SIMULACIÓN EN OPNET DE CONGESTIÓN DE RED USANDO TCP

KATHERINE PAOLA SERRANO RAMBAL
DAVID RICARDO BUENDIA BENITO-REBOLLO

DIRECTOR
GÓNZALO LÓPEZ VERGARA

MINOR EN TELECOMUNICACIONES
UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
PROGRAMA DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA
CARTAGENA DE INDIAS
2008

SIMULACIÓN EN OPNET DE CONGESTIÓN DE RED USANDO TCP

KATHERINE PAOLA SERRANO RAMBAL
DAVID RICARDO BUENDIA BENITO-REBOLLO

Trabajo de monografía presentado como requisito para obtener el Título de Ingeniero Electrónico.

DIRECTOR
OSCAR ACEVEDO PATIÑO
Ing. Electrónico

MINOR EN TELECOMUNICACIONES
UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
PROGRAMA DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA
CARTAGENA DE INDIAS
2008

Nota de Aceptación

_____

_____

_____

_____

Jurado

_____

Jurado

_____

Cartagena D. T. Y C., Abril de 2008

Señores

COMITÉ CURRICULAR

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

La Ciudad

Respetados Señores:

Con toda atención nos dirigimos a ustedes con el fin de presentarles a su consideración, estudio y aprobación la monografía titulada **SIMULACIÓN EN OPNET DE CONGESTIÓN DE RED USANDO TCP** como requisito parcial para optar al título de Ingeniero Electrónico.

Atentamente,

_____

KATHERINE SERRANO RAMBAL

Cartagena D. T. Y C., Abril de 2008

Señores

COMITÉ CURRICULAR

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

La Ciudad

Respetados Señores:

Con toda atención nos dirigimos a ustedes con el fin de presentarles a su consideración, estudio y aprobación la monografía titulada **SIMULACIÓN EN OPNET DE CONGESTIÓN DE RED USANDO TCP** como requisito parcial para optar al título de Ingeniero Electrónico.

Atentamente,

_____

DAVID RICADO BUEDIA BENITO-REVOLLO

Cartagena D. T. Y C., Abril de 2008

Señores

COMITÉ CURRICULAR

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

La Ciudad

Respetados Señores:

A través de la presente me permito entregar la monografía titulada **SIMULACIÓN EN OPNET DE CONGESTIÓN DE RED USANDO TCP S** para su estudio y evaluación la cual fue realizada por los estudiantes KATHERINE PAOLA SERRANO RAMBAL y DAVID RICARDO BUENDIA BENITO-REVOLLO, bajo mi dirección.

Atentamente,

_____

OSCAR ACEVEDO PATIÑO

Ingeniero Electrónico

AUTORIZACIÓN

Yo, DAVID RICARDO BUENDIA BENITO REBOLLO, identificado con cédula de ciudadanía número 73.209.064 de Cartagena, autorizo a la Universidad Tecnológica de Bolívar, para hacer uso de mi trabajo de monografía y publicarlo en el catálogo online de la biblioteca.

_____

DAVID RICARDO BUENDIA BENITO REBOLLO

AUTORIZACIÓN

Yo, KATHERINE PAOLA SERRANO RAMBAL, identificada con cédula de ciudadanía número 57.461.305 de Santa Marta, autorizo a la Universidad Tecnológica de Bolívar, para hacer uso de mi trabajo de monografía y publicarlo en el catálogo online de la biblioteca.

_____

KATHERINE PAOLA SERRANO RAMBAL

DEDICATORIA

Doy Gracias a Dios por iluminar mi camino y permitirme alcanzar mis metas.

A mis Padres por su compañía y por brindarme siempre su apoyo incondicional.

A mis familiares, por darme fuerza para seguir adelante.

A mis amigos, por compartir este triunfo conmigo y por su amistad incondicional.

KATHERINE PAOLA SERRANO RAMBAL

DEDICATORIA

Doy Gracias a Dios por iluminar mi camino y permitirme alcanzar mis metas.

A mis Padres por brindarme siempre su apoyo incondicional.

A mis familiares, por darme fuerza para seguir adelante.

A mis amigos, por compartir este triunfo conmigo y por su amistad incondicional.

DAVID RICARDO BUENDIA BENITO-REVOLLO

TABLA DE CONTENIDO

LISTA DE FIGURAS

1. INTRODUCCION

Internet es un conglomerado muy amplio y extenso en el que se encuentran ordenadores con sistemas operativos incompatibles, redes más pequeñas y distintos servicios con su propio conjunto de protocolos para la comunicación. Ante tanta diversidad resulta necesario establecer un conjunto de reglas comunes para la comunicación entre estos diferentes elementos y que además optimice la utilización de recursos tan distantes. Este papel lo tiene el protocolo TCP. También, TCP puede usarse como protocolo de comunicación en las redes privadas intranet y extranet. Las siglas TCP se refieren a dos protocolos de red, que son **Transmission Control Protocol** (*Protocolo de Control de Transmisión).*

TCP Controla la división de la información en unidades individuales de datos (llamadas paquetes) para que estos paquetes sean encaminados de la forma más eficiente hacia su punto de destino. En dicho punto, TCP se encargará de reensamblar dichos paquetes para reconstruir el fichero o mensaje que se envió. Por ejemplo, cuando se nos envía un fichero HTML desde un servidor Web, el protocolo de control de transmisión en ese servidor divide el fichero en uno o más paquetes, numera dichos paquetes y se los pasa al protocolo IP.

Aunque cada paquete tenga la misma dirección IP de destino, puede seguir una ruta diferente a través de la red. Del otro lado (el programa cliente en nuestro ordenador), TCP

reconstruye los paquetes individuales y espera hasta que hayan llegado todos para presentárnoslos como un solo fichero.

El Protocolo de Transmisión (TCP), fue desarrollado inicialmente en 1973 por el informático estadounidense Vinton Cerf como parte de un proyecto dirigido por el ingeniero norteamericano Robert Kahn y patrocinado por la Agencia de Programas Avanzados de Investigación (ARPA, siglas en inglés) del Departamento Estadounidense de Defensa. La finalidad principal de esta red era la capacidad de resistir un ataque nuclear de la URSS para lo que se pensó en una administración descentralizada. De este modo, si algunos ordenadores eran destruidos, la red seguiría funcionando. Aunque dicha red funcionaba bien, estaba sujeta a algunas caidas periódicas del sistema. De este modo, la expansión a largo plazo de esta red podría resultar difícil y costosa. Se inició entonces una búsqueda de un conjunto de protocolos más fiables para la misma. Dicha búsqueda finalizó, a mediados de los 70, con el desarrollo de TCP y de IP.

El Control de la Congestión en TCP se maneja presenta al detectar una pérdida de paquetes ya que no se puede enviar paquetes nuevos sin recuperar los paquetes perdidos. El control se logra manipulando dinámicamente el tamaño de la ventana. Las causas de la pérdida de paquetes puede ser ruido en la línea (actualmente se desprecia este caso salvo en líneas inalámbricas) y congestión; la gran mayoría se debe a congestión. Al establecerse la conexión TCP, el receptor propones un tamaño de ventana en función de su buffer. A medida que se presenta los problemas de congestión se van optando por soluciones que permitan la transmisión eficiente de los datos.

Existen variedades de métodos que permiten el flujo normal de los datos después de haberse presentado congestión en el envió y recepción, mecanismos como: "evitación de congestion" *congestion avoidance*, "arranque lento" *slow start*, "retransmisión rápida" *fast retransmi*, "tiempo fuera" *timeout*.

La monografía presentada es una introducción a TCP y al Control de Congestión en TCP, incluyendo la realización de prácticas con el programa OPNET IT Guru, en las cuales se explica la creación de escenarios que permitan visualizar la congestión en TCP y la evitación de la misma, estadísticas representativas del comportamiento de la red creada, y finalmente el análisis de las imágenes que se despliegan como resultado de la simulación de cada red.

FUNDAMENTOS DE TCP

2. FUNDAMENTOS DE TCP:

Transmission Control Protocol

El *Protocolo de Control de Transmisión* (TCP en sus siglas en inglés) fue creado entre los años 1973-1974 por Vint Cerf y Robert Kahn. Es uno de los protocolos fundamentales en Internet.

Muchos programas dentro de una red de datos compuesta por ordenadores pueden usar TCP para crear conexiones entre ellos a través de las cuales se envían datos. TCP, fue diseñado para trabajar una capa de red no confiable, lo cual en un futuro, cuando se cuente con medios físicos de transmisión libres de errores no será necesario. Sin embargo, TCP tiene muchas más características que lo hacen robusto y práctico. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto (computación). TCP soporta muchas de las aplicaciones más populares de Internet, incluidas HTTP, SMTP y SSH.

El **Protocolo de Control de Transmisión** (TCP) es un protocolo de comunicación orientado a conexión y fiable del nivel de transporte, actualmente documentado por IETF RFC 793[1]. TCP encaja en una arquitectura de protocolos en capas justo por encima del *Protocolo de Internet* IP, protocolo básico que proporciona un medio para TCP de enviar y recibir segmentos de longitud variable de información envuelta en "sobres" de datagramas de Internet. El datagrama de Internet proporciona un medio de direccionar TCPs de origen y de destino situados en redes diferentes. El protocolo de Internet también trata con la fragmentación y el reensamble de segmentos de TCP que sean necesarios para conseguir el transporte y la entrega sobre múltiples redes y las puertas de enlace que las

[1] **IETF RFC 793** (-Internet Engineering Task Force- Request For Comments 793): Documento que describe las funciones que debe realizar el protocolo de control de transmisión, el programa que lo implementa, y su interfaz con los programas o usuarios que requieran de sus servicios.

interconectan. El protocolo de Internet también lleva información sobre la prioridad, clasificación de seguridad y compartimentación de los segmentos de TCP, de tal forma que esta información pueda ser comunicada de extremo a extremo entre múltiples redes.

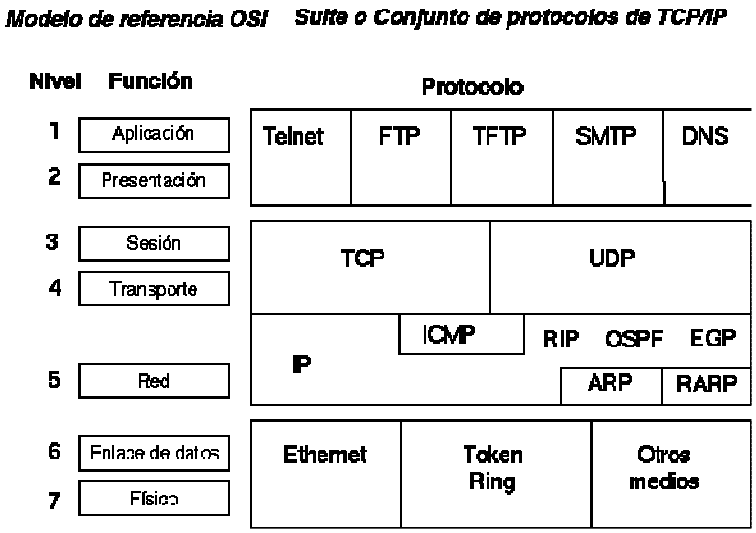


Figura 2. 1 Capas de Protocolos

En la **figura 2.1** podemos observar las capas de protocolos, donde cada una tiene la responsabilidad de manejar una parte del problema.

Para proporcionar un servicio fiable a la aplicación, el TCP se basa en los siguientes principios:

1. **Transmisión libre de error**. El TCP debe entregar a la aplicación de destino exactamente la misma información que le entregó la aplicación de origen. De hecho, se trata de una entrega "casi libre" de errores, puesto que puede haber algunos que un mecanismo de detección de errores no pueda detectar.
2. **Garantía de entrega de la información**. El TCP garantiza que toda la información transmitida por la aplicación de origen se entregue a la aplicación de destino. Si no es posible, el TCP debe avisar a la aplicación.
3. **Garantía de mantenimiento de la secuencia de transmisión**. El TCP garantiza la entrega del flujo de información en el mismo orden en que le fue entregado por la aplicación de origen.

4. **Eliminación de duplicados**. El TCP garantiza que sólo entregará una copia de la información transmitida a la aplicación de destino. En caso de que reciba copias a causa del funcionamiento de la red o de los protocolos que se implementan por debajo del nivel de transporte, el TCP las eliminará.

La fiabilidad de la transmisión que proporciona TCP se consigue gracias a las siguientes estrategias:

- **El TCP está orientado a la conexión:** tiene una fase de establecimiento de la conexión, una de transmisión de datos y una de desconexión.
- **El TCP utiliza el concepto *buffered transfer*:** cuando se transfiere información, el TCP divide los flujos de datos (*byte stream*) que le pasa la aplicación en segmentos del tamaño que le convenga. El TCP decide el tamaño de los segmentos tanto si la aplicación genera un byte de información, como si genera flujos de grandes dimensiones. En el primer caso, el TCP puede esperar a que la memoria intermedia se llene más antes de transferir la información, o bien la puede transferir de inmediato (mecanismo *push*). En caso de que los flujos sean muy grandes, el TCP puede dividir la información en tamaños más pequeños antes de transferirlos.
- **El TCP utiliza una conexión *full duplex*:** la transferencia de información es en ambos sentidos. La aplicación ve dos flujos independientes. En caso de que la aplicación cierre uno de los flujos, la conexión pasa a ser *half duplex*. Ello significa que uno de los extremos (el que no ha cerrado la conexión) puede continuar enviando información por el canal, mientras que el otro extremo (el que ha cerrado la conexión) se limita a reconocer la información. No obstante, no es normal encontrar este caso.

2. FUNCIONES DE TCP

En la pila de protocolos TCP/IP, TCP es la capa intermedia entre el Protocolo de Internet (IP) y la aplicación. Habitualmente, las aplicaciones necesitan que la comunicación sea fiable y, dado que la capa IP aporta un servicio de datagramas no fiable (sin confirmación), TCP añade las funciones necesarias para prestar un servicio que permita que la comunicación entre dos sistemas se efectúe:

1. libre de errores.

2. en orden.
3. sin pérdidas.
4. sin duplicaciones.

Dentro de las funciones principales de TCP, encontramos:

- Dividir la información que recibe de la capa de aplicación en segmentos que pasarán a la capa de red.
- Al enviar un segmento inicializa un reloj, en espera de una contraseña (indicando que el mensaje se recibió); si el reloj expira antes que esta última se reciba, reenvía el segmento suponiendo que el segmento se ha perdido.
- Cuando TCP recibe un mensaje, envía al remitente una contraseña confirmando la recepción.
- Implementa algoritmos para verificar que la información recibida fue la misma que la enviada; en caso de que el segmento llegue dañado a su destino, se indica al remitente del hecho y este último lo reenvía. Puesto que IP no garantiza el orden de llegada de los segmentos que envía, TCP debe reordenarlos si es necesario.
- Da la impresión a una aplicación de tener una línea directa en ambos sentidos (full duplex) a través de la cual se realiza la comunicación. TCP otorga a la capa de aplicación una comunicación libre de errores punto a punto (de fuente a destino) que aparenta ser orientada a conexión (aun cuando siempre se implemente mediante servicios no orientados a conexión); a esta conexión se le conoce como conexión TCP. TCP define un nivel de direccionamiento, llamado puerto, que permite distinguir entre diferentes conexiones que se estén realizando simultáneamente. Cada puerto es identificado con un número de 16 bits. Su uso es claramente ejemplificado por el modelo cliente-servidor. Para que el cliente pueda conectarse con el servidor, es necesario que el primero sepa dónde encontrar al segundo; para resolver este problema, varios números de puertos están reservados para algunas aplicaciones (correo electrónico, telnet, ftp, web, etc.).

3. FORMATO DEL SEGMENTO TCP

La unidad de información del protocolo TCP se llama *segmento TCP* y su formato es el siguiente:

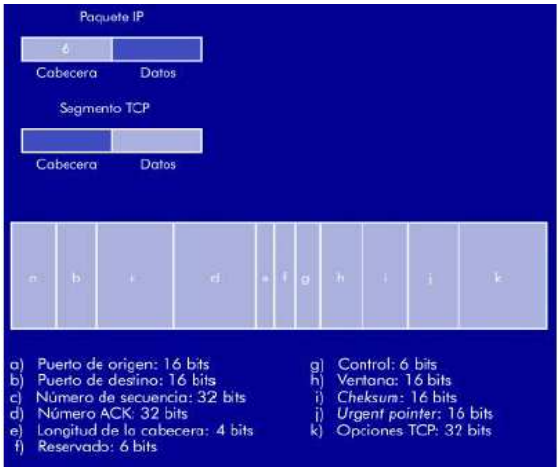


Figura 2.2.1 Formato de segmento TCP

En la **figura 2.2.1** observamos el formato de segmento de TCP, donde el segmento TCP consta de una cabecera y un cuerpo para encapsular datos.

La cabecera consta de los siguientes campos:

1. El campo *Puerto de origen*, que identifica la aplicación en el terminal de origen, su ubicación es la observada en la **figura 2.2.2**

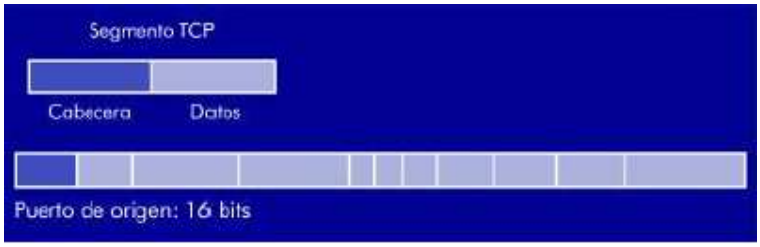


Figura 2.2.2 Campo Puerto de Origen

2. El campo *Puerto de destino*, que identifica la aplicación en el terminal de destino. Como podemos ver en la **figura 2.2.3**, la ubicación del campo puerto de destino es continua a la del campo puerto de origen.

