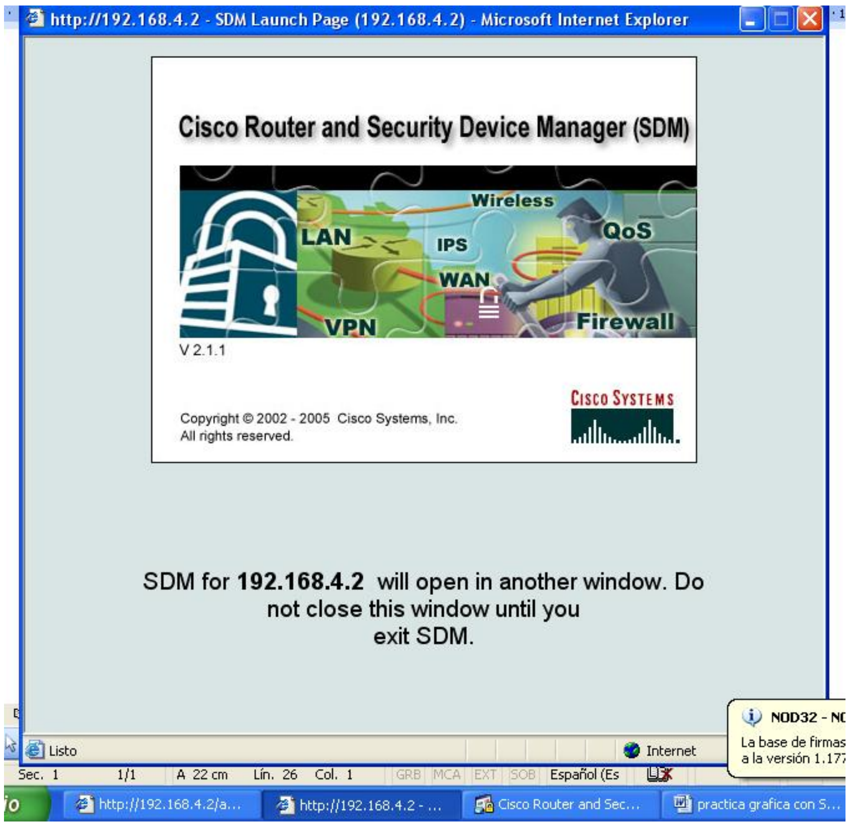
Cisco Router and Security Device Manager (SDM)

Browser:	Internet Explorer 6.0
Java:	Enabled

Application will open in another window. If you wish, you can close this window.

Copyright © 2002-2005 Cisco Systems, Inc. All rights reserved.

Posteriormente se comienza a cargar la aplicación:

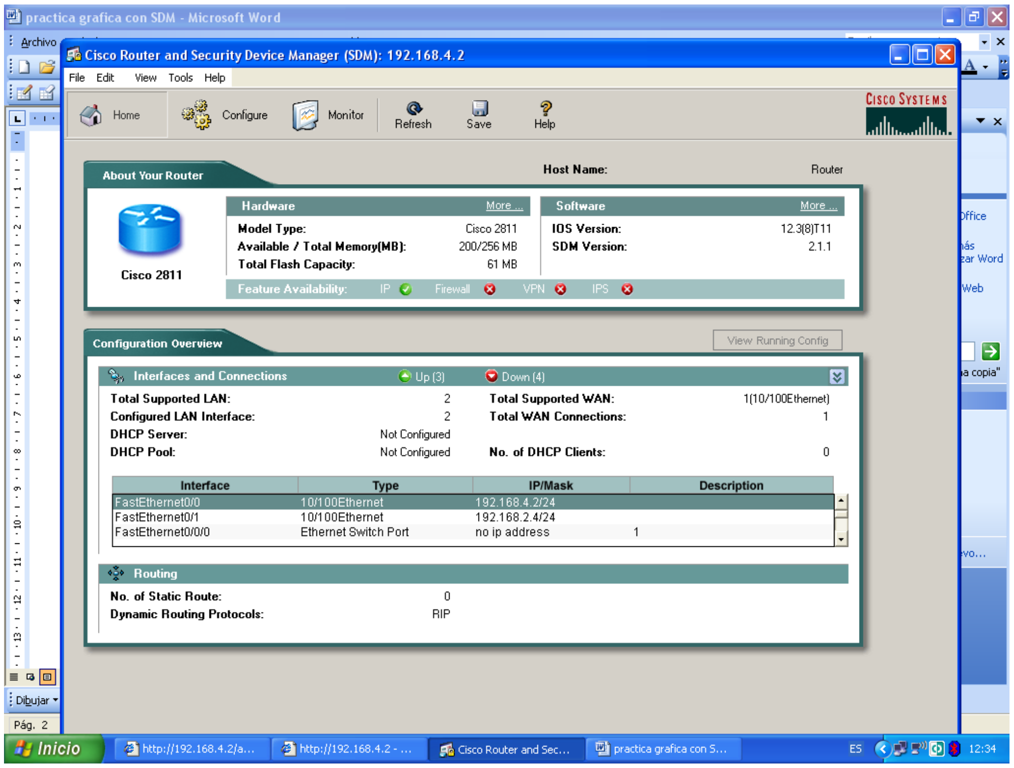

http://192.168.4.2 - SDM Launch Page (192.168.4.2) - Microsoft Internet Explorer
Cisco Router and Security Device Manager (SDM)
Wireless
LAN
IPS
QoS
WAN
VPN
Firewall
V 2.1.1
Cisco Systems
Listo
Internet
NOD32 - NO
Sec. 1 1/1 A 22 cm Lín. 26 Col. 1 GRB MCA EXT SOB Español (Es
http://192.168.4.2/a...
http://192.168.4.2 - ...
Cisco Router and Sec...
practica grafica con S...

Copyright © 2002 - 2005 Cisco Systems, Inc.
All rights reserved.

La base de firmas
a la versión 1.177

SDM for **192.168.4.2** will open in another window. Do not close this window until you exit SDM.

Al obtener esta ventana, nos iremos a configurar dando un clic en **configure**



About Your Router

Host Name: Router

Hardware

Model Type:	Cisco 2811
Available / Total Memory(MB):	200/256 MB
Total Flash Capacity:	61 MB

Software

IOS Version:	12.3(8)T11
SDM Version:	2.1.1

Feature Availability: IP ✓ Firewall ✗ VPN ✗ IPS ✗

Configuration Overview

Interfaces and Connections — Up (3) — Down (4)

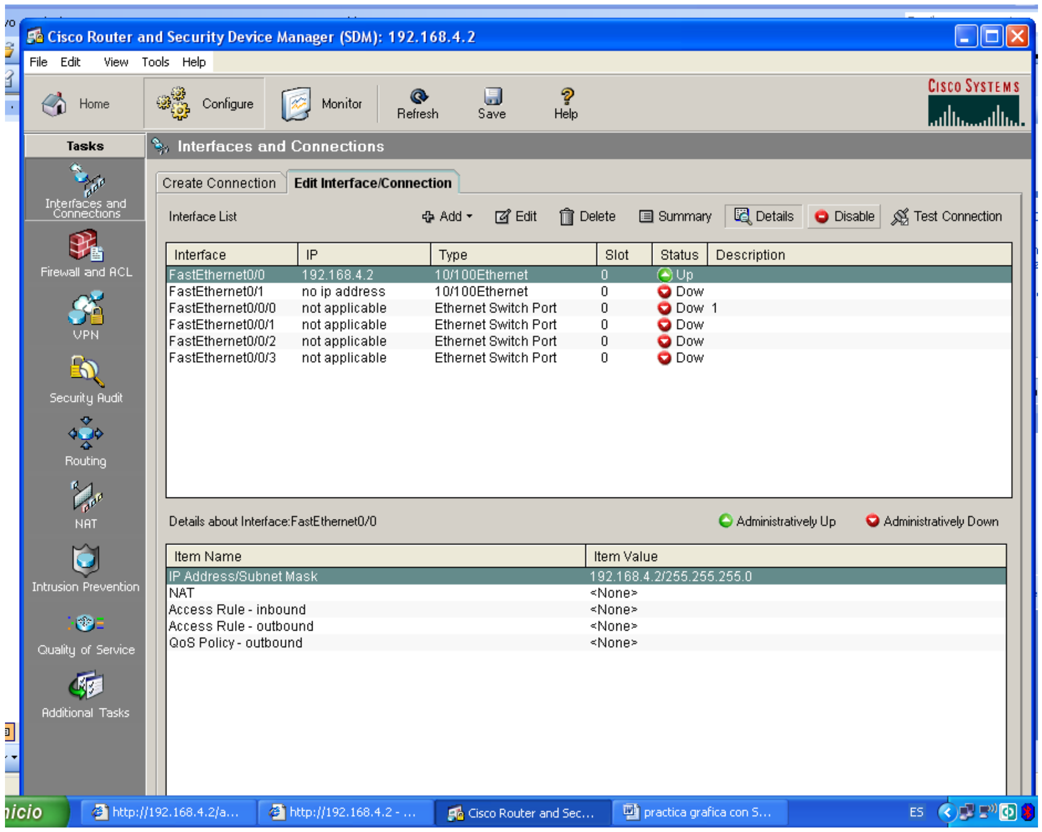
Total Supported LAN:	2	Total Supported WAN:	1(10/100Ethernet)
Configured LAN Interface:	2	Total WAN Connections:	1
DHCP Server:	Not Configured		
DHCP Pool:	Not Configured	No. of DHCP Clients:	0

Interface	Type	IP/Mask	Description
FastEthernet0/0	10/100Ethernet	192.168.4.2/24	
FastEthernet0/1	10/100Ethernet	192.168.2.4/24	
FastEthernet0/0/0	Ethernet Switch Port	no ip address	1

Routing

No. of Static Route:	0
Dynamic Routing Protocols:	RIP

Aqui nos mostrara las conexiones existentes obviamente se encuentra configurada la conexión realizada desde la Terminal.



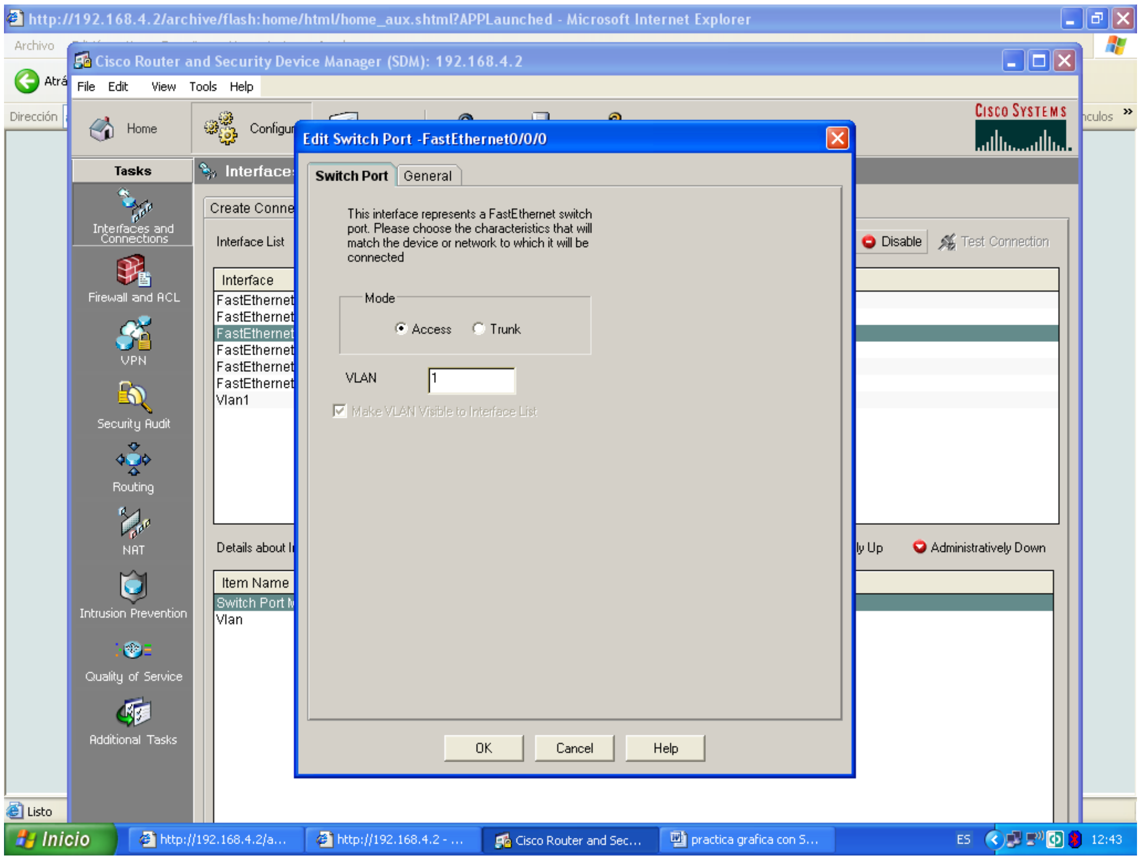
Interfaces and Connections

Interface	IP	Type	Slot	Status	Description
FastEthernet0/0	192.168.4.2	10/100Ethernet	0	Up	
FastEthernet0/1	no ip address	10/100Ethernet	0	Dow	
FastEthernet0/0/0	not applicable	Ethernet Switch Port	0	Dow	1
FastEthernet0/0/1	not applicable	Ethernet Switch Port	0	Dow	
FastEthernet0/0/2	not applicable	Ethernet Switch Port	0	Dow	
FastEthernet0/0/3	not applicable	Ethernet Switch Port	0	Dow	

Details about Interface:FastEthernet0/0

Item Name	Item Value
IP Address/Subnet Mask	192.168.4.2/255.255.255.0
NAT	<None>
Access Rule - inbound	<None>
Access Rule - outbound	<None>
QoS Policy - outbound	<None>

Configuraremos una VLAN dando doble clic sobre un puerto del ethernet switch port 0/0/0 y saldra la siguiente ventana


http://192.168.4.2/archive/flash:home/html/home_aux.shtml?APPLaunched - Microsoft Internet Explorer
Cisco Router and Security Device Manager (SDM): 192.168.4.2
Edit Switch Port -FastEthernet0/0/0
Switch Port
General
OK
Cancel
Help

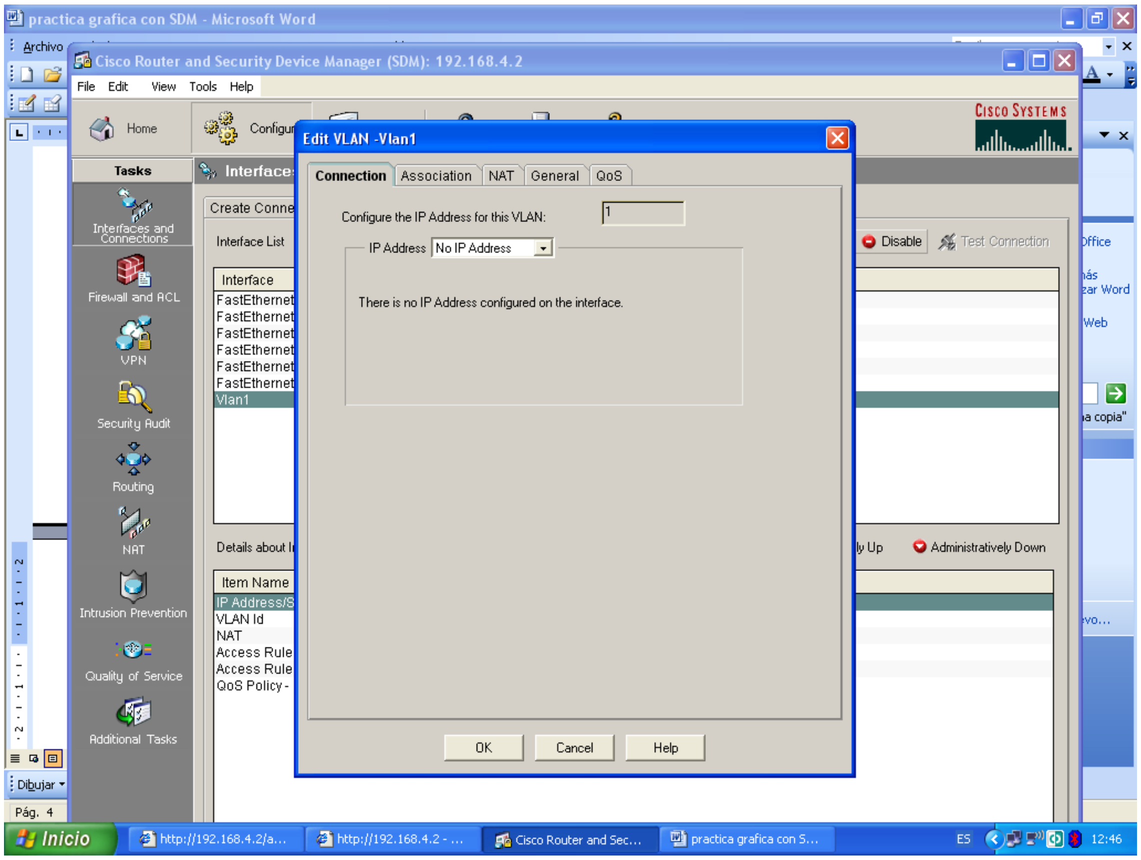
This interface represents a FastEthernet switch port. Please choose the characteristics that will match the device or network to which it will be connected

Mode: [✓ Access] Trunk

VLAN: 1

[✓ Make VLAN Visible to Interface List]

Damos ok, y salimos a la ventana anterior **(interfaces connection)** pero ya dada de alta la VLAN escogida, ahora en la lista ya tenemos a una VLAN, para configurarla hacemos doble clic sobre esta y saldrá la siguiente ventana:



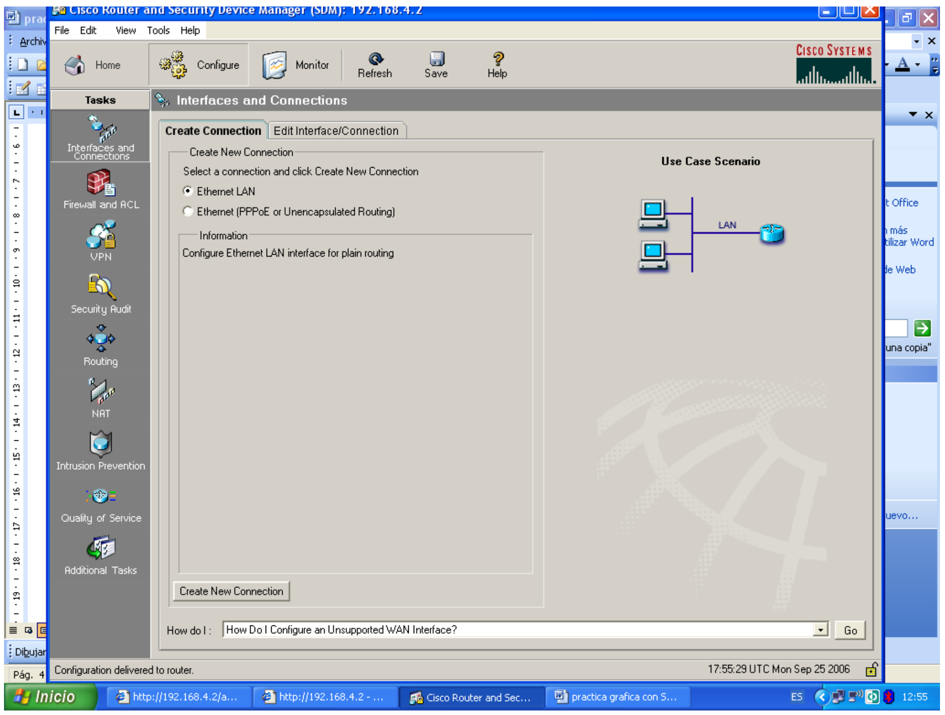
Configure the IP Address for this VLAN: 1

IP Address: No IP Address

There is no IP Address configured on the interface.

Aquí escogeremos la ip que tendra la VLAN (se uso la ip 192.168.3.2), ya asignada la ip, configuraremos en la opción NAT la VLAN, damos clic en la opcion NAT y damos como opcion **inside,** posteriormente damos ok y el router realizara las actualizaciones y volveremos a la pantalla de conexiones.

Ahora crearemos una conexión para el otro puerto del router para que nos sirva como salida de la red NAT, damos un clic en **created connection**, nos saldra la siguiente pantalla:



Interfaces and Connections

Create New Connection

Select a connection and click Create New Connection

(●) Ethernet LAN

( ) Ethernet (PPPoE or Unencapsulated Routing)

Information

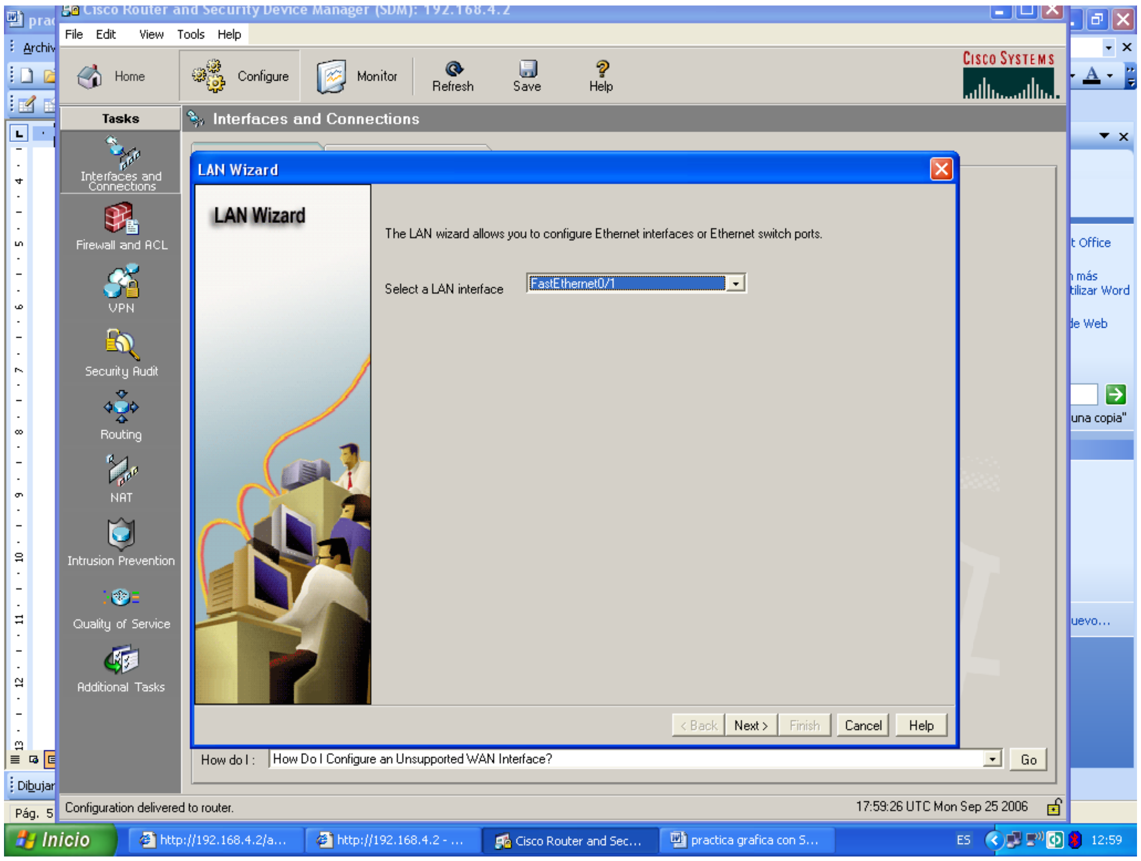
Configure Ethernet LAN interface for plain routing

How do I : How Do I Configure an Unsupported WAN Interface?

Configuration delivered to router.

17:55:29 UTC Mon Sep 25 2006

En esta ventana dejamos activada la opción de hacer una conexión del tipo ethernet LAN y posteriormente damos un clic en **create new connection**, saldra un asistente que nos preguntara sobre la interfaz a configurar(0/1):



LAN Wizard

The LAN wizard allows you to configure Ethernet interfaces or Ethernet switch ports.

Select a LAN interface: FastEthernet0/1

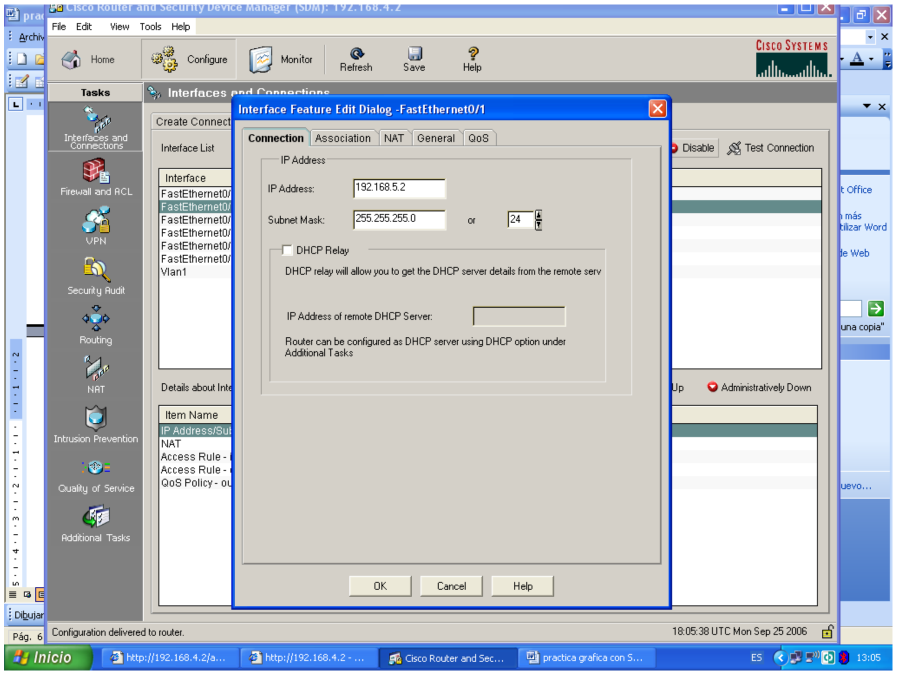
How do I : How Do I Configure an Unsupported WAN Interface?

Configuration delivered to router.

17:59:26 UTC Mon Sep 25 2006

Nos preguntará sobre la ip (usamos la ip 192.168.5.2) y la mascara de la interfaz, y si sobre queremos activar un DHCP a lo cual le decimos que no, al final nos da un resumen de lo que se realizo, le damos finalizar y el router actualizara cambios .

Al regresar a la pantalla donde se visualizan las conexiones, podemos ver que tenemos dada de alta la interfaz configurada, damos doble clic sobre de ella y saldra la siguiente ventana:

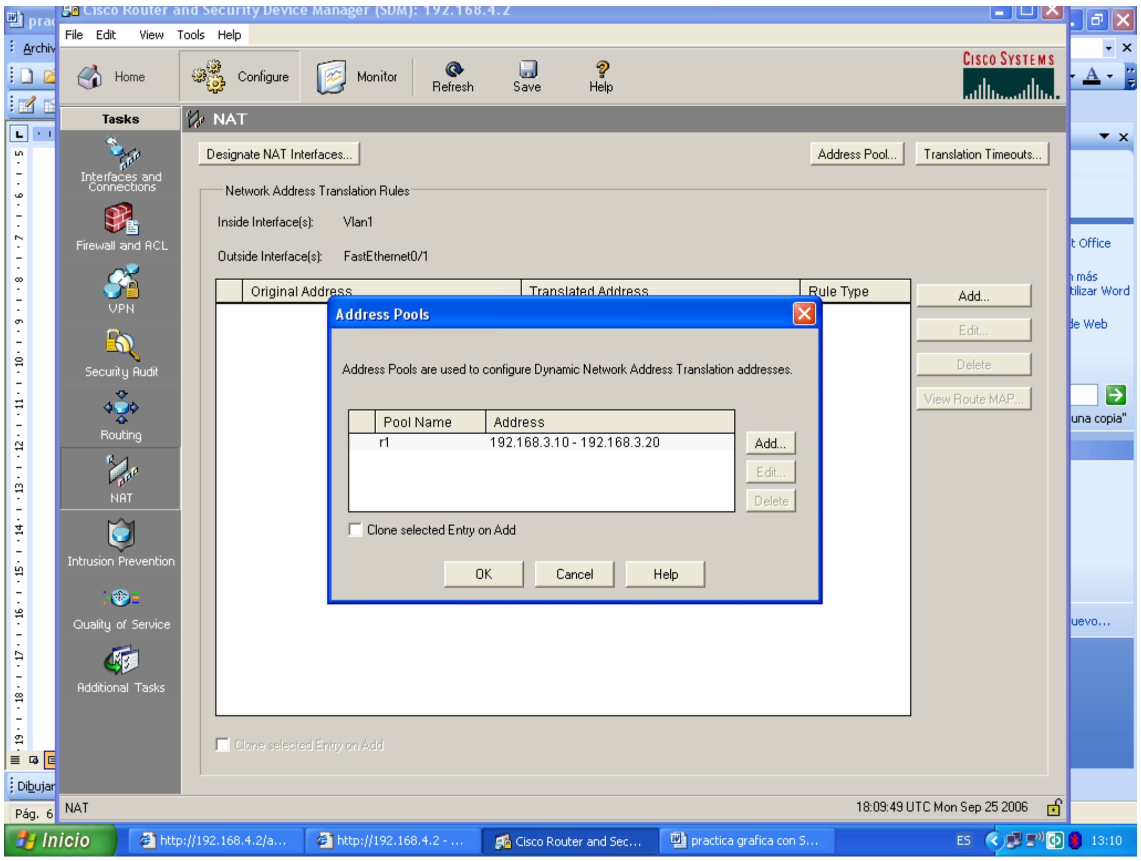


IP Address: 192.168.5.2

Subnet Mask: 255.255.255.0 or 24

Donde aparece la configuración realizada, para dar a esta red la salida de la NAT nos vamos a la opción NAT y escogemos como outside, damos ok para que el router actualice y regresamos.

Ahora asignaremos un pool de direcciones para la NAT, en el menú del lado izquierdo damos clic donde dice NAT y posteriormente nos vamos a la opcion **addres pool** en donde adicionamos el numero de IPs que se quieran asignar a esta red (asignamos el siguiente intervalo 192.168.3.10-192.168.3.20)



NAT

Network Address Translation Rules

Inside Interface(s): Vlan1

Outside Interface(s): FastEthernet0/1

Original Address	Translated Address	Rule Type

Address Pools

Address Pools are used to configure Dynamic Network Address Translation addresses.

Pool Name	Address
r1	192.168.3.10 - 192.168.3.20

Clone selected Entry on Add

18:09:49 UTC Mon Sep 25 2006

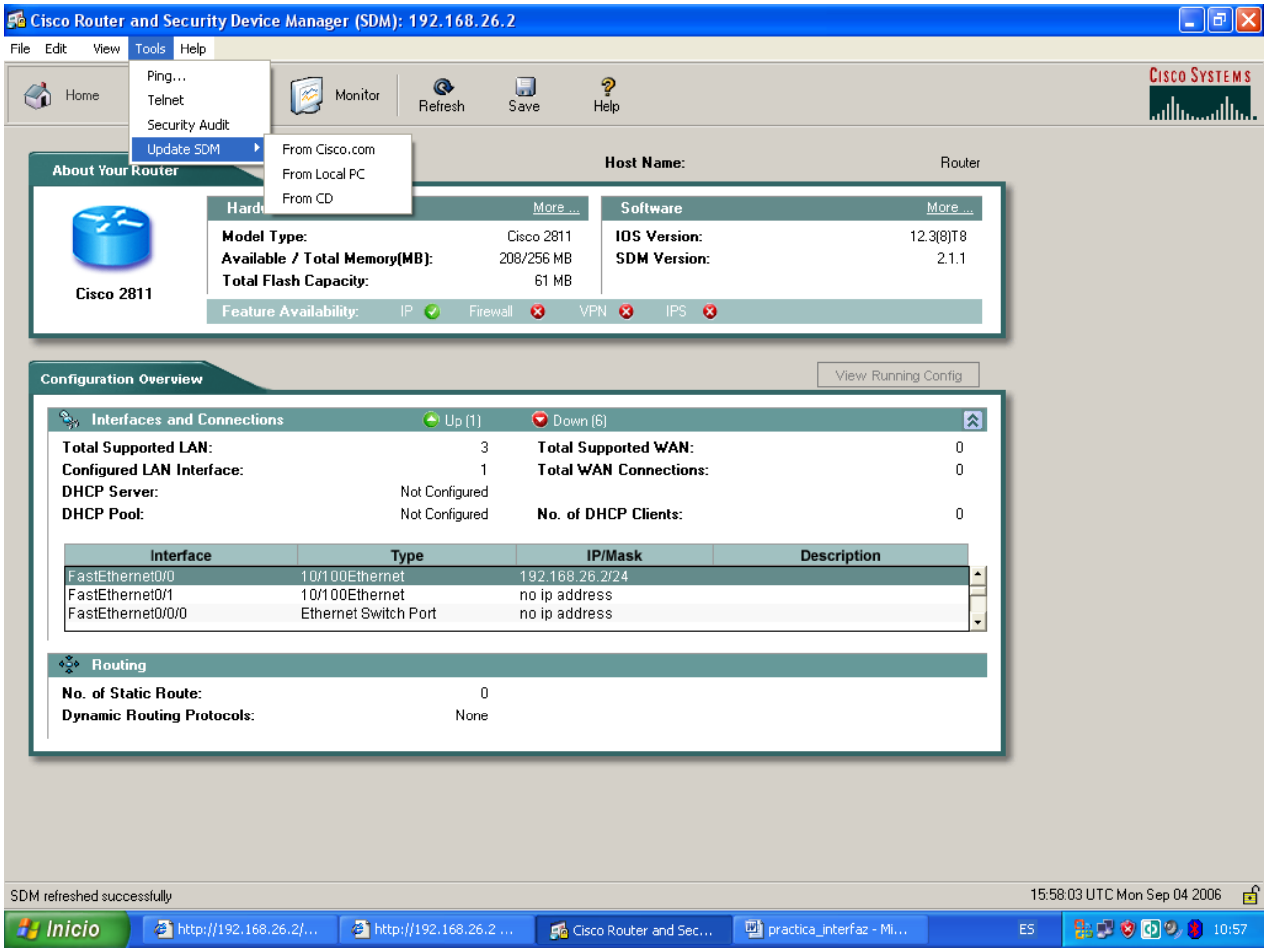
Damos ok y dejamos que el router actualice cambios

Para probar conectividad conectamos los routers, del router configurado en modo grafico, conectar la interfaz 0/0 a la interfaz 0/0 del router2.

Ahora utilizaremos algunas herramientas para probar conectividad y ejecutar comandos:

En el menú superior tenemos la opción Tools:

De aquí podemos usar el comando Ping para probar conectividad, y usar una Terminal de acceso remoto Telnet



Host Name: Router

Hardware	
Model Type:	Cisco 2811
Available / Total Memory(MB):	208/256 MB
Total Flash Capacity:	61 MB

Software	
IOS Version:	12.3(8)T8
SDM Version:	2.1.1

Feature Availability: IP ✓ Firewall ✗ VPN ✗ IPS ✗

Interfaces and Connections — Up (1) — Down (6)

Total Supported LAN:	3	Total Supported WAN:	0
Configured LAN Interface:	1	Total WAN Connections:	0
DHCP Server:	Not Configured		
DHCP Pool:	Not Configured	No. of DHCP Clients:	0

Interface	Type	IP/Mask	Description
FastEthernet0/0	10/100Ethernet	192.168.26.2/24	
FastEthernet0/1	10/100Ethernet	no ip address	
FastEthernet0/0/0	Ethernet Switch Port	no ip address	

Routing

No. of Static Route:	0
Dynamic Routing Protocols:	None

15:58:03 UTC Mon Sep 04 2006

En el menú superior esta la opción **view** de donde podemos ejecutar algunos comandos por medio de las opciones **Runing config y Show commands**, para verificar configuraciones como si estuviéramos en modo texto.

También tenemos la opción de salvar la configuración del router a la PC y de cargar una configuración de inicio, o en su defecto borrar la configuración para dejar en blanco la configuración, esto lo podemos realizar desde el menú superior en la opcion **file.**

Realizar la ejecución de las opciones descritas anteriormente para probar el funcionamiento de la configuración realizada, y posteriormente ejecutar comandos para verificar el estado de la configuración y sus formas de respaldo.

Práctica 10

Crear una red virtual utilizando Linux

Antecedentes

En esta práctica se requiere.

- Dos Pc. con sistema operativo Linux con una versión de kernel 2.4 o superior
- Tener instalados los paquetes openvpn y openssl.
- Tener acceso en ambas máquina a una red publica común (Internet)

La VPN es una tecnología de red que permite una extensión de la red local sobre una red pública o no controlada, como por ejemplo Internet.

El ejemplo más común es la posibilidad de conectar dos o más sucursales de una empresa utilizando como vínculo Internet, permitir a los miembros del equipo de soporte técnico la conexión desde su casa al centro de cómputo, o que un usuario pueda acceder a su equipo doméstico desde un sitio remoto, como por ejemplo un hotel. Todo esto utilizando la infraestructura de Internet.

Para hacerlo posible de manera segura es necesario proveer los medios para garantizar la autenticación, integridad y confidencialidad de toda la comunicación:

- Autenticación y autorización: Usuario/equipo y qué nivel de acceso debe tener.
- Integridad: La garantía de que los datos enviados no han sido alterados.
- Confidencialidad: Dado que los datos viajan a través de un medio potencialmente hostil, los mismos son susceptibles de interceptación, por lo que es fundamental el cifrado de los mismos.

El tunneling básicamente, consiste en abrir conexiones entre dos máquinas por medio de un protocolo seguro, como puede ser SSH, a través de las cuales realizaremos las transferencias inseguras, que pasarán de este modo a ser seguras. De esta analogía viene el nombre de la técnica, siendo la conexión segura (en este caso de ssh) el túnel por el cual enviamos nuestros datos para que nadie más aparte de los interlocutores que se sitúan a cada extremo del túnel, pueda ver dichos datos.

Utilizando un túnel entre los dos equipos Linux se obtiene la VPN.

La VPN que se planea que se monte se describe en la figura 1.

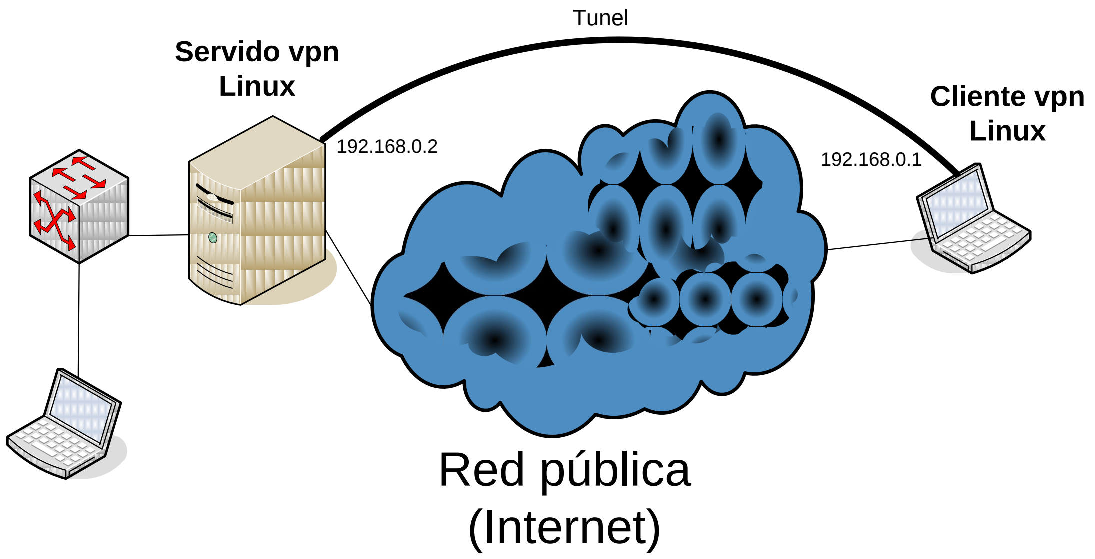


Figura 1. Configuración de Servidor a cliente.

Desarrollo.

Se supondrá que ya se tiene instalado los Linux, los paquetes y montada la infraestructura.
Para ambos equipos eiiciar sesión de *root* e ingresar al directorio */etc/openvpn*, si no existe crearlo.

```
        root# cd /etc/openvpn
```

Del lado del servidor se debe generar una llave común y agregar el archivo de configuración "*openvpn.conf*", se debe ejecutar el comando *openvpn* para generar la llave.

```
root# openvpn --genkey --secret static.key
root# vi openvpn.conf
```

```
        # Servidor
        # Puerto a conectarse
        port 1194
        # Protocolo de comunicación
        proto udp
        dev tun

        # Se agregan las direcciones con ifconfig, primero
        # la del servidor y luego la del receptor
        # En el receptor  la inversa del lado receptor
        ifconfig 192.168.0.110 192.168.0.111

        # Ubicación de la llave
        secret /etc/openvpn/static.key

        comp-lzo
        ping-timer-rem
        persist-tun
        persist-key
        keepalive 10 60

        user nobody
        group nobody
```

Para concluir se tiene que levantar el servicio.

```
root# /etc/init.d/openvpn start
```

En el lado del cliente se debe traer la llave, realizar la configuración y levantar el servicio.

```
root# scp root@servidor:/etc/openvpn/static.key /etc/openvpn/
root# vi openvpn.conf
```

```
        # Cliente
        dev tun
        port 1194
        # Se agregan las direcciones con ifconfig, primero
        # la del servidor y luego la del receptor
        # En el servidor la inversa
        ifconfig 192.168.0.111 192.168.0.110
        # Ubicación de la llave
        secret /etc/openvpn/static.key

        comp-lzo
        ping-timer-rem
        persist-tun
        persist-key
        keepalive 10 60
        user nobody
        group nogroup

        # A quién me conecto
        remote 132.248.237.221
```

Para concluir se tiene que levantar el servicio.

```
root# /etc/init.d/openvpn start
```

En caso de no tener en archivo `/etc/init.d/openvpn` se puede crear de la siguiente manera.

```
root# vi /usr/local/etc/rc.d/openvpn.sh
```

```
#! /bin/sh
case x$1 in
xstart)
/usr/local/sbin/openvpn   --cd   /usr/local/etc/openvpn   --config
openvpn.conf;;
xstop)
#echo -n ' if_tap' ;
killall -9 openvpn;;
*) echo >&2 "Usage: $0 {start|stop}"
esac
```

Pruebas de funcionamiento.

- Se realizar ping de servidor al cliente.
- Usar un servicio en una de las máquinas y verificar que responda del otro lado del tunel.

Práctica 11

Diseño e implantación de un protocolo de comunicación entre procesos

I. Motivación, objetivos y planeación de la práctica

Los mecanismos de comunicación entre procesos hacen posible la interacción entre los diversos componentes de una arquitectura de red. El objetivo fundamental de este trabajo consiste en familiarizarse con un método de comunicación entre procesos y usarlo para implantar un protocolo de comunicaciones que utilice encapsulado, fragmentación, ensamble y control de flujo. Objetivos específicos: a) utilizar *pipes* para realizar comunicaciones entre procesos y c) desarrollo e implantación de un protocolo de comunicaciones.

Parte A. Diseño de un protocolo de comunicación entre dos procesos.

Siga los siguientes pasos:

1. Busque, e incluya en su reporte, una tabla del conjunto de caracteres ASCII. Nota: el código ASCII estándar utiliza 7 bits/carácter con lo cual se logra codificar hasta 128 caracteres diferentes. Sin embargo, en las computadoras personales se utiliza un conjunto ASCII extendido de 8 bits/carácter con lo cual se logra codificar hasta 256 caracteres diferentes. En este trabajo nos referimos al conjunto estándar, no al extendido.

2. Los primeros 32 caracteres del conjunto ASCII (0 al 31) tienen diversas aplicaciones de control. Describa en su reporte el uso originalmente propuesto para los códigos 1 al 4 (SOH, STX, ETX y EOT), 28 al 31 (FS, GS, RS y US), 6 (ACK), 21 (NAK), 5 (ENQ), 22 (SYN), 23 (ETB) y 24 (CAN).

3. Diseñe un protocolo de comunicaciones (en papel) que permita la transmisión de un archivo entre un proceso transmisor y uno receptor. En su diseño considere que el archivo a transmitir contendrá únicamente texto ASCII estándar sin caracteres de control (a excepción de los caracteres 10 y 13, denominados "*line feed*" y "*carriage return*", que sí pueden ocurrir en el archivo). Asuma también que el archivo puede ser arbitrariamente grande de modo que se requerirá transmitirlo por fragmentos. Utilice los caracteres de control del código ASCII para encapsular cada fragmento del archivo y así controlar el proceso de transferencia de datos. Para este propósito se sugiere utilizar el control de flujo "detener y esperar". En su reporte incluya: a) la estructura de los PDUs que se envían entre el proceso transmisor y el receptor y b) un diagrama de tiempo describiendo el protocolo de comunicaciones utilizando.

Parte B. Antecedentes a la implantación

El objetivo de esta parte de la práctica es familiarizar al alumno con algunos elementos requeridos. En su reporte incluya las respuestas a los siguientes puntos.

1. Con referencia al sistema operativo UNIX (Linux), explique lo que son las "llamadas al sistema" (*system calls*). Describa lo que hacen las siguientes llamadas al sistema: `getpid()`, `getppid()`, `read()` y `write()`. Nota: existen algunos comandos homónimos, asegúrese que las definiciones que encontró en su investigación sean, efectivamente, llamadas al sistema.

2. Escriba, compile y ejecute un programa en lenguaje C que despliegue su propio número de proceso y el número de proceso del proceso desde el cual se creó. Ayuda: use algunas de las llamadas al sistema descritas en el punto anterior.

3. Investigue lo que hacen las llamadas al sistema `fork ()` y `wait()`. Con base en su investigación reporte lo que hace el programa 1 del Apéndice. ¿Cuál es el objetivo de la instrucción que contiene la llamada al sistema `wait()`? ¿Qué sucede cuando se elimina la línea que contiene esta instrucción? Nota: el comportamiento del programa sin esta instrucción puede ser aleatorio.

4. Investigue lo que hacen las llamadas al sistema `pipe()` y `close()`. Reporte lo que hace el programa 2 del Apéndice. ¿Qué pasa cuando se intenta leer dos veces consecutivas el contenido del "pipe" sin que haya habido una operación de escritura (al pipe) intermedia?

Parte C. Implantación

Usando dos procesos UNIX (Linux) implemente y pruebe el protocolo de comunicaciones que diseñó en la parte A. Los procesos se comunicarán entre ellos usando *pipes*. Un proceso representa el transmisor y el otro el receptor. El proceso receptor deberá crear una copia en disco del archivo recibido. Independientemente de la forma de escritura, la lectura de datos del *pipe* tiene que realizarse carácter por carácter.

Como ayuda para desarrollar la aplicación se proporciona el programa 3. El código del programa crea dos procesos que pueden comunicarse entre sí a través de canales de comunicación unidireccionales (*pipes*) como se describe gráficamente en la figura 1. Al correr el programa, la función `tx()` se ejecuta en el proceso padre y `rx()` en el hijo, por lo que la implantación del protocolo básicamente consiste en codificar el contenido de estas dos funciones. Con referencia al mismo programa, las funciones `send_byte()` y `receive_byte()` se pueden usar para simplificar ligeramente las tareas de transmisión y recepción de datos.

