# Práctica 3: SDH

**Nombre y Apellidos: Sara Barberá Boix**

## Objetivo de la práctica

El objetivo de esta práctica es familiarizarse con la Jerarquía Digital Síncrona (SDH, *Synchronous Digital Hierarchy*) y su implementación en una red de transmisión mediante simulaciones con OMNeT++ e INET. Se analizará el formato de tramas STM-n, su funcionamiento en enlaces punto a punto, y el uso de protección ante fallos. Además, se configurará y observará el tráfico de señales PDH dentro de tramas SDH.

## Fundamento teórico

Las redes SDH se basan en el transporte de información digital estructurada en tramas STM (*Synchronous Transport Module*), con una cadencia fija de 125 microsegundos. Estas tramas pueden transportar múltiples flujos de menor velocidad encapsulados como contenedores virtuales (VC). La infraestructura SDH permite implementar distintas topologías, siendo las más comunes las redes punto a punto y las redes en anillo.

En una red punto a punto, dos nodos SDH (como LTE o ADM) se comunican directamente mediante enlaces dedicados. Es una configuración simple, adecuada para conexiones directas entre equipos que generan y consumen tráfico.

Las redes en anillo SDH están formadas por varios nodos interconectados de manera circular. Esta topología ofrece redundancia natural, ya que el tráfico puede circular en ambas direcciones, permitiendo el reenvío en caso de fallo de un enlace o nodo. En este entorno, los ADMs insertan o extraen flujos concretos sin interrumpir el paso del resto del tráfico.

Las redes SDH implementan mecanismos de protección ante fallos para garantizar alta disponibilidad. En la protección dedicada (1+1), se utiliza un enlace principal y uno de respaldo (de igual capacidad), transmitiendo simultáneamente por ambos. El receptor selecciona el flujo válido, lo que permite una conmutación inmediata y transparente. Esta técnica ofrece alta fiabilidad, pero requiere el doble de recursos.

En la protección compartida, varios enlaces comparten un recurso de protección. En caso de fallo, el tráfico se redirige dinámicamente al enlace de respaldo si está libre. Aunque es más eficiente en términos de capacidad utilizada, la conmutación puede requerir más tiempo y no siempre está garantizada si el recurso ya está en uso.

## Instalación de módulo SDH

Para crear simulaciones con SDH emplearemos la biblioteca de modelos INET. A pesar del gran número de protocolos y tecnologías existentes en dicha biblioteca, INET no dispone de una implementación de SDH, así que emplearemos en esta práctica una implementación propia que podéis descargar desde el Campus Virtual de la asignatura en un archivo comprimido. Deberéis descomprimirlo dentro de la carpeta *src/inet/node* tal y como muestra la siguiente figura.

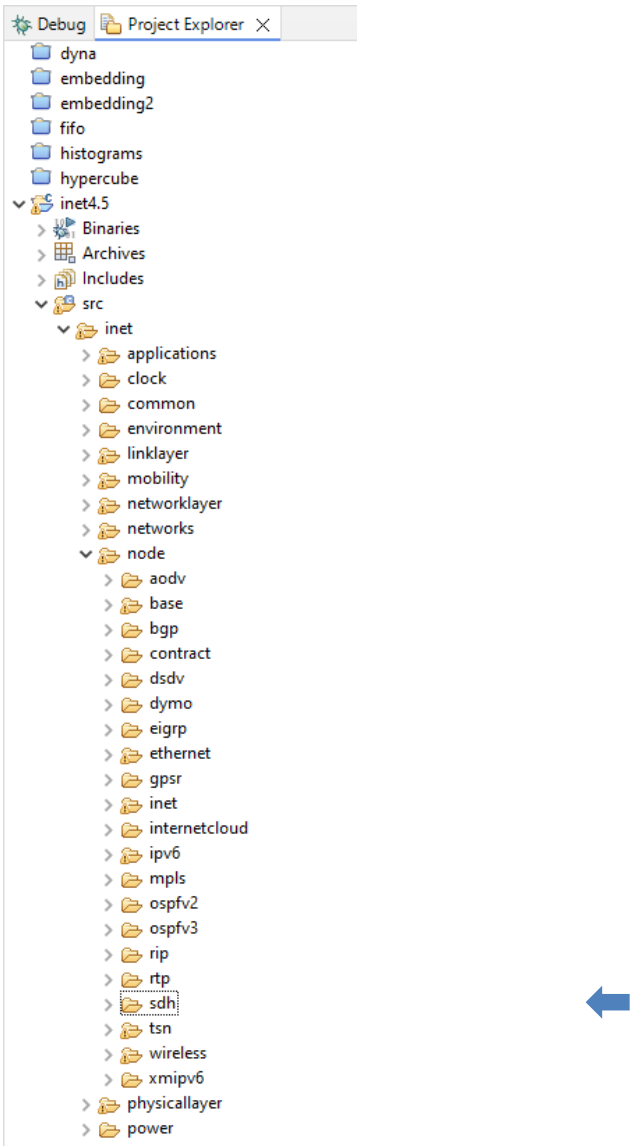

Figura 1. Espacio de trabajo con la nueva carpeta “sdh” en *inet4.5*.

Para asegurar el correcto funcionamiento, compile el proyecto INET y verifica que puedes lanzar alguna de las simulaciones que se proporcionan dentro de la carpeta *sdh*, como la *p2pNetwork*.

Para la realización de esta práctica y la entrega del código fuente y los resultados, crea un nuevo repositorio en GitHub, tal y como se hizo en las prácticas anteriores. Esta vez, el repositorio será la propia carpeta *sdh*.

## Modelo de SDH implementado

La práctica se apoya en una implementación funcional del modelo SDH dentro del simulador OMNeT++ e INET, utilizando módulos personalizados para representar los distintos tipos de nodos y dispositivos de una red de transporte síncrona.

Los nodos *PDHSource* y *PDHSink* representan dispositivos generadores y receptores de tráfico digital de baja velocidad (PDH). El *PDHSource* genera paquetes periódicamente con un tamaño

configurable, mientras que el *PDHSink* recibe y contabiliza esos paquetes. Estos nodos permiten evaluar la cantidad de tráfico que puede ser transportada por una red SDH y visualizar el comportamiento del sistema ante distintas condiciones de carga.

Los nodos SDHLTE (*Line Terminating Equipment*) se encargan de encapsular los paquetes recibidos desde los *PDHSource* en tramas STM, con una cadencia fija de 125 microsegundos. Estas tramas pueden incluir uno o varios contenedores virtuales (VC), dependiendo del nivel STM (STM-1, STM-4, etc.). El LTE también es responsable de extraer los paquetes del tráfico SDH que llegan desde la red y entregarlos a los *PDHSink*.

Los nodos SDHADM (*Add-Drop Multiplexers*) permiten insertar o extraer tráfico en puntos intermedios de la red, sin interrumpir el resto del flujo. En la práctica, su comportamiento depende de si están conectados directamente a fuentes o sumideros PDH, o si simplemente deben reenviar las tramas por el anillo. El ADM analiza cada trama STM que recibe, extrae únicamente los contenedores virtuales destinados a él (en función de las salidas conectadas), y reenvía el resto.

## Ejercicio 1:

El objetivo de este ejercicio es calcular la combinación óptima de tamaño e intervalo de paquetes para aprovechar al máximo una trama STM-1, STM-4 y STM-16 en una topología punto-a-punto sencilla con dos LTE, nodos fuente y destino. Para ello, contesta a las siguientes cuestiones empleando la red *p2pNetwork.ned*.

a) Determina la combinación óptima de tamaño de paquete (*packetSize*, en bytes) e intervalo de tiempo entre paquetes (*packetInterval*) para que se llenen completamente las tramas STM-1, STM-4 y STM-16.

