ANÁLISIS DEL IMPACTO DEL NIVEL DE ENLACE DE DATOS EN EL DESEMPEÑO DEL PROTOCOLO TCP EN LAS REDES SATELITALES.

ANEXOS DEL TRABAJO DE GRADO

Víctor Hugo Ruiz Guachetá.
Juan Esteban Nava Villarreal.

Director: Mg Francisco Javier Terán.

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
Departamento de Telecomunicaciones
Línea de Investigación: Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, Junio de 2010

ANEXO A

GENERALIDADES DE LAS REDES SATELITALES Y EL PROTOCOLO TCP.

1. LA RED SATELITAL.

Son redes que hacen uso de los satélites artificiales, los cuales se encuentran localizados en orbitas alrededor de la tierra. Este tipo de redes tienen la capacidad de recibir y enviar datos a través del satélite y las estaciones terrenas.

Una red satelital está conformada por un satélite, una estación terrena que controla su funcionamiento, y las diferentes estaciones terrestres las cuales ayudan a proporcionar las facilidades para realizar la transmisión y recepción del tráfico a través de la red satelital. Este tipo de redes presentan las siguientes características [1]:

- Las transmisiones de datos se encuentran en el orden de los Kbps o Mbps, dentro de las bandas de frecuencia C (4Ghz, 6Ghz), Ku (11Ghz, 14Ghz) y Ka (20Ghz, 30Ghz).
- Son redes que tienen un alto costo para su implementación, ya que tienen que hacer uso de un satélite artificial, ya sea propio o alquilado.
- Son redes que rompen las distancias y el tiempo, es decir la red satelital es independiente de la distancia por qué se puede establecer comunicación desde cualquier lugar de la tierra siempre y cuando este dentro del área de cobertura del satélite.

1.1. TIPOS DE ORBITAS EN UNA RED SATELITAL

1.1.1. Órbita Geo- Estacionaria GEO.

Los satélites de órbita terrestre geoestacionaria se encuentran en la órbita del ecuador a una distancia de 36000Km, giran con un patrón circular con una velocidad igual al de la tierra, en teoría dan cobertura a la superficie terrestre a excepción de los polos con solo 3 satélites, pero en la práctica se requieren de 6 satélites, se caracteriza por que sus satélites se encuentran referenciado a un mismo lugar de superficie, la distancia entre cada uno de los satélites es de 2 grados lo que presenta interferencias con las señales que reciben de las estaciones terrenas; las redes satelitales geoestacionarias presentan un retardo de 125 ms para cada salto, luego para una red típica que presente dos saltos el retardo de propagación estará en el orden de los 250 ms a 270ms aproximadamente a excepción de las redes VSAT en configuración en estrella, además se necesitan de dispositivos de propulsión para mantenerlos en órbita [2,3].

1.1.2. MEO.

Los satélites de órbita media (MEO) están a una altura entre los 10075 y 20150 Km, para dar cobertura a la superficie terrestre se necesitan muchos más satélites en órbita, pero debido a la altura que están de la superficie terrestre, estos no se encuentran en una posición fija, los satélites en órbita hacen uso de mecanismos de sincronización y localización para poder dar cobertura, sin embargo, la latencia en sus trayectos son mucho menores comparado con los

sistemas GEO ya que la distancia que tiene que recorrer la señal desde la tierra al satélite y del satélite a la tierra es mucho menor [3,4].

1.1.3. LEO.

Las órbitas terrestres de baja altura prometen un ancho de banda extraordinario y una latencia mucho más reducida. Los LEO orbitan generalmente por debajo de los 5035 kilómetros, y la mayoría de ellos se encuentran mucho más abajo, entre los 600 y los 1600 kilómetros. A tan baja altura, la latencia adquiere valores casi despreciables de unas pocas centésimas de segundo. Tres tipos de LEO manejan diferentes cantidades de ancho de banda. Los LEO pequeños están destinados a aplicaciones de bajo ancho de banda (de decenas a centenares de Kbps), como los buscapersonas, e incluyen a sistemas como OrbComm. Los grandes LEO pueden manejar buscapersonas, servicios de telefonía móvil y algo de transmisión de datos (de cientos a miles de Kbps). Los LEO de banda ancha (también denominados megaLEO) operan en la franja de los Mbps y entre ellos se encuentran *Teledesic, Celestri y SkyBridge* [3,4].

1.2. ARQUITECTURA DE LA RED SATELITAL.

La tecnología de redes satelitales, representada por satélites y el perfeccionamiento de las estaciones terrenas están revolucionando el mundo; la arquitectura de este tipo de redes se puede definir en función de la presencia o no de canal de retorno, es decir desde los usuarios hacia la red, es por esto que se pueden definir las siguientes arquitecturas:

1.2.1. Redes Unidireccionales.

Este tipo de redes se caracterizan por no tener un canal de retorno, son sistemas que se fundamentan en el uso de una estación transmisora principal por donde son enviadas las señales al satélite y que posteriormente son recibidas en tierra por un gran número de estaciones exclusivamente receptoras; un ejemplo típico para estas redes son los servicios de difusión como la TV [5].

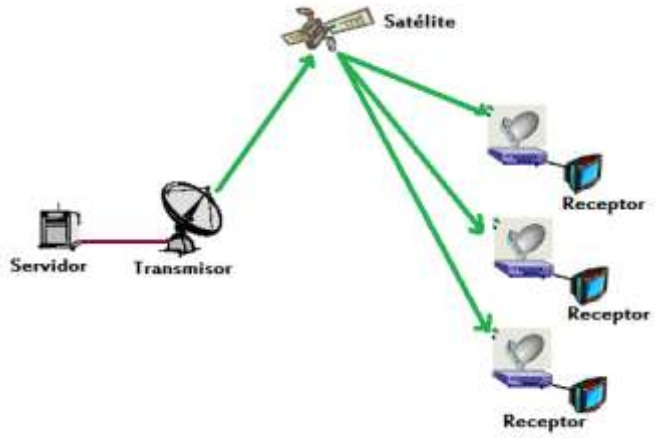


Figura 1. Topología de redes unidireccionales.

1.2.2. Sistemas Bidireccionales.

La arquitectura de estas redes es algo similar a las unidireccionales con la gran diferencia de que poseen un canal de retorno, donde generalmente la capacidad del canal de subida es menor que el canal de bajada. Una estación central (Hub) transmite una o varias portadoras a un conjunto de estaciones remotas asociadas. La estructura de la información contenida en cada portadora es una multiplexación de múltiples canales donde cada uno puede ser asignado para su recepción por una o varias estaciones remotas; los métodos de multiplexación más comunes son TDM FDM [5].

Este tipo de redes tienen una ventaja sobre las unidireccionales, que es la no dependencia de otro tipo de redes para retornar la información, son sistemas que hacen uso de antenas grandes los cuales requieren de personal especializado para su instalación; estos sistemas desde su creación han estado enfocados hacia el sector comercial y empresarial, pero tras el paso de los años y con el avance de la ciencia y la tecnología se han extendido al sector residencial, al bajar sus costos y el aumentar su desempeño.

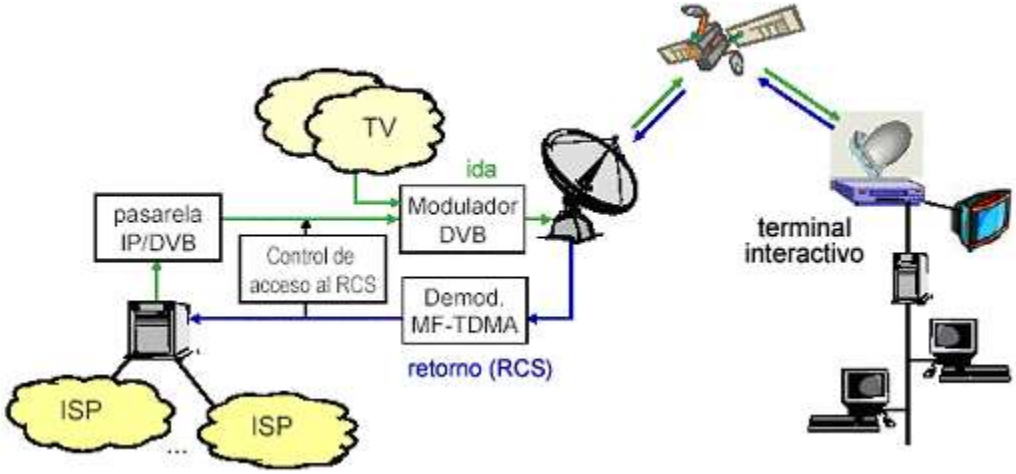


Figura 2. Arquitectura red bidireccional [5].

1.2.3. Redes Híbridas.

Este tipo de redes combinan la transmisión vía satélite junto con la transmisión a través de una red terrestre complementaria. Estas redes tienen un canal de retorno a través de una red diferente a la satelital, el cual permite la interacción entre el satélite y el prestador del servicio; la parte satelital se encuentra formada por un satélite que proporciona cobertura en un determinado territorio, la otra parte por una red terrestre.

Existen diversas formas de coordinar el canal de ida por satélite con el de retorno por la otra red, de tal manera que la información que el usuario pide por el canal de retorno sea encaminada por el satélite, de tal forma que no se presenten problemas a la hora del intercambio de información [5].

Figura 3. Arquitectura de red satelital híbrida [5].

Las redes hibridas permiten prestar servicios interactivos asimétricos, como por ejemplo la navegación por Web en Internet o redes VSAT de capacidad limitada, una de sus ventajas radica en que los terminales son más baratos y pueden ser instalados por el propio usuario.

1.2.4. Redes Satelitales de Múltiples Saltos en los Satélites.

Estas redes se caracterizan por que en algunas ocasiones el tráfico puede pasar por más de un salto para llegar a su destino, lo que genera un problema genérico en los enlaces satelitales, como lo son las limitaciones debido a los largos tiempos de propagación, errores en la información y el ancho de banda a utilizar [6].

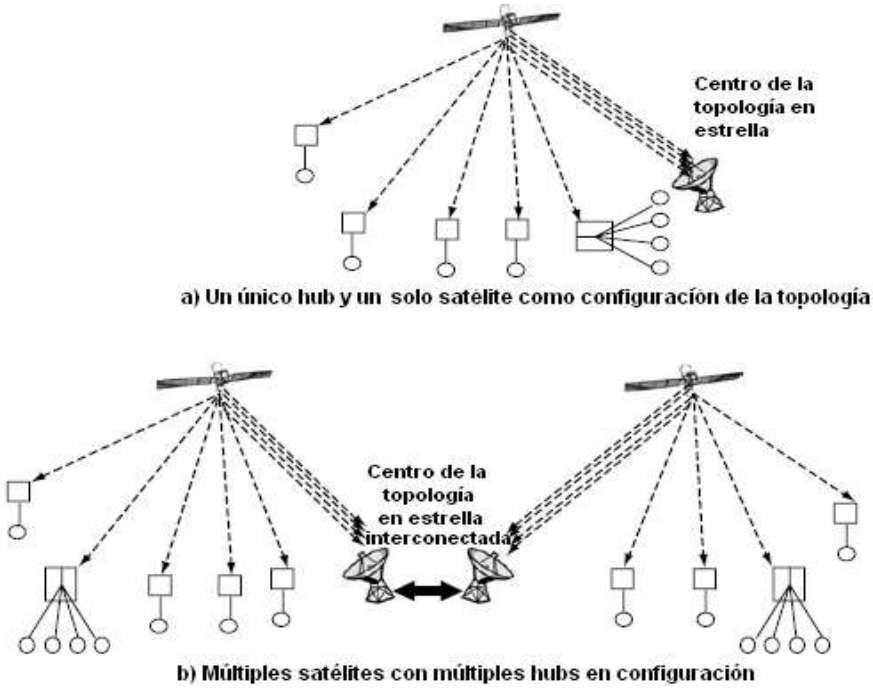


Figura 4. Topología de múltiples saltos con *hub* en el centro [6].

En este tipo de configuración, una conexión extremo a extremo se enruta a través de la red satelital más de una vez, pasando por el mismo satélite o diferentes satélites. El primer caso, es ampliamente utilizado en terminales de muy pequeña apertura (las redes VSAT), donde la señal de las redes entre dos terminales es demasiado débil para hacer una comunicación directa, un gran concentrador (*hub*) en tierra es utilizado para impulsar la señal entre las terminales de comunicación. En este último caso, un salto no puede ser lo suficientemente lejos para llegar a terminales remotos, por lo tanto, más saltos se utilizan para las conexiones. La topología de la red satelital forma una estrella con un concentrador en tierra, en el centro de la estrella o múltiples estrellas donde los centros están interconectados para proporcionar una conexión total en la red como se muestra en la figura 4.

1.2.5. Constelación de Redes Satelitales con y sin Conexión Intersatelital (ISL).

Establecer una red inalámbrica tipo Ad-Hoc en el espacio, ha despertado gran interés en el sector de telecomunicaciones por satélite bajo las premisas de extender la zona de cobertura y evitar saltos adicionales que incorporan retardos poco convenientes. Los enlaces ópticos o de radiofrecuencia entre satélites se plantean con la existencia de Procesamiento a Bordo (OBP, *On-Board Processing*) disponible, puesto que es necesario conocer la información recibida para ejecutar un redireccionamiento a una estación terrena o simplemente interpretarla para responder a una petición de la red.

Existen tres tipos diferentes de enlaces intersatelitales (ISL, *Inter Satellite Links*), estos son: entre satélites de órbita geoestacionaria, entre un satélite geoestacionario y uno no geoestacionario y entre satélites no geoestacionarios. Estos dos últimos son conocidos como enlaces InterOrbitales.

Los ISLs se realizan mediante la transmisión de señales de radiofrecuencia o señales ópticas, donde optar por una de ellas implica considerar los costos, peso de los equipos y desempeño. En este sentido, los costos resultan ser más representativos para enlaces en RF en relación con el peso de quipos a poner en órbita, no obstante, la implementación con RF resulta ser más fácil dado a un mayor ancho de haz y en consecuencia mayor facilidad para capturar la señal tanto de sincronización como de alineación de ambos extremos antes de dar inicio a la transmisión de datos. Esto mismo no sucede para una señal óptica cuyo haz es del orden de los micro-radianes bajo un control de alineación del orden de los mili-radianes que representa un mayor problema técnico, pero que se intenta resolver con el desarrollo de dispositivos de apuntamiento avanzado en transmisión y tecnología de detección por píxeles activos APS (*Active Píxel Sensor*) que busca aumentar el nivel de sensibilidad de los receptores.

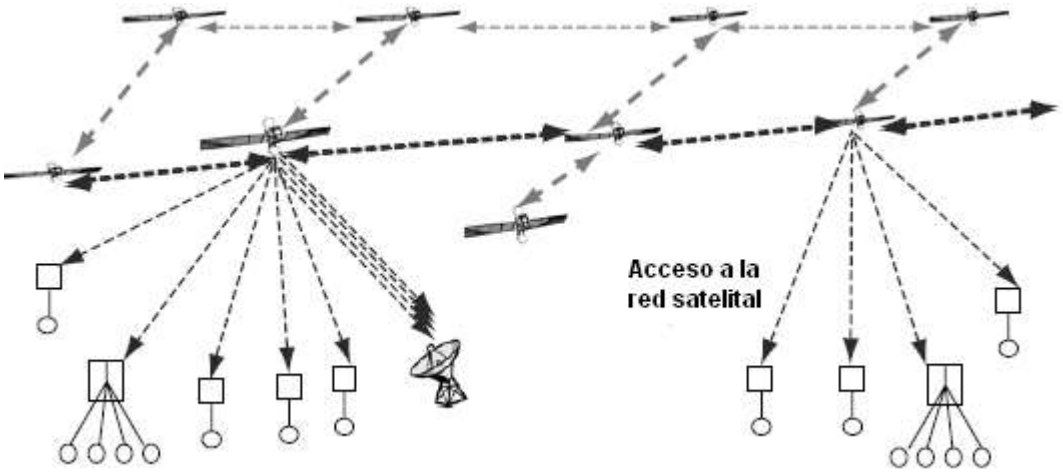


Figura 5. Redes satelitales con ISL [6].

1.3. CARACTERÍSTICAS DE LOS CANALES SATELITALES.

Los canales satelitales tipo GEO presentan cuatro problemas, principalmente debido a que están posicionados a una distancia de 36000 Km. aproximadamente donde sus enlaces desde la estación terrena satélite o viceversa producen mucha atenuación de la señal [6]. Los problemas de mayor impacto en este tipo de plataformas son:

1.3.1. Retardo de Propagación

Para los satélites GEO, el tiempo de retardo desde la estación terrena al satélite es de 125ms por cada enlace, 500ms en un enlace bidireccional para un ACK, son tiempos muy altos si se comparan con los enlaces de radio ya que se aumenta notoriamente la trayectoria del enlace, si se toma en cuenta el tiempo de procesamiento a bordo del satélite, y el retardo en la colas de una pasarela (Gateway) o de un *router*; este tiempo aumentaría lo que agudiza aun más el problema [6,7].

1.3.2. Pérdidas de Propagación y Potencia Limitada.

Cuando se habla de un enlace satelital se conoce que sus pérdidas de propagación que están del orden de los 200 dB dependiendo de la frecuencia que se utilice estas pérdidas aumentaran o disminuirán según sea el caso, ya que para algunas frecuencias se ven afectadas por otros fenómenos atmosféricos como vapor de agua, absorción atmosférica entre otros.

Los enlaces desde las estaciones terrenas al satélite no se ven tan afectados ya que estas pueden aumentar la potencia de transmisión y aumentar la ganancia de las antenas para compensar las pérdidas, mientras que para el enlace satélite estación terrena se ve afectado por dos simples razones, la primera es en las frecuencias de uso comercial no se puede interferir su servicio y se debe garantizar su no interferencia, y la segunda razón es debido al propio satélite ya que los niveles de potencia son limitados, ya que parte de la potencia utilizada por el satélite es suministrada por los paneles solares y por ello las estaciones terrenas tienen antenas muy sensibles para recepcionar estas señales del espacio [2,6].

1.3.3. Limitado Ancho de Banda y Espacio Orbital

Los satélites geoestacionario cada día están ocupando más la órbita ecuatorial lo que ha significado que se presenten interferencias debido al espacio tan reducido que hay de un satélite al otro, como consecuencia se hace uso de antenas más pequeñas en las estaciones terrenas con una configuración del ancho del haz mucho más enfocado tratando de evitar la interferencia que se consigue con los lóbulos laterales de estas mismas antenas.

El espectro de radio es limitado, y además se deben respetar los acuerdos internacionales de comunicaciones comerciales por lo tanto hay una limitación de ancho de banda para utilizar como frecuencias de portadoras, en el transpondedor y a bordo del satélite [6].

1.3.4. El Ruido.

Los enlaces satelitales se caracterizan por tener una muy baja señal a ruido debido a que las pérdidas en el trayecto son muy grandes ya que disminuyen con el cuadrado de la distancia. Un valor típico de tasa de error de bit (BER) es de 10^{-7} para mejorar la relación de BER se pueden utilizar códigos de control de errores [7].

Los tipos de ruidos que están presentes en la señal recibida son generados por una fuente interna (receptor) o una fuente externa (contribución de la antena), estas pueden ser causadas por la interferencia de los satélites, productos de intermodulación o canales adyacentes o simplemente por la multitrayectoria de la señal [6,8].

1.4. VENTAJAS Y DESVENTAJAS DE LAS REDES SATELITALES.

Los sistemas de comunicación hoy por hoy han evolucionando a las necesidades de los usuarios y los sistemas satelitales están desempeñando una tarea necesaria para la prestación de servicios multimedia por Internet, su principal virtud para ofrecerlo es poder brindar una amplia cobertura al hacer uso de un solo transpondedor el cual podría cubrir cualquier tipo de zona que se requiera, se tiene también que las comunicaciones satelitales soportan comunicaciones fijas y móviles, un ejemplo de ello es la tecnología 3G, cuando requiera prestar su servicio en áreas donde la estación terrena no ofrece ninguna cobertura y no haya tecnologías de acceso comunes como es RSDI, ADSL, HFC, o simplemente no hay cobertura en la zona por ser áreas de difícil acceso, lo podrían hacer sin ningún inconveniente si se emplea una red satelital [4].

Uno de sus principales inconvenientes es la facilidad de los usuarios de recepcionar la señal, prácticamente desde cualquier punto de la tierra, estas pueden ser interceptadas violando así la confiabilidad del servicio, para evitarlo las comunicaciones tienen que ser encriptados.

Otro problema a tener en cuenta en las redes satelitales son las altas pérdidas de propagación en el espacio libre, esto se debe a que la señal tiene que recorrer distancias muy largas y en las que algunos protocolos como TCP debido a su estructura no funcionan muy bien, adicionalmente se pueden percibir interferencias en la recepción de la señal y el aumento de la potencia de transmisión de las estaciones terrenas como del satélite es limitado.

2. EL PROTOCOLO TCP ESTANDAR.

TCP es un protocolo de transporte orientado a conexión lo que significa que hace uso de acuses de recibo ACK para confirmar que en el destino ha llegado correctamente la información, actualmente se encuentra documentado en el RFC 793 actualizado en el RFC1122 y ampliado en el RFC1323 para trabajar en redes heterogéneas IP [7].

2.1. CARACTERISTÍCAS GENERALES DE TCP.

TCP es un protocolo que proporciona un servicio de transporte de datos que ofrece al nivel superior, sus principales características son [9]:

- Fiabilidad.
- Control de Flujo.
- Orientación a conexión.
- Multiplexación.
- Orientación a flujo de octetos.
- Transferencia con almacenamiento.

2.1.1. Fiabilidad.

TCP esta diseñado para recuperarse de cualquier situación; como es el caso de congestión (pérdida de segmentos debido al desbordamiento del búfer del *router* o el inminente desbordamiento del búfer), pérdida de corrupción (pérdida de segmentos debido a los bits dañados) o algún desorden en la llegada de los datos, y para ello utiliza retransmisiones y reconocimientos [9].

Cuando TCP Inicia una transmisión, este hace una copia de la información enviada en la cola de retransmisión y espera a que arribe su respectivo reconocimiento, si la información llega correctamente, se borra inmediatamente de la cola de retransmisión y si no lo hace se entiende que ha expirado el tiempo del temporizador y se vuelve a retransmitir la información; en recepción los números de secuencia del segmento TCP le permiten ordenar los segmentos que han llegado al receptor y a conocer cuál es el número de secuencia que se espera recibir, de esta manera TCP le da fiabilidad a la información que se transmite.

Una de las características de TCP para que sea considerado como un protocolo fiable, es que hace uso de un esquema de reconocimientos acumulativos, es decir, el receptor informa con el número de reconocimiento de hasta qué octeto del flujo de datos enviados ha recibido correctamente. Este sistema presenta varias ventajas [6]:

- Por un lado los reconocimientos son fáciles de generar y no resultan ambiguos.

- Por otro, la pérdida de reconocimientos no fuerza, necesariamente, la retransmisión de segmentos.

Esta característica de TCP permitirá mejorar su comportamiento en enlaces asimétricos, sin embargo, el carácter acumulativo de los reconocimientos hace que el emisor no reciba

información de todas las transmisiones correctas sino únicamente del último octeto de flujo continuo recibido sin errores

Para optimizar el recurso de la red y establecer mecanismos de temporización adecuados se han formulado las siguientes estrategias.

- **Algoritmo de retransmisión adaptativo.**

Es el encargado de ajustar el temporizador de transmisión RTO el cual no debe ser tan pequeño como para responder a la pérdida y no tan grande como para forzar la retransmisión. TCP utiliza este algoritmo para monitorear el retardo en cada conexión y ajusta el valor de RTO de acuerdo con ese valor [9].

Para obtener un cálculo adecuado del RTO (tiempo de retransmisión) se toman muestras del RTT (tiempo de ida y vuelta) y de esta forma se calcula un tiempo promedio de RTT, como se indica en la ecuación 1.

$$RTT = \alpha RTT_{Anterior} + (1-\alpha) RTT_{Nuevo} \quad (1)$$

Por lo tanto el tiempo de retransmisión RTO está dado por:

$$RTo = \beta * RTT \quad (2)$$

Los valores que se recomiendan de de α y β son 0.9 y 2 respectivamente, para redes fijas, estos valores se han encontrando de manera experimental y no aseguran unl buen desempeño del algoritmo en otras circunstancias particulares [10].

- **Algoritmo de Karn.**

Este algoritmo se encarga de determinar si el reconocimiento recibido fue debido al segmento transmitido por primera vez o a su retransmisión, esto es importante ya que para el cálculo del RTO no se debe considerar los RTTs de los segmentos retransmitidos [9].

Después de un largo tiempo de usar TCP, se descubrió una falla para el cálculo de RTO los ACKs no reconocen una transmisión, sino la recepción de datos. En otras palabras, cada vez que un segmento es transmitido y un ACK llega al transmisor, es imposible determinar si este ACK se debe asociar con la primera o segunda transmisión de un segmento para la medición del nuevo RTT. Este efecto se puede apreciar mejor en la figura 6.1, en este caso se asumía que el ACK era de la transmisión original, pero era realmente del segundo. El caso contrario se muestra en la figura 6.2. En este caso se asumía que el ACK era el de la segunda transmisión, pero era realmente del primero [11].

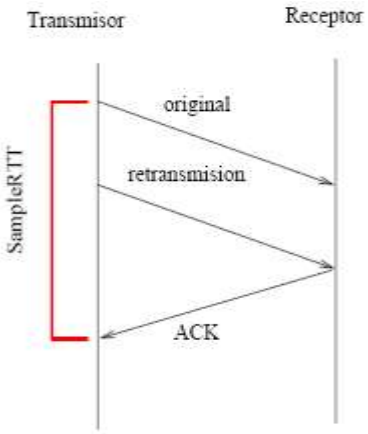


Figura 6.1 ACK transmitido por segunda vez [11].

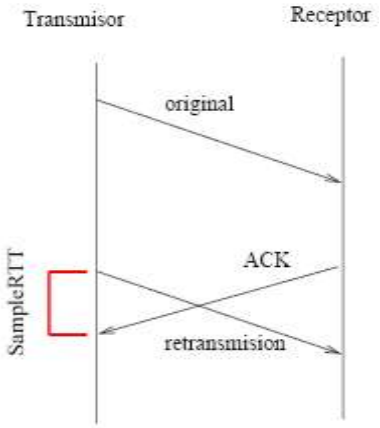


Figura 6.2 ACK de la primera transmisión [11].

Para dar solución a este problema, toda vez que TCP retransmita un segmento, deja de tomar muestras de RTT. Solo toma muestras de segmentos que han sido enviados una vez. Además se agrega otro cambio: cada vez que TCP transmite, el siguiente *timeout* lo fija como el doble del último en vez de basarse en el último RTT estimado.

En otras palabras este algoritmo se encarga de determinar si el reconocimiento recibido fue debido al segmento transmitido por primera vez o a su retransmisión, esto es importante ya que para el cálculo del RTO no se debe considerar los RTTs de los segmentos retransmitidos, el Algoritmo Karn usa *backoff* exponencial cuya función es impedir que TCP actué agresivamente frente al *timeout* [11].

- **Marca Temporal o *Timestamp.***

Esta opción aprovecha el ancho de banda para enviar información sobre el tiempo de cada segmento. La opción *Timestamps* permite al emisor TCP tomar más muestras de los valores RTT (*Round Trip Time*), por lo que el valor RTO (*Retransmission Time Out*) se calcula con mayor precisión, este mecanismo es útil cuando el recurso de ancho de banda no es escaso de lo contrario es más efectivo el algoritmo de Karn [9].

- **Backoff exponencial.**

Este algoritmo se emplea cuando se producen retransmisiones y el tiempo del temporizador expira**,** su funcionamiento consiste en incrementar el valor del RTO si las retransmisiones persisten de una manera continua; algunas implementaciones utilizan técnicas diferentes para establecer el *backoff*, la mayoría usan un factor multiplicativo como se muestra en la ecuación 3.

$$RTo_{Nuevo} = \beta RTo_{Anterior} \qquad \text{Con } \beta = 2 \text{ (normalmente)} \qquad (3)$$

2.1.2. Control de Flujo.

TCP utiliza un mecanismo de ventana deslizante para que el emisor envíe múltiples paquetes sin esperar recibir sus ACKs correspondientes, aunque existen otros mecanismos como por ejemplo el *Stop&Wait* que no permiten que se envíen los segmentos hasta que el transmisor los haya reconocido. Para que el mecanismo funcione correctamente es necesario que el receptor informe sobre el tamaño de ventana que puede aceptar en el momento, junto con cada reconocimiento. El valor óptimo para optimizar el ancho de banda disponible (BW) es [7,9]:

$$\text{Ventana} = \text{RTT} * \text{BW} \tag{4}$$

2.1.3. Multiplexación.

Permite que varias aplicaciones de la misma máquina empleen TCP mediante el uso de puertos.

2.1.4. Orientado a Conexión.

TCP utiliza ACKs como un método de reconocimiento y de confirmación, para conocer que la información ha llegado correctamente a su destino, y de esta forma brindar fiabilidad al protocolo. TCP inicializa y mantiene cierta información sobre el estado de flujo de datos [9].

2.1.5. Orientado a Flujo de Octetos.

La información que es transferida en TCP entre sus aplicaciones es un flujo de octetos sin marca, cualquier extremo de la conexión puede inyectar un volumen de tráfico y su receptor lo recibe con la misma secuencia de datos sin importar el número de segmentos necesarios para su transferencia.

2.1.6. Transferencia con Almacenamiento.

Es la capacidad de TCP de almacenar o dividir los octetos necesarios para conformar el o los datagrama(s) para poder ser enviados a la red.

2.2. CONEXIÓNES EN TCP.

Las conexiones TCP se componen de tres etapas: establecimiento de conexión, transferencia de datos y fin de la conexión. Para establecer la conexión se usa el procedimiento llamado negociación en tres pasos (*3-way handshake*). Al establecerse la conexión hay un intercambio de parámetros entre el transmisor y el receptor como el número de secuencia y el tamaño de ventana [8,12].

2.2.1. Establecimiento de la Conexión.

Para el establecimiento de la conexión es necesario que un socket de un determinado puerto se mantenga escuchando en espera de una nueva conexión (apertura pasiva), al otro lado de la conexión se realiza una apertura activa que se encarga de enviar un segmento SYN al servidor, este lado se encarga de responder la petición con un paquete SYN/ACK, para finalizar el cliente debería responder con un ACK, completando así la negociación en tres pasos (SYN, SYN/ACK y ACK) y la fase de establecimiento de conexión.

2.2.2. Fin de la Conexión.

La etapa de finalización se efectúa cuando cualquier aplicación ya no tiene más datos para transmitir, para realizarlo envía un segmento FIN, el otro extremo al reconocerlo le confirma a la aplicación con un ACK, diciéndole que ya no hay más datos, la figura 7 muestra estos intercambios de mensajes. En ella, además, se identifican los diferentes estados en los que se encuentra el TCP.

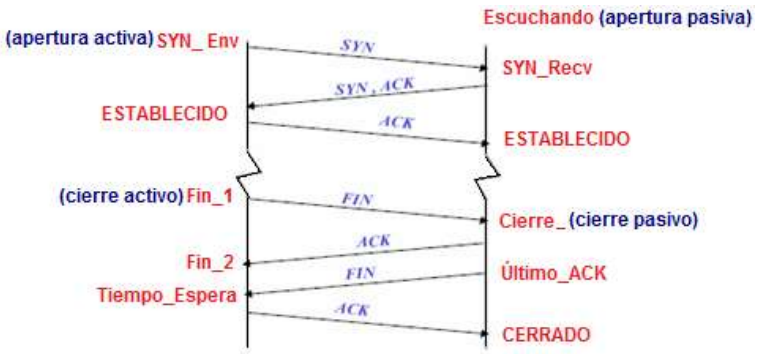


Figura 7. Intercambio de mensajes para el inicio y el cierre de la conexión en TCP [13].