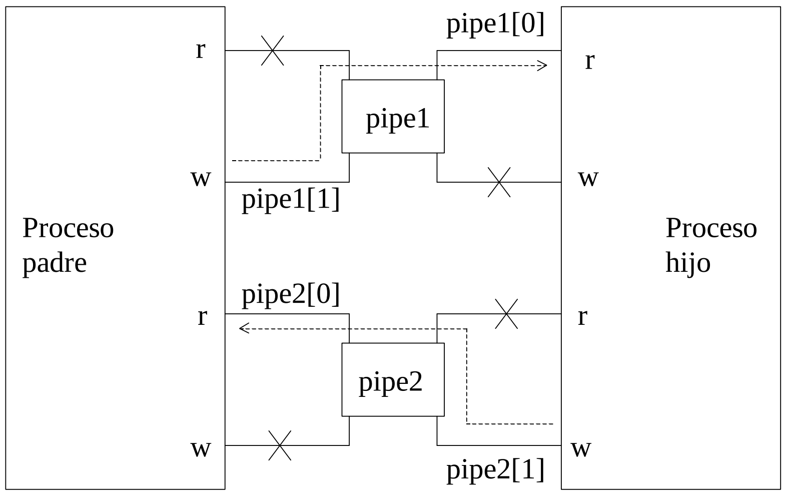


Figura 1. Diagrama que muestra la estructura creada por la ejecución del programa 3. El canal de escritura se identifica con (w) y el de lectura con la letra (r).

III. Evaluación

- Los reportes de las partes A y B consistirán de las repuestas a los requerimientos planteados en las secciones correspondientes. El reporte de la parte C deberá consistir únicamente de un listado del código fuente del programa desarrollado. Una copia de dicho código fuente. Al entregar el reporte de la parte C se deben incluir las partes A y B calificadas.
- Se enfatiza que, aunque se sugiere la discusión entre compañeros del grupo acerca de la solución a diversas partes de la práctica, el trabajo debe realizarse en forma individual.

Bibliografía

De las siguientes referencias la [1] es una excelente fuente de consulta acerca de comunicación entre procesos. La referencia [2] contiene una buena descripción del conjunto de caracteres ASCII de control. La 6ª edición de la referencia [2] también se encuentra disponible en español bajo el título *Comunicaciones y Redes de Computadores*.

[1] R. Stevens, *UNIX Network Programming*, Englewood Cliffs, N.J.: Prentice-Hall, 1990.
[2] W. Stallings, *Data & Computer Communications*, 5*th*, 6*th* *editions*, N.J.: Prentice Hall.

Anexos.

```
/* Programa 1.

   Miguel Lopez Guerrero            Nov. 1, 2004

*/


#include <stdio.h>


main() {

  int childpid;


  childpid=fork();


  printf("En el proceso %d la instruccion fork regresa: %d\n",

         getpid(), childpid);


  if(childpid==-1) {

    printf("No se pudo ejecutar la instruccion fork");

    exit (1);

  }

  else if (childpid==0)

    /* Proceso hijo */

    printf ("PROGRAMA HIJO: proceso hijo: %d, proceso padre(*): %d\n",

            getpid(), getppid());

  else {

    /* Proceso padre */

    printf ("PROCESO PADRE: proceso padre: %d, proceso hijo %d\n",
            getpid(), childpid);

    while(wait((int *) 0) != childpid)

      ;

  }



  exit (0);

}
```

```
/* Programa 2.


   Miguel Lopez Guerrero      Nov. 1, 2004

*/


#include <stdio.h>

#include <unistd.h>


main() {

  int pipefd[2], n;

  char buff[100];


  if (pipe(pipefd) < 0) {

    printf ("Error al crear la estructura \"pipe\"\n");

    exit (1);

  }


  printf("Descriptor de archivo para lectura: %d, "

         "descriptor de archivo para escritura: %d\n",

         pipefd[0], pipefd[1]);


  /* Operacion de escritura al pipe, maximo usualmente 4096 bytes */

  if (write(pipefd[1], "Programa de prueba\n", 19) != 19) {

    printf("Error en la operacion de escritura al \"pipe\"\n");

    exit (1);

  }
  /* Operacion de lectura al pipe */

  if ((n = read(pipefd[0], buff, sizeof(buff))) <= 0)  {

    printf ("Error en la operacion de lectura al \"pipe\"\n");

    exit (1);

  }



  write(1,buff,n);



  exit (0);

}
```

```
/* Programa 3. Miguel Lopez-Guerrero. Junio 13, 2005 */

#include <stdio.h>
#include <unistd.h>

void tx(int, int);
void rx(int, int);

/* Funcion para transmitir un caracter (almacenado en byte) */
int send_byte(int write_fd, char byte) {
return write(write_fd, &byte, 1);
}

/* Funcion para recibir un caracter (almacenado en *p_byte) */
int receive_byte(int read_fd, char *p_byte) {
return read(read_fd, p_byte, 1);
}

main() {
int childpid, pipe1[2], pipe2[2];
if ((pipe(pipe1) < 0) || (pipe(pipe2) < 0)) {
   printf("Pipe creation error\n");
   exit(1);
}
childpid=fork();
if (childpid > 0)
{
  /* Parent process, childpid = child process number */
  close(pipe1[0]);   /* close reading pipe */
  close(pipe2[1]);   /* close writing pipe */
  tx(pipe2[0], pipe1[1]);
  while(wait((int *) 0) != childpid)
    ;
  close(pipe1[1]);
  close(pipe2[0]);
}
else
{
  /* Child process, childpid=0 */
  close(pipe1[1]);  /* close writing pipe */
  close(pipe2[0]);  /* close reading pipe */
  rx(pipe1[0], pipe2[1]);
  close(pipe1[0]);
  close(pipe2[1]);
}
}

void tx(int read_fd, int write_fd) {
}
```

```
void rx(int read_fd, int write_fd) {

}
```

Práctica 12

Switcheo de Paquetes I (Packet Switching I)

Introduction

In this tutorial, you simulate the behavior of a packet switching network. As part of creating the network, you will use the packet and link editors, new Kernel Procedures, and define your own statistics.

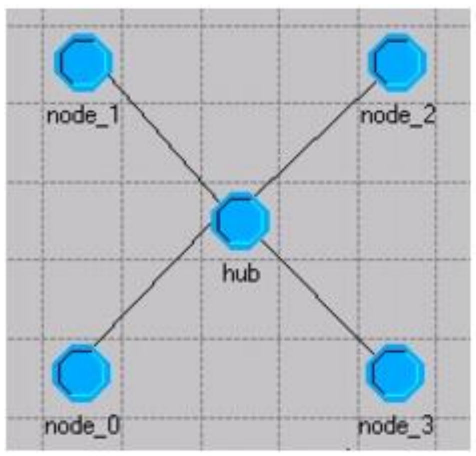
You will also

- Learn some principles of model design
- Become more familiar with process and node models and how they function in a network model
- Do parametric studies of various statistics

In a simple packet switching network, the performance of the network can be measured by the end-to-end delay experienced by traffic on the network.

In this lesson, your goal is to model a simple network in which four peripheral nodes generate traffic while a central hub node relays the traffic to the appropriate destination within the network.

A Small Packet Switching Network



Getting Started

There are several design concepts you should consider before you start building the network model:

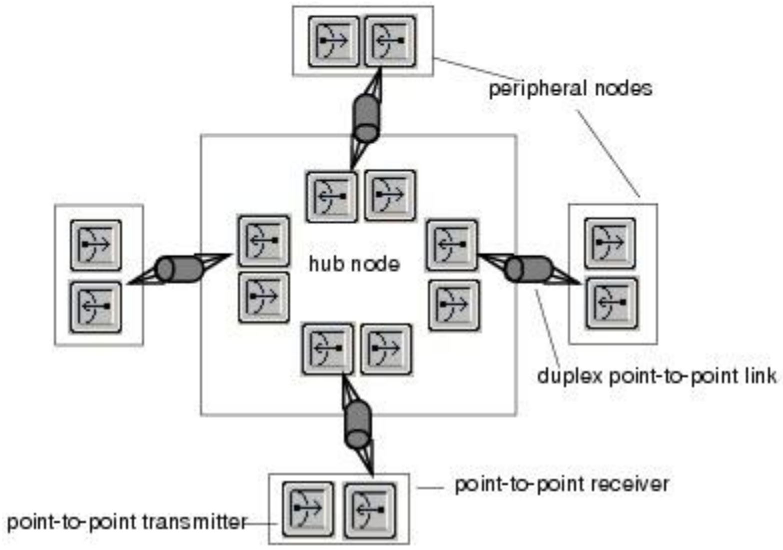
- Network topology and the physical communication medium
- The functions of the different node types
- The method the process model uses to determine which point-to-point transmitter addresses a particular peripheral node
- The role of peripheral nodes

Network Topology

The initial network will consist of four peripheral nodes connected to a hub node by point-to-point links.

Point-to-point links can either be simplex (unidirectional) or duplex (bidirectional). In this case, you will use custom duplex links to connect transmitter-receiver pairs. Looking only at the communications medium as represented, the network would look like this:

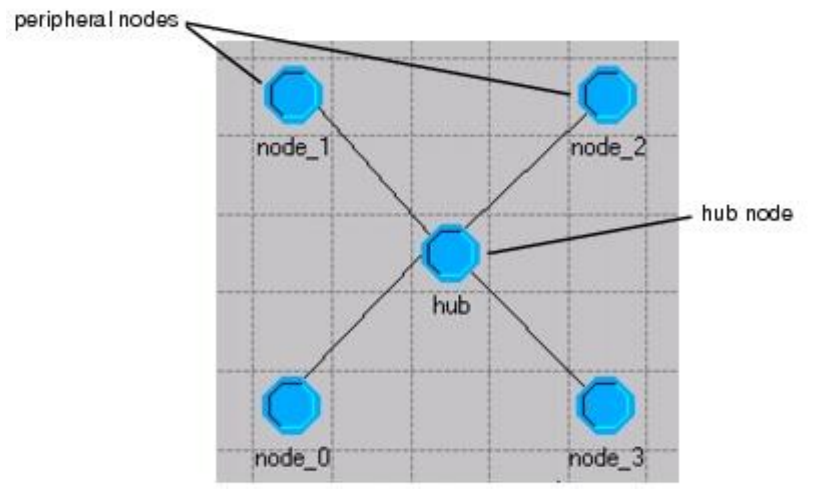
Communications Medium of the Network



Functions of the Different Node Types

The purpose of the model is to simulate packets traveling from one peripheral node to another peripheral node through the packet switching hub node. In the hub node, you can assume that packets containing destination addresses will arrive randomly on the four incoming links from the four peripheral nodes. The destination address is an integer value specifying a destination peripheral node. The hub node must contain a process model that can retrieve the incoming packets, read the destination address, and send the packets to the appropriate point-to-point transmitter.

Different Node Types

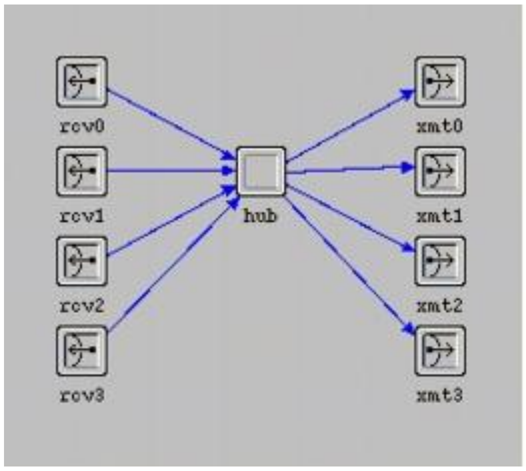


The Role of the Hub Node Model

Packet streams each have a unique index. The easiest method is to set up a direct association between the hub process outgoing packet stream indices and the peripheral destination address values. In a more adaptive model, the hub process model could maintain a table for translating destination address values to transmitter stream indices. In this tutorial, a direct correspondence between designating addresses and packet stream indices is appropriate.

In summary, the hub node model will consist of a point-to-point transmitter/receiver pair for each peripheral node, and a process model used to relay packets from a receiver to the appropriate transmitter.

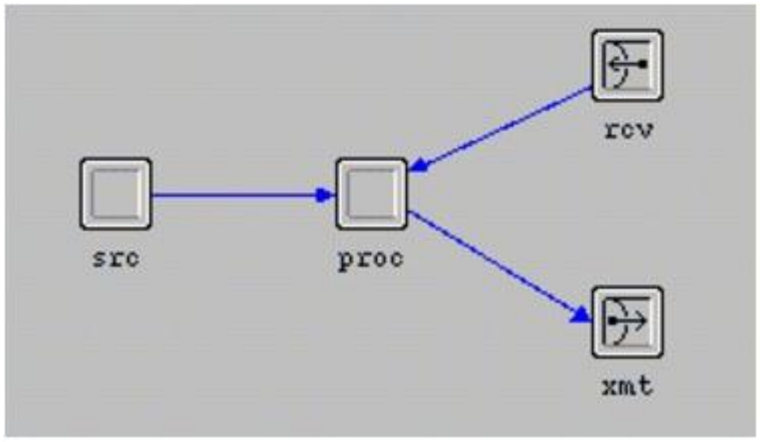
Hub Node



The Role of Peripheral Nodes

The peripheral node model must generate packets, assign destination addresses, and process received packets. The peripheral node model will use a generator module to create packets. It will use a user-defined process model to assign destination addresses to packets and send them to the node's point-to-point transmitter. This process model will retrieve packets arriving from the point-to-point receiver. Upon receiving a packet, the same process model will calculate the packet's end-to-end delay and write the value to a global statistic (a global statistic is accessible to multiple processes throughout the system).

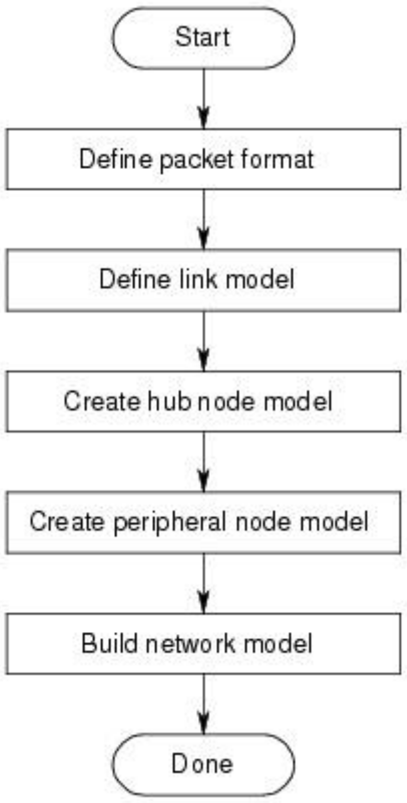
Peripheral Node



Building the Models

To build this network model, you must do the following steps:

Building the Packet Switching Network



Creating a New Packet Format

The **Packet Format Editor** can create packets containing any number of different fields that affect packet size.

The packets in this network contain a single field with the destination address of the packet. After the packet format has been created, it can be specified as an attribute in a generator so that packets created by the generator will be formatted accordingly.

To create a new packet:

1. If not already running, start the program.
2. Close any open projects.
3. Choose **File** > **New...** and select **Packet Format** from the pull-down list. Click **OK**.
4. Click on the **Create New Field** tool button, then move the cursor into the editor window.
   ▤
5. Left-click to place a field in the editor window, then right-click to end the field creation operation.

   A new field is created in the editor window.

Setting the Packet Field Attributes

To set the packet field attributes:

1. Right-click on the packet field and select **Edit Attributes** to open its **Attributes** dialog box.
2. Set the **name** attribute to **dest_address**.
3. Make sure the **type** attribute is set to **integer**.
4. Set the **size** attribute to **2**. This defines the size of the actual field in bits. For 4 possible addresses, 2 bits is enough.
5. Change the **set at creation** attribute to **unset**. This ensures that the field will not be assigned a default value when the packet is created.
6. Click **OK** when you are finished.

   The packet field in the editor window displays its name and field size.

   Packet Field

7. Choose **File** > **Save**. Name the packet **<initials>_pksw_format**, then click **Save**. Close the Packet Format Editor when you are done.

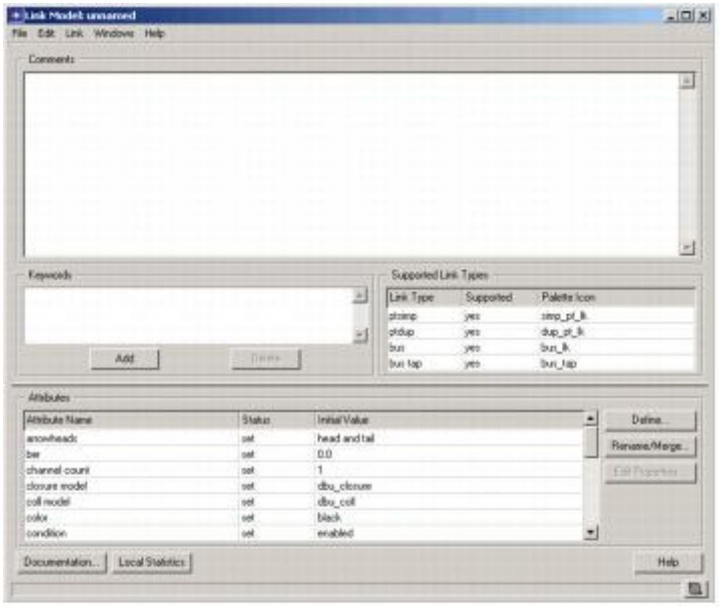
Creating a Link Model

Use the **Link Model Editor** to create custom links.

Create a link model that connects the hub and peripheral nodes. This duplex link model should support your packet format.

1. Choose **File** > **New...** and select **Link Model** from the pull-down list. Click **OK**.

   Link Model Editor



Link Type	Supported	Palette Icon
ptsimp	yes	simp_pt_lk
ptdup	yes	dup_pt_lk
bus	yes	bus_lk
bus tap	yes	bus_tap

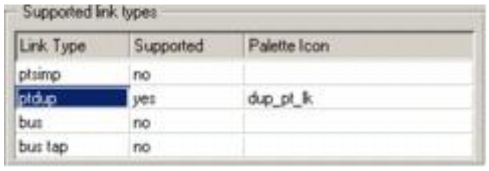
Attribute Name	Status	Initial Value
arrowheads	set	head and tail
ber	set	0.0
channel count	set	1
closure model	set	dbu_closure
coll model	set	dbu_coll
color	set	black
condition	set	enabled

Setting the Supported Packet Format

Because the link you are creating supports the packet format you just defined, there are several attributes that must be set:

1. In the **Supported link types** table, change the value under the **Supported** field to **no** for everything except **ptdup**. The model only needs to support point-to-point duplex.

   Link Model Editor (Detail)

Link Type	Supported	Palette Icon
ptsimp	no	
ptdup	yes	dup_pt_lk
bus	no	
bus tap	no	

2. In the **Attributes** table, scroll down to the **packet formats** attribute and click under the **Initial Value** field.

   The **Select Supported Packet Formats** dialog box opens.

3. Unselect the **Support all packet formats** and **Support unformatted packets** checkboxes.
4. Change the **<initials>_pksw_format** packet format's status value to **supported**.
5. Click **OK** to close the dialog box.

Setting Other Attributes

In addition to setting the supported packet format type, you must set the following attributes in the attribute table:

1. Set the **data rate** attribute to **9600**.
2. Set the **ecc model** to **ecc_zero_err** (error correction model).
3. Set the **error model** to **NONE**.
4. Set the **propdel model** to **dpt_propdel** (point-to-point propagation delay model).
5. Change **txdel model** attribute to **dpt_txdel** (point-to-point transmission delay).

You might also want to add a comment to the link model to describe it.

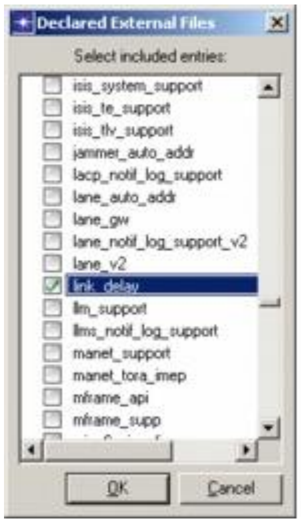
Include the link_delay external file:

1. Choose **File** > **Declare External Files...**

   The Declare External Files dialog box appears.

2. In the Entries column, look for **link_delay**.

   Declared External Files Dialog Box



Declared External Files

Select included entries:

- ☐ isis_system_support
- ☐ isis_te_support
- ☐ isis_tlv_support
- ☐ jammer_auto_addr
- ☐ lacp_notif_log_support
- ☐ lane_auto_addr
- ☐ lane_gw
- ☐ lane_notif_log_support_v2
- ☐ lane_v2
- ✓ link_delay
- ☐ llm_support
- ☐ llms_notif_log_support
- ☐ manet_support
- ☐ manet_tora_imep
- ☐ mframe_api
- ☐ mframe_supp

3. Select the link_delay entry.
4. Click OK to close the dialog box.
5. Choose **File** > **Save**, name the link model **<initials>_pksw_link**, and click **Save**.
6. Close the Link Model Editor.

Creating the Hub Node

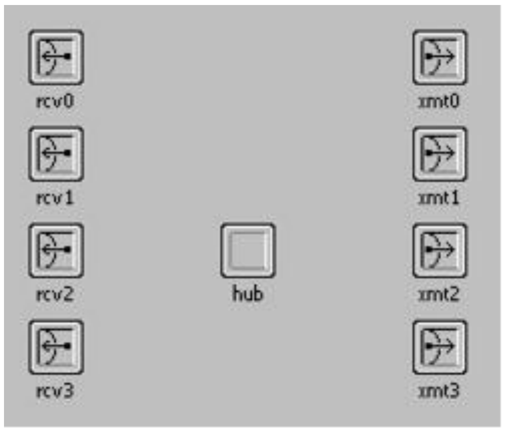
Now that you have defined a packet format and link model to be used in the model, you can create the hub and peripheral nodes. This process requires two steps for each node type: defining the node model and defining the process model.

Start by defining the node model for the hub. The hub needs four sets of transmitters and receivers for incoming and outgoing packets (one set per peripheral node), as well as a central processor to distribute the packets correctly.

Creating the Node Model

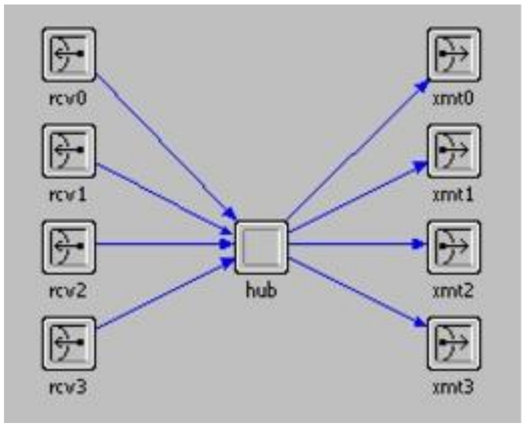
1. Choose **File** > **New...** and select **Node Model** from the pull-down list. Click **OK**.
2. Place a processor, four point-to-point transmitters, and four point-to-point receivers in the editor window as shown in the following figure.
3. Change the **name** attribute on each object as shown.

Hub Node Model



4. Create packet streams to connect the modules as shown. Start with top transmitter and receiver and work your way down. Be sure to draw the streams from the receivers to the hub, then from the hub to the transmitters.

Hub Node Model Packet Steams



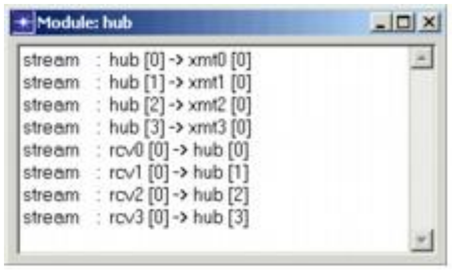
Checking the Packet Stream Assignments

To check the packet stream assignments, do the following:

1. Right-click on the **hub** process module and select **Show Connectivity** from the Object pop-up menu.

   The list of streams connecting to the **hub** module appears.

   Viewing the List of Packet Streams Connected to the Hub



```
stream  : hub [0] -> xmt0 [0]
stream  : hub [1] -> xmt1 [0]
stream  : hub [2] -> xmt2 [0]
stream  : hub [3] -> xmt3 [0]
stream  : rcv0 [0] -> hub [0]
stream  : rcv1 [0] -> hub [1]
stream  : rcv2 [0] -> hub [2]
stream  : rcv3 [0] -> hub [3]
```

2. Close the dialog box.

Setting Other Attributes

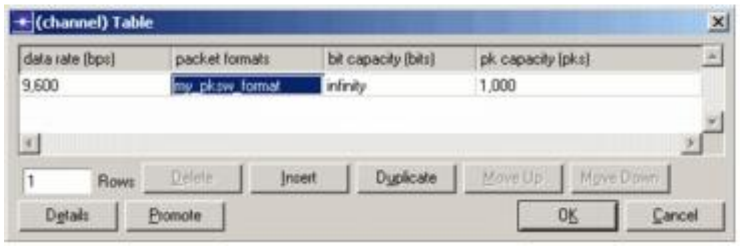
Now that the hub node model has been created, you need to set the channel data rate and supported packet format for each receiver and transmitter:

1. Hold down the **Shift** key and left-click on each transmitter module icon to select them. Make sure the packet streams are unselected.
2. When all the transmitter modules are selected, right-click on one of the modules and select **Edit Attributes** to open its Attributes dialog box.
3. Click on the value of the **channel** attribute. In the **(channel) Table** dialog box, change the **data rate** to **9600**.
4. Support the **<initials>_pksw_format** format:
   a. Click on the value of the **packet formats** attribute.

b. Unselect both checkboxes to turn off **Support all packet formats** and **Support unformatted packets**.
c. Click on the status field for **<initials>_pksw_format** and change the status to **supported**.
d. Click **OK** in the **Select Supported Packet Formats** dialog box.

5. Verify that the correct data rate and packet formats display in the **(channel) Table** dialog box, then click **OK**.

Supporting the Custom Packet Format


(channel) Table

data rate (bps)	packet formats	bit capacity (bits)	pk capacity (pks)
9,600	my_pksw_format	infinity	1,000

6. To apply changes to all selected objects, click on the **Apply changes to selected objects** checkbox, then click **OK**.

   If a warning message with information about "undoing" appears on your screen, click **Yes** to continue.

   You should see the text "4 objects changed" in the message area at the bottom of the Node Editor.

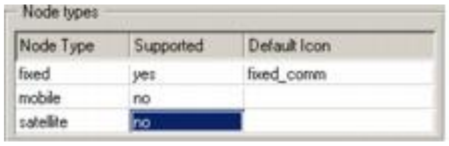
7. Left-click in a blank area of the workspace to deselect all modules.
8. Repeat steps 1–7 for the receiver modules. (You cannot update all modules at once because transmitter and receiver channels have different attributes.)

Setting Interfaces

The last step to defining the hub node module is to specify that only fixed node types are supported:

1. Choose **Interfaces > Node Interfaces**.
2. In the **Node types** table, change the **Supported** value to **no** for both **mobile** and **satellite** node types.

Node Types Table


Node types

Node Type	Supported	Default Icon
fixed	yes	fixed_comm
mobile	no	
satellite	no	

3. You may wish to add a comment to describe the node. When you are finished, click **OK** to close the dialog box.

The hub node model is now complete, except for specifying the process model for the **hub** processor module (which you can do once the process model has been created). For now, save the model as **<initials>_pksw_hub**, BUT DO NOT CLOSE IT.

Creating the Process Model of the Hub Node

In this model, the hub receives a packet and, based on the destination address, forwards it to the correct transmitter, which sends it to the destination node.

Now, create the hub process model.

In your node model, the hub processor module is connected to the transmitters and receivers by packet streams. Because each packet is associated with an interrupt, the hub process model receives an interrupt whenever a packet arrives from a receiver. Because this is the only expected type of interrupt, the hub process FSM (finite state machine) can be defined using one unforced idle state to rest between events and a transition executive containing the code for processing packets.

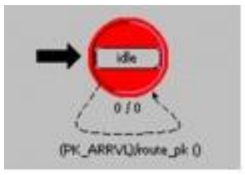
To create the process model of the hub:

1. Choose **File** > **New...** and select **Process Model** from the pull-down list. Click **OK**.
2. Using the **Create State** tool button, place one state in the editor window.
3. Change the **name** attribute of the state to **idle**.
4. Use the **Create Transition** tool button to draw a transition from the **idle** state to itself.
5. Right-click on the transition and select **Edit Attributes**. Change the **condition** attribute to **PK_ARRVL** and the **executive** attribute to **route_pk()**.

   When a packet arrives from a point-to-point receiver, the process model is invoked by a stream interrupt. The transition must test for this condition.

6. Click **OK** to close the attribute dialog box for the transition.

   Initial State and Transition



Next, define the **PK_ARRVL** macro:

1. Click on the **Edit Header Block** tool button.
2. Enter the following macro definition for **PK_ARRVL:**

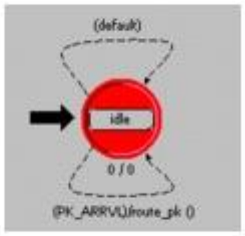
```
#define PK_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM)
```

3. Choose **File > Save** when you are finished.

   The **PK_ARRVL** condition compares the delivered interrupt type with the predefined symbolic constant **OPC_INTRPT_STRM**, which indicates a stream interrupt. Although this is the only expected type of interrupt for this model, it is good to safeguard against run-time missing transition errors. To do this, you can create a default transition loop on unforced states.

To create a default transition on the unforced state:

1. Create a transition from the **idle** state back to itself.
2. Change the **condition** attribute to **default** and click **OK** to close the dialog box.



You also must define the **route_pk() transition executive**:

1. Click on the **Edit Function Block** tool button.
2. Enter the following code:

```
static void route_pk(void)
          {
          int dest_address;
          Packet * pkptr;
          FIN(route_pk());
          pkptr = op_pk_get(op_intrpt_strm ());
          op_pk_nfd_get_int32 (pkptr, "dest_address",
                  &dest_address);
          op_pk_send (pkptr, dest_address);
          FOUT;
          }
```

3. Choose **File > Save** when you are finished.

   This code executes when the **FSM** follows the transition.

The first line after `FIN(route_pk())` has two effects: first, it retrieves the arriving packet from the appropriate input stream (whose index is determined by **op_intrpt_strm()**). Then, **op_pk_get()** uses the packet-stream index argument to return a pointer to the packet.

Next, the process must obtain the destination address contained in the packet. The destination address is held in the packet's **dest_address** field, which is an integer data type. The second line of code assigns the destination address to the **dest_address** local variable.

Recall that the destination address corresponds directly to the outgoing packet stream index of the hub node. The final line of code sends the packet to the correct point-to-point transmitter based on the destination address.

Finally, you can define the model interface attributes for the process and compile the model. First, enable the begsim intrpt attribute:

1. Choose **Interfaces > Process Interfaces**.
2. Set the **begsim intrpt** attribute to enabled.
3. Add a comment if you want, then click **OK** when you are finished with the Process Interfaces.

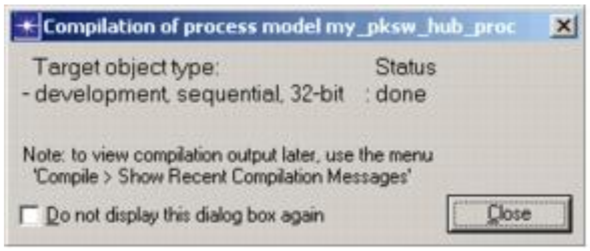
Next, compile the model:

1. Click on the **Compile Process Model** tool button.
2. Name the file **<initials>_pksw_hub_proc**, then click **Save**.

   The compilation status dialog box appears.

3. When the Status changes to **done**, click **Close**.

   Compilation Status Dialog Box



Compilation of process model my_pksw_hub_proc

```
Target object type:                  Status
- development, sequential, 32-bit   : done

Note: to view compilation output later, use the menu
 'Compile > Show Recent Compilation Messages'
```

☐ Do not display this dialog box again   Close

4. Choose **File > Close** to close the Process Editor.

After you define the hub process model, you can set the **process model** attribute of the **hub** processor module in the Node Editor.

1. Choose **Windows > Node Model > <initials>_pksw_hub**.

   The Node Model Editor window becomes active.

2. Right-click on the **hub** module and select **Edit Attributes**. Change the **process model** attribute to **<initials>_pksw_hub_proc**.
3. Click **OK** to close the Attributes dialog box.

4. Save the node model.

Creating the Peripheral Node Model

When a peripheral node generates a packet, it must assign a destination address to the packet, then transmit it to the hub. When it receives a packet, the node must record the packet's end-to-end delay. To accomplish these tasks, a peripheral node model must consist of a generator module, a processor module, and a point-to-point transmitter and receiver.

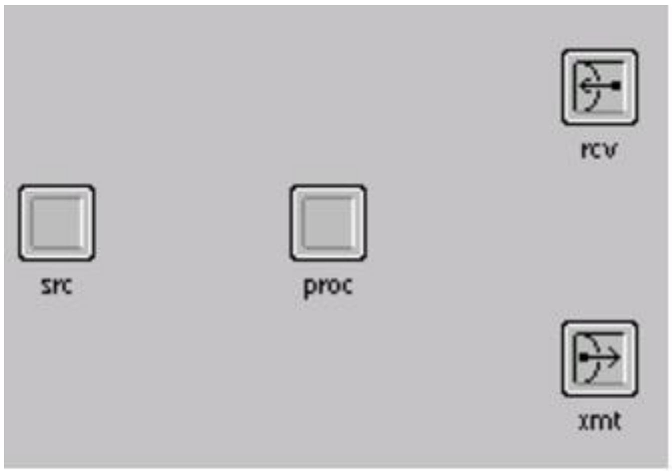
To create the peripheral node model:

1. Make sure you have saved your hub model, then choose **Edit > Clear Model**.

   The workspace is cleared. This avoids the step of closing and reopening the Node Editor.

2. **Save As... <initials>_pksw_node** to save the model you are about to construct.
3. Place and name the modules as shown.

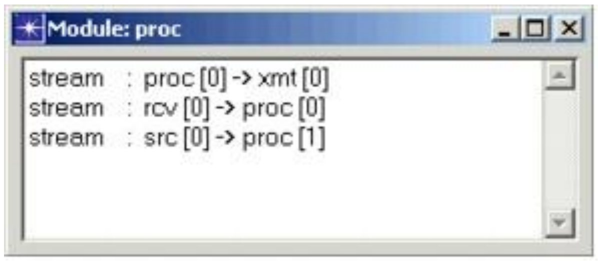
   Initial Modules



4. Create packet streams to connect the different modules in the following order:
   - **rcv** to **proc**
   - **proc** to **xmt**
   - **src** to **proc**

Verify the assignment of packet streams between the modules. Right-click on the **proc** module and choose **Show Connectivity** from the pop-up menu to see the following table.

Show Connectivity Dialog Box (Detail)



```
stream  : proc [0] -> xmt [0]
stream  : rcv [0] -> proc [0]
stream  : src [0] -> proc [1]
```

If the streams are not correct, the simulation might not generate results. If necessary, correct the streams by deleting all links and recreating them as specified in Step 4.

After you create the network models, you will want to run simulations for different packet interarrival times, setting the **Packet Interarrival Time** attribute just before running the simulations. To make it possible to set the attribute then, you need to promote the attribute. To do this, you need to edit the attributes of the **src** module:

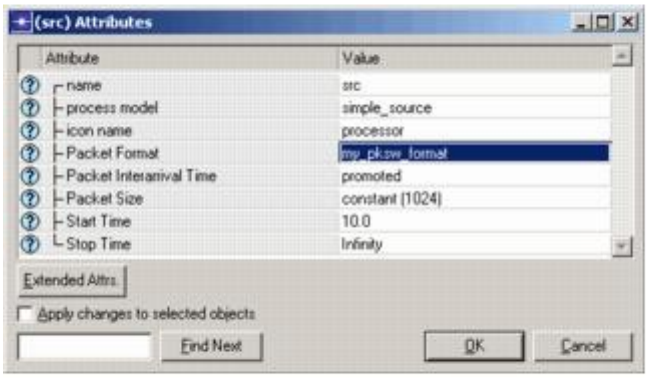
1. Right-click on the **src** module to open its **Attributes** dialog box.
2. Change the **process model** attribute to **simple_source**.
3. Click on **Packet Interarrival Time** in the left column to highlight the Attribute name, then right-click and select **Promote Attribute to Higher Level** from the pop-up menu.

   The **Packet Interarrival Time** attribute is promoted so that its value can be set later.

   We also want the processor to create packets with the previously-defined format.

4. Specify **<initials>_pksw_format** as the value for the **Packet Format** attribute.
5. Verify that your **(src) Attributes** table is similar to the one in the following figure, then click **OK**.

   Packet Format Attribute is <initials>_pksw_format

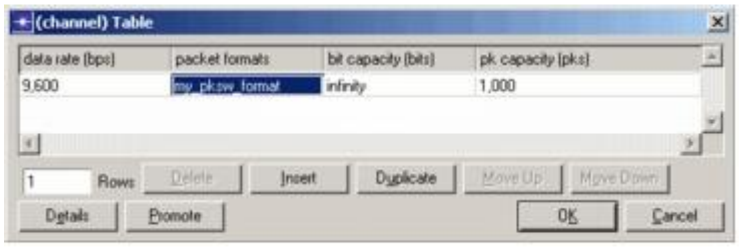


Attribute	Value
name	src
process model	simple_source
icon name	processor
Packet Format	my_pksw_format
Packet Interarrival Time	promoted
Packet Size	constant (1024)
Start Time	10.0
Stop Time	Infinity

Next, we want to change the data rate and supported packet formats for the receiver and transmitter. You did this same operation earlier, in the hub module:

1. Make sure the packet streams are unselected. This could happen when you select the first receiver.
2. Right-click on the transmitter module and select **Edit Attributes**.
3. Click on the value of the **channel** attribute. In the **(channel) Table** dialog box, set the **data rate (bps)** to **9600**.
4. Support the **<initials>_pksw_format** format, as follows:
    a. Click on the value of the **packet formats** attribute.
    b. Unselect both checkboxes to turn off **Support all packet formats** and **Support unformatted packets**.
    c. Click on the status field for **<initials>_pksw_format**.
    d. Click **OK** to close the **Select Supported Packet Formats** dialog box.
5. Verify that the correct data rate and packet formats display in the **(channel) Table** dialog box, then click **OK**.

Supporting the Custom Packet Format


(channel) Table

data rate (bps)	packet formats	bit capacity (bits)	pk capacity (pks)
9,600	my_pksw_format	infinity	1,000

6. In the open **Attributes** dialog box, click **OK**.

    The message "1 object changed" appears in the status bar.

7. Repeat steps 2–6 for the receiver module.

The next step in defining the peripheral node model is to specify that only fixed node types are supported. You did this same operation earlier, in the hub module.

Because the node is fixed, not mobile or satellite, you need to edit the model attribute interfaces:

1. Choose **Interfaces** > **Node Interfaces**.
2. In the **Node types** table, change the **Supported** value for **mobile** and **satellite** types to **no**, and add a comment to describe the model, if you want.
3. Do not close the **Node Interfaces** dialog box. You still need to use it.

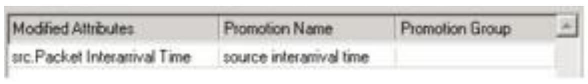
Renaming attributes allows you to simplify complex attribute names or to expand terse ones.

You might want to simplify attribute names, especially when long hierarchical names are confusing and unnecessary. You might also want to expand a highly-abbreviated attribute name to make it more understandable.

For this model, you will rename the interarrival time attribute:

1. In the **Node Interfaces** dialog box, click on the **Rename/Merge...** button.
2. In the **Unmodified Attributes** pane, select the **src.Packet Interarrival Time** attribute, then click on the **>>** button to move it to the **Modified Attributes** pane.
3. Change the text in the **Promotion Name** column to **source interarrival time.**

   Renaming an Attribute



Modified Attributes	Promotion Name	Promotion Group
src.Packet Interarrival Time	source interarrival time	

4. Click **OK** to close the **Rename/Merge Attributes** dialog box.

You can assign a set of symbolic values to an attribute to provide a simpler user interface. Your assigned values are displayed as options in a list.

There are several advantages to assigning symbolic names to attribute values:

- Options can be limited to appropriate values.
- Options can be assigned descriptive names, making it easier for a user to select the correct value.
- Users can select values from a list, avoiding typographical errors.

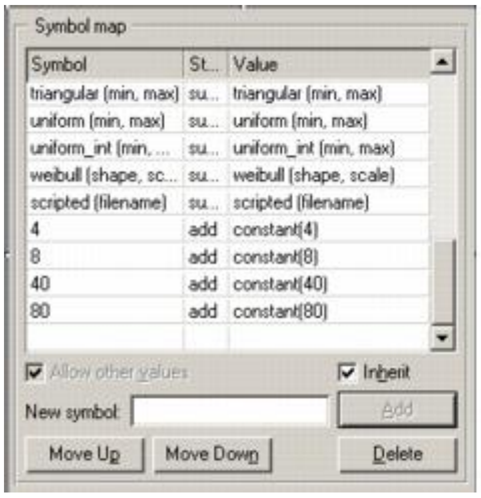
Add symbolic values to the **source interarrival time** attribute:

1. In the Attributes table of the **Node Interfaces** dialog box, select the newly-renamed **source interarrival time** attribute, then click on the **Edit Properties...** button.

   The **Attribute: source interarrival time** dialog box appears.

2. In the **Symbol map** table, change the status of all symbols to **suppress** by clicking in the **status** column of each row.
3. Add entries in the **Symbol map** as shown, then click **OK** to close the dialog box.

   Adding Symbolic Values



Symbol map

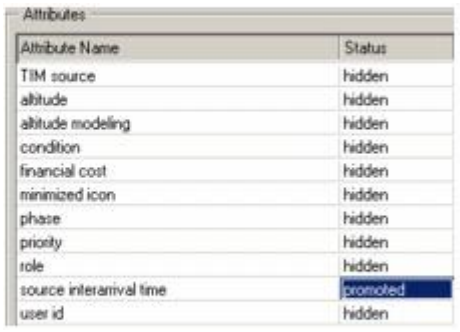
Symbol	St...	Value
triangular (min, max)	su...	triangular (min, max)
uniform (min, max)	su...	uniform (min, max)
uniform_int (min, ...	su...	uniform_int (min, max)
weibull (shape, sc...	su...	weibull (shape, scale)
scripted (filename)	su...	scripted (filename)
4	add	constant(4)
8	add	constant(8)
40	add	constant(40)
80	add	constant(80)

Hiding attributes simplifies an interface. This is desirable when the end user does not need to change an attribute value. Hiding an attribute does not affect its value during simulation.

Many attributes that appear in the peripheral node are not relevant to this model. To avoid confusion and clutter, these attributes can be hidden:

1. In the **Attributes** table of the **Node Interfaces** dialog box, change the status of all attributes except **source interarrival time** to **hidden.**

   Node Interfaces Dialog Box



Attributes

Attribute Name	Status
TIM source	hidden
altitude	hidden
altitude modeling	hidden
condition	hidden
financial cost	hidden
minimized icon	hidden
phase	hidden
priority	hidden
role	hidden
source interarrival time	promoted
user id	hidden

2. Click **OK** to close the **Node Interfaces** dialog box.
3. Choose **File** > **Save** to save the model.

Creating the Process Model of the Peripheral Node

The peripheral node process model does two main functions: (1) assigns destination addresses and sends packets, and (2) keeps track of end-to-end delay.

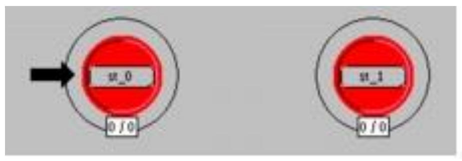
To complete its tasks, the peripheral node process model needs two states:

- an initial state
- an idle state

To create the process model:

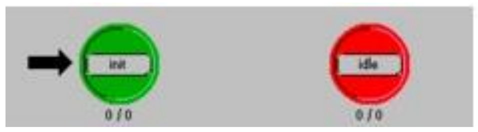
1. Choose **File > New...** and select **Process Model** from the pull-down list. Click **OK**.
2. Put two states in the editor window as follows:

   Initial States

3. Change the attributes of the states, as indicated:
   a. For the initial state, change the **name** attribute to **init** and the **status** to **forced**.

      Changed States

   b. For the other state, change the **name** attribute to **idle**. (Leave the **status** as **unforced**.)

In the **init** state, the process model will load a uniform PDF in a range of 0 to 3.

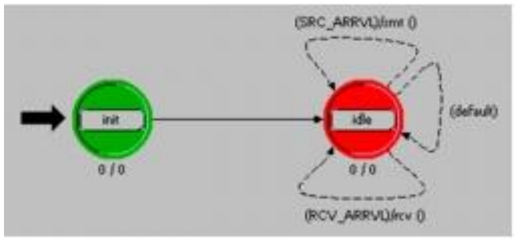
Next, create transitions between the states in the process model:

1. Draw the transitions between the states and specify the transition conditions and executives as shown below.

   The **xmt()** transition executive will generate and assign destination addresses to packets as they arrive from a generator. Packets will then be sent on to the point-to-point transmitter.

   The **rcv()** transition executive is entered when a packet arrives. In the executive, the process model will determine the packet's end-to-end delay, update the global statistic, and destroy the packets.

   Drawing Transitions



2. Next, click on the **Edit Header Block** tool button to define the transition macros for **SRC_ARRVL** and **RCV_ARRVL** in the header block.
   HB
3. Enter the following code:

```
/* packet stream definitions */
#define RCV_IN_STRM 0
#define SRC_IN_STRM 1
#define XMT_OUT_STRM 0
/* transition macros */
#define SRC_ARRVL (op_intrpt_type () == \
          OPC_INTRPT_STRM && op_intrpt_strm () == SRC_IN_STRM)

#define RCV_ARRVL (op_intrpt_type () == \
          OPC_INTRPT_STRM && op_intrpt_strm () == RCV_IN_STRM)
```

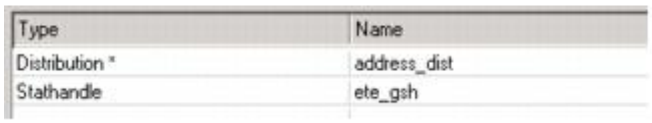
Note the use of constant definitions of stream indices (`RCV_IN_STRM`, `SRC_IN_STRM`, and XMT_OUT_STRM) for easier code interpretation. The stream indices correspond to the values set in the node model.

4. Choose **File > Save**.

Next, define the state and temporary variables:

1. Click on the **Edit State Variables** tool button.
   SV
2. Enter the following information in the table:

State Variable Dialog Box (Detail)



Type	Name
Distribution *	address_dist
Stathandle	ete_gsh

3. Click **OK** when you are finished with the state variables.

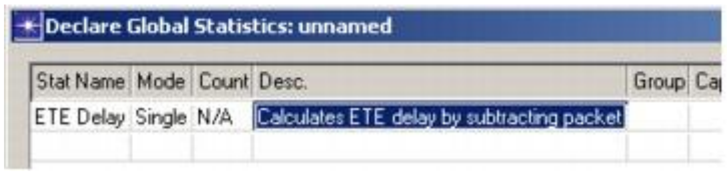
Next, you need to define the end-to-end delay statistic that will be collected when each packet arrives at its destination.

1. Choose **Interfaces > Global Statistics**.
2. Enter the **Stat Name** as **ETE Delay**.
3. Click in the **Desc.** column and enter the following text in the dialog box:

```
Calculates ETE delay by subtracting packet
creation time from current simulation time.
```

4. Choose **File > Save**.
5. Verify that the completed **Declare Global Statistics** dialog box looks like this:

Declare Global Statistics Dialog Box (Detail)



Declare Global Statistics: unnamed

Stat Name	Mode	Count	Desc.	Group	Ca
ETE Delay	Single	N/A	Calculates ETE delay by subtracting packet		

6. Close the **Declare Global Statistics** dialog box by clicking on the **OK** button.

Finally, you need to add the executives for some states and transitions in the process model. First add the enter execs for the **init** state:

1. Double-click on the top half of the **init** state to open its **enter execs**, then enter the following code:

```
address_dist = op_dist_load ("uniform_int", 0, 3);
ete_gsh = op_stat_reg ("ETE Delay",
          OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
```

2. Choose **File** > **Save**.

The **xmt()** transition executive acts when a packet arrives from the generator module. A destination address must be assigned before the packet is sent to the point-to-point transmitter.