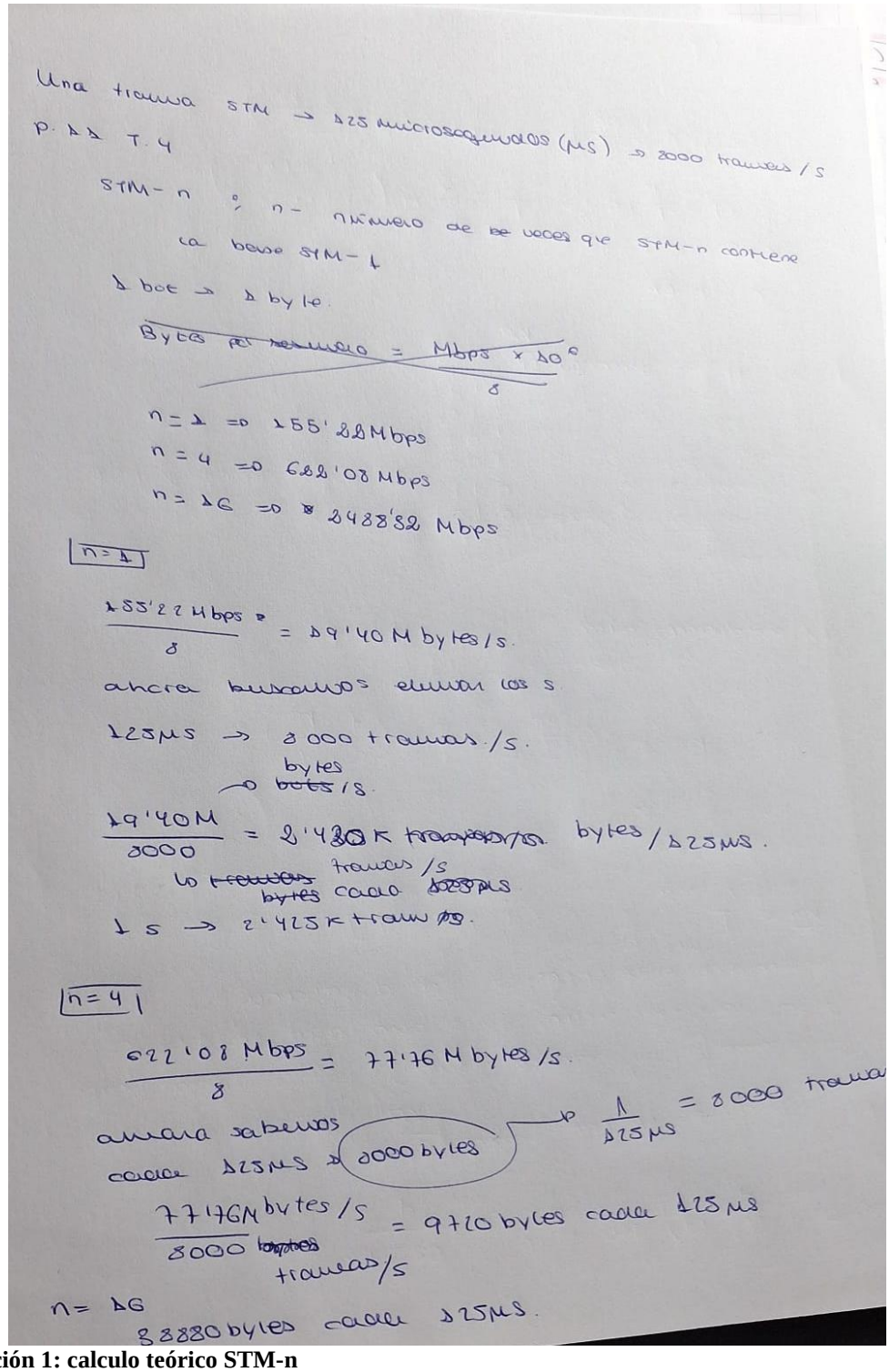


**Ilustración 1: calculo teórico STM-n**

b) Implementa en el archivo *omnetpp.ini* tres configuraciones diferentes llamadas STM1, STM4 y STM16 en las que los valores de *paquetSize* y *packetInterval* de las fuentes sean los calculados en el apartado anterior.

```
[STM1]
*.lteA.stmLevel = 1 #nivel que queremos stm-n
*.lteB.stmLevel = 1
*.lteA.numTributaries = 1
*.lteB.numTributaries = 1

# Traffic generation
*.sourceA.packetInterval = 125us
*.sourceA.packetSize =  2425# in bytes


*.sourceB.packetInterval = 125us
*.sourceB.packetSize =  2425# in bytes

[STM4]
*.lteA.stmLevel = 4 #nivel que queremos stm-n
*.lteB.stmLevel = 4
*.lteA.numTributaries = 1
*.lteB.numTributaries = 1

# Traffic generation
*.sourceA.packetInterval = 125us
*.sourceA.packetSize =  9720# in bytes


*.sourceB.packetInterval = 125us
*.sourceB.packetSize =  9720# in bytes

[STM16]
*.lteA.stmLevel = 16 #nivel que queremos stm-n
*.lteB.stmLevel = 16
*.lteA.numTributaries = 1
*.lteB.numTributaries = 1

# Traffic generation
*.sourceA.packetInterval = 125us
*.sourceA.packetSize =  38880# in bytes


*.sourceB.packetInterval = 125us
*.sourceB.packetSize =  38880# in bytes
```

**Ilustración 2: Código STM-n**

El *PacketInterval* lo hemos puesto a 125 micro segundos eso quiere decir que se manda un paquete cada ese intervalo de tiempo.

*PacketSize* es la unidad de datos que se envía un paquete su longitud . Esta especifica cuantos bytes tiene cada uno de nuestros paquetes.

c) Implementa un estadístico para representar gráficamente la carga utilizada en cada trama STM a lo largo del tiempo. Para ello añade el vector *loadSTM* y sus estadísticos en *SDHLTE.ned* y emítelo en el código. Dibuja las gráficas resultantes con OMNeT++ y verifica que la carga coincide con los valores teóricos esperados según los cálculos de los apartados anteriores.

Pd: en las gráficas vemos los bytes absolutos no su ratio (lo he puesto así para poder ver exactamente su carga)