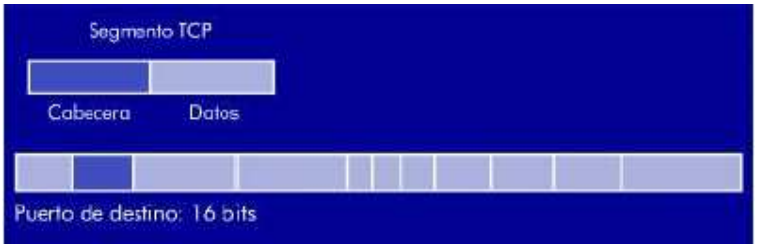


Figura 2.2.3 Campo Puerto de destino

Además **figura 2.2.2** y la **figura 2.2.3** nos muestran dos campos de 16 bits cada uno, los cuales identifican a los programas de aplicación de nivel superior que utiliza la conexión TCP.

3. El campo *Número de secuencia*, el cual identifica el primer byte del campo de datos. En TCP no se numeran segmentos, sino bytes. Por tanto, el número de secuencia identifica el primer byte de los datos que envía el segmento: al principio de la conexión se asigna un número de secuencia inicial (ISN, del inglés *inicial sequence number*), a partir del cual el TCP numera los bytes consecutivamente.

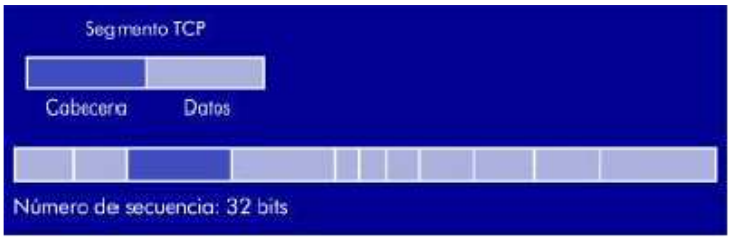


Figura 2.2.4 Campo número de secuencia

La **figura 2.2.4** nos muestra el campo de número de secuencia, que contiene el número de secuencia del primer octeto del campo de datos de usuario. Su valor especifica la posición de la cadena de bits del módulo de transmisor.

4. El campo *Número ACK*.: TCP reconoce datos por medio de la técnica de *piggybacking*. Al activar un bit de la cabecera (el bit ACK), el TCP tiene en cuenta el número de secuencia ACK que indica al otro extremo TCP el próximo byte que está dispuesto a recibir. Dicho de otra manera, el

número ACK menos uno indica el último byte reconocido. La estructura del campo puede verse en la **figura 2.2.5**

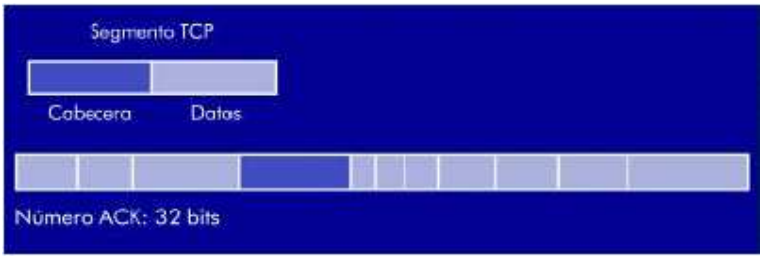


Figura 2.2.5 Campo Número ACK

5. El campo *Longitud de la cabecera*, indica la longitud de la cabecera, que puede ser variable. La longitud típica es de 20 bytes; sin embargo, si el TCP utiliza el campo de opciones, puede llegar a una longitud máxima de 60 bytes. De este modo, el TCP sabe dónde empiezan los datos.

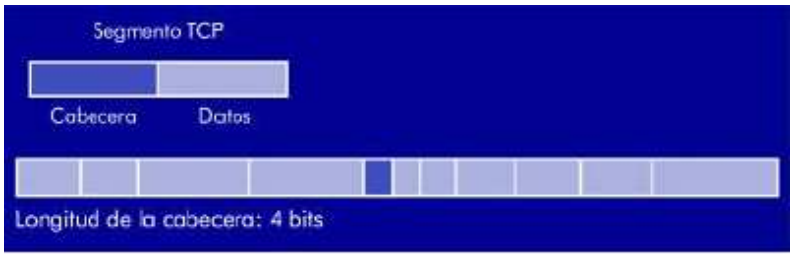


Figura 2.2.6 Campo Longitud de cabecera

La **figura 2.2.6** nos muestra la posición del campo de longitud de cabecera, el cual se puede iniciar en 4 bits y dependiendo del campo de opciones está puede variar.

6. El campo *Reservado*, como su nombre lo indica, está reservado y se inicializa en cero. La **figura 2.2.7** nos muestra la ubicación del campo reservado.

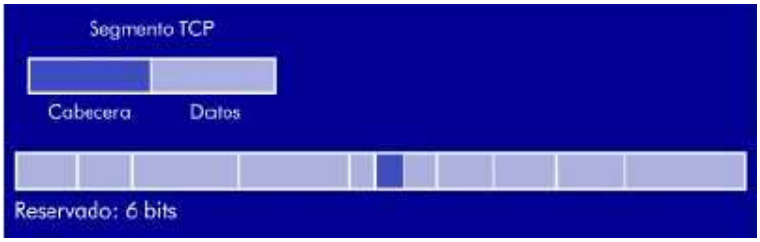


Figura 2.2.7 Campo Reservado

7. El campo de *Control* está formado por seis indicadores independientes, cada uno de los cuales señala una función específica del protocolo cuando está activo (a 1). La **figura 2.2.8** nos muestra el campo de control, donde se encuentran ubicados los *flags*, bits de control que se utilizan para especificar ciertos servicios o utilidades que se pueden emplear durante la sesión.

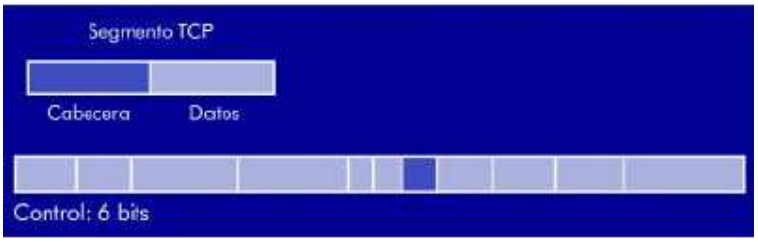


Figura 2.2.8 Campo de Control

INDICADOR	FUNCIÓN
URG	Indica que hay datos urgentes (y el campo *Urgent pointer* indica la cantidad de datos urgentes existentes en el segmento).
ACK	Cuando este bit está activo, el campo *Número ACK* indica el byte siguiente que espera recibir conexión TCP. Si este bit no está activo, el campo *Número ACK* no tiene ningún significado para el TCP.
PSH	Invoca la función *push* en el protocolo. Esta función dice al receptor que entregue a la aplicación todos los datos que tenga disponibles en la memoria intermedia de reopción sin esperar a completarlos con datos adicionales. De este modo, los datos no esperan en la memoria intermedia receptora hasta completar un segmento de dimensión máxima.
RST	Realiza un *reset* de la conexión.
SYN	Se utiliza para iniciar una conexión y también sirve para resincronizar los números de secuencia.
FIN	Indica que el transmisor ha acabado la conexión.

Tabla 2 Indicadores Independientes

8. El campo *Ventana* indica cuántos bytes componen la ventana de transmisión del protocolo de control de flujo por ventana deslizante. A diferencia de los protocolos de nivel de enlace, en que la ventana era constante y contaba tramas, en el TCP la ventana es variable y cuenta bytes. Con cada segmento transmitido, un extremo TCP advierte al otro extremo de la cantidad de datos que está dispuesto a recibir en cada momento. De

este modo, el extremo que recibe un segmento actualiza el tamaño de su ventana de transmisión.

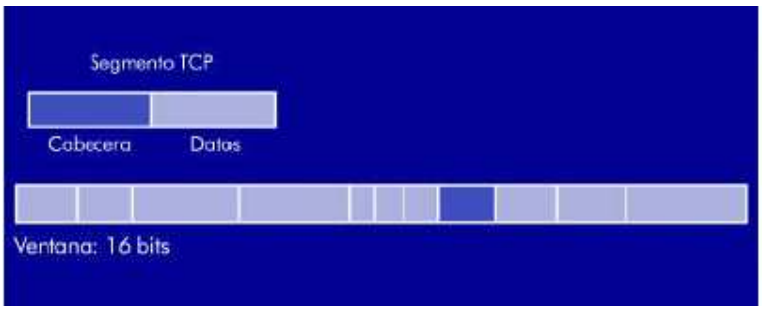


Figura 2.2.9 Campo Ventana

Como observamos en la **figura 2.2.9** el campo Ventana, se pone a un valor que indica cuántos octetos desea aceptar el receptor. Este valor se establece teniendo en cuenta el valor del campo de aceptación.

9. El campo *Checksum* se utiliza para detectar errores.

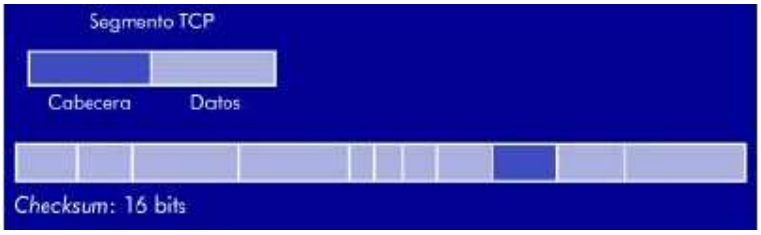


Figura 2.2.10 Campo Checksum

El campo checksum contiene el complemento a uno de 16 bits del complemento a uno de la suma de todas las palabras de 16 bits del segmento, incluyendo la cabecera y el texto, como lo muestra la **figura 2.2.10** Esto se realiza para determinar si el segmento procedente del transmisor ha llegado libre de errores.

10. El campo *Urgent pointer* tiene sentido cuando el bit de control URG está activo. Indica que los datos que envía el origen son urgentes e identifica el último byte de dicho campo. La aplicación es la que indica que estos últimos son urgentes y lo sabe porque el TCP se lo indica en lea recepción.

Figura 2.2.11 Campo Puntero Urgente

La **figura 2.2.11** nos muestra la ubicación del campo puntero urgente, el cual se utiliza sólo si el indicador URG está en 1. su función es identificar el octeto da datos al que siguen datos urgente.

11. El campo *Opciones TCP* permite añadir campos a la cabecera para realizar las siguientes operaciones:

1. Marcar el tiempo (tiemstamp) en que se transmitió el segmento y de este modo poder monitorizar los retardos que experimentan los segmentos desde el origen hasta el destino.
2. Aumentar el tamaño de la ventana.
3. Indicar el tamaño máximo del segmento (MSS, del inglés *maximum segment size*) que el origen está preparado para recibir. Por tanto, el receptor no le puede transmitir segmentos por encima de este valor.

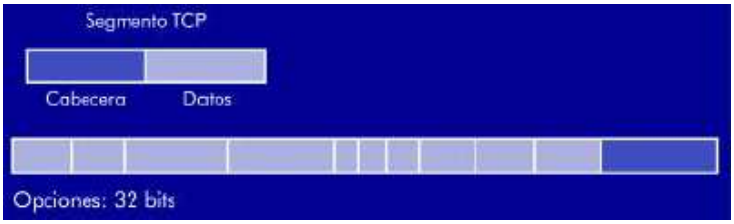


Figura 2.2.12 Campo de Opciones

La **figura 2.2.12** nos muestra la última ubicación de los Formatos del Segmento de TCP, que equivale al campo de Opciones, el cual está concebido para posibilitar futuras mejoras de TCP. Cada opción se especifica mediante un byte que especifica el número d opción, un campo que contiene la longitud de la opción y finalmente, los valores de la opción propiamente dichos.

4. INTERFACEZ

TCP presenta interfaz por un lado con el usuario o los procesos de aplicación y por el otro con un protocolo de más bajo nivel como es el protocolo de Internet (IP).

La interfaz entre un proceso de aplicación y TCP se ilustra con un detalle razonable. Esta interfaz consiste en un conjunto de llamadas, de forma muy similar a las llamadas que un sistema operativo proporciona a los procesos de aplicación para manipular ficheros. Por ejemplo, hay llamadas para abrir y cerrar conexiones y para enviar y recibir datos por las conexiones establecidas. Se exige también que TCP pueda comunicarse asíncronamente con los programas de aplicación.

Aunque se deja considerable libertad a los fabricantes de implementaciones de TCP a la hora de diseñar las interfaces que sean apropiadas para el entorno de un sistema operativo particular, se exige un mínimo de funcionalidad en la interfaz TCP/usuario de cualquier implementación válida. La interfaz entre TCP y el protocolo de nivel inferior queda esencialmente sin especificar, exceptuando el hecho de que se asume que hay un mecanismo por el cual los dos niveles pueden pasar información asíncronamente el uno al otro. Típicamente, se espera que el protocolo de nivel inferior especifique esta interfaz. Se ha diseñado TCP para trabajar en un entorno muy genérico de redes interconectadas.

5. OPERACIÓN

Como ha sido mencionado anteriormente, el propósito principal de TCP consiste en proporcionar un servicio de conexión o circuito lógico fiable y seguro entre pares de procesos. Para proporcionar este servicio encima de un entorno de Internet menos fiable, el sistema de comunicación requiere de mecanismos relacionados con las siguientes áreas:

1. Transferencia básica de datos
2. Fiabilidad
3. Control de flujo
4. Multiplexamiento
5. Conexiones
6. Prioridad y seguridad.

La operación básica de TCP en cada una de estas áreas se describe a acontinuación:

Transferencia básica de datos: TCP es capaz de transferir un flujo continuo de octetos en cada sentido entre sus usuarios empaquetando un cierto número de octetos en segmentos para su transmisión a través del sistema de Internet. En general, los módulos de TCP deciden cuándo bloquear y enviar datos según su propia conveniencia.

Algunas veces los usuarios necesitan estar seguros de que todos los datos que habían entregado al módulo de TCP han sido transmitidos. Para este propósito se define una función 'push' ("enviar inmediatamente"). Para asegurar que los datos entregados al módulo de TCP son realmente transmitidos, el usuario emisor debe indicarlo mediante la función 'push'. Un 'push' en un cierto instante causa que los módulos de TCP envíen y entreguen inmediatamente al usuario receptor los datos almacenados hasta ese instante. El instante exacto en que se ejecuta la función 'push' podría no ser visible para el usuario receptor. Tampoco la función 'push' proporciona una marca de límite de registros.

Fiabilidad: el módulo de TCP debe poder recuperar los datos que se corrompan, pierdan, dupliquen o se entreguen desordenados por el sistema de comunicación del entorno de Internet. Esto se consigue asignando un número de secuencia a cada octeto transmitido, y exigiendo un acuse de recibo (ACK, N.T.: del inglés 'acknowledgment') del módulo de TCP receptor. Si no se recibe un ACK dentro de un cierto plazo de expiración prefijado, los datos se retransmiten. En el receptor, se utilizan los números de secuencia para ordenar correctamente los segmentos que puedan haber llegado desordenados para eliminar los duplicados. La corrupción de datos se trata añadiendo un campo de suma de control ('checksum') a cada segmento transmitido, comprobándose en el receptor y descartando los segmentos dañados. Mientras los módulos de TCP continúen funcionando adecuadamente y el sistema de Internet no llegue a quedar particionando de forma completa, los errores de transmisión no afectarán la correcta entrega de datos. TCP se recupera de los errores del sistema de comunicación de Internet.

Flujo de control: TCP proporciona al receptor un medio para controlar la cantidad de datos enviados por el emisor. Esto se consigue devolviendo una "ventana" con cada ACK, indicando el rango de números de secuencia aceptables más allá del último segmento recibido con éxito. La ventan indica el número de octetos que se permiten que el emisor transmita antes de que reciba el siguiente permiso.

Multiplexamiento: para permitir que muchos procesos dentro de un único 'host' utilicen simultáneamente las posibilidades de comunicación de TCP, el módulo de TCP proporciona una serie de direcciones o puertos dentro de cada 'host'. Conectadas con las direcciones de red y de 'host' de la capa de comunicación de Internet conforman lo que se denomina una dirección de conector ('socket'). Untar de direcciones de conector identifica de forma única la conexión. Es decir, un conector puede utilizarse simultáneamente en múltiples conexiones.

La asignación de puertos a los procesos se gestiona de forma independiente en cada 'host'. Sin embargo, resulta de la máxima utilidad asignar a los procesos más utilizados frecuentemente conectores fijos que se hacen conocer de forma pública. Estos servicios pueden entonces ser accedidos a través de direcciones conocidas públicamente. El establecimiento y aprendizaje de la direcciones de los puertos de otros procesos puede involucrar otros mecanismos más dinámicos.

Conexiones: la fiabilidad y los mecanismos de control de flujo descritos anteriormente, exigen que los módulos de TCP inicialicen y mantengan una información de estado para cada flujo de datos. La combinación de esta información, incluyendo las direcciones de los conectores, los números de secuencia y los tamaños de las ventanas, se denomina una conexión. Cada conexión queda especificada de forma única por un par de conectores que corresponden con sus dos extremos.

Cuando dos procesos desean comunicarse, sus módulos de TCP deben establecer primero una conexión (inicializar la información de estado en cada lado). Cuando la comunicación se ha completado, la conexión se termina o cierra con la intención de liberar recursos para otros usos.

Como las conexiones tiene que establecerse entre 'host' no fiables y sobre un sistema de comunicación Internet no fiable, se utiliza un mecanismo de acuerdo que usa números de secuencia basados en tiempo de reloj para evitar una inicialización errónea de conexiones.

Prioridad y seguridad: los usuarios TCP pueden indicar el nivel de seguridad y prioridad de su comunicación. Se emplean valores por defecto cuando estas características no se necesiten.

6. PUERTOS TCP

TCP usa el concepto de *número de puerto* para identificar a las aplicaciones emisoras y receptoras. Cada lado de la conexión TCP tiene asociado un número de puerto (de 16 bits sin signo, con lo que existen 65536 puertos posibles) asignado por la aplicación emisora o receptora. Los puertos son clasificados en tres categorías: bien conocidos, registrados y dinámicos/privados. Los puertos bien conocidos son asignados por la Internet Assigned Numbers Authority (IANA), van del 0 al 1023 y son usados normalmente por el sistema o por procesos con privilegios. Las aplicaciones que usan este tipo de puertos son ejecutadas como servidores y se quedan a la escucha de conexiones. Algunos ejemplos son: FTP (21), SSH (22), Telnet (23), SMTP (25) y HTTP (80). Los puertos registrados son normalmente empleados por las aplicaciones de usuario de forma temporal cuando conectan con los servidores, pero también pueden representar servicios que hayan sido registrados por un tercero. Los puertos dinámicos/privados también pueden ser usados por las aplicaciones de usuario, pero este caso es menos común. Los puertos dinámicos/privados no tienen significado fuera de la conexión TCP en la que fueron usados.