1. In the **Function Block**, enter the following code:

```
static void xmt(void)
          {
          Packet * pkptr;
          FIN(xmt());
          pkptr = op_pk_get (SRC_IN_STRM);
          op_pk_nfd_set_int32 (pkptr, "dest_address",
                   (int)op_dist_outcome (address_dist));
          op_pk_send (pkptr, XMT_OUT_STRM);
          FOUT;
          }
```

The first statement obtains a pointer to the packet arriving from the generator. The next statement sets the value of the **dest_address** field, which is an integer data type, to the value returned by the **op_dist_outcome()** procedure.

op_dist_outcome() returns a random number according to the distribution given as an argument. In this case, **address_dist** is a pointer to the uniform integer distribution loaded in the **init** state.

The last statement sends the packet to the output stream, which is connected to the point-to-point transmitter.

The **rcv** transition executive is entered when a packet arrives on the stream connected to the point-to-point receiver. The purpose of the **rcv** executive is to calculate and record the end-to-end delay of the packets using a global statistic.

1. In the **Function Block**, enter the following code:

```
static void rcv(void)
          {
          Packet *pkptr;
          double ete_delay;
          FIN (rcv());
          pkptr = op_pk_get (RCV_IN_STRM);
          ete_delay = op_sim_time () -
                  op_pk_creation_time_get (pkptr);
          op_stat_write (ete_gsh, ete_delay);
          op_pk_destroy (pkptr);
          FOUT;
          }
```

The first statement in the code after `FIN (rcv())` obtains a pointer to the packet arriving from the point-to-point receiver.

The next statement calculates the end-to-end delay by subtracting the packet's creation time from the current simulation time.

The third statement writes the end-to-end delay to a global statistic, and the fourth statement destroys the packet.

2. Choose **File** > **Save** to close the edit pad.

Enable the beginning simulation interrupt. This interrupt starts the process.

1. Choose **Interfaces > Process Interfaces**.
2. In the **Process Interfaces** dialog box, change the initial value of the **begsim intrpt** attribute to **enabled**.

   You may want to add a comment to describe the process.

3. When you are finished, click **OK** to close the dialog box.

Finally, compile the process model, as follows:

1. Click on the **Compile Process Model** tool button.
2. Supply the file name **<initials>_pksw_nd_proc** and click **Save**.
3. When the process model has finished compiling, close the **Compilation** dialog box then close the Process Model Editor. (If the model does not compile, check for typographical errors.)

Now that the peripheral node process model is complete, you can update the process model attribute of the **proc** processor module:

1. If a Node Editor is not already open with the **<initials>_pksw_node** node model, choose **File > Recent Files > Node Model > <initials>_pksw_node**.
2. Right-click on the **proc** module and select **Edit Attributes** from the Object pop-up menu, then change the **process model** attribute to **<initials>_pksw_nd_proc**.
3. Click **OK** in the **Attributes** dialog box after you have changed the **process model** attribute.
4. **Save** the changes made to the node model, then close the Node Editor.

You are now ready to build the network model.

Building the Network

For the packet switching network, neither the subnet extent nor the grid properties are important.

Now that you have built the underlying node, process, and link models, you can build the network model. Remember that the initial network contains one hub node and four peripheral nodes. To begin:

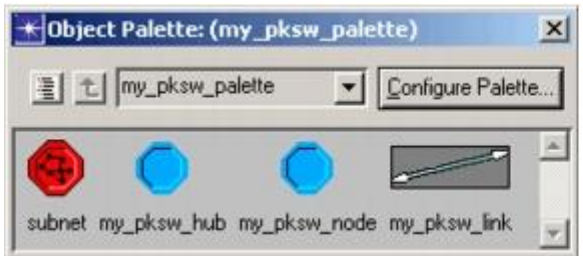
1. Choose **File** > **New...** and select **Project** from the pull-down list. Click **OK.**
2. Enter the project name as **<initials>_pksw_net**, and the scenario name as **baseline**, then click **OK**.
3. When the Startup Wizard appears, click **Quit**. You will specify the scale of the network and a palette manually.

The first thing to do is create an object palette that contains the necessary models.

1. Click on the **Open Object Palette** tool button.
2. Switch to the icon view by clicking on the button in the upper-left corner of the dialog box.
3. Click on the **Configure Palette...** button in the object palette.

4. In the **Configure Palette** dialog box, click on the **Clear** button, then click on the **Node Models** button.
5. Include **<initials>_pksw_hub** and **<initials>_pksw_node**, then click **OK**.
6. In the **Configure Palette** dialog box, click on the **Link Models** button.
7. Include the **<initials>_pksw_link** model, then click **OK**.
8. Click **OK** in the **Configure Palette** dialog box, name the palette **<initials>_pksw_palette**, and click **Save**. If you are using the Wireless module, you will also see mobile and satellite subnet nodes, but your palette should look similar to this:

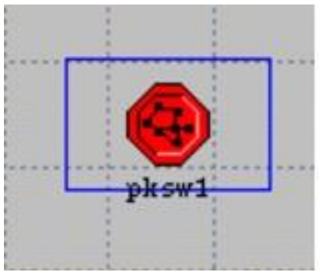
Customized Object Palette



You are now ready to build the network:

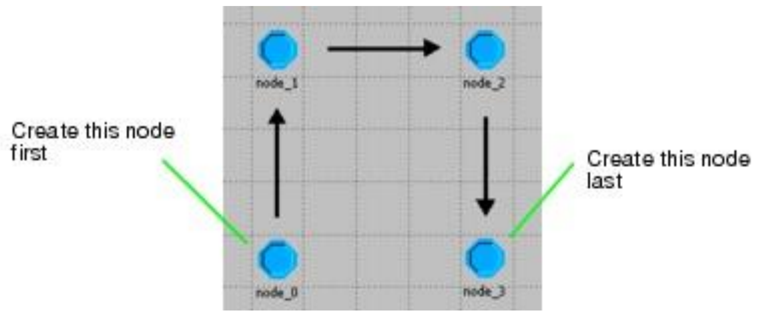
1. Place a subnet in the editor window and name it **pksw1**.

Network Model



2. Double click on the subnet to move to the subnet view.
3. Place four **<initials>_pksw_node** objects in the window as shown in the following diagram:

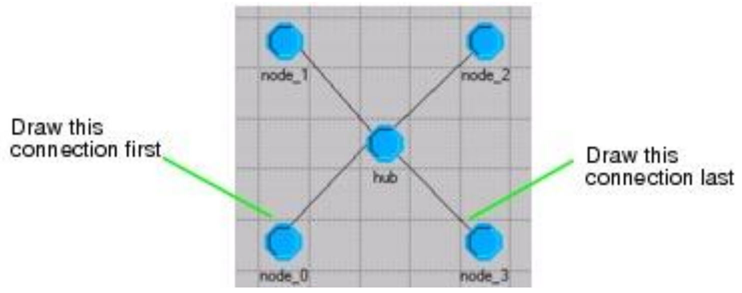
Placing Nodes



4. Place one **<initials>_pksw_hub** in the window, in the middle of the four existing nodes. Name that node **hub**.
5. Beginning at node_0, and using **<initials>_pksw_link**, connect each peripheral node to the hub as shown in the following figure.

   Drawing connections in the order node_0. node_1, node_2, and node_3 ensures that your statistics match those shown in this lesson. If you make connections in a different order, your simulation results might differ.

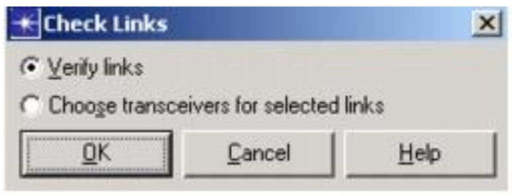
   Connecting Peripheral Nodes to the Hub



Before you save the project, it is a good idea to get into the habit of checking the links in the network for consistency:

1. Choose **Topology** > **Verify Links** (or press **Ctrl+L**).
2. Make sure that the **Verify links** radio button is selected, then click **OK**.

   Check Links Dialog Box



3. If a red **X** appears over a link, the link is not consistent. In that case:
   a. Choose **Topology > Verify Links** again.

b. Select the **Choose transceivers for selected links** radio button, then click **OK**.

   The red **X** disappears when the link inconsistency is resolved.

For a link to be consistent, the packet formats and data rates of the transmitters and receivers within the objects must match those defined for the link.

Choosing, Collecting, and Analyzing Results

Now that you have built the model, you are ready to simulate network behavior.

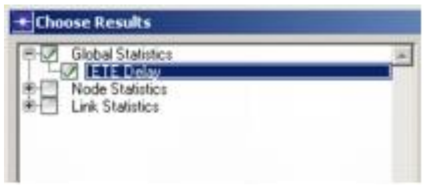
To see the effect of the packet generation rate on the performance of the network, you will use two simulation runs with different packet interarrival times.

Choosing Results

First, choose the results to be collected during the simulation: end-to-end delay and link utilization.

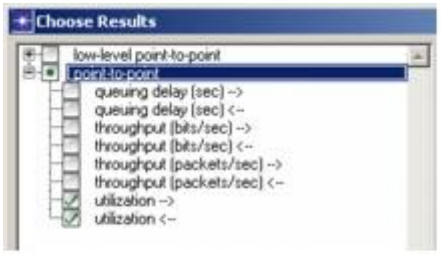
1. Right-click in a blank area of the editor window, then select **Choose Individual DES Statistics** from the pop-up menu.
2. Under **Global Statistics**, choose **ETE Delay**. This was the statistic you defined in the peripheral nodes' process model. When you are finished, click **OK**.

Choose Results Dialog Box



3. Right-click on the link between **node_0** and the **hub**, then select **Choose Individual DES Statistics** from the Object pop-up menu.
4. Under **point-to-point**, choose both **utilization** statistics. Click **OK** when you are finished.

Utilization Statistics Selected



5. Save the project with the default name.

Make sure that the **repositories** preference is set correctly before configuring and running the simulation.

1. Choose **Edit** > **Preferences**.
2. Make sure the **Network Simulation Repositories** preference has the value **()**.
3. Close the dialog box for the preferences.

Configuring two Simulation Runs

In this simulation, the source processor creates packets of a constant size. This setting, in combination with the fixed data rate of the point-to-point transmitters and receivers, results in a fixed end-to-end delay for the packets.

However, if packets are sent to a transmitter quickly enough, some of the packets will be delayed in the transmitter's queue. If the packet interarrival time (**source interarrival time** attribute) is varied, the end-to-end delay will be affected. To model this, configure two simulation runs with different packet interarrival times:

1. Choose **DES** > **Configure/Run Discrete Event Simulation**.

   The **Configure/Run DES** dialog box opens.

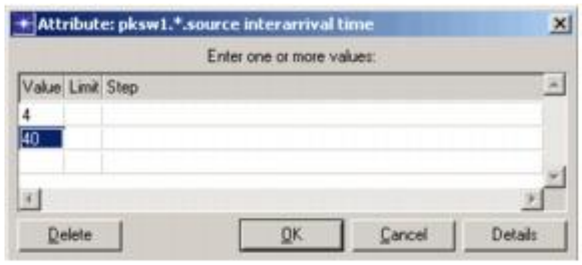
   Now you will define two simulation runs for two different packet interarrival times. Recall that we promoted the generator's **interarrival time** attribute so we could define it now, at simulation time.

   To define two simulation runs with packet interarrival times of, respectively, 4 and 40:

2. Change the **Duration** to **1000 seconds** and the **Seed** to **21**.
3. Define the **interarrival time** attribute, as follows:
   a. Click on the **Inputs** element in the tree.
   b. Click on the **Object Attributes** element.
   c. Click the **Add...** button.
   d. In the Add Attribute dialog box, click in the **Add?** column next to the **pksw1.*.source interarrival time** attribute and click **OK**.

e. In the Configure/Run DES dialog box, click on the newly added attribute name, then click on the **Enter Multiple Values...** button.
f. In the Attribute dialog box, add the Values 4 and 40

Setting Multiple Values for an Attribute

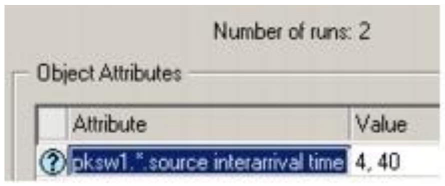

Attribute: pksw1.*.source interarrival time
Enter one or more values:
Delete
OK
Cancel
Details

Value	Limit	Step
4		
40		

g. Click **OK** to accept the attribute values.

The two values are listed next to the attribute and the total number of runs is now 2.

Configure/Run DES Dialog Box (Detail) for the Attribute

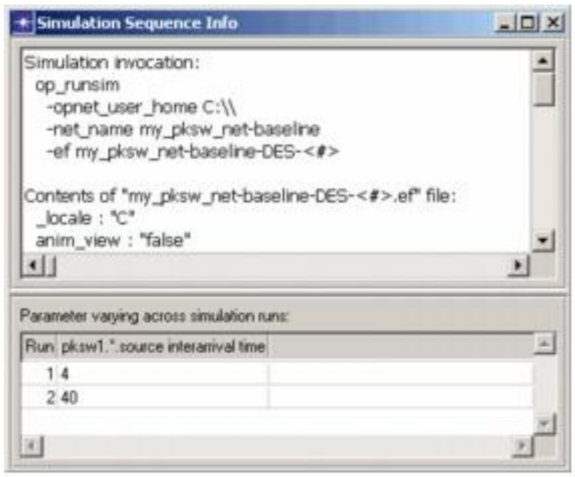


Attribute	Value
pksw1.*.source interarrival time	4, 40

h. Click on the **Preview Simulation Set** button to review how the two runs are set up.

The **Simulation Sequence Info** dialog box opens, showing the common and per-run information.

Simulation Sequence Info Dialog Box



```
Simulation invocation:
 op_runsim
  -opnet_user_home C:\\
  -net_name my_pksw_net-baseline
  -ef my_pksw_net-baseline-DES-<#>

Contents of "my_pksw_net-baseline-DES-<#>.ef" file:
 _locale : "C"
 anim_view : "false"
```

Parameter varying across simulation runs:

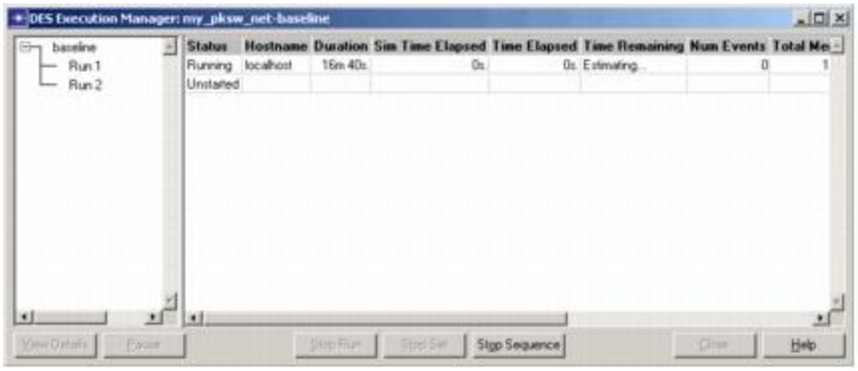
Run	pksw1.*.source interarrival time
1	4
2	40

i. Close the Simulation Sequence Info dialog box.

4. Click **Run** to save your changes, close the dialog box, and execute the simulations.

   The DES Execution Manager dialog box opens, showing the two configured runs for the **baseline** scenario. Each simulation run takes a few seconds to run, depending on the speed of your machine.

   DES Execution Manager Dialog Box



Status	Hostname	Duration	Sim Time Elapsed	Time Elapsed	Time Remaining	Num Events	Total Me
Running	localhost	16m 40s	0s	0s	Estimating...	0	1
Unstarted							

5. After the simulations complete, close the **DES Execution Manager** dialog box. If you had problems, see *"Troubleshooting Tutorials"*.

Analyzing Results

To compare the end-to-end delay and utilization between the two simulation runs:

1. Right-click in the Workspace and choose **View Results** from the pop-up menu.

   The **Results Browser** opens. It lists results for the two runs of the scenario.

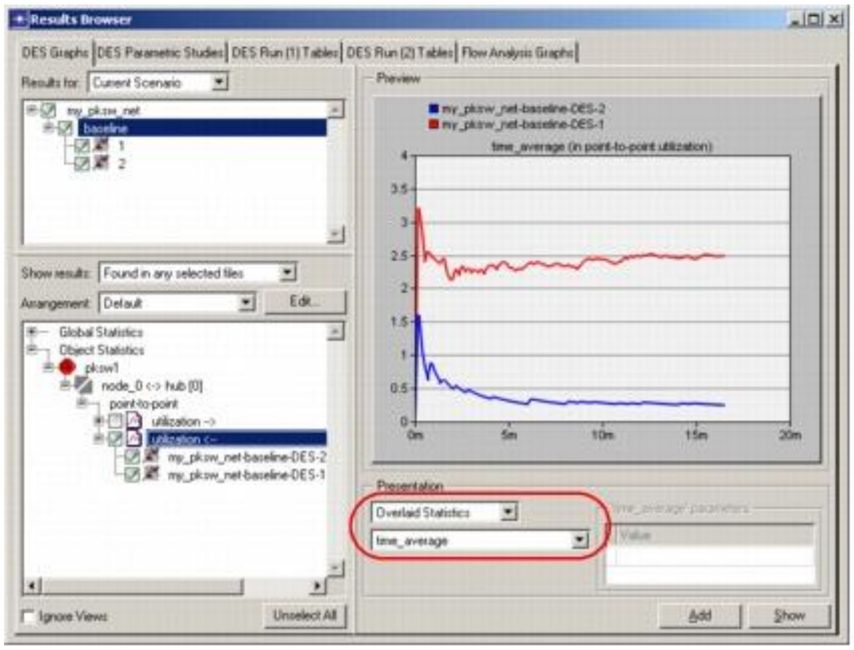
2. In the statistics tree, expand:

   Object Statistics > pksw1 > node_0 <-> hub[0] > point-to-point and select **utilization <--.**

   Two graphs are shown, corresponding to the two simulation runs for that scenario. You can view the runs in question by expanding the **<initial>_pksw_net > baseline** tree item.

3. Set the graphing option **Overlaid Statistics**, then select **time_average** from the pull-down list of available filters.

   Utilization <-- Statistic is Checked for Both Output Vector Files



Results for: Current Scenario

- my_pksw_net
  - baseline
    - 1
    - 2

Show results: Found in any selected files

Arrangement: Default

- Global Statistics
- Object Statistics
  - pksw1
    - node_0 <-> hub [0]
      - point-to-point
        - utilization ->
        - utilization <--
          - my_pksw_net-baseline-DES-2
          - my_pksw_net-baseline-DES-1

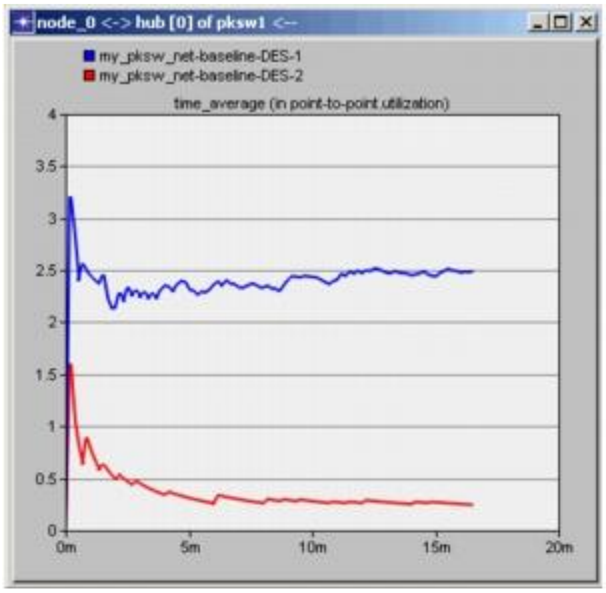
Overlaid Statistics

time_average

4. Click **Show**.

The time-averaged graph of utilization for this lesson is shown in the following figure.

Time-averaged Utilization for Both Simulations



A large reduction in packet generation results in a corresponding reduction in link utilization.

You can change the x-axis to display seconds by right-clicking in the panel (not the graph), then selecting **Time Axis > seconds** from the pop-up menu.

Close and delete the graph when you are finished.

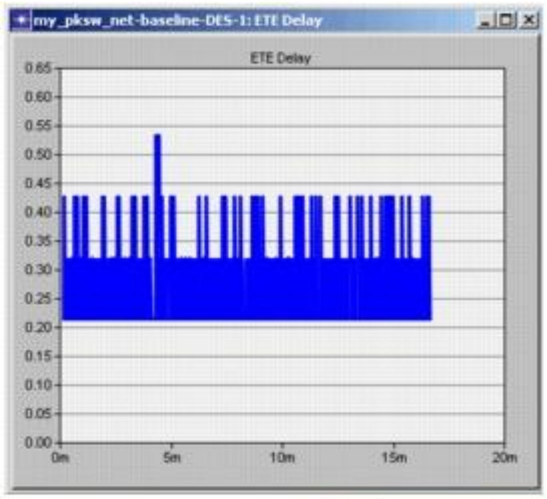
You also want to look at the end-to-end delay of the packets to see if any queuing occurred.

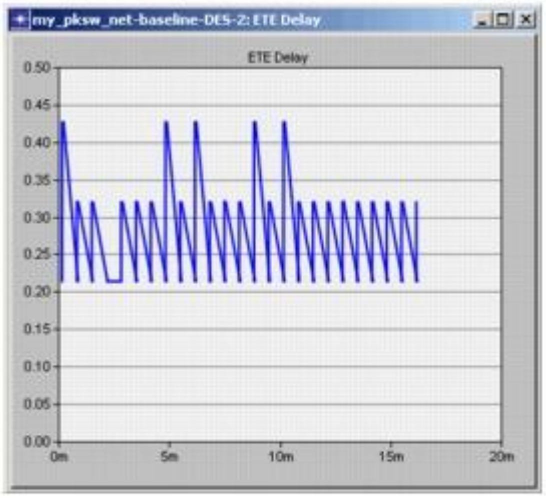
1. Unselect **utilization <--** in the **Results Browser** dialog box.
2. Expand the **Global Statistics** statistic tree node, then the **ETE Delay** node, and finally click **<initials>_pksw_net-baseline-DES-1**. This allows you to select one data source from the list of available data files.
3. Change the filter type back to **As Is then click Show**.
4. Move the graph so that it does not overlap the **Results Browser**.
5. Unselect **<initials>_pksw_net-baseline-DES-1**, select **<initials>_pksw_net-baseline-DES-2**, and finally click **Show**.

   The two graphs are shown in the following figure.

The graphs for end-to-end delay should resemble the ones in the following figure.

End-to-End Delay for Both Simulations (Linear Draw Style)





In linear draw style, the graphs do not clearly show the delay experienced by individual packets. To show this, you can change the draw style to discrete.

To change the draw style of one of the ETE delay graphs to discrete:

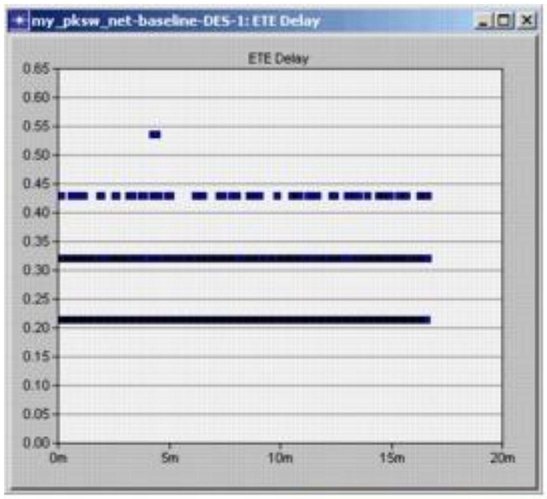
1. Right-click anywhere in the graph, and choose **Draw style > Discrete** from the pop-up menu.

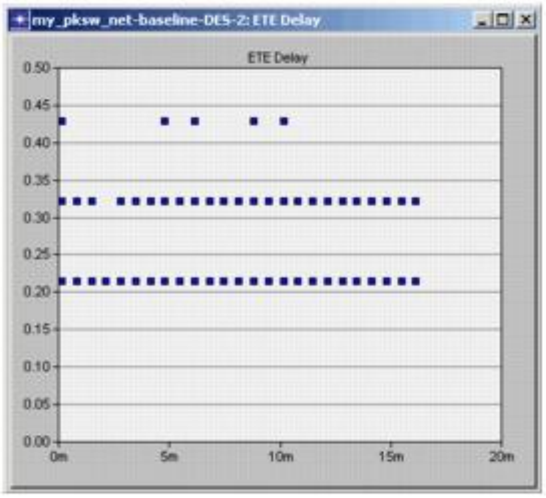
   The graph changes from a linear to discrete draw style.

2. Perform the same operation for the other ETE Delay graph.

The discrete graphs of ETE delay should resemble those in the following figure:

End-to-End Delay for Both Simulations (Discrete Draw Style)





In the next tutorial, you enhance the current network model by adding a second star network. Return to the list of tutorials in the Contents pane and choose **Packet Switching II**.

Práctica 13

Switcheo de Paquetes II (Packet Switching II)

Introduction

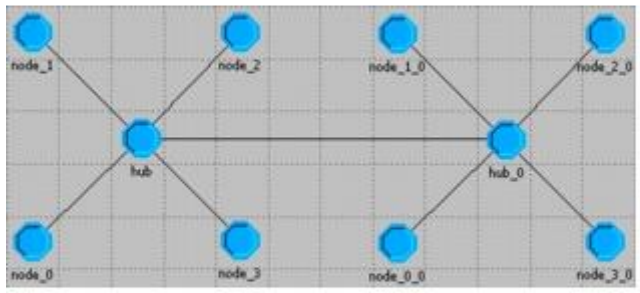
This tutorial assumes that you completed the Packet Switching I tutorial.

In Packet Switching I, you created a simple packet switching network with a hub and four peripheral nodes.

After running the simulation, you discovered that packets went across the network with or without queuing delays, depending on the packet generation rate.

In this lesson, you will see how adding a second network affects the end-to-end delays of packets in the network.

The Expanded Network



You will also accomplish the following goals:

- Learn about logical associations in node models
- Redesign the hub model to handle an additional subnet

Getting Started

Begin by opening the project you created in the Packet Switching I lesson.

1. Close any editors you still have open from the last lesson.
2. Choose **File** > **Open...** and verify that the "Files of type:" menu (Windows) or Filters menu (Linux) is set to **Project Files (*.prj)**.
3. If the displayed directory is not your active working directory (typically **<username>/op_models**), then navigate to it.
4. Choose **<initials>_pksw_net** and click **Open**.
5. Select **Scenarios > Duplicate Scenarios...** and name the new scenario **dual_subnet**.

Enhancing the Node Model

To accommodate more connections, the hub node model must have an additional point-to-point transmitter and receiver.

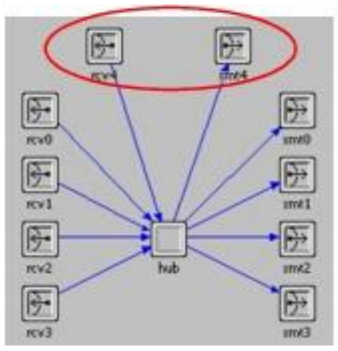
Because the expanded network will connect through the hub, you must add a transmitter/receiver pair to the hub node model.

Adding the Transmitter/Receiver Pair

To add a transmitter/receiver pair, perform the following steps:

1. Choose **File** > **Open...** and change the "Files of type" menu (Windows) or Filters menu (Linux) to **Node Model Files (*.nd.m)**. Then select the **<initials>_pksw_hub** model and click **Open**.
2. Add a point-to-point **transmitter** and **receiver**, as shown in the next figure.

Transmitter and Receiver Added to the Hub Node



3. Set the **name** attributes to **rcv4** and **xmt4**.
4. Connect the new modules to the hub with packet streams.

5. Right-click on the packet stream from the hub to xmt4 and select **Edit Attributes**. Note that the src stream is [4], which means that in the process model, the processor should send packets to the transmitter over stream 4.
6. Close the Attributes dialog box.

Setting the Channel Data Rate and Packet Formats

Set the channel data rate and supported packet formats for both new modules:

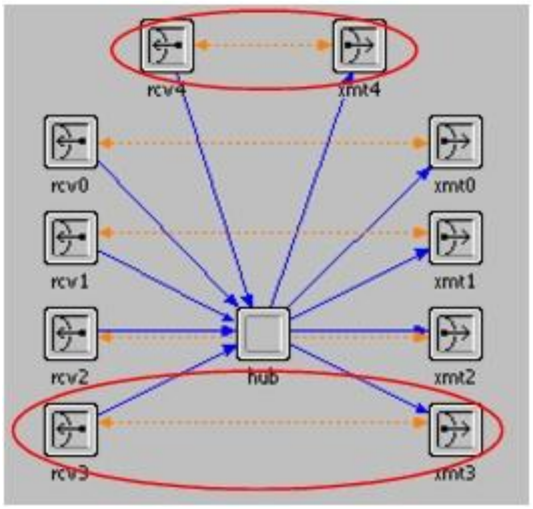
1. Select the new transmitter.
2. Right-click on the transmitter and select **Edit Attributes**.
3. Click on the value of the **channel** attribute.
4. Change the data rate to **9600**.
5. Support the **<initials>_pksw_format** format.
    a. Click on the value of the **packet formats** attribute.
    b. Click on both checkboxes to turn off **Support all packet formats** and **Support unformatted packets**.
    c. Click on the **Status** field for **<initials>_pksw_format**.
    d. Close the **Select Supported Packet Formats** dialog box.
    e. Close the **(channel) Table** dialog box.
6. Click **OK** to close the **Attributes** dialog box.
7. Repeat steps 2 through 6 for the receiver module.

Associating Transmitters and Receivers

To ensure the correct transmitter/receiver pairs are used when connecting links to nodes at the network level, you can logically associate pairs of transmitters and receivers:

1. Click the **Create Logical Tx/Rx Association** tool button.
2. Connect the modules as shown in the following figure:

Two of Five Logical Associations Circled



3. When you are finished, save the node model and close the Node Editor.

Adding a New Subnet

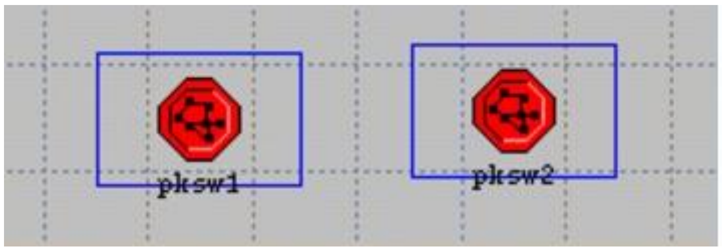
Now that an extra transmitter/receiver pair is available, the network model can be expanded to connect an additional subnet using point-to-point links.

Adding the Second Subnet

To add a second subnet to the existing network, you can copy the original network and connect the two.

1. If the **<initials>_pksw_net** project is not already open, open it.
2. Click on the **Go to Parent Subnetwork** tool button.
3. Left-click on the subnet to select it, select **Edit > Copy**, then **Edit > Paste**.
4. Position the cursor to the right of the existing subnet and left-click to place the second subnet.

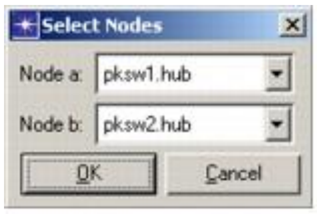
Placing the Second Subnet



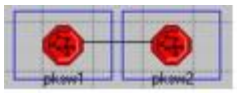
5. Open the **<initials>_pksw_palette** object palette.

6. Select the **<initials>_pksw_link** link model in the **<initials>_pksw_palette** object palette and connect the two subnets. Click **OK** in the **Select Nodes** dialog box.

   Connecting the Hubs of the Two Subnets

   The Link Appears Between the Selected Subnets

Adding a user id Attribute

You also need to add a **user id** attribute so each subnet can be uniquely identified when routing packets.

1. Right-click on the **pksw1** subnet and choose **Edit Attributes (Advanced)** from the pksw1 pop-up menu.
2. Change the **user id** attribute to **1**, then click **OK**.
3. Right-click on the **pksw2** subnet and choose **Edit Attributes (Advanced)** from the pksw2 pop-up menu.
4. Change the **user id** attribute to **2**, then click **OK** to close the dialog box.

   The user IDs are assigned to both subnets.

5. **Save** the project, then close the Project Editor.

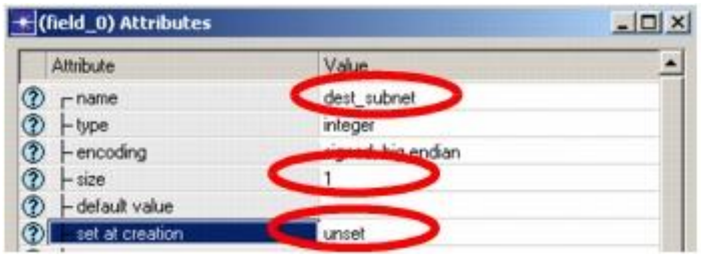
Redefining the Packet Format

There are several ways to deal with the problem of node addressing in different subnets. Adding a packet field that identifies the subnet of the destination node is an easy method.

1. Choose **File >** Open... and change the "Files of type" menu (Windows) or Filters menu (Linux) to **Packet Format Files (*.pk.m)**. Select the **<initials>_pksw_format** and click **Open**.

2. Click on the **Create New Field** tool button, then place the new field directly beneath the **dest_address** field.
3. Right-click on the new field and select **Edit Attributes**, then change the attributes as follows:
   - **name** = dest_subnet
   - **size** = 1 (We only need 1 bit to choose between the two subnets.)
   - **set at creation** = unset

   Changing the Attributes



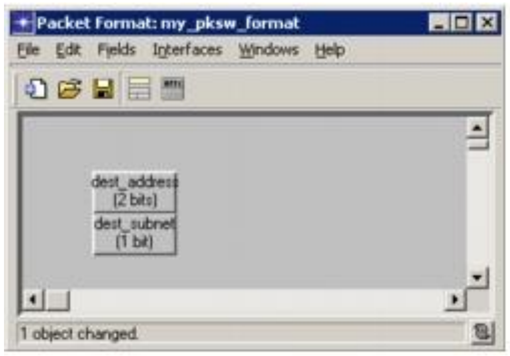
(field_0) Attributes

Attribute	Value
name	dest_subnet
type	integer
encoding	signed, big endian
size	1
default value	
set at creation	unset

4. Click **OK** when you are finished to close the dialog box.

   The name and size of the packet field are displayed in the field.

   Packet with New Packet Field



1 object changed.

5. **Save** the packet format and exit the Packet Format Editor.

Modifying the Process Model

Before the new network configuration will work, the hub and peripheral node process models must be modified to handle the routing of packets between the two subnets.

Peripheral nodes must assign a destination subnet and destination node address. The subnet address will be either **1** or **2** (these values correspond to a subnet **user id** attribute). A new line of code in the **xmt transition executive** will assign a destination subnet randomly.

Modifying the Node Process Model

1. Open the **<initials>_pksw_nd_proc** model in the Process Editor.
2. Open the **Function Block** and add the line of code marked in bold text to the xmt() function:

```
. . .
   pkptr = op_pk_get (SRC_IN_STRM);

   op_pk_nfd_set_int32 (pkptr, "dest_subnet",
      (int)op_dist_uniform (2.0) + 1);

   op_pk_nfd_set_int32 (pkptr, "dest_address",
      (int)op_dist_outcome (address_dist));

   op_pk_send (pkptr, XMT_OUT_STRM);

FOUT;

}
```

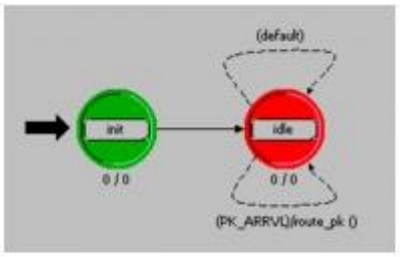
3. The **dest_subnet** field in the packet format has been defined as a 1-bit field. It is going to store the user ID of the destination subnet, i.e. a value of 1 or 2. In a real packet, storing a value of `2' would need more than 1 bit, but in the simulated space, we are just using two arbitrary values that are using just 1 bit of packet space, regardless of their actual binary representation.
4. Select **File > Save** from the **Function Block** menu.
5. Compile the process model. You do not need to save the model because it is saved before the compilation.

Modifying the Hub Process Model

With the addition of the second subnet, the hub node process model must now also determine which subnet a packet belongs in and route packets accordingly. An initial state must be added in which the process model determines its surrounding subnet.

1. Open the **<initials>_pksw_hub_proc** model in the Process Editor.
2. Add a new state to the left of the idle state. Make it an initial state by selecting it and clicking the **Set Initial State** tool button.
3. Change the **name** attribute to **init** and change the **status** attribute to **forced**.

Adding a New State and Transition



4. Create an unconditional transition from the **init** state to the **idle** state, as shown above.

Modify the code in the process model as follows:

1. Open the Header Block and add the line of code marked in bold text:

```
#define PK_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM)
#define OTHER_SUBNET_STRM 4
```

The **OTHER_SUBNET_STRM** symbolic constant refers to the stream attached to the newly created transmitter linked to the other subnet. This is the same as the src stream index value of the packet stream from the hub to xmt4.

2. Close the Header Block and save your changes.
3. Open the **state variable block** and add the variable **subnet_id** of type **int.**
4. Open the **init** state's **Enter Executives** and add the following lines of code:

```
Objid parent_subnet;
parent_subnet = op_topo_parent (op_topo_parent
   (op_id_self ()));
op_ima_obj_attr_get_int32 (parent_subnet, "user id",
   &subnet_id);
```

The first statement determines the object identifier of the parent subnet using chained calls to the **op_topo_parent()** procedure.

The second statement uses this value as an argument to the **op_ima_obj_attr_get_int32()** procedure to obtain the **user id** attribute of the subnet.

5. Close the Enter Executives and save your changes.
6. Open the **Function Block** and add the new lines of code marked in bold text to the **route_pk()** function:

```
int dest_address;
int dest_subnet;
Packet *pkptr;
FIN (route_pk());
pkptr = op_pk_get (op_intrpt_strm ());
op_pk_nfd_get_int32 (pkptr, "dest_subnet", &dest_subnet);
if (dest_subnet == subnet_id)
   {
   op_pk_nfd_get_int32 (pkptr, "dest_address",
      &dest_address);
   op_pk_send (pkptr, dest_address);
   }
else
   op_pk_send (pkptr, OTHER_SUBNET_STRM);
```

The modified **route_pk()** function sends the packet to a local node if the destination subnet value matches the **subnet_id** obtained in the **init** state. Otherwise, the packet is sent to the other subnet.

7. Select **File > Save** from the **Function Block** menu.
8. **Compile** the process model, then close the two open Process Editors.

Running the Simulation

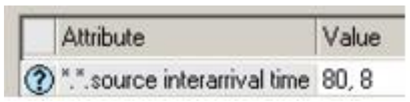
Now that all the enhancements to the model are complete, you can run the simulations.

1. In the **<intials>_pksw_net** Project Editor, select **DES > Configure/Run Discrete Event Simulation...**

   Note—If the **Configure/Run Discrete Event Simulation** dialog box is in **Simple** view (without a treeview in the left pane), click the **Detailed** button in the lower-left corner.

2. Click **Common** in the treeview and change the **Simulation set name** to **<initials>_pksw_out80** (in the right panel).
3. Verify that **Duration** is set to **1000** seconds.
4. Select and expand the **Inputs** node, then click on **Object Attributes**.
5. Delete the **pksw1.*.source interarrival time** attribute, as follows:
   a. Click the attribute name in the **Attribute** column to select the attribute.
   b. Click the **Delete** button.
6. Add ***.*.source interarrival time**.
7. In the **Simulation Set** dialog box, set the value for ***.*.source interarrival time** to **80, 8 using the Enter Multiple Values... button**.

Setting the Value of *.*.source.interarrival time to 80, 8



Attribute	Value
..source interarrival time	80, 8

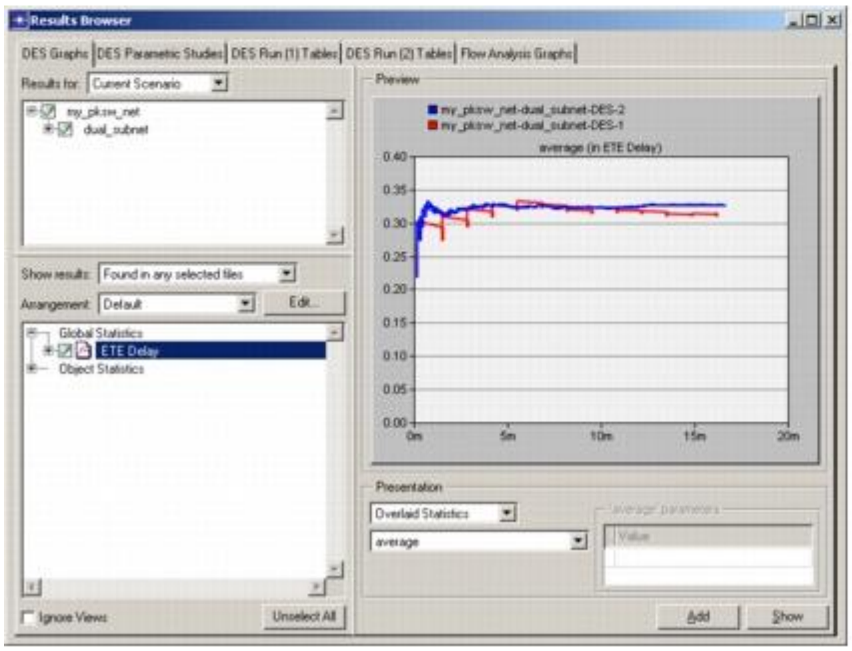
8. Click **Run** to save and execute the new settings. The simulation runs should take a few seconds to run.
9. When the simulation completes, close the DES Execution Manager dialog box. If you had problems, see *"Troubleshooting Tutorials"*.

Viewing Results

You can now determine how the addition of a second subnet affects the simulation results by reviewing the graph of ETE delay:

1. Right-click in the workspace and select **View Results**.
2. Select the **Global Statistics > ETE Delay** global statistic.
3. Make sure the graph is set to **Overlaid Statistics** and change the graph type to **average**.

Choosing ETE Delay Statistics

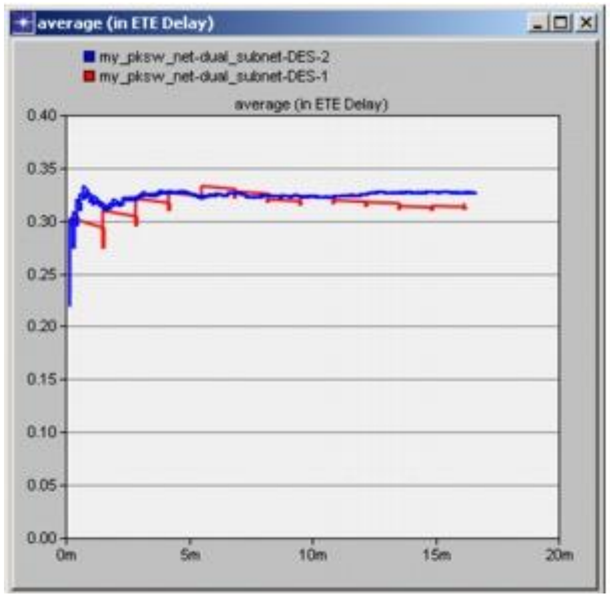


4. Click **Show**.

   The graph opens in a new window.

ETE Delay should resemble the following graph. Your results may differ somewhat because of the position of the nodes in your model.

ETE Delay Graph



Notice both simulations converge to steady state after initial spikes. Further, ETE Delay is lower for the first simulation, which is experiencing less traffic. These results seem reasonable.

You might also want to look at other graphs, such as the time average end-to-end delay. When you are finished with this tutorial, close all open editors.

Return to the list of tutorials in the Contents pane and choose **CSMA/CD**.

Práctica 14

CDMA/CD

Introduction

This tutorial presents detailed examples that illustrate the modeling and analysis of the well-known **Aloha** and **CSMA** channel access protocols. In this lesson, you will learn how to:

- Construct more advanced protocols
- Design a simple channel interface to a multi-tap bus
- Execute parametric simulations
- Analyze the simulated results against theoretical predictions

You will build two models: an **Aloha** model and a **CSMA** model. Because it is the simplest of the channel access methods, we will build the **Aloha** model first.

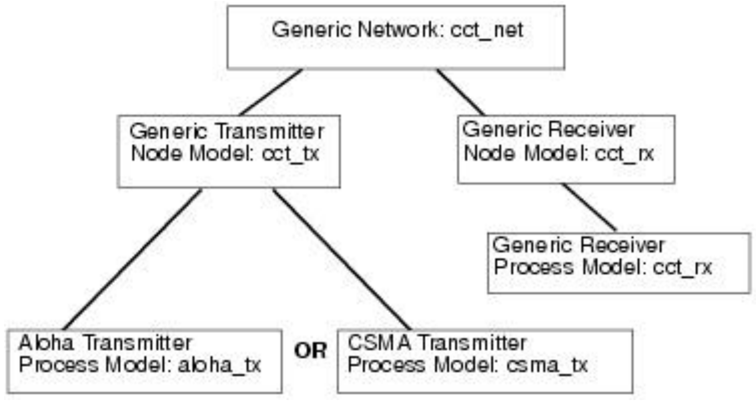
The main task is to design models that incorporate the **Aloha** random-channel-access method and the **1-persistent** carrier-sense-multiple-access (**CSMA**) method on a multi-tap bus link, where multiple nodes are connected through a shared channel. We will compare the performance of each method.

Getting Started

Before designing the models, you may be interested in an overview of the model hierarchy.

The design strategy for the **Aloha** and **CSMA** models is to employ the same network model. Both network models will use a common transmitter node model that sends packets and a common receiver node model that performs network monitoring. By changing the process model attribute of the node models, new simulations using either **Aloha** or **CSMA** properties can be built quickly. The transmitter node process models will be unique, whereas the receiver node process model is generic and will remain unchanged.

Aloha and CSMA Modeling Hierarchy

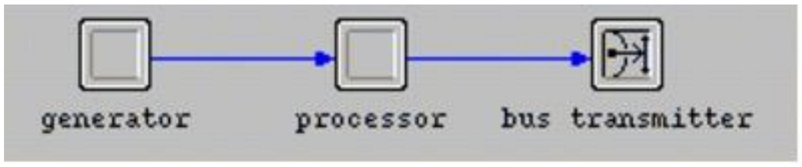

Generic Network: cct_net
Generic Transmitter Node Model: cct_tx
Generic Receiver Node Model: cct_rx
Generic Receiver Process Model: cct_rx
Aloha Transmitter Process Model: aloha_tx
OR
CSMA Transmitter Process Model: csma_tx

Designing the Generic Transmitter Node Model

In theory, the **Aloha** system could be modeled as just a simple source generator and a bus transmitter. However, by designing a more generalized model, you can reuse it later for the **CSMA** model.

The transmitter node must generate packets, process them, and send them on to the bus. This can be modeled using a simple source processor to generate packets, another processor to perform any necessary operations, and a bus transmitter to transmit the packets on the bus link.

Generic Transmitter Node Model



Bus transmitters also have internal queuing capability— they will issue all submitted packets onto the bus in FIFO order.

Designing the Aloha Transmitter Process Model

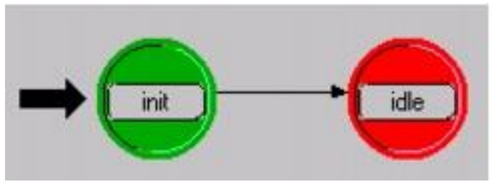
The Aloha transmitter process only has to receive packets from the generator and send them on to the transmitter.

The Aloha transmitter process has only one unforced state: waiting for the arrival of a packet from the generator. Because the generic transmitter node does not gather statistics, the **aloha_tx** process does not need to initialize or maintain state or global variables of its own. It does, however, need to retrieve a global attribute value that defines the number of generated packets. The transmitter process will retrieve this value once, before entering the main loop.