STM-1

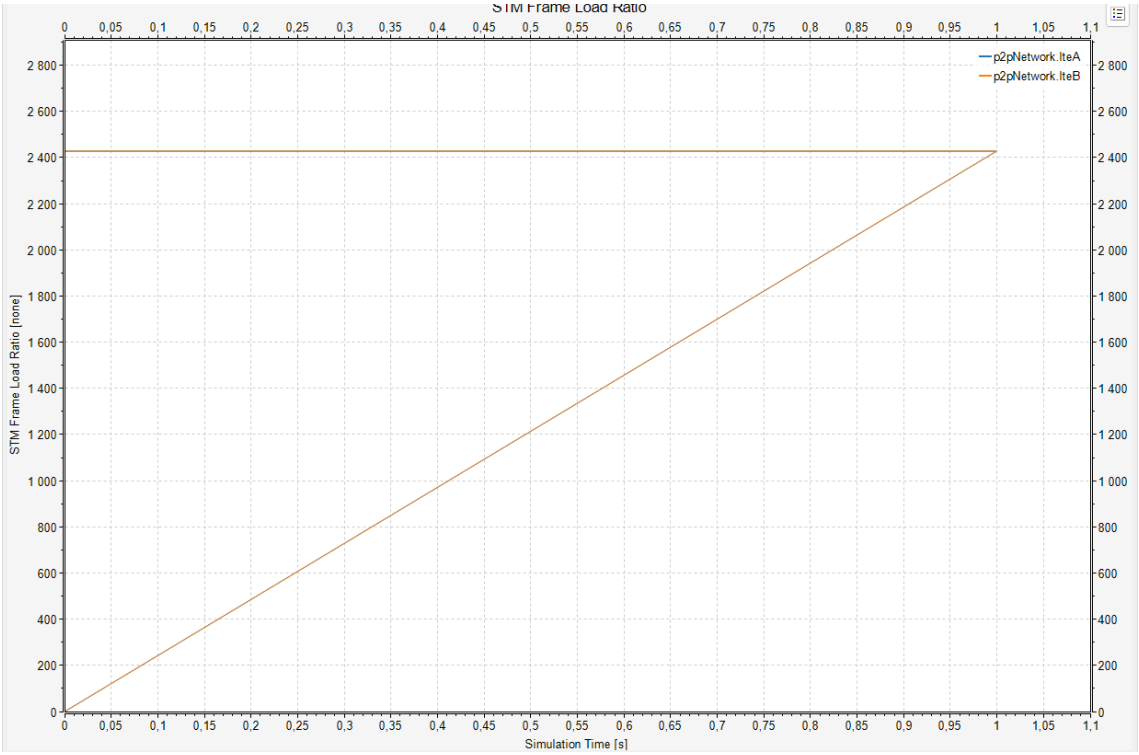


**Ilustración 3: carga STM-1**

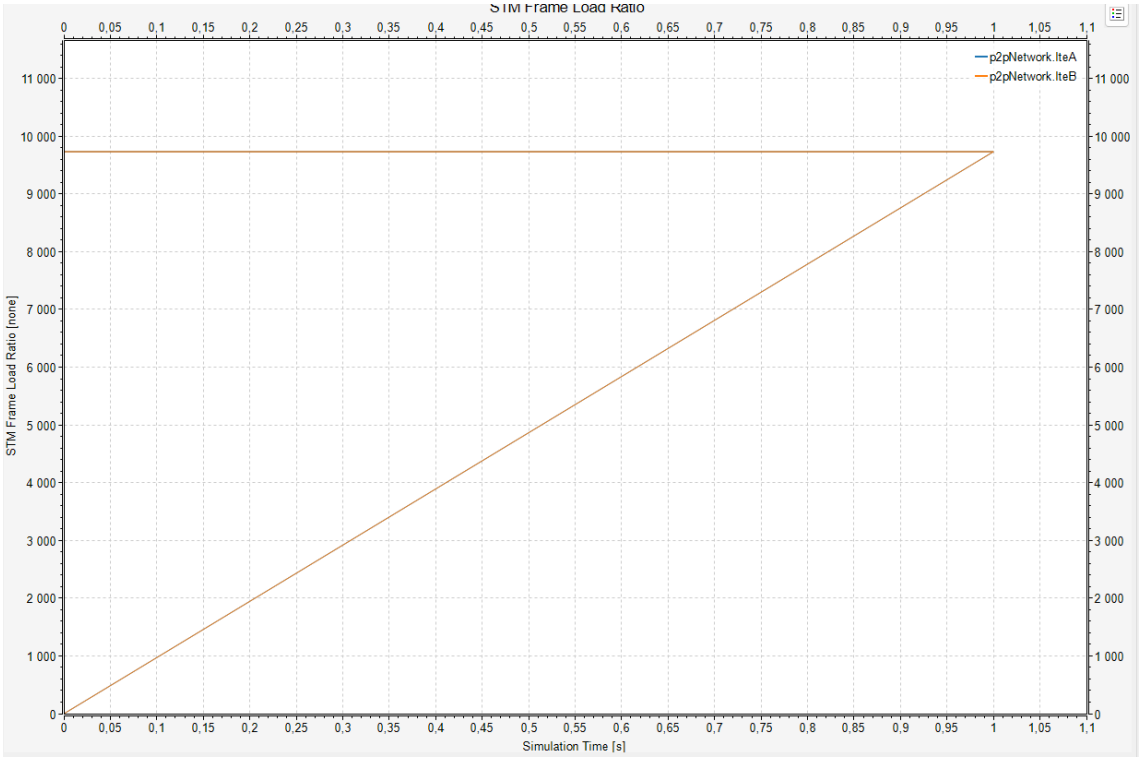


**Ilustración 4: Carga STM-4**

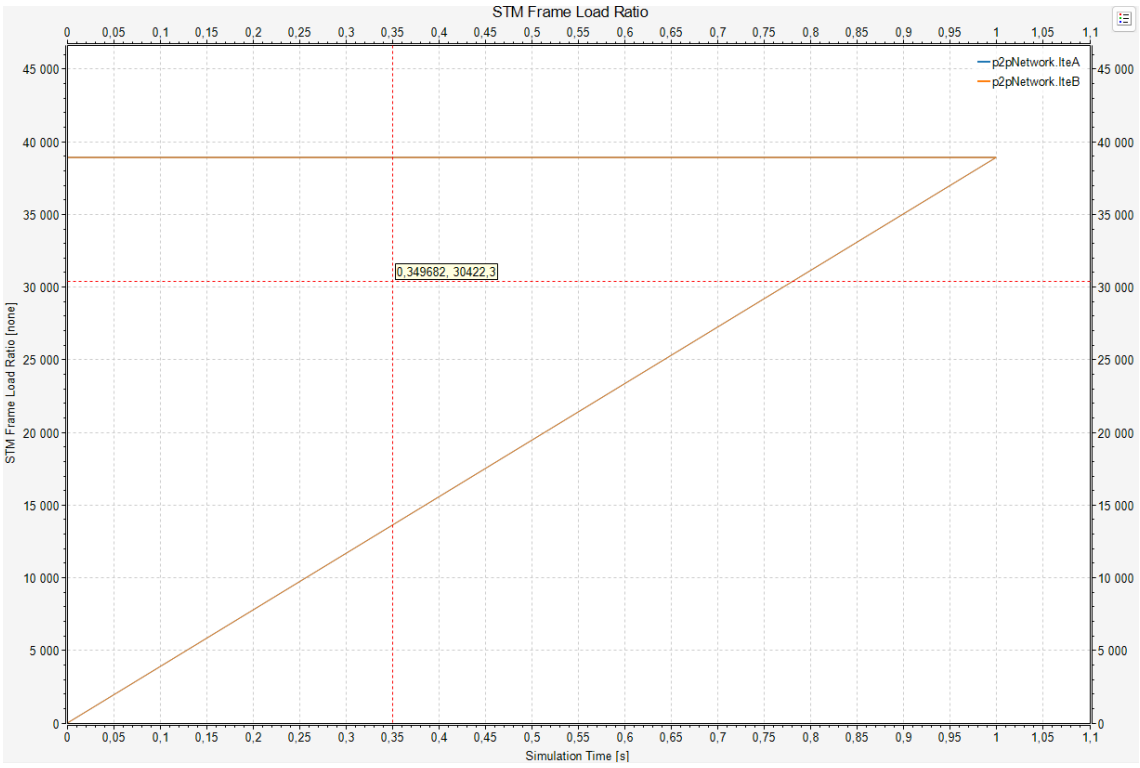


**Ilustración 5: Carga STM-16**

Lo que estamos viendo en nuestra simulación es una simulación de transporte IP sobre SDH

En la STM-4, estas utilizando un ancho de banda de 9720 bytes por cada trama SDH (2430x4). En nuestra simulación cada fuente (A y B ) generan paquetes de exactamente 9720 bytes cada 125 micro segundo (tiempo exacto de transmisión de una trama ) , que es el tamaño máximo que puede transportar STM-4 en un intervalo , cada paquete generado llena completamente una trama STM.
En la Ilustración 4: Carga STM-4vemos una línea constante que llega hasta 9720 bytes en 1 segundo, lo que indica que se está generando una trama completamente cargada cada 125 micro segundos. Esto refleja que la carga de tráfico es completamente constante y llena su ancho de banda

En la Ilustración 3: carga STM-1y Ilustración 5: Carga STM-16, se observa el mismo comportamiento que la anterior, donde la carga crece linealmente hasta alcanzar su valor máximo. Para STM-1, cada paquete de 2425 bytes generado cada 125 microsegundos llenan completamente la trama SDH. Sin embargo, STM-16 se usan paquetes de 38880 bytes, donde se ajusta al tamaño de la trama STM-16 . En ambos casos, la gráfica muestra un crecimiento constante que refleja una transmisión continua y eficiente, ya que en cada intervalo de tiempo se llena exactamente una trama, sin dejar capacidad libre ni satura la red.

d) Crea una nueva subcarpeta llamada *p2pNetworkExtended*. Crea una nueva red denominada *p2pNetworkExtended* a partir de la red *p2pNetwork* y guárdala en la subcarpeta que has creado. Crea un archivo *omnetpp.ini* a partir del utilizado para la red p2pNetwork y guárdalo también en la misma subcarpeta. Ahora extiende la red para que haya 2 *PDHSource* y 2 *PDHSink* conectados a cada LTE empleando STM-16. Establece tamaño e intervalo iguales para todas las fuentes. Dibuja la carga y el número de tramas STM que se entregan en los nodos destino.