2.3. TAMAÑO DE LA VENTANA TCP.

El tamaño de ventana de un receptor le permite informar al transmisor del número de paquetes que es capaz de recibir en el búfer de recepción sin que se pierdan datos, el tamaño de ventana durante la conexión puede cambiar dependiendo del ancho de banda disponible y del RTT calculado en el momento [6,9].

2.3.1. Formato del Segmento TCP.

El segmento es la unidad establecida de transferencia de datos entre los niveles de transporte, mediante este se puede enviar y recibir reconocimientos, transferir datos, informar sobre el tamaño de ventana y además se realiza los procesos de conexión. El formato de un segmento TCP se muestra en la figura 8.

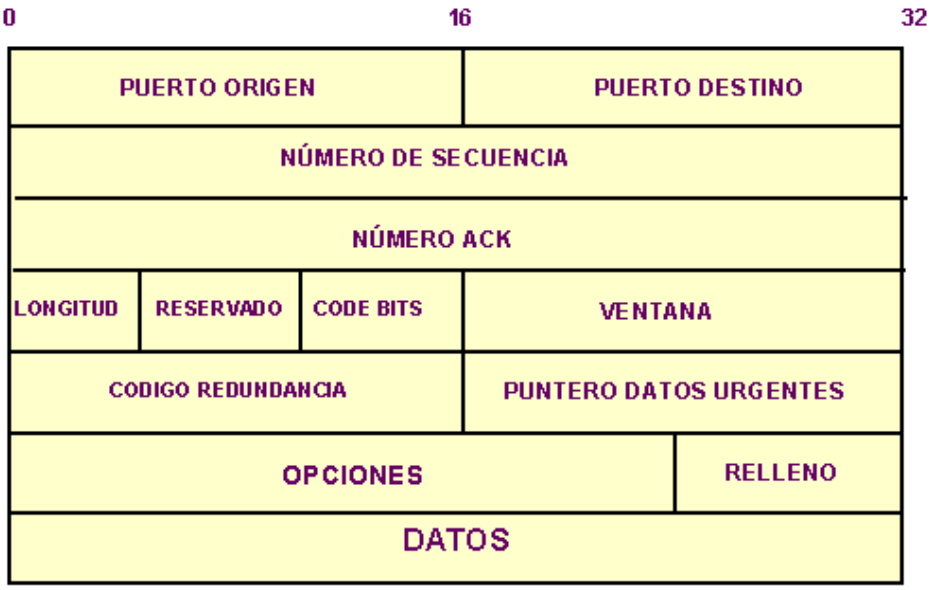


Figura 8. Formato del segmento TCP [9].

2.3.2. Control de Flujo de TCP, Control de Congestión y Recuperación de Error.

Con el objetivo de mejorar la eficiencia de TCP y evitar generar una cantidad inadecuada de tráfico, el protocolo TCP emplea los siguientes cuatro algoritmos [4,9]:

- Inicio lento (*Slow Start*).
- Evasión de congestión (*Congestion Avoidance*).
- Retransmisión rápida *(Fast Retransmit).*
- Recuperación rápida *(Fast recovery)*.

Estos algoritmos se encargan de acordar la cantidad de datos que pueden ser inyectados en la red y para la retransmisión de segmentos perdidos en la red. Durante la transmisión del protocolo TCP se emplean dos variables las cuales se encargan de gestionar el estado de control de congestión [6,12], a continuación se describirá de manera general los mecanismos de control de flujo y de congestión del protocolo TCP.

Ventana de congestión (*cwnd*): Es el límite máximo para la cantidad de datos que el transmisor puede enviar a la red antes de recibir el reconocimiento de un ACK, su valor se encuentra restringido por la ventana de congestión del receptor.

Umbral del Inicio lento (*ssthresh*): Esta variable es la encargada de decidir cuál es el tipo de algoritmo que se utiliza para aumentar el control de ventana. Su valor inicial es el tamaño de ventana advertido por el receptor y se establece cuando se detecta congestión.

Si *cwnd* ≤ *ssthresh*, se utiliza el algoritmo de inicio lento.

Si *cwnd* > *ssthresh***,** se utiliza el algoritmo de evasión de congestión.

2.3.2.1. Inicio lento:

Este algoritmo se emplea para incrementar gradualmente el tamaño de ventana de TCP cuando se inicia la conexión, para evitar la congestión en la red y la pérdida de paquetes. En la medida que el receptor reciba los reconocimientos en la red, podrá inyectar el mismo número de paquetes que son reconocidos y de esta forma alcanzar el máximo tamaño de ventana (el advertido por el receptor).Este algoritmo es ineficiente para transferencias cortas si se compara con el producto por retardo por ancho de banda en la red [6,7].

Al inicio de una conexión TCP o al reiniciarse debido a la pérdida de paquetes, el valor de *cwnd* toma el de un segmento, y por cada ráfaga de segmentos reconocida se duplica el valor del *cwnd* o es lo mismo decir que por cada ACK reconocido, el valor del *cwnd* se incrementa en un segmento. A continuación se muestra el proceso del algoritmo mediante la figura 9.

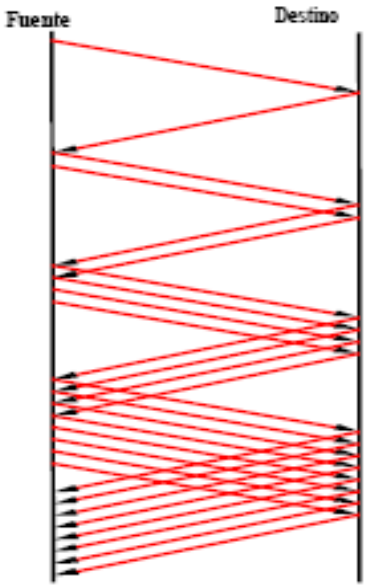


Figura 9. Inicio lento [13].

El tiempo para alcanzar la máxima velocidad en ausencia de pérdidas está dado por la ecuación 5:

$$\text{Tiempo de Inicio Lento} = RTT * \log_2 W \quad (5)$$

2.3.2.2. Evasión de congestión.

Es un algoritmo que está diseñado para procurar evitar congestión en la red, y se adapta a las condiciones de la red para disminuir o incrementar el tamaño de ventana.

Las pérdidas de paquetes en TCP ocurren más frecuente por pérdida por congestión que por pérdida de bits dañados [13], por lo tanto esta pérdida de paquetes es señal para el transmisor de que el *timeout* a expirado y la recepción de ACKs duplicados, en ese caso el algoritmo de evasión de congestión asigna la mitad del valor actual de *cwnd*, y se comienza nuevamente con el crecimiento exponencial del algoritmo de inicio lento hasta alcanzar el nuevo valor de *cwnd*.

Cuando en la red no se ha producido pérdida de paquetes y ya se alcanzado el tamaño actual de ventana, se procede a hacer un incremento lineal el cual permite aumentar la capacidad de la red, el valor del *cwnd* se incrementa en 1/*cwnd* por cada ACK que llegue al receptor, este es un algoritmo más conservador [6, 13].

2.3.2.3. Retransmisión rápida.

Este algoritmo se origina bajo la necesidad de aprovechar los recursos de la red, y evitar activar el algoritmo de inicio lento cuando un segmento no llega al destino [6,13].

El algoritmo se aprovecha del principio de que cada vez que llega un paquete al receptor este responde con un ACK, sin embargo, en la red ya sea por problemas de congestión, expiración del *timeout* o simplemente cuando un paquete llega fuera de orden (debido a que segmentos anteriores no han sido reconocidos), TCP envía un reconocimiento duplicado, el cual consiste en informar al transmisor que se ha recibido un paquete fuera de orden, lo que le sugiere que un paquete enviado con anterioridad se perdió y entonces el receptor envía el anterior ACK reconocido, aunque pudo haber ocurrido que el segmento perdido haya sufrido un retraso, la red espera hasta tres ACKs duplicados para retransmitir el paquete nuevamente y no esperar a que se expire el temporizador del segmento.

Por lo general este algoritmo puede enviar sólo un segmento por ventana de datos enviada, es decir si ocurren más segmentos perdidos deben esperar a que se expiren sus RTOs, lo que conlleva a que TCP regrese con el inicio lento. A continuación se muestra en la figura 10 como opera el algoritmo de retransmisión rápida.

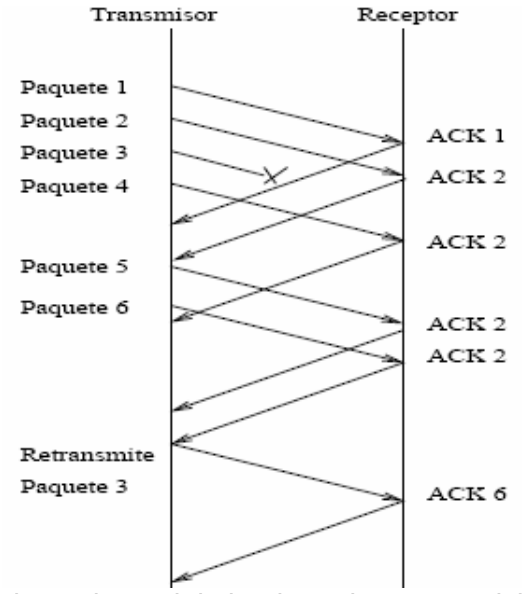


Figura 10. Funcionamiento del algoritmo de retransmisión rápida [13].

2.3.2.4. Recuperación rápida y retransmisión rápida

Este algoritmo hace una mejora a la combinación del algoritmo de inicio lento y evasión de congestión cuando ocurre la pérdida de un segmento. Cuando el algoritmo de retransmisión rápida se activa por congestión, en vez de reducir el tamaño de segmentos en la red, espera la llegada del tercer ACK duplicado para retransmitir el segmento perdido y se fija el valor del *ssthresh* a la mitad del valor actual de *cwnd* y además se asigna el valor de *cwnd* al valor del *ssthresh* mas tres [14].

En la figura 11 se indica la evolución de la ventana de congestión para los casos de inicio lento, evasión de congestión, y retransmisión rápida conocido como TCP Tahoe (línea verde) y la versión de TCP Reno (línea roja) la cual incorpora los mecanismos anteriores más la retransmisión rápida. Para el ejemplo el valor del *ssthresh* se ha fijado en 32 segmentos, en las dos versiones de TCP se presenta el mismo comportamiento en las fases de inicio lento y evasión de congestión siempre y cuando no se produzca ninguna pérdida, cuando el tamaño de la ventana de congestión se encuentra en 40 segmentos se presenta una pérdida, para ese instante las dos versiones de TCP anteriormente nombradas presentan un comportamiento diferente, como se puede ver en la figura 11 TCP Reno hace un mejor uso del ancho de banda de la red [7,13,14].

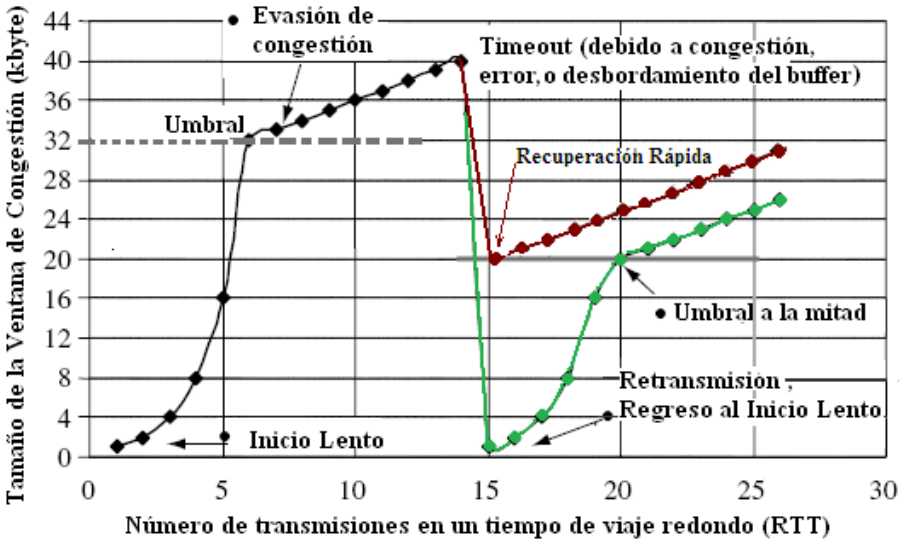


Figura 11. Ventana de congestión de TCP Tahoe y TCP Reno [7,13].

3. MEJORAS AL INICIO LENTO EN REDES SATELITALES.

Como se ha mencionado en este capítulo las características de los entornos satelitales al implementar el protocolo TCP como mecanismo de transporte, han hecho que su rendimiento sea muy pobre teniendo en cuenta que este protocolo es robusto para diferentes tecnologías y aplicaciones, tampoco lo hace muy eficaz en este tipo de redes. Los mecanismos de mejora son muchos y muy variados dependiendo de qué tipo de parámetros se modifique en las redes satelitales.

- Se podría aumentar el tamaño del segmento mínimo de los paquetes de transmisión, considerando que está limitado por el umbral del inicio lento, el tamaño de la ventana de congestión y el tamaño del búfer del receptor.
- Mejorar el algoritmo de inicio lento cuando ocurre la pérdida de un segmento para que pueda aprovechar adecuadamente el ancho de banda disponible.
- Mejorar el mecanismo de ACKS, que implicaría un mayor uso del búfer a causa de un mejor rendimiento.
- Mejorar los mecanismos de detección temprana por pérdidas de errores de transmisión y no por congestión de error.

- El aumento del tamaño del segmento mínimo de los paquetes de transmisión Tb, pero está limitado por el umbral del inicio lento, el tamaño de la ventana de congestión y el tamaño del búfer del receptor.
- Mejoras al mecanismo de evasión de congestión. Este presenta problemas similares a los del algoritmo de inicio lento.
- Definir un mecanismo que tenga el conocimiento sobre el tamaño total de datos y el ancho disponible en la red, ya que el ancho de banda es compartido por muchas conexiones TCP y por lo tanto no es justa su asignación de ancho de banda.

Otro problema es que TCP desconoce como la capa IP lleva el segmento a través de Internet, estos paquetes son de tamaño limitado o se subdividen para su transporte.
Los siguientes mecanismos que mejoran el rendimiento de TCP, se basan en condiciones particulares de la red e incluso su implementación realiza algunas modificaciones en el transmisor, en el receptor o más complejos que deben modificar su arquitectura, por consiguiente algunas de estas técnicas no pueden ser aplicables para todas las configuraciones de red satelital y presentan efectos secundarios que son más críticos para algunas aplicaciones.

3.1. TCP para las Transacciones.

Este mecanismo obvia el procedimiento del *handshake*, para configurar la conexión entre dos computadores, cuando ya ha sido realizado este procedimiento con anterioridad, lo que le permite el envío de paquetes desde el primer segmento enviado, es particularmente útil para datos pequeños y para sesiones TCP, aunque para su implementación es necesario hacer modificaciones tanto en el transmisor y el receptor [6,15]. Este tiempo de inicio puede ser eliminado usando las extensiones TCP para las transacciones (T/TCP).

3.1.1. Inicio lento y Acuse de Recibo Retrasado (ACK).

El algoritmo de inicio lento es el encargado de controlar una cantidad adecuada de datos en el comienzo de la comunicación, el incremento de ventana se produce siempre y cuando los nuevos paquetes sean reconocidos, según [9] *TCP es un protocolo autosincronizado ya que utiliza los reconocimientos como marcas para inyectar nuevos paquetes en la red*. El problema es más crítico al usar ACK retrasados ya que es necesario que un ACK sea reconocido por el transmisor para que otro sea enviado, lo que aumenta el tiempo de transferencia [6].

3.1.2. Una Ventana Inicial más Grande.

Al aumentar el valor inicial del *cwnd* se mejora la capacidad del canal y se reduce el tiempo de su incremento; permitiendo que la ventana de congestión se abra más rápido debido al incremento de los ACKs. Para su implementación es necesario hacer cambios en el *stack* del transmisor TCP. En algunos casos este incremento de ventana degrada el rendimiento ya que no pueden hacer frente a tal ráfaga.
Las opciones de ventanas pueden ser usadas en ambientes satelitales, como también los algoritmos PAWS y RTTM (medidas del tiempo de ida y vuelta) [6].

3.1.3. Finalización del Inicio Lento.

Este mecanismo consiste en la utilización de un algoritmo de paquete-par y la asignación de un RTT medido más apropiado del *ssthresh*, el cual debe ser menor que el tamaño de ventana anunciado por el receptor. Para determinar un tamaño adecuado para el *ssthresh*, el algoritmo observa el espaciamiento entre los primeros ACKs para establecer el ancho de banda del cuelo de botella de la conexión y conjuntamente con el RTT medido se calcula el producto por retardo de ancho de banda y este valor obtenido es el asignado al *ssthresh*. El desafío de este mecanismo es poder encontrar un valor correcto en una red dinámica, ya que disminuiría la pérdida de paquetes. El mecanismo requiere cambios en el *stack* del transmisor de datos de TCP [6].

Este mecanismo presenta inconvenientes en redes de ancho de banda asimétricas donde el valor del *ssthresh* se fijaría muy bajo, lo que produciría un mayor tiempo para alcanzar el tamaño de ventana.

3.1.4. Mejoras a la Pérdida de Recuperación.

En los enlaces satelitales es muy importante poder diferenciar cual fue la causa de la pérdida de segmentos, ya que cada uno de estos tiene un procedimiento diferente, si la red diagnostica pérdida de segmentos por congestión TCP regresa nuevamente con el algoritmo de inicio lento y si hay pérdida por corrupción es debido a bits dañados y el transmisor retransmite el segmento, pero su ventana no se ve afectada. Para TCP es muy importante encontrar un método que los pueda identificar [6,7].

Muchos algoritmos se han desarrollado para que TCP sea más eficiente sin tener que depender del *timeout*, y de esta manera prevenir que TCP se vuelva ineficiente. A continuación se describen algunos algoritmos que solucionan este tipo de problemas.

3.1.5. Reconocimiento Selectivo (SACK).

TCP en algunos casos experimenta pérdida de múltiples segmentos en una misma ventana, cuando esto ocurre los algoritmos de retransmisión rápida y recuperación rápida no son eficientes ya que el transmisor TCP puede enterarse de un sólo paquete perdido por RTT, para solucionar este problema se implementa el mecanismo SACK, el cual puede identificar segmentos de datos perdidos y retransmitirlos en un mismo RTT, agrega información adicional de los números de secuencia recibidos donde se informa al transmisor de los segmentos que no se han recibido para que no sean transmitidos y de esta forma mejorar el rendimiento de la red [6 11].

3.1.6. SACK como Mecanismo de Mejora.

El algoritmo funciona de la misma manera que cuando se activa el algoritmo de recuperación rápida, el algoritmo utiliza una variable *pipe* que es una estimación del número de segmentos pendientes en la red, esta variable se incrementa por cada nuevo segmento o por cada segmento que es retransmitido, el valor de *pipe* puede ser enviado siempre y cuando su valor sea menor que *cwnd*. El valor de *pipe* se decrementa en uno cuando el transmisor recibe un paquete ACK duplicado con la opción SACK informando que un nuevo dato ha sido recibido por el receptor [16].

Este algoritmo se implementa en el *stack* del transmisor TCP, sin embargo, se basa en la información generada por el SACK del receptor [6].

3.1.7. Control de Congestión ACK.

En los mecanismos que se han mencionado de control de congestión ninguno hacen referencia al control de flujo de paquetes de ACKs, el problema se presenta en redes asimétricas donde el trafico de los ACKs puede sobrecargar el enlace de retorno y en algunos casos el rendimiento de un enlace de alta velocidad es limitado por el flujo de acuses de recibo devueltos por el transmisor. El problema se agrava cuando estos flujos de reconocimiento son descartados por los *routers*, ya que no manejan mecanismos de priorización, además por que el *router* limita la longitud de la cola al contar paquetes y no bytes implicando que los ACKs se han desechados [6].

3.1.8. Filtrado ACK.

Es una técnica que viola el principio extremo a extremo de TCP y requiere la modificación de operación en los rotures, se basa en la acumulación de reconocimientos almacenados en el router del enlace de retorno. Al recibir un segmento de confirmación, el *router* explora los ACKs que sean redundantes de la misma conexión para sólo incluir el reconocimiento más reciente y eliminar los anteriores; el *router* no almacena información de estado, pero sí tiene la necesidad de implementar los procesos adicionales requeridos para encontrar y eliminar los segmentos de la cola a partir de la recepción de un ACK.

3.1.9. Notificación Explícita de Congestión (ECN).

La ECN permite que los *routers* notifiquen congestión a los transmisores sin que hayan ocurrido segmentos perdidos. Hay dos tipos de notificación explicita de congestión [7]:

BECN (Notificación explícita de congestión hacia atrás): Un *router* utiliza BECN para transmitir mensajes que informan el estado de congestión directamente al transmisor, lo que le indica al emisor que debe reducir el tamaño de ventana.

FECN (Notificación explícita de congestión hacia adelante): En esta notificación los *routers* marcan segmentos de datos cuando la congestión es inminente, pero solo para el reenvío de segmentos. El receptor le envía después la información de congestión al emisor en el paquete ACK.

Este mecanismo para su implementación requiere cambios tanto en el receptor y transmisor, además de una infraestructura para el manejo activo de colas en los enrutadores. ECN es útil para conexiones cortas e interactivas de TCP.

3.1.10. Detectando pérdida por corrupción.

Para TCP es muy importante poder diferenciar cuál fue la causa de la pérdida de segmentos, porque según sea el caso el sistema recurre a un proceso distinto, si es por congestión en la red, TCP utiliza los algoritmos de control de congestión, pero si es pérdida por corrupción (segmentos perdidos por bits dañados) TCP retransmite el segmento dañado sin tener que disminuir su tamaño de ventana [6].

Los segmentos perdidos por bits dañados se producen con mayor frecuencia por la intervención de los *routers*, cuando a nivel de enlace se detecta mediante el mecanismo de *checksum*, una solución a este problema sería incorporar códigos de corrección de errores hacia adelante FEC en el enlace satelital, aunque estos requieran un hardware adicional y utilizan un mayor ancho de banda para transmitir la señal, esto permitiría corregir cierto número de errores de transmisión sin necesidad de una nueva transmisión [6,7].

3.1.11. Mejoras a la fase de evasión de congestión.

El proceso de evasión de congestión, en ausencia de pérdida, consiste en agregar un segmento por cada RTT a la ventana de congestión, este procedimiento no es el más adecuado cuando múltiples conexiones con RTTs diferentes comparten un mismo enlace, donde cada conexión obtiene una pequeña fracción del ancho de banda del enlace [6,16]. Para solucionar este problema se emplea un encolamiento justo en el búfer de TCP y en los *routers*. Para ello según [6] se pueden implementar dos alternativas al mecanismo de evasión de congestión en el transmisor TCP:

a) El incremento de una tasa constante durante la fase de evasión de congestión, es una forma de corregir el prejuicio en contra de las conexiones con largos RTTs, sin embargo se necesitan algunos estudios para determinar cuál es esa tasa constante de incremento.
b) El incremento de una tasa lineal, para un determinado umbral se adiciona K segmentos a la ventana de congestión por cada RTT. Se podría usar en redes con altos RTTs en un ambiente heterogéneo.

BIBLIOGRAFIA

[1] R. Calero, Sistemas de Comunicaciones Satelitales. Documento PDF disponible en: http://materias.fi.uba.ar/6679/apuntes/Redes_Satelitales_v2.pdf. Consultado 25 Octubre del 2008.

[2] L. Virues, Caracteristicas de las Comunicaciones por Satelite. Documento PDF disponible en:
http://www.monografias.com/trabajos11/caracsat/caracsat.shtml. Consultado 24 Octubre del 2008.

[3] OPCIONES ORBITALES, Universidad Politécnica de Valencia, España. Documento disponible en:
http://www.upv.es/satelite/trabajos/pract_6/opciones.htm. Consultado 25 Octubre del 2008.

[4] Tipos de sistemas de satélites, Universidad Politécnica de Valencia, España. Documento disponible en:
http://www.upv.es/satelite/trabajos/pracGrupo17/sistemas.html. Consultado 25 Octubre del 2008.

[5] Redes de Acceso de Banda Ancha en Navarra, Universidad Pública de Navarra, España. Pagina Web disponible en:
http://www.unavarra.es/organiza/etsiit/cas/estudiantes. Consultado el 14 de Noviembre del 2008.

[6] Z. Sun, *Satellite Networking: Principles and Protocols*, Editorial John Wiley & Sons, Universidad de Surrey, Inglaterra, Oct. 2005. Consultado 10 Septiembre del 2008.

[7] M. Graca, Internet vía satélite, Universidad del Cauca, Colombia, 2000.

[8] D. Brown, *Handbook on Satellite Communications (HSC),* 3 Edición Wiley.

[9] "2 EL Protocolo TCP ".Universidad Politécnica de Cataluña, España, 2002. Documento PDF disponible en:http://www.tesisenxarxa.net/TESIS_UPC/AVAILABLE/TDX-1222106-164746//04AMCA04de15.pdf. Consultado 21 Noviembre del 2009.

[10] Redes T6, El Protocolo TCP/IP, Universidad de Oviedo", España, 2003. Documento PDF disponible en: http://www.isa.uniovi.es/docencia/redes/Apuntes/tema6.pdf. Documento consultado el 20 de febrero del 2009.

[11] J. Cañas, Sistema de Comunicaciones Retransmisión Adaptativa en el Protocolo TCP, España, 13 de Octubre del 2005. Artículo PDF disponible en http://www.inf.utfsm.cl/~jcanas/ramos/SistemasCom/Apuntes/tema4-2.pdf. Consultado el 20 de febrero del 2009.

[12] J. Postel, Ed. *Transmission Control Protocol*. RFC 793, Sep. 1981. Documento disponible en:
http://www.faqs.org/rfcs/rfc793.html. Consultado 19 Septiembre del 2008.

[13] M. Allman, V. Paxson, and W. Stevens, *TCP Congestion Control*, RFC 2581, 1999. Documento disponible en: http://www.faqs.org/rfcs/rfc2581.html. Documento consultado 19 Septiembre del 2008.

[14] SATELITES, Documento PDF disponible en: http://usuarios.lycos.es/araure/satelites1.htm . El documento fue consultado el 20 Octubre del 2008.

[15] D. Roddy, *Satellite Comunications*, Edit McGraw-Hill.

[16] L. Wu, F. Peng, y V. Leung, "*Dynamic Congestion Control for Satellite Networks Employing TCP Performance Enhancement Proxies",* Universidad de British Columbia, Canada, 2004. Artículo de Tesis disponible en: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/9435/29943/01368363.pdf?arnumber=1368363. Consultado 4 Noviembre del 2008

ANEXO B

MECANISMOS DE CONTROL DE FLUJO TCP Y MECANISMOS DE LA CAPA DE ENLACE.

1.0. MECANISMOS TCP

1.1. TCP VENO.

TCP Veno es la combinación de las versiones de TCP Reno y TCP Vegas, se caracteriza principalmente por utilizar el mecanismo de TCP Vegas para conocer la disponibilidad del ancho de banda pero ya no para ser un mecanismo proactivo que le permita modificar el tamaño de ventana para prevenir la congestión sino más bien como un indicador de congestión para poder determinar que si la pérdida fue ocasionada por la congestión o por un error en el enlace [1,2].

TCP Veno implementa casi los mismos algoritmos de TCP Reno pero con algunas modificaciones, la principal diferencia está en la manera como se implementa los algoritmos de incremento aditivo y decremento multiplicativo. Con respecto al algoritmo de inicio lento es idéntico al de Reno, dentro de la fase de evasión de congestión el incremento aditivo se presentan algunos cambios al utilizar el algoritmo de estimación del ancho de banda disponible de TCP Vegas [2].

Las ecuaciones 1, 2 ,3 y 4 de TCP Vegas son las mismas que utiliza Veno, hay que tener en cuenta que si el RTT es mayor que BaseRTT se presenta una conexión de cuello de botella en la que permanentemente se acumulan los paquetes, por ello el valor de RTT es el asignado en la ecuación 5, donde N es el retraso en la cola del enrutador [1].

$$\text{Diff} = (\text{Esperado} - \text{Actual})\ \text{BaseRTT} \tag{1}$$

Donde $\text{Esperado} = \dfrac{\text{cwnd}}{\text{BaseRTT}}$ (2) y $\text{Actual} = \dfrac{\text{cwnd}}{\text{RTT}}$ (2)

por lo tanto

$$\text{Diff} = cwnd\ \frac{\text{RTT} - \text{BaseRTT}}{\text{RTT}} \tag{3}$$

La ecuación 3, Diff indica el número de paquetes que se encuentran atrasados en la red, a partir de este valor se calcula el nivel de congestión y se actualiza el tamaño de ventana de la siguiente manera [3,4].

$$cwnd = \begin{cases} cwnd + 1; \text{si Diff} < \alpha \\ cwnd - 1; \text{si Diff} > \beta \ \ldots \\ cwnd; \ \text{En otro caso} \end{cases} \tag{4}$$

$$\text{RTT} = \text{BaseRTT} + \text{N/Actual} \tag{5}$$

y por lo tanto

$$N = Diff * BaseRTT \quad (6)$$

Durante la fase de evasión de congestión, el algoritmo modificado de incremento aditivo de Veno hace uso del valor de N para poder conocer la utilización del ancho de banda disponible de la red, de la siguiente manera:

if N < β, la red no está completamente utilizada.

Se fija *cwnd* al valor de *cwnd* + *1/cwnd*; el cual se aumenta por cada nuevo ACK recibido.

else if N ≥ β, el ancho de banda disponible está plenamente utilizado, entonces se incrementa más lento la ventana de congestión.

Se fija *cwnd* + 1/*cwnd*; se aumenta en uno por cada otro nuevo ACK recibido, que es lo mismo decir que *cwnd* se incrementa en uno por cada dos RTTs.

Cuando se produce la pérdida de un segmento Veno hace una pequeña modificación a los algoritmos de retransmisión y recuperación rápida de Reno, utiliza el valor de N para diferenciar el tipo de pérdida que presenta la red [5,6]; en vez de ajustar el valor del ssthresh después de la pérdida de un segmento como en Reno, TCP Veno hace lo siguiente:

if (N <β) pérdida aleatoria se debe a los errores de bits producidos en el enlace.

Se asigna $ssthresh = \frac{4}{5} cwnd$

else

Se fija ssthresh = *cwnd* / 2; pérdida por congestión.

Cuando se expira el temporizador de un segmento TCP Veno, este se comporta igual que TCP Reno.

1.2. TCP VEGAS.

TCP Vegas es un mecanismo proactivo, ya que mediante sus mecanismos puede identificar la disponibilidad del ancho de banda y así evita una congestión en la red al determinar su capacidad. Presenta tres modificaciones importantes comparándolo con TCP Reno, las cuales se nombran a continuación [1,7].

El primer cambio se realiza al algoritmo de Inicio lento, el tamaño de ventana de congestión se ajusta en dos paquetes al inicializar el algoritmo y después del *timeout* [8], TCP Vegas con el propósito de encontrar un punto optimo en el funcionamiento sin congestionar la red en el que no se presenten múltiples pérdidas, lo efectúa mediante un incremento de ventana más lento y por ello se ve reducido a la mitad; puesto que desconoce inicialmente el ancho de banda disponible, la manera de realizarlo es midiendo el rendimiento continuamente después de cada incremento para detectar esa disponibilidad de ancho de banda, es decir, Vegas permite un crecimiento exponencial solo por cada RTT [3,9], mientras tanto calcula la tasa real de envío, este aumento se realiza hasta que supere un umbral, para luego dar comienzo a la fase de evasión de congestión.