The process begins the simulation in a forced initialization state, then moves to an unforced idle state where it waits for packets to arrive.

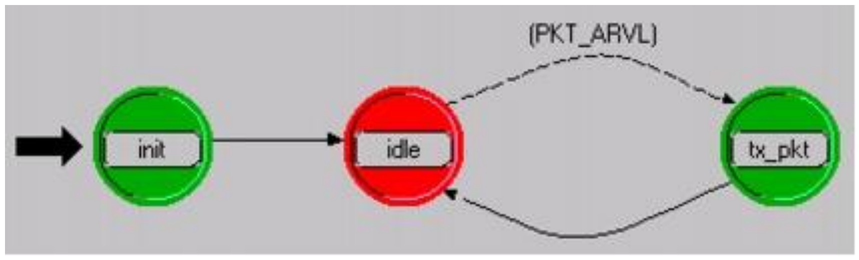
The process is activated with a begin simulation interrupt so that when the simulation starts, the FSM executes the forced initialization state and then waits in the idle state, ready to transition when the first packet arrives.

Intermediate aloha_tx FSM



There is only one distinct event in the **aloha_tx** FSM— the arrival of a generated packet. At the unforced idle state, the packet arrival interrupt can be selectively detected by an appropriate transition.

Complete aloha_tx FSM



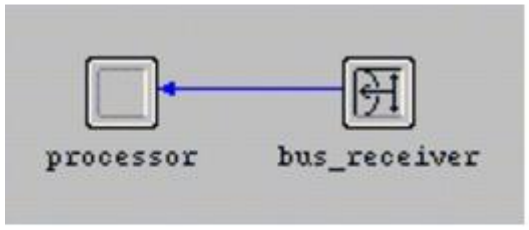
Packet arrival interrupts are the only interrupts expected, so it is safe to omit a default transition for the unforced idle state. When a packet arrival interrupt is delivered, the FSM should perform executives to acquire and transmit the packet in the **tx_pkt** state, then transition back to the **idle** state.

Designing the Generic Receiver Node Model

The generic receiver node model monitors the movement of packets across the bus.

The next step is to design the generic receiver node model. The model does not require a generator because it simply monitors packets moving across the bus. The node model consists of a bus receiver and a processor module.

Conceptual Generic Receiver Node Model

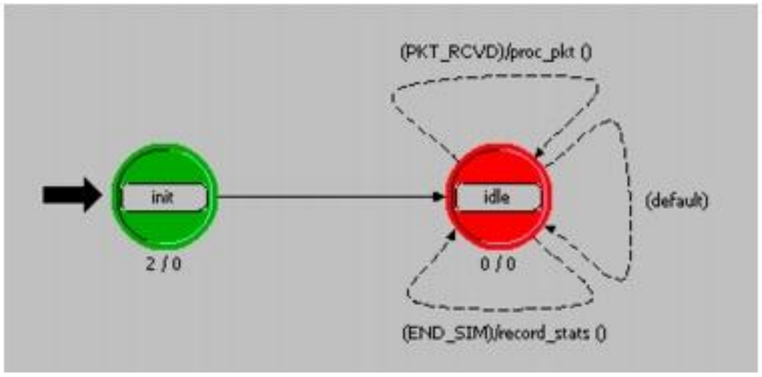


Designing the Generic Receiver Process Model

The generic receiver process model is responsible for handling received packets for statistics-gathering purposes.

To process received packets for statistics collection, the **cct_rx** process needs one unforced state where it waits to receive collision-free packets (how the collisions are detected is presented later in this tutorial). At the end of the simulation, the process records the channel throughput and channel traffic values for analysis. Because the receiver node process manages the statistics-gathering variables, the process should initialize the variables at the start of the simulation. This leads to the design shown. Note the reference to the user-defined C functions **proc_pkt()** and **record_stats()** in the transition executives (these will be written later).

Complete cct_rx FSM



Building the Aloha Model

The **Aloha** process and node models will be created first. These models serve as the basis for an enhanced model that will be used to represent the **CSMA** system.

Building the Aloha model involves several steps:

- Creating the Aloha transmitter process model
- Creating a generic transmitter node model
- Creating a generic receiver process model
- Creating a generic receiver node model
- Building the network model

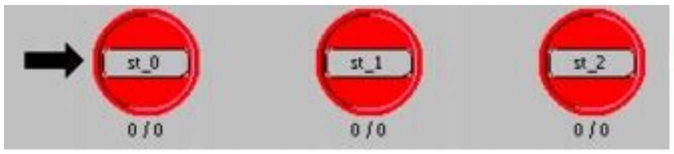
Creating the Aloha Transmitter Process Model

You will build the Aloha transmitter process model first:

1. Start OPNET Modeler if it is not already running.
2. Choose **File > New...** and select **Process Model** from the pull-down menu. Click **OK**.

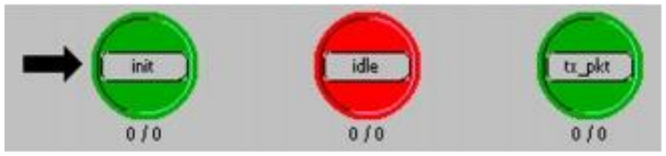
3. Using the **Create State** tool button, place three states in the workspace.

   Initial States of the Transmitter Process Model



4. Make the following changes to the three states, from left to right:
   a. To the first state, change the **name** attribute to **init** and the **status** to **forced**.
   b. To the second state, change the **name** attribute to **idle**. Leave the **status** as **unforced**.
   c. To the third state, change the **name** attribute to **tx_pkt** and the **status** to **forced**.

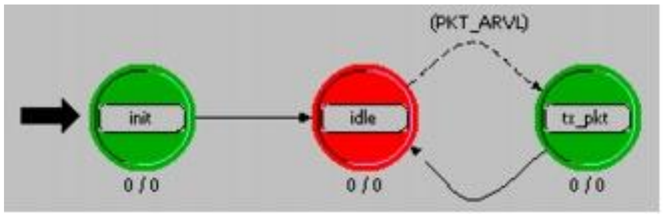
   States of the Transmitter Process Model



Next, add the transitions between the states:

1. Draw the three transitions as shown.

   Transitions of the Transmitter Process Model



2. For the transition from **idle** to **tx_pkt**, change the **condition** attribute to **PKT_ARVL** (using capital letters). To move the condition label, left-click on the label and drag it to a new position.

The **PKT_ARVL** macro determines if an interrupt received by the process is associated with a packet arriving on a stream. In this model, interrupts are only expected on the input stream from the generator, so the macro does not need to determine which input stream received the packet.

You will define this macro in the next step.

You are now ready to specify the code for the process model. Start with the Header Block:

1. Open the **Header Block** and enter the following code.

```
/* Input stream from generator module */
#define IN_STRM    0

/* Output stream to bus transmitter module */
#define OUT_STRM   0

/* Conditional macros */
#define PKT_ARVL (op_intrpt_type() == OPC_INTRPT_STRM)

/* Global Variable */
extern int subm_pkts;
```

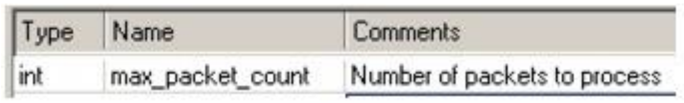
2. Save the changes.

The symbolic constants **IN_STRM** and **OUT_STRM** will be used in calls to Kernel Procedures that get packets from streams or send packets to streams. To achieve the desired functionality, these stream indices must be consistent with those defined at the node level.

Next, enter the state variables:

1. Open the **State Variable Block** and enter the following information. The default type, **int**, is acceptable.

Values for State Variable Block



Type	Name	Comments
int	max_packet_count	Number of packets to process

2. Click **OK** to close the dialog box.

The variable **max_packet_count** will hold the maximum number of packets to be processed in the simulation. This will be retrieved from a simulation attribute and compared with the packet count.

Define the actions for the init state in its Enter Executives block:

1. Double-click on the top of the **init** state to open the Enter Executives block and enter the following code.

```
/* Get the maximum packet count, */
/* set at simulation run-time */
op_ima_sim_attr_get_int32 ("max packet count",
          &max_packet_count);
```

2. Save your changes.

Also, specify the actions for the **tx_pkt** state:

1. Double-click on the top of the **tx_pkt** state to open the Enter Executives block, and enter the following code:

```
/* Outgoing packet */
Packet *out_pkt;
/* A packet has arrived for transmission.  Acquire */
/* the packet from the input stream, send the packet */
/* and update the global submitted packet counter. */
out_pkt = op_pk_get (IN_STRM);
op_pk_send (out_pkt, OUT_STRM);
++subm_pkts;

/* Compare the total number of packets submitted with */
/* the maximum set for this simulation run.  If equal */
/* end the simulation run. */
if (subm_pkts == max_packet_count)
          op_sim_end ("max packet count reached.", "", "", "");
```

2. Save your changes.

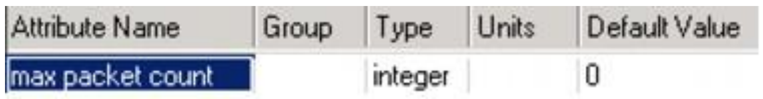
The **tx_pkt** state executive is entered when the process receives a stream interrupt from the Simulation Kernel. This interrupt coincides with the arrival of the generated packet. After completing the executives of the **tx_pkt** state, the FSM transitions back to the **idle** state. Because there are no other unforced states in the transition path, the FSM always re-enters the **idle** state before the next packet arrives and waits for the packet.

The **cct_rx** process model will later declare the global variable, **subm_pkts**, to globally accumulate the number of transmitted packets. Access to this variable in the **aloha_tx** process model is gained by declaring it in the model's Header Block using the C language extern storage class.

Next define the global attribute that will be set at simulation run-time and loaded into the state variable **max_packet_count**.

1. Choose **Interfaces > Global Attributes**.
2. Enter an attribute "max packet count" into the dialog box table, as shown:

Defining the Global Attribute



Attribute Name	Group	Type	Units	Default Value
max packet count		integer		0

3. Save your changes by clicking on the **OK** button.

The model is now complete, except for the model interface attributes.

You must also edit the process interfaces:

1. Choose **Interfaces > Process Interfaces**.
2. Change the initial value of the **begsim intrpt** attribute to **enabled**.
3. Change the **Status** of all the attributes to **hidden**.

You may want to add a comment to describe the process. When finished, click **OK** to close the dialog box.

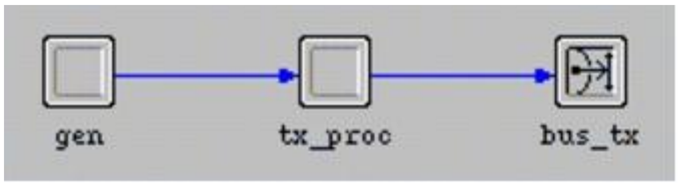
1. **Compile** the process model. Supply the name **<initials>_aloha_tx**.
2. When the process model is finished compiling, close the Process Model Editor.

Creating the Generic Transmitter Node Model

You'll now create a node model of a generic transmitter that can support either Aloha or CSMA.

1. Choose **File > New...** and select **Node Model** from the pull-down menu. Click **OK**.
2. Using the appropriate tool buttons, create two processor modules and one bus transmitter module. (Display the tooltip to verify that you selected a **bus** transmitter.)

Modules of the Generic Transmitter Node Model



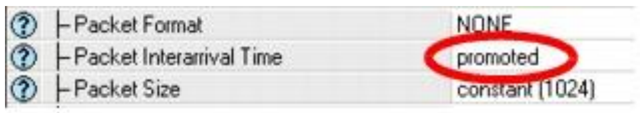
3. For each module, set the name attribute with the names shown above.
4. Connect the modules with packet streams as shown above.
5. Open the packet streams' attribute dialog boxes to see that **src stream** is set to **src stream [0]** and **dest stream** is set to **dest stream [0]**, conforming to the indices declared in the **<initials>_aloha_tx** process model Header Block.

Because you are interested in assigning different values to the generator's **interarrival time** attribute, you must promote it so its value can be set more easily at simulation time.

1. Open the **gen** processor's attribute dialog box.
2. Set the **process model** attribute to **simple_source**.
3. Click on **Packet Interarrival Time** in the left column to highlight the attribute name, then right-click and select **Promote Attribute to Higher Level** from the pop-up menu.

   The word **promoted** appears in the Value cell of the attribute.

   Promoting the Attribute



Packet Format	NONE
Packet Interarrival Time	promoted
Packet Size	constant (1024)

4. Click **OK** to close the attribute dialog box.

You also need to set the processor's attributes appropriately:

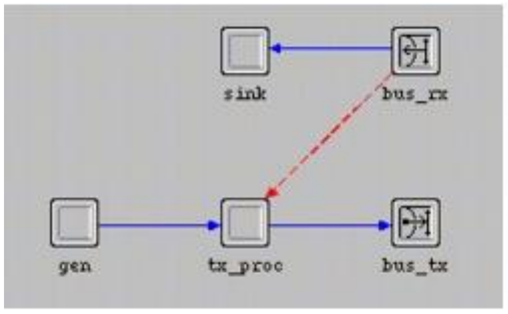
1. Open the attribute dialog box for **tx_proc** and set the **process model** attribute to **<initials>_aloha_tx**.
2. Click **OK** to close the dialog box.

Enhancing the Generic Transmitter Node Model

The generic transmitter node model you just created has the functionality necessary for the underlying **aloha_tx** process model. However, because you plan to exchange CSMA for the Aloha process model, it is useful to build hooks for the anticipated enhancements.

The enhancements consist of a bus receiver module (to support the eventual full duplex capability of the CSMA protocol), and a sink processor to accept and destroy packets received by the receiver module. The enhancements also include an inactive (disabled) statistic wire which, when enabled in the CSMA model, both inform the process (contained in the **tx_proc** module) of the busy status of the channel and provide interrupts to the process when the channel condition changes.

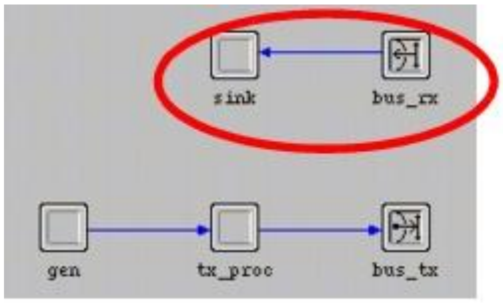
Enhanced Transmitter Node Model



Add the following features to the node model:

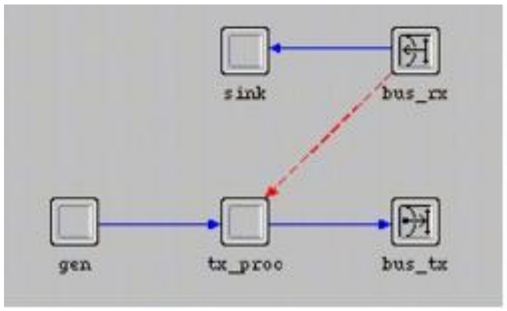
1. Using the appropriate tool buttons, add one processor module and one bus receiver module.

   Adding Modules



2. Change the **name** of the new processor module to **sink** and the **name** of the bus receiver to **bus_rx**.
3. Connect the new modules with a packet stream as shown.
4. Using the **Create Statistic Wire** tool button, connect the **bus_rx** module with the **tx_proc** module.

   Adding a Statistic Wire

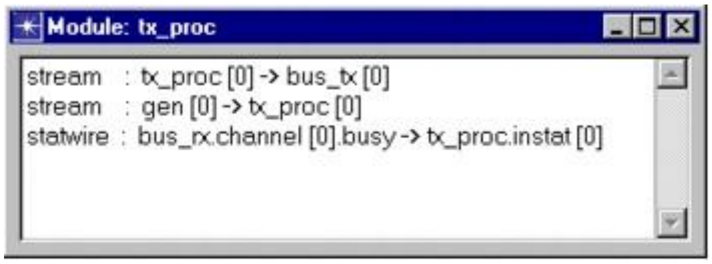


5. Open the attribute dialog box for the **statistic wire** and change both the **rising edge trigger** and **falling edge trigger** attributes to **disabled**. Click **OK** to close the dialog box when finished.

Double-check the module connectivity to make sure all objects in the model have been connected in the correct order:

1. Right-click on the **tx_proc** module and choose **Show Connectivity** from the Object pop-up menu. The objects should be connected as shown in the following figure.

   Checking Connectivity



```
stream   : tx_proc [0] -> bus_tx [0]
stream   : gen [0] -> tx_proc [0]
statwire : bus_rx.channel [0].busy -> tx_proc.instat [0]
```

2. If the connections do not match the figure, modify the connectors as follows:
    a. Right-click on the packet stream between the **gen** and **tx_proc** modules.
    b. Choose **Edit Attributes**.
    c. Change the value of the **src stream** attribute to **src stream** [0].
    d. Click **OK** to close the **Attributes** dialog box.
    e. Right-click on the statistic wire between the **bus_rx** and **tx_proc** modules.
    f. Choose **Edit Attributes**.
    g. Change the value of the **dest stat** attribute to **instat [0]**.
    h. Click **OK** to close the **Attributes** dialog box.

Next, define the interface attributes and write the completed model to disk.

1. Choose Interfaces > Node Interfaces.
2. In the **Node types** table, change the **Supported** value to **no** for the **mobile** and **satellite** types.
3. Change the **Status** of all the attributes to **hidden**, except for the one with **promoted** status, **gen.Packet Interarrival Time**.
4. If you would like, add a comment to describe the node. When you are finished, click **OK** to save the changes.
5. Save the model as **<initials>_cct_tx** and close the Node Editor.

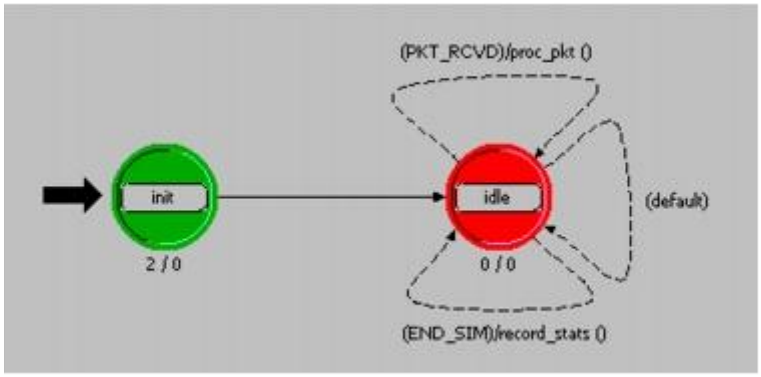
Creating the Generic Receiver Process and Node Models

Next, you can create the generic receiver process and node models. Because the sole purpose of the receiver process is to count packets and record statistics, it can be used to monitor network performance whether the packets are transmitted in accordance with the **Aloha** or the **CSMA** channel access methods.

1. Choose **File > New...** and select **Process Model** from the pull-down menu. Click **OK**.
2. Using the **Create State** tool button, place two states in the tool window.
3. For the initial state, change the **name** attribute to **init** and the **status** to **forced**.
4. For the other state, change the **name** attribute to **idle**. (Leave the **status** as **unforced**.)

Draw the four state transitions shown in the following figure.

Adding Transitions to the Generic Receiver Node



1. For the first (top) transition from **idle** back to itself, change the **condition** attribute to **PKT_RCVD** and the **executive** attribute to **proc_pkt()**.
2. For the second (middle) transition from **idle** back to itself, change the **condition** attribute to **default**.
3. For the third (bottom) transition from **idle** back to itself, change the **condition** attribute to **END_SIM** and the **executive** attribute to **record_stats()**.

Next, enter the code for the Header Block and the state variables.

1. Using the appropriate tool button, open the **Header Block** and type in the definitions shown.

```
/* Input stream from bus receiver */
#define IN_STRM 0

/* Conditional macros */
#define PKT_RCVD (op_intrpt_type () == OPC_INTRPT_STRM)
#define END_SIM (op_intrpt_type () == OPC_INTRPT_ENDSIM)

/* Global variable */
int subm_pkts = 0;
```

2. Save the Header Block.

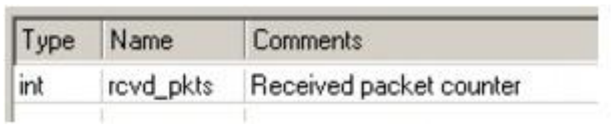
The index for the input stream from the bus receiver module (**IN_STRM**) is defined here. The **PKT_RCVD** macro determines if the interrupt delivered to the process is a stream interrupt. Only one kind of stream interrupt is ever expected, so no further qualifications are necessary. The **END_SIM** macro determines if the interrupt received by the process is associated with an end-of-simulation interrupt from the Simulation Kernel.

The global variable **subm_pkts** is used so that all transmitting nodes can contribute their individual transmission attempts to this accumulator. Declaring a variable in a process model Header Block causes it to behave as a global variable within the executable simulation.

The generic receiver process uses the **rcvd_pkts** state variable to keep track of the number of valid received packets. Define this variable as follows:

1. Open the **State Variables** block and define the following variable:

Defining the rcvd_pkts State Variable



Type	Name	Comments
int	rcvd_pkts	Received packet counter

2. Save the state variables block.

Next, enter the code that defines the functionality of the process model.

1. Open the **Function Block** and enter the following code:

```
/* This function gets the received packet, destroys  */
/* it, and logs the incremented received packet total        */
static void proc_pkt (void)
          {
          Packet* in_pkt;
          FIN (proc_pkt());
          /* Get packet from bus receiver input stream */
          in_pkt = op_pk_get (IN_STRM);

          /*Destroy the received packet */
          op_pk_destroy (in_pkt);

          /* Increment the count of received packet */
          ++rcvd_pkts;
          FOUT;
          }

/* This function writes the end-of-simulation channel        */
/* traffic and channel throughput statistics to a            */
/* vector file
static void record_stats (void)
          {
          double cur_time;
          FIN (record_stats());
          cur_time = op_sim_time();
          /* Record final statistics */
          op_stat_scalar_write ("Channel Traffic G",
                   (double) subm_pkts / cur_time);
          op_stat_scalar_write ("Channel Throughput S",
                   (double) rcvd_pkts / cur_time);
          FOUT;
          }
```

2. Save the Function Block.

As defined earlier in the Function Block, the **proc_pkt()** function acquires each received packet as it arrives, destroys it, and increments the count of received packets. The **record_stats()** function is called when the simulation terminates.

The **op_stat_scalar_write()** function sends the channel throughput and traffic data to the output vector file for the simulation run.

The init state initializes the state variable used to count received packets. Define it as follows:

1. Double-click on the top of the **init** state to open the **Enter Executives** block and enter the following code:

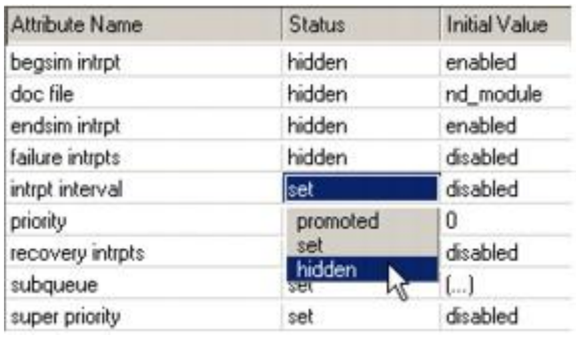
```
/* Initialize accumulator */
rcvd_pkts = 0;
```

2. Save the Enter Executives.

Finally, you can define the process interfaces.

1. Choose Interfaces > Process Interfaces.
2. Change the initial value of the **begsim intrpt** and **endsim intrpt** attributes to **enabled**.
3. Change the **Status** of all the attributes to **hidden**.

Hiding Attributes



Attribute Name	Status	Initial Value
begsim intrpt	hidden	enabled
doc file	hidden	nd_module
endsim intrpt	hidden	enabled
failure intrpts	hidden	disabled
intrpt interval	set	disabled
priority	promoted	0
recovery intrpts	set	disabled
subqueue	hidden	(...)
super priority	set	disabled

4. If you want, add a comment to describe the process, then click **OK** to save your changes.

Now compile the model.

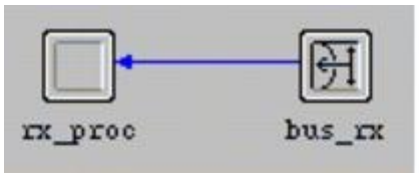
1. Click the **Compile Process Model** tool button.
2. Supply the file name **<initials>_cct_rx** and click on the **Save** button.
3. Close the compilation dialog box and the Process Model Editor.

Creating the Generic Receiver Node Model

The next step is to create a generic receiver node model.

1. Choose **File > New...** and select **Node Model** from the pull-down menu. Click **OK**.
2. Using the appropriate tool buttons, create one processor module and one bus receiver module. (Display the tooltip to verify that you selected a **bus** receiver.)

Modules of the Generic Receiver Node Model



3. For each module, change the **name** attribute as shown.
4. Connect the modules with a packet stream as shown.

   The input stream index defaults to stream 0, conforming to the index declared in the **cct_rx** process model Header Block.

5. Open the processor's attribute dialog box and set the **process model** attribute to **<initials>_cct_rx**. Close the dialog box when finished.

The generic receiver node model is now complete, except for the interface attributes.

1. Choose **Interfaces > Node Interfaces**.
2. In the **Node types** table, change the **Supported** value to **no** for the **mobile** and **satellite** types.
3. In the **Attributes** table, change the **Status** of all the attributes to **hidden**.
4. If you wish, add a comment to describe the node model. When you are finished, click **OK** to exit the dialog box.
5. **Save** the node model as **<initials>_cct_rx**, then close the Node Model Editor.

Creating a New Link Model

The behavior of a bus link is defined by its Transceiver Pipeline stages. The pipeline is a series of C or C++ procedures that can be modified to customize the link model.

For this lesson, you will create a custom bus link model whose pipeline stages use the default bus models, denoted by the **dbu_** model prefix. The following table lists pipeline stages by function.

Bus Transceiver Pipeline Model Stages

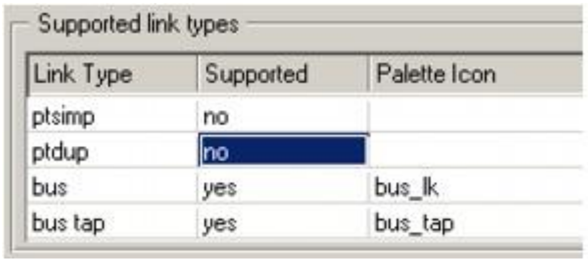
Model	Function
txdel	Computes the transmission delay associated with the transmission of a packet over a bus link (transmission delay is the time required to transmit the packet at the bit rate defined in the relevant bus transmitter module).
closure	Determines the connectivity between any two stations on the bus.
propdel	Calculates the propagation delay between a given transmitter and a receiver.
coll	Determines whether a packet has collided on the bus.
error	Calculates the number of bit errors in a packet.

ecc	Rejects packets exceeding the error correction threshold, as well as any collided packets.

To create a new bus link model:

1. Choose **File > New...** and select **Link Model** from the pull-down menu. Click **OK**.
2. In the **Supported link types** table, change the **Supported** value to **no** for the **ptsimp** and **ptdup** types.

   Modifying the Link Types Supported

   Supported link types

Link Type	Supported	Palette Icon
ptsimp	no	
ptdup	no	
bus	yes	bus_lk
bus tap	yes	bus_tap

   This link model supports only the bus and bus tap types.

3. If you wish, add a comment to describe the link.
4. Save the file as **<initials>_cct_link** and close the Link Model Editor.

Creating the Network Model

The network model will be built so that it can be used when analyzing both the Aloha and CSMA protocols. This will be done by defining the nodes so that they reference the generic node models, and later changing the referenced process models at the node level.

The analytical Aloha model assumes that packets are always introduced into the network at exponentially distributed interarrival times. However, in this tutorial, the network model has a finite number of nodes that hold packets in their buffers until the previous outstanding transaction finishes. To closely follow the analytical model's assumptions, there must be a large number of transmitter nodes on the bus.

The network model will be constructed within a subnet so that a small-scale coordinate system can be used.

1. Choose **File > New...** and select **Project** from the pull-down menu. Click **OK**.
2. Name the project **<initials>_cct_network** and the scenario **aloha**, then click **OK**.
3. In the Startup Wizard, use the following settings:

   Startup Wizard Settings

Dialog Box Name	Value
Initial Topology	Default value: **Create empty scenario**
Choose Network Scale	**Office** ("Use metric units" selected)
Specify Size	**700 x 700 Meters**
Select Technologies	None
Review	Check values, then click **Finish**

To build your network more easily, you need a custom palette that has the necessary objects. To create the palette:

1. In the object palette, switch to the icon view by clicking on the button in the upper-left corner of the dialog box.
2. Next, click on the **Configure Palette...** button.
3. In the **Configure Palette** dialog box, click **Clear**.

   All objects except the subnet are removed from the palette. If you have the Wireless module installed, you will also see the mobile and satellite subnets.

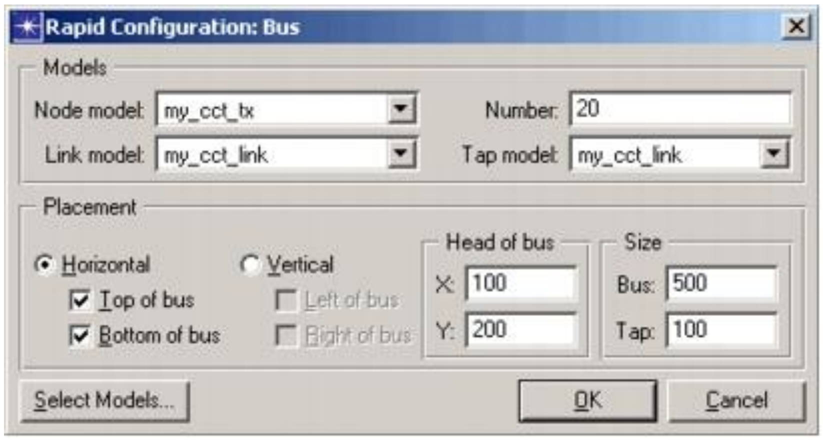
4. Click on the **Link Models** button, then add **<initials>_cct_link** from the list of available link models. Click **OK** to close the dialog box when you are finished.
5. Click on the **Node Models** button, then add **<initials>_cct_rx** and **<initials>_cct_tx** from the list of available node models. Click **OK** to close the dialog box when you are finished.
6. Save the object palette by clicking on the **Save As...** button in the **Configure Palette** dialog box. Use **<initials>_cct** as the file name.
7. Click **OK** to close the **Configure Palette** dialog box.

   The **<initials>_cct** Object Palette is ready for use.

Instead of creating the entire bus network by hand, you can use rapid configuration to build it quickly:

1. Choose **Topology > Rapid Configuration...**
2. Select **Bus** from the menu of available configurations, then click **Next...**
3. Use the values shown in the following figure to complete the **Rapid Configuration: Bus** dialog box.

   Rapid Configuration: Bus Dialog Box

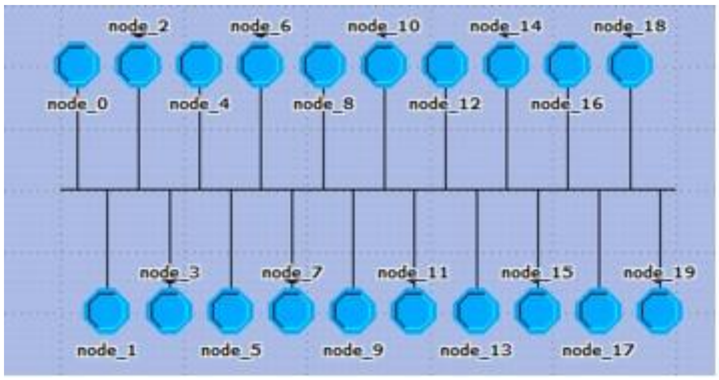


Node model: my_cct_tx
Number: 20
Link model: my_cct_link
Tap model: my_cct_link
[✓ Horizontal] Vertical
[✓] Top of bus
[✓] Bottom of bus
[ ] Left of bus
[ ] Right of bus
Head of bus — X: 100, Y: 200
Size — Bus: 500, Tap: 100

4. Click **OK** when all the values are entered.

   The network is drawn in the workspace.

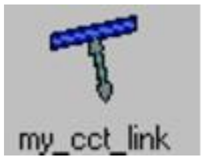
   Bus Network Created



This network still needs a receiver node. To add this node and connect it to the network:

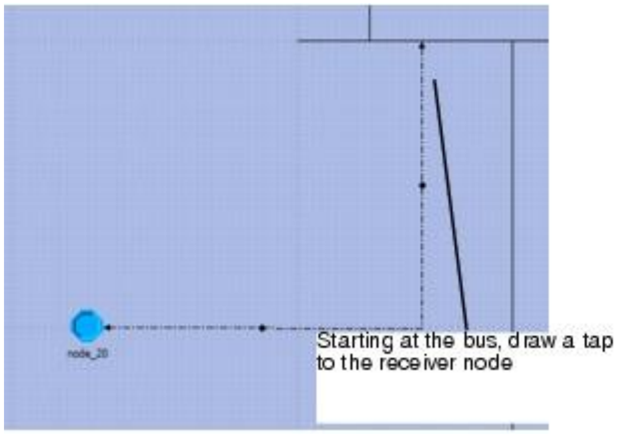
1. Click and drag the receiver node **<initials>_cct_rx** from the object palette into the left side of the workspace.
2. Click on the **<initials>_cct_link** tap link in the palette. Be sure to use the tap link.

   Bus Tap Icon



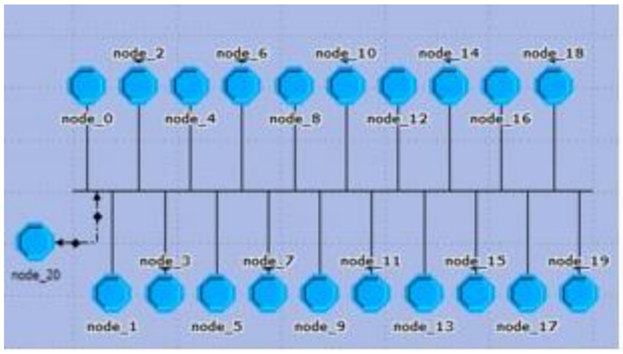
3. Draw a tap from the bus to the receiver node. Be sure to start at the bus. Drawing the tap from the node to the bus might produce different results.

Drawing the Tap



4. Verify that the completed bus model looks like this:

Completed Bus Model



5. Save the model with the default name, **<initials>_cct_network**, and close the object palette.

   Do not exit the Project Editor.

Executing the Aloha Simulation

The goal of this lesson is to observe how the performance of the protocols varies as a function of channel traffic. The interarrival time input parameter will be varied in a series of simulations to produce different levels of traffic and, therefore, different levels of throughput. You will run 12 simulations, each with a different interarrival time value, analyze the results, and draw conclusions.

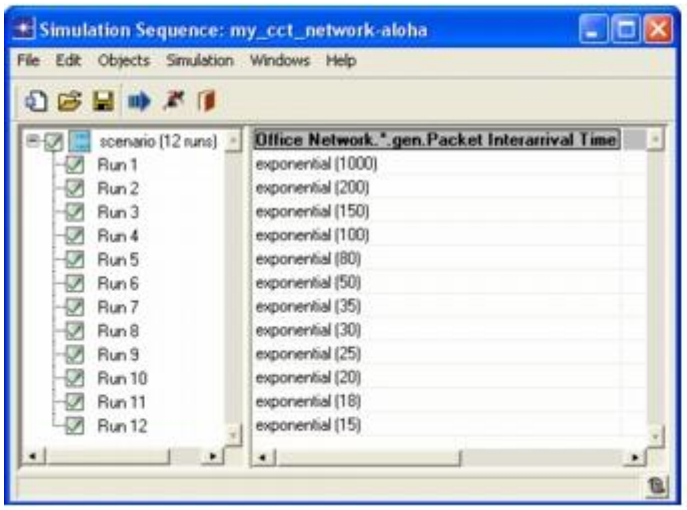
Importing and Configuring the Simulation Sequence

1. Choose **Scenarios > Scenario Components > Import...**

2. Select **Simulation Sequence** from the pull-down menu, then select **cct_network-CSMA**. Click **OK**.
3. Save the project.
4. Choose **DES > Configure/Run Discrete Event Simulation (Advanced)**.

   The **Simulation Sequence** dialog box opens. Notice that it displays a scenario with 12 simulation runs with the varying parameter for each. This is the file you imported.

   Simulation Sequence with 12 Simulation Runs


Simulation Sequence: my_cct_network-aloha
File Edit Objects Simulation Windows Help

scenario (12 runs)	Office Network.*.gen.Packet Interarrival Time
Run 1	exponential (1000)
Run 2	exponential (200)
Run 3	exponential (150)
Run 4	exponential (100)
Run 5	exponential (80)
Run 6	exponential (50)
Run 7	exponential (35)
Run 8	exponential (30)
Run 9	exponential (25)
Run 10	exponential (20)
Run 11	exponential (18)
Run 12	exponential (15)

   Note: You would normally open the Configure/Run DES dialog box (**DES > Configure/Run Discrete Event Simulation**) and create a simulation sequence file with specific settings. To save time, a nearly-complete file has been provided.

5. Right-click on the **scenario (12 runs)** tree node and select **Edit Attributes**.
6. Expand the **Execution** tree node.
7. Expand the **Advanced** tree node and select **Application**.
8. Click on the **Application** node.

   The Application panel opens in the right pane.

9. Verify that the **Network model** is set to **<initials>_cct_network-aloha**.
10. Click on the **Outputs** tree node, then the **Statistics Collection** tree node.
11. Set **Probe file** to **<NONE>**. You do not need to create or specify a Probe file. The **op_stat_scalar_write()** function substitutes for the Probe file.
12. Verify that the **Vector file** is set to **<initials>_cct_network-aloha**.

    This file will collect the output of the **op_stat_scalar_write()** function you included in the Function Block of the **<initials>_cct_rx** model. A file will be created for each run, with an appropriate -DES-<run #> suffix added to the network model name.

The scalar information is also written out in the vector file. In this tutorial, the vector file will contain only scalar data.

13. Click on the **Inputs** tree node, then the **Global Attributes** node, and verify that **max packet count** is **1000.**
14. Click on the **Object Attributes** tree node.

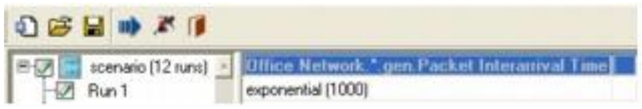
    In the Value column, notice the 12 values that have been set for the attribute **Office Network.*.gen.Packet Interarrival Time**.

15. Click **OK** to save changes and close the **Simulation Sequence** dialog box.
16. Choose **File > Save**.

The simulation can now be executed. Because the simulation sequence executes many individual simulations, the total execution time might be several minutes.

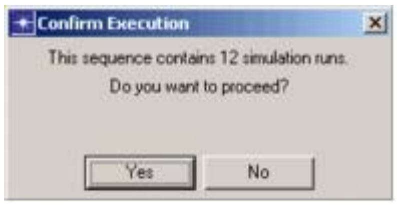
1. Make sure that all runs in the sequence are selected in the treeview.
2. Click on the **Running Man** tool button.

   Execute Simulation Sequence

3. Click **Yes** in the **Confirm Execution** dialog box. A sequence composed of many runs might be time-consuming to execute; this dialog box gives you the option of deferring the process.

   Confirm Execution Dialog Box

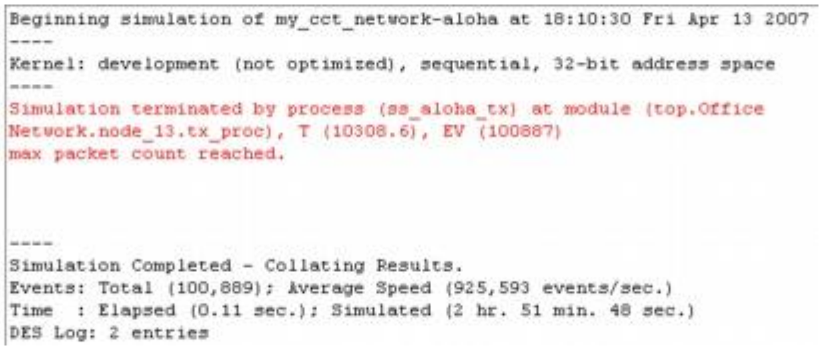
   

   Confirm Execution

   This sequence contains 12 simulation runs.
   Do you want to proceed?

   The DES Execution Manager dialog box opens, showing the progress through the 12 simulation executions. The 12 simulations display their progress as they execute. Any simulation run that generates 1000 packets (the value of **max packet count**) will terminate with a message similar to the one in the following figure:

Termination Message

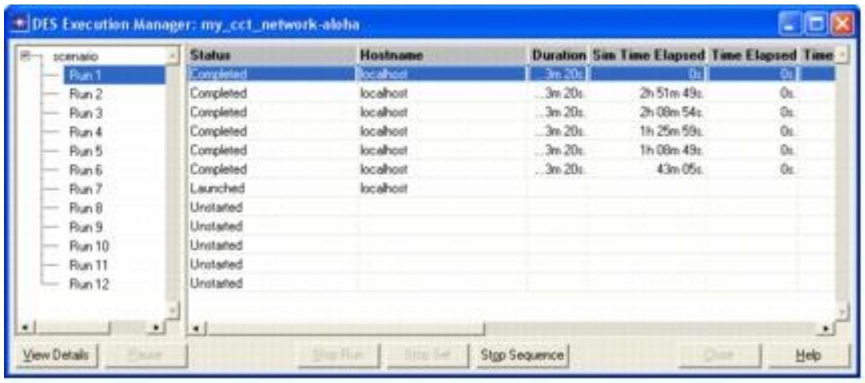


```
Beginning simulation of my_cct_network-aloha at 18:10:30 Fri Apr 13 2007
----
Kernel: development (not optimized), sequential, 32-bit address space
----
Simulation terminated by process (ss_aloha_tx) at module (top.Office
Network.node_13.tx_proc), T (10308.6), EV (100887)
max packet count reached.



----
Simulation Completed - Collating Results.
Events: Total (100,889); Average Speed (925,593 events/sec.)
Time  : Elapsed (0.11 sec.); Simulated (2 hr. 51 min. 48 sec.)
DES Log: 2 entries
```

DES Execution Manager Dialog Box



DES Execution Manager: my_cct_network-aloha

scenario	Status	Hostname	Duration	Sim Time Elapsed	Time Elapsed	Time
Run 1	Completed	localhost	3m 20s	0s	0s	
Run 2	Completed	localhost	...3m 20s	2h 51m 49s	0s	
Run 3	Completed	localhost	...3m 20s	2h 08m 54s	0s	
Run 4	Completed	localhost	...3m 20s	1h 25m 59s	0s	
Run 5	Completed	localhost	...3m 20s	1h 08m 49s	0s	
Run 6	Completed	localhost	...3m 20s	43m 05s	0s	
Run 7	Launched	localhost				
Run 8	Unstarted					
Run 9	Unstarted					
Run 10	Unstarted					
Run 11	Unstarted					
Run 12	Unstarted					

4. When the simulations are complete, close the **DES Execution Manager** dialog box and the Simulation Sequence Editor. If you had problems, see *"Troubleshooting Tutorials"*.

Analyzing the Aloha Results

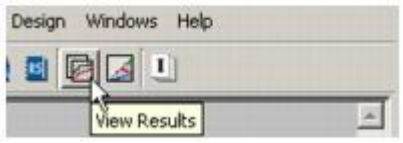
Aloha channel performance can be measured according to the number of successfully received packets as a function of the packets submitted, regardless of whether the packets are original or retransmitted. In this network, **channel throughput** is a typical measurement of network performance.

The results of each simulation are stored as two scalar values in the output vector file, allowing you to view the network's performance as a function of an input parameter rather than a function of time. The channel throughput as a function of channel traffic across all of the simulations can be viewed.

To view the scalar plot:

1. In the toolbar buttons of the Project Editor, click the **View Results** tool button.

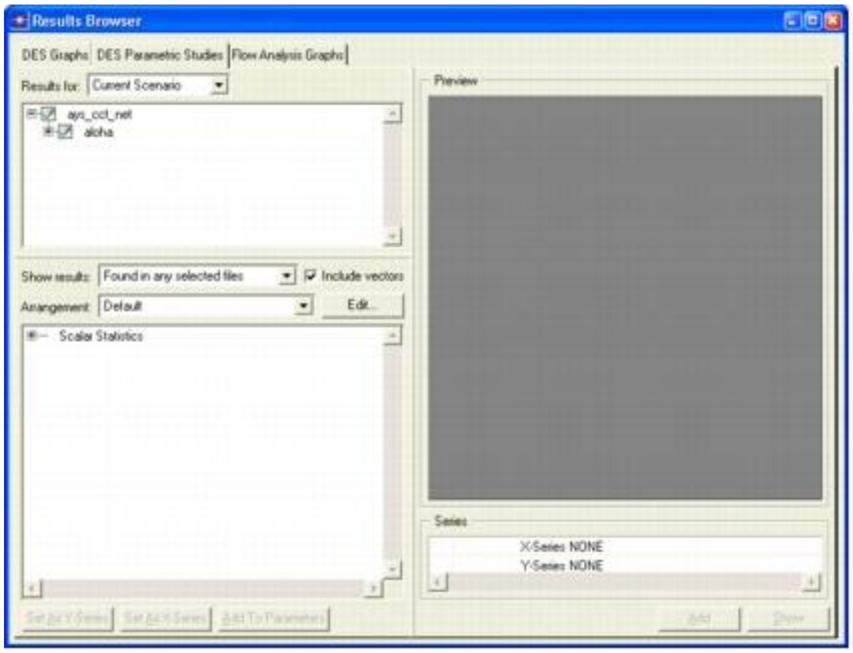
View Results Toolbar Button



The Results Browser opens.

2. Click on the **DES Parametric Studies** tab.

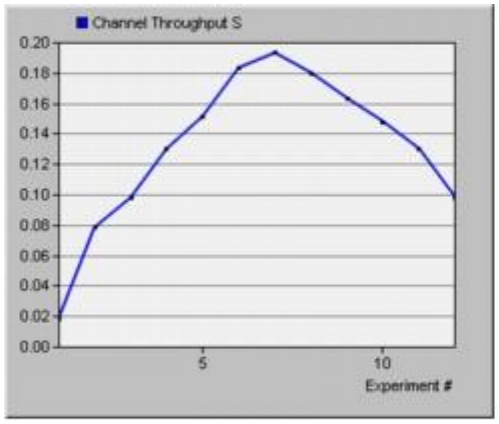
Parametric Studies Tab of the Results Browser



3. Expand the **Scalar Statistics** tree node.
4. Right-click on the **Channel Throughput S** scalar and select **Set as Y-Series** in the pop-up menu.

   The **Preview** area displays the scalar values for the various runs.

Preview of Channel Throughput S Data Across Runs



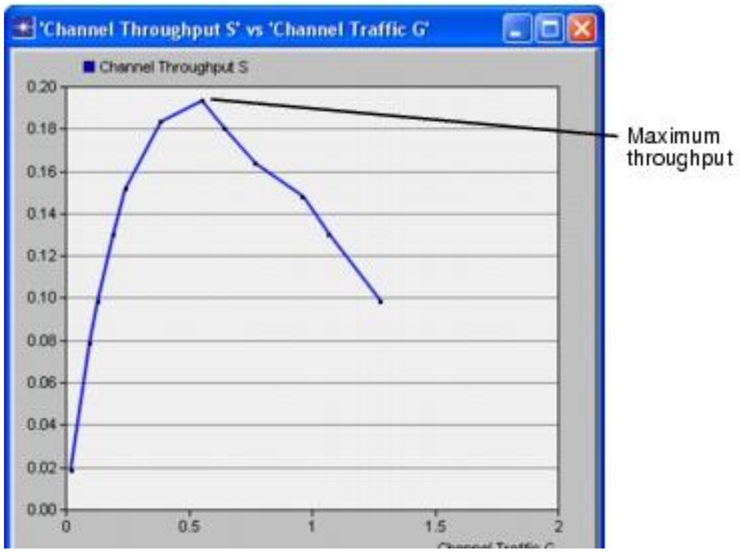
5. Right click on the **Channel Traffic G** and select **Set as X-Series**.