STM-16

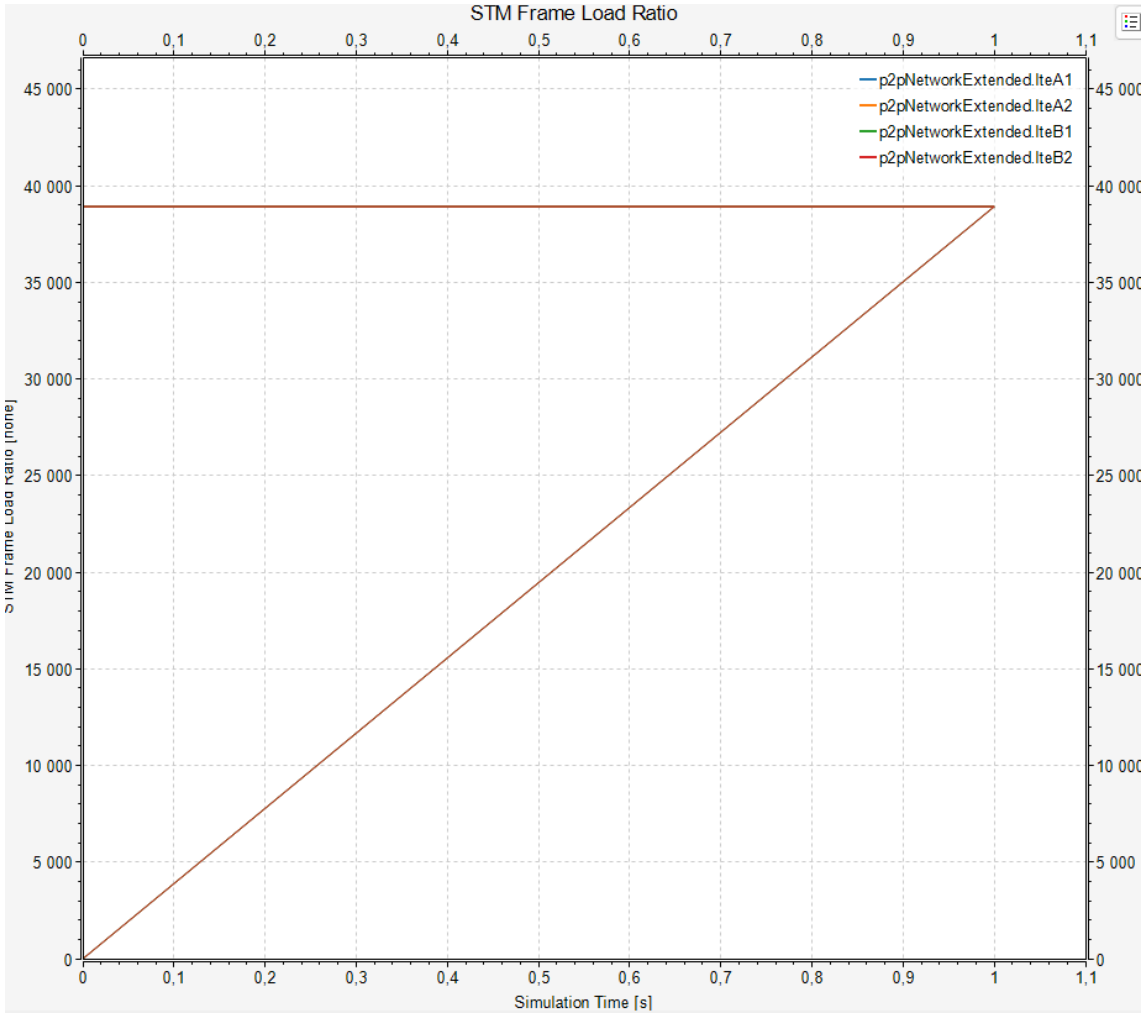


**Ilustración 6: STM-16 con dos Sources y Sink**

Cada lte se encuentra enviando o recibiendo trafico.

- lteA1 y lteA2 son los módulos del lado A que agrupan el tráfico que viene de la fuente y lo envían al lado B.
- lteB1 y lteB2, son los módulos del lado B que hacen el proceso inverso es decir que reciben del lado A y también generan trafico al lado A .

En esta Ilustración 6: STM-16 con dos Sources y Sink, de la carga STM-16 , estamos observando cómo se utilizan la capacidad de transmisión de los multiplexores SDHLTE. Vemos la curva de los módulos lteA1, lteA2, lteB1 y lteB2(se encuentran superpuestas). La recta constante indica la acumulación de tramas procesadas en nuestra red.

El parámetro *numTributaries* = 2 en cada modulo especifica la cantidad que multiplexa está combinación de intercambio de flujo. En nuestro caso, cada tributario transmite 38880 bytes cada 125 microsegundos , por lo que en cada tributario ocupamos todo el ancho de banda disponible.

## Ejercicio 2:

El objetivo de este ejercicio es estudiar un esquema de protección en un enlaces SDH punto a punto (*p2pNetwork.ned)*, observando el comportamiento tras la caída de la línea operativa. Para ello, contesta a las siguientes cuestiones:

a) Describe el tipo de protección que se ha implementado en el código y cómo se ha implementado, incluyendo los parámetros que le influyen.

Vemos un tipo de protección bidireccional en el que la red SDH en los nodos lteA y lteB envían continuamente datos por el enlace operativo y el de redundancia (protección ). En nuestro caso lo que pasa es que nuestros nodos enviaran / recibirán tramas en el enlace de trabajo y en caso de fallo en el de protección.
Vemos en el código el *useProtection*= true que sirve para definir la protección , y también observamos *protectionSwitchTime* , que define cuánto tiempo espera el nodo antes de redirigir el tráfico tras caer nuestro enlace de trabajo

b) Simula la configuración [*Protection*] de la red *p2pNetwork* disponible en el *omnetpp.ini*. Visualiza la animación y describe lo que observas.

```
INFO: PDHSink received packet:
** Event #39987  t=0.5  p2pNetwork.sourceA (PDHSource, id=2)  on selfmsg sendTimer (omnetpp::cMessage, id=359482)
** Event #39988  t=0.5  p2pNetwork.lteA (SDHLTE, id=4)  on selfmsg frameTimer (omnetpp::cMessage, id=359483)
** Event #39989  t=0.5  p2pNetwork.lteB (SDHLTE, id=5)  on selfmsg frameTimer (omnetpp::cMessage, id=359484)
** Event #39990  t=0.5  p2pNetwork.sourceB (PDHSource, id=6)  on selfmsg sendTimer (omnetpp::cMessage, id=359485)
** Event #39991  t=0.5  p2pNetwork.lteA (SDHLTE, id=4)  on PDH-3999 (omnetpp::cPacket, id=463436)
** Event #39992  t=0.5  p2pNetwork.lteB (SDHLTE, id=5)  on PDH-3999 (omnetpp::cPacket, id=463451)
** Event #39993  t=0.500002  p2pNetwork.lteB (SDHLTE, id=5)  on STM used 38880/38880 (SDHFrame, id=463437)
```

**Ilustración 7: Simulación ejecutada "Protection"**

En la simulación, hemos activado la protección en la red SDH con el uso de enlace redundante por si uno nos falla , lo hemos configurado en un tiempo de conmutación de 0,5 segundos . Sin embargo, no se ha producido en nuestra simulación ningún error, por lo que el tráfico sigue fluyendo por su camino correspondiente.

En el caso de que nuestra red fallara, la protección se activaría y la línea que estaba funcionando (la operativa), dejara de funcionar depues del tiempo que hemos definido en nuestro código (*protectionSwitchTime*), ya que se empezaran a enviar las tramas por el camino de redundancia .Esto lo que hace es asegurar la continuidad de nuestra red.

c) Muestra mediante gráficas de resultados las STM transmitidas y recibidas por línea operativa y de protección, así como la carga STM por tipo de enlace. ¿Se han producido los resultados que esperabas teniendo en cuenta la implementación del método de protección?