El segundo cambio y el más representativo se da en la fase de evasión de congestión, para ello hace uso de dos variables que le permitirán ajustar la ventana de congestión del

transmisor en un cierto rango teniendo en cuenta el rendimiento de la red, las cuales son la tasa esperada, que no es más que la relación entre el tamaño actual de la ventana de congestión y el mínimo RTT, y la tasa real que se define como la proporción de la ventana de congestión y el RTT actual. El algoritmo funciona calculando la diferencia entre estas dos tasas para estimar el ancho de banda disponible de la red, en un principio se esperaría que si la red no está congestionada, la tasa de flujo esperada este muy cercana a la tasa de flujo real, a medida que la congestión aumente la tasa de flujo actual será mucho menor que la tasa esperada [10,11].

En la ecuación 3, Diff indica el número de paquetes que se encuentran atrasados en la red, a partir de este valor se calcula el nivel de congestión y se actualiza el tamaño de ventana de la siguiente manera.

El objetivo de Vegas es mantener *cwnd* dentro de los umbrales α y β que indican el estado de la red si esta subutilizado o se encuentra congestionada; y poder estabilizarse en un punto de convergencia dentro de la ventana.

La tercera modificación tiene que ver con el algoritmo de retransmisión, en este caso TCP Vegas no está a la espera de recibir los *n* reconocimientos duplicados para retransmitir un segmento, debido a que TCP Vegas hace un continuo seguimiento del RTT de cada paquete, al recibir un ACK el algoritmo compara con el RTT estimado y si este es mayor reenvía el paquete de inmediato [7], cuando el transmisor recibe un ACK duplicado, este retransmite el paquete más antiguo no reconocido si este ya ha superado un valor del temporizador sensible, está misma situación ocurre después de una retransmisión, generalmente los dos primeros ACKs normales se observan sus valores del *timeout* y si superan a un valor estimado se retransmiten [3,8].

Luego de una retransmisión de paquetes causada por un ACK duplicado, *cwnd* se ve reducido si el tiempo transcurrido desde la última reducción del tamaño de ventana es mayor que el actual RTT; y de esta manera se soluciona el problema de múltiples pérdidas [3].

Los inconvenientes que presenta TCP Vegas tienen que ver con el enrutamiento, ya que no presenta ningún mecanismo que le notifique sobre cualesquier cambio en el estado de la ruta destino, si se presentara alguna modificación podría ocurrir que el RTT de un segmento aumente debido a un cambio de ruta, sin embargo TCP lo interpretaría como congestión y por lo tanto se disminuiría el tamaño de ventana. Otro problema se presenta en las redes asimétricas donde el RTT medido presenta un mayor retardo de propagación en el enlace de menor ancho de banda; este puede ser muy variable por la congestión que se encuentra en el cuello de botella de la red [11].

1.3. TCP *BULK REPEAT* (TCPW BR).

Es un mecanismo el cual implementa tres modificaciones del lado del transmisor, que son: Retransmisión Bulk, retransmisión fija del *timeout* y ajuste a la ventana inteligente, se debe tener en cuenta que estas modificaciones actúan bajo el supuesto que las pérdidas son producto de los errores y no de la congestión, TCPW BR para el caso de congestión se desempeña de igual manera que lo hace TCPW, la forma de determinar qué tipo de pérdida fue la que se presentó en la red es mediante la incorporación del algoritmo de diferenciación de pérdida LDA [12,13].

1.3.1. Retransmisión Bulk.

Se trata de una retransmisión que tiene como objetivo disminuir el impacto que produce el hecho de retransmitir un sólo paquete por la llegada de un ACK parcial, como lo realiza TCP

New Reno, ya que necesitaría múltiples RTTs para recuperarse de estas pérdidas, por el contrario con la retransmisión bulk cuando llega un ACK parcial este retransmite todos los paquetes pendientes en la actual ventana de congestión; y de esta manera se ahorra tiempo y hace un mejor uso del ancho de banda [13,14]. El algoritmo en TCPW se describe a continuación:

```
if ( El transmisor recibe 3 ACKs duplicados o el timeout se vence)
    Se retransmite los paquetes perdidos;
        if (El transmisor recibe un nuevo ACK parcial)
          if (el LDA indica no congestión)
             Retransmite todo los paquetes pendientes y no reconocidos en la actual
             ventana;
             Permanece en la fase de recuperación rápida;
          endif
        else
           Retransmite los paquetes perdidos;
           Permanece en la fase de recuperación rápida;
endif
```

1.3.2. Retransmisión Fija del *Timeout.*

Se emplea con el propósito de funcionar en entornos que presentan altas pérdidas, donde los errores son frecuentes lo que desencadenaría en *timeouts* continuos, repercutiendo en el rendimiento del protocolo; al incorporarse una retransmisión fija del *timeout* se congela el *timeout* de retransmisión y sólo se duplicaría cuando este sea por congestión [13,14]. El algoritmo de retransmisión fija del *timeout* se describe a continuación:

```
if (sender times out) // transmisor producen timeouts
        Retransmite los paquetes perdidos;
        if (LDA indica no congestión)
        El siguiente RTO =se fija el valor timeout;
        else
            if (backoff_factor < 64)
               backoff_factor *= 2;
            endif
              Siguiente RTO = se fija el valor timeout * backoff_factor;
          endif
endif
if (el transmisor recibe un Nuevo ACK)
    if (LDA indica no congestión)
       backoff_factor = 1;
    endif
end
```

1.3.3. Ajuste de Ventana Inteligente.

Considerando que muchas de las versiones de TCP basan sus mecanismos para ajustar *cwnd* y el *ssthresh* cuando se producen congestión, y no consideran el impacto que sería disminuir el tamaño de la ventana de congestión para el caso de una pérdida causada por el error, con el ajuste de ventana inteligente se implementa una tasa de envío más agresiva, para contrarrestar el hecho que muchos paquetes se pierden en el trayecto y de esta manera después de una pérdida por error, no se reduzca *cwnd* incluso cuando sea mayor

que el ssthresh [14,15]. El algoritmo de ajuste de ventana inteligente se describe a continuación:

If (Una pérdida ha sido detectada)
Se ajusta ssthresh = ERE; // Igual que en TCPW.
if (LDA indica no congestión)
No hacer nada, cwnd permanece constante.
else if (ventana de congestión > ssthresh)
Se fija la ventana de congestión al ssthresh;
endif
endif

1.3.4. Algoritmo de Diferenciación de Pérdida.

LDA es el concepto más importante dentro del mecanismo TCPW BR, el transmisor TCP debe determinar si las pérdidas se deben a congestión o al error, el esquema de LDA que se utiliza es la combinación del *spike* y la tasa umbral de intervalo (RGT).

El esquema *Spike* se basa en la medición del RTT, donde hace un seguimiento a los valores máximos y mínimos del RTT para determinar el grado de congestión. Los valores de RTT_{max} y RTT_{min} son actualizados cada vez que una muestra RTT está disponible. La ecuaciones 7 y 8 se establecen para ajustarlo.

$$B_{spikestart} = RTT_{min} + \alpha * (RTT_{max} - RTT_{min}) \quad (7)$$

$$B_{spikeend} = RTT_{min} + \beta * (RTT_{max} - RTT_{min}) \quad (8)$$

Una conexión TCP que entra en el estado *Spike* y si esta no se encontraba antes en ese estado y si el RTT supera el umbral $B_{spikestart}$. De la misma manera, la conexión sale del estado *Spike* cuando esta se encuentra en el estado *Spike* y el RTT cae por debajo del umbral $B_{spikeend}$. Según [16] Spike tiene un mejor funcionamiento en un escenario donde la tasa de error es baja, sin embargo el esquema RGT presenta un mejor desempeño con tasas de error más altas. El esquema de RGT utiliza la diferencia actual de envío dada por cwd / RTTmin con la tasa de estimación elegible (ERE) de TCPW, si es mayor que un determinado umbral que para este caso es el de la ecuación 9.

$$ERE < RGTtresh * (cwnd / RTTmin) \quad (9)$$

Este umbral significa que el transmisor está enviando más información de la que puede recibir y podría ser causa por la congestión [13,14]. A continuación se explica el funcionamiento del algoritmo.

if (una pérdida es detectada)
if ((RGT está en modo de error) || (Spike está en modo de error))
La pérdida es debido a un error;
else
La pérdida es debido a congestión;
endif
endif

1.4. TCP START UP.

TCP Start up es un mecanismo que modifica el inicio lento de TCP Westwood, para ajustar el *ssthresh* basándose en la estimación de tasa elegible de TCPW durante la conexión inicial y después de cada *timeout*. La adaptación de inicio impide una prematura finalización del algoritmo de inicio lento, lo que permite que *cwnd* aumente rápidamente sin incurrir el riesgo del desbordamiento en el búfer y no causar múltiples pérdidas [17,18]. Ajustar un valor adecuado del *ssthresh* inicial es importante, de tal manera que no sea tan bajo, al compararlo con el gran producto por retardo por ancho de banda, lo que implica que *cwnd* no aprovecha adecuadamente el ancho de banda de la red debido a un prematuro inicio de la fase de evasión de congestión; y por el contrario no tan alto para causar múltiples pérdidas en una misma ventana y así evitar *timeout*. El pseudo código del algoritmo se describe a continuación [18,19].

```
if ( 3 ACKS  duplicados son recibidos)
   Cambia a la fase de Evasión de Congestión;
   else ( Se recibe el  ACK );
          if (ssthresh < (ERE*RTTmin)/seg_size); Prueba el ancho de banda
disponible de cada conexión.
        ssthresh =(ERE*RTTmin)/seg_size; Se restablece el ssthresh a un mayor
valor.
      endif
if (cwnd >=ssthresh) // Fase de incremento mini lineal , para evitar desbordamiento.
   Se incrementa  cwnd en 1/cwnd;
else if (cwnd <ssthresh) //Fase de incremento mini exponencial.
       Se incrementa cwnd en 1;
   endif
endif
```

1.5. TCP ADaLR.

TCP con retardo adaptativo y respuesta a la pérdida, abreviadamente llamado TCP ADaLR se implementa para mejorar el desempeño en redes satelitales GEO, caracterizadas por sus grandes retardos de propagación, altas BER, y ACKs retrasados. Sus algoritmos pueden ser aplicados como una extensión a TCP SACK y TCP New Reno [20]; su implementación requiere modificaciones en el transmisor. TCP ADaLR emplea tres mecanismos, que son los mecanismos de incremento de la ventana de congestión y r*wnd*, y el mecanismo de recuperación de pérdida; además utiliza una variable de ajuste ρ, para ajustar la ventana de congestión. La variable ρ se normaliza a un valor de RTO típico para implementaciones TCP de 1 segundo. El componente de ajuste se calcula como lo indica la ecuación 10, donde su valor depende de las muestras RTT, *sampleRTT*.

$$\rho = (\text{sampleRTT s} / 1\text{s}) \times 60 \qquad (10)$$

El valor de 60 se utiliza porque es un valor recomendado para el máximo RTO normalizado en un segundo. El valor ρ oscila entre 1 y 60, definido para los valores mínimo y máximo respectivamente que TCP emplea para las conexiones RTTs que son demasiados cortos y muy grandes. La incorporación del RTT normalizado es muy importante porque permite un incremento más rápido del tamaño de ventana y los ACKs durante el inicio lento en TCP estándar [20]. A continuación se explican los tres mecanismos de esta versión de TCP.

1.5.1. Mecanismo de Incremento Adaptativo de CWND.

La fase de inicio lento se divide en 4 sub fases, que se basan en el actual tamaño *cwnd* y el total de datos pendientes no reconocidos en la red, *flightsize.* En cada subfase el incremento de *cwnd* depende del valor de ρ, y de la presencia o ausencia de pérdidas durante la transmisión.

En este mecanismo se establece un punto de equilibrio para el valor de ρ=15, que se escoge basado en las simulaciones de descarga de un archivo FTP de 50 MB en un enlace satelital ideal, el valor de ρ equivale a un RTT de 250 ms que es un valor apropiado al retardo de propagación en las redes GEO. Si ρ ≥ 15, *cwnd* se aumenta a un valor de (√ ρ / 4) SMSS, para ρ < 15 el incremento de *cwnd* es exponencial como TCP estándar [20,21].

El valor de (√ ρ / 4) SMSS se encuentra en un rango entre 1 y 2 SMS, el cual es propicio para evitar las minirafagas que están expuestas con la opción del ACK retrasado. Si ninguno de los cuatro casos se presentan o simplemente ocurre la pérdida de un segmento; *cwnd* se incrementa como lo hace en la fase de inicio lento de TCP estándar. Para detectar una pérdida se utiliza la variable *snd_recover*, la cual se inicializa en cero al comienzo de la conexión. A continuación se indica el seudo código del mecanismo de incremento adaptativo de *cwnd*.

```
// snd_max = número de secuencia máximo enviado (el número de secuencia más nuevo no reconocido)
// snd_una = número se secuencia del primer segmento no reconocido (el número de secuencia más viejo no reconocido.)
// snd_recover = número se secuencia que indica el final de la rápida recuperación
// acked_bytes = número de bytes reconocidos por un ACK.
flightsize = snd_max - snd_una;
// Fase de Inicio Lento
// El valor inicial del ssthresh es de 64KB y el de cwnd 16KB.
if (cwnd < ssthresh)
    {
        if ((cwnd ≤ ssthresh/4) && (flightsize < rwnd/4))
            se fija sub-fase = inicio lento sub-fase 1
        if ((cwnd > ssthresh/4) && (cwnd ≤ ssthresh/2) && (flightsize < rwnd/4))
            se fija sub-fase  = inicio lento sub-fase 2
        if ((cwnd > ssthresh/4) && (flightsize ≥ rwnd/4) && (flightsize < rwnd/2))
            se fija sub-fase = inicio lento sub-fase 3
        if ((cwnd > ssthresh/2) && (flightsize ≥ rwnd/4) && (flightsize < rwnd/2))
                se fija sub-fase = inicio lento sub-fase 4
    }
```

1.5.2. Mecanismo de Incremento Adaptativo de RWND.

Este mecanismo se encarga de ajustar un límite impuesto a los datos a ser transmitidos por el receptor definido como *rwnd*, el mínimo de *cwnd* o *rwnd* determina la cantidad de datos a ser transmitidos. El incremento de rwnd depende de los valores de ρ, *flightsize*, *cwnd* y de la presencia o ausencia de pérdida. A continuación se describe el mecanismo de incremento adaptativo de *rwnd* [20].

```
if (cwnd < ssthresh)
{
// Inicio lento sub-fase 1
    if ((cwnd ≤ ssthresh/4) && (flightsize < rwnd/4))
```

```
        {
            if ((snd_recover == 0) && (ρ ≥ 15))
                Se incrementa cwnd por (√ρ/4) × SMSS
            else
                // Inicio Lento sub-fase 2
    else if ((cwnd > ssthresh/4) && (cwnd ≤ ssthresh/2) && (flightsize < rwnd/4))
            {
             if ((snd_recover == 0) && (ρ ≥ 15))
                Se incrementa cwnd por (√ρ/4) × SMSS
             else
                El incremento cwnd es exponencial como en TCP Reno.
            }
             // Inicio Lento sub-fase 3
        else if ((cwnd > ssthresh/4) && (flightsize ≥ rwnd/4) && (flightsize < rwnd/2))
                {
                 if ((snd_recover == 0) && (ρ ≥ 15))
                Se incrementa cwnd por (√ρ/4) × SMSS
         else
              Se incrementa cwnd exponencialmente como en TCP Reno
                }
// Inicio Lento sub-fase 4
else if ((cwnd > ssthresh/2) && (flightsize ≥ rwnd/4) && (flightsize < rwnd/2))
{
   if ((snd_recover == 0) && (ρ ≥ 15))
        Se incrementa cwnd en (√ρ/4) × SMSS
else
        Se incrementa cwnd exponencialmente como en TCP Reno.
}
```

1.6. TCP PEACH.

TCP Peach es un mecanismo que hace frente a la degradación del rendimiento en las redes satelitales, e intenta resolver los problemas relacionados con los retardos de propagación y las altas tasa de error de conexión, con el propósito de poder utilizar mejor el ancho de banda disponible. TCP Peach emplea dos mecanismos que son el inicio repentino y recuperación abrupta, remplazando a los algoritmos de inicio lento y evasión de congestión de TCP estándar, TCP Peach utiliza los segmentos *dummy* para juzgar adecuadamente el ancho de banda que se desea utilizar. A continuación se describe el funcionamiento de estos segmentos, que son la herramienta principal para el mecanismo [22,23].

- **Segmento *dummy*:**

Los segmentos *dummy* son copia del último segmento enviado, por lo tanto, no transmiten ninguna información nueva al receptor, lo que los hace de muy baja prioridad, de tal manera que cuando se presenta cualquier congestión en la red son los primeros en eliminarse, su función es probar la disponibilidad de los recursos de la red. En ningún momento estos segmentos causan la disminución del desempeño de los segmentos de datos normales. Para poder diferenciar los segmentos *dummy* de los propios datos, el transmisor establece uno o más de los seis bits no utilizados en la cabecera TCP, así mismo los ACKs de estos segmentos son marcados usando uno o más de los seis bits no utilizados en el encabezado de TCP y se transportan por los segmentos IP de baja prioridad. Para poder implementar el mecanismo se requieren cambios en el lado del transmisor; para verificarlo TCP Peach en el

inicio de una transmisión comprueba que estas modificaciones estén presentes en el receptor y de no ser así se comporta como TCP Reno [24].

- **Inicio repentino.**

Al inicio de una transmisión el transmisor envía un segmento de datos al igual que Reno y RWND-1 segmentos *dummy*, si en la ruta no se presenta congestión se esperaría recibir en el transmisor los ACKS de los segmentos *dummy*, lo que significaría que la red puede aumentar *cwnd* en RWND segmentos después del siguiente RTT. En el caso de presentarse congestión los *enrutadores* eliminarían los segmentos *dummy*. A continuación se indica el algoritmo empleado por el inicio repentino [23,24].

```
Incio_Repentino()
    cwnd=1;
    τ =RTT/rwnd: // rwnd es el máximo valor de la ventana de congestión cwnd.
   send(Data_Segment) ;
   for  ( i=1 a rwnd-1)
        wait (τ );
        send(Dummy_Segment);
   end;
end
```

TCP Peach utiliza la variable *wdsn* para coincidir el comportamiento con TCP Reno, cuando la red está congestionada; *wdsn* se fija a cero al inicio de la conexión, para este valor la ventana de congestión se incrementa en uno, y si *wsdn* es diferente de cero, *wdsn* se disminuye en una unidad y *cwnd* permanece con el mismo valor.

- **Recuperación abrupta.**

Cuando un segmento perdido es detectado por los ACKs duplicados, el algoritmo de recuperación abrupta actúa de igual forma que el algoritmo de recuperación de Reno, ya que desconoce la causa de la pérdida, por lo tanto, el transmisor reduce a la mitad su ventana, es decir *cwnd*o/2 segmentos durante la fase de recuperación abrupta; con el objetivo de conocer cuál fue la causa de la pérdida (por congestión o por errores de corrupción) se aplica luego el algoritmo de recuperación abrupta, transmitiendo un cierto número de segmentos *dummy* n_{Dummy} = $cwnd_o$ para probar la disponibilidad de la red, teniendo en cuenta que el ACK de los segmentos *dummy* será recibido después del ACK del segmento perdido, lo que indicaría que se encontrará en la fase de evasión de congestión, como se observa en la figura 1.

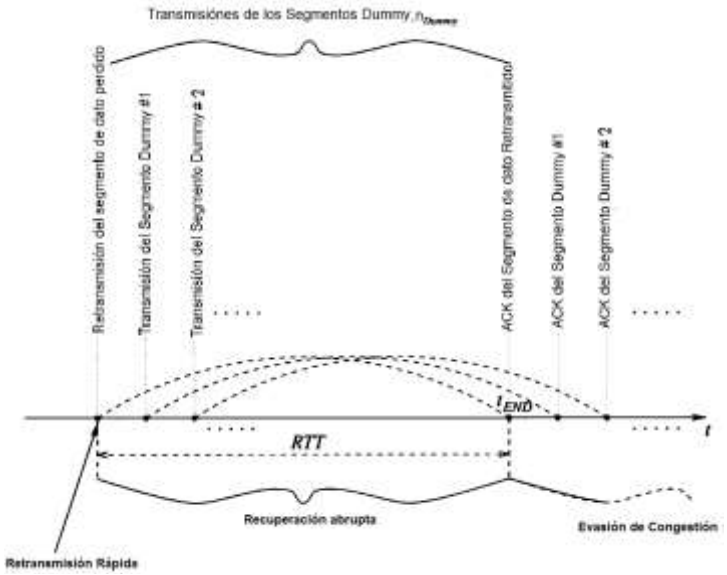


Figura 1. Fase de la recuperación abrupta [24].

Si la pérdida se debe a la congestión sólo se recibirán la mitad del tamaño de ventana, ya que los segmentos *dummy* fueron descartados por el enrutador; en caso contrario el transmisor recibe los primeros *cwnd*o/2 ACKs de los segmentos *dummy*, de los cuales no se puede asegurar nada, por ello wdsn se iguala a *cwnd*o/2 para que el algoritmo no pueda aumentar su ventana de congestión, al recibir el segundo paquete de *cwnd*o/2 segmentos *dummy*, el transmisor aumenta su ventana de congestión en uno por cada vez que reciba un ACK de un segmento *dummy* [22,24]. Se debe tener en cuenta que después de la retransmisión del segmento perdido debido a la congestión de la red, el *enrutador* podrá entregar *cwnd*o/2 segmentos por RTT, por lo tanto la red demorará un RTT para los segmentos de datos, uno más por la mitad de segmentos *dummy* y otro RTT para la otra mitad de segmentos *dummy*, como se observa en la figura 1. A continuación se indica el seudo código del algoritmo de recuperación abrupta

```
cwnd=cwnd/2;  // Fase de recuperación rápida del algoritmo de Reno.
adsn=2adsn;
wdsn=cwnd;
infl_seg=0;
t_Retr= t
END=0 // Inicio de la fase de recuperación abrupta.
While(END=0)
  If (ACK_ARRIVAL)
     If (DATA_ACK_ARRIVAL)
        cwnd=cwnd+1;
        infl_seg= infl_seg+1;
    else if(DUMMY_ACK_ARRIVAL)
          if (wdsn=0)
             cwnd=cwnd+1;
       infl_seg= infl_seg+1;
          else
             wdsn=wdsn-1;
         end;
      end;
  if(cwnd>nackseg)
    While((cwnd>nackseg))
          send (Data_ Segment);
             nackseg = nackseg+1;
```

```
    end;
  else if(adsn>0)
        send (Dummy_ Segment);
       send (Dummy_ Segment);
       adsn=adsn-2;
   end;

if (LOST_SEGMENT_ACKED)
   END=1; // Finalización de la fase de recuperación abrupta.
   cwnd= cwnd- infl_seg;
end;

end;

if(t>tRetr + RTO)
  Slow_Start();
end;
 end;
end;
```

donde *adsn*, es el número de segmentos *dummy* que el transmisor está autorizado a inyectar en la red, *infl_seg* indica la cantidad que fue aumentada *cwnd* durante la recuperación abrupta, t_{Retr} es el tiempo cuando el segmento es retransmitido y END, es una variable booleana que indica cuando finaliza o inicia la recuperación abrupta

Tres aspectos a recalcar del mecanismo son:

- Si en el momento (t_{Retr} + RTO), el ACK del segmento retransmitido aún no ha sido recibido, lo más probable es que el segmento se haya perdido, como consecuencia la recuperación abrupta finaliza y el transmisor comienza con el inicio repentino.
- TCP Peach requiere la incorporación de un sistema de prioridades en los enrutadores, en las redes IPv4 se utiliza IP TOS para este fin, y en las redes que utilizan IPv6 estos ya vienen incorporados con diferentes niveles de prioridad.
- De igual forma como lo hace TCP estándar para TCP Peach los ACKs son un mecanismo de auto sincronismo, para mantener un nivel constante de segmentos en la red.

En la figura 2 se compara, el ajuste del tamaño de la ventana de congestión en el tiempo para TCP Peach y TCP Reno.

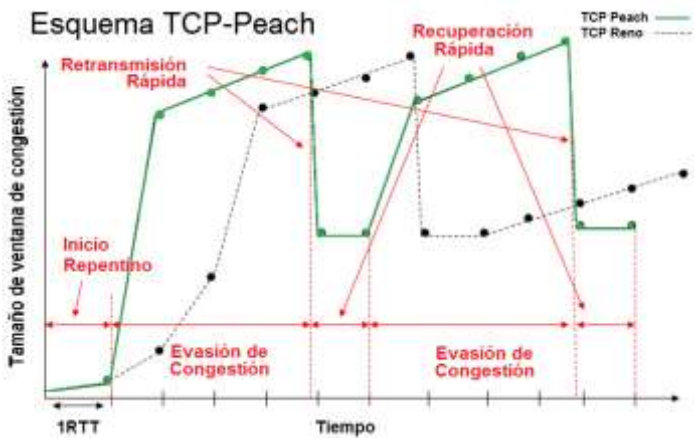


Figura 2. Esquema TCP Peach [57].

1.7. TCP PEACH+.

TCP Peach+ es una mejora a la versión de TCP Peach para redes satelitales IP, hace uso de los algoritmos de ascenso rápido y recuperación acelerada, para contrarrestar los inconvenientes de los retardos de propagación y las altas tasas de error de conexión, además adopta la opción de TCP SACK para obtener un mejor desempeño ante la múltiple pérdida de segmentos en una misma ventana de datos.

Para la estructuración de sus algoritmos se utiliza el concepto de segmento *Nil*, el cual se describe a continuación [26].

1.7.1. Segmentos *Nil*.

Los segmentos *Nil* se utilizan para probar la disponibilidad de los recursos de la red y recuperarse de los errores al mismo tiempo, cuando los ACKs *Nil* llegan al transmisor indican que hay recursos utilizados en la red por utilizar y por lo tanto se aumenta su tamaño de ventana, se diferencia con los segmentos *dummy* de TCP Peach, ya que estos transportan información duplicada en cambio en TCP Peach+ los segmentos transportan información no reconocida que le permite a los receptores recuperar segmentos perdidos. De igual forma que los segmentos *dummy*, los segmentos *Nil* son de baja prioridad lo que les permites ser descartados primero por los enrutadores en momentos de congestión, sin causar ninguna consecuencia en el rendimiento de la red. La forma de diferenciarlos con los segmentos de datos es mediante los bits del encabezado TCP, el receptor lo reconoce y de regreso se notifican ante el transmisor con los ACKs *Nil*; los cuales también se señalizan con uno o más de seis bits que no son utilizados por el encabezado del ACK y se transportan como paquetes IP de baja prioridad.

1.7.2. Algoritmo Ascenso Rápido.

El algoritmo de ascenso acelerado al establecer la conexión, fija *cwnd* a 1, luego se envía el primer segmento y espera un valor de τ = RTT/ *rwnd* para transmitir los rwnd-1 segmentos *Nil*, transcurre un RTT para conocer el estado de la red, si los ACKs de los segmentos *Nil* llegan al transmisor, significa que en la red hay recursos no utilizados y *cwnd* se puede aumentar rápidamente, de lo contrario la red se encuentra congestionada, a continuación se indica el seudo código del algoritmo de ascenso rápido [26].

```
Ascenso rápido ()
  cwnd=1;  τ =RTT/rwnd;
  send(Data Segment);
  for ( i=1 a rwnd-1 )
       wait( τ );
       send(NIL Segment);
  end;
end;
```

Donde RTT es el valor más grande que se obtiene de los receptores y *rwnd* tiene el mismo significado que en TCP Peach.

1.7.3. Algoritmo Recuperación Acelerada.

El algoritmo de recuperación acelerada se utiliza para contrarrestar los altos errores de conexión y múltiples pérdidas de segmentos en la ventana de datos que ocurren en las redes satelitales. Al momento de detectarse una pérdida mediante *ndupack*, con la opción

SACK, se retransmite el segmento perdido y se inicializa la fase de recuperación acelerada que al inicio se comporta como la recuperación rápida de Reno, reduciendo *cwnd* a la mitad, se fija pipe a 2*cwnd* - *ndupacks* +1; la variable pipe se utiliza para determinar el número estimado de segmentos pendientes que se mantienen en la red, se transmiten los *cwnd* segmentos *Nil* durante la recuperación rápida, y se espera recibirlos un RTT después en la fase de evasión de congestión, si el segmento perdido fue por causa de un error del enlace, entonces *cwnd* se incrementa en uno por cada ACK *Nil* que llegan al transmisor, de esta manera un RTT después se recuperaría el tamaño original de la ventana antes de la pérdida del segmento. Si un *timeout* se detecta durante la fase de recuperación rápida, el transmisor da por terminado la recuperación acelerada y ejecuta el ascenso rápido.

Con la opción SACK implementada en TCP Hybla, el transmisor usa una estructura de datos, llamada *scoreboard* para actualizar la información sobre los reconocimientos (ACKed) y el de los segmentos SACKed, el *scoreboard* se limpia al finalizar la recuperación acelerada.

Pipe en el algoritmo de recuperación acelerada toma el valor de 2*cwnd* - *ndupacks* +1, indicando los 2*cwnd* segmentos transmitidos antes de que se detectará la pérdida del segmento, *ndupacks* son los paquetes duplicados que activaron el algoritmo de recuperación acelerada y 1 es por el segmento retransmitido. Durante la recuperación acelerada pipe se disminuye en uno cuando recibe un ACK duplicado, y se decrementa según sea el valor *amountacked* cuando haya recibido un ACK parcial. A continuación se indica el algoritmo de recuperación acelerada [26].

En la tabla 1 se muestra los parámetros utilizados por el algoritmo de recuperación acelerada según [26].

Tabla 1. Parámetros empleados en el algoritmo de recuperación acelerada.

Variable	Descripción
High Ack	Es el número de secuencia acumulado más alto del ACK recibido en un punto fijo.
High Data	Es el número de secuencia más alto de transmisión justo antes que comience la recuperación acelerada.
Duplicate ACK	Se define como un ACK cuyo número acumulado es igual al valor actual de High Ack y también transporta la nueva información SACK del segmento o los segmentos High Ack.
Partial ACK	Es un ACK que aumenta el valor *High Ack*, pero no todos los ACK de todos hasta que se incluya *High Data*.
Recover ACK	Es el ACK para todos los datos hasta cubrir *High Data.*
ndupacks	Es el número de paquetes duplicados los cuales activan las fases de Retransmisión Rápida y Recuperación Acelerada.
Maxburst	Es un parámetro que limita el número de paquetes que se pueden ser enviados en respuesta a una sola entrada ACK, incluso si la ventana de congestión del transmisor permitiría más paquetes para ser enviados.
Amountacked	Es el número de segmentos de datos reconocidos por el receptor, cuando llega un ACK..
adsn	Es el número de segmentos que el transmisor TCP está autorizado a inyectar en la red.
adps	Es el número de segmentos de datos que permiten ser enviados.
nps	Es el número de segmentos de datos que realmente han sido enviados.