   The **Preview** area changes to show the resulting throughput as a function of traffic graph.

6. Click **Show**.

   The scalar graph appears in the workspace. Your graph should resemble the one in the following figure:

   Aloha Protocol: Channel Throughput as Function of Channel Traffic



Theoretical analyses have shown that a pure Aloha system has a channel throughput S as a function of channel traffic G given by $S = Ge^{-2G}$. This relationship gives a maximum channel throughput of $S_{max} = 1/(2e)$ ♠ 0.18.

At low traffic levels, collisions seldom occur. At high traffic levels, the channel is overwhelmed and excessive collisions prevent packets from being successfully received. This behavior is amply demonstrated by the simulation results. In particular, the maximum throughput is achieved near G = 0.5 and is close to the expected value of 0.18.

The theoretical results assume an essentially infinite number of sources to eliminate the buffering effects that emerge in a real network. The analytical model also assumes that the system is in an ideal steady state condition. Any differences in the measured performance of this model and the analytical models can be attributed to peculiarities of the random number seeds selected for individual simulations (which can be fixed by using multiple seeds) and the real-world limitations (including finite simulation time and finite number of nodes) imposed by the models.

When you are finished viewing the graph, close the graph panel and the Results Browser.

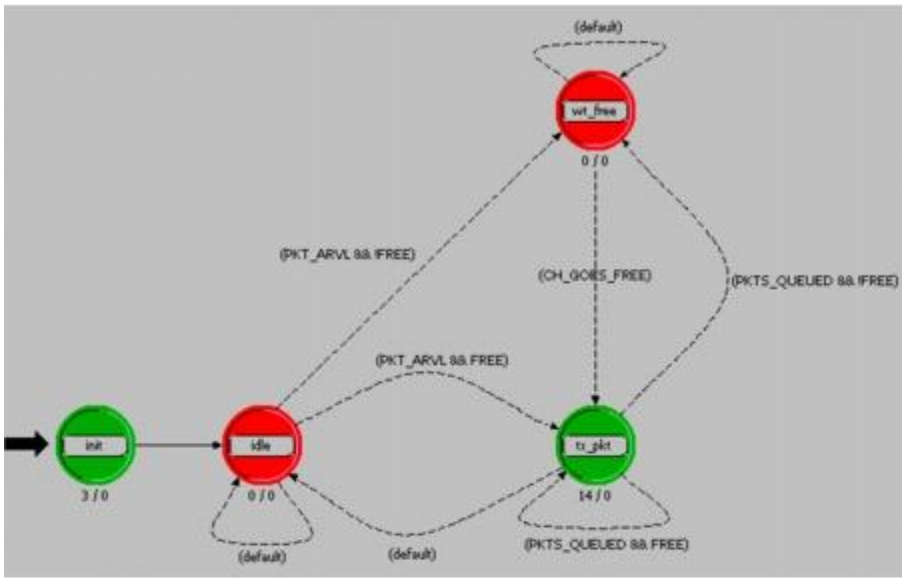
Adding Deference

The performance of the Aloha random access protocol can be enhanced by adding a carrier sense capability. The carrier sense capability is employed in the classical CSMA protocol, which requires a source node to sense the channel and determine that it is free before committing to a transmission.

You can enhance the existing **<initials>_aloha_tx** process model so that the process waits until the channel is free before transmitting a packet.

1. In the Project Editor, choose **File > Recent Files > Process Model** and select the **<initials>_aloha_tx** model.
2. Choose **File > Save As...** and rename the model **<initials>_csma_tx**.
3. Modify the states and transitions so that the model appears as shown in the following figure:

   The CSMA Process Model



a. Create a new state and name it **wt_free**.
b. Create a transition from **wt_free** to **tx_pkt**, and change the condition to **CH_GOES_FREE**.
c. Create a transition from the **wt_free** state back to itself and set the condition to **default**.
d. Create a transition from the **idle** state to **wt_free** and change the condition to **PKT_ARVL && !FREE**.
e. Add a transition from the **idle** state back to itself with a condition of **default**.
f. Change the condition on the transition from **idle** state to the **tx_pkt** state to **PKT_ARVL && FREE**.
g. Change the unconditional transition from **tx_pkt** to **idle** to conditional by setting the condition attribute to **default**.
h. Create a transition from **tx_pkt** back to itself, and set the condition to **PKTS_QUEUED && FREE**.
i. Finally, create a transition from **tx_pkt** to **wt_free** and set the condition to **PKTS_QUEUED && !FREE**.

Remember, you can move a condition label by left-clicking on the label and dragging it to a new position.

Editing the Header Block

You must change the Header Block so that the process verifies that the channel is free before transmitting. For the process to send a packet, it must first confirm that the channel is free by using the Kernel Procedure **op_stat_local_read()** to read the channel's **busy** statistic. If the channel is not free, the process enters the state **wt_free** until a "channel goes free" interrupt is received.

At the node level, the underlying statistic wire is triggered when the **busy** statistic changes to 0.0. The triggering is activated by enabling the wire's **falling edge trigger** attribute.

1. Add the following lines to the end of the process model Header Block.

```
/* input statistic indices */
#define CH_BUSY_STAT                                       0

/* Conditional macros */
#define FREE (op_stat_local_read (CH_BUSY_STAT) == 0.0)
#define PKTS_QUEUED (!op_strm_empty (IN_STRM))
#define CH_GOES_FREE (op_intrpt_type () ==
OPC_INTRPT_STAT)
```

2. Save the Header Block.
3. **Compile** the model, then close the Process Editor.

Enhancing the Generic Transmitter Node Model

You can enhance the generic transmitter node model so that the bus receiver module delivers a falling edge statistic interrupt to the processor module whenever the receiver **busy** statistic changes from "busy" (1.0) to "free" (0.0).

To enhance the generic transmitter node model so that it supports CSMA:

1. Select **File > Recent Files > Node Model** and select **<initials>_cct_tx**.
2. Choose **File > Save As...** and rename the model **<initials>_cct_csma_tx**.
3. Right-click on the **statistic wire** and choose **Edit Attributes** from the pop-up menu. Set the **falling edge trigger** attribute to **enabled**. Click **OK**.
4. Open the **Attributes** dialog box for the **tx_proc** processor module and change the **process model** attribute to **<initials>_csma_tx**. Click **OK** to close the dialog box.

   The processor now uses a process model that acts on **channel busy** statistic interrupts delivered by the receiver module.

5. Save the modified model and close the Node Editor.

Redefining the Network Model

Now that you have modified the appropriate models to support CSMA, you need to change the network model to use the new models. Instead of creating an entirely new model, you can duplicate the existing scenario (including the network model) and make the appropriate changes.

1. In the Project Editor, choose **Scenarios > Duplicate Scenario...** and name the new scenario **CSMA**.

The only change to the network model is to use the new CSMA transmitter nodes.

2. Right-click on one of the transmitter nodes and choose **Select Similar Nodes**.

   All 20 transmitter nodes are selected.

3. Right-click any of the selected nodes and choose **Edit Attributes (Advanced)** from the menu.
4. Check **Apply changes to selected objects**.
5. Change the **model** attribute to **<initials>_cct_csma_tx**.

   A dialog box appears to warn you about changes to an attribute on a node from the Standard Model Library.

6. Click **Yes**.

   The node models are changed to **<initials>_cct_csma_tx.** The phrase "20 objects changed" appears in the message buffer.

Configuring CSMA Simulations

Configure a series of simulations for the CSMA model.

1. Save the project.
2. Choose **DES > Configure/Run Discrete Event Simulation**.
3. Change the **Seed** to **11**.
4. Click **Run** to execute the simulation runs. It may take a few minutes to run the 12 simulations. When they complete, close the DES Execution Manager dialog box.

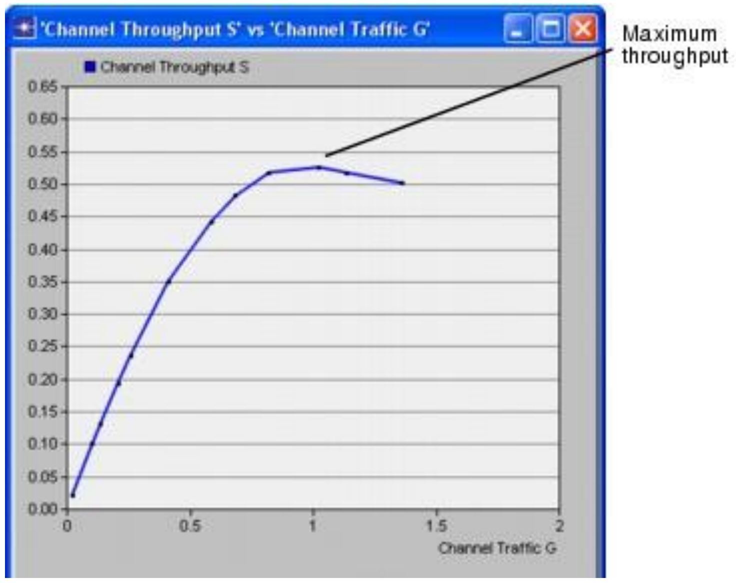
Analyzing the CSMA Results

View the results.

1. In the Project Editor, click on the **View Results** tool button.
2. In the Results Browser, click on the **DES Parametric Studies** tab if needed.
3. Expand the **Scalar Statistics**, right-click on the **Channel Throughput S** scalar, and choose **Set as Y-Series**.
4. Right-click on the **Channel Traffic G scalar and choose Set as X-Series**.
5. Click **Show**.

   Your graph should resemble the one in the following figure:

   CSMA Protocol: Channel Throughput as a Function of Channel Traffic



The CSMA protocol achieves a maximum channel throughput of about 0.53.

Viewing Both Results on the Same Graph

Your goal is to compare the Aloha and CSMA protocols. The easiest way to do so is to display both traces on the same graph.

1. In the **Results Browser**, change the **Results for:** pop-up to **Current Project**.

   The tree of result files shows the **aloha** and **CSMA** scenarios.

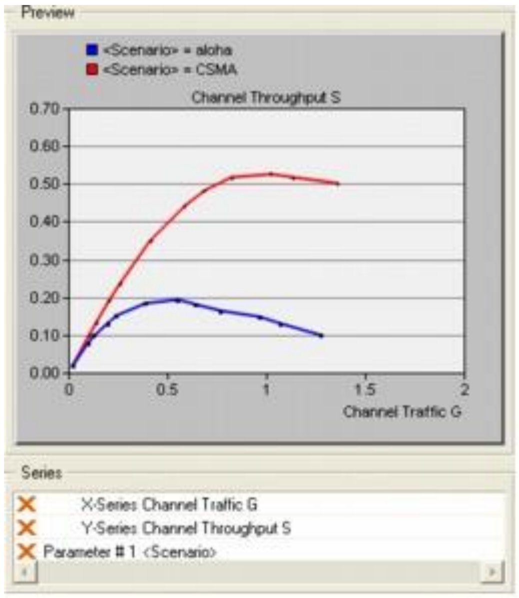
2. Select the **aloha** scenario.

   The preview graph changes to show results from both sets of runs.

3. Right-click in the **Series** table and choose **Add Scenario Name as Parameter**.

   The preview graph changes to show two graphs, one for each scenario based on its run output.

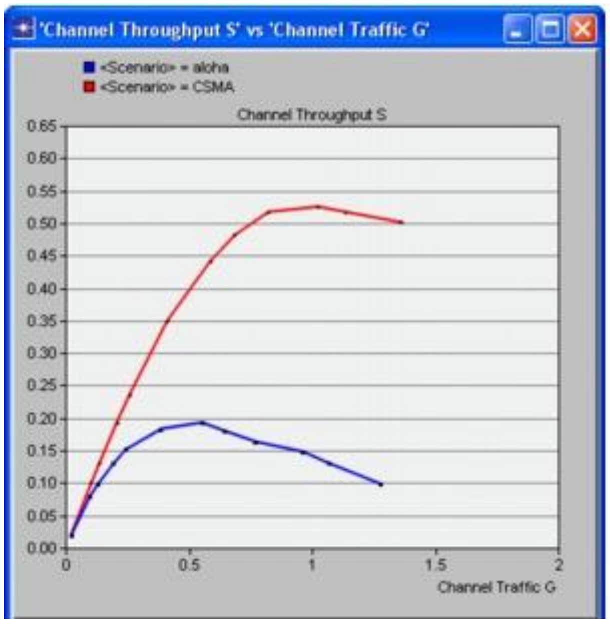
   Adding the Scenario Name as a Parameter



4. Click **Show**.

   The graph of the two scalars should resemble the following graph:

   Aloha and CSMA Protocols Compared



5. Close the graphs and the Results Browser.

Adding Collision Detection and Backoff

If a node has full-duplex capability, it can both transmit and monitor a connected bus link at the same time. This capability can be modeled using the Ethernet protocol.

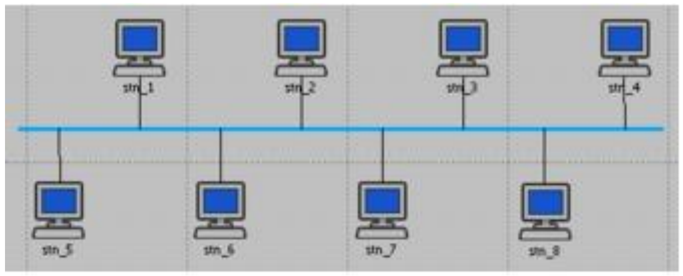
A node with full-duplex capability can both transmit and `listen' on the line to determine whether a collision condition exists. This operational mode is commonly referred to as Carrier-Sense Multiple Access with Collision Detection (or CSMA/CD). This is practiced by the commercial protocol Ethernet, and is accurately modeled by an OPNET-supplied example model.

Because Ethernet is a fairly sophisticated model, you will not build it yourself. Instead, the section provides a guided tour of the standard Ethernet process, node, and network models.

The ethcoax_net Network Model

The **ethcoax_net** network model consists of a multi-tap bus network populated by eight nodes. The nodes employ the node model **ethcoax_station_adv**.

The ethcoax_net Network Model

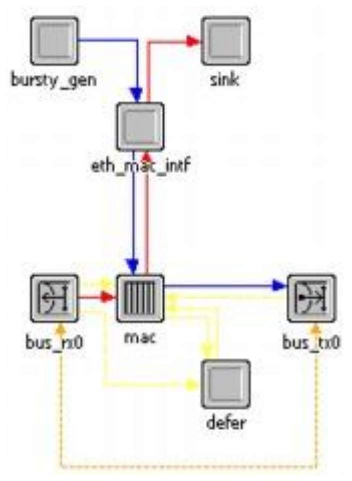


The ethcoax_station_adv Node Model

The **ethcoax_station_adv** node model is significantly more complicated than the Aloha or CSMA node models. It has four processor modules, a queue module that performs the bulk of the channel access processing, and a pair of bus receiver and transmitter modules.

The **ethcoax_station_adv** node model provides part of the functionality associated with the OSI Data Link Layer called the Media Access Control (MAC) sublayer. The functions of the individual modules are discussed in the following paragraphs.

The ethcoax_station_adv Node Model



The **bus_tx** and **bus_rx** modules serve as the bus link interface. These modules are set to transmit and receive at a data rate of 10 Mbits/second, the standard data rate used in an Ethernet network.

The **sink** processor represents higher layers and simply accepts incoming packets that have been processed through the **mac** process.

The **defer** processor independently monitors the link's condition and maintains a deference flag that the **mac** process reads over a statistic wire to decide whether transmission is allowed.

The **bursty_gen** module represents higher layer users that submit data for transmission. It uses an ON-OFF pattern for traffic generation.

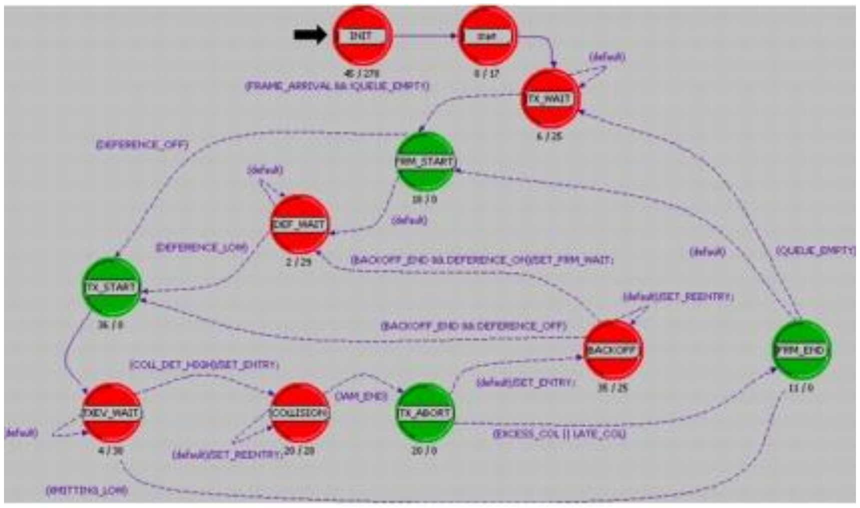
The **mac** process handles both incoming and outgoing packets. Incoming packets are decapsulated from their Ethernet frames and delivered to a higher level process. Outgoing packets are encapsulated within Ethernet frames and, when the deference flag goes low, a frame is sent to the transmitter. This process also monitors for collisions; if one occurs, the transmission is appropriately terminated and rescheduled for a later attempt.

The eth_mac_v2 Process Model

The **eth_mac_v2** process model manages the transmission and reception of packets. These tasks have been decomposed into three basic functions:

- encapsulating and queuing outgoing packets
- decapsulating and delivering incoming packets
- managing an ongoing transmission

The eth_mac_v2 Process Model

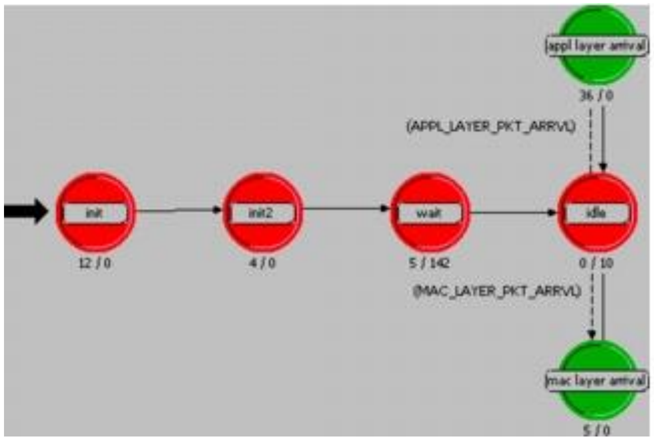


The ethernet_mac_interface Process Model

The **ethernet_mac_interface** process converts packets representing the application data into ethernet form for the mac processor.

The ethernet_mac_interface process takes packets from a traffic source, assigns a valid destination address (if random assignment is specified for traffic destination), and sends them to the mac processor. It also accepts packets from the mac processor and forwards them on to the higher layer traffic sink process.

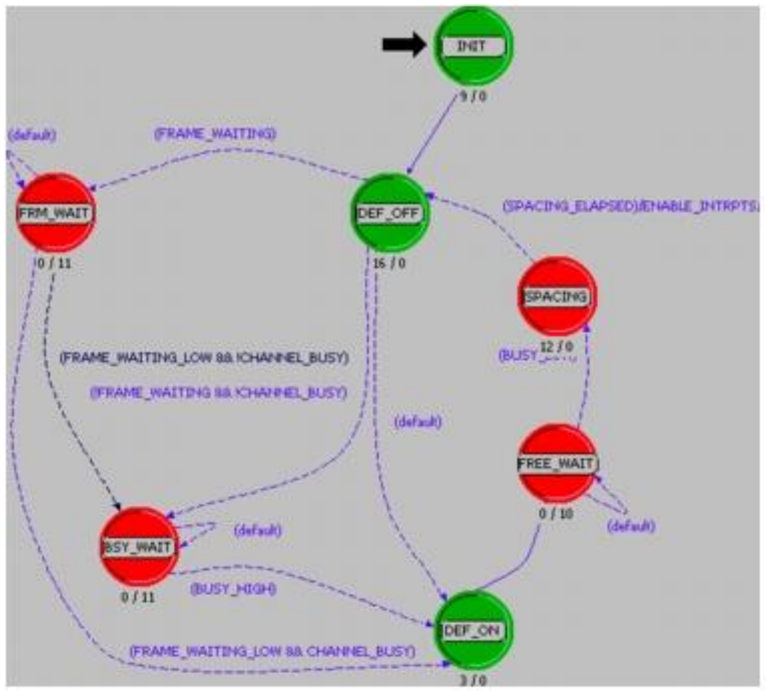
The ethernet_mac_interface Process Model



The eth_defer_v2 Process Model

The **eth_defer_v2** process determines whether the deference flag should be raised or lowered. The deference flag is read by the **eth_mac_v2** process to decide whether a transmission is permissible or whether the channel must be deferred to another user.

The eth_defer_v2 Process Model



Executing the ethcoax_net Simulation

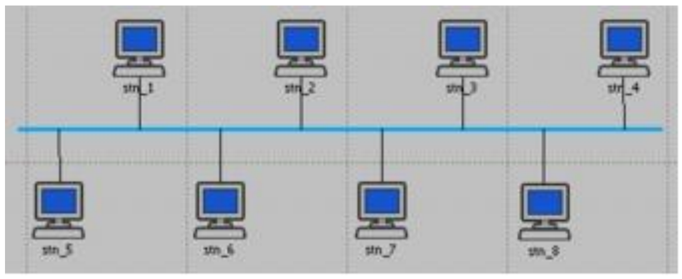
Load the pre-defined Ethernet model and run a simulation.

1. Choose **File > Open...**, select **Project**, and go to the **<install_dir>\<release>\models\std\ tutorial_req\modeler** directory.
2. Open the **ethcoax_net** project.

   The ethcoax_net model opens in the workspace.

3. Choose **File > Save As...** and save the project as **<initials>_ethcoax_net** in your default model directory.

   The ethcoax_net Model



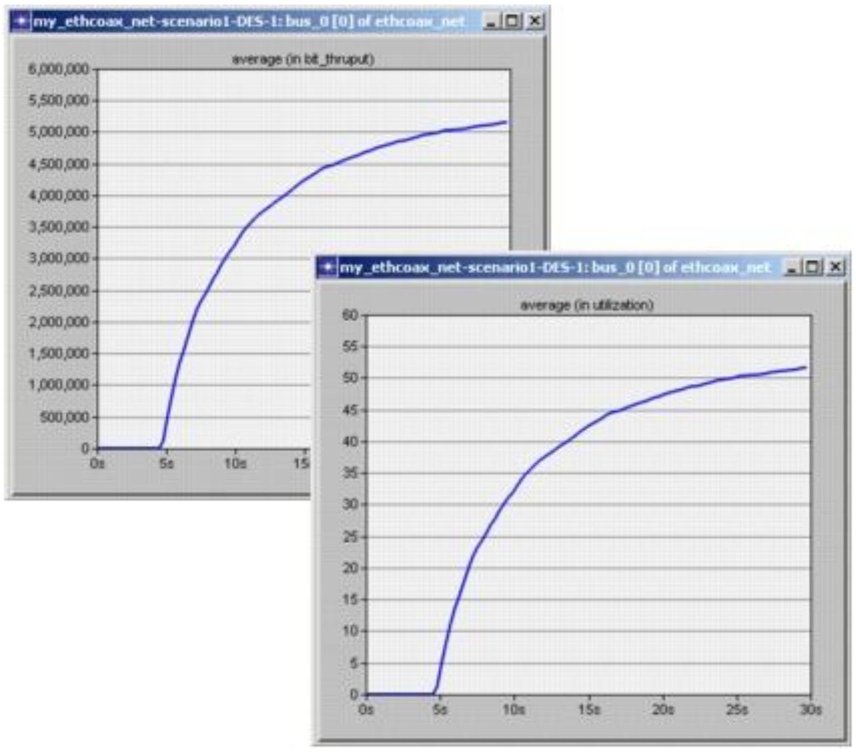
4. Choose **DES > Run Discrete Event Simulation**.
5. Close the simulation progress dialog box after the simulation ends.

Analyzing the Ethernet Results

Because the default Ethernet model collects results differently from the Aloha and CSMA simulations, you need to use a slightly different approach to view the results from this simulation.

1. In the Project Editor, choose **DES > Results > View Results...**
2. Select **Object Statistics > ethcoax_net > bus_0 [0] > utilization**.
3. Change the filter type from **As Is** to **average**, then click **Show**.
4. In the **Results Browser**, unselect **utilization** and select **bit_thruput**.
5. Click **Show**.
6. Place the graphs so you can see both clearly and consider the results. The graphs should resemble the following ones.

bit_thruput and utilization Graphs

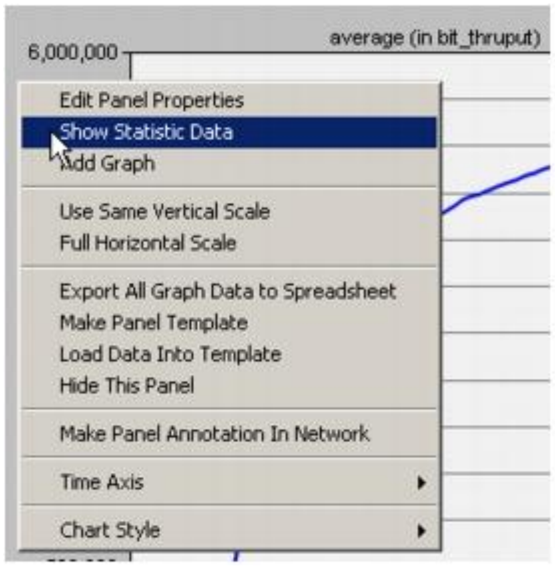


Though the general trend lines are the same, the graphs have radically different ordinate bounds.

The **bit_thruput** statistic measures the average number of bits successfully received by the receiver per unit time. By definition, this statistic only counts the bits associated with collision-free packets and can reach a maximum value of no more than 10 Mbits/second, the data rate assigned to the channel. As you can see, the throughput after 30 seconds of simulation stabilizes near 5.1 Mbits/second. To get a more accurate reading of the actual throughput, you can view the vector graph as a set of data points.

1. Verify that the **bit_thruput** panel is the active window.
2. Right-click in the panel border and choose **Show Statistic Data**.

   Selecting "Show Statistic Data"

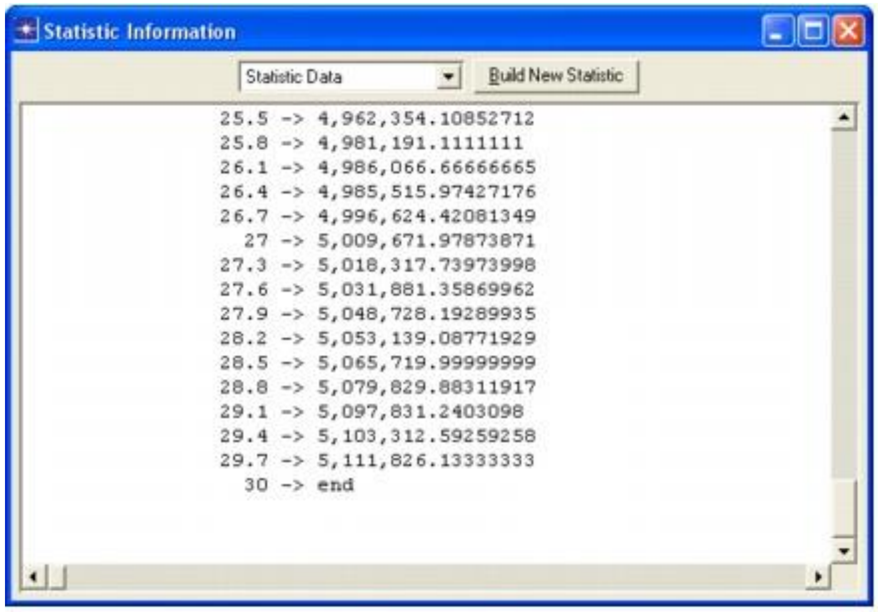
   

   A window opens, showing the sequence of ordinate and abscissa pairs for the vector, in ASCII format.

3. From the pull-down menu at the top of the **Statistic Information** dialog box, select **Statistic Data**.
4. Scroll to the bottom of the editing pad to read the final value of the vector.

The ASCII data should resemble that shown:

Bit Throughput at End of Simulation



```
25.5 -> 4,962,354.10852712
25.8 -> 4,981,191.1111111
26.1 -> 4,986,066.66666665
26.4 -> 4,985,515.97427176
26.7 -> 4,996,624.42081349
  27 -> 5,009,671.97873871
27.3 -> 5,018,317.73973998
27.6 -> 5,031,881.35869962
27.9 -> 5,048,728.19289935
28.2 -> 5,053,139.08771929
28.5 -> 5,065,719.99999999
28.8 -> 5,079,829.88311917
29.1 -> 5,097,831.2403098
29.4 -> 5,103,312.59259258
29.7 -> 5,111,826.13333333
  30 -> end
```

At the end of the simulation, the receiver's **bit_thruput** statistic is indeed almost exactly 5.1 Mbits/second. When divided by the channel capacity, this raw level of bit throughput results in a normalized channel throughput of 0.51 (that is, channel utilization of 51 percent).

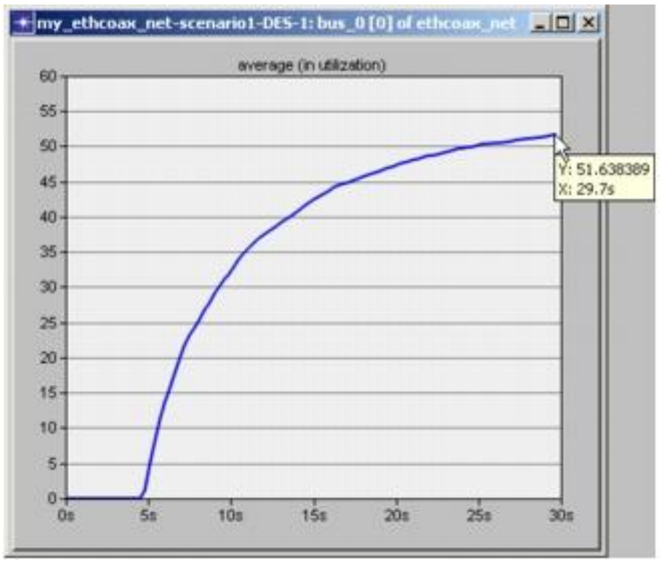
When you are finished viewing the data, close the **Statistic Information** dialog box.

You can also see these values on the utilization graph:

1. Click on the **average (in utilization)** graph to activate that window.
2. Move the cursor to the far right of the vector and let the cursor come to rest. The tooltip shows the final value of the channel utilization.

   You should see an average channel utilization of about 51 percent. This value is the percentage of the 10 Mbits/second channel that the probed transmitter uses.

   Average of Utilization Graph



This indicates that even when channel traffic is a relatively high 51 percent, the Ethernet protocol is able to carry essentially all of the submitted load. This also demonstrates the superiority of the carrier-sensing, collision-detection, and backoff strategies used by Ethernet over the less sophisticated methods used by the pure Aloha and CSMA protocols.

Congratulations! You have completed all of the OPNET Modeler tutorial lessons. By now, you should be able to build your own network model, collect statistics, run a simulation, and analyze the results on your own.

If you purchased additional modules, such as Multi-Vendor Import or ACE, continue with the tutorials that illustrate these capabilities. Return to the list of tutorials in the Contents pane and choose the desired tutorial from the list of available lessons.

From time to time, you may have questions about OPNET Modeler. Consult the documentation (**Help > Product Documentation**) first. You can also contact technical support by choosing **Help > Web - Support Center**.

Good luck model building!

Práctica 15

Creación de Procesos Básicos (Basic Processes)

Overview

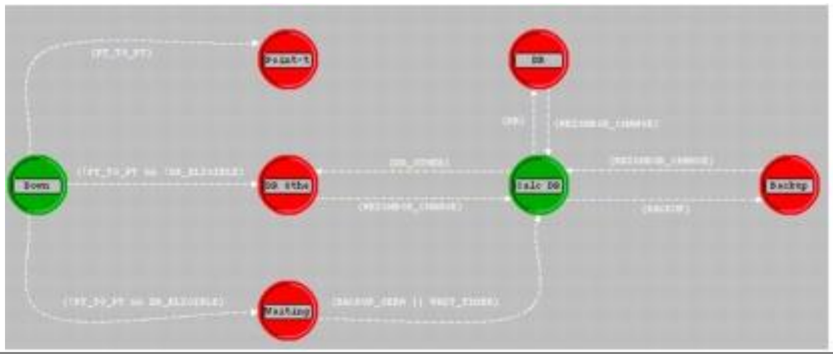
In the project editor, a network model is made up of individual nodes and a node is made of modules. The **process model** defines the behavior of a module. By understanding process models, you can build modules and combine them to build nodes to your exact specifications.

In this lesson, you will do the following things:

- Create process and node models
- Define variables, macros, and transitions
- Run a simulation
- Analyze simulation results

This lesson shows how to build a module that counts the packets it receives, then writes that number to a statistic that can be graphed. For each received packet, the process model increments the value of a variable and records the variable.

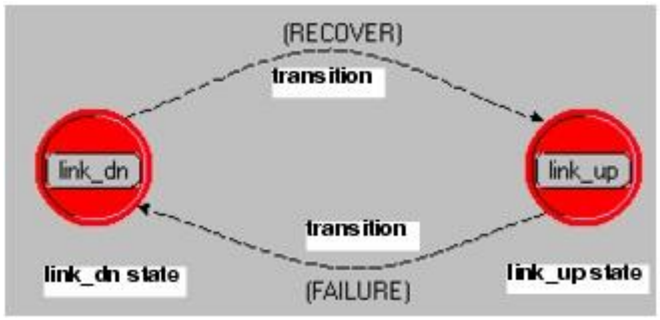
Example Process Model



A process model is a finite state machine (FSM). It represents the logic and behavior of a module. An FSM defines the states of the module and the criteria for changing the states.

An FSM implements the behavior of a module. FSMs use **states** and **transitions** to determine what actions the module can take in response to an event.

States and Transitions



- **State**—The condition of a module. For example, a module may be waiting for a link to recover.
- **Transition**—A change of state in response to an event.

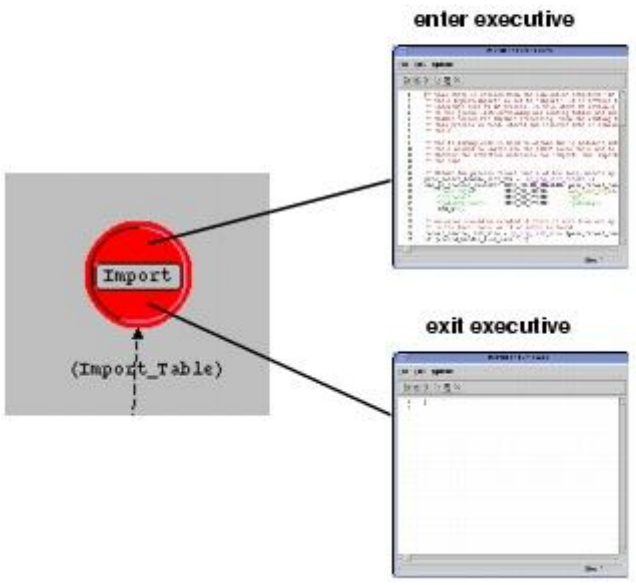
You can use an FSM to control module behavior because you can attach fragments of C/C++ code to states and transitions.

You can attach fragments of C/C++ code to each part of an FSM. This code, augmented by OPNET-specific functions, is called Proto-C.

The three primary places to use Proto-C are as follows:

- **Enter Executive**—code that is executed when the module moves into a state
- **Exit Executive**—code that is executed when the module leaves a state
- **Transition Executive**—code that is executed in response to a specific event

  Enter and Exit Executives of a State

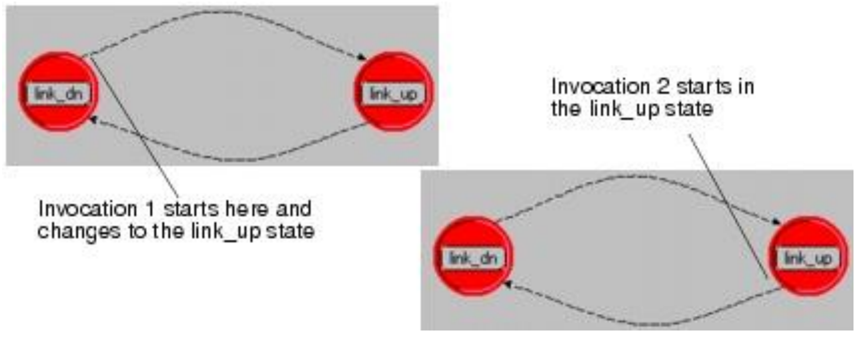


Simulations are made up of events. Process models respond to events and can schedule new ones.

When an event occurs that affects a module, the Simulation Kernel passes control to the module's process model via an interrupt. The process model responds to the event, changing state and executing related code, and then returns control to the Simulation Kernel (invocation 1).

The next time OPNET invokes the process model, it determines the state in which it was left (invocation 2), responds to the new event, and passes control back to the Simulation Kernel.

Which State is Active?


link_dn
link_up
Invocation 2 starts in the link_up state
Invocation 1 starts here and changes to the link_up state
link_dn
link_up

When a process model enters a state, it executes the enter executives and then, if the state is unforced (as above), the process model stops execution and returns control to the simulation. Forced and unforced states are discussed in more detail later.

Designing the Model

The packet-counting process model you will build contains three states: an initializing state, an idle state, and an arrival state.

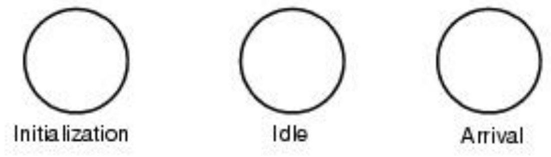
When a packet arrives at the module containing your process model, the model must increment a counter and then dispose of the packet.

Therefore, the module has two primary states:

1. **Wait** for a packet to arrive
2. **Process** the packet after it arrives

You also need an initialization state that sets the appropriate variables to zero. The process model will have the states shown in the following figure:

The Three States Required for the Model

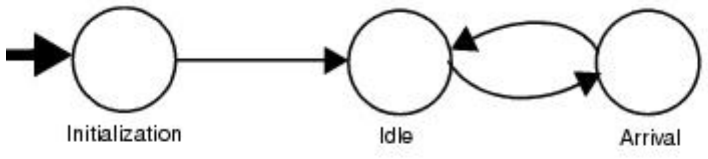


The packet-counting model contains three transitions: initialization-to-idle, idle-to-arrival, and arrival-to-idle.

The first time the process model is invoked (in this case, at the beginning of the simulation), it begins in the initialization state. After initialization—which sets the packet-counter variable to zero—the process model transitions to the idle state and waits for the first packet to arrive.

The process model is activated again and transitions to the arrival state when a packet arrives. The arrival state increments the packet-counter variable and destroys the packet. Without pausing, the process model then transitions back to the idle state to wait for the next packet.

Three Transitions are Required

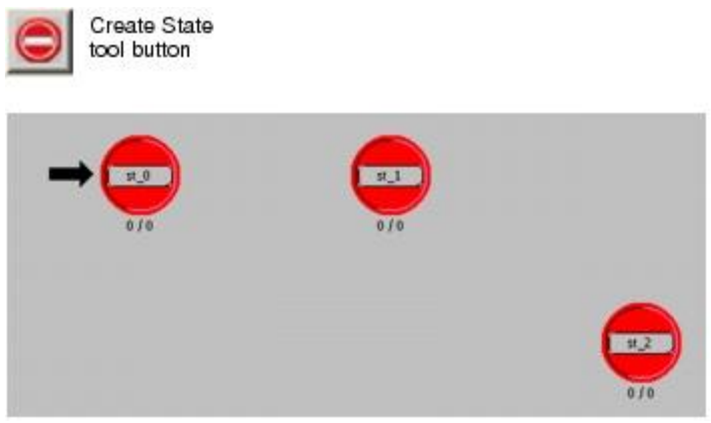


Implementing the Process Model

The first step in implementing the process model is to open the Process Editor and place the model's three states in the workspace.

1. Start OPNET Modeler if it is not already running.
2. Choose **File > New...** and select **Process Model** from the pull-down menu. Click **OK**.
3. Click the **Create State** tool button and place three states in the workspace as shown.

Placing Three States

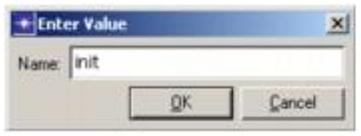


Notice the first state you create is automatically the initial state; this is indicated by a heavy arrow.

Give each state a unique name that describes its function. You will use the name **init** for the initialization state, **idle** for the idle state, and **arrival** for the arrival state.

1. Right-click on the initial state (the one with the heavy arrow) and select **Set Name** from the Object pop-up menu.
2. Name the state **init** and press **Return**.

Naming the State



Name: init

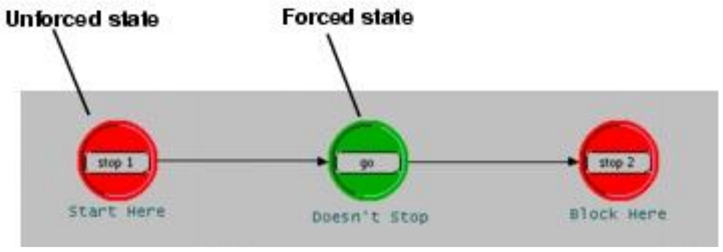
3. Repeat the procedure with the other two states and name them as follows:
   - **st_1: idle**
   - **st_2: arrival**

An unforced (red) state is one that returns control of the simulation to the Simulation Kernel after executing its enter executives. A forced (green) state is one that does not return control, but instead immediately executes the exit executives and transitions to another state.

A newly-created state is, by default, unforced: after executing its enter executives, the process model blocks (that is, stops execution and returns control to the Simulation Kernel). The next time the process model is invoked, execution begins again from the state in which it last blocked.

A forced state does not stop after the enter executives. Execution proceeds to the exit executives and transitions to the next state.

Forced and Unforced States



Two states in the process model you are building are forced states. The **init** state is forced because it can proceed directly to the **idle** state after initializing variables. The **arrival** state is forced because the process model should return to the **idle** state after counting and destroying each packet.

1. Change the **init** state to a forced state by right-clicking on it and selecting **Make State Forced** from the Object pop-up menu.

   The **init** state becomes green.

2. Repeat step 1 for the **arrival** state so that it is also a forced state.

Create State Transitions

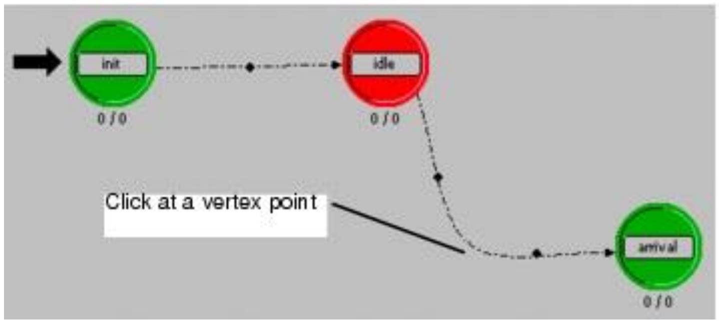
The next step in building the process models is to create the transitions between the states. There are two types of transitions, unconditional and conditional. When a transition is conditional, the transition must evaluate to true before control passes from the source state to the destination state.

1. Click the **Create Transition** tool button.
2. Draw the transition by first clicking on the **init** state and then clicking on the **idle** state.

The arrow of the transition points from the source state to the destination state. The transition is automatically selected and appears as a dotted and dashed line. (When unselected, the transition will be a solid line, indicating that it is unconditional.)

3. Draw a curved transition by clicking first on the **idle** state, then at a vertex point between the **idle** and **arrival** states, and finally on the **arrival** state.

Drawing Transitions



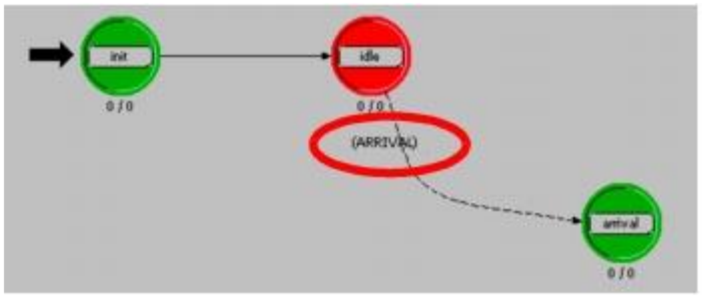
4. Right-click to stop drawing transitions.
5. Left-click in the workspace to unselect the transitions.

To create a conditional transition, you place a value in the **condition** attribute of the transition. You must make the transition between **idle** and **arrival** conditional because the process model should transition to the **arrival** state only when a packet arrives, and not in response to any other event.

1. Right-click on the transition between the **idle** and **arrival** states and select **Edit Attributes** from the pop-up menu.
2. Change the value of the **condition** attribute to **ARRIVAL** (be sure to use all capital letters), press **Return**, then click **OK**.

   The Attributes dialog box closes.

"condition" Attribute has Value of "ARRIVAL"

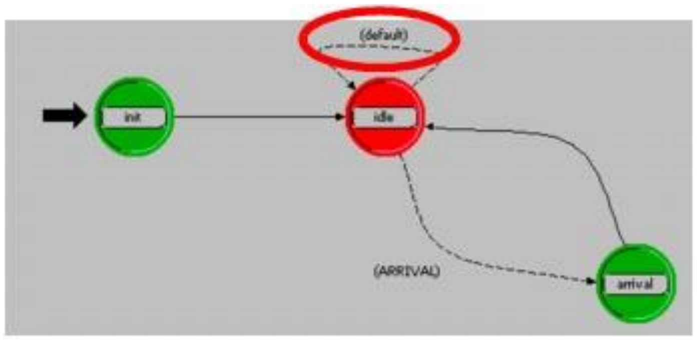


The transition is now a dashed line and has a label with the name of the condition. You will define the **ARRIVAL** condition macro later to specify that the transition will be used only when a packet arrives and the process model is in the idle state.

There are two more transitions to implement in the process model. The first is an unconditional transition from the **arrival** state to the **idle** state, and the second is a conditional transition from the **idle** state to itself.

1. Click on the **Create Transition** tool button and draw a transition from **arrival** to **idle**.
2. Draw a transition from the **idle** state back to itself, as shown in the next figure. Hint: click on two points in the workspace while drawing the transition before bringing it back to the **idle** state.
3. Right-click to stop drawing transitions.
4. Right-click on the transition from **idle** back to itself and select **Edit Attributes** from the pop-up menu to bring up the Attributes dialog box.
5. Change the value of the **condition** attribute to **default** (be sure to use all lower-case letters), press **Return**, and click **OK**.

"condition" Attribute has Value of "default"



You may be wondering why you included a transition from **idle** back to itself and named that transition **default**. As the simulation executes, the Simulation Kernel manages a list of events to take place. As each event reaches the top (or head) of that list, it becomes an interrupt. Interrupts are often delivered to specific modules, and this occurrence is what activates the module's process model.

When the process model you are building is in the **idle** state, it transitions to the **arrival** state if the **ARRIVAL** condition is true. Shortly, you will define **ARRIVAL** to evaluate to true if the interrupt delivered to the module is a packet arrival interrupt. If it is a different type of interrupt, however, there must be a transition that the process model can follow. The **default** transition handles these different interrupt types.