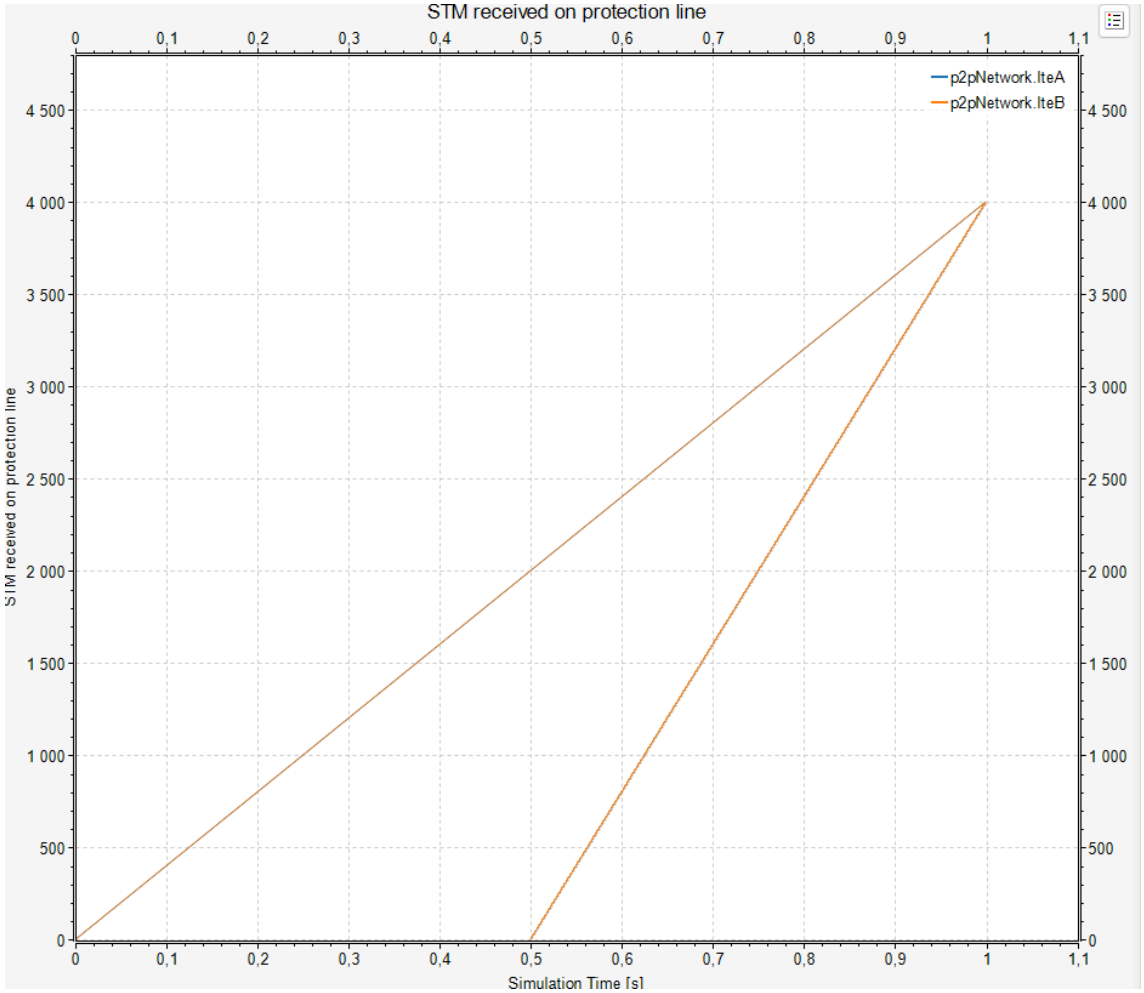


**Ilustración 8:stmProtectionReceived**

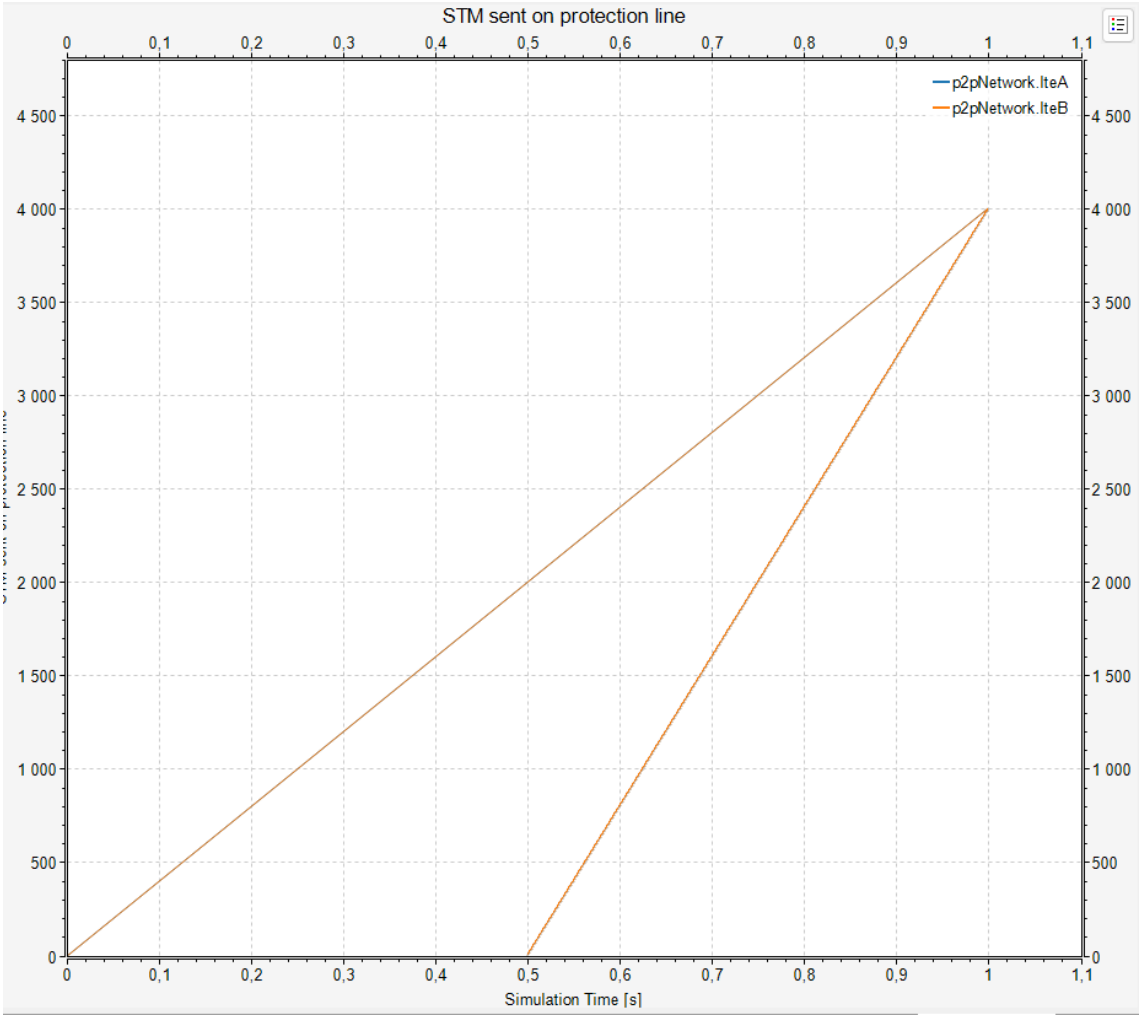


**Ilustración 9:stmsendProtection**

En la gráfica observamos dos rectas que representan el tráfico transmitido y recibido a través de la línea de protección. La segunda recta, que comienza en el punto (0.5, 4000) y continúa en línea recta, es coherente con el funcionamiento esperado del sistema, ya que la conmutación a la línea de protección se produce en t = 0.5 segundos. A partir de ese instante, la red continúa transmitiendo datos a través de la ruta alternativa, manteniendo la misma velocidad y calidad de transmisión. El hecho de que tanto el tráfico enviado como el recibido sean idénticos confirma que la protección está funcionando correctamente, garantizando la entrega de datos sin pérdidas tras el cambio de ruta.

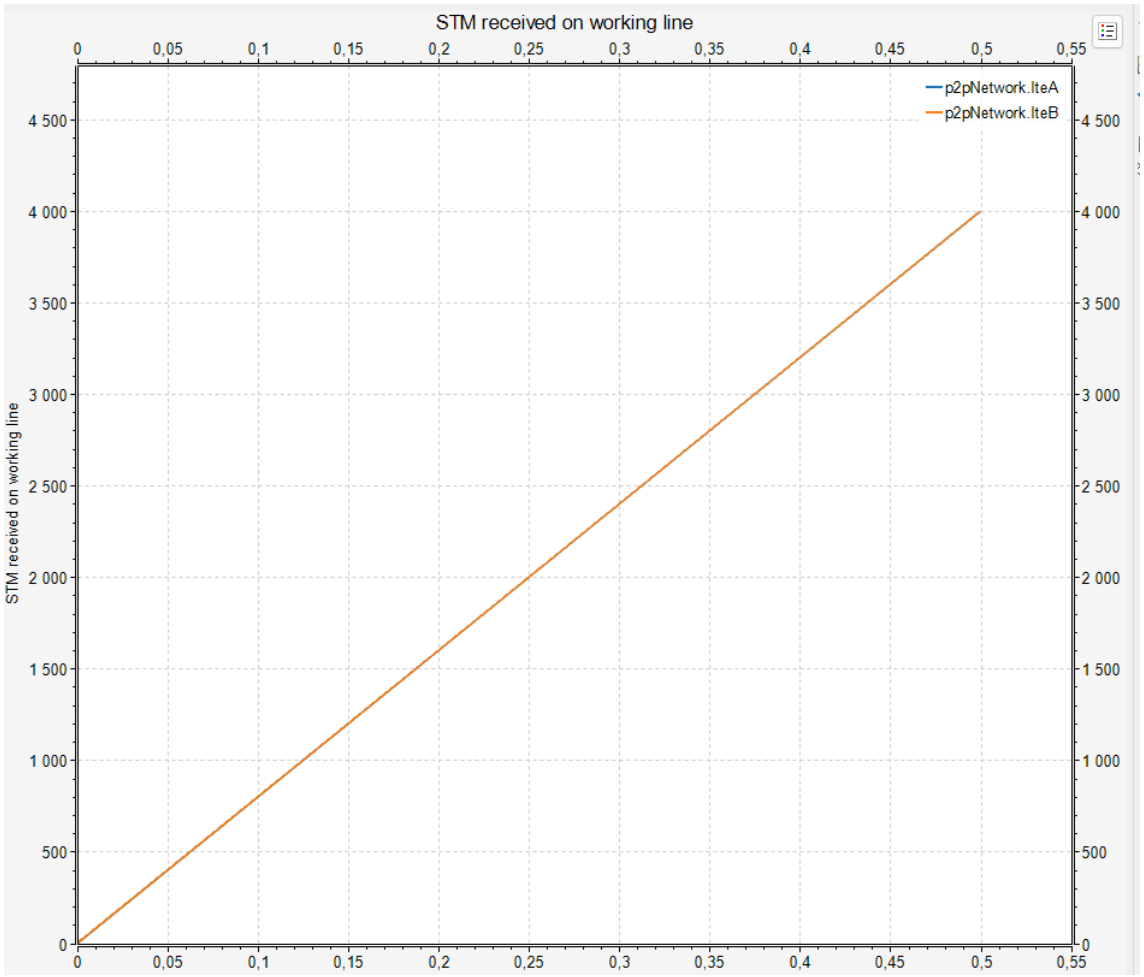


**Ilustración 10: stmWorkingRecived**

Aquí lo que vemos es una caída cuando falla la línea.