Recuperación Acelerada ()

```
    cwnd=cwnd/2;
    adsn=cwnd;
    adps=0;
    END=0;
   pipe = 2cwnd-ndupacks+1;
    while (END=0)
            if (ACK ARRIVAL)
                if(Duplicate ACK)
                   pipe=pipe-1; // Se decrementa pipe porque un segmento ha
abandonado la red;
                   update scoreboard; // Se actualiza el scoreboard;
                end;
                else if (Partial ACK)
                        pipe=pipe-amountacked;
                        update HighAck; //  Se actualiza el HighAck;
                        update scoreboard; // Se actualiza el scoreboard;
                end;
                  if (NIL ACK)
                     cwnd=cwnd+1;
                     adsn=adsn-1;
                     update scoreboard; // Se actualiza el scoreboard;
       end;
     if (Recover ACK)
        update HighAck;
        clear scoreboard; // Se limpia scoreboard;
        END=1; // Finaliza la fase de recuperación acelerada;
    end;
    adps=cwnd-pipe;
    nps=min(maxburst, adps)//Se limita el número de segmentos que se pueden
enviar por ACK;
         if (nps >0)
            send nps // Se envía nps paquetes faltantes o nuevos paquetes;
            pipe=pipe+nps;
         end;
  else if (adsn <0)
            send a NIL packet;
           adsn=adsn-1;
       end;
   end;
  end;
end;
```

2.0. MODIFICACIÓN A LA ARQUITECTURA INTRODUCIENDO NODOS INTERMEDIOS EN LA RED E IMPLEMENTACIONES ENTRE LOS NODOS ESTÁNDAR O SOLUCIONES PROPIETARIAS.

2.1. CARACTERÍSTICAS DE LOS PEP.

2.1.1. PEP en la Capa de Transporte.

Operan por encima de la capa de transporte, tratan de mejorar la disponibilidad, la confiabilidad y el desempeño de la red. Un ejemplo de estos son los proxies cache y los agentes de transferencia de correo MTA.

- **Distribución.**

Una aplicación PEP puede ser integrada o distribuida, para el caso de una PEP integrada se hace referencia a un sólo PEP implementado en un único nodo, en caso de ser distribuido se hace uso de más de dos PEPs distribuidos, un ejemplo claro de este tipo son las redes satelitales ya que estas se ubican en los extremos del satélite.

- **Implementación simétrica.**

La simetría de los PEP también puede ser de dos tipos simétrica o asimétrica, según el tipo de interfaz que se emplee. Los PEPs simétricos hacen uso de cualquier interfaz, no tienen definido un canal específico para su uso. Los PEPs asimétricos si dependen de qué tipo de interfaz se utilice, por lo general se emplean cuando las características difieren a ambos lados de los enlaces o simplemente por el tráfico asimétrico del protocolo; un ejemplo de este tipo son las redes satelitales VSAT en las cuales se toma en cuenta un canal de flujo de datos TCP y un canal de retorno para el flujo ACK.

- **Conexión dividida.**

Es una de las propiedades más importantes que tiene el uso de PEPs en la arquitectura física de la red; con la conexión dividida se espera obtener un mejor rendimiento y optimización de los protocolos que funcionan en dichos segmentos, ya que al mantenerse separado por un agente intermedio los segmentos se aíslan de las características que degradan el rendimiento como lo es el retardo de propagación, las altas BER, los problemas de conectividad, el *handover* entre otras características que al no estar presente, afectarían a todo el enlace.

- **Transparencia.**

La transparencia de PEP se ve relacionado a la manera como los proxies modifiquen o no modifiquen los extremos del sistema. Al requerir algún cambio en unos de los extremos o en ambos, no se está siendo transparente por lo menos en la capa que este se implemente. La transparencia para PEP según [27] puede ser observada desde cuatro puntos de vista, con respecto a los sistemas finales, a los parámetros de transporte, a las aplicaciones y a los usuarios.

2.2. MECANISMOS PEP.

Los mecanismo PEP, son los diferentes métodos que se podrían implementar en una solución PEP para optimizar el rendimiento en la red y obtener un mejor uso de los recursos de la red para combatir con ciertas condiciones específicas en los enlaces satelitales [11,28].

2.2.1. Espaciamiento ACK.

Esta técnica se aplica en entornos donde los ACKs tienden agruparse cuando llegan a la fuente TCP, debido al gran producto por retardo por ancho de banda de la red, provocando ráfagas indeseables en el segmento de datos. Los búferes encargados de almacenar la

información de los paquetes de la red, cuando llegan largas ráfagas tienen que descartar o eliminar muchos segmentos, activando así los mecanismos de recuperación de errores por tiempos prolongados. Al implementarse el espaciamiento ACK, se suaviza el flujo de los ACKs.

2.2.2. Reconocimientos Locales.

Con el propósito de agilizar más rápidamente la ventana de congestión, en los entornos que presentan un gran producto por retardo por ancho de banda, se emplean los reconocimientos locales en los nodos de las PEP, de esta manera le tomará mucho menos tiempo utilizar plenamente los recursos de la red al transmisor TCP. Cualquier tipo de PEP que emplea este mecanismo tiene la responsabilidad de retransmitir cualquier segmento que se haya perdido en la red, después que PEP lo haya reconocido.

2.2.3. Retransmisiones Locales.

Se emplean como un mecanismo para responder más rápido ante la pérdida de errores localmente en los PEPs. Algunos tipos de PEP lo realizan mediante reconocimientos negativos (NACK), reconocimientos duplicados o entre otras opciones que sirvan para activar el mecanismo de recuperación de pérdida.

2.2.4. Filtrado ACK y Reconstrucción.

Se presenta como una solución en entornos asimétricos donde la relación del enlace ascendente con respecto al descendente es muy alta; lo que implica pérdidas en el canal de flujo del ACK, ocasionando congestión, obteniéndose una reducción de la tasa de envío del transmisor. Este mecanismo hace un filtrado ACK, que consiste en una disminución de los ACKs de un lado del enlace y lo reconstruye eliminando los ACKs en el lado de recepción del enlace.

2.2.5. *Tunelling*.

Es un mecanismo PEP que encapsula los mensajes con el fin de obligar a atravesar a estos sobre un enlace en particular. Un agente PEP que se encuentra instalado al final del "túnel" elimina la envoltura de encapsulación antes de entregar los mensajes al receptor final.

2.2.6. Compresión.

Es una solución muy practica para emplearse en situaciones donde el ancho de banda es limitado, y se desea reducir el número de bytes que se envían a la red. Los algoritmos de compresión que usan los PEPs son para aplicaciones específicas. El uso de los procesos de compresión y descompresión podrían producir un retardo adicional en el procesamiento de datos de información en el PEP.

2.3. INCONVENIENTES DE PEP.

Aunque el objetivo de PEP sea optimizar el rendimiento puede incurrir en problemas de seguridad. Los nodos PEPs están ubicados de tal forma en la red, que son puntos vulnerables a ataques de seguridad, como son los ataques de denegación de servicio y hombre en el medio. Los PEPs, por lo tanto requieren protegerse de estos ataques, debido a

la exposición de los datos y sobre todo si se emplean en redes satelitales, donde su acceso al medio se ve compartido por múltiples usuarios.
La seguridad que emplean los PEP se aplican por lo general por encima de la capa de transporte, aunque sean muy pocos las alternativas que se utilicen. La seguridad de la red se ve implementada por la capa IP con IPsec, el inconveniente de utilizar IPsec con TCP PEP, es que los encabezados de TCP están cifrados de tal forma que la fuente origen y de recepción sólo conocen la clave, lo que evidencia un problema para PEP, ya que este se encuentra en un nodo intermedio en el que necesita leer y modificar los encabezados de TCP [17,27].

2.4. SOLUCIÓN AL CONFLICTO IPsec CON TCP PEP.

Las soluciones para que IPsec funcione con PEP no son de todas seguras, y por lo general pueden ser muy complejas en su implementación. La solución más sencilla es que IPsec confíe en los nodos intermedios PEP, de esta forma IPsec actúa por separado entre cada uno de los sistemas y las PEP. Una solución que se implementa hoy en día es mediante el uso de múltiples capas IPsec, la cual consiste en la división de la carga útil IP en dos, el encabezado TCP y los datos TCP. Dentro de los datos TCP existiría un sistema de seguridad en el que sólo estaría conformado por la fuente origen y destino, donde compartirían una llave, el encabezado manejaría una llave compartida entre los extremos del sistema y los nodos intermedios de confianza, de esta forma TCP PEP no tiene ningún inconveniente al emplear IPsec en la red, el problema estaría en que no se sería transparente con IPsec [17,27,28].

2.5. MITIGACIÓN A LA CONFIABILIDAD DE EXTREMO A EXTREMO DE PEP.

Uno de los principales inconvenientes de los PEPs es tal vez su forma de implementarse, ya que en algunos casos se viola la semántica de extremo a extremo de TCP. Con el propósito de mantener esta semántica se utiliza una arquitectura de red de retardo tolerante DTN, que consiste en hacer subdivisiones a una región heterogénea dentro de sub-regiones homogéneas basándose en la introducción de una nueva capa, la cual se llama *bundle*, que es la encargada de controlar la semántica de extremo a extremo, esta subdivisión permite acoger cualquier versión de TCP optimizada para mejorar las características de una subred en particular. Esta es una opción muy prometedora, sin embargo para su funcionamiento requiere alterar el *stack* del protocolo [28,29].

2.6. CONSIDERACIONES PEP.

Al querer utilizar PEPs dentro de una red, se tienen que considerar muchos parámetros que pueden afectar el rendimiento, la usabilidad y el costo del enlace; para su elección hay que tener muy claro en qué aspectos de la red esta podría beneficiarse y que aspectos estarían vulnerables con su uso, a continuación se nombran algunos, que podrían generar un grave impacto en el uso de la red [27].

- Mecanismo para evidenciar fallas en la red.
- Mecanismo de seguridad de la red.
- Transparencia con los protocolos y los usuarios.
- Mantener la confiabilidad de la semántica de extremo a extremo.
- Escalabilidad.
- Alternativas de enrutamiento, si se presentan problemas en los nodos.
- Transparencia en la calidad de servicio de PEP.

3.0. TECNICAS DE ACCESO AL MEDIO EN REDES SATELITALES.

3.1. ACCESO MÚLTIPLE CON ASIGNACIÓN POR DEMANDA CON CANAL ÚNICO POR PORTADORA MODULADO EN PCM (SPADE).

En el sistema SPADE cada ranura tiene su propia frecuencia portadora y su ancho de banda es ocupado por un solo canal telefónico modulado, a esta forma de transmisión se le llama canal único por portadora o SCPC (ver figura 3), y aún cuando en este caso la asignación es por demanda, es fácil comprender que puede haber otros sistemas domésticos o internacionales con SCPC pero de asignación fija; el sistema SPADE, que no es más que un sistema DAMA internacional con algunas adaptaciones, consiste en un transpondedor de 36 MHz ranurado en 800 secciones capaces de conducir simultáneamente 400 conversaciones telefónicas (400 ranuras se emplean para los canales de ida y 400 para las de regreso), cada una de las ranuras tiene su frecuencia portadora y puede ser utilizada temporal o por cualquiera de los países que integran el sistema, que se sincronizan con el banco central de frecuencias mediante un canal digital de solicitudes. Por norma general se tiene que este sistema SCPC es utilizado para comunicar puntos con tráfico ocasional como zonas rurales o de poco intercambio entre sí; para comunicar puntos donde se presente un tipo de tráfico de manera permanente se emplea la asignación fija y puedes ser SCPC o bien una portadora multicanal (MCPC) [30,31].

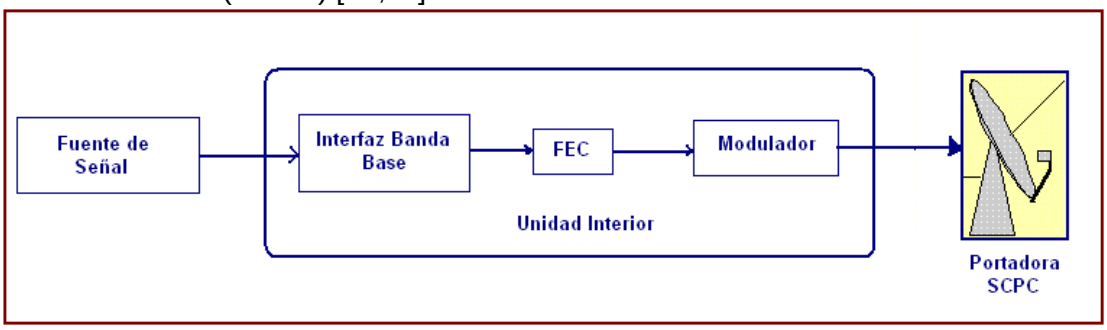


Figura 3. Modo de funcionamiento del Sistema SCPC [32].

3.2. ACCESO MÚLTIPLE POR DIVISION DE TIEMPO (TDMA).

TDMA es una técnica totalmente digital, por lo cual es aplicable a portadoras digitales y es ideal para evitar las limitaciones de FDMA originadas por los efectos de intermodulación; Lo que se logra con una sola portadora por transpondedor (SCPT), en esta circunstancia TDMA tiene una gran ventaja, el transpondedor puede trabajar a su potencia máxima en la región de saturación permitiendo un aumento en la PIRE descendente y la eliminación del ruido de intermodulación, lo que hace posible aumentar la capacidad de canales mediante los recursos de intercambio entre potencia y ancho de banda [31,33].

3.2.1. Estructura y Contenido de las Ráfagas.

La duración de cada ráfaga puede ser distinta para diferentes estaciones y puede modificarse en forma dinámica según las necesidades de tráfico de una estación. La guarda de tiempo entre ráfagas se determina teniendo en cuenta las variaciones en las condiciones de propagación, el tiempo de adquisición de la sincronización y demás retrasos causados en los equipos; cada ráfaga contiene una porción de bits de preámbulo y otros para el tráfico, a continuación se muestra su estructura [33,34].

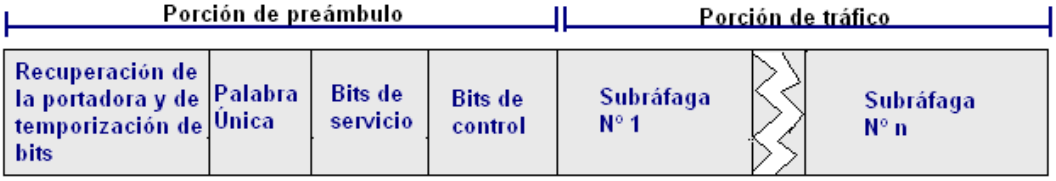


Figura 4. Formato de bits recibidos por una estación de un sistema TDMA en cada ráfaga de tráfico [33].

3.2.2. Eficiencia de la Trama.

El valor de la eficiencia de la trama se puede calcular por medio de la ecuación 11:

$$\varphi = 1 - \frac{nP + n_R P_R + (n + n_R) G R_c}{R_c T} \qquad (11)$$

Donde φ = Número de estaciones de tráfico.

P = Número de bits en el preámbulo de la ráfaga de tráfico.
n_R= Número de estaciones terrenas de referencia.
P_R= Número de bits en las ráfagas de referencia.
G = Duración de una guarda en segundos.
R_c= Velocidad binario de la portadora en bits/s.
T = Duración de la trama en segundos.

La manera de aumentar la eficiencia de la trama consiste en aumentar los bits de información de tráfico de cada ráfaga para que su porción sea más grande respecto del total, para ello es necesario que se aumente la capacidad de los dispositivos de memoria en las estaciones y la duración total de la trama para un mismo número de estaciones terrenas [32,33, 34].

3.2.3. Capacidad de la Red.

La capacidad total de tráfico en bits/s de una red TDMA, donde uno de los factores limitantes esta dado por el ancho de banda, depende fundamentalmente de la velocidad binaria de la portadora y la eficiencia de la trama; la capacidad está dada por la ecuación 12 [33,34].

$$R_t = R_c r \varphi \qquad (12)$$

La mayoría de las investigaciones realizadas estudian los esquemas de acceso múltiple dentro de los cuales se centran en los sistemas TDMA, a continuación se presentara las variantes más importantes del esquema básico de TDMA:

3.3. ACCESO MÚLTIPLE POR DIVISIÓN DE TIEMPO SATELLITE-SWICHED (SS-TDMA).

La técnica de interconexión por el método *transponder hopping* es una solución cuando el número de haces es bajo pues el número de transpondedores y filtros es al menos igual al número de haces al cuadrado. Cuando hay un número elevado de haces hay que considerar la técnica SS/TDMA (*acces*). Esta técnica es uno de los esquemas de conexión para un sistema de comunicaciones satelitales de haz múltiple donde el satélite presenta conmutación abordo; este sistema de haces múltiples con ancho de banda variable en el

enlace de subida M (M número de enlaces de subida) y en el de bajada N, (N número de enlaces de bajada) es presentado como una matriz de tráfico MxN donde cada elemento t(i,j) denota el número de intervalos de tiempo que se necesitan para transmitir, el enlace de subida i al enlace de bajada j.

Los algoritmos están basados en la formulación del problema de la asignación de intervalos de tiempo como un problema de flujo de la red para encontrar un modelo gráfico que represente el sistema de conmutación [3,15], el problema es modelado matemáticamente para encontrar un conjunto de matrices, para una matriz de tráfico dada, que salve algunas restricciones para obtener la máxima capacidad del manejo de tráfico y mínimo tiempo de asignación de intervalos. En la figura 5 se indica una red satelital con haces conmutados.

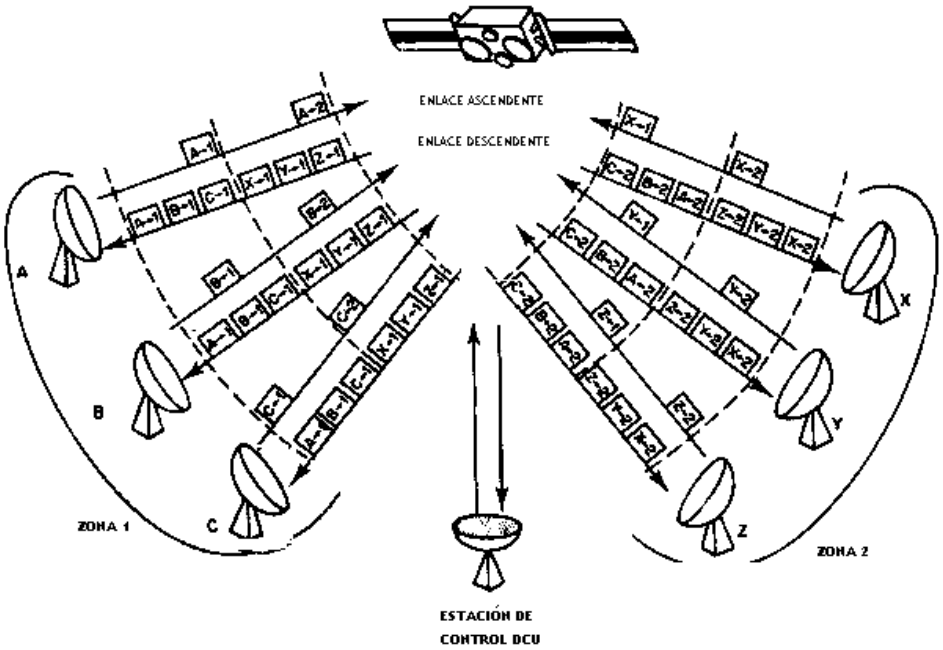


Figura 5. Red satelital con SS/TDMA [35].

La matriz de interconexión con un número de entradas y salidas es igual al número de haces de subida y bajada respectivamente como muestra la figura 6. Esta es la encargada de establecer la conexión entre un haz ascendente en particular con otro descendente. El control de la matriz de conmutación lo ejerce la unidad de control de distribución (DCU), la cual establece el orden secuencial de conexiones. El inconveniente de esta técnica deriva de la interconexión cíclica dado que las estaciones deben almacenar la información a transmitir un lapso igual al periodo del ciclo de conmutación, el cual se eleva de acuerdo al número de haces y además se exige gran sincronización entre todas las estaciones de la red para transmitir en los intervalos debidos, lo que requiere una estación en tierra adicional para esta función y sistemas de señalización que faciliten este proceso [34,35].

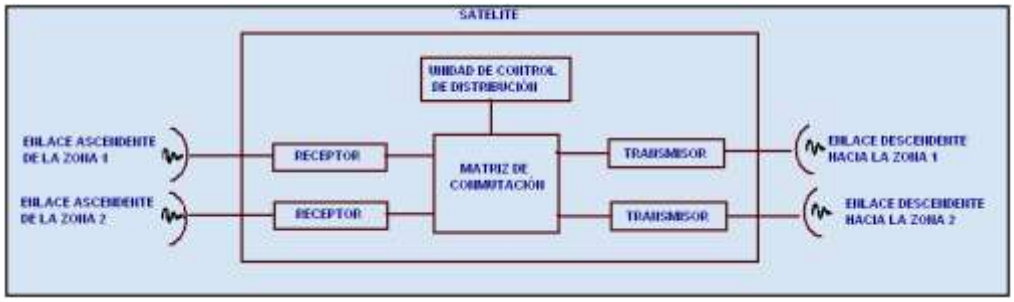


Figura 6. Matriz de interconexión [33].

3.4. ACCESO MULTIPLE POR DIVISION DE CODIGO Ó CDMA.

La técnica CDMA o espectro ensanchado, es una técnica de banda ancha, donde las estaciones pueden trasmitir durante cualquier instante de tiempo en un ancho de banda dado. La separación de cada estación se lleva a cabo mediante un código pseudoaleatorio (PN), con el que se cifra la información a la frecuencia portadora, y que sólo puede ser decodificado por la estación que posea ese código específico. El código empleado debe poseer una serie de características que le doten de la capacidad de aislar las transmisiones y evitar las escuchas ajenas. La auto-correlación de los códigos debe parecerse a la del ruido blanco, y la correlación cruzada entre códigos debe ser nula (códigos ortogonales). De esta manera las transmisiones son espectralmente planas, aparentemente sin ruido y evitan que un código ajeno que posea segmentos similares al del emisor pueda recuperar partes de la transmisión [30,33].

Es un método de acceso simple ya que no necesita sincronismo, proporcionando protección contra interferencias, permite utilizar antenas más pequeñas, es más seguro y facilita la incorporación de nuevos usuarios al sistema. Sin embargo es poco eficiente, requiere un control de potencias elevado para garantizar el correcto funcionamiento y se necesita sincronizar la secuencia en el receptor, que es un aspecto crítico para garantizar la recepción correcta del mensaje [31,32], en la figura 7 se observa la propagación de una señal en DCMA.

En los sistemas CDMA cada estación receptora identifica la señal que le corresponde utilizando para ello métodos de correlación, que permiten una alta probabilidad de recepción correcta aún en presencia de la alta interferencia causada por las demás señales, debido a que cuenta con un código individual para descifrarla y a la gran longitud de los trenes de bits que representa cada carácter.

La señal original de información en banda base posee una velocidad binaria Ri = Rb bits/s los cuales son representados por pulsos. El proceso de tratamiento para la expansión o ensanchamiento genera una gran cantidad de bits codificados los cuales son llamados *segmentos o chips* a una velocidad binaria R_{CH} bits/s. La expansión se hace mediante un código preciso y la recepción de la señal codificada realiza el proceso inverso para decodificar la información, actuando como filtro complementario para el resto de las señales; sin este proceso sería imposible transmitir una señal en CDMA ya que la relación señal a ruido seria menor que uno [6,33].

Existen varias técnicas de CDMA dentro de las cuales las más utilizadas son la de secuencia directa (SD) y la de salto de frecuencia (SF).

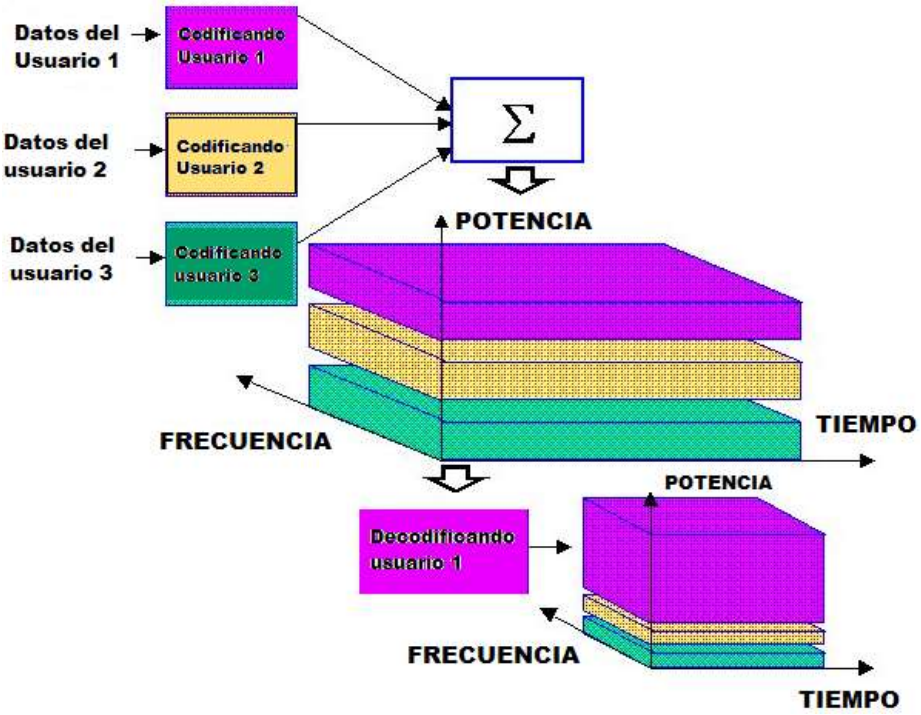


Figura 7. Propagación de una señal en DCMA [37].

3.4.1. Acceso Múltiple por División de Código de Secuencia Directa (SD-CDMA).

El esquema de acceso múltiple DS-CDMA consiste en una técnica de espectro ensanchado donde se consigue la ortogonalidad entre las señales por medio de la asignación de secuencias de código adecuadamente diseñadas a los diferentes usuarios, disponiendo cada uno de una secuencia distinta. Los usuarios transmiten de forma continua, desde la misma estación y empleando todo el ancho de banda. La separación de las señales en recepción se consigue por la característica de rechazo de interferencias propia de las señales de espectro ensanchado [38,39,40].

Una de las maneras posibles para realizar la expansión espectral es mediante la multiplicación directa de las formas de onda de la señal de información y del código de expansión. En la figura 8 se representa un diagrama de bloques simplificado de un transmisor.

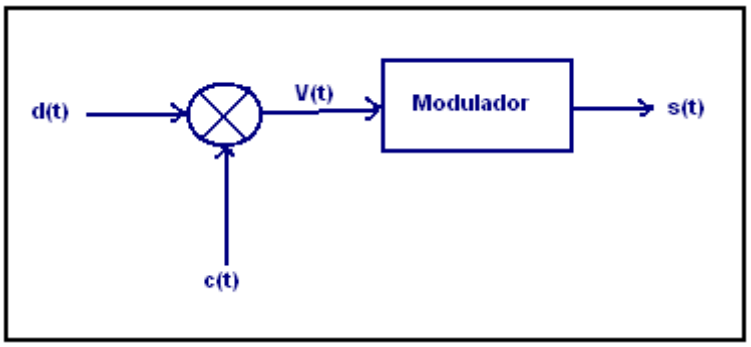


Figura 8. Diagrama simplificado de bloques para un transmisor CDMA [40].

Donde d(t) es la señal de información y c(t) es la señal de código, c(t) es una señal digital con una velocidad binaria muy superior a la señal d(t) para poder realizar la expansión, a los símbolos c(t) se los conoce con el nombre de “chips” para distinguirlos de los bits [30, 40].
Para crear secuencias Directas CDMA los códigos de secuencias más utilizados son los de longitud máxima o pseudo ruido (PN), códigos Hadamard Walsh, códigos Gold y los códigos Kasami. Ver figura 9.

Una vez aplicada la señal de chip, el estándar IEEE 802.11 ha definido dos tipos de modulación para la técnica de espectro ensanchado por secuencia directa, la modulación DBPSK (*Differential Binary Phase Shift Keying*) y la modulación DQPSK (*Differential Quadrature Phase Shift Keying*), que proporcionan una velocidad de transferencia de 1 y 2 Mbps respectivamente.

Recientemente la IEEE ha revisado este estándar, y en esta revisión, conocida como 802.11b, además de otras mejoras en seguridad, aumenta esta velocidad hasta los 11Mbps, lo que incrementa notablemente el rendimiento de este tipo de redes.

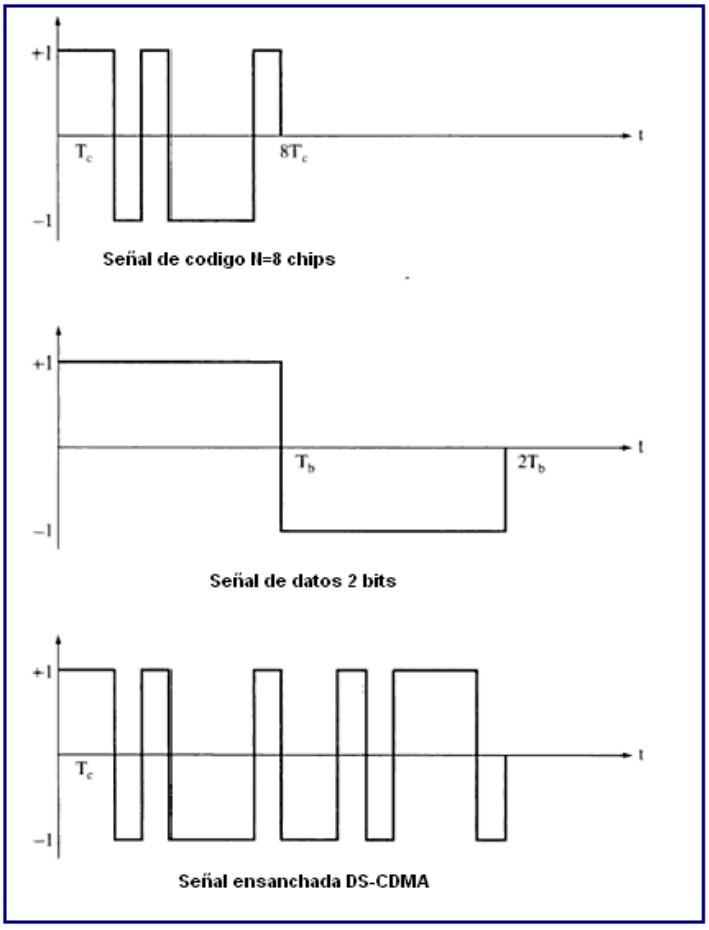


Figura 9. Ejemplo de una señal DS-CDMA [90].

3.4.2. Acceso Múltiple por División de Código por Saltos de Frecuencia. (SF-CDMA).

El método de salto en frecuencia, consiste en tomar el ancho de banda total disponible y dividirlo en subdivide en un número grande de ranuras. En cualquier intervalo de tiempo la señal ocupara una o más de las ranuras de frecuencia al trasladar la señal en una proporción determinada por una secuencia seudoaleatoria [41].

El intervalo de tiempo es llamado *dwell time* y es inferior a 400ms; pasado este tiempo se cambia la frecuencia de emisión y se sigue transmitiendo a otra frecuencia. De esta manera cada tramo de información se va transmitiendo en una frecuencia distinta durante un intervalo muy corto de tiempo. Cada una de las transmisiones a una frecuencia concreta se realiza utilizando una portadora de banda estrecha que va cambiando (saltando) a lo largo del tiempo. Este procedimiento equivale a realizar una partición de la información en el dominio temporal. La figura 10 describe de manera general esta técnica [42,43].

El transmisor como el receptor contiene una tabla de datos en donde contiene la información del orden en que se realizan los saltos en frecuencia a través de una secuencia seudoaleatoria; Este sistema debe mantener una buena sincronización entre los diferentes saltos tanto en el trasmisor como en el receptor, de esta manera se mantiene un único canal lógico así se esté cambiando constantemente el canal físico.