Define Conditions and Variables

In Proto-C, macros often replace more complicated expressions in transition conditions and executives. The use of macros saves space and simplifies the task of interpreting an FSM diagram. You specify macros with the C preprocessor **#define** statement and usually place them in the process model header block. The header block may also contain **#include** statements, **struct** and **typedef** definitions, **extern** variable declarations and global variable definitions, function declarations, and comments in C and C++ style.

The next step is to define the **ARRIVAL** macro.

1. Click the **Edit Header Block** tool button.

   An edit pad appears.

2. Enter the following code into the edit pad:

   ```
   #define ARRIVAL (op_intrpt_type () == OPC_INTRPT_STRM)
   ```

3. Choose **File > Save** from the edit pad menu to save the changes and close the edit pad.

The **ARRIVAL** condition defined above tests if the current interrupt (which caused the FSM to awaken and execute) occurred due to an arriving packet. It compares the value returned by the Kernel Procedure **op_intrpt_type()** with the constant of **OPC_INTRPT_STRM**. If the comparison evaluates to true, this indicates that the interrupt is due to a packet arriving on an input stream.

You can declare variables in three places. Variables declared in the **temporary variables block** do not retain their values between invocations of the FSM. Variables declared in the **state variables block** retain their values from invocation to invocation. Variables declared inside a state's executive are only defined while that executive is executed.

The next step is to declare two state variables. One will store the value of the packet count; the other is the "handle" for the local statistic used to analyze the count.

1. Click the **Edit State Variables** tool button.

2. Enter the following values into the dialog box:

   Values for State Variables Dialog Box

Type	Name	Comments

int	pk_count	Counts total packets
Stathandle	pk_cnt_stathandle	Statistic to record packet count

Clicking the value field for **Comments** opens an edit pad. After you type the comment, Choose **File > Save**.

3. Click the **OK** button to save the changes and close the State Variables dialog box.

Statistics save values of interest for later analysis. When creating a statistic, you must declare that statistic in the process model that records it. You will create one statistic for this process.

1. Choose **Interfaces > Local Statistics**.
2. Enter **packet count** as the first **Stat Name**.

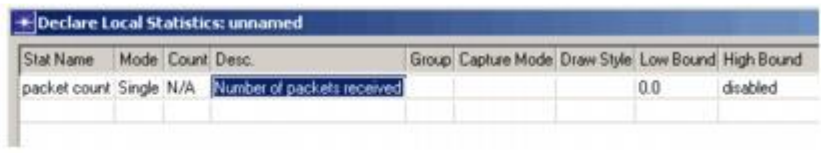
   The mode is automatically set to Single, count to N/A, Low Bound to 0.0, and High Bound to disabled.

3. Click in the **Description** field for **packet count**.

   An edit pad opens.

4. Enter a short description for the statistic: **Number of packets received.**
5. Choose **File > Save** to save the changes and close the pad.

Declare Local Statistics Dialog Box



Stat Name	Mode	Count	Desc.	Group	Capture Mode	Draw Style	Low Bound	High Bound
packet count	Single	N/A	Number of packets received				0.0	disabled

6. Close the Declare Local Statistics dialog box by clicking the **OK** button.

Create State Executives

The next step is to create the state executives needed in the FSM. In the first state, **init**, the executives set the packet count to 0 and associate the name of the local statistic with its statistic handle. Although the **enter executives** and **exit executives** of forced states are executed without interruption, standard practice is to place all forced state executives in the **enter executives** block.

1. Double-click on the top half of the **init** state.

Open the Enter Executives of the init State

The Enter Execs edit pad opens.

2. Enter the following code to initialize the state variables **pk_count** and **pk_cnt_stathandle**:

```
pk_count = 0;
pk_cnt_stathandle = op_stat_reg ("packet count",
                    OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
```

3. Choose **File > Save** to close the edit pad.

The second line of the preceding code registers the **packet count** statistic and sets up a handle for the **arrival** state to use when recording the number of packets received.

The **pk_count** and **pk_cnt_stathandle** variables are state variables you defined, but the **op_stat_reg()** function is a Simulation Kernel Procedure (KP). Hundreds of built-in functions are provided that you can use for a variety of purposes, such as manipulating a queue or creating animations.

The numbers displayed below each state indicate the line count for the enter and exit executive of the state. Now that you have entered some code in the **Enter Execs** of the **init** state, the information has changed from 0/0 to 2/0. To toggle the display of these line counts, right-click in the Process Editor workspace and choose **Hide Line Counts**. You can show them again with the **Show Line Counts** menu.

Because there are so many functions, it can be difficult to determine the most appropriate one for a particular task. However, a summary of the most frequently used KPs is provided. You will review this summary in the next step of this tutorial, when you write the code for the **Enter Execs** in the **arrival** state.

To define what happens in the arrival state, follow these steps:

1. Double-click on the top half of the **arrival** state.

   The **Enter Execs** edit pad opens.

   Now that you are In the **arrival** state, you need to increment the packet count, determine the packet stream for the current interrupt, retrieve the pointer to the packet so that you can destroy it, and write out the statistic.

2. Increment the **pk_count** variable by entering the following code into the **Enter Execs** edit pad:

    ```
    ++pk_count;
    ```

Determining the packet stream for the current interrupt, getting the packet's pointer, and destroying the packet is a bit more complex, but can be done using some of the KPs.

To determine the packet stream, get the pointer, and destroy the packet, you can use KPs from the list of most commonly used KPs, the Essential Kernel Procedures.

1. Choose **Help > Essential Kernel Procedures**.

    A list of the most frequently-used KPs opens in a browser window.

2. The first task—determining the packet stream for the current interrupt—has to do with processing an interrupt. Find the section called **Interrupt Processing**.
3. From the KP descriptions, you determine that **op_intrpt_strm()** will return the necessary information.
4. The next two tasks, getting the packet's pointer and using that pointer to destroy the packet, both involve packet manipulation. Look in the **Packet Generation and Processing** section. You can use **op_pk_get()** to determine the packet's pointer, then use that return value (the pointer) to call **op_pk_destroy()** to destroy the packet.

You can type these function calls into the process model, or copy them from the Essential Kernel Procedures file. In the following steps, you will copy these function calls.

1. Copy the **op_pk_destroy()** function call:
    a. Go to the **op_pk_destroy()** function description in Essential Kernel Procedures and select the function syntax text.
    b. Copy and paste the function call into the enter execs section of the **arrival** state, just after the line where you incremented the packet count.
    c. Enter a semicolon at the end of the function call, so the line looks like this:

        ```
        op_pk_destroy (pkptr);
        ```

2. Replace the **pkptr** argument with the following code:

    ```
    op_pk_get (op_intrpt_strm ())
    ```

    The resulting line of code should look like this:

    ```
    op_pk_destroy (op_pk_get (op_intrpt_strm ()));
    ```

The final task is to write out the value of the statistic. Refer to the **Statistic Recording** section for information on the **op_stat_write()** KP.

The final code for the Enter executive of the arrival state should look like this:

```
++pk_count;
op_pk_destroy (op_pk_get (op_intrpt_strm ()));
op_stat_write (pk_cnt_stathandle, pk_count);
```

Finally, save the enter exec code:

Choose **File > Save** to close the edit pad.

The Essential Kernel Procedures are those that are most commonly used. Many additional KPs are available. You can display a description of any KP by choosing **Help > All Kernel Procedures** from the Process Editor menu.

In both KP help files, each KP name is a hypertext link to the KP documentation.

Edit Process Interfaces

To control the attributes visible at the node level, edit the Process Interfaces. You can set initial attribute values or hide attributes.

1. Choose **Interfaces > Process Interfaces**.

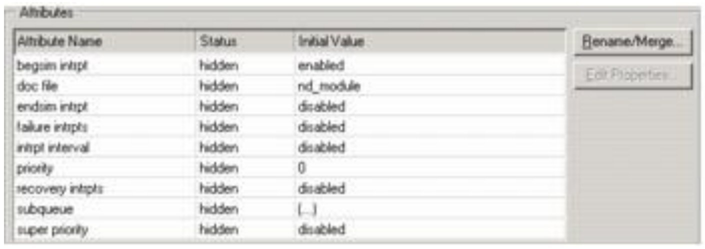
   The Process Interfaces dialog box appears.

2. Change the **Initial Value** for the **begsim intrpt** attribute to **enabled**.

   This will cause the Simulation Kernel to deliver such an interrupt to the process model instance at the start of the simulation.

3. Verify that the **Initial Value** for each of the following attributes is set to **disabled**: **endsim intrpt**, **failure intrpts**, **intrpt interval**, **recovery intrpts**, and **super priority**.
4. Verify that the **Initial Value** of the **priority** attribute is **0**.
5. For all attributes, change the value of **Status** to hidden by left-clicking in the **Status** column and selecting **hidden**, so that the attributes will not appear in the node editor.

   Changing the Value to "hidden"



Attributes

Attribute Name	Status	Initial Value
begsim intrpt	hidden	enabled
doc file	hidden	nd_module
endsim intrpt	hidden	disabled
failure intrpts	hidden	disabled
intrpt interval	hidden	disabled
priority	hidden	0
recovery intrpts	hidden	disabled
subqueue	hidden	(...)
super priority	hidden	disabled

6. Click **OK** to close the Process Interfaces dialog box.

Compile the Model

All process models must be compiled before they can be used in a simulation. Compiling a model makes its object code available to a processor module's **process model** attribute.

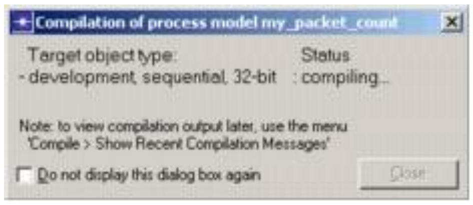
To compile a process model:

1. Click the **Compile Process Model** tool button.

2. If you did not save your model previously, the Save As dialog box opens. Save the process model as **<initials>_packet_count** in your op_models directory.

   A dialog box showing object type and compilation status appears.

   Compilation Status Dialog Box



Compilation of process model my_packet_count

Target object type: - development, sequential, 32-bit

Status: compiling...

Note: to view compilation output later, use the menu 'Compile > Show Recent Compilation Messages'

Do not display this dialog box again

   Note: If an error message appears, close the error report and verify the code you have entered for the process model so far.

3. Click **Close** to close the dialog box.

   If you did not save the process model before compiling it, you will also see this message:

```
Wrote File: (<file name>).
```

4. Close the Process Editor.

Implementing the Node and Network Models

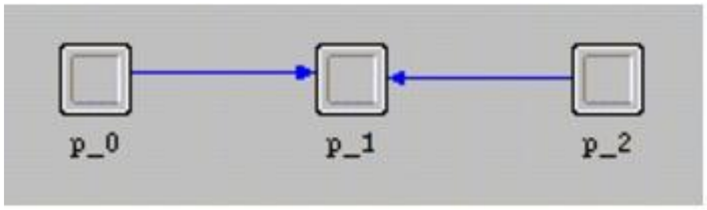
The node model in this lesson has two packet generator modules that send packets to a general processor module. The processor module uses the previously defined **packet_count** process to count and discard packets and to update an output statistic. Note that the process model could accept any number of incoming packet streams because it automatically determines the correct stream when an interrupt occurs.

Create the Node Model

First, create a node model that uses the **packet_count** process.

1. Choose **File > New...,** then select **Node Model** from the pull-down menu. Click **OK**.
2. Use the **Create Processor** tool button to create three processor modules.
3. Use the **Create Packet Stream** tool button to connect the modules with packet streams as shown below. The model should resemble the following figure:

Initial Node Model



Now change the attributes of each module. The first processor module, **p_0**, will be a generator module:

1. Right-click on the icon and select **Edit Attributes** from the pop-up menu to open the module's Attributes dialog box.
2. Change the **name** attribute to **src1**.
3. Change the **process model** attribute to **simple_source**.
4. Click on **OK** to close the dialog box.

The second processor module, **p_1**, will count packets:

1. Open the Attributes dialog box of the module.
2. Change the **name** attribute to **count**.
3. Change the **process model** attribute to **<initials>_packet_count**.
4. Click on **OK** to close the dialog box.

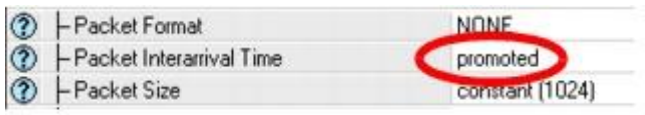
Set up the third processor module to generate packets:

1. Open the Attributes dialog box of the module.
2. Change the **name** attribute to **src2**.

3. Change the **process model** attribute to **simple_source**.
4. Click on **Packet Interarrival Time** in the left column to highlight the Attribute name, then right-click and select **Promote Attribute to Higher Level** from the pop-up menu.

   The word **promoted** is displayed in the Value cell of the attribute.

   Promoting the Attribute



Packet Format	NONE
Packet Interarrival Time	promoted
Packet Size	constant (1024)

5. Click on **OK** to close the dialog box.

Both generators send packets to **count**. The first generator sends packets at a constant rate of one packet/second and the second generator sends packets at a variable rate, following a distribution you will specify.

Sometimes it is convenient or necessary to make a statistic available at the node level. A statistic promoted to the node level can be renamed and given a different description. Before saving the node model, promote the packet count statistic and rename it.

1. Choose **Interfaces > Node Statistics**.
2. Click the first field in the **Orig. Name** column, select **count.packet count** from the list that appears, and click **Promote**.
3. Change the **Prom. Name** to **node packet count**.
4. Change the **Desc.** field to **Number of packets received at the node level**.
5. Choose **File > Save** from the edit pad's menu to save the text and close the edit pad.
6. Close the **Statistic Promotion** dialog box by clicking the **OK** button.
7. Choose **File > Save** and save the node model as **<initials>_packet_count**. (The node and process models can have the same name because OPNET appends **.nd.m** to each node model and **.pr.m** to each process model.)
8. Close the Node Editor.

Create the Network Model

The network model for this lesson consists of a single node using the **packet_count** node model. After placing the node in the Project Editor workspace, you can specify statistics and animations to collect.

First, create a new project.

1. Choose **File > New...** and open a new project.
2. Name the project **<initials>_packet_count** and the scenario **constant**.
3. Choose **Quit** on the first screen of the Startup Wizard.

Second, create a custom object palette.

1. Open the object palette by clicking on

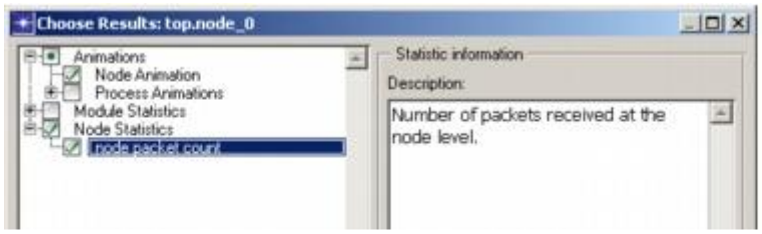
   the **Open Object Palette** tool button.
2. Switch to the icon view by clicking on the button in the upper-left corner of the object palette.
3. Click the **Configure Palette...** button.
4. Click the **Clear** button in the Configure Palette dialog box to remove all but the default icons from the palette. The subnet icon remains. (If you have the Wireless module, the mobile and satellite subnet icons remain as well.)
5. Click the **Node Models** button and set the **Status** of the **<initials>_packet_count** node model to **included**.
6. Click **OK** to close the Select Included Entries dialog box, and then click **OK** to close the Configure Palette dialog box.
7. At the prompt, save the custom model list as **<initials>_packet_count**.

Next, place the node in the workspace and select the proper statistics for collection.

1. Place an **<initials>_packet_count** fixed node in the workspace, then close the object palette.
2. Right-click on the node to open its Object pop-up menu, and then select **Choose Individual DES Statistics** from the sub-menu.

   The Choose Results statistic browser appears.

   Selecting Statistics

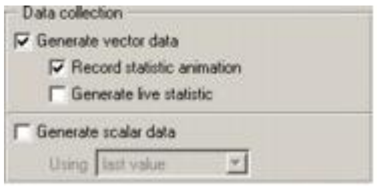


Number of packets received at the node level.

3. Select the **Animations > Node Animation** checkbox.
4. Select the **Node Statistics > node packet count checkbox**.
5. Right-click on **Node Statistics > node packet count** to open its Statistic pop-up menu, and choose **Record Statistic Animation**.

   The **Record statistic animation** checkbox in the Data collection area of the dialog box becomes enabled. You can also use that checkbox to toggle the animation on or off.

   Record Statistic Animation Checkbox


Data collection
☑ Generate vector data
☑ Record statistic animation
☐ Generate live statistic
☐ Generate scalar data
Using last value

6. Click **OK** to close the browser window.

In the preceding steps, you:

- Created an animation of packet flow within the node (**Animations > Node Animation**).
- Specified that the node packet count statistic be collected (**Node Statistics > node packet count**).
- Specified that the node packet count statistic be animated.

When you built the node model, you changed the **Packet Interarrival Time** attribute for the second generator module to **promoted** so that you could specify it later.

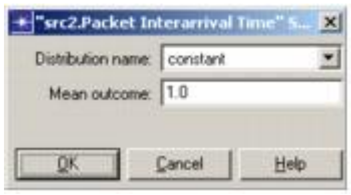
Now, you will specify the distribution for that attribute.

1. Right-click on the node to open its Object pop-up menu, then choose **Edit Attributes**.
2. Click on **promoted** in the value column of **src2.Packet Interarrival Time**.

   The Distribution Specification dialog box appears.

3. Verify that **Distribution name** is **constant** and **Mean outcome** is **1.0**.

   Distribution Specification Dialog Box


"src2.Packet Interarrival Time" S...
Distribution name: constant
Mean outcome: 1.0

4. Click **OK** to close the Specification dialog box.
5. Click **OK** to close the Attributes dialog box.

To configure the simulations:

1. Choose **DES > Configure/Run Discrete Event Simulation...**
2. Set the following values in the Configure DES dialog box:
   - **Duration**: **100 seconds**
   - **Seed**: **1471**

- **Values per statistic**: **100**

3. Expand the **Outputs** tree element, then the **Animation** one. Click on **2D** and verify that the **Send animation to history file** checkbox is selected.
4. Click on **Apply** to save the change, then **Cancel** to close the Configure/Run DES dialog box without running the simulation.

To investigate the effect of different PDFs, you can create new scenarios based on this one. After simulating them, you can then compare the results. To create a new scenario and set the PDF:

1. Choose **Scenarios > Duplicate Scenario...**
2. Name the new scenario **exponential**. Click **OK** to save the file.
3. Right-click on the node and choose **Edit Attributes** from the Object pop-up menu.
4. Click on the Value column of **src2.Packet Interarrival Time**; this opens its Specification dialog box.
5. Select **exponential** from the **Distribution name** pull-down menu and verify that **Mean outcome** is set to **1.0**.
6. Click **OK** to close the Specification dialog box.
7. Click **OK** to close the Attributes dialog box.
8. Save the project as **<initials>_packet_count**.

Running the Simulation

You are now ready to configure and run the simulations. You can run both simulations with one command from the Manage Scenarios dialog box.

To configure the repositories:

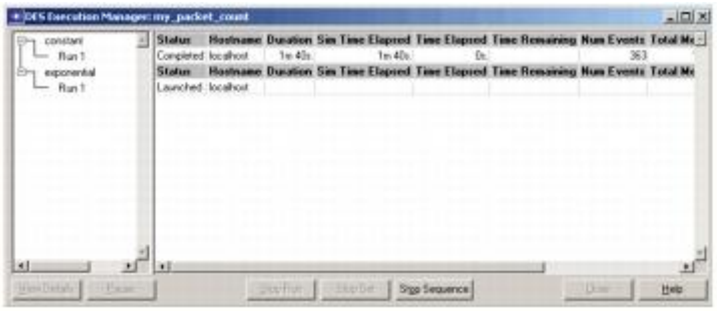
1. Choose **Edit > Preferences**.
2. Verify the **Network Simulation Repositories** preference is set to **()**, then click **Cancel**. If it is not empty, delete all entries and click **OK** to close the dialog box.

   To run the simulations:

3. Choose **Scenarios > Manage Scenarios...**
4. Change the value of the **Results** column for each scenario from **uncollected** to **<collect>**.
5. Click **OK** to begin the simulation runs.

   The DES Execution Manager dialog box opens, showing two planned simulations (one run for each scenario). As each simulation executes, its information is updated in that dialog box.

   DES Execution Manager Dialog Box



Status	Hostname	Duration	Sim Time Elapsed	Time Elapsed	Time Remaining	Num Events	Total Me
Completed	localhost	1m 40s	1m 40s	0s		363	

Status	Hostname	Duration	Sim Time Elapsed	Time Elapsed	Time Remaining	Num Events	Total Me
Launched	localhost						

The simulations should take less than one minute. If you have problems, see *"Troubleshooting Tutorials"*.

6. Close the DES Execution Manager dialog box.

Analyzing the Results

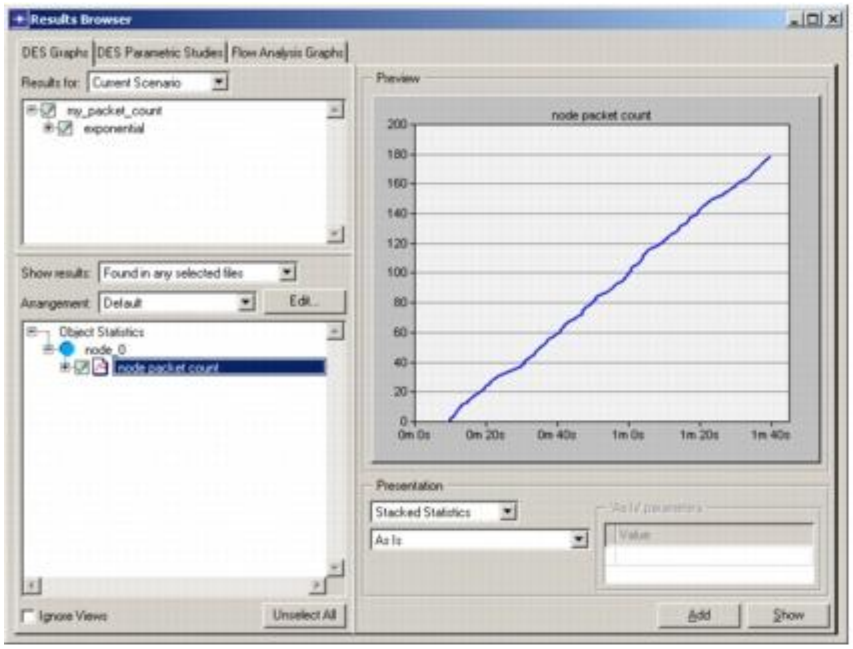
View results in the Project Editor.

1. If necessary, switch to the **exponential** scenario by choosing **Scenarios > Switch To Scenario > exponential**.
2. Right-click on **node_0** and choose **View Results** from the node_0 pop-up menu.

   The **Results Browser** opens, showing the statistics you can view.

3. Place a check in the check box for **Object Statistics > node_0 > node packet count**.

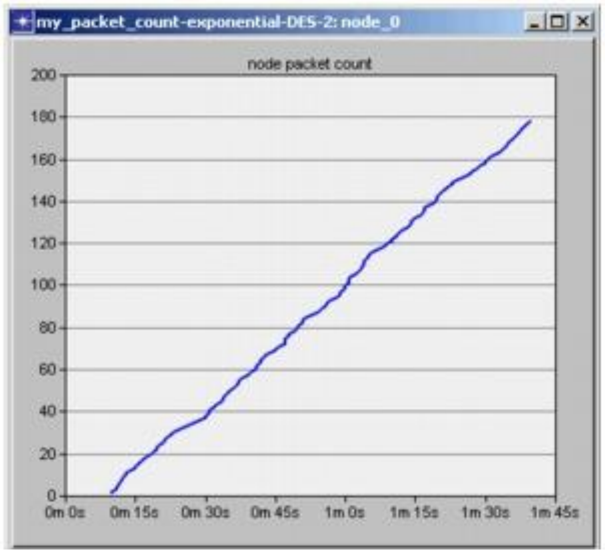
   Results Browser



4. Click the **Show** button to display the graph.

   The resulting graph shows the number of packets received by the count processor module over time during the simulation.

   Number of Packets Received

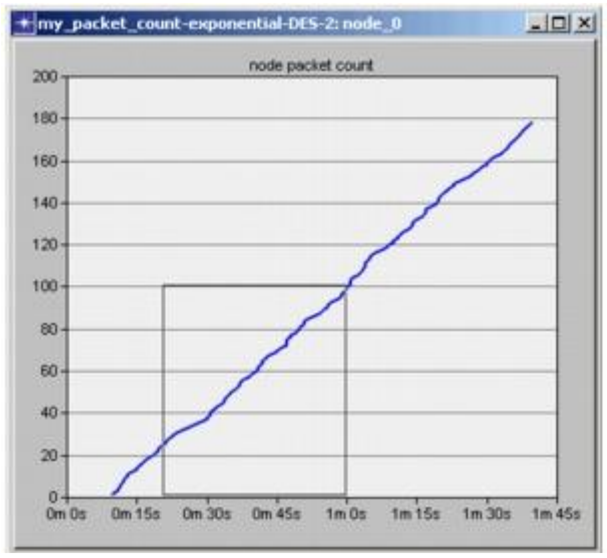


As illustrated in this graph, the results of the simulation indicate the total number of packets received steadily increases over the simulation duration, though not at a perfectly constant rate. As expected, with two sources delivering packets each at a mean of 1 packet/second, the total number of packets received after 90 seconds of packet generation (packet generation commences after 10s of simulation time) is about 180.

A close-up visual examination of the graph reveals more detail. To see the detail, magnify the graph.

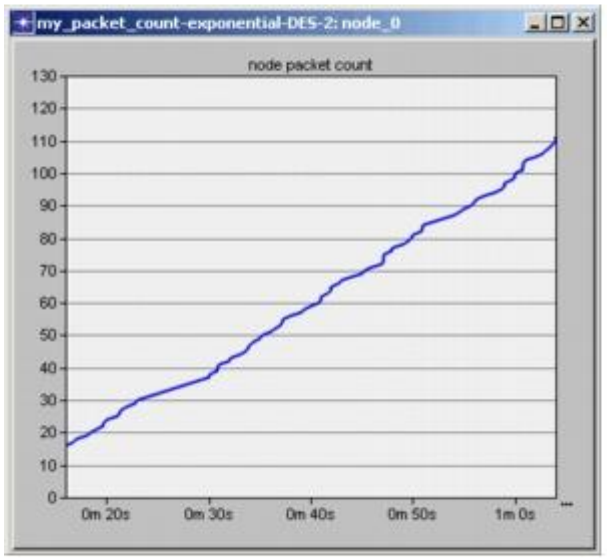
5. Drag the cursor along the graph, selecting a box that covers the trace from **20 seconds** to **1 minute** on the time axis.

Select the Area to Magnify



The graphs changes to a close-up of the selected graph area.

Magnified Area

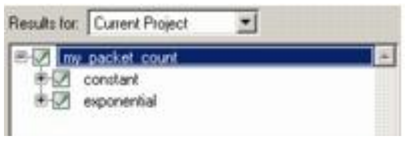


You can contrast the graph of the packet count in the **exponential** scenario with the one in the **constant** scenario.

6. Close the current graph and click **Delete**.
7. In the **Results Browser**, click on the drop-down list next to **Results for:** and select **Current Project**.

The Results tree shows available data files for all scenarios of the current project.

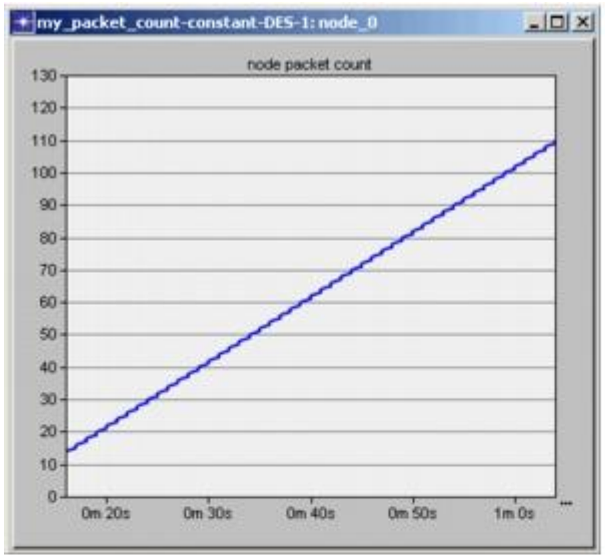
Viewing all Scenarios' Output for a Project



8. Switch to the **constant** scenario by unselecting **<initials>_packet_count > exponential** and selecting **<initials>_packet_count > constant**.
9. Show the graph for **Object Statistics > node_0 > node packet count**.
10. Zoom in on a portion of this graph as you did for the last graph.

Magnified (as shown in the following figure), this graph reveals a step pattern that results from the two generators sending packets at the same constant arrival rate.

Zooming Reveals Step Pattern

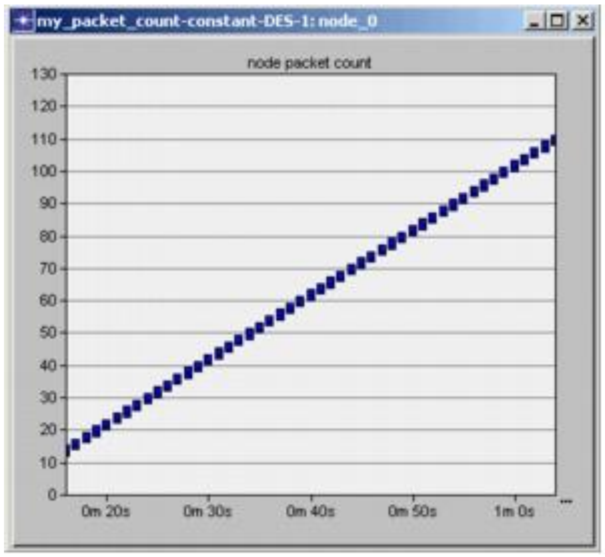


A discrete graph of this data will illustrate the true quantized property of the counting statistic and the underlying event-driven property of the packet arrivals.

11. Right-click on the graph to open the Graph pop-up menu.
12. Choose **Draw Style > Discrete** from the Graph pop-up menu.

    The graph now displays the data as individual points. The discrete plotting style depicts only the discrete statistic values without connecting them.

    Discrete Draw Style Applied



    The discrete graph reveals that the Simulation Kernel repeatedly delivered two packets during each second of simulation time, each of which was counted separately. The graph also shows the event-scheduled nature of the Simulation Kernel. The simultaneous events (multiple packets arriving every second) caused the Simulation Kernel to advance simulated time once, but invoke the **packet_count** process twice. In turn, the **packet_count** process incremented the packet count statistic once each time it was invoked.

13. Close the graph dialog box, then click **Delete**.
14. Close the Results Browser.

Finally, you can view the animations created during the simulation. The animation viewer allows you to view statistic and packet animations.

1. Choose **DES > Play 2D Animation**.

   The Animation Viewer opens, showing the statistics animation for the exponential scenario.

2. To see the packet animation, choose **Windows > Animation Viewer > top.node_0 packet flow**.
3. To exit the Animation Viewer, choose **File > Exit**.

Note the Project Editor is still open.

4. To view the animations for the *constant* scenario, change to the *constant* scenario (**Scenarios > Switch to Scenario > constant**), and then choose **DES > Play 2D Animation**.

   The animation for the constant scenario appears.

5. Exit the Animation Viewer.

For more information about the Animation Viewer, see *Animation Viewers Reference Manual*.

You have now completed the Basic Processes lesson of the tutorial. You should have a good understanding of how to control node behavior by creating custom process models.

The next lesson, Packet Switching, explores the creation of a packet switching network.

To continue with the next lesson, save your work and close all open editor windows. Return to the list of tutorials in the Contents pane and choose **Packet Switching I**.

Práctica 16

Análisis de Pequeñas Redes Interconectadas (Small Internetworks)

Introduction

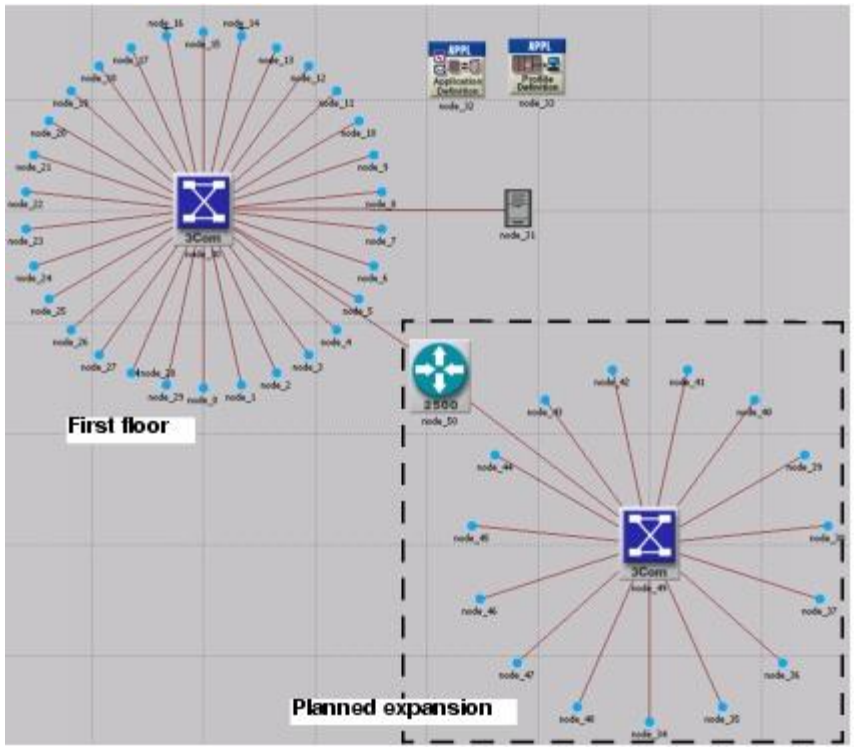
In this lesson, you will see how OPNET Modeler can do organizational scaling to solve a typical "what if" problem. You will learn how to use OPNET Modeler features to build and analyze network models. This lesson focuses on the use of the Project Editor, and how it will be used with the Node and Process editors in later lessons. In this lesson, you will

- Build a network quickly
- Collect statistics about network performance
- Analyze these statistics

In this lesson, you use the Project Editor to build a topology of a small internetwork, choose statistics to collect, run a simulation, and analyze the results.

In this lesson, you plan for the expansion of a small company's intranet. Currently, the company has a star topology network on the first floor of its office building and plans to add an additional star topology network on another floor. You will build and test this "what-if" scenario to ensure that the load added by the second network will not cause the network to fail.

The Final Network



Getting Started

When creating a new network model, you must first create a new **project** and **scenario**. A project is a group of related scenarios that each explore a different aspect of the network. Projects can contain multiple scenarios.

After you create a new project, you use the **Startup Wizard** to set up a new scenario. The options in the Wizard let you

- Define the initial topology of the network
- Define the scale and size of the network
- Select a background map for the network
- Associate an **object palette** with the scenario

The **Startup Wizard** automatically appears each time you create a new project. The Startup Wizard lets you define certain aspects of the network environment.

To use the Startup Wizard to set up a new scenario, do the following:

1. If OPNET Modeler is not already running, start it.
2. Select **File** > **New...**
3. Select **Project** from the pull-down menu and click **OK**.
4. Name the project and scenario, as follows:
    a. Name the project **<initials>_Sm_Int**.

Include your initials in the project name to distinguish it from other versions of this project.

b. Name the scenario **first_floor**.
c. Click **OK**.

The Startup Wizard opens.

5. Enter the values shown in the following table in the dialog boxes of the Startup Wizard.

Values to Enter in the Startup Wizard

Dialog Box Name	Value
1. Initial Topology	Select the default value: **Create empty scenario**.
2. Choose Network Scale	Select **Office**. Select the **Use metric units** checkbox.
3. Specify Size	Select the default size: **100 m x 100 m**
4. Select Technologies	Include the **Sm_Int_Model_List** model family.
5. Review	Check values, then click **Finish**.

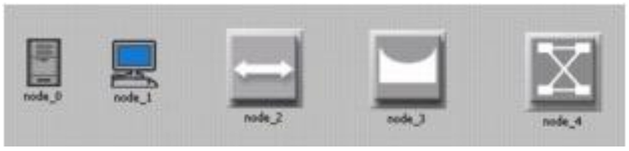
A workspace of the size you specified is created. The object palette you specified opens in a separate window.

Creating the Network

Network models are created in the Project Editor using **nodes** and **links** from the **object palette**.

Node—A representation of a real-world network object that can transmit and receive information.

Nodes



Link—A communication medium that connects nodes to one another. Links represent physical connectivity (e.g., electrical or fiber optic cables).

A Link

These objects are found in the **object palette**, a dialog box that contains graphical representations of node and link models.

If it is still open, close the object palette.

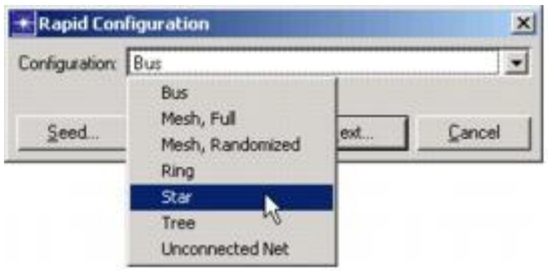
You can use any of three methods to create a network topology, or a combination of all three. One method is to import the topology (discussed in a later lesson). Another is to place individual nodes from the object palette into the workspace. The third method is to use **Rapid Configuration**.

Rapid Configuration creates a network in one action after you select a network configuration, the types of nodes within the network, and the types of links that connect the nodes.

To create the first-floor network using Rapid Configuration:

1. Select **Topology > Rapid Configuration**.
2. Select **Star** from the pull-down menu of available configurations, then click **Next...**

   Available Configurations Pull-Down Menu



Specify the node models and link models in the network. Models follow this naming scheme:

Generic Devices:
<protocol*1*>_..._<protocol*n*>_<function>**_**<mod>

Vendor Devices:
<Vendor>_<Chassis/Make>_<protocol*1*>

where:

- **<protocol>** specifies the specific protocol(s) supported by the model
- **<function>** is an abbreviation of the general function of the model

- **<mod>** indicates the level of derivation of the model

For example:

ethernet2_bridge_int

specifies the intermediate (**int**) derivation of a 2-port Ethernet (**ethernet2**) bridge (**bridge**).

Vendor models have an additional prefix that specifies the vendor and the vendor product number for that particular network object.

For example, the 3Com switch used in this lesson is named:

3C_SSII_1100_3300_4s_ae52_e48_ge3

This node is a stack of two 3Com SuperStack II 1100 and two Superstack II 3300 chassis (**3C_SSII_1100_3300**) with four slots (**4s**), 52 auto-sensing Ethernet ports (**ae52**), 48 Ethernet ports (**e48**), and 3 Gigabit Ethernet ports (**ge3**).

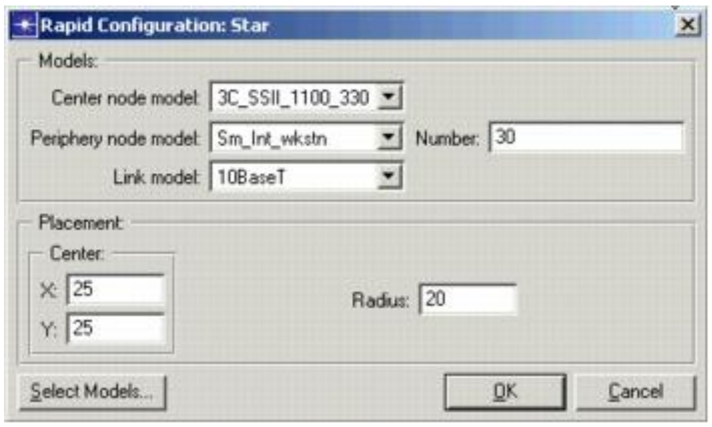
To specify the nodes and links to use to build the network:

1. Set the **Center Node Model** to **3C_SSII_1100_3300_4s_ae52_e48_ge3**. This is a 3Com switch.
2. Set the **Periphery Node Model** to **Sm_Int_wkstn**, and change the **Number** of periphery nodes to **30**. This sets 30 Ethernet workstations as the peripheral nodes.
3. Set the **Link Model** to **10BaseT**.

Specify where the new network will be placed:

1. Set the **X center** and **Y center** to **25**.
2. Set the **Radius** to **20**.

Rapid Configuration Dialog Box

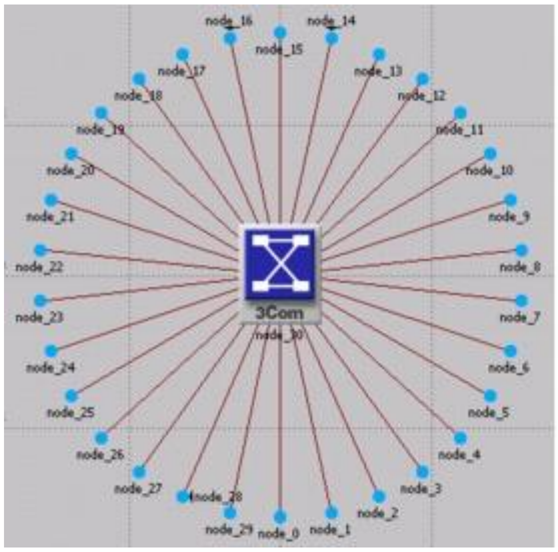


Center node model: 3C_SSII_1100_330
Periphery node model: Sm_Int_wkstn Number: 30
Link model: 10BaseT
X: 25
Y: 25
Radius: 20

3. Click **OK**.

   The network is drawn in the Project Editor.

   The First Floor Network

Now that the general network topology has been built, you need to add a server. You will use the second method of creating network objects: dragging them from the object palette into the workspace.

1. If it is not already open, open the object palette by clicking on the **Object Palette** tool button.

2. Find the **Sm_Int_server** object in the palette and drag it into the workspace.

   You will not find this exact server model on other object palettes because we created it with the correct configuration for this tutorial.

   By default, you can create additional instances of the same object by left-clicking after the initial "drag-and-drop" from the palette.

3. Because you do not need additional copies of this model, right-click to turn off node creation.

You also need to connect the server to the star network.

1. Find the **10BaseT** link object in the palette and double-click on it.

2. Move the mouse from the object palette to the project workspace. Now click on the server object to draw one endpoint of your link, then click on the switch object in the center of the star to complete the link.

   A link is drawn, connecting the two objects.

3. Right-click to turn off link creation.

Finally, you need to add configuration objects to specify the application traffic that will exist on the network. Configuring the application definition and profile definition objects can be complicated, so you do not have to do these tasks right now. For this tutorial, we included, on the object palette:

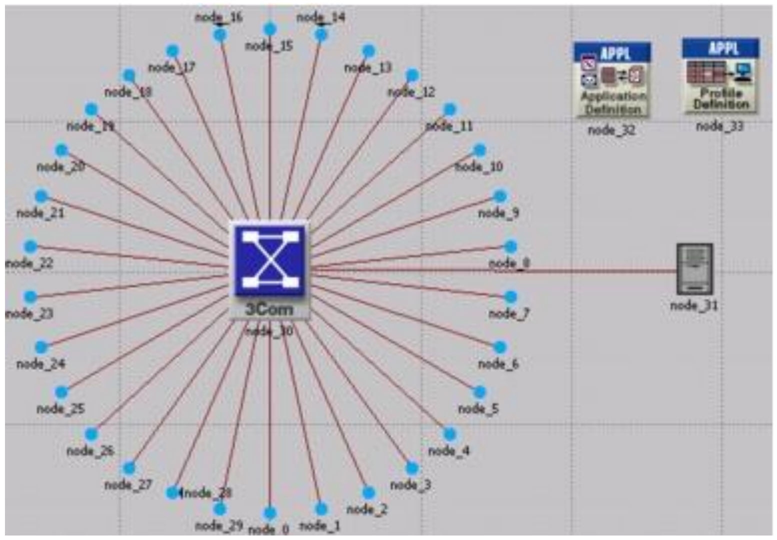
- an application definition object with the default configurations of the standard applications, and
- a profile definition object with a profile that models light database access

You need only drag the objects into your network. Doing so means that the traffic caused by workstations accessing a database at a low rate will be modeled.

1. Find the **Sm_Application_Config** object in the palette and drag it into the workspace
2. Right-click to turn off object creation.
3. Find the **Sm_Profile_Config** object in the palette, drag it into the workspace, and right-click.
4. Close the object palette.

The network is now built and should look similar to the following figure.

The Finished First Floor Network



You are now ready to collect statistics.

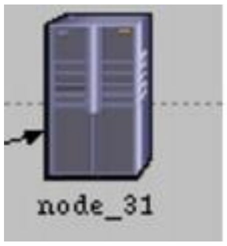
However, first let's explore the Node and Process Editors.

The Node and Process Editors are integral to the OPNET Modeler workflow. The Node Editor is used to create node models that describe the internal flow of data within a network object. The Process Editor is used to create process models that describe the behavioral logic of a module in a node model.

Every node in a network has an underlying node model that specifies the internal flow of information in the object. Node models are made up of one or more **modules** connected by **packet streams** or **statistic wires**. Node modules in turn contain process models. A process model is represented by a **state transition diagram** (**STD**) that describes the behavior of a node module in terms of **states** and **transitions**.

Let's explore the node model that controls the server in the first floor network:

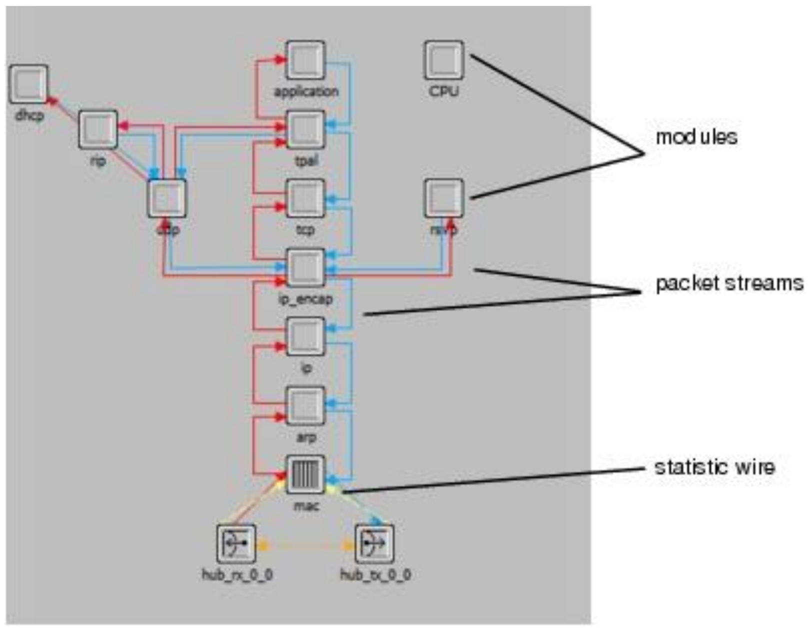
1. Double-click on **node_31** (the **Server object**) in the Project Editor.



The Node Editor opens as a new window within OPNET Modeler.

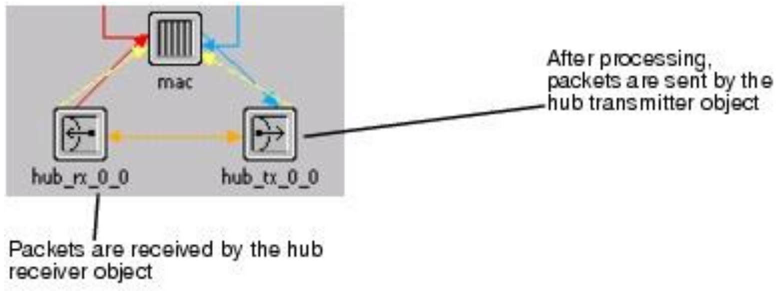
The following figure shows the node model within the Ethernet Server network object. The node model is made up of several different types of **modules,** which are described in the M/M/1 tutorial. **Packet streams** and **statistic wires** connect the modules.