3. CONTROL DE CONGESTIÓN EN TCP

Control de Congestión es el esfuerzo hecho por los nodos de la red para prevenir o responder a sobrecargas de la red que conducen a perdidas de paquetes. El objetivo es asignar los recursos de la red en forma "equitativa"; es decir cuando haya problemas compartir sus efectos entre todos los usuarios, en lugar de causar un gran problema a tan solo unos pocos.

En el Control de Congestión se presentan dos situaciones:

α. Pre-asignar recurso (ancho de banda y espacio de buffers en routers y switches) para abolir la congestión.
β. Controlar la congestión si ocurre.

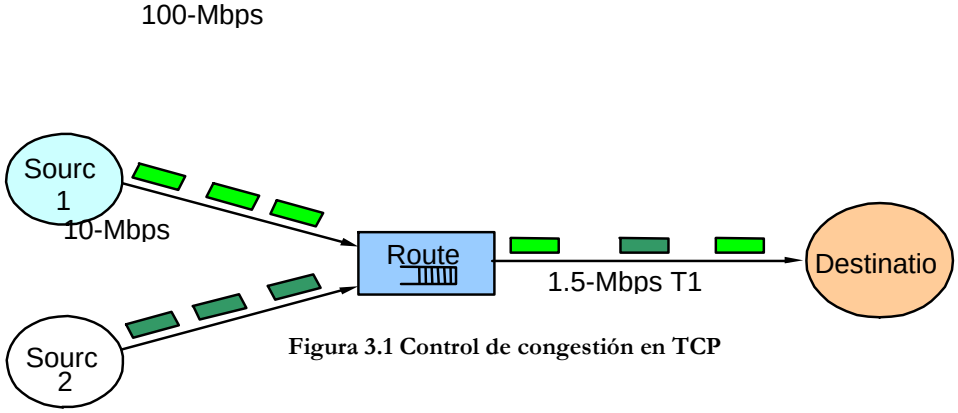


Figura 3.1 Control de congestión en TCP

La **figura 3.1** nos muestra la idea básica en el Control de Congestión, la cual es no intentar insertar un nuevo paquete en la red si es que uno anterior no la ha abandonado (se ha entregado). TCP logra la premisa anterior cambiando el tamaño de su ventana.

Cuando se inicia una conexión, el emisor manda segmentos de 1K, luego de 2K, luego de 4K y así sucesivamente esperando una confirmación en cada caso. Cuando la confirmación falla, digamos en 2K, la ventana de congestión se pone de ese tamaño. A este proceso se le llama "comienzo lento".

Cada emisor tiene dos ventanas: la de congestión y la que confirma el receptor. Si el receptor dice "envíame segmentos de 4K" pero el emisor ya descubrió que esos segmentos no son confirmados porque rebasan la capacidad de la subred, pone su "ventana confirmada" en 4K y la de congestión en 2K, y finalmente enviará segmentos del tamaño que resulte menor de las dos ventanas, lo cual será la "ventana efectiva".

El algoritmo completo contiene otra ventana llamada "holgura" y sirve para afinar el crecimiento exponencial del algoritmo de comienzo lento. La ventana de holgura tiene un tamaño inicial de 64 Kb. cada vez que hay un timeout en el comienzo lento, el valor de la holgura será puesta al valor de la ventana de congestión dividido por dos, y el comienzo lento vuelve a arrancar, excepto que el crecimiento exponencial se detiene cuando se alcanza la holgura. De ahí en adelante la ventana de congestión crece de uno en uno.

Por ejemplo, supongamos que una subred puede procesar hasta 12 Kb. Inicialmente la ventana de congestión es 1K, la de confirmación es 0 y la de holgura es 64 K. El emisor inicia el comienzo lento y envía un paquete de 1 K, y el receptor le dice "mándame 24K". El emisor pone la de congestión en 1K, la confirmada en 24K y la holgura en 64K y envía un paquete de 2K. El receptor dice "envíame 24K. El emisor pone la de congestión en 2K y envía un paquete de 4K y obtiene la misma respuesta del receptor. El emisor pone la de congestión en 4K y envía un segmento de 8K, obtiene la misma respuesta y cambia la ventana de congestión a 8K y envía un segmento de 16K, pero como la subred solo

soporta 12K, ocurre un timeout y tiene que poner su ventana de holgura a la mitad de la de congestión, es decir, a 4K y reinicia la de congestión a cero. Repite el comienzo lento hasta que su segmento enviado es de 4K, en ese momento su siguiente paquete no será de 8K, sino de 5K, luego de 6,7,8,9,10,11,12,13K donde habrá un timeout y se sabrá que el tamaño de ventana efectiva es de 12K.

En el Control de la Congestión realizado por TCP, cada fuente determina cual es la capacidad disponible en la red, con el fin de conocer cuantos paquetes puede tener en tránsito de una manera segura. Para esto, TCP mantiene una variable de estado distinta por cada conexión, que se denomina *ventana de congestión*. El host fuente utilizará esta variable para limitar la cantidad de datos que puede tener en tránsito en un momento determinado. TCP utiliza un mecanismo, llamado "incremento aditivo"/"decremento multiplicativo" (*additive increase/ multiplicative decrease*), que reduce la ventana de congestión cuando el nivel de congestión aumenta, y que incrementa la ventana de congestión cuando el nivel de congestión disminuye. TCP interpreta los *timeouts* como signo de congestión. Cada vez que ocurre un *timeout*, el host origen pone su ventana de congestión a la mitad del valor que tenía previamente. Esta división se corresponde con la parte de "decremento multiplicativo" de este mecanismo. La ventana de congestión no puede caer debajo del tamaño de un único segmento TCP, dado por el parámetro de TCP "tamaño máximo de segmento" (más conocido por su nombre en inglés *Maximum Segment Size* o *MSS*). Cada vez que un host origen envía con éxito todos los paquetes que caben en la ventana de congestión, se incremente la ventana de congestión en el tamaño equivalente a un paquete. Esta sería la parte de "incremento aditivo" del mecanismo (también llamado algoritmo de Van Jacobson).

El Control de la Congestión en TCP se maneja de la siguiente manera: Al detectar una pérdida de paquetes no se puede enviar paquetes nuevos sin recuperar los paquetes perdidos, esto se logra manipulando dinámicamente el tamaño de la ventana. Las causas de la pérdida de paquetes puede ser ruido en la línea (actualmente se desprecia este caso salvo en líneas inalámbricas) y congestión; la gran mayoría se debe a congestión.

Al establecerse la conexión TCP, el receptor propones un tamaño de ventana en función de su buffer. A medida que se presenta los problemas de congestión se van optando por soluciones que permitan la transmisión eficiente de los datos.

Existen variedades de métodos que permiten el flujo normal de los datos después de haberse presentado congestión en el envió y recepción, mecanismos como: "evitación de congestion" *congestion avoidance*, "arranque lento" *slow start*, "retransmisión rápida" *fast retransmi*, "tiempo fuera" *timeout*.

- **Prevención de Congestión** (*Congestion avoidance)*

Al detectarse congestión en TCP se entra en una fase de prevención de congestión. Si la ventana de congestión tiene un valor de N tamaño máximo de segmento MSS, cada vez que se envía con éxito toda la ventana, la ventana sube linealmente buscando encontrar el punto de equilibrio sin aumentar demasiado la congestión.

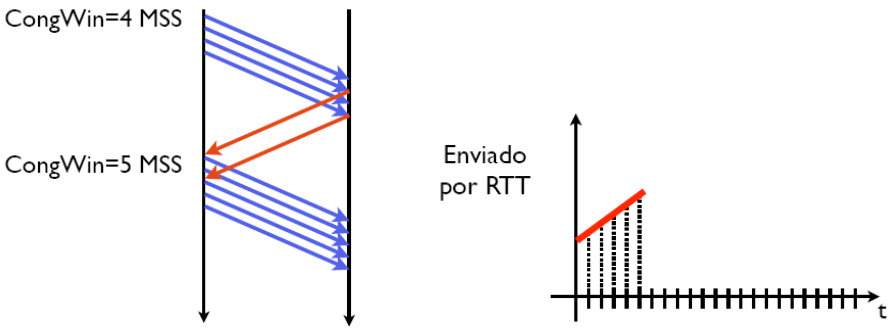


Figura 3.2 Prevención de Congestión

La **figura 3.2** nos muestra el proceso de prevención de congestión, podemos afirmar que si al realizar este procedimiento se produce una pérdida de un solo paquete, se provocarían 3 ACKs duplicados, lo que se interpreta como una congestión ligera, donde la ventana se reduce inmediatamente a la mitad a la vez que se hace *fast retrasmit* del paquete perdido, donde se busca reducir notablemente la tasa de trasmisión y colaborar en que la congestión no crezca. Si se siguen recibiendo ACKs, la ventana sigue creciendo linealmente.

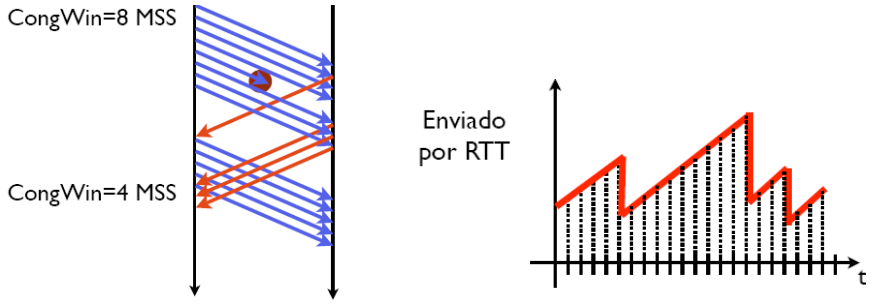


Figura 3.3 Crecimiento de la Ventana

Se puede ver en la **figura 3.3**, que la tasa se va ajustando alrededor del punto de congestión, si hay muchos errores la tasa baja y se tarda más en recuperarla, si hay pocos errores el crecimiento lineal es capaz de llegar mas a mas velocidad, este mecanismo se suele llamar **AIMD** *Additive Increase Multiplicative Decrease*[2].

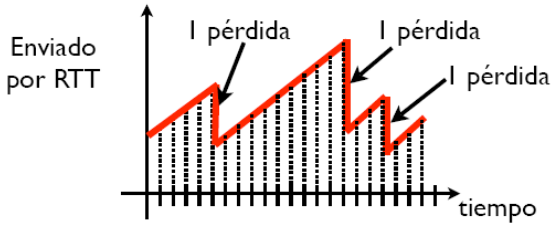


Figura 3.4 Mecanismo AIMD

[2] **AIMD** *Additive Increase Multiplicative Decrease*: Mecanismo que consiste en la partida desde una tasa de transferencia baja se va aumentando un paquete por RTT (Round Trip Time) y al observarse pérdidas la tasa se reduce a la mitad. Este mecanismo funciona muy bien para redes de 10/100Mbps locales.

La **figura 3.4** nos muestra el mecanismo AIMD y su funcionamiento de acuerdo a la baja transferencia, el aumento de un paquete y las perdidas, ocasionan reducción de la tasa a la mitad.

- **Arranque lento** (*Slow star)*

Cada vez que se transmite una ventana con éxito, la ventan se dobla. El *slow star* se mantiene hasta que se produce una pérdida (o bien se alcanza la ventana de control de flujo) haciendo entrar en congestión avoidance.

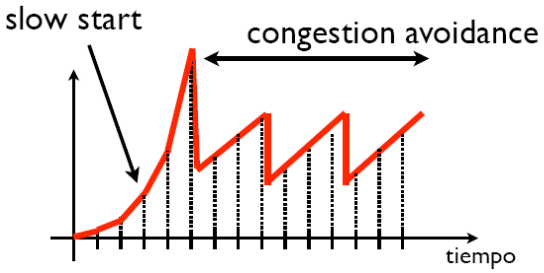


Figura 3.5 Arranque Lento (Slow Start)

La **Figura 3.5** nos muestra como funciona el Arranque lento y los efectos que se originan al producirse una perdida llevando la transmisión a una congestión Avoidance.

La pérdida se puede detectar por dos casos:

α. **Por ACKs duplicados**: en este caso la congestión es "ligera", hay pérdidas pero siguen llegando paquetes. Las acciones son: retransmitir el paquete perdido (Fast retransmit), bajar la ventana de congestión para reducir la tasa y pasar a congestión avoidance (si no se estaba).

β. **Por un timeout**: la congestión es "grave", probablemente se ha perdido toda la ventana. Si el timeout ha caducado, se lleva un rato parado sin transmitir. La tasa es igual a cero (0). Las acciones a seguir son: poner la ventana de congestión a I MSS, pasar a show Stara y recordar a cuanto estaba la ventana de congestión al producirse el error (poner la variable threshold=CongWin/2).

Después de una pérdida por timeout el show Stara no espera a otra pérdida para entrar en congestión avoidance. Pasa a congestión avoidance en cuanto llega al umbral de la mitad de la ventana de congestión al producirse el timeout.

- **Retransmisión rápida** (*Fast retransmit)*

La retransmisión rápida es una heurística que algunas veces dispara la retransmisión de paquetes perdidos en forma más rápida que el mecanismo de timeout regular. La retransmisión rápida no reemplaza los timeout regulares, sólo mejora su aplicabilidad.

El mecanismo es fácil, cada vez que llega un paquete al receptor, éste responde con un ACK, aún cuando el número de secuencia ha sido reconocido. Entonces, cuando un paquete llega fuera de orden, es decir, TCP no puede reconocer el dato que contiene el paquete porque datos anteriores aún no han llegado, TCP reenvía el mismo ACK que envió la última vez. Esta segunda transmisión del mismo ACK se denomina *ACK duplicado.* Cuando el lado transmisor ve el ACK duplicado, sabe que el otro lado ha recibido un paquete fuera de orden, lo que le sugiere que un paquete enviado con anterioridad se perdió. Como también es probable que ese paquete no se haya perdido sino esté atrasado, el transmisor espera la llegada de otros ACK duplicados antes de retransmitir el paquete perdido.

- **Recuperación rápida** (*Fast recovery)*

La recuperación rápida es un algoritmo implementado en las comunicaciones mediante el protocolo TCP. Cuando el receptor recibe un segmento con un número de secuencia que no corresponde, intuye que un segmento se ha perdido. Por cada nuevo segmento que llega, vuelve a mandar el ACK correspondiente al último segmento que recibió correctamente.

El emisor detecta que hay un problema de congestión ya que no recibe ACK del segmento que envió y además recibe ACKs repetidos del último segmento que llego correctamente. Cuando el emisor observa que le llega 3 ACKs repetidos, procede a reenviar el segmento siguiente al último asentido, y continuar con la secuencia que llevaba originalmente.

Si el receptor recibe adecuadamente el segmento reenviado, enviará un ACK correspondiente al último segmento recibido correctamente (que no tiene por qué coincidir con el segmento reenviado).

FUNDAMENTOS DE OPNET IT Guru Academic Edition

4. OPNET IT Guru Academia Edition

OPNET es un lenguaje de simulación orientado a las comunicaciones. Proporciona acceso directo al código fuente siendo esto una gran ventaja para los nuevos programadores que se aventuren a programar con OPNET.

Actualmente es utilizado por grandes empresas de telecomunicaciones, por ejemplo para desarrollar proyectos gubernamentales y del ejército, etc.

OPNET es un simulador que posee una interfaz muy seductora para los usuarios. Esto es debido a que incluye varas librerías de modelos. El código fuente de estas librerías es accesible y esto consigue que el programador pueda familiarizarse más rápidamente con toda la jerarquía interna del programa.

Para ser utilizado, primero el usuario tiene que comprender la jerarquía que se utiliza para poder plantear las simulaciones. Esta jerarquía se muestra en la **figura 4.1**

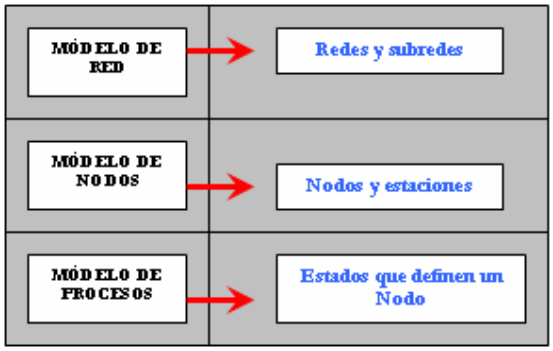


Figura 4.1 Jerarquía de diseño en OPNET

Como podemos observar en la **figura 4.2**, tenemos un modelo de res donde irán definidas las redes y subredes de la simulación. Seguidamente disponemos de un modelo de nodos

donde se define la estructura interna de éstos y por último tenemos el modelo de procesos donde se define los estados que definen un nodo.

En la correcta simulación de un proyecto se debe haber entendido los aspectos anteriores; puestos que de lo contrario la simulación saldría errónea y no serviría.

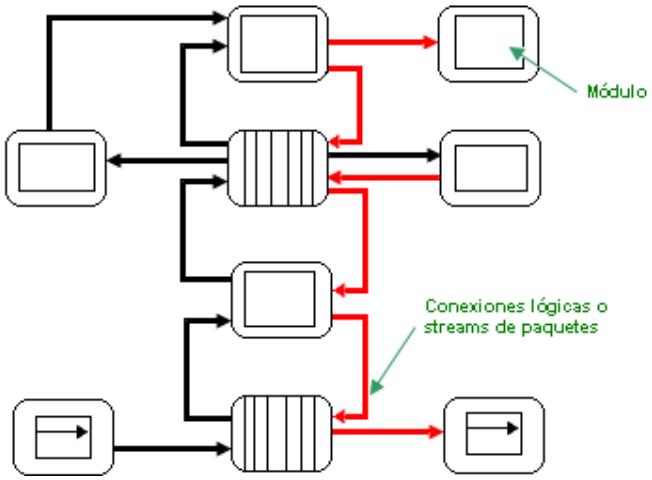


Figura 4.2 Funcionamiento del modelo de nodos

A continuación se explican algunos de las diferentes partes de que consta el OPNET, las más frecuentes en utilizar. Los editores incluidos proporcionan las herramientas necesarias para la creación de topologías e red. Cada editor se encarga de una tarea distinta, inmediatamente se explica cada uno de ellos.