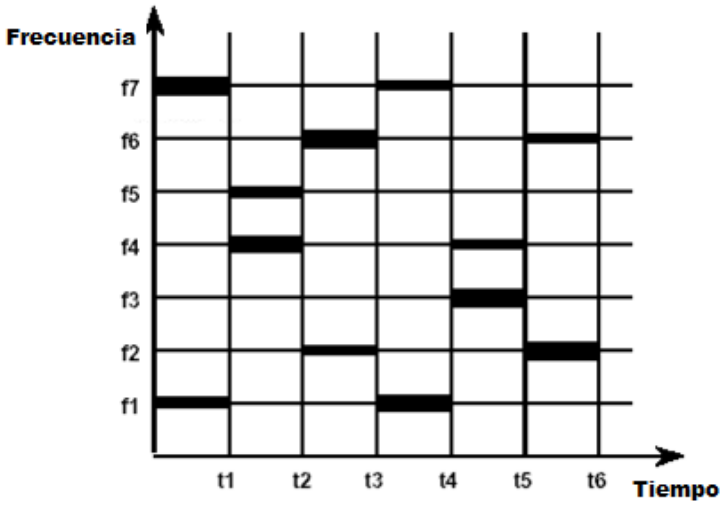


Figura 10. Modo de trabajo de SF-CDMA [43].

3.5. ACCESO MULTIPLE POR DIVISION ESPACIAL (SDMA).

En este tipo de acceso al medio los usuarios hacen uso de diferentes zonas de servicio, ubicados de tal manera que el satélite sectoriza a los usuarios según sea su posición espacial; el satélite debe ser capaz de establecer un conjunto de canales ortogonales en transmisión y recepción en base a la separación de los usuarios. En la figura 11 se muestra un ejemplo de la técnica SDMA en donde el satélite transmite con el mismo bloque de frecuencias F1 en las zonas uno y tres permitiendo de esta manera una mejor reutilización de la frecuencia [44].

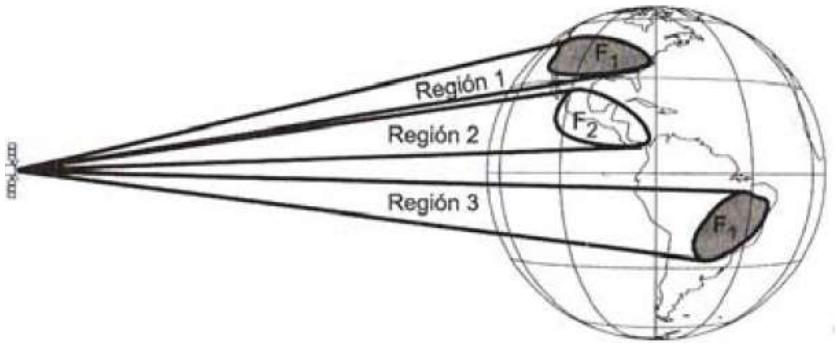


Figura 11. Técnica SDMA [45].

3.5.1. SDMA con Haces Conmutados.

Para esta técnica se requieren antenas de haces conmutados, donde se dispone de un conjunto fijo de haces para dar cobertura a una determinada región; el satélite es capaz de seleccionar para cada usuario un haz por el cual es donde recibe la mayor calidad de la señal de acuerdo a un criterio de optimización (maximización de potencia, maximización de la relación entre la potencia de señal deseada sobre la potencia de ruido e interferencias).

Cada que un usuario cambia de posición, el satélite va conmutando el haz por el que se establecerá la comunicación con dicho usuario, para ello el satélite debe disponer de una matriz de conmutación el cual puede ser implementada en RF (desfasadores de radio frecuencias) o en tecnología digital (Procesadores de señales digitales).

Una de las desventajas de SDMA con haces conmutados es que presentan una gran densidad de interferencias debido a que sus haces son fijos y no pueden cancelarse adaptativamente a las contribuciones no deseadas que aparecen en el sistema [32,34].

3.5.2. SDMA con Antenas Adaptativas.

En esta técnica las antenas van actualizando sus diagramas continuamente, adaptándose a las condiciones radioeléctricas del entorno y a la posición de los usuarios deseados e interferentes. Por ejemplo para sistemas móviles, el haz va siguiendo al usuario móvil conforme este se va desplazando, como se observa en la figura 12.

El uso de antenas adaptativas se requieren de algoritmos de conformación de haces mucho más complejos y generalmente son implementados en banda base mediante procesadores digitales de señal; a diferencias de los haces conmutados, aquí se requieren de criterios de optimización mucho más complejos ya que se dispone de un mayor grado de libertad para el haz [32]. La complejidad del sistema depende del criterio de optimización y algoritmo de conformación de haces seleccionado y del número de antenas del arreglo

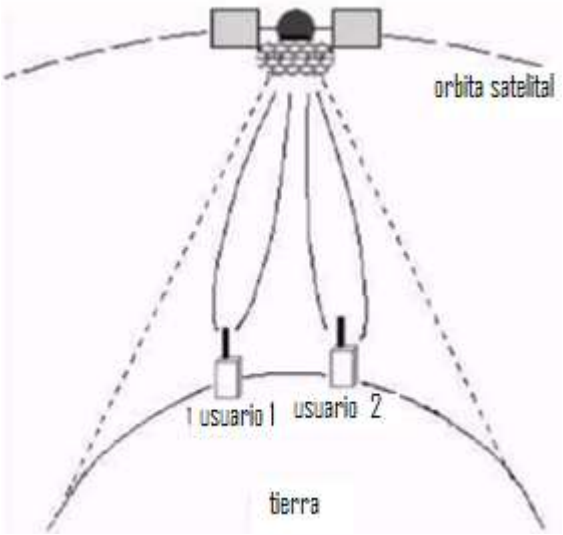


Figura 12. SDMA con Antenas Adaptativas [32].

3.5.3. Acceso Múltiple por División de Polarización PDMA.

La técnica PDMA también llamada *dual polarizarion frequency reuse* consiste en un arreglo de antenas en el satélite cada una con una polarización diferente y receptores separados lo que permite acceso simultáneo del satélite de la misma región de la tierra. Esto obliga a que cada una de las estaciones en la tierra tengan sus antenas con la misma polarización que la del receptor del satélite que corresponda. En SDMA, las bandas de frecuencia pueden reutilizarse, la figura 13 indica el funcionamiento de esta técnica [44,45].

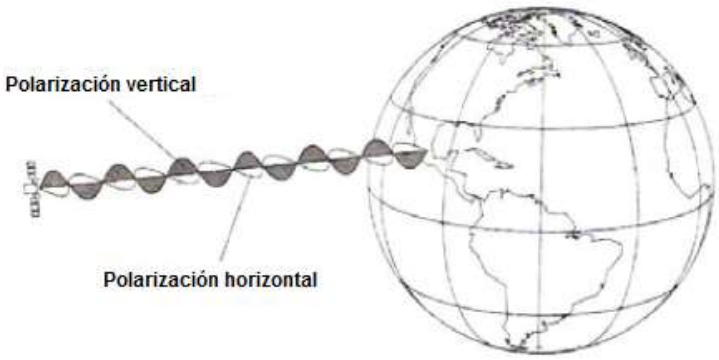


Figura 13. Acceso con PDMA [45].

3.6. ACCESO AL MEDIO ALEATORIO AMA.

Esta técnica de acceso al medio, el acceso a un transpondedor por las estaciones de una red se realiza en forma de contienda por el uso de cada portadora, y puede no haber un control centralizado de asignación permanente o temporal, o puede haber una forma simple de autocontrol de cada estación individualmente. Esta técnica no proporciona alta eficiencia en el uso de la capacidad de un transpondedor y es adecuada para redes donde este factor no es tan fundamental, es muy útil en redes VSAT interactivas en que el tráfico medio por estación sea muy bajo y en forma de paquetes [33].

3.6.1. Aloha Ranurado (S-ALOHA)

Un método para mejorar el rendimiento del canal ALOHA es el denominado ALOHA Ranurado. En este caso el eje temporal se divide en intervalos discretos que corresponden con un paquete y que se llaman ranuras. En este caso se envían a todas las estaciones pulsos de sincronización de tal forma que el tiempo entre pulsos de sincronización es el tiempo de transmisión de un paquete. Ahora el usuario no puede transmitir sus datos en el momento en que quiera, sino que tiene que esperar hasta el comienzo de una ranura. La probabilidad de colisión disminuye puesto que ésta sólo se puede producir en el comienzo de una ranura, se puede tener la seguridad de que un paquete que se ha empezado a transmitir bien completará la transmisión de manera correcta. En caso de colisión se espera ahora un número aleatorio de ranuras. La relación entre el tráfico ofrecido y el rendimiento del ALOHA Ranurado viene dada ahora por la ecuación 13.

$$\varepsilon = Ge^{-G} \tag{13}$$

Como puede observarse se consigue una significativa mejoría del protocolo, logrando un rendimiento máximo de 0.36 para G=1, es decir una utilización del canal del 36% [44,46].

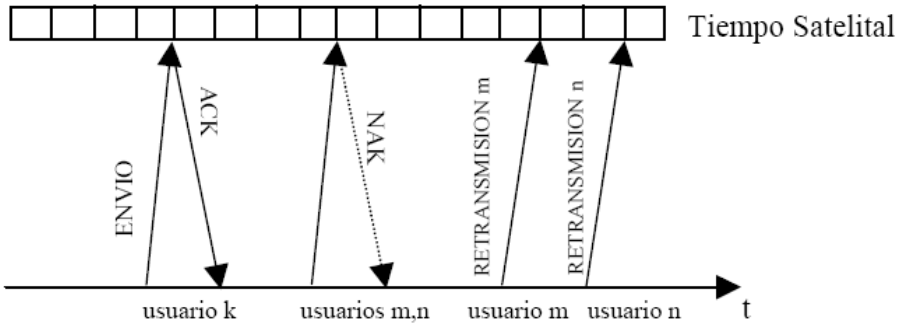


Figura 14 Retransmisión y transmisión de paquetes en S-Aloha [44].

3.6.2. Aloha con Reserva (R-ALOHA).

En este tipo el satélite presenta dos estados que se describen a continuación:

- **Modo sin Reserva:** El tiempo es dividido en pequeños sub slots, donde los usuarios hacen uso de estos para realizara las reservas de los slots necesarios para transmitir el mensaje; una vez hecha la petición de reserva los usuarios esperan un ACK y la asignación del slot, el satélite cambia a modo de reserva [31,32].

- **Modo con Reserva:** En este modo la trama se divide en M+1 slot cuando se realiza una reserva, donde los M slots de tiempo son utilizados para transmitir los mensajes y el último es dividido nuevamente en sub-slot para futuras reservas; los diferentes usuarios solo pueden enviar los mensajes en los slots asignados por el satélite.

  Para determinar cuando el satélite esta en un estado sin reserva o con reserva, todas las estaciones terrenas están sincronizadas por pulsos de sincronización los cuales son enviadas por el satélite. Comparando el R-Aloha con el S-Aloha, el tiempo de retardo es mayor en el aloha con reservas que en el aloha ranurado debido a que requiere hacer reservas de slot de tiempo [31,44], en la figura 15 se aprecia el mecanismo de Aloha con reserva.

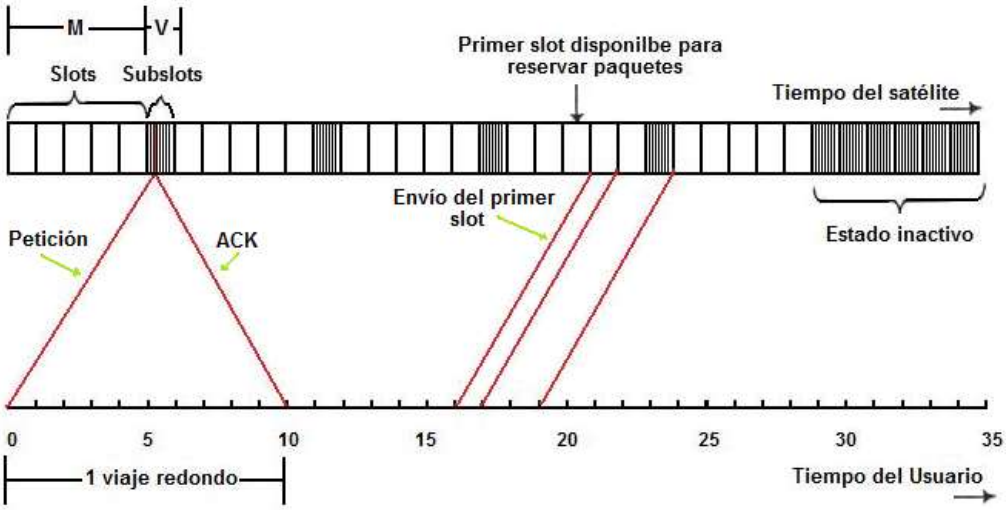


Figura 15. Aloha con reservas [44].

4.0. EL ESTANDAR DVB.

A continuación se describen de manera general las especificaciones de DVB.

4.1. CODIFICACIÓN DE FUENTE.

La codificación de la fuente de audio y de video se hace teniendo en cuenta el estándar MPEG-2, donde se hace una eliminación de la redundancia para obtener tasas de transmisión binarias menores que permitan utilizar anchos de banda apropiado. El estándar MEPG-2 requiere de una comprensión de señal donde la tasa binaria se disminuya al máximo para poder almacenar y transmitir la información, consiguiendo a cambio una degradación objetiva de la calidad de la señal después de la codificación [47].

El grado de degradación de la señal depende de los requerimientos de calidad buscados, de la técnica utilizada y su complejidad. En la codificación MPEG la señal es dividida en tres componentes: una de luminancia (Y) y dos de crominancia (C_b, C_r). La tasa de muestreo para la luminancia es de 13.5 MHz y para la crominancia es de 6,75MHz según la recomendación de la ITU-R [47].

Los algoritmos de compresión MPEG usan una combinación de técnicas de codificación que son las: DCT(Transformada discreta del coseno, para la correlación espacial) y la DPCM (Diferential Pulse code modulation, técnica de predicción temporal) para obtener un alto grado de compresión de la señal.

Dentro del estándar DVB-S se destacan varios tipos de compresión dentro de las cuales las más utilizadas son la compresión de imagen fija JPEG, la compresión de imágenes en movimiento MPEG y la compresión de audio.

4.2. CODIFICACIÓN DE CANAL.

Dentro del estándar DVB-S se realiza una codificación de canal que consiste en aplicar varios procesos de detección de errores en la información digital antes de la etapa de modulación de radio frecuencia, dentro de este modulo se destaca la codificación convolucional y el código perforado; en las figuras 16 y 17 se muestran la ubicación de estos módulos; la codificación convolucional hace uso de registros de desplazamiento de 6 etapas con 5 derivaciones cada uno en las rutas superiores e inferiores [48], esta codificación es muy útil donde la relación C/N es muy baja y es adaptable a diversos parámetros de transmisión proporcionando mayor o menor redundancia; donde se tienen buenas relaciones de C/N se puede usar perforaciones del código, es decir se realiza una codificación convolucional pero se reducen las redundancias, aumentando de esta manera la eficiencia espectral [47] además se utiliza los códigos de corrección de errores Reed-Solomon el cual es entrelazado con el código convolucional para aumentar la eficiencia del canal.

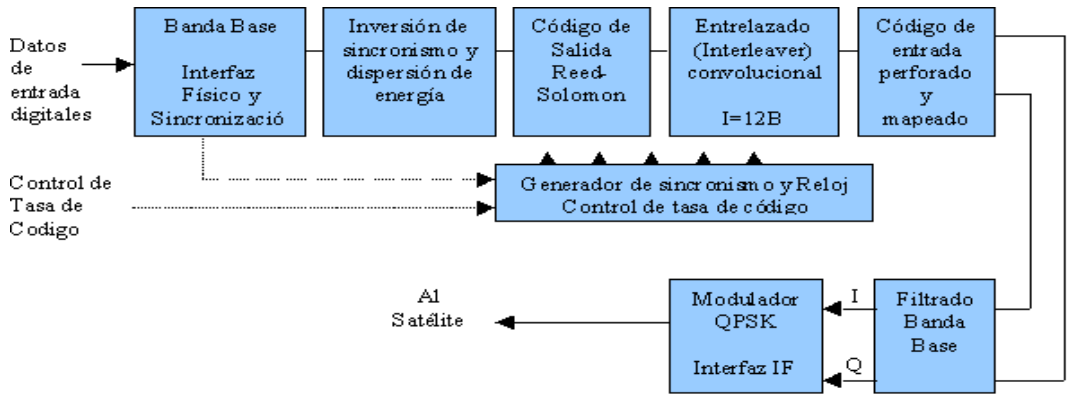


Figura 16. Diagrama de bloques para la Transmisión en DVB-S [47].

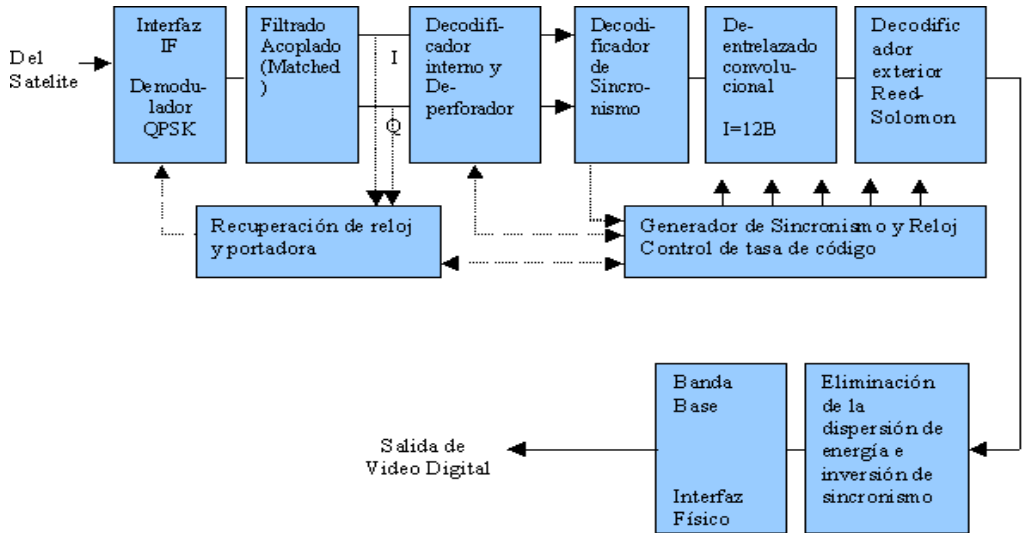


Figura 17. Diagrama de bloques para la Recepción en DVB-S [47].

4.3. MODULADOR DVB-S.

El estándar DBV-S optó por escoger la modulación QPSK, que es una modulación de amplitud constante, donde su información va incluida en la fase de la señal el cual la hace que sea casi inmune a los ruidos atmosféricos, también presenta una eficiencia espectral alta con una ocupación reducida del ancho de banda haciendo de este tipo de modulación muy eficiente, donde se tiene atenuaciones de la señal alrededor de los 200dB debido a las grandes distancias de los satélites GEO y receptores con un nivel de potencia bajo [47], en [48] se explica detalladamente el funcionamiento de este modulo.

4.4. PROCESAMIENTO DE LA SEÑAL EN DVB-S.

Dada la gran distancia que hay entre la estación terrena y el satélite GE0, la antena transmisora y la antena receptora del satélite deben poseer grandes ganancias, debido a las grandes pérdidas por espacio libre que presenta la señal, que están alrededor de los 200dB.

Una vez la señal DVB-S sale de la antena transmisora (estación terrena), que previamente ha sido filtrada por un filtro pasa-banda, en el satélite la señal presenta un cambio de frecuencia (frecuencia para el enlace de bajada), es amplificada por el TWA (amplificador por desplazamiento) y finalmente es filtrada para suprimir las componentes fuera de banda; el modulo encargado de hacer el procesamiento de la señal es el transpondedor; tanto en el enlace de subida como el de bajada se encuentran polarizados con el objetivo de aumentar el número de canales por enlace [48].

4.5. EL RECEPTOR DVB-S.

Una vez la señal sale del satélite GEO y llega a la antena receptora, esta se atenúa alrededor de los 200dB, el receptor dentro de sus módulos tiene el bloque de bajo ruido LNB, el cual está ubicado en el foco de la antena parabólica, el LNB consta de una guía de ondas con polarización horizontal y vertical, dependiendo del voltaje que le sea suministrado y de la polarización con que llegue la señal, esta podrá pasar. Dentro del receptor DVB-S una vez la señal es recibida se hacen los procesos de amplificación de la señal y conversión a frecuencia intermedia [48] ver figura 17.

4.6. ACCESO CONDICIONAL.

El acceso condicional es un método o proceso que utilizan los operadores para controlar el acceso de los usuarios a los servicios que se están prestando (programas de TV); este acceso comprende cuatro componentes que son [47]:

- Un algoritmo de cifrado por cada servicio prestado.
- Un sistema de autorización de abonado.
- Un sistema de gestión de abonado.
- Un algoritmo de aleatorización de flujos de datos.

Cabe resaltar que la norma DVB-S solo estandariza el algoritmo de aleatorización de flujos de datos, los demás se dejan como opcionales que pueden ser implementados por cada operador de acuerdo a sus necesidades.

5.0. EL ESTANDAR DVB-RCS.

5.1. SINCRONIZACIÓN EN DVB-RCST.

Este estándar define dos estados de sincronización, el procedimiento de entrada conocido también como procedimiento de *logeo* de terminal (*Terminal Logon Procedure,* TLP), donde la terminal RCST debe encontrarse en un estado de sincronización recibida y el procedimiento de salida llamado *logoff* [49]. Este estándar a definido cuatros fases o procedimientos para que una RCST pueda entrar al sistema que son: el procedimiento de entrada o TLP, el procedimiento de adquisición gruesa, el procedimiento de sincronización fina y por último el procedimiento para mantener la sincronización, en la figura 18 se muestra un diagrama de flujo los cuatro procedimientos para la entrada de una RCST al sistema.

5.2. SEGMENTACIÓN DEL CANAL DE RETORNO.

En un sistema satelital con la norma DVB-RCS los recursos de la red están organizados en supertramas, tramas y ranuras de tiempo; una supertrama es una porción de tiempo y frecuencia en el enlace de retorno y está conformada por varias tramas y esta a su vez la conforman varios slots; cada supertrama tiene asociado un identificador llamado *Superframe_ID* el cual es el encargado de identificar los recursos disponibles y asignados por la red dentro de un grupo de estaciones terrenas, en la figura 18 se muestra una supertrama y la composición de la misma [47,50].

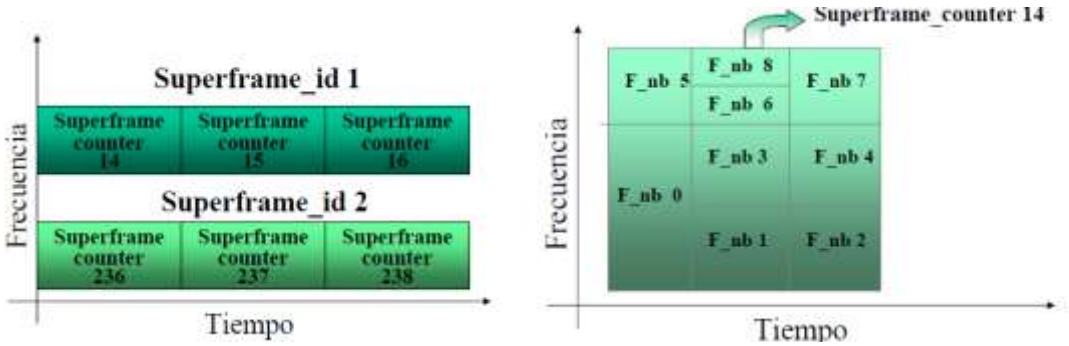


Figura 18. Ejemplo de una conformación de una supertrama [49].

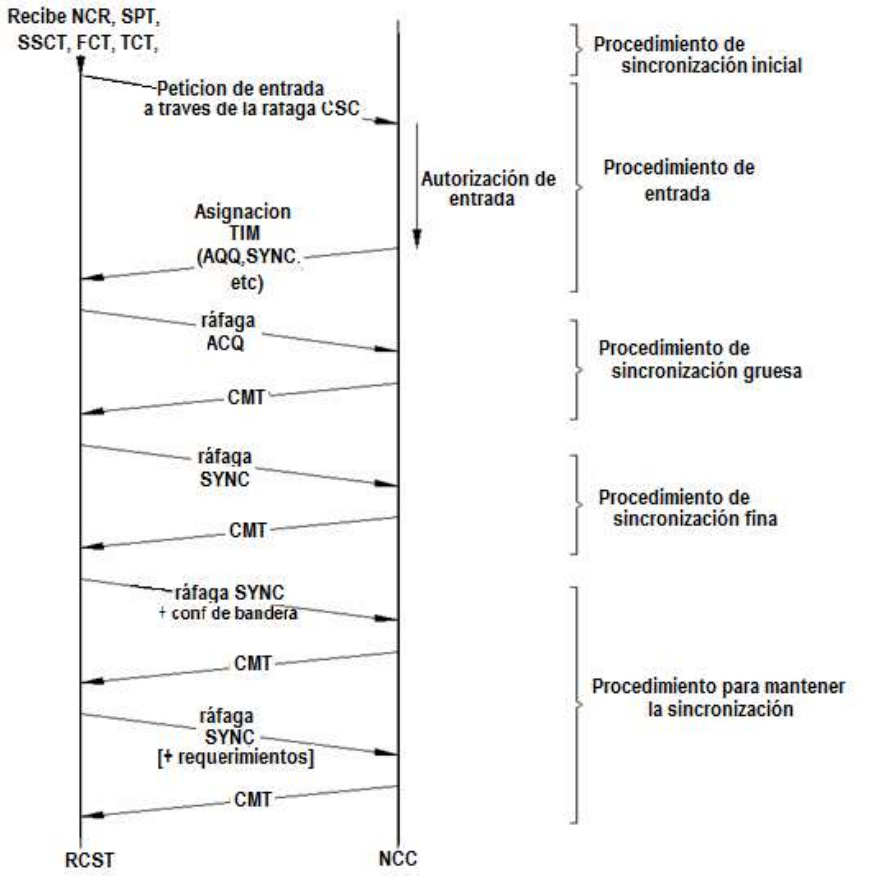


Figura 19. Procedimientos para la entrada de una RCST al sistema [49].

6.0. CODIGOS CORRECCIÓN DE ERRORES

Las estrategias que hoy existentes de codificación de error son: FEC y la solicitud de repetición automática (ARQ), a continuación se describe cada una de estas:

6.1. CODIGOS FEC.

Los códigos FEC se basan en el uso de bits redundantes en la palabra de código transmitida para poder detectar y corregir errores dentro de un canal de comunicaciones, los cuales utilizan un mayor ancho de banda para el envío de datos [41].

6.1.1. Códigos Convolucionales.

Los códigos convolucionales consisten de un registro de corrimiento de M etapas, donde se almacenan temporalmente y se realizan las operaciones de corrimiento con conexiones preestablecidas a *n* sumadores módulo 2 y a un multiplexor que coloca en serie las salidas de los sumadores [41,51].

Los códigos convolucionales manejan dos tipos de codificadores, un exterior de bloques, y un interior convolucional, los cuales se encargan de introducir redundancia en los datos para que de esta manera se pueda corregir los errores de transmisión. Además, el sistema utiliza un entrelazado para que los bloques correlativos no se transmitan juntos.

Los códigos convolucionales son códigos lineales, donde la suma de dos palabras de códigos cualesquiera también es una palabra de código y donde los datos que se han enviado dependen de los que se vayan a enviar en el momento [51].

Los parámetros a tener en cuenta por los códigos convolucionales son (n,k,m):

n: Es el número de bits de la palabra codificada.
k: Es el número de bits de la palabra de datos.
m: Es la memoria del código o longitud restringida.

El proceso de codificación, como se indica en la figura 20, el conmutador tendrá 2 entradas el mismo número de sumadores, los cuales realizaran la operación de un registro de M desplazamiento de 2 estados. Se considera que los registros intermedios están en cero.

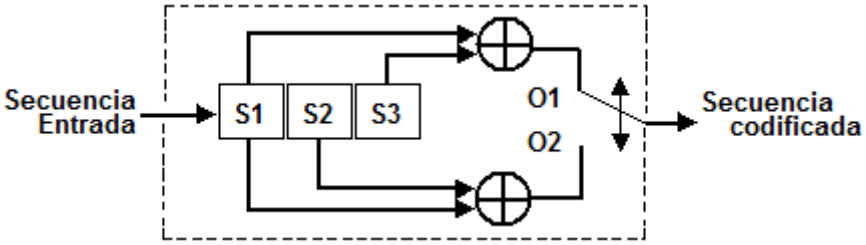


Figura 20. Codificador convolucional (2, 1, 3) [41].

Las propiedades estructurales de un codificador convolucional se representan en forma gráfica y para ello se utilizan tres métodos.

- Árbol de código.
- Diagrama de estados.
- Enramado.

Para el diagrama de árbol de código, cada rama de árbol significa un estado inicial con su correspondiente par de números binarios, según su convención, 1 es para la entrada inferior de una bifurcación y 0 para la entrada superior de esta. El mensaje de entrada se lee de izquierda a derecha. El número de ramas se va multiplicando por dos con cada nueva entrada y esta se vuelve repetitiva después de entrar el tercer bit de mensaje, dando origen a un *enramado* que tiene la misma estructura de un árbol pero con ramas que resurgen [41,51]. En la figura 21 se observa el diagrama de árbol de código descrito anteriormente.

El enramado permite realizar la decodificación de la forma más sencilla, su convención para distinguir entre los símbolos de uno y cero es la siguiente; una rama en el estado cero se dibuja con una línea de forma continua, y en el estado uno se ilustra con una línea punteada. Como se muestra en la figura 22. Para el proceso de decodificación se emplea el algoritmo de Viterbi.

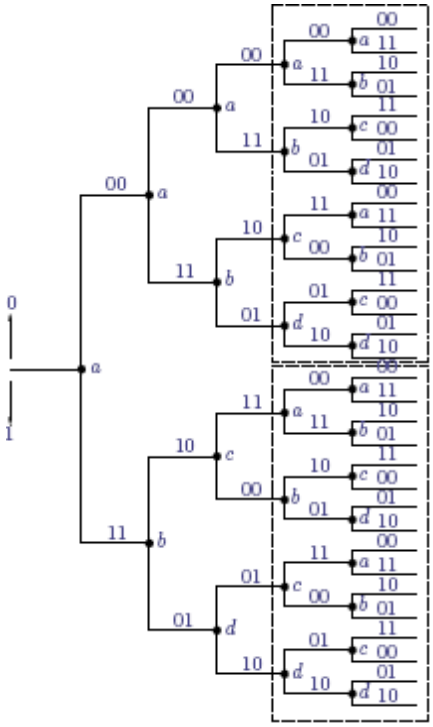


Figura 21. Diagrama de árbol de un codificador convolucional [52].

- **Algoritmo de viterbi.**

El Algoritmo de Viterbi se encarga de buscar el mejor camino en el diagrama de árbol o enramado, al pasar por cada unos de los nodos escoge la menor distancia Hamming y se guarda la mejor secuencia, este cálculo se repite en cada uno de los nivel j del enramado en el intervalo de M≤ j ≤ L, donde K-1 es la memoria del codificador y L es la longitud de secuencia de mensaje entrante [42,51].