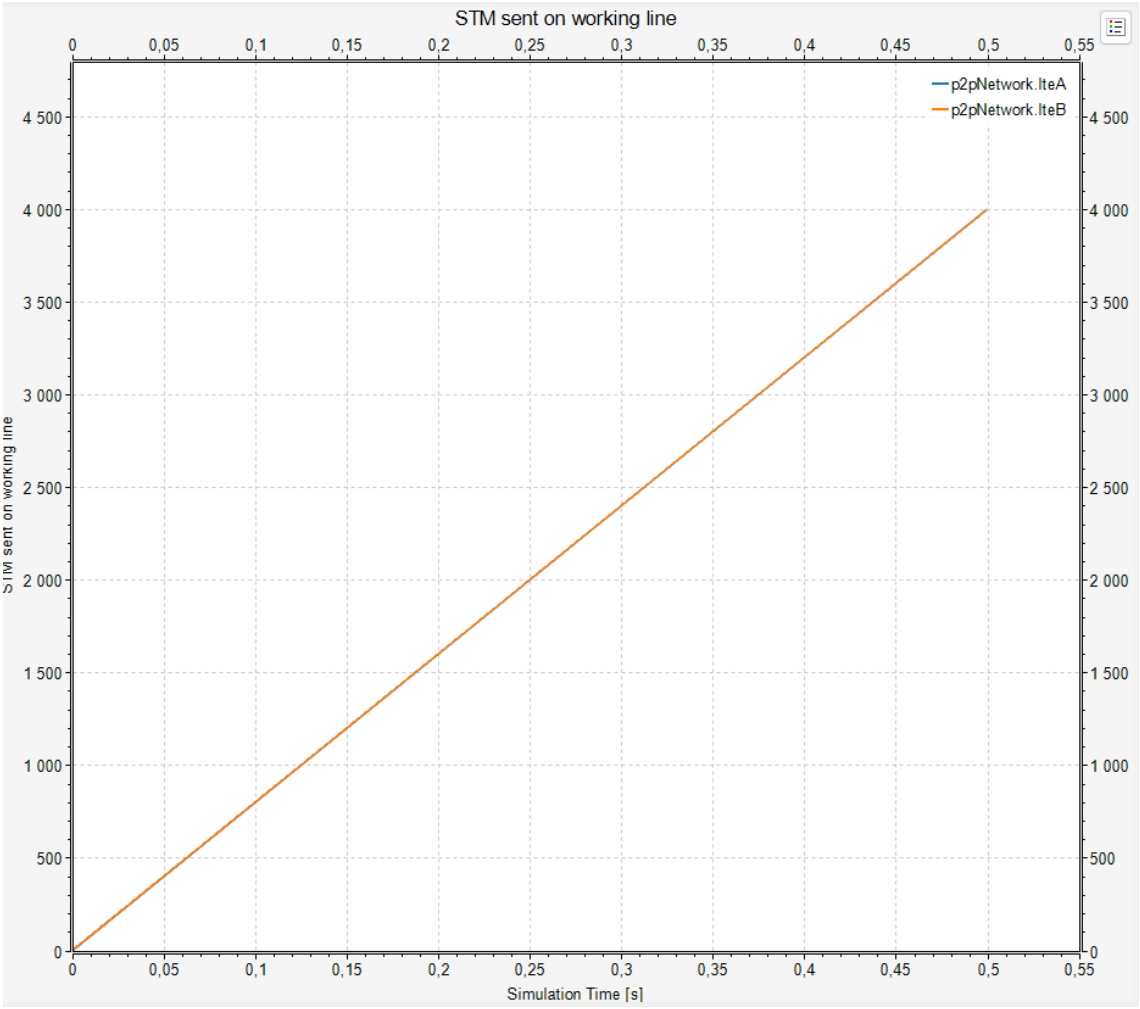


**Ilustración 11: stmSendWorking**

En la gráfica de *stmSendWorking* y *stmReceivedWorking* estamos viendo que dura 0,5 segundos de la simulación, lo que podemos creer que esta pasando es que se esta mostrando el comportamiento normal de la red ya que no hay fallos. Ambas graficas de enviadas como recibidas son iguales ya que el tráfico fluye correctamente a través del enlace en el que se trabaja eso quiere decir que no hay perdidas ni retardos.

## Ejercicio 3:

El objetivo de este ejercicio es analizar una red SDH con topología en anillo y para ello emplearemos las redes *simpleRingNetwork* y *advancedRingNetworks* proporcionadas. Conteste a las siguientes cuestiones:

a) Las redes en anillo emplean el nodo ADM para la inserción y extracción de información. Cree las señales y estadísticos necesarios para almacenar resultados sobre las tramas STM reenviadas por el anillo por cada ADM, así como los paquetes recibidos/transmitidos de/hacia los afluentes PDH.

Tenemos que registrar las señales estadísticas en el modulo ADM para poder visualizar los paquetes PDH que se tranmiten/reciben y lo reenviado.

Para poder medir esto hay que añadir parámetros de señal y estadísticos. En concreto, he definido tres señales : pdhReceived para contar los paquetes enviados, y stmForwarded para contar las tramas STM que el ADM reenvía por el anillo. Estas señales se emiten mediante el comando emit().

También he modificado el archivo .ned del módulo ya que he definido los parámetros. Los estadísticos son los que me permiten registrar los valores en vectores de una manera dinámica y después encontramos las señales.

b) Simula la red *simpleRingNetwork* configurando el tamaño y el intervalo de paquetes para simular una STM-4. Explica el comportamiento de los diferentes nodos. Analiza los resultados y explica si tienen sentido.