- **Proyect Editor**

El Proyect editor es el principal escenario en la creación del entorno de la simulación de la red. Es usado para crear un modelo de red utilizando unos ya existentes que podemos encontrar en la librería estándar, recolectar estadísticas sobre la red, comenzar la simulación y observar los resultados. También podemos crear nodos, construir formatos de paquetes, etc.

Este editor contiene tres tipos básicos de objetos: subredes, nodos y enlaces.

Las paletas (accesibles mediante un icono situado en la parte superior izquierda del editor) ordenan los objetos disponibles en categorías. Por ejemplo, en la paleta ethernet, se encuentran los nodos y enlaces más utilizados para el diseño de este tipo de red.

- **Node Editor**

El Node Editor es un editor que es usado para crear modelos de nodos especificando su estructura interna. Estos modelos son usados para crear nodos en el interior de la red en el Proyect Editor.

Internamente, los modelos de nodos tienen una estructura modular, al ser definido un nodo como la conexión entre varios módulos con paquetes de streams y cables. Esta conexión permite intercambiar información y paquetes entre ellos. Cada módulo tiene una función específica dentro del nodo, tal como: generar paquetes, encolarlos, procesarlos o transmitirlos y recibirlos.

- **Process Model Editor**

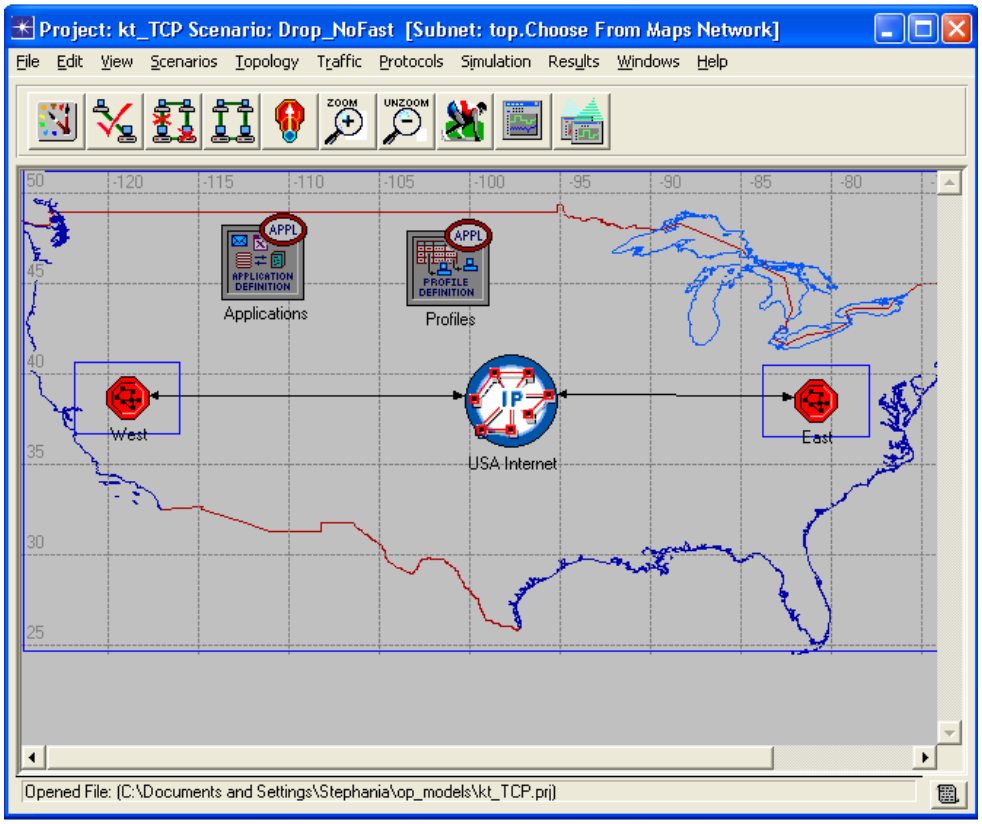
Se utiliza en la creación de modelos de procesos, que a su vez controlan los modelos de nodo creados en el Node Editor los Process Model son representados por estados (FSM), y son creados por iconos que representan estos estados y por líneas que representan las transiciones entre ellos.

- **Link Model Editor**

Este editor ofrece la posibilidad de crear nuevos tipos de objetos link. Cada nuevo tipo de link puede tener diferentes atributos y representaciones.

- **Path Editor**

Es usado para crear nuevos objetos path que sirven para definir un traficc route.



TUTORIAL:

CONGESTION EN TCP

5. TUTORIAL DE

CONGESTIÓN EN TCP

TCP: Control de Congestión

I. Objetivo

Este tutorial se ha diseñado para mostrar los algoritmos de control de congestión implementados en TCP. En el desarrollo del mismo se proporcionan distintos escenarios para simular estos algoritmos. Podremos comparar las prestaciones de los algoritmos a través del análisis de los resultados de las simulaciones.

II. Descripción

El protocolo TCP para Internet garantiza la entrega fiable y en orden de un flujo de bytes (*byte stream*). Incluye un mecanismo de control de flujo del stream de bytes que permite al receptor limitar la cantidad de datos que el emisor puede transmitir en un instante determinado. Además, TCP implementa un mecanismo de control de la congestión altamente sintonizable. La idea de este segundo mecanismo es regular la velocidad de envío de datos por parte de TCP, con el fin de evitar que el emisor sobrecargue la red.

En el control de la congestión realizado por TCP, cada fuente determina cual es la capacidad disponible en la red, con el fin de conocer cuantos paquetes puede tener en tránsito de una manera segura. Para esto, TCP mantiene una variable de estado distinta por cada conexión, que se denomina *ventana de congestión*. El host fuente utilizará esta variable para limitar la cantidad de datos que puede tener en tránsito en un momento determinado. TCP utiliza un mecanismo, llamado "incremento aditivo"/"decremento multiplicativo" (*additive increase/ multiplicative decrease*), que reduce la ventana de congestión cuando el nivel de congestión aumenta, y que incrementa la ventana de congestión cuando el nivel de congestión disminuye. TCP interpreta los *timeouts* como signo de congestión. Cada vez que ocurre un *timeout*, el host origen pone su ventana de congestión a la mitad del valor que tenía previamente. Esta división se corresponde con la parte de "decremento

multiplicativo" de este mecanismo. La ventana de congestión no puede caer debajo del tamaño de un único segmento TCP, dado por el parámetro de TCP "tamaño máximo de segmento" (más conocido por su nombre en inglés *Maximum Segment Size* o *MSS*). Cada vez que un host origen envía con éxito todos los paquetes que caben en la ventana de congestión, se incremente la ventana de congestión en el tamaño equivalente a un paquete. Esta sería la parte de "incremento aditivo" del mecanismo (también llamado algoritmo de Van Jacobson).

En este tutorial, vamos a comenzar con ejemplos sobre el principio lento y fases de anulación de congestión. También ilustramos el efecto de Paquetes únicos de descenso (*Single Packet Drop*) sobre ventanas de congestión usando varios grupos TCP: Tahoe, Reno con SACK y Escalamiento de Ventana (*Window Scaling*). Los dos últimos argumentos (guiones) muestran el efecto de usar la opción de *Timestamp* para calcular el tiempo de viaje de ida y vuelta del segmento.

III. Pasos

Descripción de Escenarios

1. **Arranque Lento y Prevención de Congestión (Slow Start & Congestion Avoidance):** Describe las dos fases de crecimiento de la ventana de congestión.
2. **Por defecto de una caída (Default with one drop):** Demuestra el excesivo tiempo de recuperación adoptadas por TCP en un solo paquete de caída.
3. **Tahoe con una caída (Tahoe with one drop):** Muestra la mejora en el tiempo de recuperación debido a la utilización de *Fast Retramsmit*.
4. **Reno con una caída (Reno with one drop):** Muestra nuevas mejoras en el tiempo de recuperación debido a la utilización de *Fast Retransmit & Fast Recovery*.
5. **SACK con una caída (SACK with one drop):** Demuestra la mejor recuperación de los TCP.

6. **Escalamiento de Ventana con una caída (Window Scaling with one drop):** Muestra cómo la ventana de ampliación puede aumentar para el tiempo de recuperación, incluso con SACK TCP activas.
7. **Ida y vuelta calculando el tiempo sin *Time Stamp* (Round trip time calculation without Time Stamp):** Muestra el efecto de la serie de sesiones de ida y vuelta a tiempo. Método de cálculo relativo a la aplicación del tiempo de respuesta.
8. **Ida y vuelta calculando el tiempo con *Time Stamp* (Round trip time calculation with Time Stamp):** Muestra el efecto de la serie de sesiones de ida y vuelta a tiempo. Método de cálculo relativo a la aplicación del tiempo de respuesta.

IV. *Creación de Escenarios*

Escenario 1: Arranque Lento y Prevención de Congestión.

Para crear un proyecto nuevo:

1. Ejecutar **OPNET IT Guru Academic Edition** → Seleccionar **New** del menu **File**.
2. Seleccionar **Project** → Clic en **OK** → Nombre del proyecto <Sus_iniciales>**_TCP** y el escenario **show_start_and_congestion_avoidance** → Clic en **OK**.
3. En el *Startup Wizard:* En la ventana de diálogo *Inicial Topology*, Asegurarsé de que **Create Empty Scenario** esté seleccionado → Clic en **Next** → Elegir **Office** de la lista *Network Scale* → Clic en **Next** → Mantener **X Span** y **Y Span** en 100 → Clic en **Next** dos veces → Clic en **Ok**.

4. De la ventana de diálogo **Object Palette** arrastrar hasta el panel los siguientes elementos:

Figura 5.3.1.1 Elementos de la Paleta de Objetos

5. En la misma ventana de diálogo configurar la paleta a **links_ advanced** y seleccionar el siguiente elemento:

Figura 5.3.1.2 Elemento de la Paleta de Objetos para unión

6. Cerrar la paleta.
7. Renombrar los objetos que se han añadido como se muestra en la siguiente figura, y después guardar el proyecto.

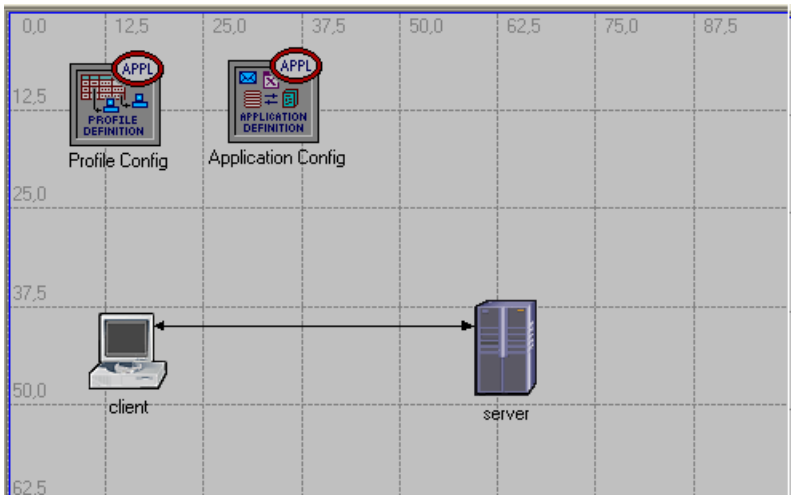


Figura 5.3.1.3 Escenario Arranque Lento y Prevención de Congestión

Configurar las aplicaciones:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Applications Config**→ Edit Attributes→ expandir el atributo **Application Definitions** y poner **rows** a 1→ expandir la nueva fila y llamar a esa fila **File Transfer (custom).**
2. Hacer clic en OK y guardar el proyecto.

Configurar de perfiles:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Profiles Config** → Edit Attributes → expandir el atributo **Profile Configuration** y poner **rows** a 1.
   a) Para **row** 0, llamar y poner los atributos con los valores que se muestran en la siguiente figura → Hacer clic en OK.

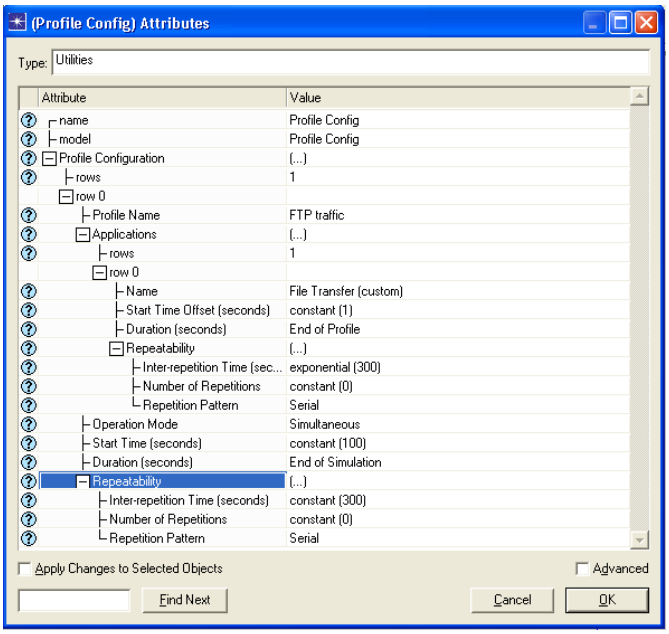


Figura 5.3.1.4 Configuración de Atributos de Perfiles

Con esto se han configurado las aplicaciones y los perfiles del TCP.

2. Hacer clic en el nodo **Client** → Edit Attributes:

   a) Expandir el árbol **Application: supported Profiles** → poner **rows** a 1 → expandir el árbol **row 0** → poner **Profile Name** a **FTP traficc.**

3. Hacer clic en el nodo **Server** → Edit Attributes como se muestra en la figura:

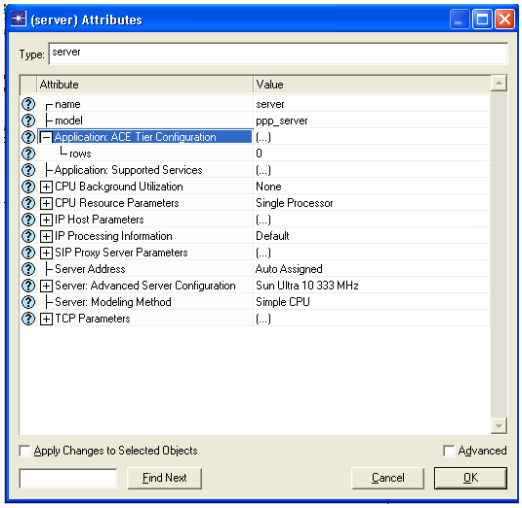


Figura 5.3.1.5 Configuración de Atributos del Servidor

4. Hacer clic en OK y guardar el proyecto.

Al realizar una simulación del escenario las gráficas arrojadas muestran el crecimiento de la ventana de congestión de en dos modos:

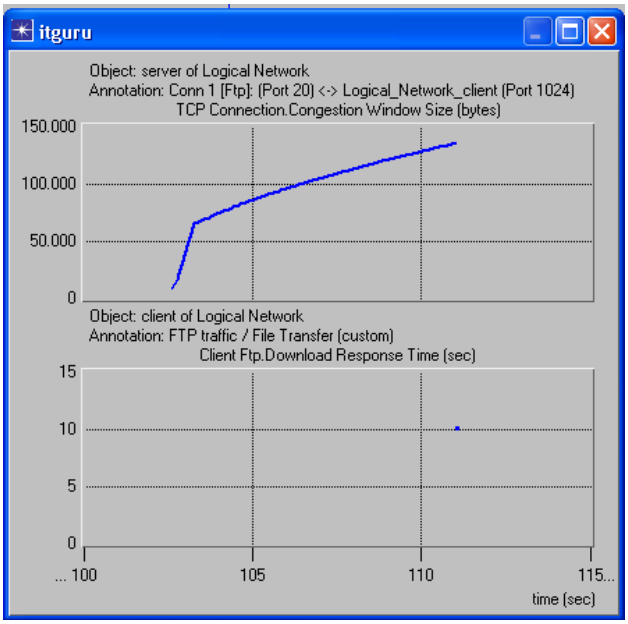


Figura 5.3.1.6 Gráfica de Estadísticas

El primero nos muestra el Arranque Lento en crecimiento exponencial, mientras que el segundo nos muestra la Anulación de Congestión con un crecimiento lineal.

Como se puede ver, el Arranque lento lo que hace es doblar la cantidad de datos en cada transmisión hasta que una de ellas vence el temporizador. El último dato enviado es que el lleva el tamaño de la ventana.

La Prevención de congestión permite que cada vez que se envía un paquete con éxito, la ventana sube linealmente hasta encontrar el punto de equilibrio sin aumentar demasiado la congestión.

Escenario 2: Por defecto de una caída.