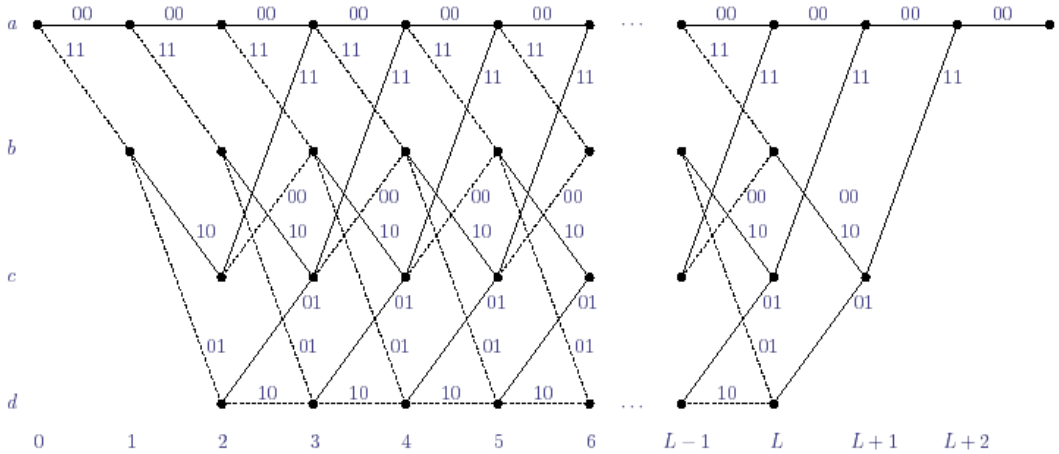


Figura 22. Diagrama de un código enramado [52].

Un detalle a tener en cuenta en el codificador, es el hecho de que este separa el mensaje en tramas de longitud L, y adiciona al final de la trama codificada un cero, de esta manera se conoce la siguiente trama a decodificar.

A continuación se describen los pasos a seguir en la decodificación del Algoritmo:

Paso 1: En el nivel j, se calcula la distancia de *Hamming* de cada uno de las trayectorias que entran en cada nodo desde el nodo del nivel j-1 hasta el nodo del nivel j a través del camino superviviente.

Paso 2: Para cada nodo del diagrama de *Trellis* en el nivel j, se selecciona el camino que presente la menor distancia y se almacena su trayectoria como su métrica. Cuando a un nodo llegan dos caminos con la misma distancia se toma el superior.

Paso 3: Al pasar al nivel j+1 se repiten los pasos 1 y 2. Estos pasos se aplican para j mayor o igual que 2. Hasta ese valor se expanden los caminos.

6.1.2. Códigos Lineales:

Un código lineal debe cumplir que la suma de dos palabras de código en modulo 2, su resultado es también otra palabra de código, el flujo de un código de bloque (n,k) requiere que se divida dentro de bloques de k bits, donde los k bits conforman una palabra de código, el número posible de palabras de código es de 2^k, los k bits se codifican dentro de una palabra de longitud n, y los n-k bits restantes se conocen como bits de paridad. Los códigos de bloques tienen la particularidad que los bits de mensaje al transmitirse no se alteran, por eso se dice que los códigos lineales son códigos sistemáticos, esta propiedad es muy importante ya que permite la simplificación en la puesta en práctica del decodificador [34,41].

El número de errores que se pueden corregir por palabra de código está relacionado con la distancia mínima d. y esta a su vez con la distancia *Hamming*. La distancia *Hamming* d(c1,c2) entre un par de vectores se define como el número de localidades en las cuales difieren sus elementos respectivos [5].

6.1.2.1. Códigos cíclicos.

Los códigos cíclicos se caracterizan por que poseen una estructura matemática ya definida, son una subclase de los códigos de bloques lineales, presentan una ventaja importante al poder ser fácilmente implementados en la práctica a través del uso de registros de corrimiento y sumadores modo 2. Según [41] se afirma que un código binario será cíclico si están presentes estas dos propiedades:

1. La suma de cualesquiera dos palabras de código da como resultado otra palabra de código.
2. Cualquier corrimiento cíclico de una palabra de código resulta otra palabra de código.

Estos códigos utilizan un polinomio generador g(*X*) de grado n-k, definen a un polinomio de verificación de paridad h(*X*) de grado k de coeficientes h_i 0 ó 1, donde g(*X*) y h(*X*) son factores del polinomio X^n +1. Los códigos cíclicos son ampliamente usados en las redes satelitales, los cuales se explicaran a continuación.

6.1.2.2. Códigos BCH.

Los códigos BCH, son códigos de bloques lineales, los cuales emplean una gran cantidad de de parámetros, se caracterizan por que pueden detectar y corregir *t* errores aleatorios por palabra de código. La ejecución de los circuitos de codificación como de detección de errores es muy práctica [34].

Para un código BCH binario, el codificador opera sobre bits individuales y si $m \geq 3$ y $t \leq \frac{2^m - 1}{2}$, se tienen los siguientes parámetros.

Tamaño del bloque: $n = 2^m - 1$

Numero de bits del mensaje: $k = n - mt$

Distancia mínima: $d_{mín} \geq 2t + 1$

- **Código Reed Salomon.**

Forman parte de una subclase de los códigos BCH no binarios, ya que opera el codificador sobre bits múltiples. Se utilizan cuando los errores ocurren en ráfagas donde se pueden ver afectados un cierto número de bits debido a un tipo de error impulsivo o de interferencia. Son códigos cíclicos en un campo de símbolo, donde su polinomio generador está en función de los campos de *Galois* que son Campos Finitos, con la propiedad de que sus operaciones de suma y multiplicación con elementos de campo, dan siempre un resultado en el campo, y están basados en el modulo q, se denotan por GF(q) cuyos elementos son {0,1,…,q-1}. El campo finito GF(q) sólo puede ser construido si q es primo o una potencia de un primo. Si $q = p^b$, donde p es primo y b es un entero positivo, se extiende el campo GF(p) al campo GF(p^b), el cual es su raíz conocida como campo extendido GF(p) donde se realizan las operaciones algebraicas [34,53].

Para un código RS (n,k) se emplea para su codificación símbolos de m bits, con un tamaño de bloque de $n = 2^m - 1$ símbolos, para $m \geq 1$, lo que implica que el algoritmo de codificación se expanda un bloque de k símbolos a n símbolos adicionando n-k símbolos redundantes o de paridad [8]. Un decodificador RS podrá detectar y corregir hasta t símbolos que contienen error en una palabra código donde $2t = n - k$. El número de palabras codificadas es q^N pero únicamente q^k contienen los estados de estas. El código RS es muy empleado en las redes satelitales.

6.1.3. TCM.

Este tipo de código se caracteriza por hacer uso de la modulación y codificación como procesos combinados dependientes, y de esta manera poder aprovechar mejor los recursos de ancho de banda y de potencia limitados, para ello hacen uso de la técnica de modulación por codificación de enramado [41,54].

TCM hace uso de la combinación de las funciones de la tasa convolucional k/k+1 y la señal de mapeo de una M-aria, que mapea $M = 2^K$ puntos de entrada dentro de una constelación de la señal expandida de 2^{K+1} puntos de la constelación, al adicionar un bit en la codificación es necesario que se doble el tamaño de la constelación [41,54]. En TCM se conserva el ancho de banda así se duplique el número de puntos de las constelaciones, incrementando la tasa de bits pero su tasa de símbolo no varía. Según [41] el esquema de TCM se puede definir por las siguientes características:

a) Una señal de constelación bidimensional es ampliada de 2^m puntos a 2^{m+1}.
b) Se particiona la constelación original dentro de 2^{k+1} subconjuntos.
c) Una regla de mapeo que define la relación entre los m bits binarios de la fuente, los bits codificados, los bits no codificados y los puntos de la señal de la constelación.
d) Un codificador convolucional de tasa, con $k \leq m$, y v memorias.

Los códigos TCM presentan una constelación ampliada, que le permite mediante los puntos adicionales ser redundantes para la codificación directa del control de errores, al hacer uso de una constelación ampliada se incrementa el tamaño de la constelación lo que implica que los bits distribuidos tengan una mayor probabilidad de error, sin embargo, con el uso del codificador convolucional se logra introducir alguna independencia entre los puntos de señal consecutivos, y en el decodificador la señal es dividida sucesivamente en subconjuntos M/2, M/4, M/8 cada vez más pequeños que tendrán una menor probabilidad de error al aumentar la división en los subconjuntos, la regla de mapeo que se emplea para ello es una estructura de enramado, donde se busca obtener la máxima distancia euclidiana entre los vectores de código [30,54] .

La figura 23 indica el procedimiento de enramado, donde se observa como la distancias euclidianas van aumentando según se aumente el nivel de subconjuntos, en este caso es para una constelación QAM-16.

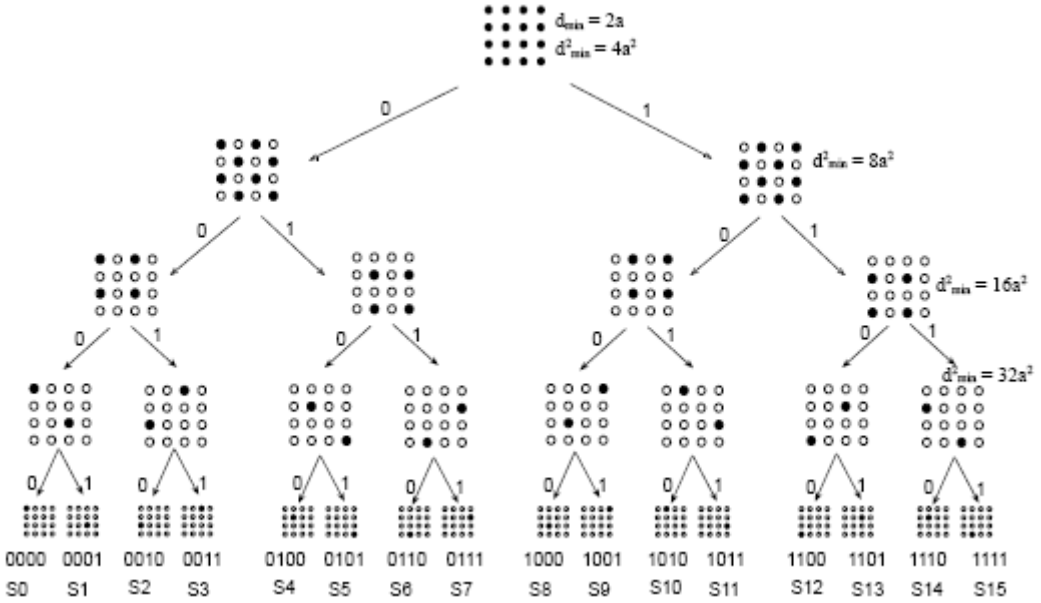


Figura 23. División de la constelación QAM-16 [54].

El proceso que se realiza en un codificador/modulador en TCM es el siguiente, para el envío de m bits de una fuente binaria, si k ≤ m, se codifican sólo los k símbolos binarios donde se utilizan los k+1 símbolos para seleccionar el subconjunto, y con los restante m-k bits no codificados se encargan de seleccionar el punto interior del subconjunto acorde a las reglas de mapeo. En la figura 24 se puede observar este procedimiento [30,54].

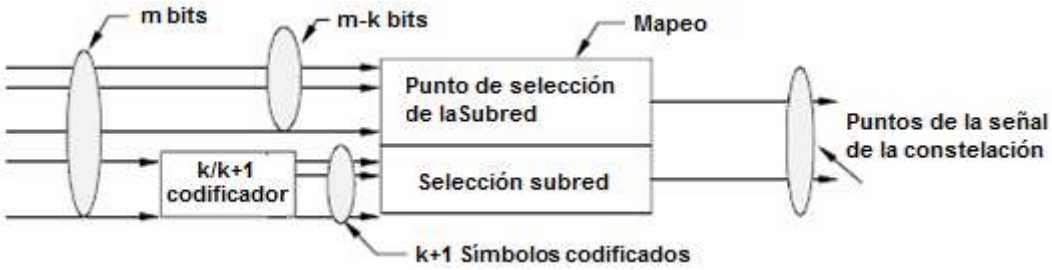


Figura 24. Codificador/modulador TCM [30].

Para decodificar la señal se usa también el algoritmo de descodificación de *viterbi*, donde se modela una estructura de enramado con decisión flexible [30,41].

El uso de los FEC es muy útil en los sistemas satelitales donde su potencia de transmisión es limitada, se presentan pérdidas de propagación considerables y se presenta un

incremento de la potencia de ruido; según [30] su uso puede mejorar la calidad de un enlace de transmisión digital y esto se puede observar en la mejora de los siguientes aspectos:

a) Se reduce la BER, estrechamente relacionada con los criterios de calidad de servicio.
b) En la Eb/N0, ya que al disminuir su valor también se reduce los costos de los elementos de RF (menor potencia de transmisión).

6.2. ARQ.

ARQ es otro mecanismo que se utiliza para resolver el control de errores de un canal, es una técnica que utiliza el mecanismo de retransmisión para poder corregir los errores de transmisión, para ello el receptor debe solicitar la retransmisión de la palabra de código corrompida haciendo uso de un canal de retorno. ARQ no permite la transmisión en tiempo real (voz y vídeo). Un inconveniente de este mecanismo es que debe utilizar un canal de retorno, para informar mediante ACKs sobre el estado de la información transmitida [41,55].

Para el enlace semiduplex se puede transmitir la información pero no de manera simultánea y se utiliza un esquema de ARQ simple. Este mecanismo informa mediante ACKs sobre el estado de la información transmitida. El inconveniente del método de pare y espere es que no hace un buen uso del ancho de banda, debido a que el transmisor en su mayoría de tiempo se la pasa esperando la confirmación de la información por parte del receptor para poder transmitir la siguiente palabra de código.

Otro técnica que se utiliza es el ARQ continuo con retirada, este mecanismo permite que el transmisor pueda enviar los paquetes de código sin tener que esperar su reconocimiento, aunque si un paquete se pierde en el camino, el retransmite desde el paquete que se ha perdido hacia delante así los otros paquetes posteriores al paquete corrompido ya hayan sido confirmados por el destino.

Otra versión de ARQ es el método con repetición selectiva, el transmisor puede enviar todos los paquetes que van a ser transmitidos sin esperar su confirmación, si ocurre una pérdida de corrupción en alguno de los paquetes o el *timeout* de alguno de estos se vence, el transmisor sólo transmite el paquete que se detectó con errores, de esta manera se hace un mejor aprovechamiento del ancho de banda.

En la tabla 2 se observan las técnicas de codificación de control de errores que están presentes en la red, con cada uno de los parámetros que las caracterizan para poder ver sus ventajas y desventajas de su uso en la red.

Tabla 2. Técnicas de control de error [56].

Técnica codificación	Througput	paquete en búfer	Rendimiento	Retraso	Canal de retorno	Decodificación	Tasa de bit
FEC	Alto	no	Medio Alto	Corto	no	Complejo	Alto Muy alto
ARQ	bajo	si	Muy alto	Largo	si	Simple	bajo medio
Hibrido (HEC)	Medio	si	Alto	Medio	si	Medio	Alto

BIBLIOGRAFIA

[1] M. Todorović, *Comparative Study of the End-To-End Compliant TCP Protocols for Wireless Networks*, Universidad de Texas, Dic. 2005. Disponible en: http://etd.lib.ttu.edu/theses/available/etd-11282005-100301/unrestricted/todorovic_final.pdf. Consultado 12 Diciembre del 2008.

[2] P. Rajashree y L. Trajković, *Selective-TCP for Wired/Wireless Networks*, Universidad de Vancouver, Canada, 2005. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.5484&rep=rep1&type=pdf Consultado 12 Enero del 2009.

[3] D. Guzmán, y C. Lara, Métodos de control de congestión. Documento disponible en: http://profesores.elo.utfsm.cl. Consultado 20 Enero del 2009.

[4] K. Fall y S. Floyd, *Simulation-based Comparisons of Tahoe, Reno, and SACK TCP,* U.S, 1997. Documento disponible en: http://reference.kfupm.edu.sa/content/s/i/simulation_based_comparisons_of_tahoe__r_107568.pdf. Consultado 12 Enero del 2009.

[5] M. Barreto, M. Belino, y M. San Martín, TCP SACK, Universidad de la República, 2002. Documento disponible en: http://iie.fing.edu.uy/ense/asign/perfredes/trabajos/trabajos_2004/TCPSack/TCPSack.pdf. Consultado 12 Diciembre del 2008.

[6] C. Fu, *TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks*, Universidad de Hong Kong Shatin, Feb. 2003. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.1469&rep=rep1&type=pdf. Consultado 12 Enero del 2009.

[7] M. Carter, *A Review of Transport Protocols as Candidates For Use in a Tactical Environment*, Dic. 2005. Artículo de Tesis disponible en: http://www.dsto.defence.gov.au/publications/4318/DSTO-TR-1808.pdf. Consultado 10 Septiembre del 2008.

[8] C. Samios y M. Vernon, *Modeling the Throughput of TCP Vegas*, Universidad de Wisconsin Madison, 2003. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.7204&rep=rep1&type=pdf. Consultado 12 Diciembre del 2008.

[9] L. Wu, F. Peng, y V. Leung, "*Dynamic Congestion Control for Satellite Networks Employing TCP Performance Enhancement Proxies",* Universidad de British Columbia, Canada, 2004. Artículo de Tesis disponible en: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/9435/29943/01368363.pdf?arnumber=1368363. Consultado 4 Noviembre del 2008

[10] K. Srijith, L. Jacob, y A. Ananda, *TCP Vegas-A: Solving the Fairness and Rerouting Issues of TCP Vegas*, Universidad de Singapore, 2002. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.6157&rep=rep1&type=pdf Consultado 12 Diciembre del 2008.

[11] J. Mo, R. La, V. Anantharam, y J. Walrand, *Analysis and Comparison of TCP Reno and Vegas*, Universidad de California, Jul. 1998. Documento disponible en: http://www.eecs.berkeley.edu/~ananth/1999-2001/Richard/MoLaInfocom1999.pdf. Consultado 12 Diciembre del 2008.

[12] S. Kerkar, *Performance Analysis of TCP/IP Over High Bandwidth Delay Product Networks*, Universidad del Sur de Florida, Jul. 2006. Documento disponible en: http://etd.fcla.edu/SF/SFE0000453/final_thesis_subodh_071604.pdf. Consultado 20 Febrero del 2009.

[13] G. Yang, R. Wang, M. Sanadidi, y M. Gerla, *TCPW with Bulk Repeat in Next Generation Wireless Networks*, Universidad de California, U.S, 2002. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.1717&rep=rep1&type=pdf. Consultado 20 Febrero del 2009.

[14] G. Yang, R. Wang, M. Gerla, y M. Sanadidi, *TCP Bulk Repeat*, Universidad de California, U.S, 2004. Documento disponible en: http://nrlweb.cs.ucla.edu/publication/download/453/yangg2005comcom.pdf. Consultado 20 Febrero del 2009.

[15] G. Yang, R. Wang, M. Sanadidi, y M. Gerla, *Performance of TCPW BR in Next Generation Wireless and Satellite Networks*, Universidad de California, U.S, 2003. Documento disponible en: http://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_papers/tech020025-tcpwbr.pdf. Consultado 22 Febrero del 2009.

[16] S. Mascolo, L. Grieco, R. Ferorelli, P. Camarda, y G. Piscitelli, *Performance Evaluation of Westwood+ TCP Congestion Control*, Politécnico di Bari, Italia, Abril 2003. Documento disponible en: http://c3lab.poliba.it/images/7/76/Perf-04.pdf. Consultado 12 Febrero del 2009.

[17] C. Caini, R. Firrincieli, M. Marchese, T. de Cola, M. Luglio, C. Roseti, N. Celandroni, y F. Potortí, *Transport layer protocols and architectures for satellite networks*, Jul. 2006. Documento disponible en: http://www.tlcsat.uniroma2.it/wp-content/uploads/2007/04/tcpsat-ijsc06.pdf. Consultado 20 Enero del 2009.

[18] R. Wang, G. Pau, K. Yamada, M. Sanadidi, y M. Gerla, *TCP Startup Performance in Large Bandwidth Delay Networks*, Universidad de California, U.S, 2004. Documento disponible en: http://www.cs.ucla.edu/~gpau/Papers-C/C018-2004-Infocom.PDF. Consultado 12 Febrero del 2009.

[19] R. Wang, G. Pau, M. Sanadidi y M. Gerla, *Improving TCP Start-Up over High Bandwidth Delay Paths*, Universidad de California, U.S, 2003. Documento disponible en: http://www.cs.ucla.edu/classes/fall03/cs218/slides/hsn2003-TCP-astart-color.pdf. Consultado 22 Febrero del 2009.

[20] M. Omueti, TCP-ADaLR: *TCP with Adaptive Delay And Loss Response for Broadband Geo Satellite Networks,* Universidad de Simon Fraser, 2007. Documento disponible en: http://www.ensc.sfu.ca/people/faculty/ljilja/cnl/pdf/omueti_masc_thesis_final.pdf. Consultado 1 Marzo del 2009.

[21] M. Omueti y L. Trajković, *TCP with Adaptive Delay and Loss Response for Heterogeneous Networks*, Universidad de Simon Fraser, Canada, 2007. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.7340&rep=rep1&type=pdf Consultado 1 Marzo del 2009.

[22] X. Zhou y J. Baras, TCP over Satellite Hybrid Networks: A Survey, Universidad de Maryland, 2002. Disponible en: http://www.isr.umd.edu/~baras/publications/reports/2002/ZhouB_TR_2002-27.pdf. Consultado 12 Diciembre del 2008.

[23] Y. Tian, K. XU, y N. Ansari, *TCP in Wireless Environments: Problems and Solutions*, Mar. 2005. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.114.5941&rep=rep1&type=pdf. Consultado 1 Marzo del 2009.

[24] I. Akyildiz, G. Morabito, y S. Palazzo, *TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks,* Jun. 2001. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.5266&rep=rep1&type=pdf. Consultado 1 Marzo del 2009.

[25] S. Utsumi, S. Muhamaad, S. Zabir, y N. Shiratori, *TCP-Cherry : A New Scheme of TCP Congestion Control for Satellite IP Networks,* Universidad Tohoku, 2008. Documento disponible en: http://www.shiratori.riec.tohoku.ac.jp/~u-satoshi/soim-coe08.ppt. Consultado 22 Febrero del 2009.

[26] I. Akyildiz, X. Zhang, y J. Fang, *TCP PeachPlus: Enhancement of TCP Peach for Satellite IP Networks,* Instituto de Tecnología Georgia, 2001. Documento disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.5670. Consultado 1 Marzo del 2009.

[27] J. Border, M. Kojo, J. Griner, G. Montenegro, y Z. Shelby, *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*. RFC 3135, Internet Engineering Task Force, Jun. 2001. Documento disponible en: http://www.faqs.org/rfcs/rfc3135.html . Consultado 25 Febrero del 2009.

[28] K. Selén, *Evaluation of Performance Enhancing Proxies in a Global Network,* Universidad de Helsinki, Mar. 2008. Documento disponible en: http://www.tml.tkk.fi/~anttiyj/Selen-Acceleration.pdf. Consultado 4 Marzo del 2009.

[29] T. Anderson, *TCP/IP over Satellite: Optimization vs. Acceleration, End II End Communications*, Inc. - White Paper, Abril 2005. Documento disponible en: http://www.endiiend.com/admin/image/whitepaper/white_17.pdf. Consultado 12 Marzo del 2009.

[30] D. Brown," *Handbook on Satellite Communications (HSC)",* 3 Edición Wiley.

[31] Rodolfo, Neri. Comunicaciones por satélite. Editorial Thomson, Bogotá Colombia 2002.

[32] M. Calvo y R. Ramón. Comunicaciones por Satélite, Acceso Múltiple, Universidad Politécnica de Madrid, 2008. http://www.gr.ssr.upm.es/docencia/grado/csat/material/CSA08-2-FormatoTLE_2p.pdf

[33] Rosado. Comunicaciones por Satélite. Editorial Limusa, México, 1999.

[34] D. Roddy,"*Satellite Comunications*", Edit McGraw-Hill.

[35] R. García y A. García, Antenas embarcadas en satélites, Universidad Politécnica de Valencia. Documento disponible en: http://www.upv.es/satelite/trabajos/Grupo9_99.00/haces.htm. Consultado el 22 de Febrero del 2009.

[36] S. Perlaza y G. Villalobos. Las Tecnologías Asociadas a Los Sistemas Satelitales de Telecomunicaciones, Trabajo de grado. Universidad del Cauca, Facultad de Ingeniería de Telecomunicaciones, Colombia, Feb. 2005.

[37] UMTS, *Access Schemes*, 1999. Documento disponible en:

http://www.umtsworld.com/technology/cdmabasics.htm. Consultado el 20 de Mayo del 2009.

[38] D. Lorenzo, M. Blanco, P. Amo, y F. Cruz Roldán, *Performance Evaluation of Polyphase Sequences in DS/CDMA Communication Systems*, Departamento de Teoría de la Señal y Comunicaciones, Universidad de Alcalá de Henares. Documento disponible en: http://w3.iec.csic.es/ursi/articulos_modernos/articulos_gandia_2005/articulos/SC4/340.pdf. Consultado el 20 de Mayo del 2009.

[39] J. Pérez, O. Sallent, y R. Agustí, Técnicas de ARQ en un Esquema de Acceso por Paquetes DS-CDMA. Departamento de Teoría de la Señal y Comunicaciones. Universidad Politécnica de Cataluña, España. Documento disponible en: http://www.gcr.tsc.upc.edu/publications/proceedings/1998/Tecnicas_de_ARQ.pdf. Consultado el 24 de Mayo del 2009.

[40] Rábanos, José María. Comunicaciones Móviles segunda edición. Editorial centro de estudios Ramón Areces. Universidad Politécnica de Madrid, 1998.

[41] S. Haykin, Sistemas de comunicación, 4 edición Limusa Willey.

[42] E. Ponce, E. Molina, y V. Mompo, Redes Inalámbricas IEEE 802.11. Documento disponible en: http://www.ieee802.org/11. Consultado el 20 de Febrero del 2009.

[43] D. Hernanz, Estudio de la capa física del 802.11, Abril 2002. Documento disponible en: http://agora.ya.com/biblio81/wireless/802-11-PHY.pdf. Consultado el 20 de Mayo del 2009.

[44] L. Boggio y C. Lana, Multiplexación y acceso múltiple, 1999. Documento disponible en: http://www.eie.fceia.unr.edu.ar/ftp/Comunicaciones/MUX.PDF. Consultado 30 Abril del 2009.

[45] C. Carreón, V. Gatica, Evaluación del Desempeño del Protocolo Aloha. Trabajo de grado, Instituto Politécnico Nacional, México, Nov. 2008. Documento disponible en: http://itzamna.bnct.ipn.mx:8080/dspace/bitstream/123456789/1714/1/Evaluaciodsempenrotocolo.pdf. Consultado el 30 de Mayo del 2009.

[46] J. Alcober, Aportación al Estudio de Protocolos de Acceso Múltiple: El protocolo Aloha Estabilizado de Ventana, Universidad Politécnica de Cataluña, Mar. 1997. Documento disponible en: http://www.tdx.cesca.es/TDX/TDX_UPC/TESIS/AVAILABLE/TDX-0727105-123058//01Jaas01de01.pdf. Consultado el 22 de Octubre del 2009.

[47] M. Alonso Cuevas, M.Casas Lago,"Television Digital Via Satelite", Escuela Tecnica Superior de Ingenieros de Telecomunicacion de Vigo, España 2003. Documeto disponible en: http://www.com.uvigo.es/asignaturas/scvs/trabajos/curso0203/dvb-s/DVB-S.pdf. Consultado el 30 de Octubre del 2009.

[48] Transmision de señales de TV digital por satélite, DVB-S, documento PDF disponible en:http://www.books.rohde-schwarz.com/go/rohdeschwarz/_ws/resource/_ts_1246551688329/rO0ABXQAJXN0YXRfcHJpdjpwcm9kdWN0czp3Zl9lc190cGxyZGR2eWE7ZW4=/data_info_en/digitv_sp_lp.pdf. Consultado el 30 de Octubre del 2009.

[49] H. Barbara, J.Chafla, Simulación de una red VSAT Full-Duplex para el acceso a internet usando la plataforma DVB-S y DVB-RCS", Escuela Politécnica Nacional, Quito Ecuador Marzo 2006. Documento disponible en: http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/415/1/2006AJIEE-21.pdf. Consultado el 20 Mayo del 2009.

[50] H. Barbara, Capitulo 2 transmision de señales de TV digital por satélite DVB-S, BIEE DSpace Escuela Poliécnica Nacional,Facultad de Ingeniería Eléctrica y Electrónica, Ecuador 2006, documento consultado en: http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/1419/4/T11385%20CAP%202.pdf

[51] L. Virues, “Características de las Comunicaciones por Satélite”. Documento PDF disponible en: http://www.monografias.com/trabajos11/caracsat/caracsat.shtml. Consultado 24 Octubre del 2008.

[52] L. Murray, Introducción a los Códigos Convolucionales, Universidad Nacional de Rosario, 2000.http://www.eie.fceia.unr.edu.ar/~comunica/cx/conv.pdf. Consultado 22 Noviembre del 2008.

[53] R. Escamilla, Códigos para corrección de errores en comunicaciones digitales, Dic. 2004.http://ingenierias.uanl.mx/25/25_codigos.pdf . Consultado 22 Noviembre del 2008.

[54] “Trellis codec modulation”. Documento disponible en: http:// www.complextoreal.com. Consultado 22 Noviembre del 2008.

[55] E. Montoya, control de errores, Universidad de la EAFIT. Documento disponible en: http://www.mhe.es/cf/ciclos_informatica/844819974X/archivos/unidad4_recurso5.pdf. Consultado 18 Diciembre del 2008.

[56] M. Allman, D. Glover, y L. Sanchez, "*Enhancing TCP over Satellite Channels using Standard Mechanism*", IETF RFC 2488, En. 1999. Documento disponible en: http://www.faqs.org/rfcs/rfc2488.html. Consultado18 Noviembre 2008.

ANEXO C

RESULTADOS OBTENIDOS CON LA VERSIÓN TCP HYBLA.

Evaluación del desempeño del protocolo TCP Hybla, para el escenario 1, con las políticas de asignación al medio por defecto.

Escenario 1, configurado con los parámetros de la tabla 1 slot1.

Análisis del comportamiento de TCP Hybla.

Tamaño de segmento 1420 paquetes, y ventana ofrecida 800, opción habilitada *timestamp*.

```
# Parámetros del objeto solicitador en cada nodo.
set streq       [$hub install-requester Requester/Constant]
set dtreq(0)    [$rcst(0) install-requester Requester/RVBDC]
set dtreq(1)    [$rcst(1) install-requester Requester/Constant]
set dtreq(2)    [$rcst(2) install-requester Requester/Constant]
set dtreq(3)    [$rcst(3) install-requester Requester/Constant]
set dtreq(4) [  [$rcst(4) install-requester Requester/Constant]
```

TCP Hybla en redes satelitales, es una versión que presenta un buen rendimiento en entornos satelitales caracterizados por tener grandes retardos de propagación alrededor de los 300 ms para redes GEO, el objetivo principal de este mecanismo es eliminar la dependencia del RTT para determinar el valor de ventana de congestión, para evitar este problema TCP Hybla ajusta su tasa de transmisión de forma independiente al RTT y para evitar los efectos del RTT, hace uso de las opción SACK para informarle al transmisor de las múltiples pérdidas que se presenta cuando se hace uso de una ventana más grande y se emplea la opción *timestamp* para ajustar el valor RTO que en el momento en que ocurren muchas pérdidas por lo general este valor no se actualiza oportunamente [1]. Los resultados encontrados en TCP Hybla se enfocan en verificar cual es su desempeño al modificar parámetros de la capa de enlace de datos para luego determinar cómo afectan a los parámetros propios de TCP como son el tamaño de ventana, el *ssthresh*, RTT, *backoff*, *ndatapack*, el número de secuencia, número de retransmisiones, y el número de ACKs. Los parámetros a variar en la simulación se indican en la tabla 1 igualmente se indica los scripts utilizados en la simulación.

Tabla 1. Parámetros de configuración para los diferentes scripts de simulación.