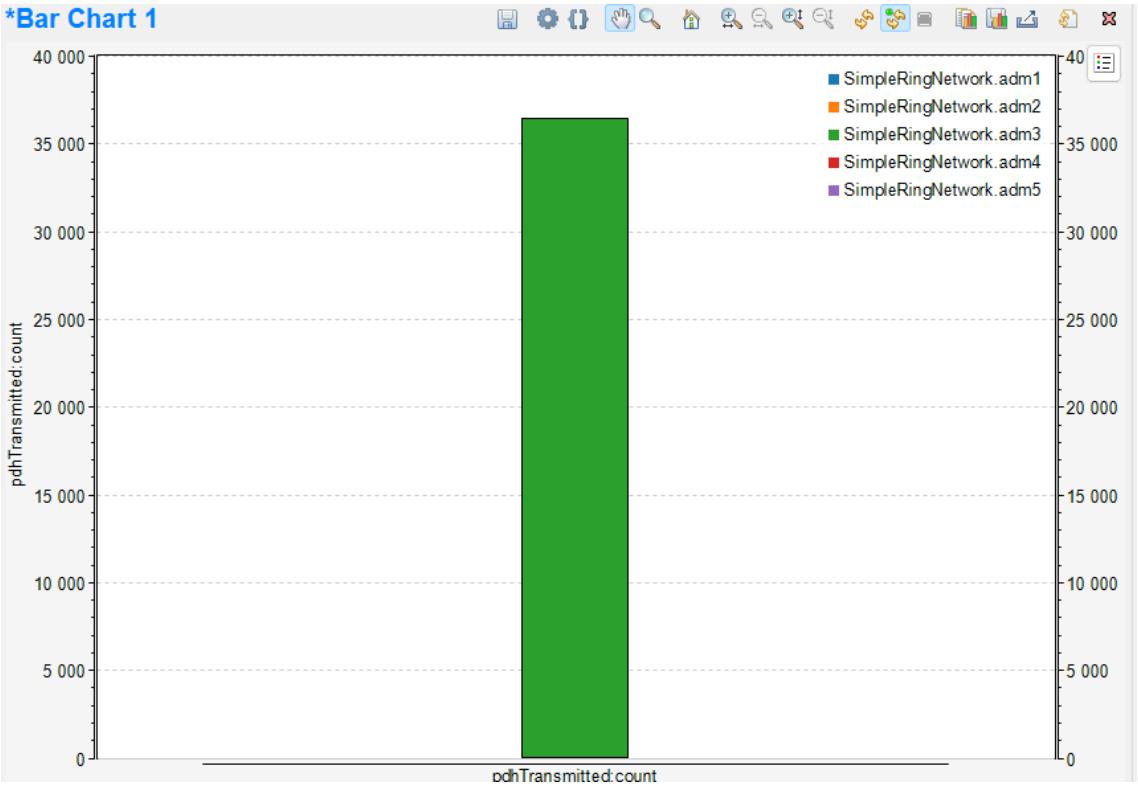


**Ilustración 12:Anillo simple pdhtransimitted**

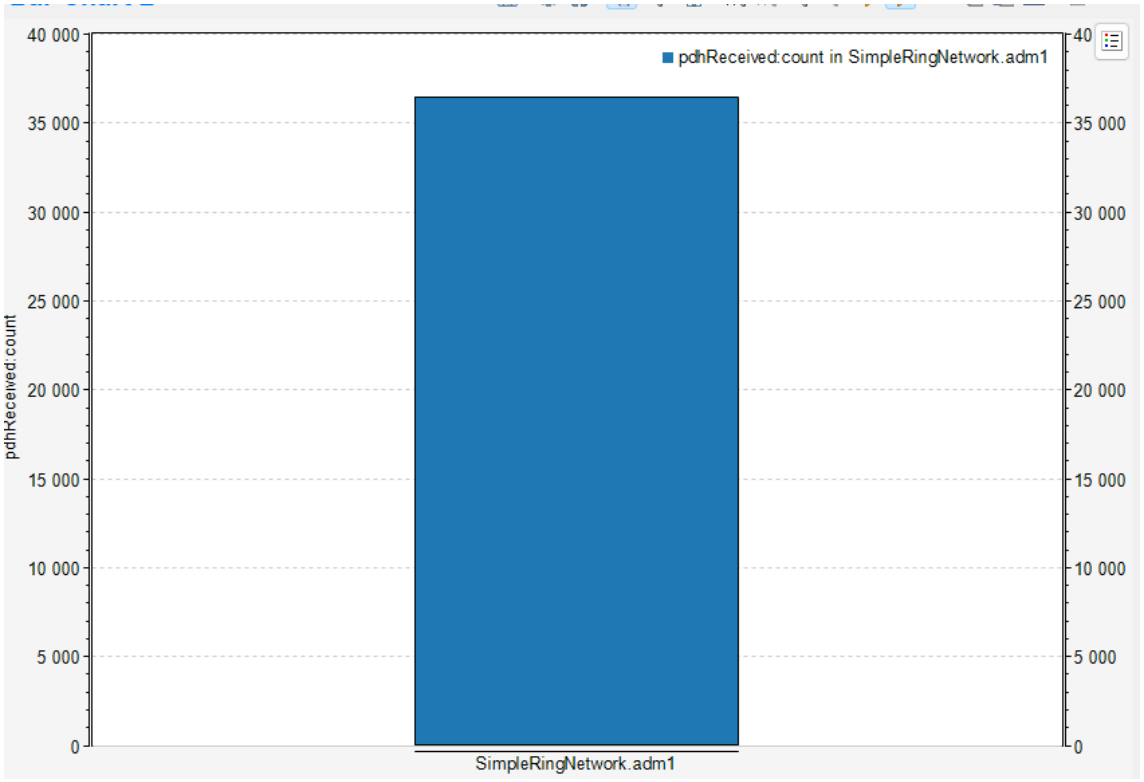


**Ilustración 13: Anillo simple PDH received**

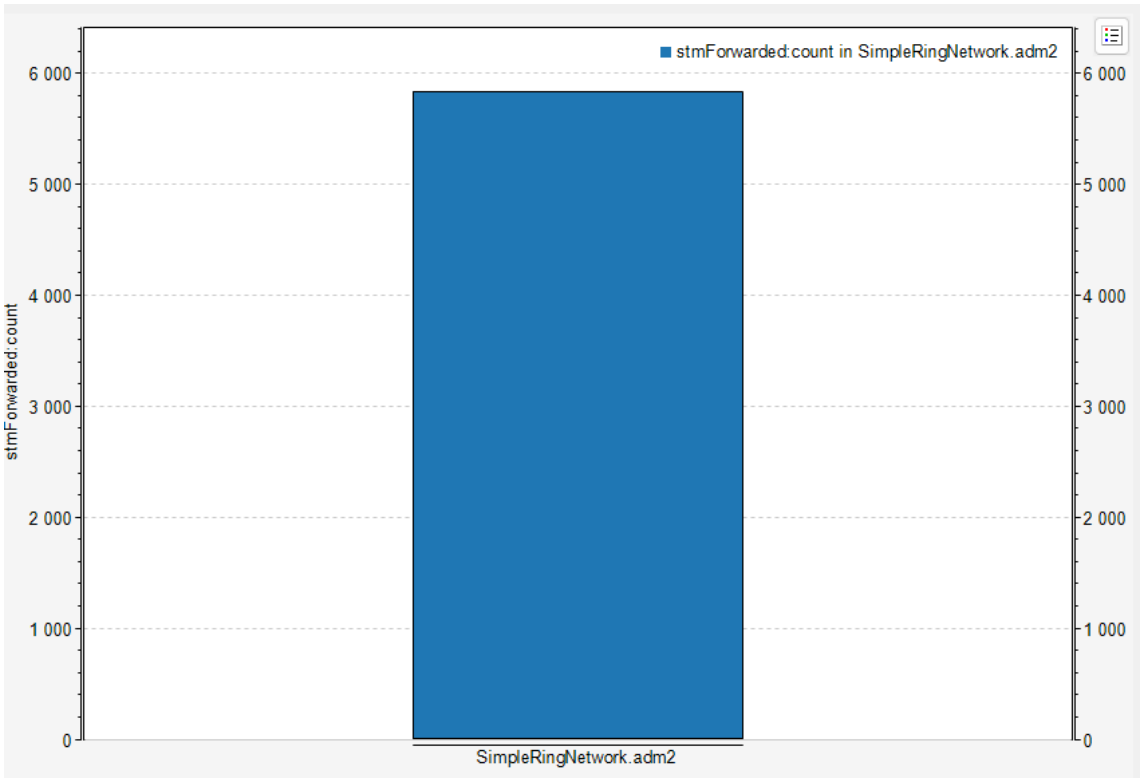


**Ilustración 14: Anillo simple STM forwarded**

Lo que vemos, es que solo en los nodos adm3 muestra un total de 36000 paquetes transmitidos , lo que indica que es el único nodo transmite. Después en PDH recived, vemos que adm1 ha recibido también unos 36000 paquetes lo que podemos decir que en este nodo se inyectan los datos PDH en la red.

Por otro lado, en la gráfica de Ilustración 14: Anillo simple STM forwarded, el nodo adm2 es el único que muestra tráfico.
En resumen, el flujo de datos sigue un único camino: inserción en ADM1-> reenvío por ADM2 -> extracción por ADM3

c) Simula la red *advancedRingNetwork* configurando el tamaño y el intervalo de paquetes para simular una STM-16. Explica el comportamiento de los diferentes nodos. Analiza los resultados y explica si tienen sentido.
Pd: lo he hecho con STM-1 y num tributarios =2

pdhTransmitted

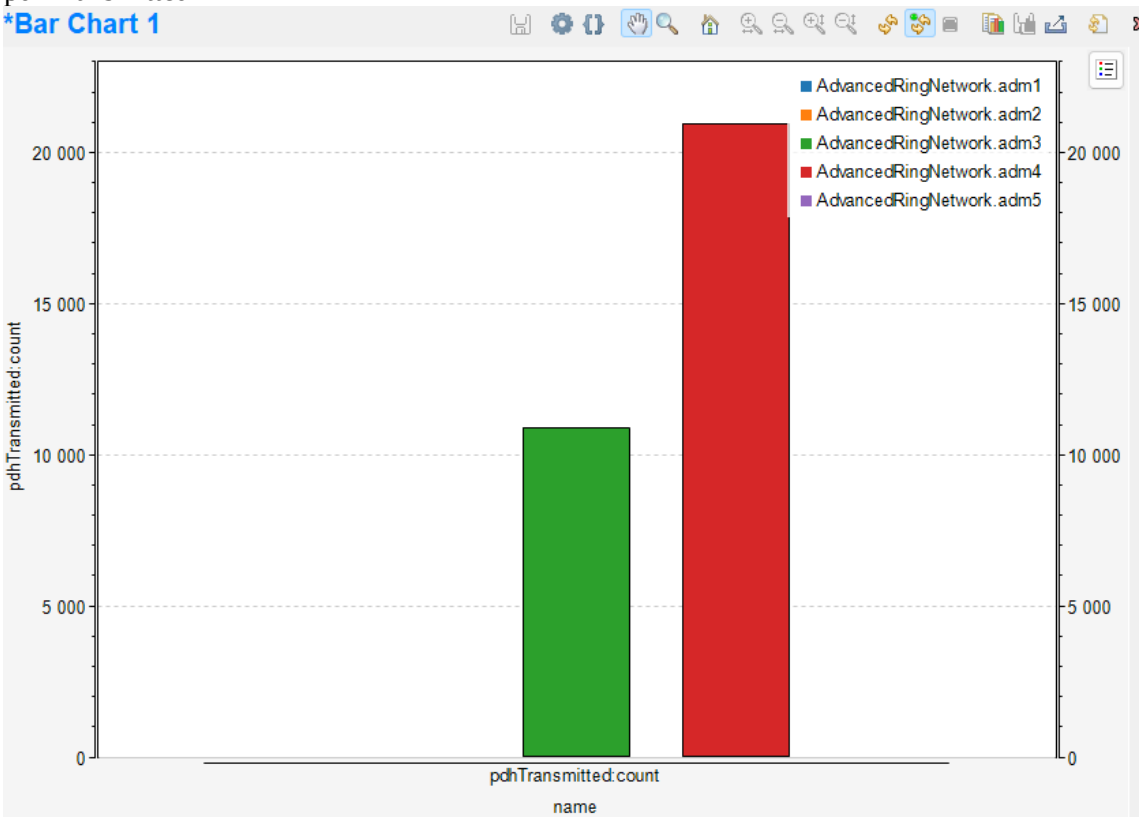


**Ilustración 15: anillo Avanzado pdhtransmitted**

Representa el número acumulado de paquetes PDH extraídos desde la trama SDH y reenviados por *pdhOut[]* .Cada vez que un SDH llega a un nodo y contiene información para un tributario concreto, se extraen los paquetes PDH encapsulados llevan a su VC(Canal Virtual) y se reenvían. Esta gráfica lo que podemos ver es cuantos paquetes han sido entregados a los destinos correspondientes.