Para crear un proyecto nuevo:

1. Ejecutar **OPNET IT Guru Academic Edition** → Seleccionar **New** del menu **File**.

2. Seleccionar **Project** → Clic en **OK** → Nombre del proyecto <Sus_iniciales>**_TCP** y el escenario **default_with_one_drop** → Clic en **OK**.
3. En el *Startup Wizard:* En la ventana de diálogo *Inicial Topology*, Asegurarse de que **Create Empty Scenario** esté seleccionado → Clic en **Next** → Elegir **Office** de la lista *Network Scale* → Clic en **Next** → Mantener **X Span** y **Y Span** en 100 → Clic en **Next** dos veces → Clic en **Ok**.
4. De la ventana de diálogo **Object Palette** arrastrar hasta el panel los siguientes elementos:

Figura 5.3.2.1 Elementos de la Paleta de Objetos

5. En la misma ventana de diálogo configurar la paleta a **links_ advanced** y seleccionar el siguiente elemento:

Figura 5.3.2.2 Elemento de la Paleta de Objetos para unión

6. Cerrar la paleta.

7. Renombrar los objetos que se han añadido como se muestra en la siguiente figura, y después guardar el proyecto.

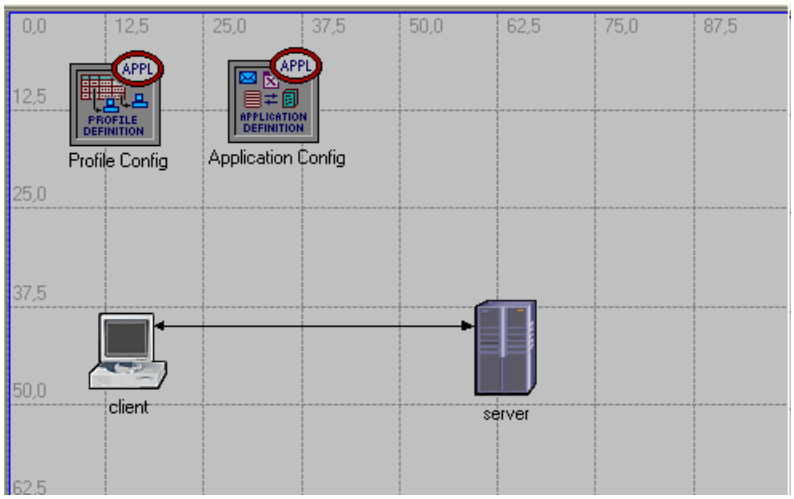


Figura 5.3.2.3 Escenario Defecto de caída

Configurar las aplicaciones:

3. Hacer clic con el botón derecho del ratón sobre el nodo **Applications Config**→ Edit Attributes→ expandir el atributo **Application Definitions** y poner **rows** a 1→ expandir la nueva fila y llamar a esa fila **File Transfer (custom).**
4. Hacer clic en OK y guardar el proyecto.

Configurar de perfiles:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Profiles Config** → Edit Attributes → expandir el atributo **Profile Configuration** y poner **rows** a 1.
   a) Para **row** 0, llamar y poner los atributos con los valores que se muestran en la siguiente figura → Hacer clic en OK.

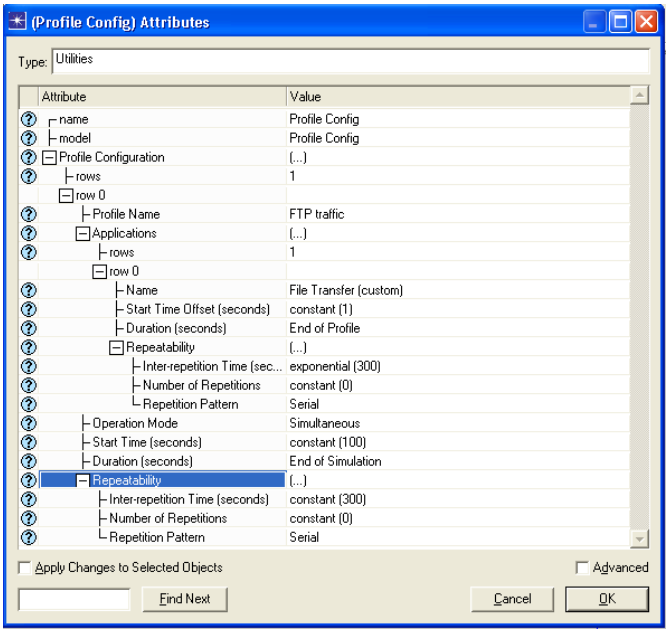


Figura 5.3.2.4 Configuración de Atributos de Perfiles

Con esto se han configurado las aplicaciones y los perfiles del TCP.

2. Hacer clic en el nodo **Client** → Edit Attributes:

   a) Expandir el árbol **Application: supported Profiles** → poner **rows** a 1 → expandir el árbol **row 0** → poner **Profile Name** a **FTP traficc.**

3. Hacer clic en el nodo **Server** → Edit Attributes como se muestra en la figura:

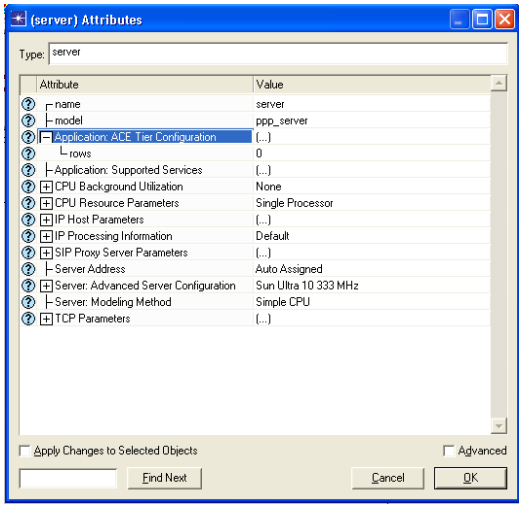


Figura 5.3.2.5 Configuración de Atributos del Servidor

4. Hacer clic en OK y guardar el proyecto.

Al realizar una simulación del escenario y observar los resultados de las gráficas arrojadas obtenemos:

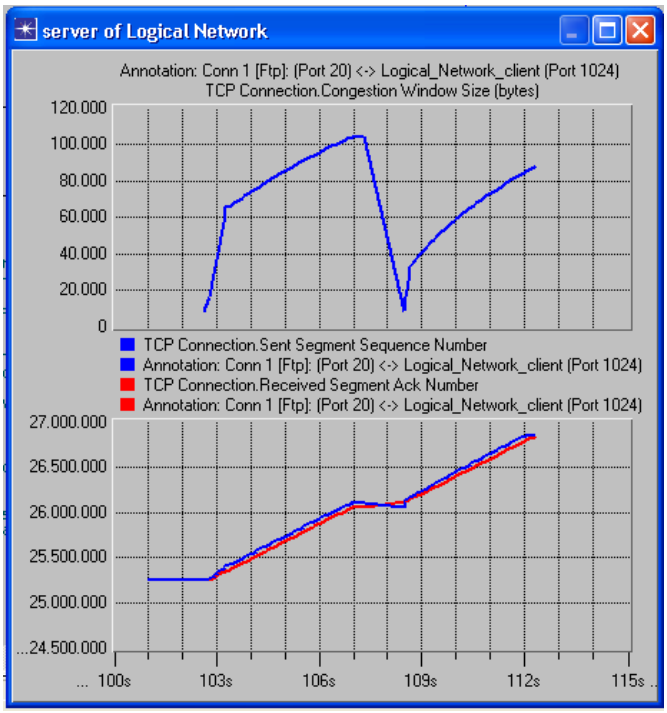


Figura 5.3.2.6 Gráfica de Estadísticas

La gráfica superior nos muestra el efecto de una trama del paquete solo sobre la ventana de congestión. La ventana de congestión se cae a 1 MSS sobre la nueva transmisión y crece otra vez.

La gráfica inferior nos muestra la relación entre el número de secuencia del segmento enviado y el número del segmento recibido sobre el servidor. Debido a una trama del paquete en el tiempo 107segundos, el segmento que se envía es parado hasta que el temporizador de la nueva transmisión expire para el segmento perdido.

Escenario 3: Tahoe con una caída

Para crear un proyecto nuevo:

1. Ejecutar **OPNET IT Guru Academic Edition** → Seleccionar **New** del menu **File**.
2. Seleccionar **Project** → Clic en **OK** → Nombre del proyecto <Sus_iniciales>**_TCP** y el escenario **tahoe_with_one_drop**→ Clic en **OK**.
3. En el *Startup Wizard:* En la ventana de diálogo *Inicial Topology*, Asegurarse de que **Create Empty Scenario** esté seleccionado → Clic en **Next** → Elegir **Office** de la lista *Network Scale* → Clic en **Next** → Mantener **X Span** y **Y Span** en 100 → Clic en **Next** dos veces → Clic en **Ok**.
4. De la ventana de diálogo **Object Palette** arrastrar hasta el panel los siguientes elementos:

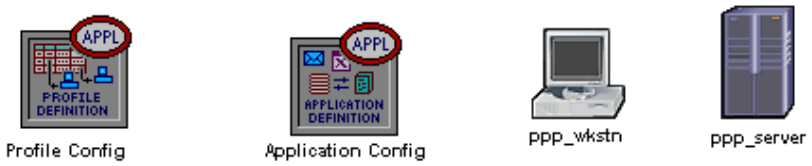


Figura 5.3.3.1 Elementos de la Paleta de Objetos

5. En la misma ventana de diálogo configurar la paleta a **utilities** y seleccionar el siguiente elemento:

Figura 5.3.3.2 Elemento de la Paleta de utilidades

6. En la misma ventana de diálogo configurar la paleta a **links_ advanced** y seleccionar el siguiente elemento:

Figura 5.3.3.3 Elemento de Unión

7. Cerrar la paleta.
8. Renombrar los objetos que se han añadido como se muestra en la siguiente figura, y después guardar el proyecto.

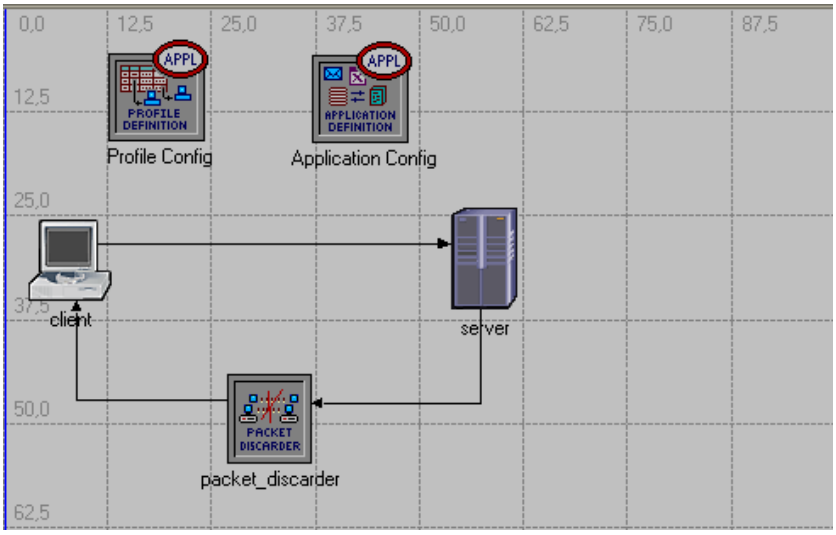


Figura 5.3.3.4 Escenario Tahoe con caída

Configurar las aplicaciones:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Applications Config**→ Edit Attributes→ expandir el atributo **Application Definitions** y poner **rows** a 1→ expandir la nueva fila y llamar a esa fila **File Transfer (custom).**
2. Hacer clic en OK y guardar el proyecto.

Configurar de perfiles:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Profiles Config** → Edit Attributes → expandir el atributo **Profile Configuration** y poner **rows** a 1.
   - Para **row** 0, llamar y poner los atributos con los valores que se muestran en la siguiente figura → Hacer clic en OK.

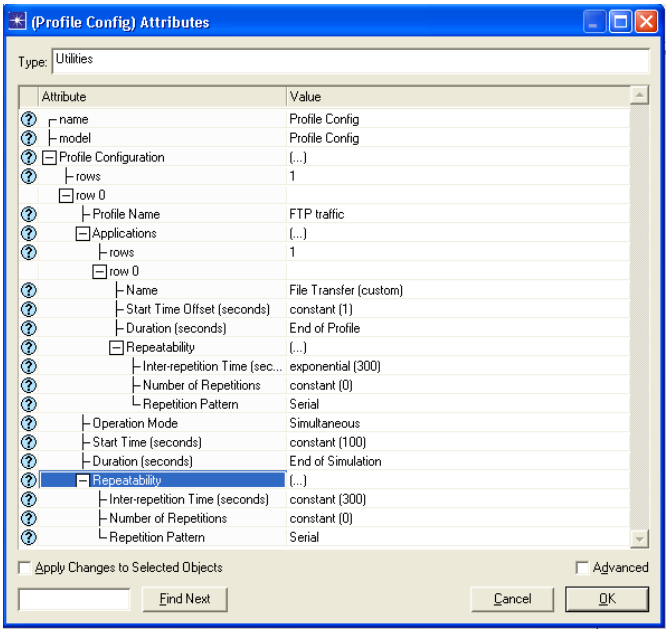


Figura 5.3.3.5 Configuración de Perfil

Con esto se han configurado las aplicaciones y los perfiles del TCP.

2. Hacer clic en el nodo **Client** → Edit Attributes:
   - Expandir el árbol **Application: supported Profiles** → poner **rows** a 1 → expandir el árbol **row 0** → poner **Profile Name** a **FTP traficc.**
3. Hacer clic en el nodo **Server** → Edit Attributes como se muestra en la figura:

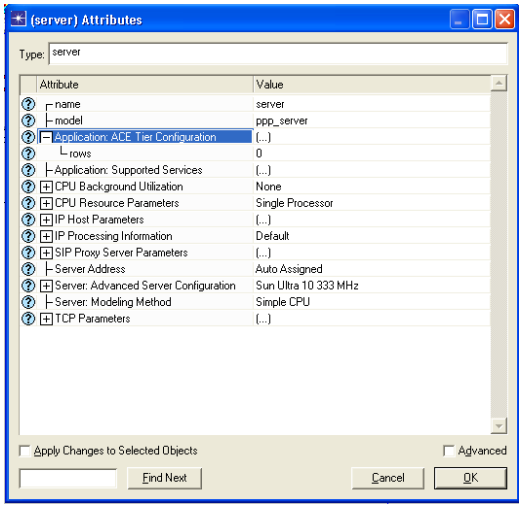


Figura 5.3.3.6 Configuración del Servidor

4. Hacer clic en OK y guardar el proyecto.

5. Hacer clic en el nodo **packet_discarder** → Edit Attributes como se muestra en la figura:

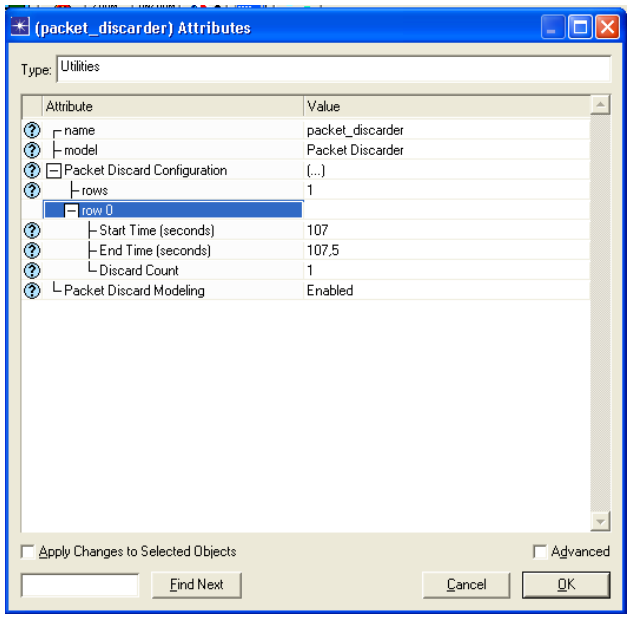


Figura 5.3.3.7 Configuración del packet discarder

6. Hacer clic en OK y guardar el proyecto.

Al realizar una simulación del escenario y observar los resultados de las gráficas arrojadas obtenemos:

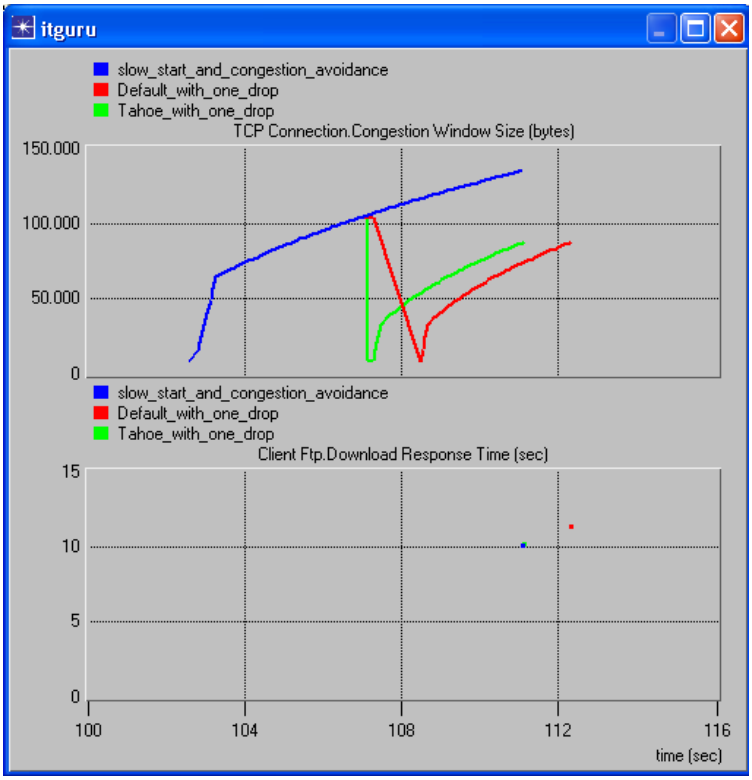


Figura 5.3.3.8 Análisis de Estadísticas

El tráfico en la gráfica superior se compara con la Ventana de Congestión para ninguna trama de paquete y para una trama de paquete, con los argumentos TAHOE y DEFAULT. La gráfica inferior es la misma comparación pero en forma lineal.

Tahoe es implementado en versiones para comienzo rápido y evitación de cogestión, los cuales ya se analizaron anteriormente, y para *fast retransmit*, donde el comienzo rápido y la evitación de congestión incrementan el tamaño de la ventana; mientras que *fast retransmit* detecta la congestión.

Escenario 4: Reno con una caída

Para crear un proyecto nuevo:

1. Ejecutar **OPNET IT Guru Academic Edition** → Seleccionar **New** del menu **File**.
2. Seleccionar **Project** → Clic en **OK** → Nombre del proyecto <Sus_iniciales>**_TCP** y el escenario **reno_with_one_drop**→ Clic en **OK**.
3. En el *Startup Wizard:* En la ventana de diálogo *Inicial Topology*, Asegurarse de que **Create Empty Scenario** esté seleccionado → Clic en **Next** → Elegir **Office** de la lista *Network Scale* → Clic en **Next** → Mantener **X Span** y **Y Span** en 100 → Clic en **Next** dos veces → Clic en **Ok**.
4. De la ventana de diálogo **Object Palette** arrastrar hasta el panel los siguientes elementos:

Figura 5.3.4.1 Elementos de la Paleta de Objetos

5. En la misma ventana de diálogo configurar la paleta a **utilities** y seleccionar el siguiente elemento:

Figura 5.3.4.2 Elemento de la Paleta packet discader

6. En la misma ventana de diálogo configurar la paleta a **links_ advanced** y seleccionar el siguiente elemento:

Figura 5.3.4.3 elemento de unión

7. Cerrar la paleta.
8. Renombrar los objetos que se han añadido como se muestra en la siguiente figura, y después guardar el proyecto.