Script	Lengs2	Lengs1	Slot1	Slot2	Slot3	Slot4
Duración trama	0,04	0,05	0,04	0,05	0,06	0,04
N^0 Slots	14	14	14	20	24	16
Tamaño Slot	1504	1504	184	1504	1504	1504
Ancho de Banda	4.211.200	3.368.960	515.200	4.812.800	4.812.800	4.812.800
Supertrama	25	20	25	20	17	25

Para la descripción de TCP Hybla se emplean los mismos parámetros asignados en la tabla 1 con el script Slot1, al observar la figura 1 se indica el tamaño de ventana de congestión y *ssthresh*, se observa que TCP Hybla tiene una apertura más rápida de la ventana de congestión aprovechando adecuadamente los recursos de la red; para calcular el valor inicial del *ssthresh*,

TCP Hybla utiliza el algoritmo de Hoe que realiza una estimación del ancho de banda de la red, este valor se ajusta al PDB de la red.

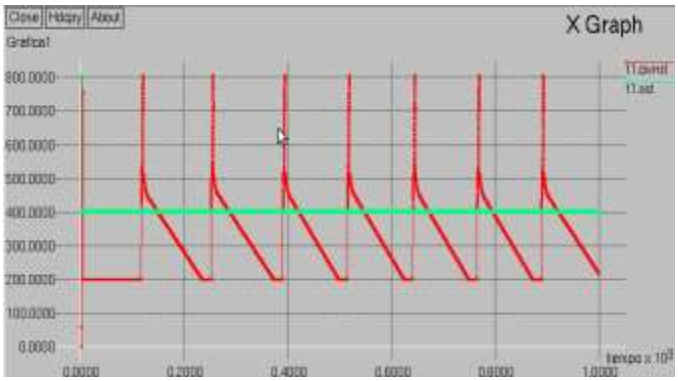


Figura 1 Comportamiento ***cwnd*** y ***ssthresh*** para el script Slot1.

En la figura 1 se observa el comportamiento de los mecanismos de inicio lento y evasión de congestión, debido a que TCP Hybla utiliza una ventana de congestión más grande, el número de pérdidas se incrementa de igual forma como la ventana se desempeñe, durante la fase de recuperación el número de segmentos transmitidos disminuye como se observa en las figuras 1 y 2, mediante el parámetro ndatapack**,** al final de esta fase se observa que *cwnd* crece rápidamente hasta que se detecta una nueva pérdida, algunos motivos del comportamiento oscilante de *cwnd* y *ssthresh* pueden ser causados por ráfagas de datos que desbordan el tamaño del búfer.

Figura 2 Paquetes transmitidos por Slot1.

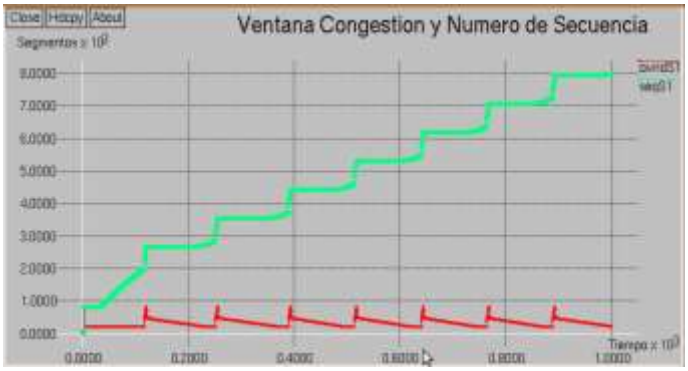
La figura 2 muestra el comportamiento de *ndatapack* el cual indica el inicio y la finalización de la fase de recuperación, en el instante que ocurre una pérdida la variable *ndatapack* presenta una disminución en su pendiente, dando inicio a la fase de evasión de congestión, al finalizar esta fase hay un aumento de los paquetes de datos transmitidos, este comportamiento es repetitivo como se observa en el intervalo 125 hasta 260 aproximadamente. En la media que aumente el tamaño de ventana, *ndatapack* también lo hará [2].

En la figura 3, se observa el comportamiento de las variables pktx y seq, cuando ocurre la pérdida de un segmento se inicia la retransmisión del segmento, el valor de pktx se incrementa hasta que la red se recupera de la pérdida, mientras tanto que el número de secuencia permanece constante, para cuando el número de secuencia aumenta significativamente pktx permanecerá constante hasta que ocurra otra pérdida es decir se inicie el mecanismo de recuperación.

Paquetes retransmitidos **número de secuencia**

Figura 3 Paquetes retransmitidos y número de secuencia por Slot1.

Cuando *cwnd* aumenta, ndatapack y seq, también lo hacen como se observa en la figura 4, los paquetes de datos transmitidos presentan un inicio muy rápido antes de entrar a la fase de recuperación.



cwnd **Número de secuencia**

Figura 4 Ventana de congestión y número de secuencia por Slot1.

Los parámetros que indican cual ha sido el comportamiento del RTO en la red es el *backoff* y *nrexmit* (número de retransmisiones del *timeout*), estos parámetros también indican cual ha sido el comportamiento de la ventana de congestión, en la figura 5 el *backoff* permanece constante en uno y *nrexmit* en cero (estos valores son los que trae por defecto el simulador), por lo tanto se confirma que no hubo ninguna retransmisión del *timeout* como se observa en el comportamiento de *cwnd* en la figura 1. Por lo general cuando el temporizador de retransmisión expira, el valor del RTO se ajusta al valor recomendado por la opción *timestamp*.

Figura 5 *Back-off* y número de retransmisiones del *timeout*.

Al habilitar la opción *timestamp*, el transmisor hace uso de los ACKs para actualizar el valor del RTT, en las figura 6 se muestra el comportamiento del RTT si este parámetro se aumenta significa que el valor ρ también lo hará lo que implica que la ventana de congestión se comportará más agresiva congestionando aún más la red [3].

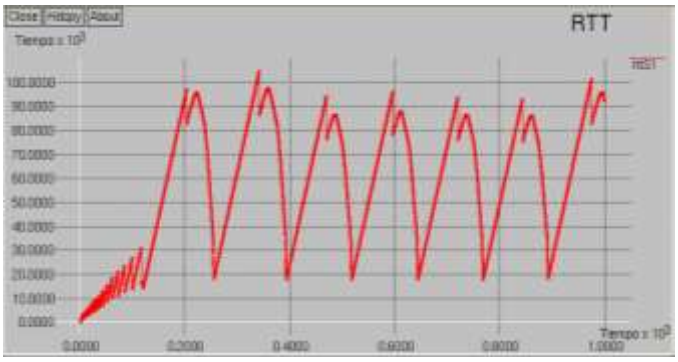


Figura 6 RTT de la conexión cero.

Para observar el comportamiento de los ACKs en el desempeño de los mecanismos de congestión, se tiene en cuenta la figura 7 en la medida que un mayor número de ACKs retornen al transmisor su tamaño de ventana se incrementa así como su número de secuencia, al producirse una pérdida el número de ACKs permanecerá casi constante, de la misma forma que el parámetro seq se ha descrito anteriormente.

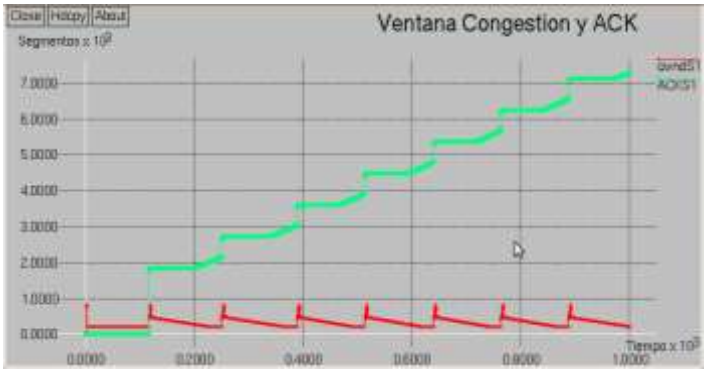
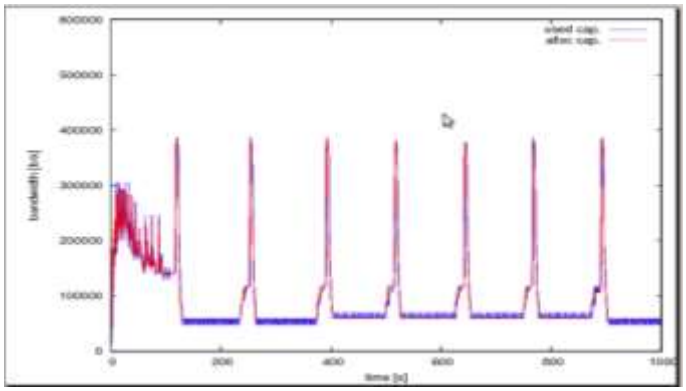


Figura 7 Relación entre la ventana de congestión y los ACKs.

Al analizar la figura 8, permite determinar cuál ha sido la capacidad usada y utilizada por cada terminal RCST, en esta gráfica se observa cual ha sido la capacidad ocupada en el enlace de subida.

Si se observa el comportamiento de *cwnd* en la figura 1 con la figura 8 se relacionan en la manera como el nodo transmisor realiza la transmisión de paquetes y como estos segmentos utilizan el ancho de banda del canal de subida, el número de slots asignados al inicio de la conexión es mayor ya que la política de crecimiento se encuentra en la fase de inicio lento, de la misma manera las asignaciones de los slots por parte de la NCC a la RCST.

En el momento en que finaliza la fase de recuperación la ventana de congestión crece más rápido así mismo que su utilización del canal para el enlace de subida, en los intervalos en el que la capacidad asignada y usada permanecen constantes coincide en que la ventana de congestión se encuentra en la fase de recuperación debido a la pérdida de un segmento.



Capacidad usada **Capacidad asignada**

Figura 8 Utilización del canal en el enlace *uplink* para la conexión cero.

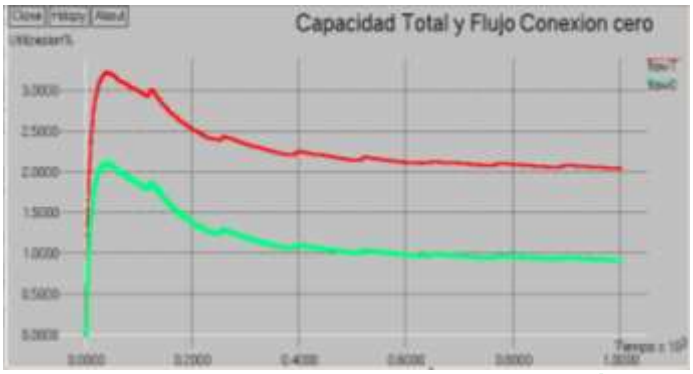
Al ejecutar el script, se crea una carpeta con el nombre tdma-dama que contiene los parámetros más importantes para cada agente de transmisión.

La figura 9 indica el tráfico que alcanza el agente sumidero de la conexión cero, los picos corresponde cuando el tamaño de ventana se encuentra al final de la fase de recuperación.

Figura 9 Bits por segundo en la conexión cero.

La figura 10 permite ver la capacidad del canal utilizada desde la estación receptora al nodo de recepción para la conexión cero mediante la variable flow0, en esta misma gráfica se observa la

capacidad utilizada de todo el canal por todas las conexiones mediante la variable flowT. Al inicio de la transmisión de segmentos la capacidad del canal es mayor, hasta que *cwnd* se estabiliza a un valor, los rizos que se observan coinciden con el incremento del tamaño de ventana.



Capacidad Total **Capacidad conexión 0**

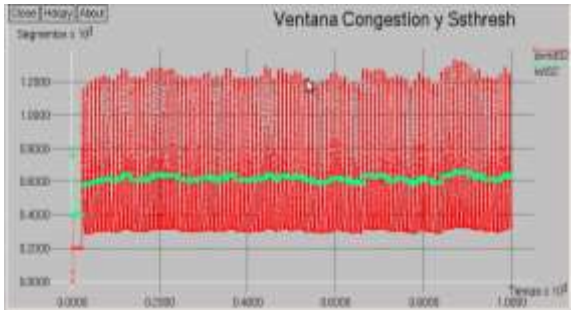
Figura 10 Capacidad Total y de la conexión cero para el Slot1.

1.0. IMPACTO AL VARIAR EL NÚMERO DE SLOTS.

La siguiente prueba permite, evaluar cuál es el impacto de utilizar un mayor o menor número de slots en el desempeño del protocolo TCP Hybla en una red satelital, para esta primera prueba se toman los script Slot2 y Lenghsl1, cada uno se ajusta a los datos de la tabla 1.

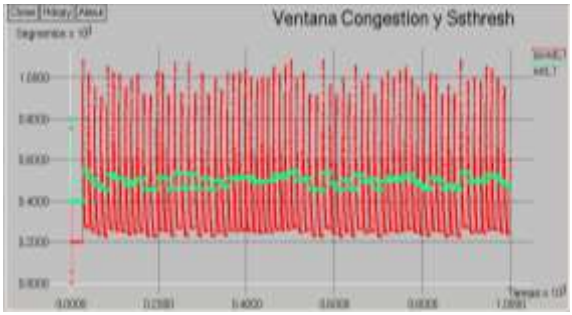
1.1. RESULTADOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

Según las simulaciones realizadas la ventana de congestión en Slot2, es más oscilante, indicando que un mayor número de paquetes son trasmitidos en un menor tiempo donde su ventana se recupera más rápido de la fase de recuperación, el nivel de *ssthresh* se ajusta a un nivel más alto lo que implica que un mayor número de paquetes son transmitidos, esto se debe a que se está haciendo uso de un mayor ancho de banda. En la figura 11 se muestra el comportamiento *cwnd* y el *ssthresh*, para los dos scripts.



Ssthresh ***cwnd***

Figura11.1 Ventana de congestión y *ssthresh* en el script Slot2.

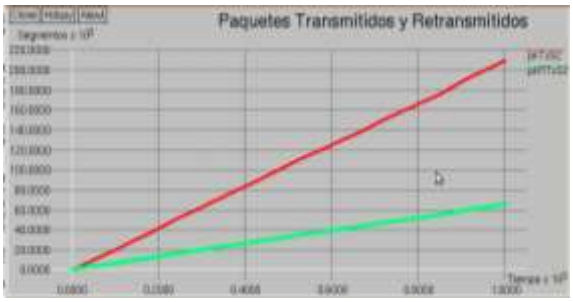


Ssthresh ***cwnd***

Figura11.2 Ventana de congestión y *ssthresh* en el script Lenghsl1.

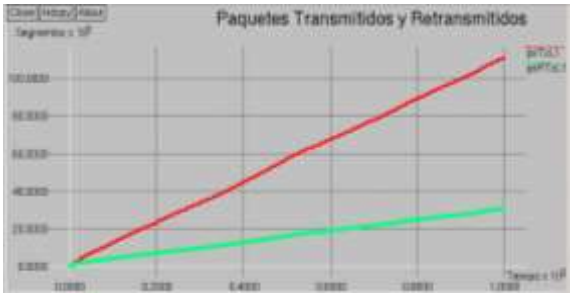
Entre más estrecha sea la ventana de congestión y oscile sobre un nivel del *ssthresh* más alto, el número de paquetes transmitidos será mayor, que es el caso del escenario Slot2, como lo indica la figura 11.

Los paquetes retransmitidos aumentan considerablemente para un mayor ancho de banda, en cambio para el script Lenghslt1 maneja una ventana más pequeña que implica un menor número de paquetes serán transmitidos y por lo tanto los paquetes retransmitidos también se disminuyen, a continuación se muestran las gráficas 12 que describen este comportamiento.



Paquetes Tx **Paquetes retransmitidos**

Figura 12.1 Paquetes transmitidos y retransmitidos por el script Slot2.



Paquetes Tx **Paquetes retransmitidos**

Figura 12.2 Paquetes transmitidos y retransmitidos por el script Lenghsl1.

1.2. DATOS OBTENIDOS DE ACK, RTT Y NÚMERO DE SECUENCIA.

Un mayor número de ACKs recibidos por el receptor indica que la ventana de congestión tendrá una mejor apertura, en el caso de TCP Hybla luego de que ocurre la primera pérdida de un segmento la ventana se configura a la fase de recuperación, el trasmisor está esperando que lleguen ACKs para incrementar su ventana, según lo observado en las simulaciones un mayor ancho de banda permite recuperarse más rápido de esta fase, esto se visualiza en el tiempo en que los ACKs empiezan a incrementarse de una forma más vertiginosa, y en el crecimiento de la ventana de congestión. A continuación se muestra la figura 13 para apreciar el comportamiento de los ACKs.

ACKs Slot2 **ACKs Lengsl1**

Figura 13 Crecimiento de los ACKs para los dos scripts.

Lengsl1 presenta un RTT mayor lo que implica que un menor número de ACKs pueden llegar al trasmisor y el número de paquetes transmitidos disminuye, en TCP Hybla el valor del RTT se incrementa cuando se encuentra al final de la fase de recuperación y disminuye después de

que ha ocurrido una pérdida. A continuación se indica la figura 14 muestra el comportamiento del RTT para los dos scripts.

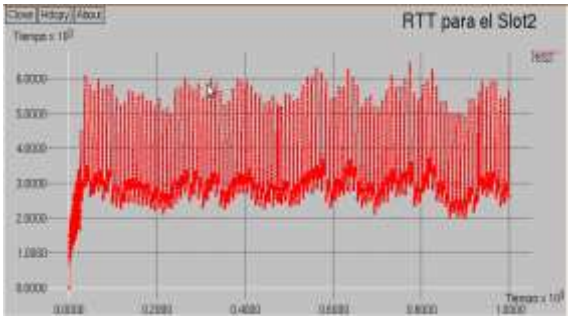


Figura 14.1 Comportamiento del RTT en el Slot2.

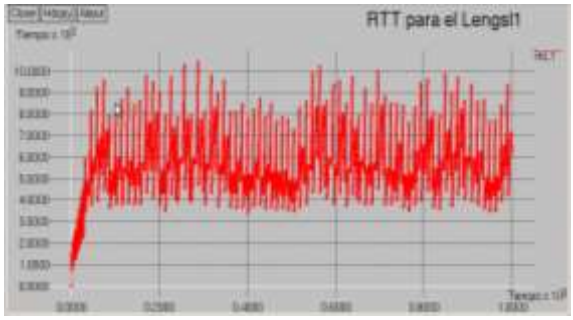
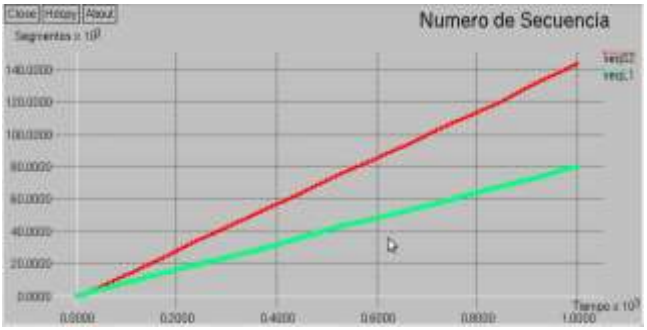


Figura 14.2 Comportamiento del RTT en Lenghsl1.

El comportamiento general del número de secuencia se incrementa de forma lineal, donde su rizado indica el momento en que se encuentra en fase de recuperación entre más pequeño sea este intervalo significa que la ventana se recupera más rápido. En Slot2, el patrón general indica que este rizado es menor y el número de secuencia es mucho más alto que en Lengsl1, como lo indica la figura 15.



N° Secuencia Slot2 **N° secuencia Lengsl1**

Figura 15. Evolución del número de secuencia.

1.3. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO DE LA RED.

Un menor retardo significa que la red no se encuentra tan congestionada, el RTT que percibe la red también es menor, por lo tanto un mayor número de paquetes serán transmitidos, este es el caso de Slot2, mediante la figura 16 se muestra el comportamiento del retardo para los dos scripts.

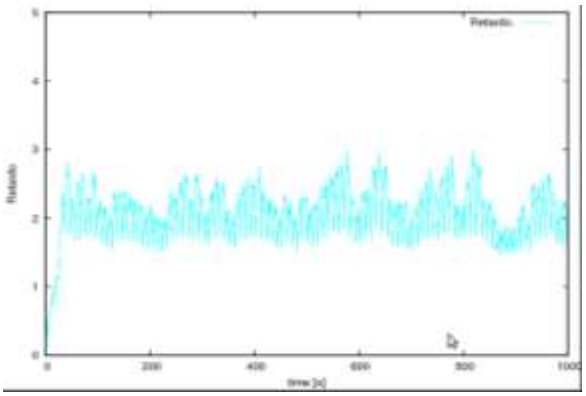


Figura 16.1 Retardo de la red en el script Slot2.

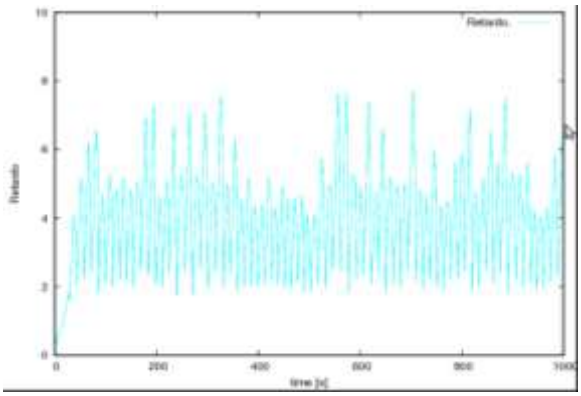


Figura 16.2 Retardo de la red en el script Lenghsl1.

El encolamiento está estrechamente relacionado con el comportamiento de la ventana de congestión, entre menos espaciada esté, indica un mayor número de paquetes que han sido transmitidos, para la gráfica 17.2 se visualiza un mayor encolamiento al inicio de la transmisión de datos debido a que la ventana le toma más tiempo recuperarse de la primera pérdida, además su encolamiento está más espaciado que en el script Slot2 lo que conlleva a que un menor número de paquetes sean transmitidos.

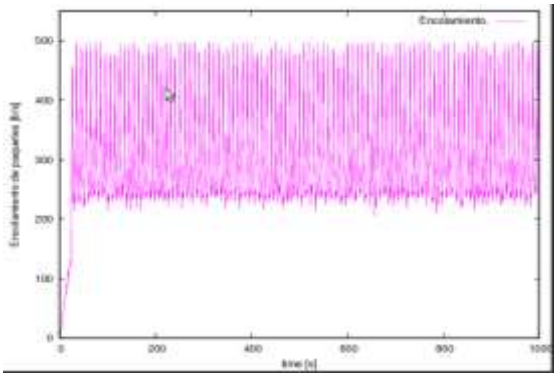


Figura 17.1 Encolamiento de paquetes en Slot2.

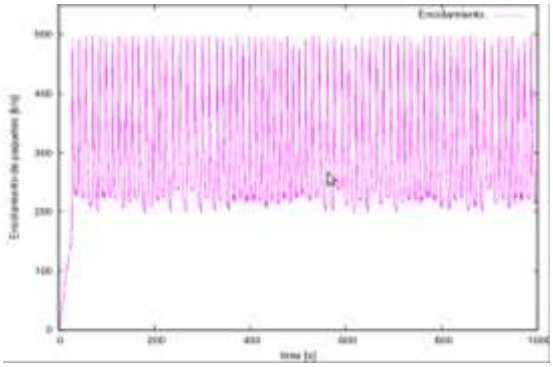


Figura 17.2 Encolamiento de paquetes en Lenghsl1

1.4. DATOS OBTENIDOS DEL *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

El *throughput* total alcanzando por todas las conexiones se indica en la figura 18, donde Slot2 presenta un mejor rendimiento, ya que el volumen de datos transmitidos es mayor que en Lengsl1.

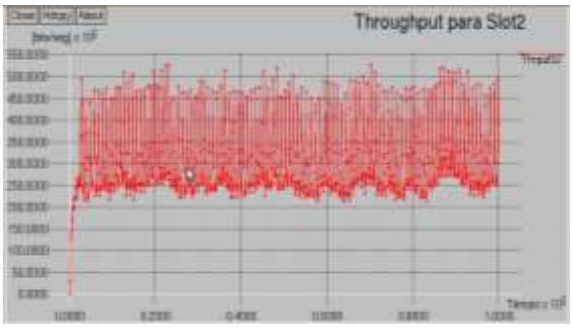


Figura 18.1 *Throughput* para Slot2.

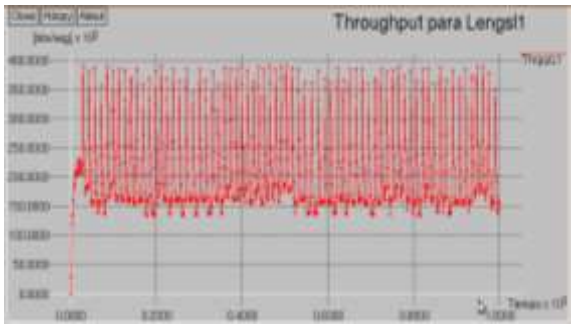
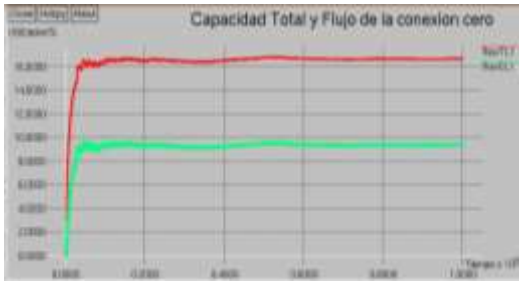


Figura 18.2 *Throughput* para Lenghsl1.

La capacidad del enlace utilizado indica que porcentaje del canal está siendo utilizado por la red, la figura 19 muestra el comportamiento para los dos casos de evaluación.

Capacidad Total **Conexión cero**

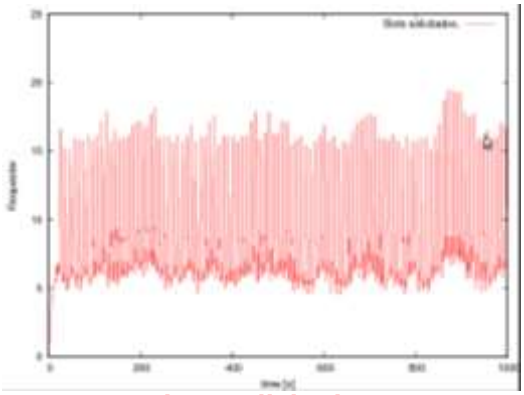
Figura 19.1 Capacidad Total y flujo de la conexión cero en Slot2.



Capacidad Total **Conexión cero**

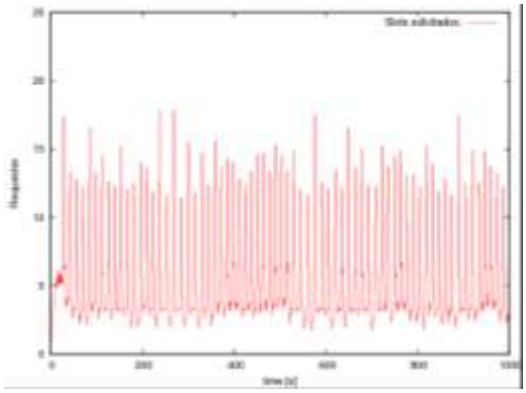
Figura 19.2 Capacidad Total y flujo de la conexión cero en Lenghsl1.

Las figuras 20**,** indican las solicitudes realizadas a la NCC cada 50 milisegundos que corresponden a la duración de la trama**,** se puede ver que las peticiones para Slot2 son más estrechas y retornan sobre un valor más alto que en Lenghsl1, además no permanecen mucho tiempo sobre un mismo nivel.



Slots solicitados

Figura 20.1 Solicitudes RVBDC a la NCC en el Slot2.



Slots solicitados

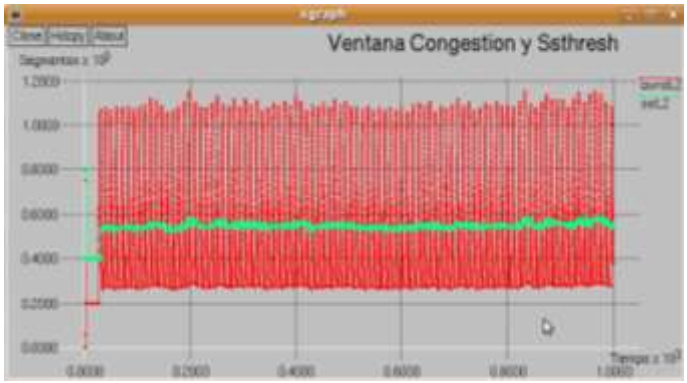
Figura 20.2 Solicitudes RVBDC a la NCC en Lenghsl1.

2.0. IMPACTO AL VARIAR EL TAMAÑO DEL SLOT.

La siguiente prueba permite evaluar el impacto que tiene utilizar tamaños de slots diferentes en el desempeño del protocolo TCP, para esta prueba se configuraron el script Slot1 y Lenghsl2, cada uno se configuró de acuerdo a los datos de la tabla 1.

2.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

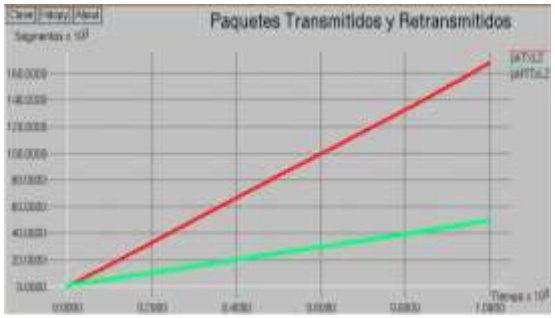
La ventana de congestión en Slot1, indica que el valor del *ssthresh* se encuentra en un menor valor que la ventana Lengsl2, además no alcanza su mismo tamaño. El comportamiento observado en las simulaciones indican que un menor tamaño del slot requiere un mayor número de paquetes a transmitir para obtener un mejor rendimiento, la ventana en Slot1 indica que tarda mucho tiempo en restaurarse de la fase de recuperación al momento de una pérdida, en los primeros segundos la ventana permanece constante lo que da entender que se encuentra en congestión, los ACKs se están demorando mucho más tiempo para poder incrementar su ventana. En la figura 21 se indica el valor almacenado para *cwnd* y *ssthresh* para los escenarios en evaluación.



Cwnd ***Ssthresh***

Figura 21. Comportamiento de la ventana congestión y *ssthresh* en Lenghsl2.

Un mejor crecimiento de la ventana, siempre indicará un mayor volumen de paquetes transmitidos, sin embargo una ventana grande implica muchas más pérdidas, la línea rizada en el script Slot1 indica cuanto tiempo se tarda el trasmisor en salir de la fase de recuperación, caso contrario se observa con la variable ndatapack en Lengsl2 su comportamiento es casi lineal. Las figuras 22 muestran las variables pktx y ndatapack para los dos escenarios.



Paquetes Tx **Paquetes Retransmitidos**

Figura 22. Paquetes transmitidos y retransmitidos por Lenghsl2.

Paquetes Tx **Paquetes Retransmitidos**

Figura 22. Paquetes transmitidos y retransmitidos por Slot1.

2.2. DATOS OBTENIDOS ACK, RTT Y NÚMERO DE SECUENCIA.

El número de ACKs al utilizar un tamaño de slot menor, genera en la red una permanentemente congestión, debido a que la red está esperando transmitir un mayor volumen de datos, pero como los ACKs se retrasan, el retardo en la red aumenta y por lo tanto el RTT de la red se va incrementado, obteniendo un peor rendimiento de la ventana de congestión. Al utilizar un tamaño de slot más grande con un ancho de banda mayor los paquetes de datos transmitidos periódicamente están aumentando y hacen uso de una mayor capacidad de la red, lo que permite que los ACKs estén continuamente creciendo. En la figura 23 se aprecia el comportamiento del ACK para los dos scripts.