La gráfica muestra la cantidad de paquetes pdh transmitidos por cada nodo ADM en el anillo SDH. En este caso la barra de adm4 (la barra roja), es el nodo que más tráfico ha transmitido, con más de 20000 paquetes , seguido por adm3 (barra verde ) con unos 10000.

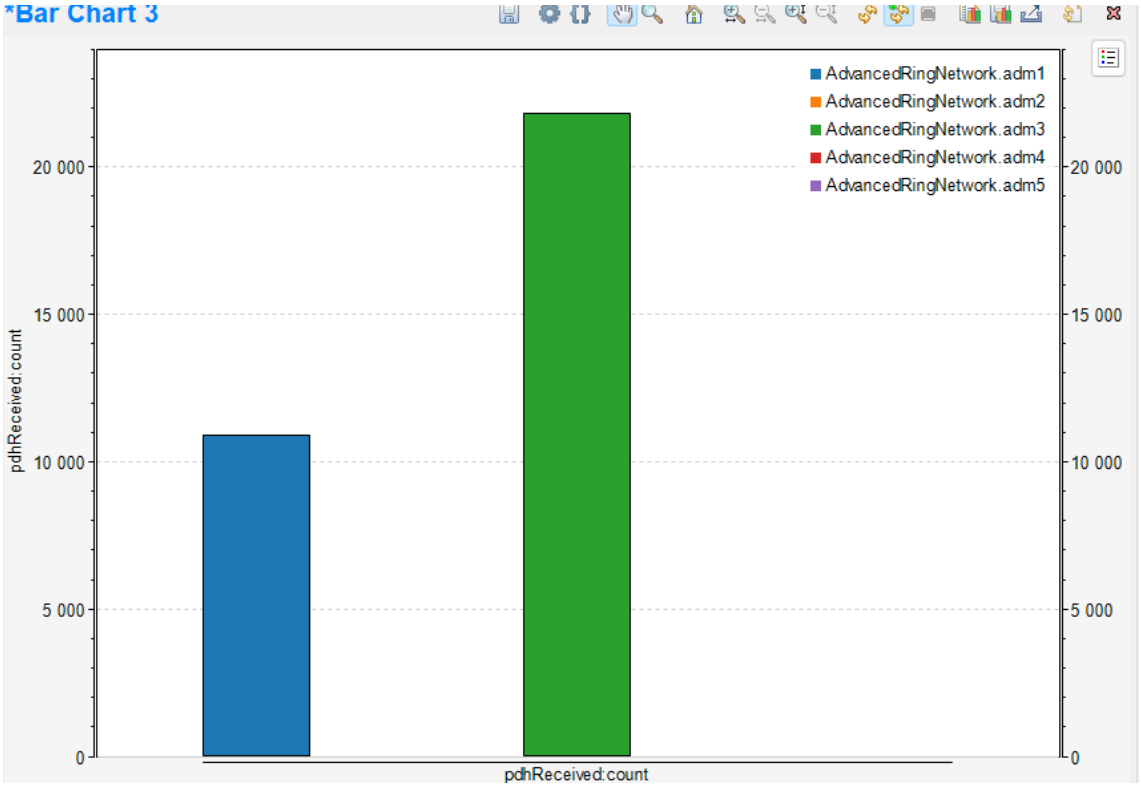


**Ilustración 16: Anillo avanzado pdhreceived**

En esta gráfica , se muestran cuantos paquetes PDH han sido recibidos por cada nodo ADM desde sus entradas. La barra azul correspondiente a adm1 , indica que este nodo ha inyectado apx. 11000 paquetes PDH al anillo. Por otro lado, la barra verde correspondiente a adm4 es el nodo que inyecta más tráfico al anillo (este nodo coincide con la gráfica anterior ).

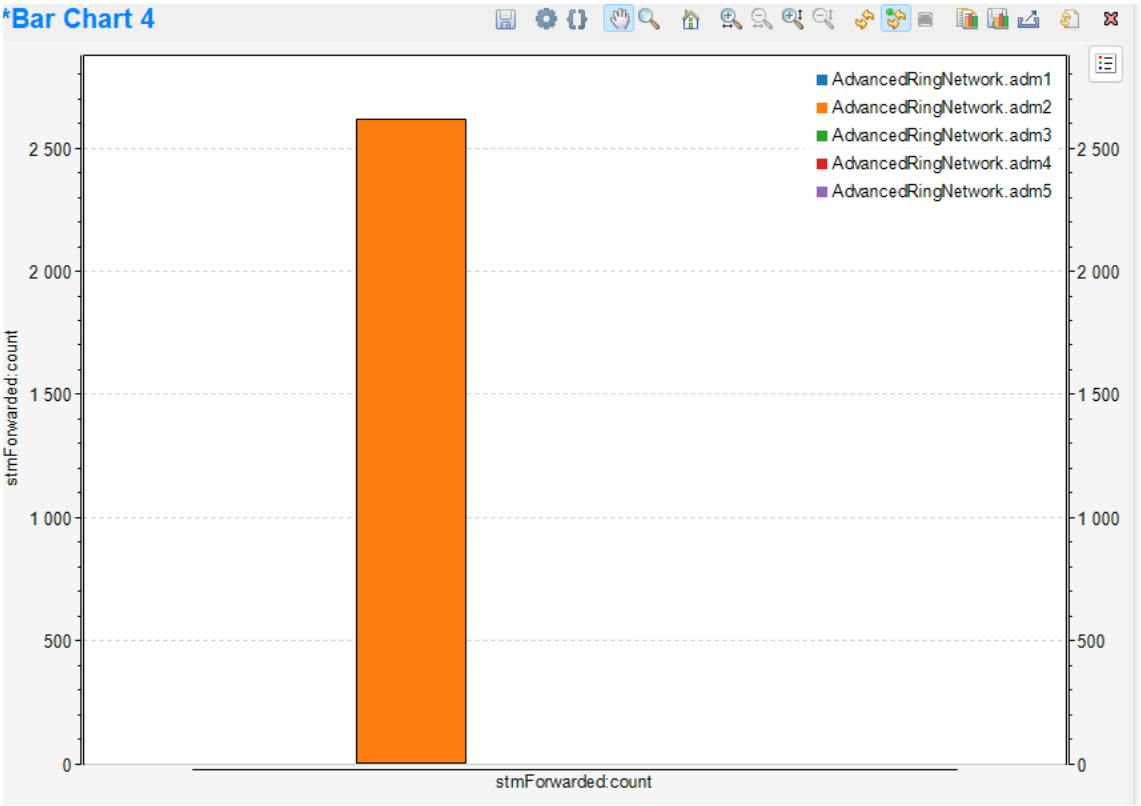


**Ilustración 17: Anillo avanzado stmForwarded**

Esta grafica indica el número de tramas STM que han sido reenviadas por el nodo, es decir, que no tienen un destino.

Tras el análisis podemos ver que hay una única barra que pertenece a adm2 , con un valor aprox de 26000. Esto indica que el nodo adm2 ha actuado como un nodo de retransmisión dentro de nuestro anillo, es decir , este nodo lo único que ha hecho es enviar tramas SDH que contienen datos que vienen de otros nodos

Finalmente podemos decir que la diferencia entre un anillo SDH simple y un anillo SDH avanzado radica en la flexibilidad de transmisión. El anillo simple transmite datos en una sola dirección, lo que lo hace mas vulnerable a fallos de enlace o nodo, mientras que el anillo avanzado permite transmisión bidireccional . En cuanto al STM.n , este valor determina la capacidad total del anillo , cuanto mayor sea más canales PDH podrá tener.

## Evaluación

Para la evaluación de la práctica, cada alumno/a deberá entregar este informe contestando a las cuestiones planteadas, y un enlace al repositorio GitHub en el que el profesor pueda descargar la carpeta *sdh*.

El ejercicio opcional podrá contar hasta un 30% del valor del resto de ejercicios de la práctica. La nota de la práctica será la suma de la nota del resto de ejercicios y la del ejercicio opcional.