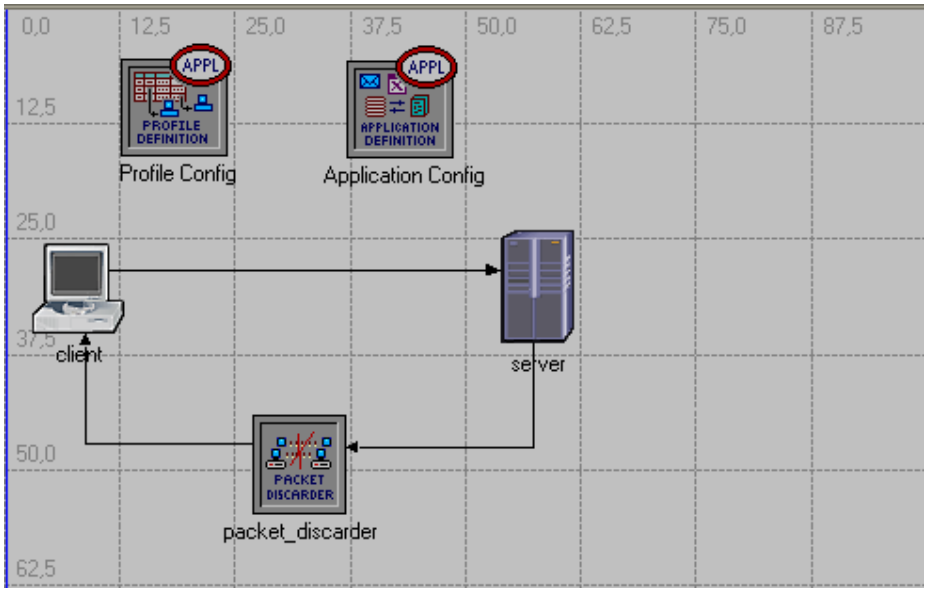


Figura 5.3.4.4 Escenario de Reno por una caída

Configurar las aplicaciones:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Applications Config**→ Edit Attributes→ expandir el atributo **Application Definitions** y poner **rows** a 1→ expandir la nueva fila y llamar a esa fila **File Transfer (custom).**
2. Hacer clic en OK y guardar el proyecto.

Configurar de perfiles:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Profiles Config** → Edit Attributes → expandir el atributo **Porfile Configuration** y poner **rows** a 1.

   - Para **row** 0, llamar y poner los atributos con los valores que se muestran en la siguiente figura → Hacer clic en OK.

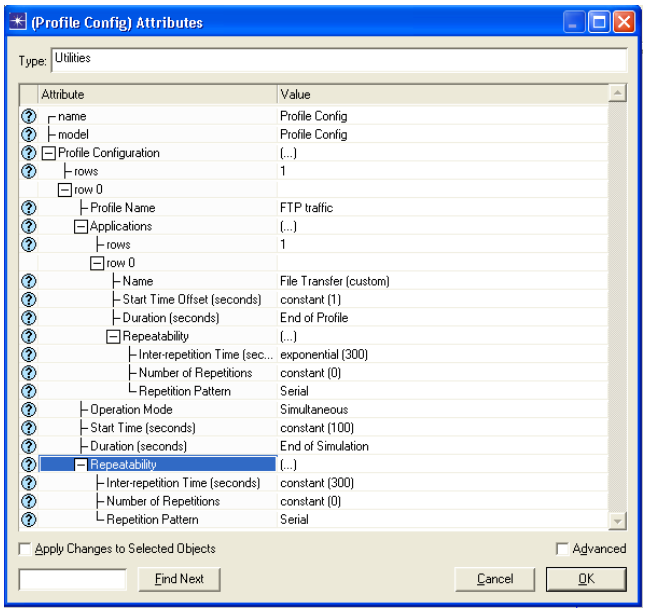


Figura 5.3.4.5 Configuración de Perfil

Con esto se han configurado las aplicaciones y los perfiles del TCP.

1. Hacer clic en el nodo **Client** → Edit Attributes:

   - Expandir el árbol **Application: supported Profiles** → poner **rows** a 1 → expandir el árbol **row 0** → poner **Profile Name** a **FTP traficc.**

2. Hacer clic en el nodo **Server** → Edit Attributes como se muestra en la figura:

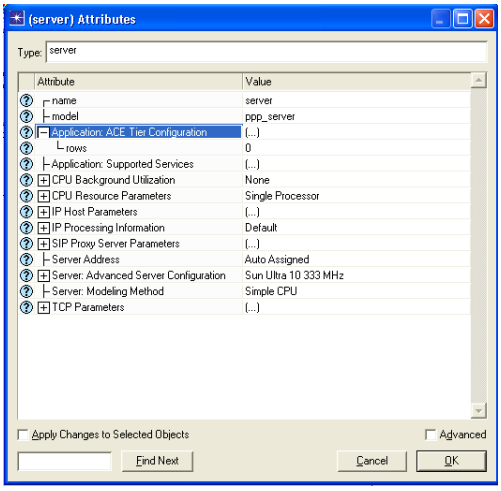


Figura 5.3.4.6 Configuración del Servidor

3. Hacer clic en OK y guardar el proyecto.

4. Hacer clic en el nodo **packet_discarder** → Edit Attributes como se muestra en la figura:

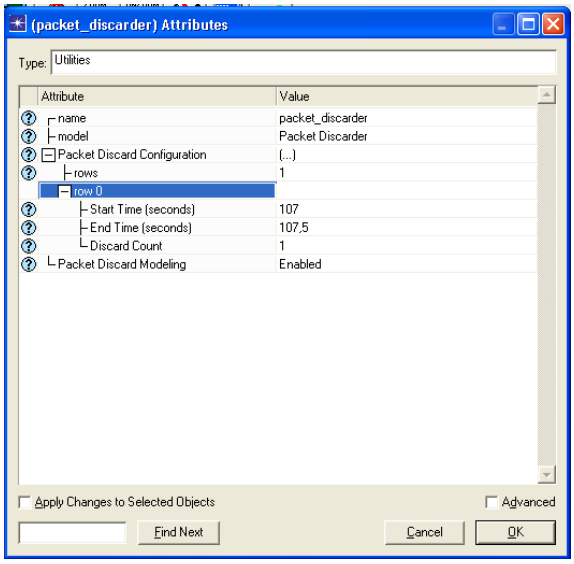


Figura 5.3.4.7 Configuración de packet discarder

5. Hacer clic en OK y guardar el proyecto.

Al realizar una simulación del escenario y observar los resultados de las gráficas arrojadas obtenemos:

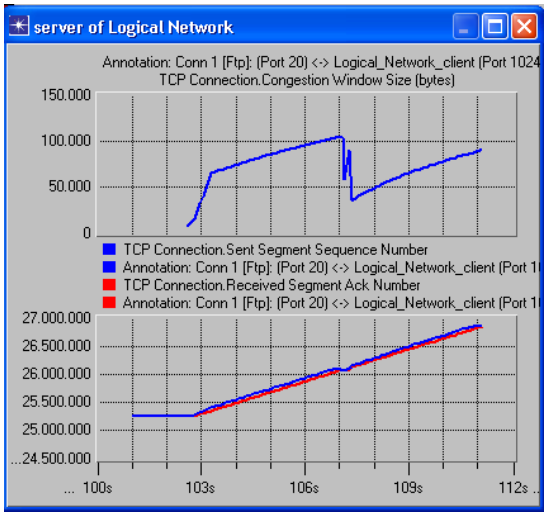


Figura 5.3.4.8 Análisis de Estadísticas

La gráfica superior muestra la Recuperación Rápida de la ventana de congestión debido a Reno TCP. Aquí sobre nuevas transmisiones, solamente se reduce la ventana de congestión a la mitad en ves de 1 MSS, causando la recuperación rápida.

La gráfica inferior muestra la relación entre el número de secuencia del segmento enviado y el número de secuencia del segmento recibido, sobre el servidor.

Incluso aunque haya una caída de paquetes en el tiempo 107segundos, el segmento que envía son hincados atrás, después del encubrimiento de los tres ACK's dobles que demuestran la recuperación rápida.

Esta implementación previene que se vacié el *pipe* después del *Fast Retransmit*, por lo que evita la necesidad de llenarlo con algoritmo *Slow-Start* luego de la pérdida de un solo paquete.

Reno mejora significativamente con respecto a Tahoe cuando un paquete de datos e pierde de una ventana de datos, pero sufre problemas de rendimiento cuando múltiples paquetes se pierden de una ventana de datos.

Escenario 5: SACK con una caída

Para crear un proyecto nuevo:

1. Ejecutar **OPNET IT Guru Academic Edition** → Seleccionar **New** del menu **File**.
2. Seleccionar **Project** → Clic en **OK** → Nombre del proyecto <Sus_iniciales>**_TCP** y el escenario **sack_with_one_drop**→ Clic en **OK**.
3. En el *Startup Wizard:* En la ventana de diálogo *Inicial Topology*, Asegurarse de que **Create Empty Scenario** esté seleccionado → Clic en **Next** → Elegir **Office** de la lista *Network Scale* → Clic en **Next** → Mantener **X Span** y **Y Span** en 100 → Clic en **Next** dos veces → Clic en **Ok**.
4. De la ventana de diálogo **Object Palette** arrastrar hasta el panel los siguientes elementos:

Figura 5.3.5.1 Elementos de la Paleta de Objetos

5. En la misma ventana de diálogo configurar la paleta a **utilities** y seleccionar el siguiente elemento:

Figura 5.3.5.2 Elemento de la Paleta packet discader

6. En la misma ventana de diálogo configurar la paleta a **links_ advanced** y seleccionar el siguiente elemento:

Figura 5.3.5.3 elemento de unión

7. Cerrar la paleta.
8. Renombrar los objetos que se han añadido como se muestra en la siguiente figura, y después guardar el proyecto.

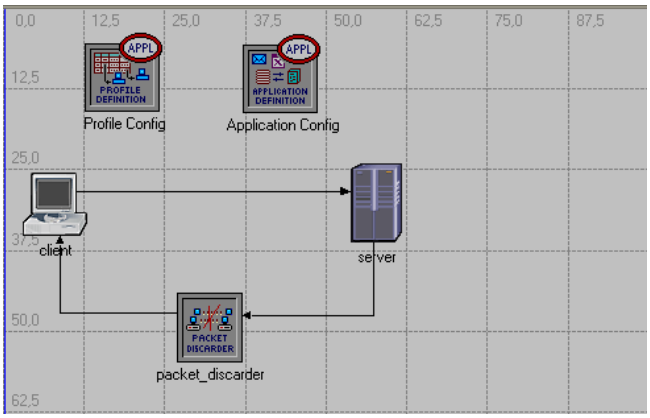


Figura 5.3.5.4 Escenario de SACK con una caída

Configurar las aplicaciones:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Applications Config**→ Edit Attributes→ expandir el atributo **Application Definitions** y poner **rows** a 1→ expandir la nueva fila y llamar a esa fila **File Transfer (custom).**
2. Hacer clic en OK y guardar el proyecto.

Configurar de perfiles:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Profiles Config** → Edit Attributes → expandir el atributo **Porfile Configuration** y poner **rows** a 1.

   - Para **row** 0, llamar y poner los atributos con los valores que se muestran en la siguiente figura → Hacer clic en OK.

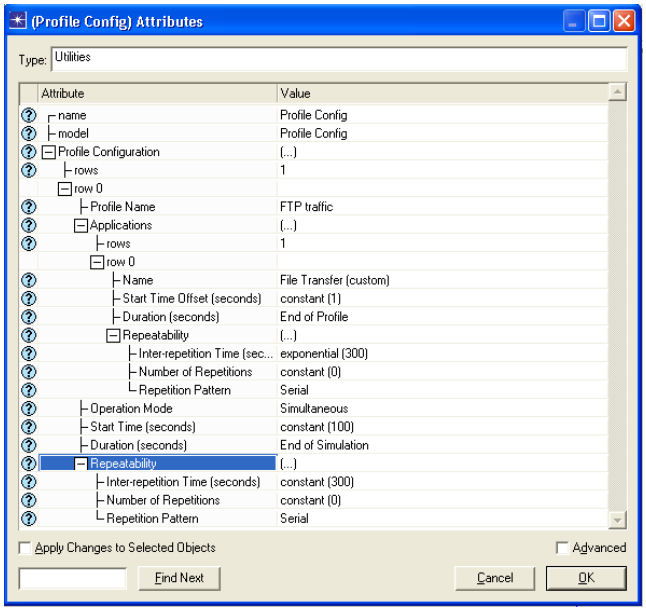


Figura 5.3.5.5 Configuración de Perfil

Con esto se han configurado las aplicaciones y los perfiles del TCP.

1. Hacer clic en el nodo **Client** → Edit Attributes:

   - Expandir el árbol **Application: supported Profiles** → poner **rows** a 1 → expandir el árbol **row 0** → poner **Profile Name** a **FTP traficc.**

2. Hacer clic en el nodo **Server** → Edit Attributes como se muestra en la figura:

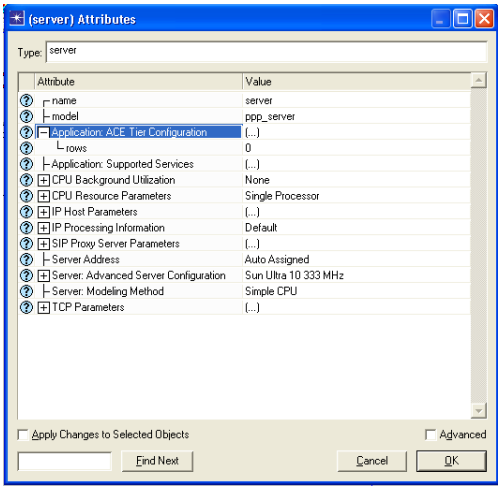


Figura 5.3.5.6 Configuración del Servidor

3. Hacer clic en OK y guardar el proyecto.
4. Hacer clic en el nodo **packet_discarder** → Edit Attributes como se muestra en la figura:

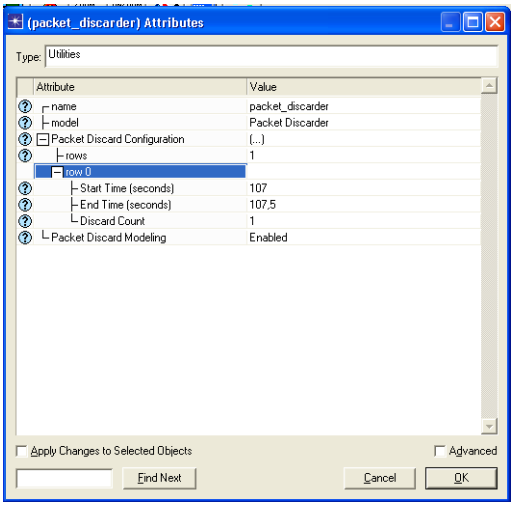


Figura 5.3.5.7 Configuración de packet discarder

5. Hacer clic en OK y guardar el proyecto.

Al realizar una simulación del escenario y observar los resultados de las gráficas arrojadas obtenemos:

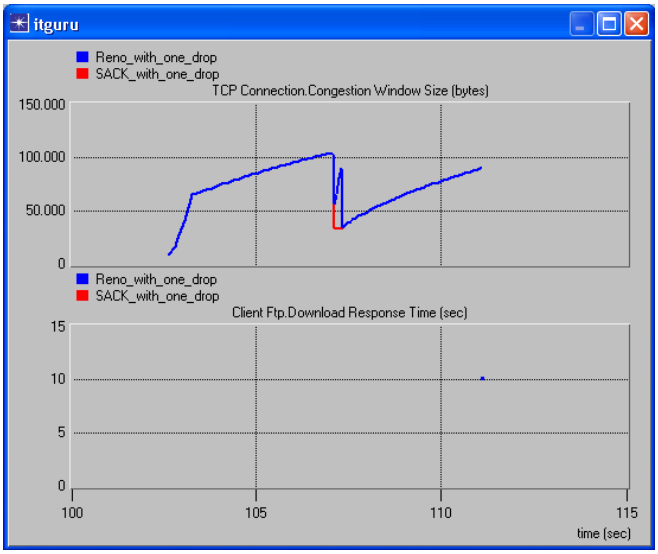


Figura 5.3.5.8 Análisis de Estadísticas

La gráfica muestra la recuperación ligeramente más rápida de la ventana de congestión debido a la opción SACK comparada con TCP Reno y TCP Tahoe. Esta diferencia será más sensible si hubiera múltiples caídas de paquetes.

En el caso de que al destino lleguen paquetes fuera de orden el receptor transmite ACKs duplicados reconociendo que se han recibido paquetes fuera de orden. El transmisor TCP reconstruye la información sobre los segmentos que no fueron recibidos en el destino.
El transmisor mantiene un estructura de datos, que recuerda los conocimientos previos lo cual es una opción SACK.