Ethernet Server Node Model



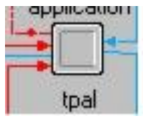
During a simulation, packets sent from a client machine are received by the hub receiver object (hub_rx_0_0) and processed up the protocol stack to the application module. After processing, they are sent down the stack to the transmitter (hub_tx_0_0), then back to the client machine.

Packet Processing by the Node Model


mac
hub_rx_0_0
hub_tx_0_0
After processing, packets are sent by the hub transmitter object
Packets are received by the hub receiver object

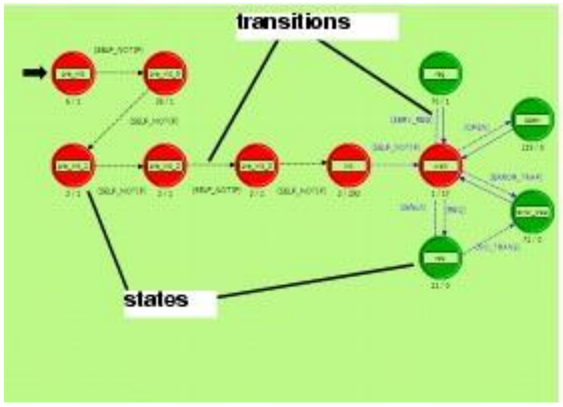
Next, let's look at the process model that defines the behavior of the **tpal** module. To view the process model:

1. Double-click on the **tpal** module in the Node Editor.



The Process Model Editor opens in a new window.

Example Process Model

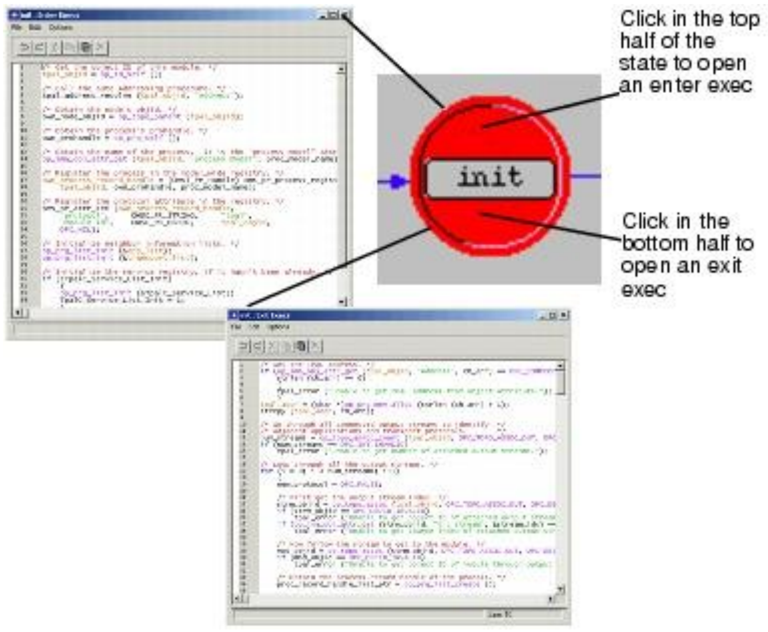


2. Note the red and green states (these will be discussed in greater detail in the Basic Processes lesson) and the solid and dotted lines indicating transitions between the states.

   Each state in the process model contains an **enter executive** and an **exit executive**. Enter executives are executed when a process enters a state. Exit executives are executed when the process leaves the state. Operations performed in the state are described in C or C++.

3. Open an enter exec by double-clicking on the top half of the **init** state.
4. Open an exit exec by double-clicking on the bottom half of the state.

   Opening the Enter Exec or Exit Exec of a State

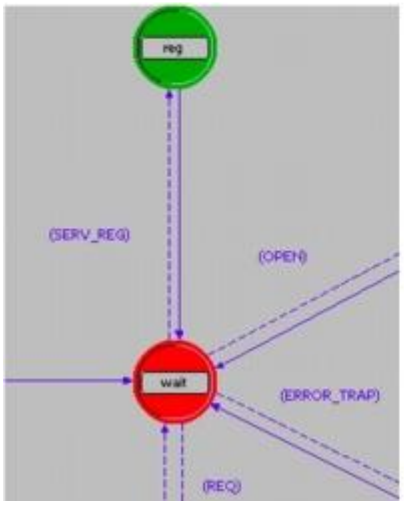

init
Click in the top half of the state to open an enter exec
Click in the bottom half to open an exit exec

5. Close both exec windows.

States are connected via **transitions**. Transitions can be either conditional (that is, they have a logical test that must be true before the transition occurs) or unconditional (no logical test).

The following figure shows a conditional transition (the dotted line) from the **wait** state to the **reg** state. The condition **SERV_REG** must be true before the transition can occur. However, the transition from **reg** to **wait** (solid line) is unconditional, and occurs whenever the code in the **reg** state has finished execution.

Conditional and Unconditional Transitions



Later lessons explore these editors in greater depth.

6. Close the Node and Process editors. If prompted, do not save changes.

Collecting Statistics

You can collect statistics from individual nodes in your network (**object statistics**) or from the entire network (**global statistics**).

Now that you have created the network, you should decide which statistics you need to collect to answer the questions presented earlier in this lesson:

- Will the server be able to handle the additional load of the second network?
- Will the total delay across the network be acceptable once the second network is installed?

To answer these questions, you need a snapshot of current performance for comparison. To get this baseline, you will collect one object statistic, **Server Load**, and one global statistic, **Ethernet Delay**.

Server load is a key statistic that reflects the performance of the entire network. To collect statistics related to the server's load, do the following steps:

1. Right-click on the server node (**node_31**) and select **Choose Individual DES Statistics** from the server Object pop-up menu.

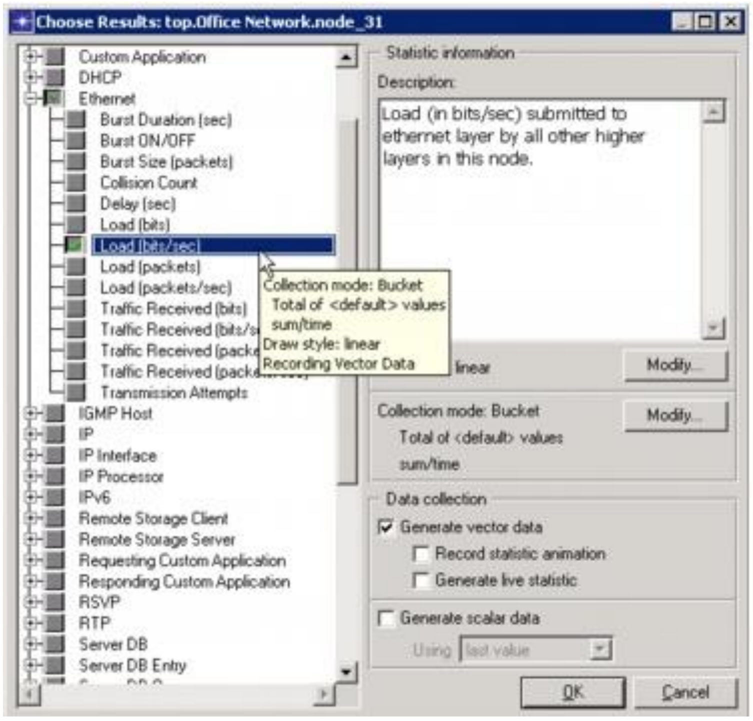
   The Choose Results dialog box for node_31 appears.

   The Choose Results dialog box hierarchically organizes the statistics you may collect.

2. To collect the Ethernet load on the server:
3. Expand the treeview for **Ethernet** in the **Choose Results** dialog box to see the Ethernet statistic hierarchy.

   Note—The list of statistics can vary from the ones shown in this tutorial; they depend on the set of software components you have installed.

   Choose Results Dialog Box



Choose Results: top.Office Network.node_31

- Custom Application
- DHCP
- Ethernet
  - Burst Duration (sec)
  - Burst ON/OFF
  - Burst Size (packets)
  - Collision Count
  - Delay (sec)
  - Load (bits)
  - Load (bits/sec)
  - Load (packets)
  - Load (packets/sec)
  - Traffic Received (bits)
  - Traffic Received (bits/s
  - Traffic Received (packe
  - Traffic Received (pack
  - Transmission Attempts
- IGMP Host
- IP
- IP Interface
- IP Processor
- IPv6
- Remote Storage Client
- Remote Storage Server
- Requesting Custom Application
- Responding Custom Application
- RSVP
- RTP
- Server DB
- Server DB Entry

Collection mode: Bucket
Total of <default> values
sum/time
Draw style: linear
Recording Vector Data

Statistic information
Description:
Load (in bits/sec) submitted to ethernet layer by all other higher layers in this node.

linear Modify...

Collection mode: Bucket
Total of <default> values
sum/time
Modify...

Data collection
[✓] Generate vector data
[ ] Record statistic animation
[ ] Generate live statistic
[ ] Generate scalar data
Using last value

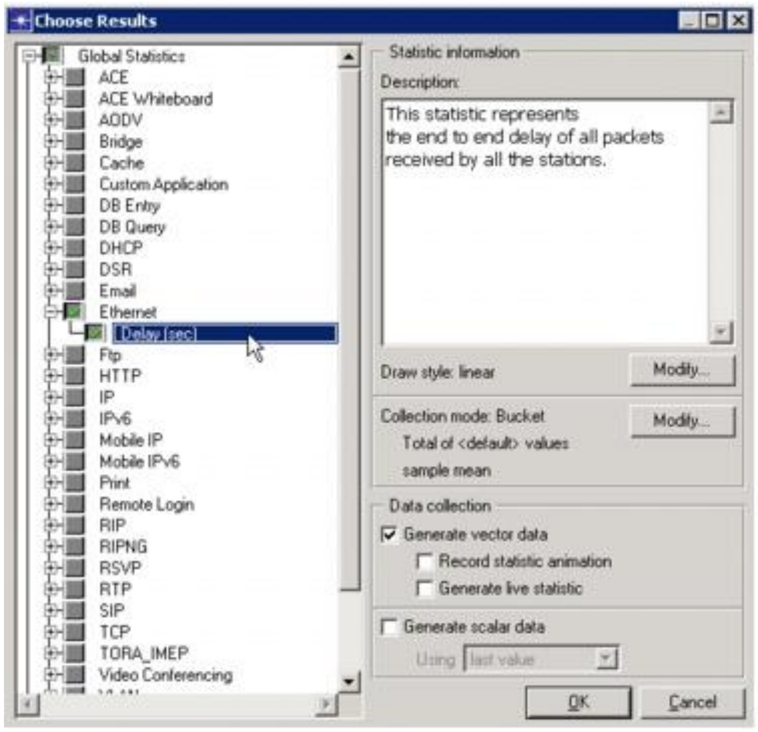
OK Cancel

4. Click the checkbox next to **Load (bits/sec)** to enable collection for that statistic.
5. Click **OK** to close the dialog box.

Global statistics can be used to gather information about the network as a whole. For example, you can find out the delay for the entire network by collecting the global **Delay** statistic:

1. Right-click in the workspace (but not on an object) and select **Choose Individual DES Statistics** from the workspace pop-up menu.

   Global Statistic Chosen



This statistic represents the end to end delay of all packets received by all the stations.

2. Expand the Global Statistics hierarchy.
3. Expand the Ethernet hierarchy.
4. Click the checkbox next to **Delay (sec)** to enable data collection.
5. Click **OK** to close the Choose Results dialog box.
6. It is good to get into the habit of saving your project every so often. Choose **File > Save**, then click **Save** (the project already has a name, so you don't need to rename it).

Now that you have specified the statistics to collect and saved the project, you are almost ready to run your simulation.

First, though, verify that your **Network Simulation Repositories** preference is set appropriately.

1. Choose **Edit > Preferences**.
2. Type **network sim** in the **Search for:** field and click the **Find** button.
3. If the **Value** field for the **Network Simulation Repositories** preference is not **stdmod**, click on the field.

   The **Network Simulation Repositories** dialog box opens.

4. Click the **Insert** button, then type **stdmod** in the field.

5. Click **OK** twice to close the **Network Simulation Repositories** and **Preferences** dialog boxes.

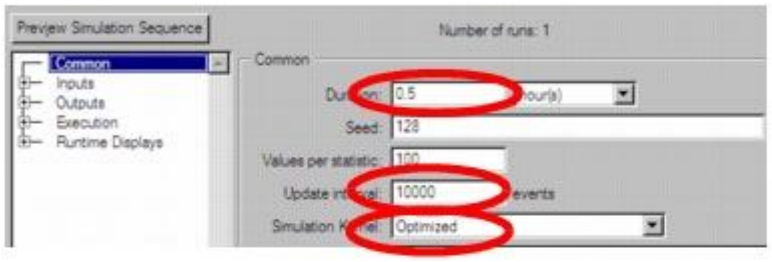
To run a simulation:

1. Select **DES > Configure/Run Discrete Event Simulation...**

   You can also open the Configure Discrete Event Simulation dialog box by clicking on the **Configure/Run Discrete Event Simulation (DES)** tool button.

2. Click on the **Detailed...** button, if it is present.

   Configure Discrete Event Simulation Dialog Box



Number of runs: 1

Common

Duration: 0.5 hour(s)

Seed: 128

Values per statistic: 100

Update interval: 10000 events

Simulation Kernel: Optimized

3. Type **0.5** in the **Duration:** field to simulate one-half hour of network activity.
4. Type **10000** (events) in the **Update interval:** field to specify how often the simulation calculates events/second data.

   In this case, the simulation calculates and displays events/second data at 10,000-event intervals. The default setting for this is 500,000 for larger network simulations.

5. Set the Simulation Kernel to **Optimized**.

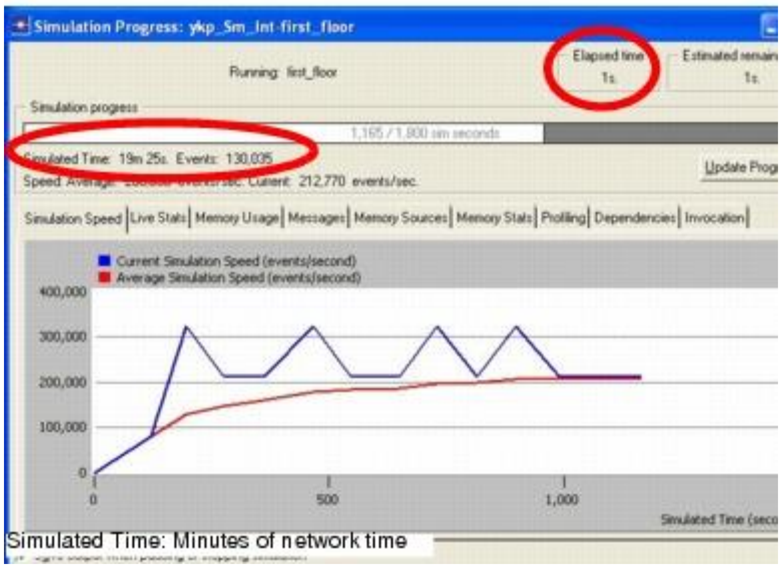
   You can use one of two types of kernels to run your simulation. The development kernel collects simulation data you can use to debug your models, but the optimized kernel runs faster.

6. Click the **Run** button to begin the simulation.

   While the simulation runs, a dialog box appears showing the simulation's progress.

   Simulation Progress Dialog Box

Elapsed Time: Number of seconds the simulation has run


Simulation Progress: ykp_Sm_Int-first_floor
Running: first_floor
Elapsed time
1s
Estimated remain
1s
Simulation progress
1,165 / 1,800 sim seconds
Simulated Time: 19m 25s Events: 130,035
Speed: Average: ... events/sec Current: 212,770 events/sec
Update Prog
Simulation Speed
Live Stats
Memory Usage
Messages
Memory Sources
Memory Stats
Profiling
Dependencies
Invocation
Current Simulation Speed (events/second)
Average Simulation Speed (events/second)
400,000
300,000
200,000
100,000
0
0
500
1,000
Simulated Time (seco

Simulated Time: Minutes of network time

The dialog box above shows that, in 1 second of elapsed (actual) time, OPNET Modeler has simulated 19 minutes and 25 seconds of network time. The entire simulation should take less than one minute to complete—the elapsed time varies according to the speed of your computer.

7. When the simulation finishes, the contents of the Messages tab appears. Click the **Close** button in the Simulation Sequence dialog box.
8. If your simulation does not complete, if no results were collected, or if the results vary significantly from those shown, you will have to troubleshoot your simulation. See *"Troubleshooting Tutorials"*.

Viewing Results

After your simulation has executed, you will want to see the information collected for each statistic. There are several ways to view results; in this lesson you will use the View Results option in the Workspace pop-up menu.

You will learn different ways to view results in later lessons.

To view the server Ethernet load for the simulation:

1. Right-click on the server node (**node_31**) choose **View Results** from the server's Object pop-up menu.

   The Results Browser opens.

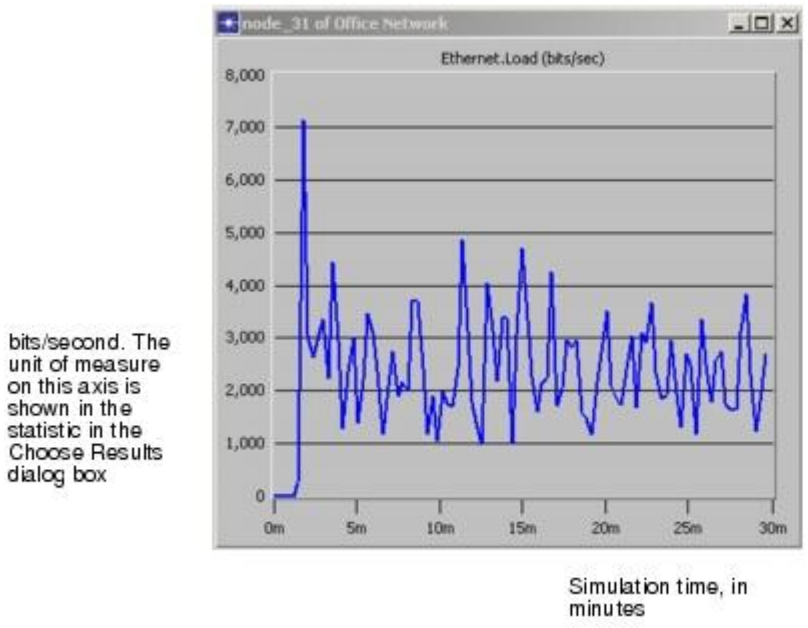
2. Expand the **Office network.node_31 > Ethernet** hierarchy.
3. Click on the checkbox next to **Load (bits/sec)** to indicate that you want to view that result.

4. Click the **Show** button in the Results Browser.

   The graph of the server load appears in the Project Editor, as shown in the following figure.

The graph of the server load should resemble the following graph. Your results may differ slightly due to differences in node placement and link length, but the general trends should be consistent.

Server Load Graph



Note that at its peak, the load on the server is about 7,000 bits/second. You will need this baseline for comparison after you add the second network.

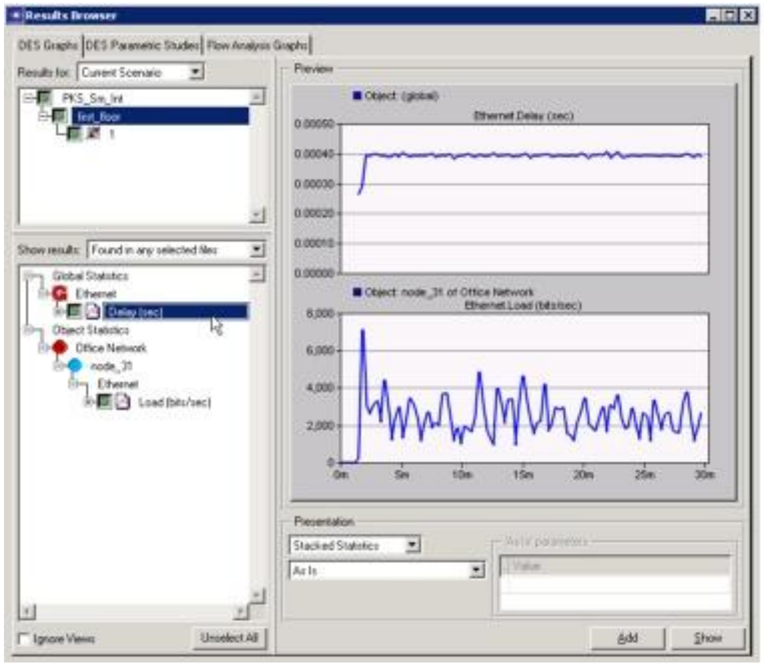
When you finish viewing the server load graph, close this dialog box and the Results Browser. (If the system prompts you, choose to delete the graph panel.)

The **View Results** option from the Workspace pop-up menu allows you to obtain global statistics and individual object statistics from one treeview.

You also should look at the Global Ethernet Delay on the network. To view this statistic:

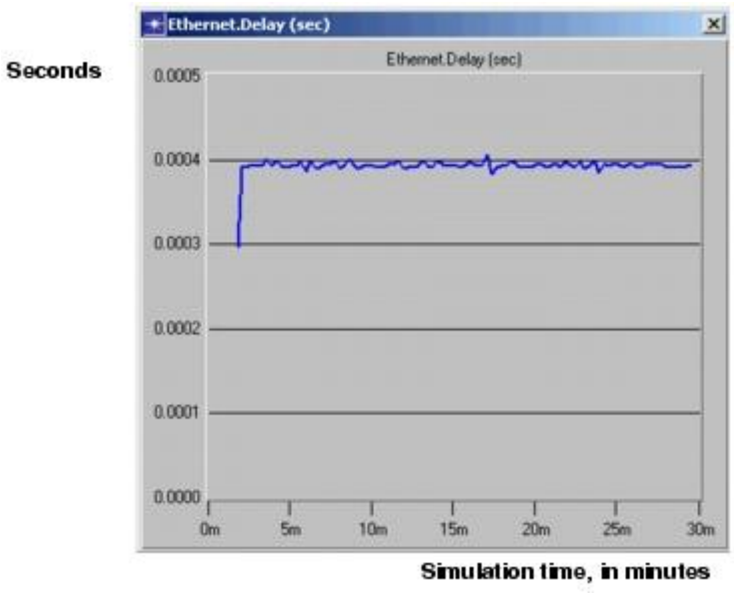
1. Right-click in the workspace, then select **View Results** from the pop-up menu.
2. Check the box next to **Global Statistics > Ethernet > Delay (sec)**.
3. Check the box next to **Object Statistics > Office Network.node_31 > Ethernet > Load (bits/sec)**, and then click the Show button to view the Ethernet delay for the whole network.

   Viewing Ethernet Delay for the Whole Network



The Ethernet delay graph appears in the Project Editor. The graph should resemble the following figure.

Ethernet Delay Graph



Note that after the network reaches steady state the maximum delay is around 0.4 milliseconds.

When you finish viewing the graph, close the graph and the Results Browser.

Expanding the Network

You have created a baseline network and gathered statistics about it. Now you are ready to expand the network and verify that it still operates sufficiently well with the additional load.

When performing a "what-if" comparison, it is convenient to store the baseline network as one scenario and create the experimental network as a different scenario. You will duplicate the existing scenario and make changes to it instead of building the new topology from the beginning.

To duplicate a scenario:

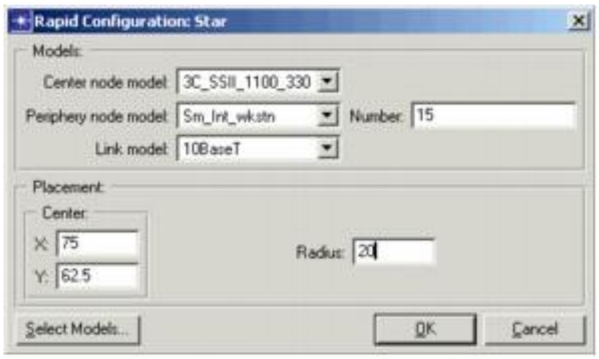
1. Choose **Scenarios > Duplicate Scenario...**
2. Enter **expansion** as the name for the new scenario.
3. Click **OK**.

   The scenario, with all the nodes, links, statistics, and the simulation configuration, is duplicated and named **expansion**.

The second-floor segment will resemble the first-floor segment, but will not have a server of its own. To build the new segment:

1. Select **Topology > Rapid Configuration**.
2. Choose **Star** for the topology and click **Next...**
3. Complete the Rapid Configuration dialog box with these values:
   - Center Node Model: **3C_SSII_1100_3300_4s_ae52_e48_ge3**
   - Periphery Node Model: **Sm_Int_wkstn**
   - Number: **15**
   - Link model: **10BaseT**
   - X: **75**, Y: **62.5**, Radius: **20**

   Rapid Configuration Dialog Box



Rapid Configuration: Star

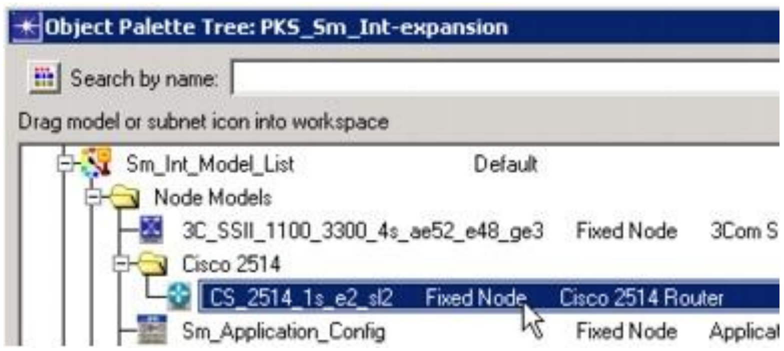
Models:
Center node model: 3C_SSII_1100_330
Periphery node model: Sm_Int_wkstn
Number: 15
Link model: 10BaseT

Placement:
Center:
X: 75
Y: 62.5
Radius: 20

4. Click **OK** to create the network.

Join the two networks:

1. If it is not already open, click the tool button to open the object palette.
2. Expand the Cisco 2514 folder.

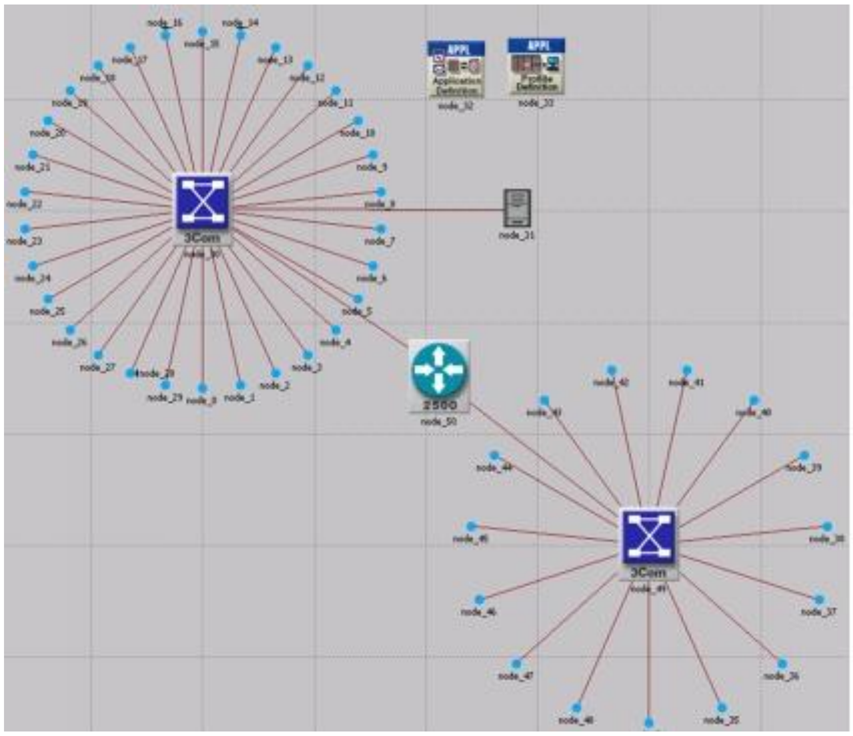


```
Object Palette Tree: PKS_Sm_Int-expansion
Search by name:
Drag model or subnet icon into workspace
Sm_Int_Model_List                           Default
  Node Models
    3C_SSII_1100_3300_4s_ae52_e48_ge3   Fixed Node   3Com S
    Cisco 2514
      CS_2514_1s_e2_sl2   Fixed Node   Cisco 2514 Router
    Sm_Application_Config               Fixed Node   Applicat
```

3. Drag the **Cisco 2514** router icon into the workspace between the two networks. Right-click to turn off node creation.
4. Expand the Link Models folder and double-click on the **10BaseT** link icon in the object palette.
5. Create **10BaseT** links between the Cisco router (**node_50**) and the 3Com switches at the center of each star.
6. Right-click to turn off link creation.
7. Close the object palette.
8. Select **File > Save**.

The final network should look like this:

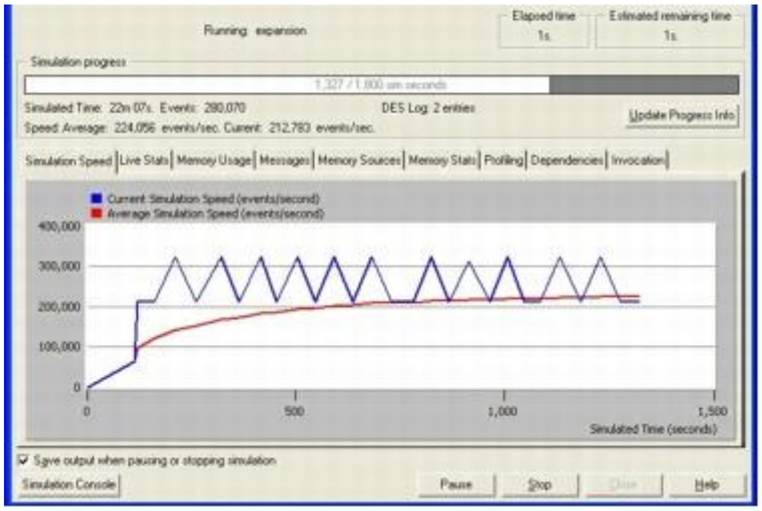
The Final Network



To run the expansion scenario:

1. Select **DES > Configure/Run Discrete Event Simulation...**
2. Click the **Detailed...** button, if it appears, and verify that the **Duration** is set to **0.5** hours and the **Update interval** is set to **10000**.
3. Click the **Run** button to begin the simulation.

Simulation Progress Dialog Box, Simulation Speed Tab Selected


Running: expansion

Elapsed time: 1s

Estimated remaining time: 1s

Simulation progress

1,327 / 1,800 sim seconds

Simulated Time: 22m 07s. Events: 280,070

DES Log: 2 entries

Speed: Average: 224,056 events/sec. Current: 212,783 events/sec.

As before, a window displays simulation start-up messages first, and then an animated graph shows both the current and average speed in events per second during the simulation. When the simulation is completed, you can view the event/second graph results from the Simulation Speed tab.

4. When the simulation is done, close the Simulation Progress dialog box. If you had problems, see *"Troubleshooting Tutorials"*.

Comparing Results

To answer the questions posed about the addition of a second network to the existing LAN, you need to compare the results from both of the simulations you ran.

You will use the **View Results** menu item in the Object and Workspace pop-up menus to combine statistics from different scenarios in the same graph.

To view the server load from both scenarios at once:

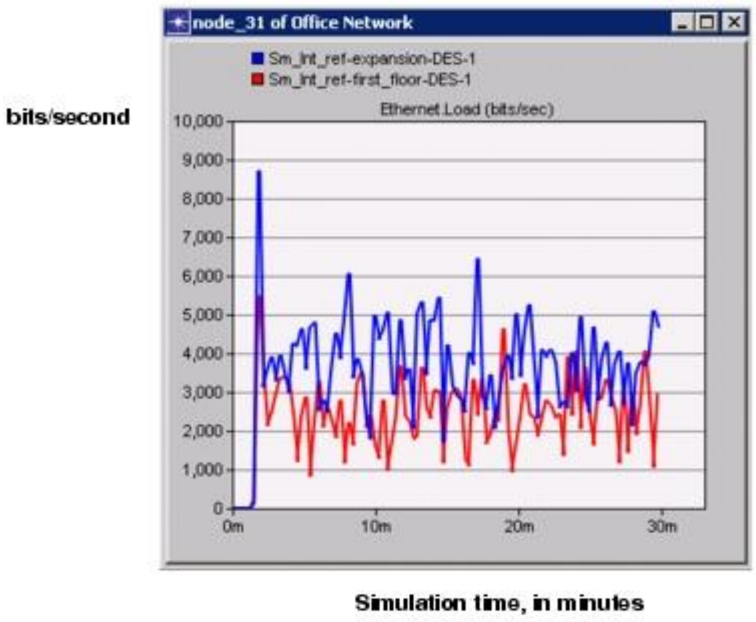
1. Right-click on the server node (**node_31**) to display the pop-up menu and choose **View Results**.
2. Select **Current Project** from the pull down menu for Results.
3. Check the box next to both the scenarios listed.
4. Select Overlaid Statistics from the pull down menu for Presentation.

If your results differ radically from those shown in the following figures, you will have to troubleshoot your simulation. See *"Troubleshooting Tutorials"*.

When comparing results, choosing a statistic in one scenario produces a graph showing the value of that statistic in all scenarios. To view the results:

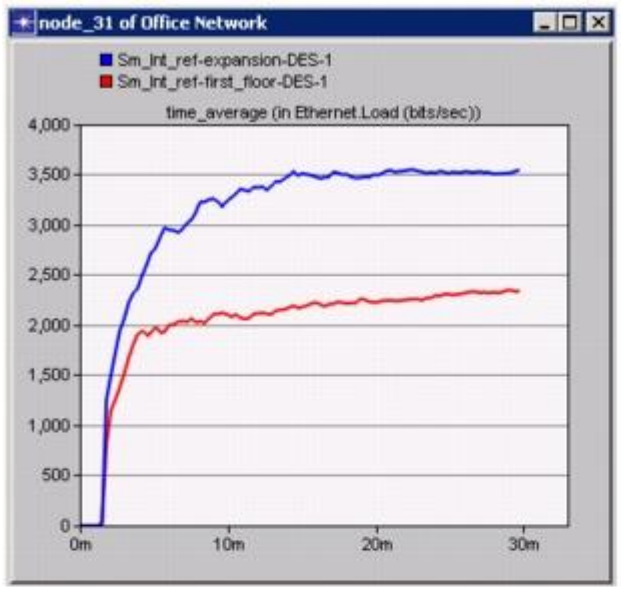
1. Select the **Office Network.node_31 > Ethernet > Load (bits/sec)** statistic and click the **Show** button. Your results should resemble those in the following figure (but may not be identical):

   Ethernet Load Compared



The following graph is the time average of the Ethernet load between the baseline (first_floor) scenario and the expansion scenario. You will learn how to create a graph of the time average in the next lesson.

Time-Averaged Server Load Compared



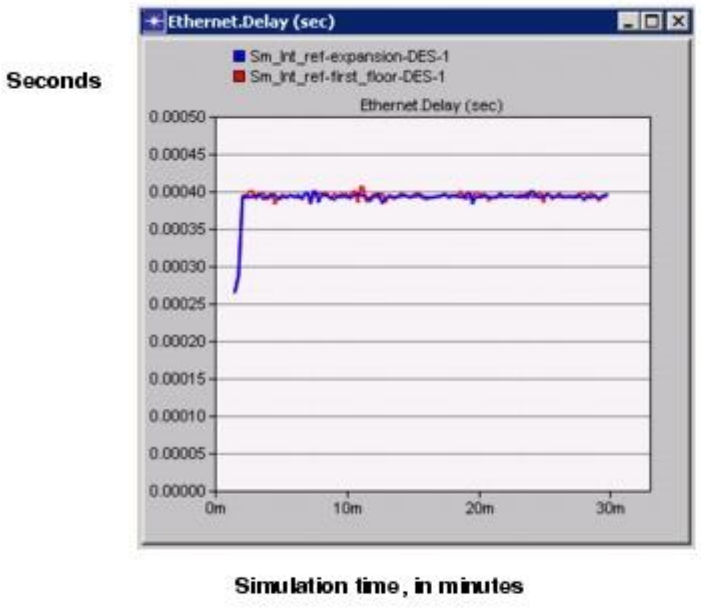
Note that while the average load for the expansion scenario is higher (as expected), the load as a whole appears to be leveling off (that is, not monotonically increasing), indicating a stable network.

The last step is to see how much the network's delay is affected by adding a second floor. To compare Ethernet delay for the two scenarios:

1. Close the graph and the Results Browser for the server.
2. Right-click in the workspace to display the pop-up menu and choose **View Results**.
3. Under Show Results, select **Global Statistics** -> **Ethernet** -> **Delay (sec)**.
4. Click the **Show** button to display the graph.

   Your graph of Ethernet Delay should resemble the following:

   Ethernet Delay Compared



   This graph shows that there is no significant change in Ethernet delay on the network. Although server load has increased, delay has not.

5. Close the graph and the Results Browser.
6. Select **File** > **Close** and save changes before closing.

Now you are ready to go on to the M/M/1 Queue tutorial. This lesson explores the use of node models in the workflow. Return to the main tutorial menu and choose **M/M/1 Queue** from the list of available lessons. Or choose another tutorial of interest.

Note—Be sure to delete the **stdmod** setting for the **Network Simulation Repositories** preference when you are finished doing tutorials. To delete the setting, select **Edit > Preferences**, search for **Network Simulation Repositories**, click on the value, and choose **Delete**.

Práctica 17

Modelado de Redes LAN (LAN Modeling)

Overview

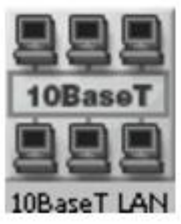
This lesson focuses on the use of LAN models and background link utilization. You will learn to

- Configure the object palette with the models you need
- Set up application and profile configurations
- Model a LAN as a single node
- Specify background utilization that changes over time on a link
- Simulate multiple scenarios simultaneously
- Apply filters to graphs of results and analyze the results

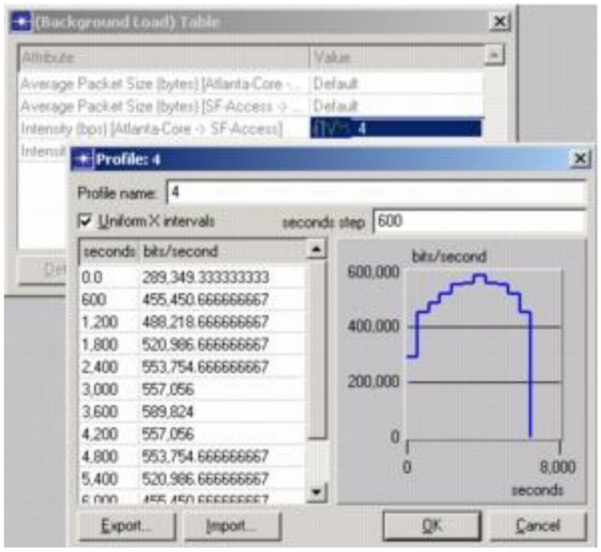
This lesson focuses on two features, **LAN models**, which model entire LANs, and the link **Background Load** attribute, which is used to model existing link traffic.

- **LAN Models:** If you are modeling a medium or large

  internetwork, you may only be interested in specific aspects of the network's behavior (whether a key router will be overloaded, for example). In such cases, single nodes can model entire LANs.
- **Link Background Load:** Use the **Background Load** attribute to model existing traffic on a link instead of explicitly modeling each packet. You can also specify changes in this background traffic over the course of a simulation.



Background Load Attribute Dialog Box



(Background Load) Table

Attribute	Value
Average Packet Size (bytes) [Atlanta-Core -...	Default
Average Packet Size (bytes) [SF-Access -> ...	Default
Intensity (bps) [Atlanta-Core -> SF-Access]	4

Profile: 4

Profile name: 4

Uniform X intervals ✓ seconds step 600

seconds	bits/second
0.0	289,349.333333333
600	455,450.666666667
1,200	488,218.666666667
1,800	520,986.666666667
2,400	553,754.666666667
3,000	557,056
3,600	589,824
4,200	557,056
4,800	553,754.666666667
5,400	520,986.666666667
6,000	455,450.666666667

Setting Up the Scenario

In this lesson, your job is to model an east coast company's WAN. The company has offices in Atlanta, Philadelphia, New York, and Boston, which are connected to the central network in Washington, D.C. The offices use phone lines to connect to each other, and are therefore susceptible to delays caused by additional, unrelated traffic on the lines.

This company wants you to determine how this background traffic is affecting FTP traffic on their network. To do this, you will model FTP performance on the network, first without background traffic and then with background traffic. Because you are not interested in modeling the details of each office's LAN, you will use LAN models to model the individual LANs as single nodes.

The first step in setting up the WAN is to specify the overall context for the network with the Startup Wizard. Once that is done, you can proceed with building the network itself. This topic focuses on:

- Configuring an object palette
- Specifying a map background
- Zooming in on a background

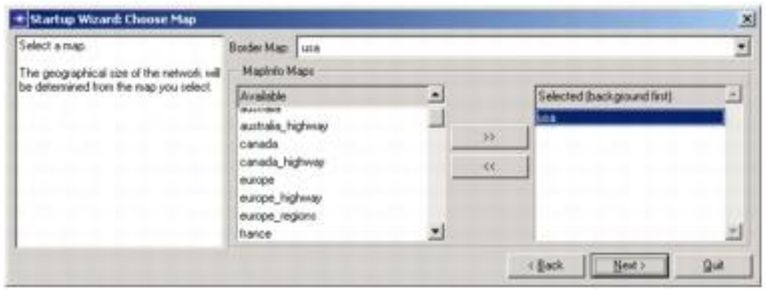
Begin by opening a new project in OPNET Modeler and configuring the scenario context using the Startup Wizard:

1. If it is not already running, start it, select **File > New... > Project**, then click **OK**.
2. Name the new project **<initials>_LAN_Mod** and the scenario **no_back_load**, then click **OK**.
3. Click **Next >** when the Startup Wizard opens and select "Create empty scenario" for the **Initial Topology**.

You can specify a map to use as a background for your network. To set a map background:

1. Click **Choose from maps** for **Network Scale** and click **Next >**.
2. Choose **usa** from the **Border Map** list.
3. Choose **usa** from the **MapInfo Maps** list and click **>>** to move it to the **Selected (background first)** pane.

   Choose Map Wizard



Startup Wizard: Choose Map

Select a map

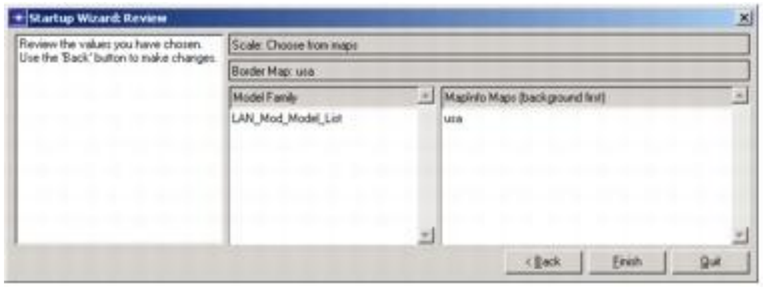
The geographical size of the network will be determined from the map you select.

Border Map: usa

MapInfo Maps

Available	Selected (background first)
australia_highway	usa
canada	
canada_highway	
europe	
europe_highway	
europe_regions	
france	

4. Click **Next >**.
5. Select **LAN_Mod_Model_List** to be included in your network by clicking on the **Include?** cell and changing the value from **No** to **Yes**. Click **Next >**.
6. Review your settings and click **Finish** to close the Startup Wizard.



Startup Wizard: Review

Review the values you have chosen.
Use the 'Back' button to make changes.

Scale: Choose from maps

Border Map: usa

Model Family	MapInfo Maps (background first)
LAN_Mod_Model_List	usa

The workspace now shows the specified map and object palette.

OPNET Modeler's full set of node and link models would be overwhelming to work with all at once, so the object palette can be configured to show only a specific subset, or model list. You can use the standard model lists, adapt them for your own needs, or make your own list.

You can create your own custom object palettes for use in modeling projects.

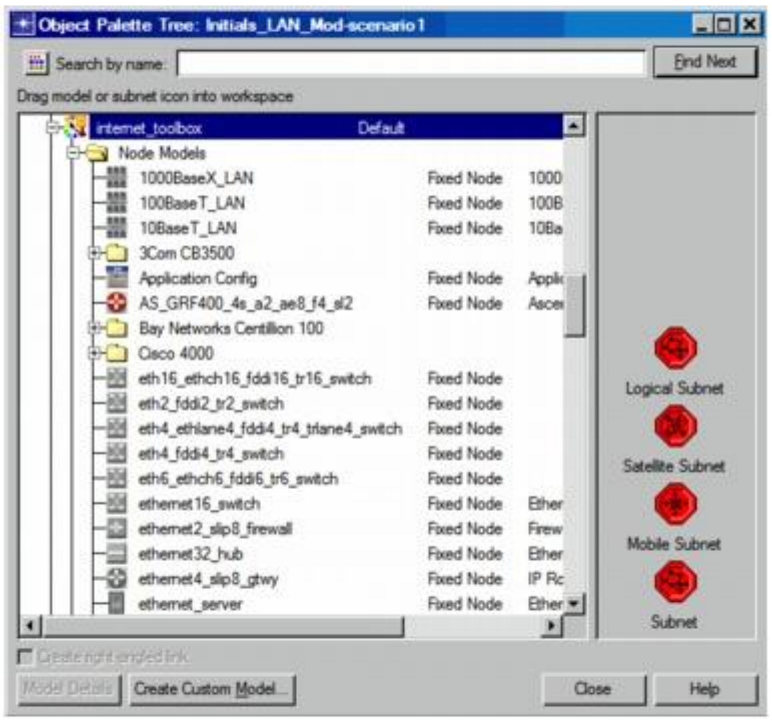
For this lesson, we created the **LAN_Mod_Model_List**. Now you will adapt that model list by adding the LAN node model to it:

1. Click the **Object Palette Icon** (or choose **Open Object Palette** from the **Topology** menu if Object Palette dialog box is not open).

   Object Palette Icon

   The Object Palette dialog box opens.

   Object Palette Dialog Box



```
Object Palette Tree: Initials_LAN_Mod-scenario1
Search by name:                                   Find Next
Drag model or subnet icon into workspace
internet_toolbox                        Default
  Node Models
    1000BaseX_LAN                        Fixed Node   1000
    100BaseT_LAN                         Fixed Node   100B
    10BaseT_LAN                          Fixed Node   10Ba
    3Com CB3500
    Application Config                   Fixed Node   Applic
    AS_GRF400_4s_a2_ae8_f4_sl2           Fixed Node   Ascer
    Bay Networks Centillion 100
    Cisco 4000
    eth16_ethch16_fddi16_tr16_switch     Fixed Node
    eth2_fddi2_tr2_switch                Fixed Node
    eth4_ethlane4_fddi4_tr4_trlane4_switch Fixed Node
    eth4_fddi4_tr4_switch                Fixed Node
    eth6_ethch6_fddi6_tr6_switch         Fixed Node
    ethernet16_switch                    Fixed Node   Ether
    ethernet2_slip8_firewall             Fixed Node   Firew
    ethernet32_hub                       Fixed Node   Ether
    ethernet4_slip8_gtwy                 Fixed Node   IP Ro
    ethernet_server                      Fixed Node   Ether
Create right angled link
```

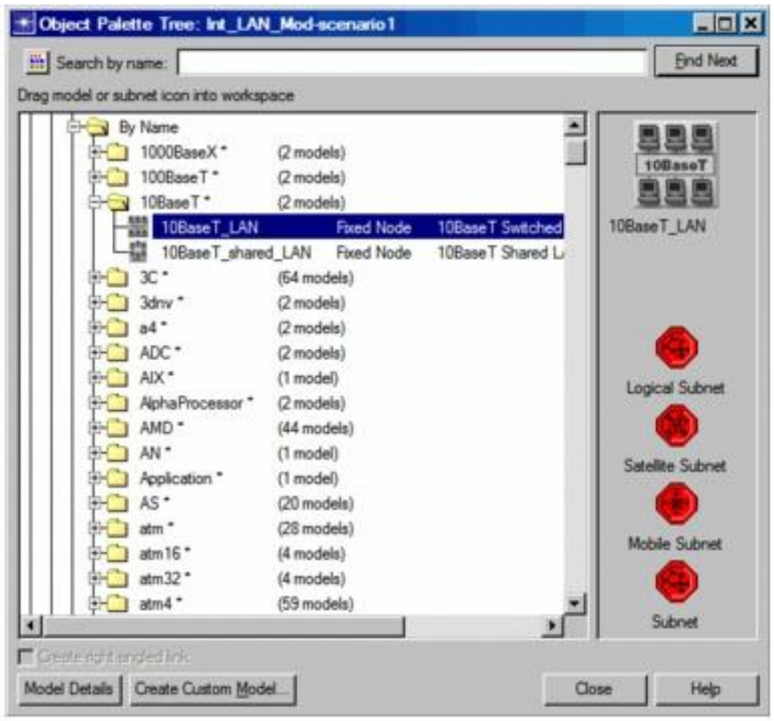
   The Object Palette dialog box lets you change the object palette and then save it.