Escenario 6: Escalamiento de Ventana con una caída

Para crear un proyecto nuevo:

1. Ejecutar **OPNET IT Guru Academic Edition** → Seleccionar **New** del menu **File**.

2. Seleccionar **Project** → Clic en **OK** → Nombre del proyecto <Sus_iniciales>_**TCP** y el escenario **window_scaling_with_one_drop**→ Clic en **OK**.
3. En el *Startup Wizard:* En la ventana de diálogo *Inicial Topology*, Asegurarse de que **Create Empty Scenario** esté seleccionado → Clic en **Next** → Elegir **Office** de la lista *Network Scale* → Clic en **Next** → Mantener **X Span** y **Y Span** en 100 → Clic en **Next** dos veces → Clic en **Ok**.
4. De la ventana de diálogo **Object Palette** arrastrar hasta el panel los siguientes elementos:

Figura 5.3.6.1 Elementos de la Paleta de Objetos

5. En la misma ventana de diálogo configurar la paleta a **utilities** y seleccionar el siguiente elemento:

Figura 5.3.6.2 Elemento de la Paleta packet discader

6. En la misma ventana de diálogo configurar la paleta a **links_ advanced** y seleccionar el siguiente elemento:

Figura 5.3.6.3 elemento de unión

7. Cerrar la paleta.

8. Renombrar los objetos que se han añadido como se muestra en la siguiente figura, y después guardar el proyecto.

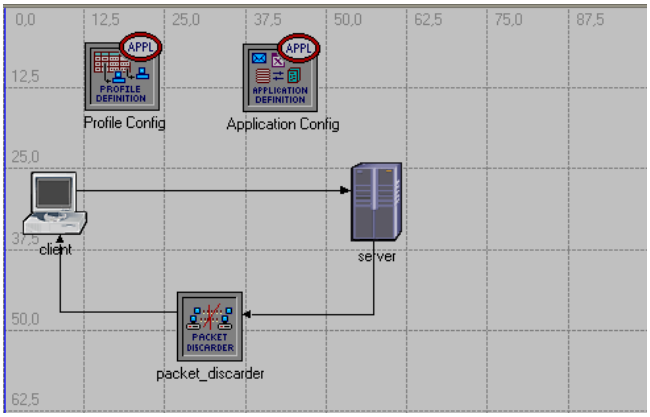


Figura 5.3.6.4 Escenario de Escalamiento de ventana con una caída

Configurar las aplicaciones:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Applications Config**→ Edit Attributes→ expandir el atributo **Application Definitions** y poner **rows** a 1→ expandir la nueva fila y llamar a esa fila **File Transfer (custom).**
2. Hacer clic en OK y guardar el proyecto.

Configurar de perfiles:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Profiles Config** → Edit Attributes → expandir el atributo **Porfile Configuration** y poner **rows** a 1.
   - Para **row** 0, llamar y poner los atributos con los valores que se muestran en la siguiente figura → Hacer clic en OK.

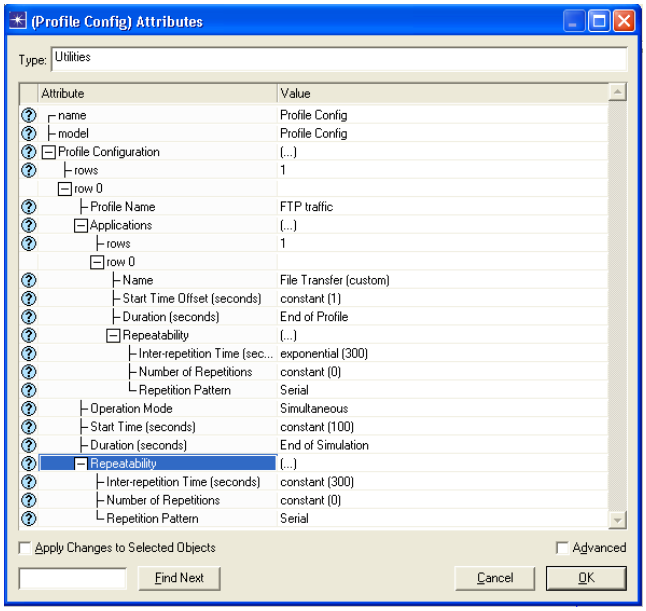


Figura 5.3.6.5 Configuración de Perfil

Con esto se han configurado las aplicaciones y los perfiles del TCP.

1. Hacer clic en el nodo **Client** → Edit Attributes:
   - Expandir el árbol **Application: supported Profiles** → poner **rows** a 1 → expandir el árbol **row 0** → poner **Profile Name** a **FTP traficc.**
2. Hacer clic en el nodo **Server** → Edit Attributes como se muestra en la figura:

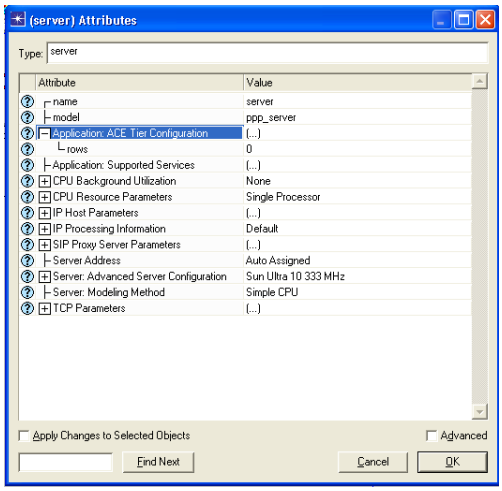


Figura 5.3.6.6 Configuración del Servidor

3. Hacer clic en OK y guardar el proyecto.
4. Hacer clic en el nodo **packet_discarder** → Edit Attributes como se muestra en la figura:

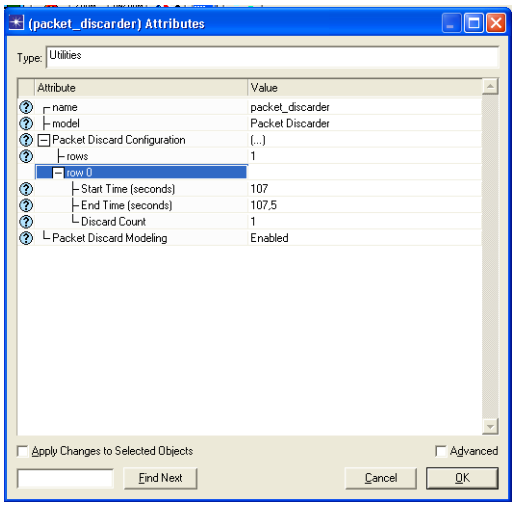


Figura 5.3.6.7 Configuración de packet discarder

5. Hacer clic en OK y guardar el proyecto.

Al realizar una simulación del escenario y observar los resultados de las gráficas arrojadas obtenemos:

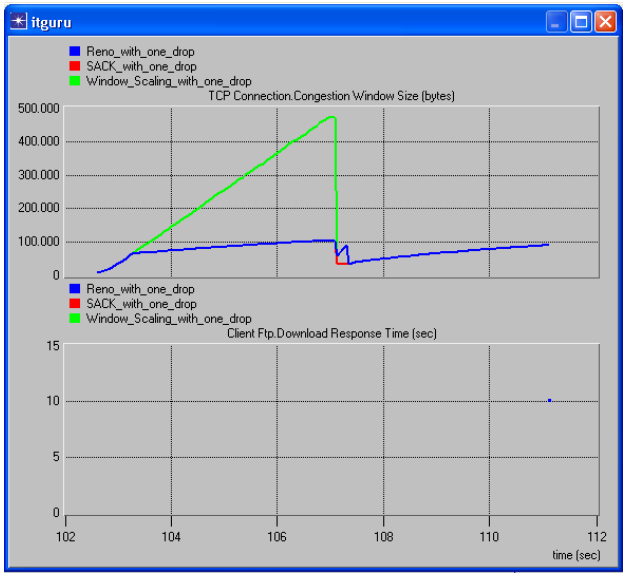


Figura 5.3.6.8 Análisis de Estadísticas

La gráfica compara la ventana de congestión y el tiempo de respuesta de la aplicación para la carrera de TCP Reno, TCP SACK y Escalamiento de Ventanas.

En esta opción se aumenta el tamaño de la ventana TCP, donde el rendimiento de la comunicación está limitada por dos ventanas: ventana de congestión, la cual trata de no superar la capacidad de la red, y recibir ventana, la cual trata de no superar la capacidad del receptor para procesar datos.

Esta opción es necesaria para una eficiente transferencia de datos cuando el ancho de banda de demora es superior a 64K.

Escenario 7: Ida y vuelta calculando el tiempo sin Time Stamp

Para crear un proyecto nuevo:

1. Ejecutar **OPNET IT Guru Academic Edition** → Seleccionar **New** del menu **File**.

2. Seleccionar **Project** → Clic en **OK** → Nombre del proyecto <Sus_iniciales>**_TCP** y el escenario **rtt_without_timestamps**→ Clic en **OK**.
3. En el *Startup Wizard:* En la ventana de diálogo *Inicial Topology*, Asegurarse de que **Create Empty Scenario** esté seleccionado → Clic en **Next** → Elegir **Office** de la lista *Network Scale* → Clic en **Next** → Mantener **X Span** y **Y Span** en 100 → Clic en **Next** dos veces → Clic en **Ok**.
4. De la ventana de diálogo **Object Palette** arrastrar hasta el panel los siguientes elementos:

Figura 5.3.7.1 Elementos de la Paleta de Objetos

5. En la misma ventana de diálogo configurar la paleta a **links_ advanced** y seleccionar el siguiente elemento:

Figura 5.3.7.2 Elemento de unión

6. Cerrar la paleta.
7. Renombrar los objetos que se han añadido como se muestra en la siguiente figura, y después guardar el proyecto.

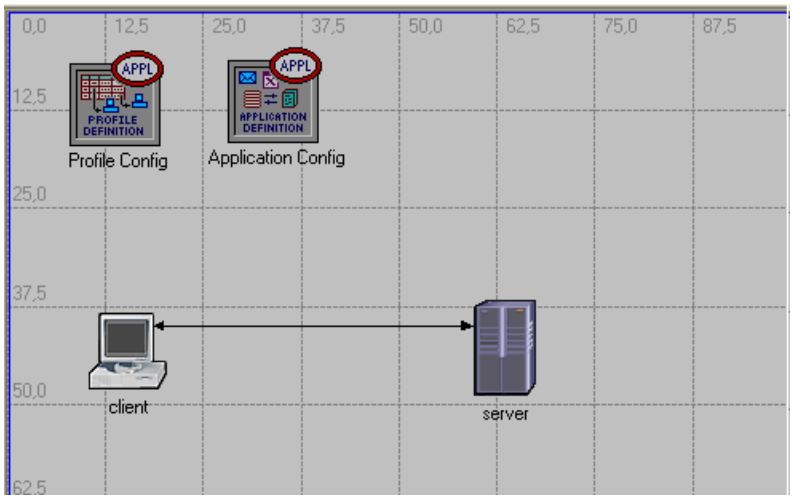


Figura 5.3.7.3 Escenario de *Ida y vuelta calculando el tiempo sin Time Stamp*

Configurar las aplicaciones:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Applications Config**→ Edit Attributes→ expandir el atributo **Application Definitions** y poner **rows** a 1→ expandir la nueva fila y llamar a esa fila **File Transfer (custom).**
2. Hacer clic en OK y guardar el proyecto.

Configurar de perfiles:

1. Hacer clic con el botón derecho del ratón sobre el nodo **Profiles Config** → Edit Attributes → expandir el atributo **Porfile Configuration** y poner **rows** a 1.
   - Para **row** 0, llamar y poner los atributos con los valores que se muestran en la siguiente figura → Hacer clic en OK.

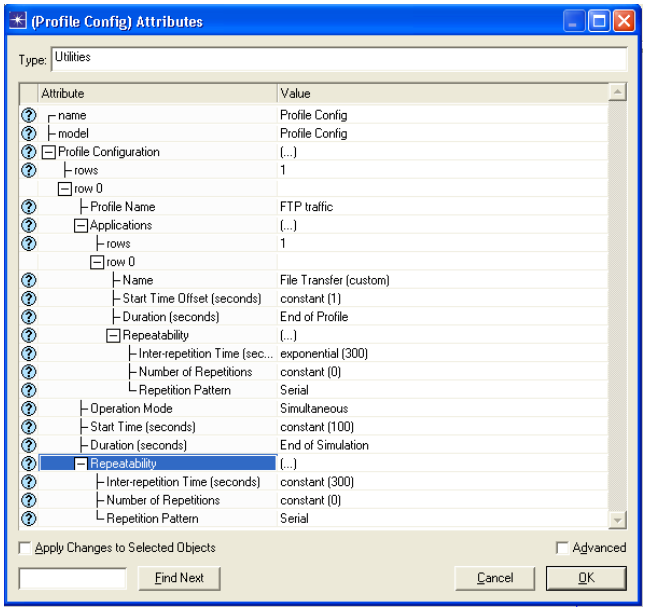


Figura 5.3.7.4 Configuración de Perfil

Con esto se han configurado las aplicaciones y los perfiles del TCP.

1. Hacer clic en el nodo **Client** → Edit Attributes:
   - Expandir el árbol **Application: supported Profiles** → poner **rows** a 1 → expandir el árbol **row 0** → poner **Profile Name** a **FTP traficc.**
2. Hacer clic en el nodo **Server** → Edit Attributes como se muestra en la figura:

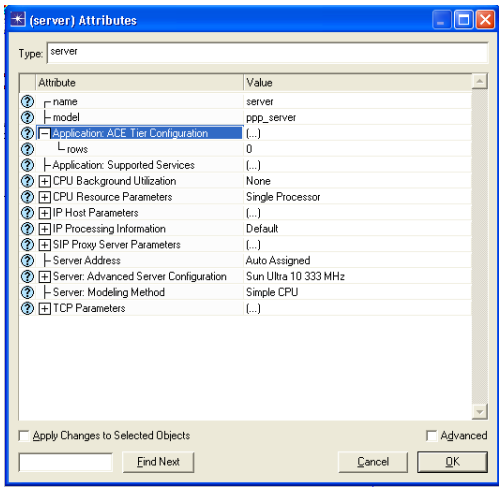


Figura 5.3.7.5 Configuración del Servidor

3. Hacer clic en OK y guardar el proyecto.

Al realizar una simulación del escenario y observar los resultados de las gráficas arrojadas obtenemos:

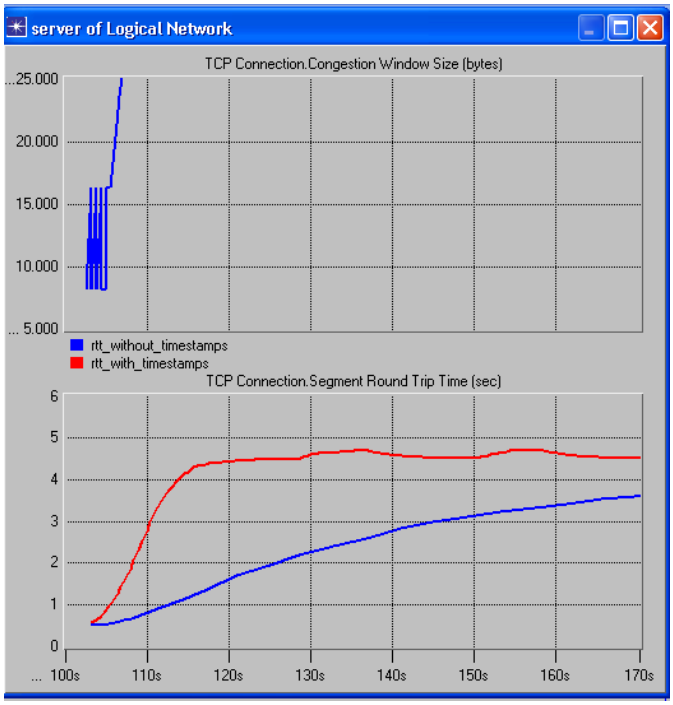


Figura 5.3.7.6 Análisis de Estadísticas

El eslabón entre el cliente y nodos de servidor es puesto a funcionar en 128Kbps que hace el tiempo de viaje de ida y vuelta (RTT) este alrededor de los 4 segundos.

La primera gráfica muestra la valoración de la ventana de congestión inicial cuando la opción de *TimeStamp* no es usada. En este caso TCP sólo ajusta despacio su valoración RTT con la estimación del valor real. Esto causa nuevas transmisiones innecesarias y un tamaño de ventana de congestión disminuido.

La segunda gráfica compara la valoración estimada del RTT de TCP basada en el apoyo de *TimeStamp*. *TimeStamp* proporciona mejor opción TCP de ida y mediciones de tiempo. Para permitir la compresión de la cabecera TCP en un enlace serial, el *timestamp* se desactiva.

Escenario 8: Ida y vuelta calculando el tiempo con Time Stamp

Para crear un proyecto nuevo:

8. Ejecutar **OPNET IT Guru Academic Edition** → Seleccionar **New** del menu **File**.
9. Seleccionar **Project** → Clic en **OK** → Nombre del proyecto <Sus_iniciales>**_TCP** y el escenario **rtt_with_timestamps**→ Clic en **OK**.
10. En el *Startup Wizard:* En la ventana de diálogo *Inicial Topology*, Asegurarse de que **Create Empty Scenario** esté seleccionado → Clic en **Next** → Elegir **Office** de la lista *Network Scale* → Clic en **Next** → Mantener **X Span** y **Y Span** en 100 → Clic en **Next** dos veces → Clic en **Ok**.
11. De la ventana de diálogo **Object Palette** arrastrar hasta el panel los siguientes elementos:

Figura 5.3.8.1 Elementos de la Paleta de Objetos

12. En la misma ventana de diálogo configurar la paleta a **links_ advanced** y seleccionar el siguiente elemento:

Figura 5.3.8.2 Elemento de unión

13. Cerrar la paleta.

14. Renombrar los objetos que se han añadido como se muestra en la siguiente figura, y después guardar el proyecto.

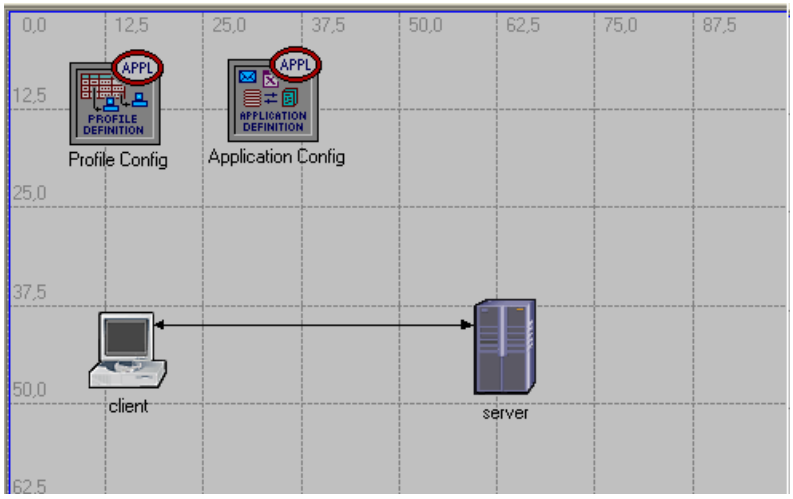


Figura 5.3.8.3 Escenario de Ida y vuelta calculando el tiempo sin Time Stamp

Configurar las aplicaciones:

3. Hacer clic con el botón derecho del ratón sobre el nodo **Applications Config**→ Edit Attributes→ expandir el atributo **Application Definitions** y poner **rows** a 1→ expandir la nueva fila y llamar a esa fila **File Transfer (custom).**

4. Hacer clic en OK y guardar el proyecto.

Configurar de perfiles:

2. Hacer clic con el botón derecho del ratón sobre el nodo **Profiles Config** → Edit Attributes → expandir el atributo **Porfile Configuration** y poner **rows** a 1.

   - Para **row** 0, llamar y poner los atributos con los valores que se muestran en la siguiente figura → Hacer clic en OK.

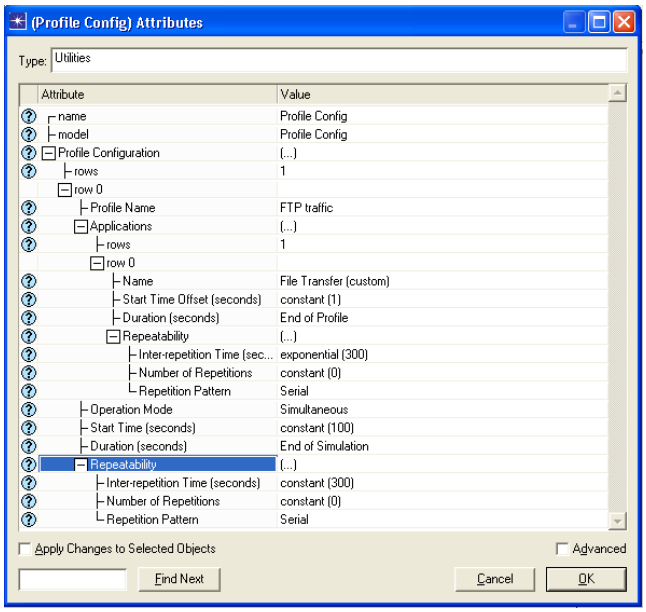


Figura 5.3.8.4 Configuración de Perfil

Con esto se han configurado las aplicaciones y los perfiles del TCP.

4. Hacer clic en el nodo **Client** → Edit Attributes:

   - Expandir el árbol **Application: supported Profiles** → poner **rows** a 1 → expandir el árbol **row 0** → poner **Profile Name** a **FTP traficc.**

5. Hacer clic en el nodo **Server** → Edit Attributes como se muestra en la figura:

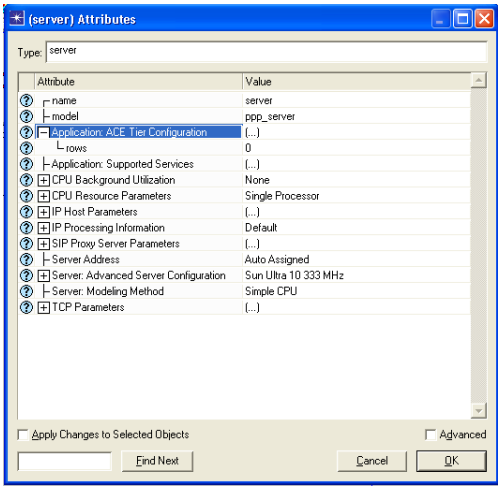


Figura 5.3.8.5 Configuración del Servidor

6. Hacer clic en OK y guardar el proyecto.

Al realizar una simulación del escenario y observar los resultados de las gráficas arrojadas obtenemos:

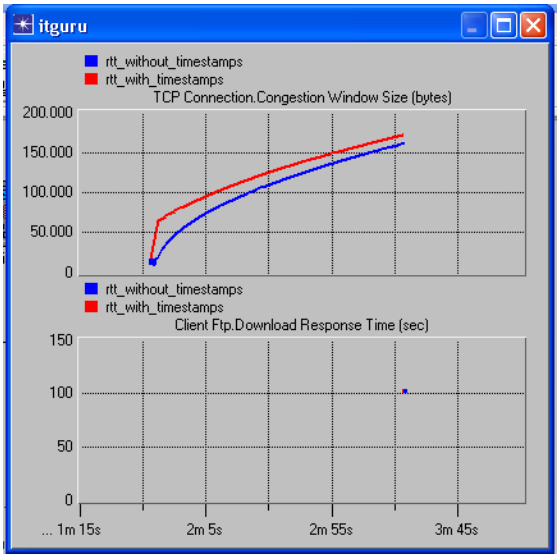


Figura 5.3.8.6 Análisis de Estadísticas

Esta es la misma configuración que la descrita en el escenario **rtt_without_timestamps**. El primer gráfico compara el FTP descargado del tiempo de respuesta dependiendo del *timestamps*.

El segundo gráfico compara el tamaño de la ventana de congestión para escenarios con o sin *timestamps*.

Como se sabe el *TimeStamp* mide el round-trip time de la conexión, protege a las conexiones de segmentos "viejos", en caso de que se utilicen grandes ventanas TCP. Los timestamp son una extensión del número de secuencia, se sugiere una frecuencia entre 1/1s y 1/1ms.

6. CONCLUSIONES

Teniendo en cuenta los resultados obtenidos de las simulaciones implementadas con OPNET, en general, puedo concluir que el programa y sus funcionalidades nos permiten desarrollar diferentes tipos de simulaciones de casos de la vida real en el área de las Computadoras y las Redes.

Mediante el desarrollo de las prácticas y la simulación de las mismas para diferentes escenarios, se puede visualizar claramente la Congestión en TCP y las diferentes circunstancias donde se presenta debido al envío rápido de paquetes. Este programa es una herramienta interactiva que por medio de escenarios creados por los programadores se puede determinar la configuración del sistema antes de ser implementado.

De acuerdo a las gráficas ahorradas por cada escenario de acuerda a la simulación, podemos concluir que el algoritmo que ayuda a recuperar la gran mayoría de los datos perdidos es el SACK con una caída, ya que este algoritmo se usan los reconocimientos para que el transmisor tome decisiones inteligentes respecto a los paquetes descartados, con esto se aumenta el throughput y evita innecesarias retransmisiones de paquetes que ya fueron entregados al receptor mejorando así la comunicación.

Como se pudo ver la Congestión en TCP se presenta debido al envío de paquetes nuevos sin haberse entregado los antiguos. Esto se manejo con diferentes mecanismos que permitieron manipular el tamaño de la ventana. También se presentó congestión debido al ruido en la línea, el cual se pudo despreciar, aunque en líneas inalámbricas no ocurre lo mismo. Internet acepta la existencia de dos problemas: Capacidad del receptor y Capacidad de la Red, donde cada problema se trata por separado.

Una propiedad típica que se pudo determinar en la congestión es la retroalimentación, que hace que la situación empeore, ya que los nodos entran en una situación de bloqueo debido a la saturación de la ventana. Debido a los retardos que se ocasionan en el envío de los

paquetes y a la saturación de la ventana, se puede decir que la Congestión en TCP se presentó debido ha:

- Retardos: Trabajar cerca de la capacidad de los enlaces es ideal desde el unto de vista de la productividad, pero se experimentan grandes retardos en una cola según la tasa de llegadas de paquetes cuando se acerca a la capacidad del enlace.
- Pérdidas: Como los búfers no son de tamaño infinito el emisor debe realizar retransmisiones para compensar los paquetes perdidos debido al desbordamiento de los búfers.
- Desperdicio de recursos: Las retransmisiones innecesarias realizadas por el emisor en presencia de grandes retardos, provocaron que se venza los temporizadores de retransmisión antes de que lleguen a su destino, además de hacer que el ancho de banda de los enlaces se utilice para encaminar copias innecesarias. Cuando un paquete es desechado a lo largo del camino, la capacidad de almacenamiento, procesamiento y transmisión que fue utilizada para cada uno de los nodos y enlaces, se desperdicia.

Los mecanismos de control de acuerdo a su desempeño se pudieron clasificar en: ***Preventivos*** los cuales pretenden prevenir la congestión, denominados a lazo abierto; y ***Reactivos*** los cuales intentan resolver el problema de congestión cuando ésta parece, denominados de lazo cerrado o con realimentación.

En las actividades desarrolladas, las simulaciones de los escenarios y los análisis de los diferentes resultados obtenidos en forma gráfica con la ayuda de OPNET, dieron la oportunidad de confrontar lo aprendido durante el transcurso del Minor de Telecomunicaciones, permitiendo validar lo aprendido con la tecnología simulada en este trabajo, donde se observa el comportamiento de las redes TCP.

De esta forma, se presentó una alternativa para el diseño de redes (software) y una herramienta de referencia para los desarrolladores de redes o todo aquel que trabaje con las *Tecnologías de la Información,* protocolos y desarrollo de los mismos.

7. GLOSARIO

- **ACK:** ACKNOWLEDGEMENT (ACK) (en español **acuse de recibo**), en comunicaciones entre computadores, es un mensaje que se envía para confirmar que un mensaje o un conjunto de mensajes han llegado. Si el terminal de destino tiene capacidad para detectar errores, el significado de ACK es "ha llegado y además ha llegado correctamente". Hay tipos más complejos de ACK cuyo significado podría traducirse como "reenvíame la trama 2" o "he recibido tu último mensaje, pero no puedo recibir más hasta que termine de procesar los anteriores". La forma exacta del mensaje, es decir, la combinación de unos y ceros que lo caracterizan y su posición dentro de una trama, varía según el protocolo utilizado.
- **ALGORITMO:** Es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema. Es un método para resolver un problema a través de una secuencia de pasos lógicos que lo llevará a cumplir un objetivo ó solución. Características de los algoritmos: Debe ser eficiente e indicar el orden de realización de cada paso, debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez y, debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea debe de tener un número finito de pasos.
- **BIT:** Es el acrónimo de **B**inary dig**it**. (dígito binario). Un bit es un dígito del sistema de numeración binario. El bit es la unidad mínima de información empleada en informática, en cualquier dispositivo digital, o en la teoría de la información. Con él, se puede representar dos valores cualesquiera, como verdadero o falso, abierto o

cerrado, blanco o negro, norte o sur, masculino o femenino, amarillo o azul, etc. Basta con asignar uno de esos valores al estado de "apagado" (0), y el otro al estado de "encendido" (1).

- **BUFFER:** En informática, un **buffer de datos** es una ubicación de la memoria en una computadora o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada. Por ejemplo, un analizador TRF tendrá uno o varios buffers de entrada, donde se guardan las palabras digitales que representan las muestras de la señal de entrada.
- **BYTE:** Es una secuencia contigua de un número de bits *fijo*. También, considerado como una secuencia contigua de bits en una computadora binaria que comprende el *sub-campo direccionable más pequeño* del tamaño de palabra natural de la computadora. Esto es, la unidad de datos binarios más pequeña en que la computación es significativa, o se pueden aplicar las cotas de datos naturales.
- **CONGESTIÓN:** Se define como una excesiva cantidad de paquetes almacenados en los buffers de varios nodos en espera de ser transmitidos. La congestión es indeseable porque aumenta los tiempos de viaje de los paquetes y retrasa la comunicación entre usuarios. Este es un gran problema que debe ser evitado.
- **DATAGRAMA:** Es un fragmento de paquete que es enviado con la suficiente información como para que la red pueda simplemente encaminar el fragmento hacia el ordenador receptor, de manera independiente a los fragmentos restantes. Esto puede provocar una recomposición desordenada o incompleta del paquete en el ordenador destino. La estructura de un datagrama es: cabecera y datos.
- **HTTP:** El protocolo de transferencia de hipertexto (***HTTP***, *HyperText Transfer Protocol*) es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el

contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido. También sirve el protocolo para enviar **información adicional** en ambos sentidos, como formularios con campos de texto. HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. Al finalizar la transacción todos los datos se pierden.

- **HOST:** A una máquina conectada a una red de ordenadores y que tiene un nombre de equipo (en inglés, *hostname*). Es un nombre único que se le da a un dispositivo conectado a una red informática. Puede ser un ordenador, un servidor de archivos, un dispositivo de almacenamiento por red, una máquina de fax, impresora, etc. Este nombre ayuda al administrador de la red a identificar las máquinas sin tener que memorizar una dirección IP para cada una de ellas.
- **INTERFAZ:** es el *puerto* (circuito físico) a través del que se envían o reciben señales desde un sistema o subsistemas hacia otros. No existe un interfaz universal, sino que existen diferentes estándares (Interfaz USB, interfaz SCSI, etc.) que establecen especificaciones técnicas concretas (características comunes), con lo que la interconexión sólo es posible utilizando el mismo interfaz en origen y destino.
- **INTERNET:** Es una red mundial de computadoras con un conjunto de protocolos, el más destacado, el TCP/IP. Aparece por primera vez en 1969, cuando ARPAnet establece su primera conexión entre tres universidades en California y una en Utah. También se usa el término Internet como sustantivo común y por tanto en minúsculas para designar a cualquier red de redes que use las mismas tecnologías que Internet, independientemente de su extensión o de que sea pública o privada. Cuando se dice *red*

de redes se hace referencia a que es una red formada por la interconexión de otras redes menores.

- **PIPE:** Se utiliza normalmente con la aplicación SACK TCP. Representa el número estimado de paquetes pendientes en el camino de una activa conexión TCP.
- **RENO:** Es una aplicación TCP que tiene la capacidad de *Fast Retransmit & Fast Recover* simultáneamente.
- **ROUTER:** Enrutador, encaminador. Dispositivo de **hardware** o software para interconexión de redes de computadoras que opera en la capa tres (nivel de red) del modelo OSI. El router interconecta segmentos de red, o algunas veces hasta redes enteras. Hace pasar paquetes de datos entre redes tomando como base la información de la capa de red. El router toma decisiones basadas en diversos parámetros con respecto a la mejor ruta para el envío de datos a través de una red interconectada y luego redirige los paquetes hacia el segmento y el puerto de salida adecuados. Una de las más importantes es decidir la dirección de la red hacia la que va destinado el paquete (En el caso del protocolo *IP* esta sería la dirección IP). Otras decisiones son la carga de tráfico de red en las distintas interfaces de red del router y establecer la velocidad de cada uno de ellos, dependiendo del protocolo que se utilice.
- **RTO:** Especifica retransmisión por la pérdida de valor del tiempo, paquete o paquetes muy retrasados.
- **RTT:** Soporte de ida y vuelta en el tiempo, es decir, el tiempo que lleva para un segmento TCP para llegar al destino y ser respaldado por ACK.
- **SACK:** Soportes de que el selectivo informa al receptor sobre el remitente y sólo la pérdida de paquetes. Esto ayuda a aumentar el rendimiento y la reducción en el tiempo de recuperación.

- **SIMULACIÓN:** Es una técnica que imita el comportamiento de un sistema del mundo real conforme evoluciona en el tiempo. Por lo tanto, se podrá analizar y observar características, sin la necesidad de acudir al sistema real.
- **SISTEMA:** Conjunto de elementos que, actúan e interactúan para lograr algún fin logico.
- **SMTP:** Protocolo simple de transferencia de correo electrónico. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos (PDA's, teléfonos móviles, etc.). Está definido en el RFC 2821 y es un estándar oficial de Internet.
- **SSH:** Es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo el ordenador mediante un intérprete de comandos, y también puede redirigir el tráfico de X para poder ejecutar programas gráficos si tenemos un Servidor X arrancado. Además de la conexión a otras máquinas, SSH nos permite copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a las máquinas y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.
- **SWITCH:** (en castellano "conmutador") Es un dispositivo electrónico de interconexión de redes de computadoras que opera en la capa 2 (nivel de enlace de datos) del modelo OSI (*Open Systems Interconection*). Un conmutador interconecta dos o más segmentos de red, funcionando de manera similar a los puentes (bridges), pasando datos de un segmento a otro, de acuerdo con la dirección MAC de destino de los datagramas en la red. Los conmutadores se utilizan cuando se desea conectar múltiples redes, fusionándolas en una sola. Al igual que los puentes, dado que

funcionan como un *filtro* en la red, mejoran el rendimiento y la seguridad de las LANs (*Local Area Network*- Red de Área Local).

- **TAHOE:** Es una aplicación TCP adornado con la capacidad de *Fast Retransmit*.

8. BIBLIOGRAFIA

- Información técnica del glosario obtenida de http://www.wikipedia.com
- **OPNET IT Guru Academia Edition 9.1.** Programa obtenido de la página en internet del fabricante. http://www.opnet.com/products/itguru
- Práctica 9 de la asignatura Redes de computadores, 4º curso de la Facultad de Ingeniería de la Universidad Politécnica de Valencia, España. http://www.redes.upv.es/redesfi/pract/TCP.pdf
- Control de congestión TCP. http://www-gris.det.uvigo.es/~estela/SC0708/control_congestion.pdf
- Protocolos TCP/IP. http://usuarios.lycos.es/janjo/janjo1.html
- Tutorial y descripción técnica de TCP/IP. http://ditec.um.es/laso/docs/tut-tcpip
- Control de Congestión en TCP. http://webdelprofesor.ula.ve/ingenieria/gilberto/redes/10_capaTransporteCongestion.pdf
- *Redes e Internet de Alta Velocidad*, STALLINGS WILLIAM. Prentice Hall año 2004. 2da Edición. ***Capítulo 1****: Introducción, Sección 1.3 Redes TCP/IP y ATM Avanzadas.* ***Capítulo 2****: Protocolos y la arquitectura TCP/IP.* ***Capítulo3****: TCP e IP, Sección 3.1 El protocolo de control de transmisión (TCP).*

- Evitación de Congestión en TCP.

  http://helios.tlm.unvarra.es/asignaturas/ro/ro05_06/clases-pdf/RO_trnasporte-7.pdf

- Arranque Lento en TCP.

  http://www.it.uc3m.es/~pablo/asignaturas/rysc1/alumnos/06-Congestion_TCP.pdf.