2. Scroll to the top of folder treeview and click the **Node Models** folder in the Object Palette dialog box.

   Under Node Models, click **Fixed Node Models > By Name >10BaseT**.

3. Under 10BaseT, right click on **10BaseT_LAN** and click on **Add to Default Palette**: **LAN_Mod_Model_List**.

   10BaseT LAN Added to Object Palette



By Name
1000BaseX * (2 models)
100BaseT * (2 models)
10BaseT * (2 models)
10BaseT_LAN Fixed Node 10BaseT Switched
10BaseT_shared_LAN Fixed Node 10BaseT Shared L
3C * (64 models)
3dnv * (2 models)
a4 * (2 models)
ADC * (2 models)
AIX * (1 model)
AlphaProcessor * (2 models)
AMD * (44 models)
AN * (1 model)
Application * (1 model)
AS * (20 models)
atm * (28 models)
atm16 * (4 models)
atm32 * (4 models)
atm4 * (59 models)

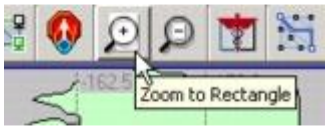
The **10BaseT_LAN** icon appears in the **LAN_Mod_Model_List** in the object palette.

4. Click **Close** on the Object Palette.

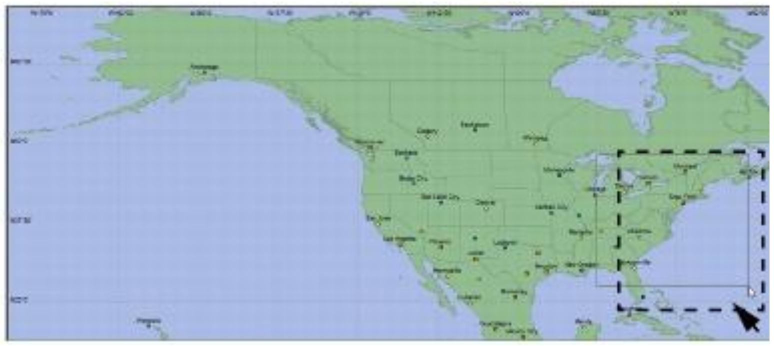
Because the network you will create is on the east coast of the U.S., there is no need to view the entire map. To zoom in on the east coast:

1. Click the **Zoom** toolbar button.

2. Select the top left corner of the area to zoom in on and drag a rectangle to bottom right corner. Be certain that both Atlanta and Boston are included in the zoomed view.



Using the Zoom Button



Tip: If you are not happy with your first zoom, you can click the **Zoom to Previous** toolbar button, next to the **Zoom** toolbar button, and then zoom in again.

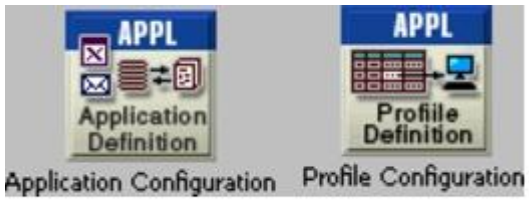
3. Continue to zoom until the names of cities, including Boston and Atlanta, appear on the map.

Configuring Applications

It is a good idea to define the profiles and applications that will be used by the LAN before you begin constructing the network. You define the profiles in the profile definition object and applications in the application definition object.

- **A profile** is applied to a workstation, server, or LAN. It specifies the applications used by a particular group of users. You might have one profile for Marketing (heavy use of email; light use of file transfer) and another profile for Engineering (light use of email; heavy use of file transfer).
- **An application** may be any of the common applications (email, file transfer) or a custom application you define. Eight common ("standard") applications are already defined: Database Access, Email, File Transfer, File Print, Telnet Session, Video conferencing, Voice over IP Call, and Web Browsing.

Profile Definition and Application Definition Objects



To Configure the Application Configuration Object

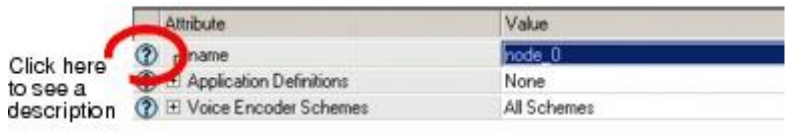
Follow these steps.

1. Open the object palette (if it is not already open).
2. Drag an **Application Config** object to the project workspace.
3. Right-click on the new object and select **Edit Attributes**.

   The Attributes dialog box opens.

4. Click on the question mark next to the **name** attribute to see a description of the attribute. Close the attribute description dialog box when done.

Application Definitions Set to Default



Attribute	Value
name	node_0
Application Definitions	None
Voice Encoder Schemes	All Schemes

5. Set the **name** attribute to **Application Configuration**.
6. Change the **Application Definitions** attribute to **Default** by clicking in the attribute's Value column and selecting **Default** from the drop-down list.

   Selecting **Default** configures the application definition object to make 16 pre-configured applications available for use. These applications are different configurations of the eight standard applications mentioned earlier. Now you will be able to include these applications in the profile you are about to create.

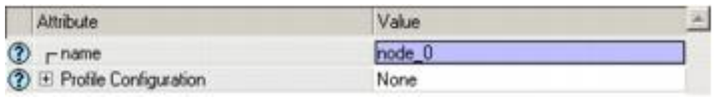
7. Click **OK** to accept the changes and close the **Attributes** dialog box.

To Configure the Profile Configuration Object

Follow these steps.

1. Drag a **Profile Config** object from the object palette to the project workspace.
2. Right-click on the object and select **Edit Attributes.**
3. Set the **name** attribute to **Profile Configuration**.
4. Change the **Profile Configuration** attribute by clicking in its Value column and selecting **Edit...** from the drop-down list.

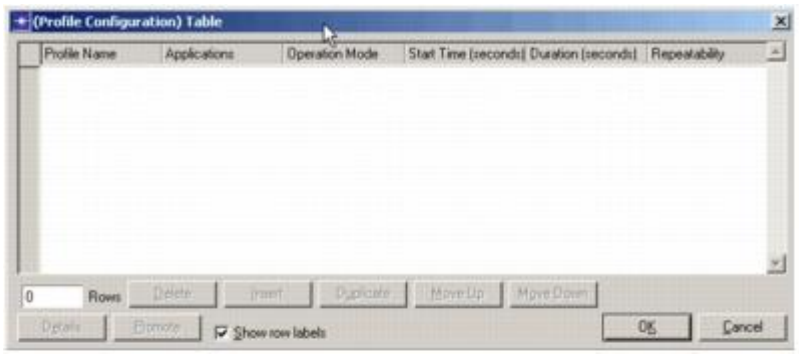
   Selecting Edit... from the Menu



Attribute	Value
name	node_0
Profile Configuration	None

   The Profile Configuration dialog box appears.

   Profile Configuration Dialog Box

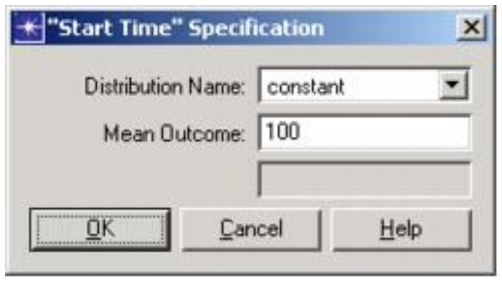


To Define a New Profile and Add It to the Table

Follow these steps.

1. Change the number of rows to **1**.
2. Name the new profile **LAN Client** by clicking in the Profile Name column of the first row.
3. Skip the next column, Application. We'll configure the other attributes first and come back to this one later.
4. Change the **Operation Mode** to **Simultaneous**.
5. Click in the profile's **Start Time (seconds)** cell to open the Start Time Specification dialog box.
6. Select **constant** from the **Distribution Name** pull-down menu.

Start Time Specification Dialog Box



Distribution Name: constant
Mean Outcome: 100

7. Set **Mean Outcome** to 100, then click **OK** to close the Start Time Specification dialog box.

   The Start Time attribute has a value of **constant (100)**.

   Do not close the profile configuration attribute dialog box.

You will be modeling FTP performance. That application should be included in the profile.

1. Click in the Applications column and choose **Edit...** from the pop-up menu.

   The **Applications** dialog box appears.

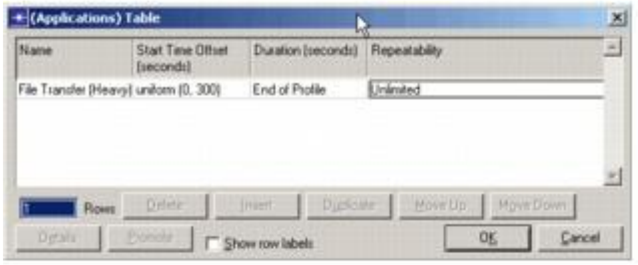
2. Change the number of rows to **1**.
3. Set the **Name** to **File Transfer (Heavy)** by clicking in the cell and selecting the application from the drop-down list.

   The contents of the drop-down list are controlled by the Applications Configuration object. When you selected **Default** as the value for the **Application Definitions** attribute in this object, you enabled this list of applications.

Note that the list includes 16 entries, a heavy and light version for each of the eight standard applications.

4. Set **Start Time Offset** to **uniform (0, 300).**
   a. Verify that the **Distribution Name** is **uniform**.
   b. Set **Minimum Outcome** to **0**.
   c. Set **Maximum Outcome** to **300**, then click **OK**.
5. Verify that the completed dialog box looks like this:

Completed Applications Dialog Box



(Applications) Table

Name	Start Time Offset (seconds)	Duration (seconds)	Repeatability
File Transfer (Heavy)	uniform (0, 300)	End of Profile	Unlimited

6. Click **OK** to close the Applications Table dialog box.
7. Click **OK** to close the Profile Configuration Table, then click **OK** again to close the Attributes dialog box.

Building the Network

Now that you have set up the scenario, configured the application, and created a profile that uses the application, you are ready to begin constructing the WAN. Because the network contains four identical subnets, you can create the first subnet in Atlanta, with its nodes inside it, and then copy the subnet to Boston, New York, and Philadelphia. You will also copy it to Washington, D.C. and modify it further. This topic focuses on:

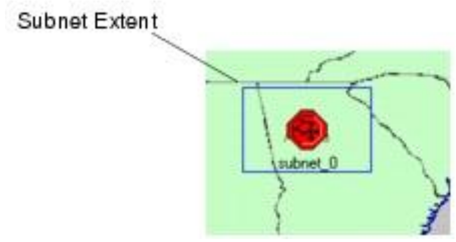
- Building subnets
- Copying and pasting network objects
- Modifying subnets
- Connecting subnets

A **subnet** is a single network object that contains other network objects (links, nodes, and other subnets). Subnetworks allow you to simplify the display of a complex network through abstraction.

Subnets help you logically organize your network model. You can nest subnets within subnets to an unlimited degree. For this lesson, you can use subnets as logical containers for the offices in each of the cities. To create a subnet:

1. Open the object palette if it is not already open, and move it to the lower right corner of the screen so that it is out of the way.
2. Place a fixed subnet over Atlanta.
   a. Click the subnet icon in the object palette and drag it to the workspace.
   b. Right-click to turn off node creation.
3. Modify the subnet extent of the Atlanta subnet. The subnet extent is the geographic area covered by the subnet, which may be much larger than the actual area you wish to model.

   Subnet Extent

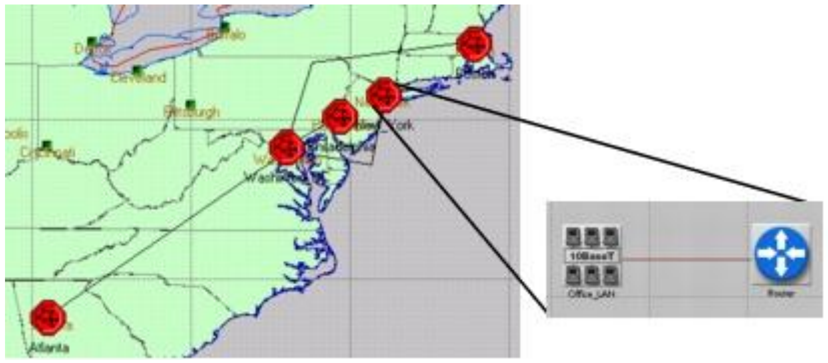
   a. Right-click and select **Edit Attributes (Advanced)**.
   b. Set the name attribute to Atlanta.
   c. Change the **x span** and **y span** attributes to **.25**. The unit of measure of these attributes is determined by the unit of measure of the top-level area, degrees in this case. The area covered by .25 is the area covered by one-quarter of one degree of latitude or longitude.
   d. Click **OK**.

      Notice that the subnet extent is now much smaller. (It may be hidden underneath the subnet icon.)

Double-clicking on a subnet object allows you to see what is "inside" the subnet.

When you double-click on a subnet, OPNET Modeler changes the view to show you what is inside that subnet. Subnets can contain nodes, links, and other subnets. An example topology follows. You will not see this topology right now. You are about to create it.

A Subnet Hierarchy



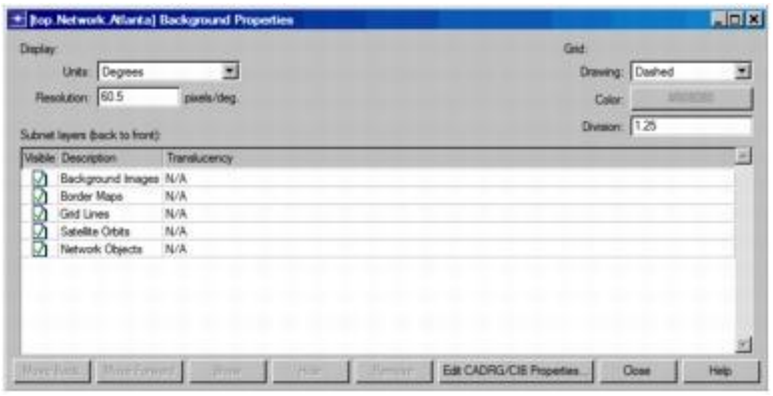
Initially, a subnet's grid properties are based on its parent subnet. If the subnet's grid settings are inappropriate, you can change them to fit your network. To change the grid inside a subnet:

1. **Double-click** on the Atlanta subnet.
2. Select **View > Background > Set Properties**.

   Note that the display grid is in degrees, which is not appropriate for an office.

3. Set units to **Meters**.
4. Set resolution to **10 pixels/m**.
5. Uncheck the Visible checkbox for Satellite orbits.
6. Verify that Drawing is set to **Dashed**.
7. Set division to **10**.

   Set View Properties Dialog Box



[top.Network.Atlanta] Background Properties

Display

Units: Degrees

Resolution: 60.5 pixels/deg.

Grid

Drawing: Dashed

Color:

Division: 1.25

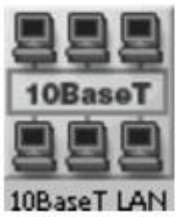
Subnet layers (back to front):

Visible	Description	Translucency
✓	Background Images	N/A
✓	Border Maps	N/A
✓	Grid Lines	N/A
✓	Satellite Orbits	N/A
✓	Network Objects	N/A

8. Click the **Close** button.

Modeling the East Coast company's network does not require modeling the precise nature of each node in each subnet, so you can represent the subnets with a LAN model. To create a LAN model:

1. Place a **10BaseT_LAN** in the workspace.



2. Right-click on the **10BaseT_LAN** and choose the **Edit Attributes** menu item.

You can change a LAN model's attributes so that it represents a network with a certain number of workstations and a particular traffic profile. To represent one of the East Coast company's satellite offices:

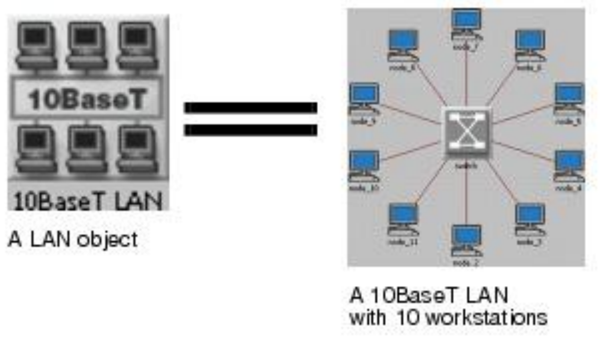
1. Change the LAN model's **name** attribute to **Office_LAN**.
2. Expand the Applications attribute group.
3. Choose **Edit...** for the **Application: Supported Profiles** attribute.
4. Change the number of rows to **1**.
5. Change **Profile Name** to **LAN Client**.

   Now this LAN will use the LAN Client profile you created earlier. This profile includes the File Transfer (Heavy) application. The LAN will receive and send traffic that models heavy FTP use.

6. Set **Number of Clients** to 10, then click **OK**.
7. Click **OK**.

You have now modeled a 10 workstation LAN inside the Atlanta subnet. This single object is equivalent to a 10-workstation star topology LAN.

A Single LAN Object is Equivalent to a LAN Composed of Many Objects



Because this LAN model is composed of workstations and links only, it must be connected to a router. The router can then be connected to other routers in the network.

To create a router:

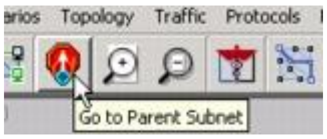
1. Drag a **BN_BLN_4s_e4_f_sl8_tr4** node (a Nortel Networks router) from the object palette to the workspace near the **Office_LAN** node.
2. Name the new node **router**.
3. Connect the **router** and the **Office_LAN** nodes with a **10BaseT** link. Right-click to turn off link creation.

The Atlanta subnet is now configured. Because the subnets in the other cities are identical, you can copy the Atlanta subnet and place it appropriately.

When several subnets or network objects have an identical configuration, you can copy and paste these objects.

To copy the subnet:

1. Return to the parent subnet view by clicking on the **Go to Parent Subnet** button (you can also right-click on the workspace to display the workspace pop-up menu, then choose **Go to Parent Subnet** from the menu).
2. Select the subnet.
3. Select **Edit > Copy** or press **<Control>**+**c**.



Paste the subnet to each of the four different cities:

1. Select **Edit > Paste** or press **<Control>**+**v** and click on the Washington, DC icon.

   Pasting the Subnet

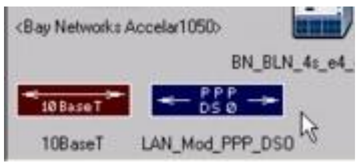
Philadelphia
Washington

   A new subnet appears.

2. Press **<Control>**+**v** again to paste subnets over Philadelphia, New York, and Boston.
3. Right-click on each subnet and select Set Name to rename each city's subnet as follows:
   - Washington, D.C.: **Washington_DC**
   - Philadelphia: **Philadelphia**
   - New York: **New_York**
   - Boston: **Boston**

Connect each subnet to the Washington_DC office:

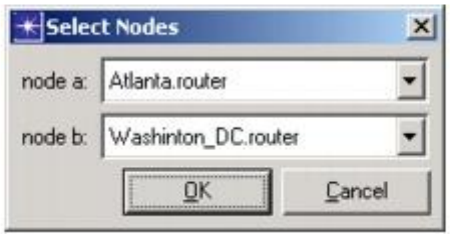
1. Select the **LAN_Mod_PPP_DS0** link in the object palette.
2. Draw a **LAN_Mod_PPP_DS0** link from **Atlanta** to **Washington_DC**.



A Select Nodes dialog box appears asking which nodes in each subnet are to be endpoints of the link.

3. For **node a**, choose the **Atlanta.router** node.
4. For **node b**, choose the **Washington_DC.router** node.

Select Nodes Dialog Box



node a: Atlanta.router

node b: Washinton_DC.router

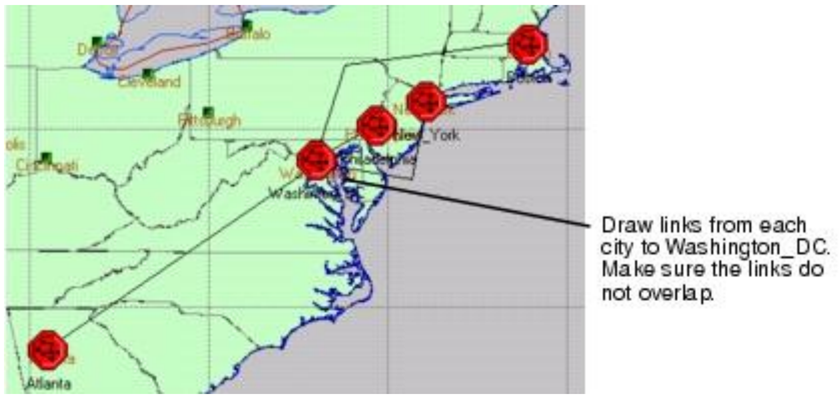
5. Click **OK** to establish the link.
6. Repeat this process, drawing links from each city to **Washington_DC**, specifying each city's router as the links' endpoints.

   To prevent overlapping links, you can click on intermediate points in the workspace to make a link follow a path before finally clicking on its destination node.

7. Right-click to turn off link creation.

The network should resemble the following:

The Initial Topology



To complete the network, the main office in Washington D.C. needs to have a switch and a server added to it. To configure the network in D.C.:

1. Double-click on the **Washington_DC** subnet to enter its subnet view.
2. Place one **<Bay Network Accelar1050>** switch and one **ethernet_server** node in the workspace.
3. Rename the **<Bay Network Accelar1050>** node to **switch**.

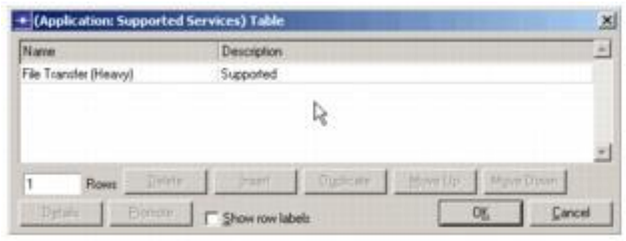
4. Rename the **ethernet_server** to **FTP**.
5. Connect the "router" and the "FTP" nodes to the switch with **10BaseT** links. Right-click to turn off link creation.
6. Close the object palette.

Configure the Server to Support the FTP Application

Follow these steps.

1. Open the Attributes dialog box for the **FTP** server.
2. Expand the Applications attribute group.
3. Choose **Edit...** for the **Application: Supported Services** attribute.
4. Change the number of rows to **1**.
5. Select **File Transfer (Heavy)** from the **Name** column pop-up menu.

Configure the Server to Support the Application



(Application: Supported Services) Table

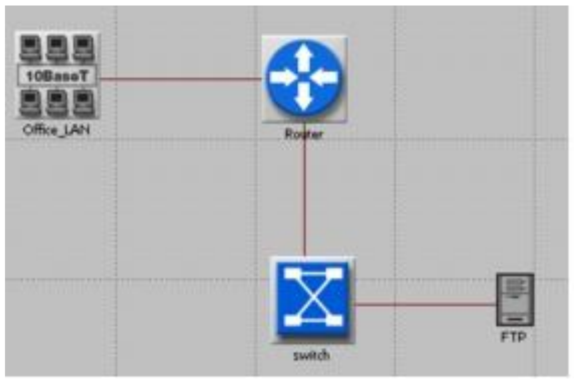
Name	Description
File Transfer (Heavy)	Supported

Rows: 1

6. Click **OK** to close the Supported Services dialog box, and then click **OK** to close the (FTP) Attributes dialog box.

   The Washington, D.C. subnet is now complete and should resemble the following figure.

Washington, D.C. Subnet



7. Use the workspace pop-up menu to return to the parent subnet view.
8. Save the project by selecting **File > Save**.

Background Load

Now that you have created a model to act as a baseline for the performance of the East Coast company's network, you can add background traffic to the links connecting the cities and compare the results from the two scenarios.

This topic focuses on

- Duplicating a scenario
- Implementing a varying background load on the links

Background loading is an efficient means of modeling a known traffic load on a link.

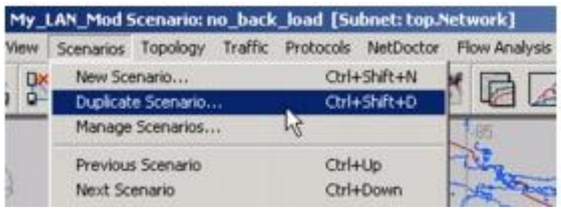
Network studies show that traffic rises gradually over the course of the day as employees arrive and begin work. You can use background link utilization to model this pattern.

Because you want to compare the performance of the network with and without background utilization, you need to prepare two scenarios, one for each situation.

You will duplicate the existing scenario, then add background traffic to it:

1. Select **Scenarios > Duplicate Scenario...**

   Scenarios Menu



2. Name the scenario **back_load**. and click **OK**.

Background Load is an attribute of each link. To set background load on the links between the cities:

1. Select the links between subnets. Right-click on the link between Washington, D.C. and Atlanta, then choose **Select Similar Links** from the pop-up menu.
2. Display the **Attributes** dialog box for the link between Washington, D.C. and Atlanta.
3. Click in the Value cell for the **Traffic Information** attribute and select **Edit...**

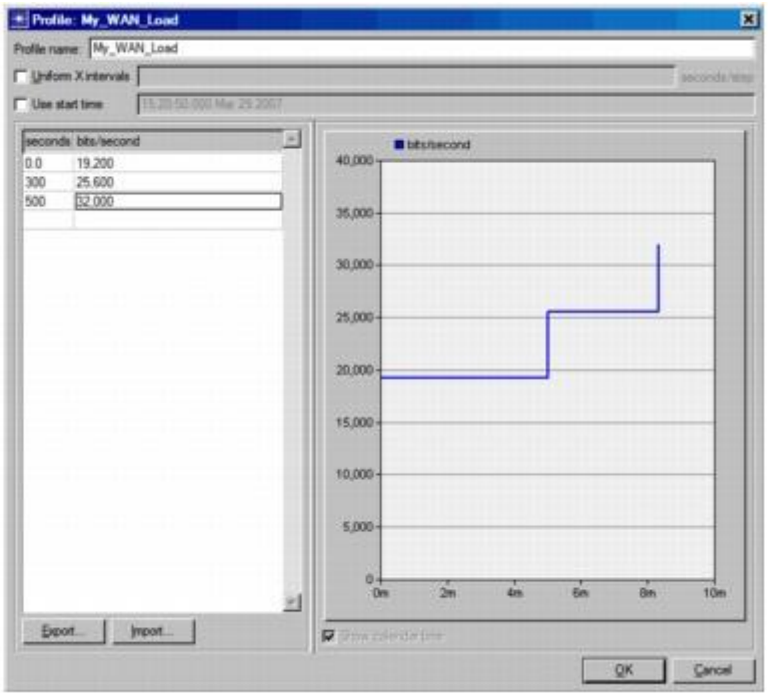
   A **Traffic Information** table appears.

4. Change the number of rows to 1.
5. Click in the [Atlanta.router -> Washington_DC.router] column and choose **Edit...**
6. Set Average Packet Size to Default.
7. Click on the value cell for "Traffic Load [bps]" and select **Edit...**

   The Traffic Intensity Attribute Profile dialog box appears.

8. Set the Profile name to "My_WAN_Load".
9. Complete the Profile table as shown in the following figure.

   Profile Table



Profile: My_WAN_Load

Profile name: My_WAN_Load

☐ Uniform X intervals

☐ Use start time 15:20:50.000 Mar 29 2007

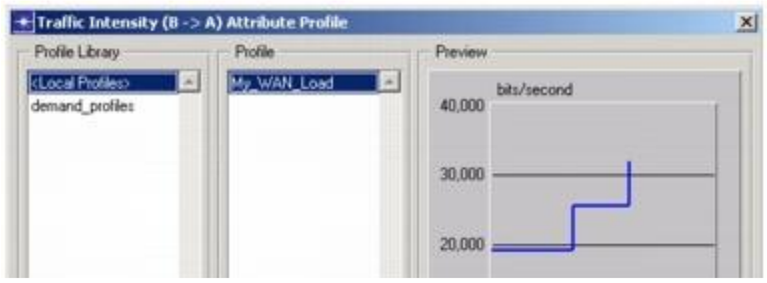
seconds	bits/second
0.0	19.200
300	25.600
500	32.000

   The intensity table above indicates that for the first 300 seconds of the simulation, the background load on this link will be 19,200 bps (that is, 30 percent of the DS0 capacity of the link). For the next 200 seconds, the background load on the link will be 25,600 bps (about 40 percent of the link capacity), and for the last part of the simulation, the load will be 32,000 bps (about 50 percent of the link capacity).

10. Click **OK** to close the profile table.
11. Click in the [Washington_DC.router -> Atlanta.router] and choose **Edit...**
12. Set Average Packet Size (bytes) to Default.
13. Click on the value cell of the Traffic Load (bps) and choose **Select...**

    The following table appears.

    Attribute Profile Table



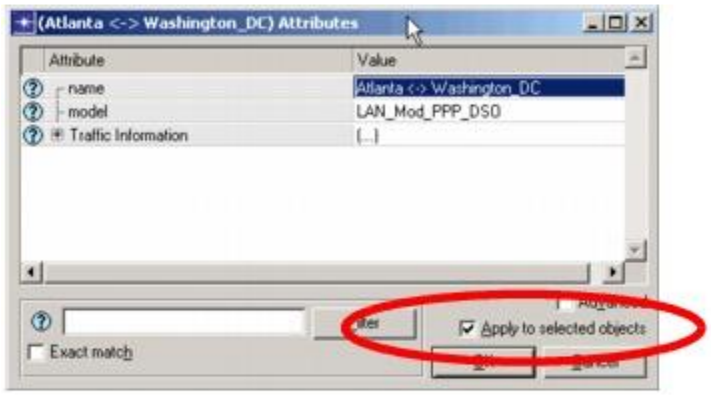
14. Select <Local Profiles> from the Profile Library list and "My_WAN_Load" from the Profile list.
15. Click **OK** to close the **Profile Selection** dialog box.
16. Click **OK** to close the **Traffic Information** table dialog box.

Do not close the link attributes dialog box.

The last step in setting Traffic Information is to apply the changes made to the Atlanta–Washington, D.C. link to all selected links.

1. Check the **Apply changes to selected objects** check box in the **Atlanta** <-> **Washington_DC** Attributes dialog box.

   Link Attributes Dialog Box

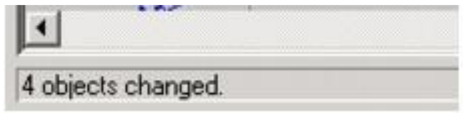

(Atlanta <-> Washington_DC) Attributes

Attribute	Value
name	Atlanta <-> Washington_DC
model	LAN_Mod_PPP_DS0
Traffic Information	(...)

2. Click **OK** to close the Attributes dialog box, then click **Yes** if you are prompted to continue.

   Note that "**4 objects changed.**" appears in the message area.

   Confirming Message


4 objects changed.

3. Save the project by selecting **File** > **Save**.

Collecting Statistics

Now that you have configured both scenarios (one without background load as a baseline, and one with background load), you are ready to collect data and analyze it.

The relevant statistics for this network are the throughput and utilization statistics for the links and the global FTP download time for the network.

This topic focuses on:

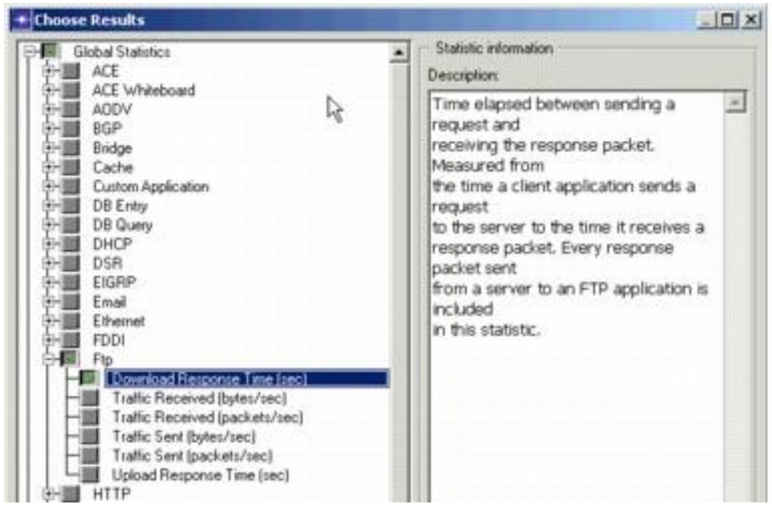
- Specifying statistics to collect in each scenario
- Switching between scenarios
- Running multiple simulations

To Collect Statistics in the back_load Scenario

Follow these steps.

1. Right-click in the workspace to display the workspace pop-up menu, and select **Choose Individual DES Statistics**.
2. Select the **Global Statistics > Ftp >** Download Response Time (sec) statistic.

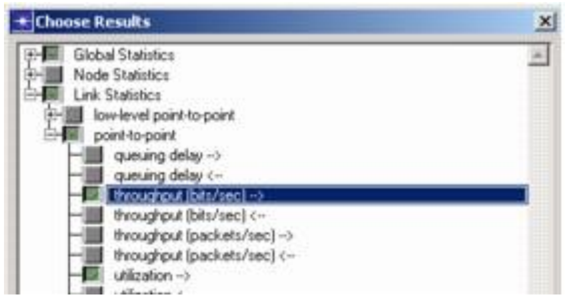
Selecting the FTP Download Response Time Statistic



Time elapsed between sending a request and receiving the response packet. Measured from the time a client application sends a request to the server to the time it receives a response packet. Every response packet sent from a server to an FTP application is included in this statistic.

3. Select the **Link Statistics >** point-to-point > **throughput (bits/sec) -->** and **utilization -->** statistic.

Selecting the Throughput and Utilization Statistics

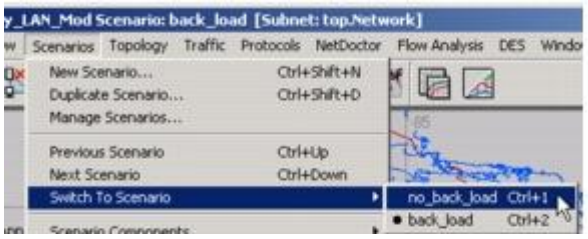


4. Click **OK** to close the **Choose Results** dialog box.

The throughput and utilization statistics that you have specified will be collected for every link in the simulation. In order to compare the statistics in the **back_load** scenario to the **no_back_load** scenario, the same statistics must be collected in the **no_back_load** scenario. To change scenarios and collect statistics:

1. Select **Scenarios** > **Switch To Scenario**, then choose **no_back_load**.

   Selecting the no_back_load Scenario


y_LAN_Mod Scenario: back_load [Subnet: top.Network]
Scenarios Topology Traffic Protocols NetDoctor Flow Analysis DES Windo
New Scenario... Ctrl+Shift+N
Duplicate Scenario... Ctrl+Shift+D
Manage Scenarios...
Previous Scenario Ctrl+Up
Next Scenario Ctrl+Down
Switch To Scenario
Scenario Components
no_back_load Ctrl+1
back_load Ctrl+2

2. Collect the same statistics that you did in the **back_load** scenario:
   - Global Statistics **>** Ftp **>** Download Response Time (sec)
   - Link Statistics **>** point-to-point **>** throughput (bits/sec)-->
   - Link Statistics **>** point-to-point **>** utilization -->
3. Close the **Choose Results** dialog box and save the project.

You are now almost ready to run the simulations to collect the statistics you have specified.

First, though, verify that your **Network Simulation Repositories** preference is set appropriately.

1. Choose **Edit > Preferences**.
2. Type **network sim** in the **Search for:** field and click the **Find** button.
3. If the **Value** field for the **Network Simulation Repositories** preference is not **stdmod**, click on that field.

   The **Network Simulation Repositories** dialog box opens.

4. Click the **Insert** button, then type **stdmod** in the field.
5. Click **OK** twice to close the **Network Simulation Repositories** and **Preferences** dialog boxes.

Now you are ready to run the simulations to collect the statistics you have specified.

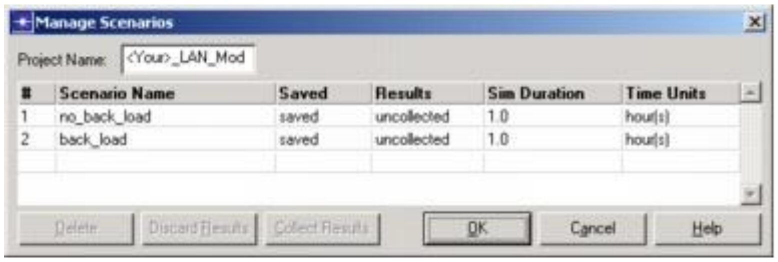
Using the **Manage Scenarios** dialog box, you can rename scenarios, change their order, and run single or multiple simulations.

Instead of running each simulation separately, you can batch them together to run consecutively. To run multiple simulations:

1. Select **Scenarios > Manage Scenarios...**

   The Manage Scenarios dialog box appears.

   Manage Scenarios Dialog Box

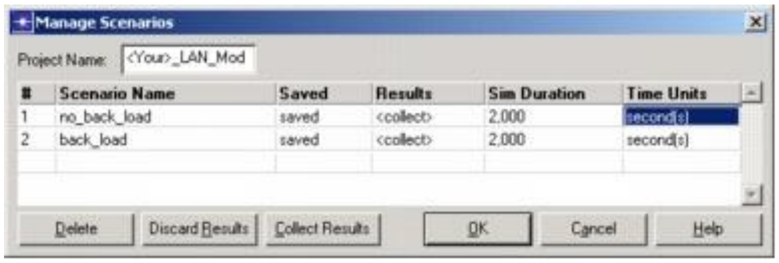


Project Name: <Your>_LAN_Mod

#	Scenario Name	Saved	Results	Sim Duration	Time Units
1	no_back_load	saved	uncollected	1.0	hour(s)
2	back_load	saved	uncollected	1.0	hour(s)

2. Click on the **Results** value for the **no_back_load** and **back_load** scenarios and change the value to **<collect>**.
3. Set the **Sim Duration** value for each scenario to **2,000** and the **Time Units** to **seconds**.

   Running the Simulation from the Manage Scenarios Dialog Box



Project Name: <Your>_LAN_Mod

#	Scenario Name	Saved	Results	Sim Duration	Time Units
1	no_back_load	saved	<collect>	2,000	second(s)
2	back_load	saved	<collect>	2,000	second(s)

4. Click **OK**.

   This runs simulations for both scenarios. A DES Execution Manager dialog box shows the simulation progress. Close the dialog box when the simulations are done.

If your simulation does not complete, if no results were collected, or if the results vary significantly from those shown, you will have to troubleshoot your simulation. See *"Troubleshooting Tutorials"*.

Comparing Results

You are now ready to examine the results of the two scenarios. Because you collected the same statistics in each scenario, you can use the Compare Results feature to look at them together.

To view the results from two or more different scenarios against each other, you can use the **Compare Results** feature. You can also apply different built-in filters to the graphs.

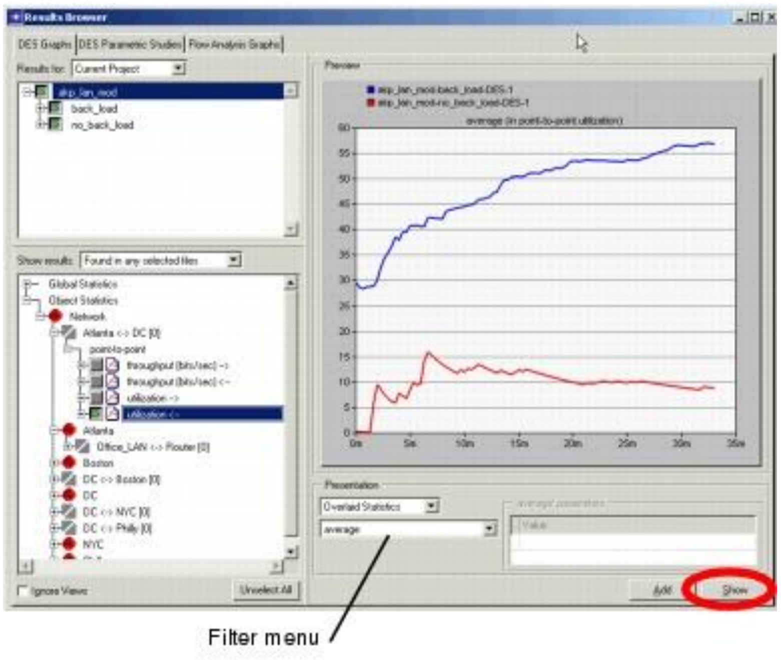
This topic focuses on:

- Comparing results between scenarios
- Applying filters to graphs

The **Compare Results** feature shows results from two or more scenarios. To look at link utilization:

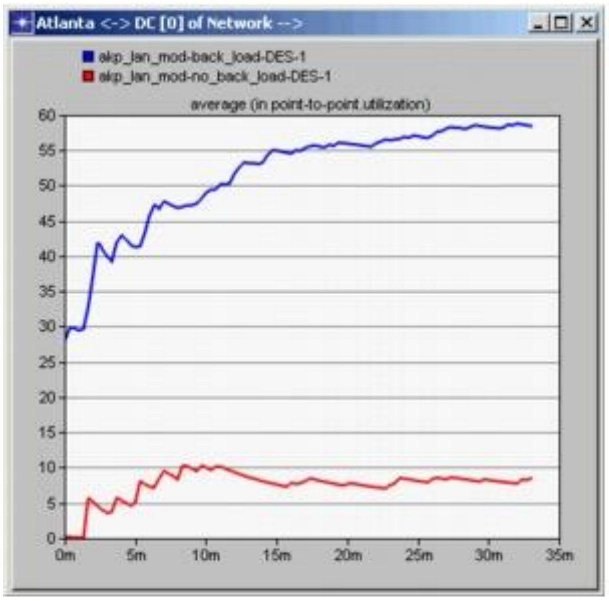
1. Right-click on the workspace to display the pop-up menu and choose **View Results**.
2. Under the DES Graphs tab, selects Results for **Current Project** and then select both scenarios.
3. Under Show Results, select
   Object Statistics > Network **>**
   Atlanta <-> Washington_DC [0] **>** point-to-point **>** utilization -->.
4. Because utilization varies over the course of a simulation, it is helpful to look at the time average for this statistic. Change the Filter menu from **As Is** to **Average**.
5. Select **Overlaid Statistics** from the pull down menu for Presentation.

   Setting the Filter



6. Click **Show** to display the graph. Your graph should resemble the following figure, though it will not match exactly:

Average Utilization Compared



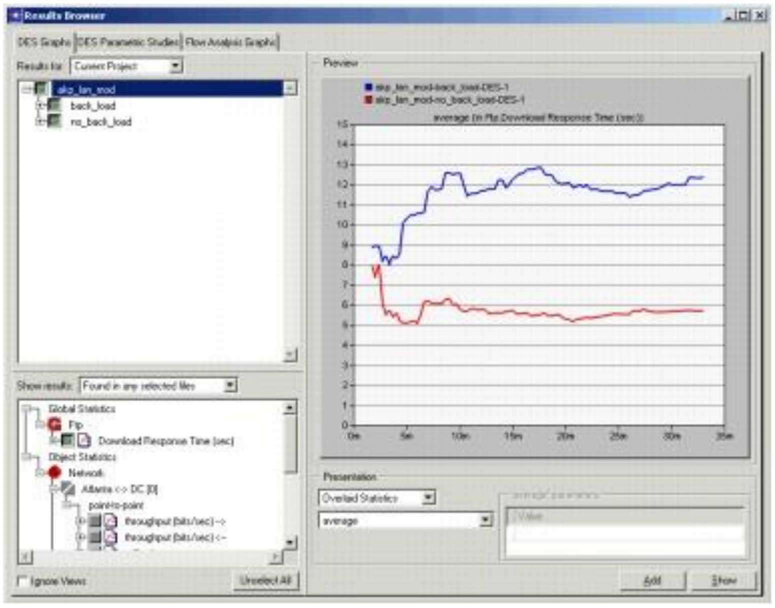
The network with the background load (back_load) shows higher utilization. You may want to look at the utilization of other links to determine the maximum utilization of any link.

You can also compare the link throughputs (bps) by selecting **Object Statistics >** Network **>**
Atlanta <-> Washington_DC [0] **>** point-to-point **>** throughput (bits/sec) -->.

Next, let's look at Global FTP response time:

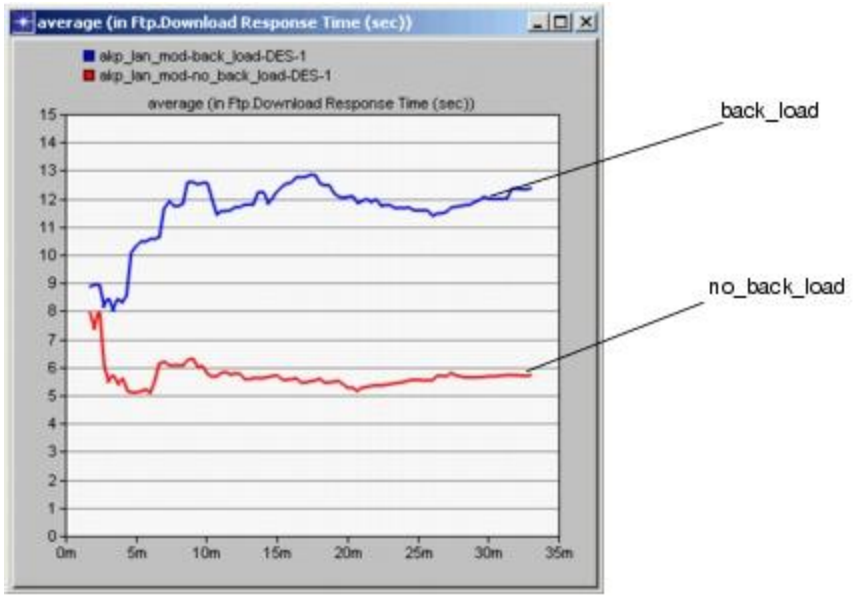
1. Move the link utilization graph aside and click the **Unselect** button in the Compare Results dialog box.
2. Check the **Global Statistics >** Ftp **>** Download Response Time (sec) statistic in the Compare Statistics dialog box.
3. Verify that the Filter menu shows **average**, then click **Show**.

Select Average from the Filter Menu



The graph should resemble the following figure:

Average Download Response Time Compared



This graph shows that, as steady state is reached, response time increases by several seconds when the link is heavily loaded.

1. Select **File > Close** and save changes before closing.

Now that you have completed the LAN lesson, you can move on to the Web Reporting lesson. This lesson presents the web reporting feature and how it can help you to make informed decisions about a network's behavior. Return to the main tutorial menu and choose **Web Reporting** from the list of available lessons.

Note—Be sure to delete the **stdmod** setting for the **Network Simulation Repositories** preference when you are finished doing tutorials. To delete the setting, select **Edit > Preferences**, search for **Network Simulation Repositories**, click on the value, and choose **Delete**.