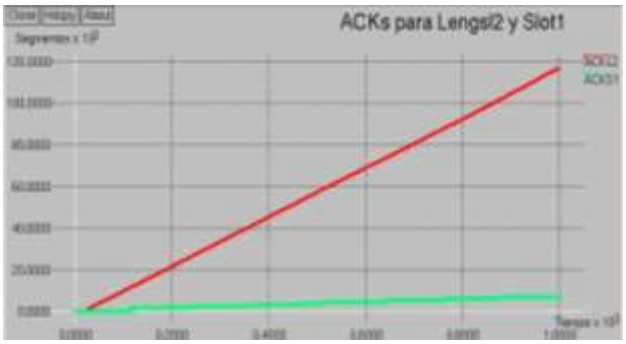


Figura 23 Evolución de los ACKs en Lenghsl2 y Slot1.

El RTT en el escenario Slot1 se incrementa cada vez que la ventana de congestión está transmitiendo un mayor número de paquetes, si se continua con este incremento la red llega a un punto de congestión que la ventana tendrá que reducirse por la gran cantidad de pérdidas; disminuyendo nuevamente el RTT, según la figura 24 el RTT para el escenario Lengsl2 es bastante menor, ésta es una de las razones de su buen desempeño.

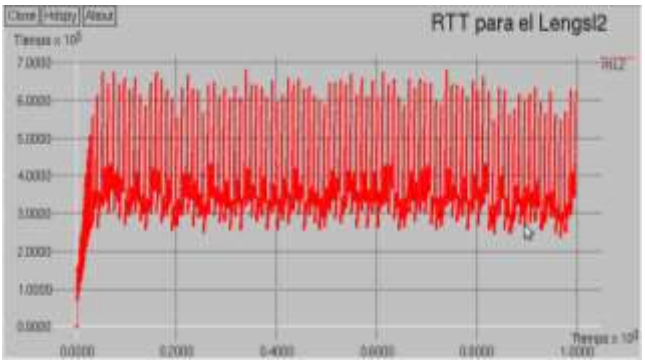
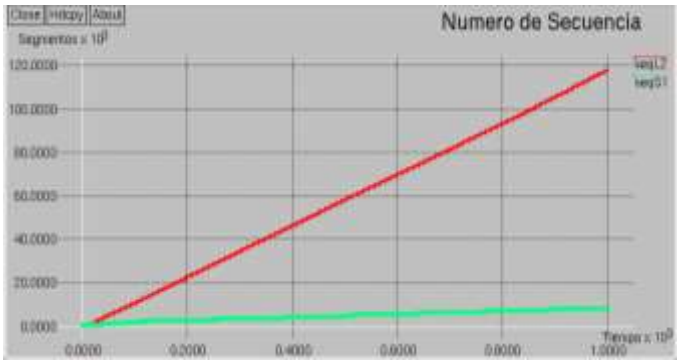


Figura 24. Comportamiento del RTT para el script Lenghsl2.

El número de secuencia se relaciona en la forma como sean recibidos los ACKs, entre más datos se han confirmados el parámetro sec se incrementará instantáneamente, indicando mayor volumen de datos transmitidos. Al presentarse mucha congestión su incremento es lento tal y como se observa en la figura 25.



N° Secuencia Lenghsl2 N° Secuencia Slot1

Figura 25 Evolución del Número de Secuencia para Lenghsl2 y Slot1.

2.3. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO EN LA RED.

El retardo en los dos scripts es muy diferente, en Lengsl2 es menor debido a que la red no esta tan congestionada a pesar de que presenta un mayor volumen de datos a transmitir, el retardo presenta un comportamiento similar a la variación del RTT, como lo indica la gráfica 26. En Slot1 su retardo es muy alto evitando aumentar su ventana de congestión sin la llegada oportuna de los ACKs.

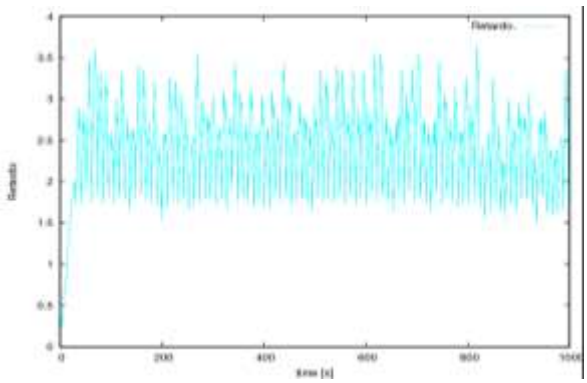


Figura 26.1 Retardo en la red para Lengsl2.

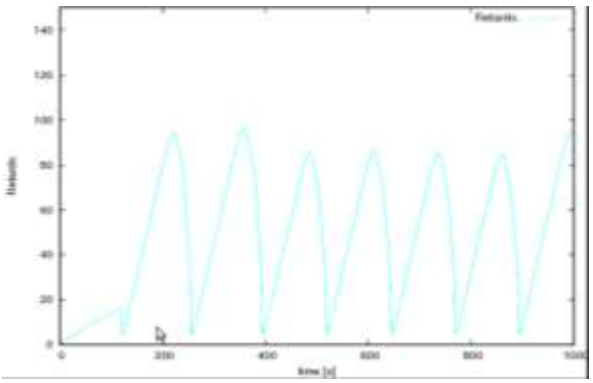


Figura 26.2 Retardo en la red para Slot1.

Un buen encolamiento de paquetes se presenta cuando se desocupan el mayor número de paquetes en el menor tiempo, una mayor pendiente da entender que un gran volumen de datos han sido transmitidos, este es el caso de Lengsl2, caso contrario ocurre en Slot1, tal como se indica en las figuras 27. En el momento de que ocurre una pérdida la red necesita transmitir los paquetes que se encuentran encolados en el búfer de la estación solicitadora, sin embargo si la red se encuentra congestionada le tardara más tiempo desocuparse de ellos lo que implica que el receptor no puede enviar ACKs para que se incremente su ventana de congestión.

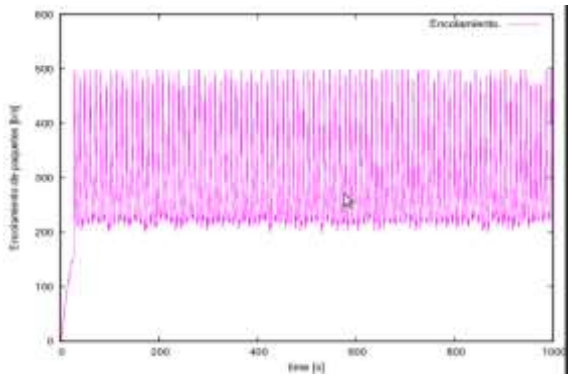


Figura 27.1 Encolamiento en el terminal transmisor en Lengsl2.

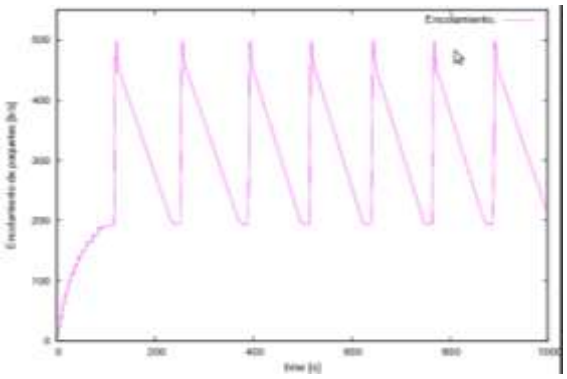


Figura 27.2 Encolamiento en el terminal transmisor en Slot1.

2.4. DATOS OBTENIDOS DEL *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

El impacto que se ocasiona en una red al utilizar un tamaño de slot pequeño y un menor ancho de banda en la asignación de recursos se observa en las gráficas 28 y 29, donde el escenario configurado con un tamaño de slot más grande presenta un rendimiento muy superior al obtenido por Slot1, así mismo la capacidad total empleada por las diferentes conexiones presenta una mejor eficiencia.

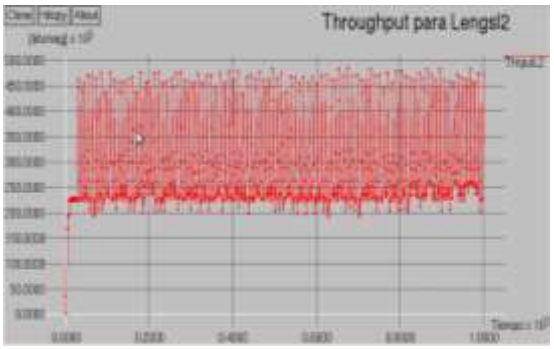


Figura 28.1 Evolución del *throughput* total de la red en Lengsl2.

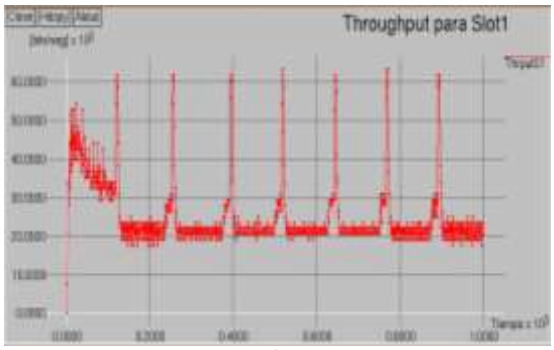
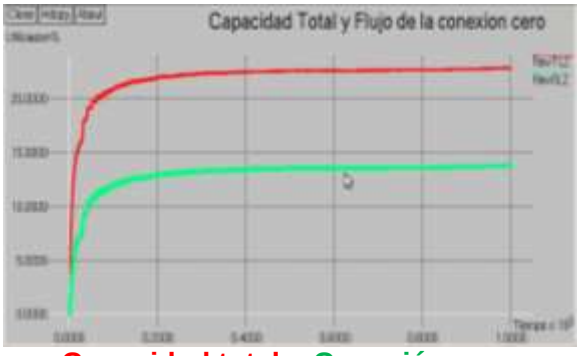


Figura 28.2 Evolución del *throughput* total de la red en Slot1



Capacidad total **Conexión cero**

Figura 29. Capacidad Total y Flujo de la conexión cero en Lengsl2.

3.0. IMPACTO AL VARIAR LA DURACIÓN DE LA TRAMA.

La siguiente prueba se diseño para medir el impacto que presenta variar la duración de la trama en el desempeño del protocolo TCP, para esta prueba se configuran los valores de la tabla 1 a los scripts de estudio Slot2 y Slot4.

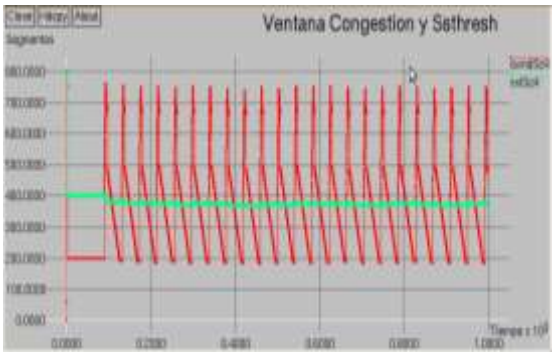
Los resultados arrojados en la simulación indican que a menor duración de trama el rendimiento es mejor, teniendo en cuenta que a partir de un valor umbral el rendimiento se empeora.

En la versión TCP Hybla los valores obtenidos, para cada unas de las conexiones son muy similares, el patrón que se observo fue que para la primer conexión una duración de trama más grande presenta mejor rendimiento, sin embargo para las demás conexiones una menor duración de trama su rendimiento es ligeramente mejor.

3.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

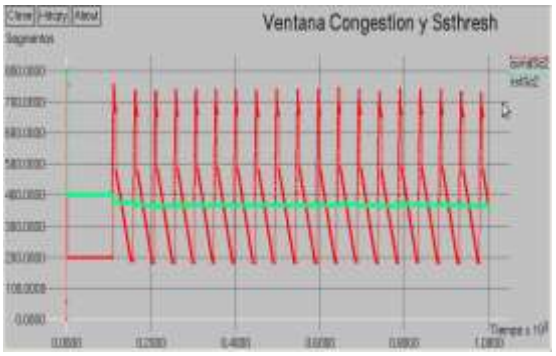
La ventana de congestión para el script que presenta menor duración de trama indica que su nivel del *ssthresh* se encuentra en un nivel más alto, por lo tanto indica que se están transmitiendo más datos en los nodos transmisores; lo que conlleva a una mejor utilización del canal y alcanzar un buen rendimiento. En los intervalos donde la ventana de congestión se disminuye significa que la red ha experimentado un mayor retardo causado por un incremento en la congestión. La ventana de Slot2 tarda más tiempo en obtener un buen tamaño de ventana en la fase de inicio lento, y en su fase de evasión de congestión le cuesta más tiempo recuperarse de las pérdidas ocasionadas por el sudimensionamiento del búfer. En la figura 30 se indica los parámetros *cwnd*, *ssthresh* para los dos escenarios.

Los paquetes transmitidos por el escenario Slot4 son mayores que en el escenario Slot2, sin embargo la proporción se mantiene al aumentar la ventana con respecto a los paquetes retransmitidos, como lo indica las figuras 31 para los dos escenarios de simulación. Un mayor número de paquetes transmitidos significa que la ventana de congestión será más oscilante, esa oscilación es debido a que el tamaño del búfer de la estación receptora es pequeño en el instante que un paquete es descartado por el búfer, se le informa al transmisor que reduzca su ventana, y por lo tanto la ventana se recuperará de la pérdidas según sea el nivel de congestión que presente la red.



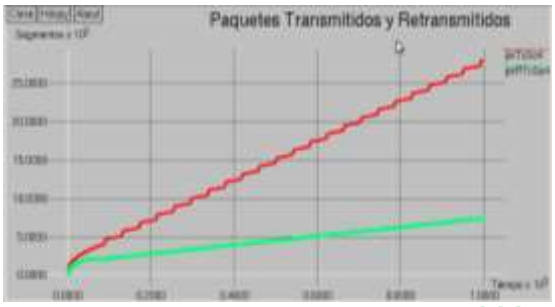
Cwnd ***ssthresh***

Figura 30.1 Comportamiento de la ventana congestión y *ssthresh* en Slot4.



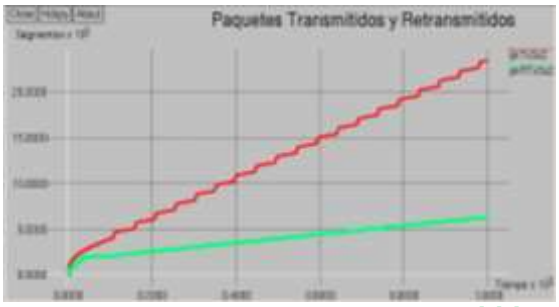
Cwnd ***ssthresh***

Figura 30.2 Comportamiento de la ventana congestión y *ssthresh* en Slot2.



Paquetes Tx Paquetes Retransmitidos

Figura 31.1 Paquetes transmitidos y retransmitidos por la conexión uno en Slot4.



Paquetes Tx Paquetes Retransmitidos

Figura 31.2 Paquetes transmitidos y retransmitidos por la conexión uno en Slot2.

3.2. DATOS OBTENIDOS PARA EL ACK Y RTT.

Para el escenario Slot4, durante el inicio de la transmisión los ACKs se incrementan más rápido permitiendo transmitir una mayor cantidad de datos al comienzo de la simulación. En las figuras 32 se indica su comportamiento para los dos scripts.

Slot 4 Slot2

Figura 32 Evolución de los ACKs para Slot4 y Slot2 en la conexión dos.

Los valores obtenidos por el RTT son muy similares, sin embargo el caso Slot4 presentó un mejor comportamiento, con un RTT menor los paquetes a transmitir se incrementan y de esta manera el receptor podrá enviar más ACKs al transmisor. Mediante las figuras 33 se observa el comportamiento del RTT para los scripts Slot2 y Slot4.

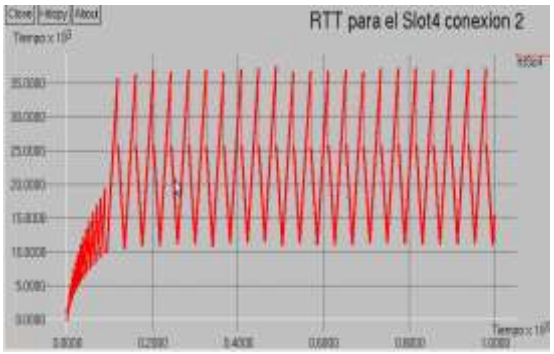


Figura 33.1 Comportamiento del RTT para la conexión dos para el Slot4.

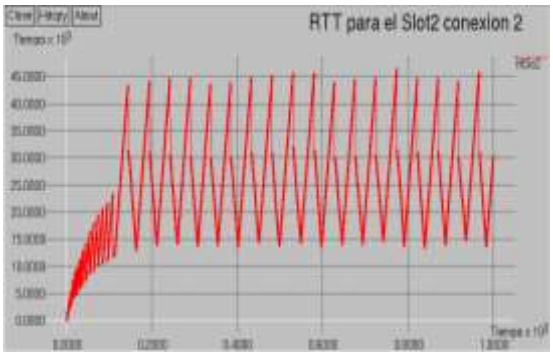


Figura 33.2 Comportamiento del RTT para la conexión dos para el Slot2.

3.3. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO DE LA RED.

El encolamiento para esta conexión indica que el escenario Slot4 presenta un mayor número de paquetes encolados, si se tiene en cuenta que el desempeño de los paquetes transmitidos es bueno y se obtiene un menor retardo. Cuando se presenta una pérdida la ventana de congestión se recupera más rápido y permite que la ventana nuevamente reinicie con la transmisión de segmentos hacia el receptor. En la figura 34 se observa el comportamiento anteriormente descrito.

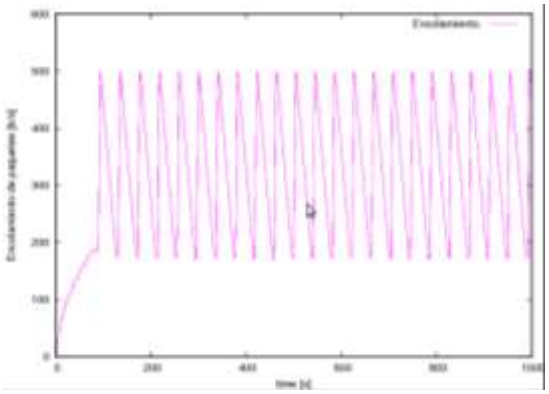


Figura 34.1 Encolamiento de paquetes para el terminal dos con el script Slot4.

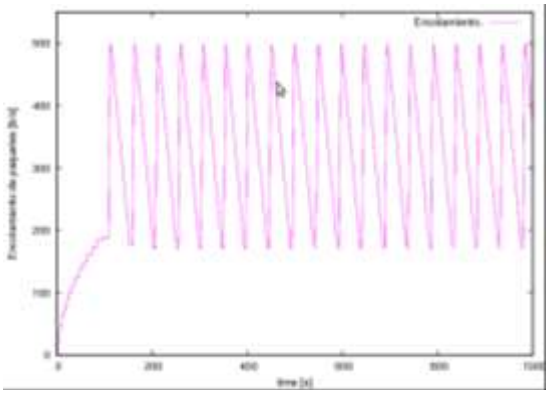


Figura 34.2 Encolamiento de paquetes para el terminal dos con el script Slot2.

El retardo para las dos conexiones es malo indicando que su desempeño se ve muy afectado por la congestión en la red, las figuras 35 muestran el encolamiento para los dos escenarios.

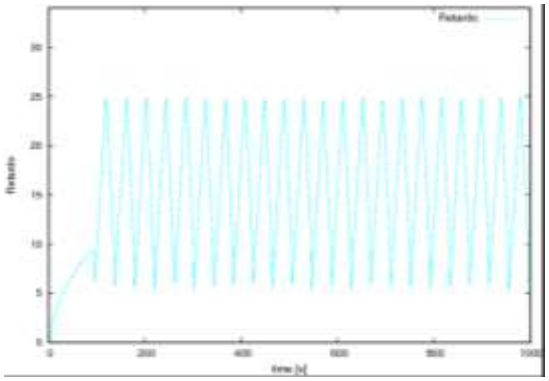


Figura 35.1 Retardo en la red para el script Slot4.

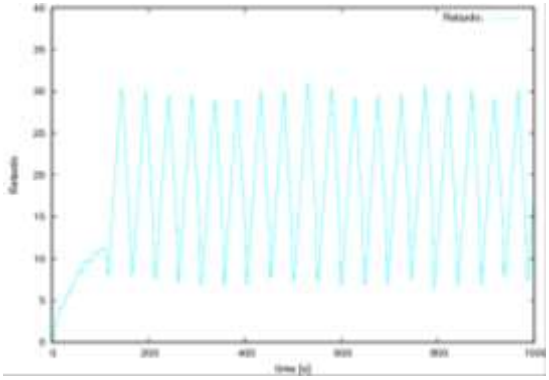


Figura 35.2 Retardo en la red para el script Slot2.

3.4. DATOS OBTENIDOS DEL *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

La utilización del canal durante los primeros segundos para Slot4 es mejor, conllevando a un mayor crecimiento de la ventana de congestión**,** en general un menor tiempo en la trama implica un mejor uso de la utilización del canal y por lo tanto presenta un nivel mayor de rendimiento**.** Las figuras 36 y 37, visualizan el comportamiento del *throughput* y utilización total de la red, para los dos escenarios.

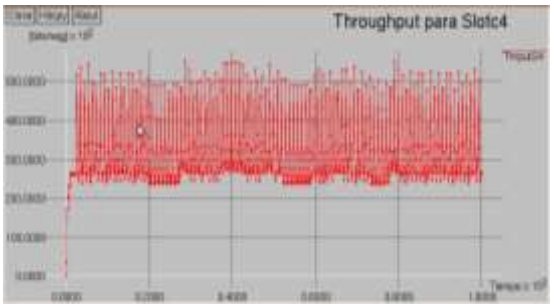


Figura 36.1 Comportamiento del *Throughput* en Slot4.

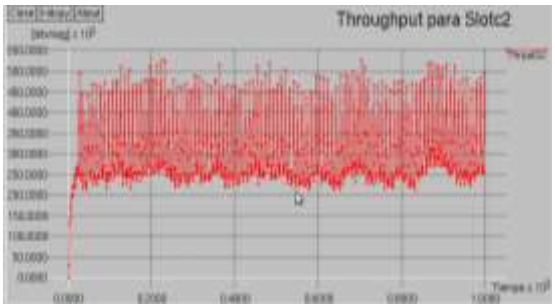
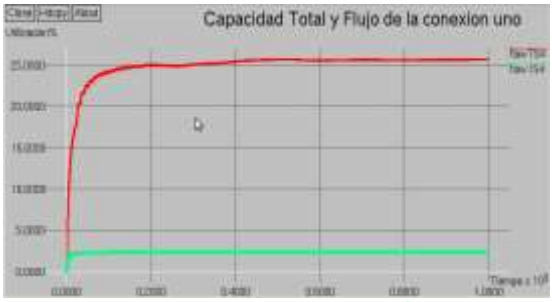
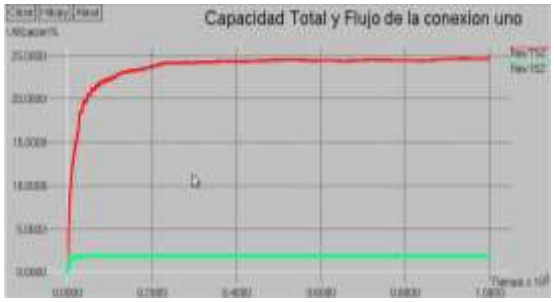


Figura 36.2 Comportamiento del *Throughput* en Slot2.



Capacidad total **Conexión Dos**

Figura 37.1 Capacidad total y flujo de la conexión dos para el Slot4.



Capacidad total **Conexión Dos**

Figura 37.1 Capacidad total y flujo de la conexión dos para el Slot2.

4.0. IMPACTO AL VARIAR EL TIEMPO DE REPROGRAMACIÓN TBTP.

La siguiente prueba se diseño para medir el impacto que presenta variar la duración del TBTP en el desempeño del protocolo TCP, para esta prueba se configuran los valores de la tabla 2 a los scripts TBTP y 1.5TBTP**.** El único dato que varía en esta prueba es el número de supertramas las cuales permiten fijar el valor del tiempo de reprogramación TBTP en la estación de control, de acuerdo a los resultados obtenidos un mayor tiempo de planificación para la actualización de las asignaciones del ancho de banda obtiene un mejor rendimiento en la red. En [4] sugiere un TBTP alrededor de 1 segundo, los datos a evaluar en esta prueba son 0.8 y 1.5 segundos.

Tabla 2. Parámetros de configuración en la prueba TBTP.

Script	**TBTP**	**1.5TBTP**
Duración trama	0,025	0,025
Número Slots	24	24
Tamaño Slot	534	534
Ancho de Banda	4.101.120	4.101.120
Supertrama	32	44

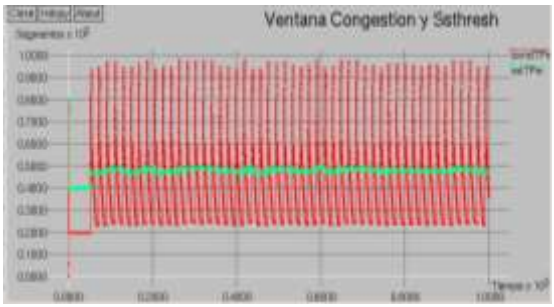
4.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

La ventana de congestión en TBTP indica que se recupera más rápido de la primer pérdida por esta razón su ventana crece primero, su ventana se caracteriza por un nivel del *ssthresh* mayor, sin embargo las dos ventanas se recuperan de la misma forma. Un tiempo de planificación de

asignación de recursos menor es más conveniente ya que permite una mayor apertura al inicio de la transmisión. A continuación en la figura 38 se observa el comportamiento de *cwnd* y *ssthresh* para los dos valores de TBTP.

Cwnd **ssthresh**

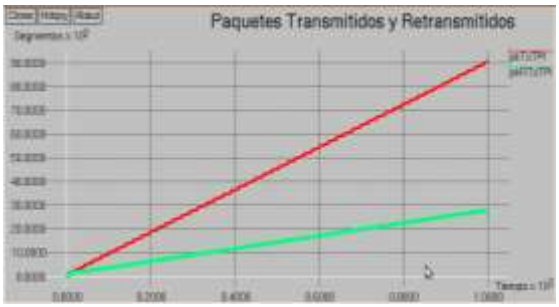
Figura 38.1 Comportamiento de la ventana de congestión y *ssthresh* en el script TBTP.



Cwnd **ssthresh**

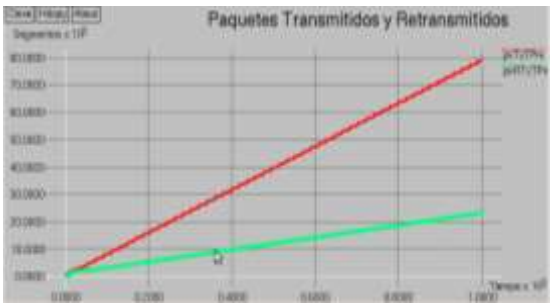
Figura 38.2 Comportamiento de la ventana de congestión y *ssthresh* en el script 1.5TBTP.

Los paquetes transmitidos indican cuál es el comportamiento de la ventana de congestión, y definen en qué fase se encuentra la ventana de congestión. Como era de esperarse la variable ndatapack para el escenario TBTP presenta un mejor desempeño, lo que conlleva a obtener un mayor *throughput* en la red, tal como lo indican las figuras 39.



Paquetes Tx **Paquetes Retransmitidos**

Figura 39.1 Paquetes transmitidos y retransmitidos en el script TBTP.



Paquetes Tx **Paquetes Retransmitidos**

Figura 39.2 Paquetes transmitidos y retransmitidos en el script 1.5TBTP.

Los paquetes retransmitidos presentes en TCP Hybla, son bastantes teniendo en cuenta a los obtenidos en TCP Westwood y TCP HS, lo cual es un comportamiento característico de la versión de TCP Hybla.

4.2. DATOS OBTENIDOS DE ACK, RTTVAR Y NÚMERO DE SECUENCIA.

En todas las simulaciones realizadas para los dos scripts en evaluación se observó que al utilizar un TBTP menor, el incremento de los ACKs ocurre más pronto para el primer intervalo donde se finaliza la fase de recuperación, este período es importante ya que indica que tan congestionada puede estar la red. Si se aprecia la figura 40, el incremento de los ACKs se relaciona con el número de secuencia en la medida que más ACKs sean recibidos por la fuente origen la variable sec se incrementa instantáneamente.

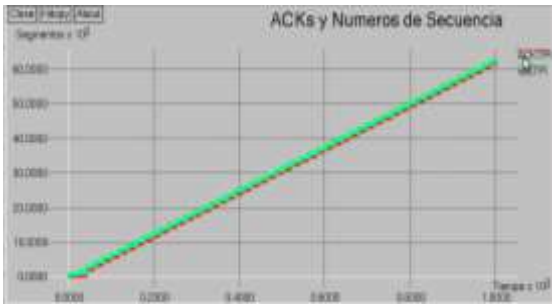


Figura 40.1 Comportamiento de los ACKs y los números de secuencia, para el script TBTP.

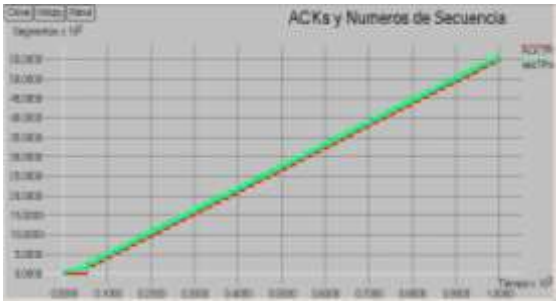


Figura 40.2 Comportamiento de los ACKs y los números de secuencia, para el script 1.5TBTP.

El RTT en el script 1.5TBTP es mayor. Las variaciones que presenta la variable RTTvar al incrementar la ventana son menores que en TBTP indicando que la red no se encuentra tan congestionada, estas pérdidas probablemente ocurren por el canal de transmisión, la gráfica 41 indica el comportamiento anteriormente descrito.

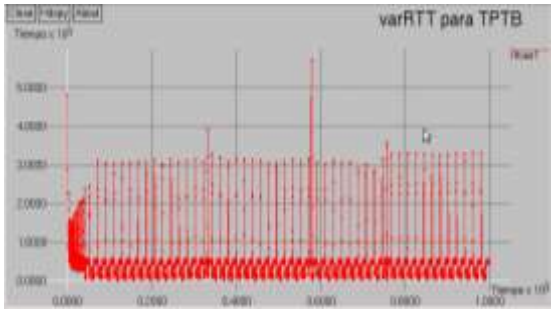


Figura 41.1 Comportamiento del parámetro varRTT con el script TBTP.

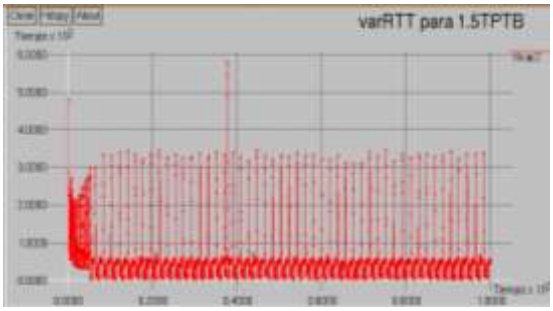


Figura 41.2 Comportamiento del parámetro varRTT con el script 1.5TBTP.

4.3. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO DE LA RED.

El encolamiento indica el nivel de paquetes que están siendo transmitidos por el nodo transmisor. En TBTP el encolamiento inicia un menor tiempo, y el valor de recuperación de la ventana de congestión es más alto que en el caso de 1.5TBTP, lo que resulta en un rendimiento más óptimo como se observa en la gráfica 42.

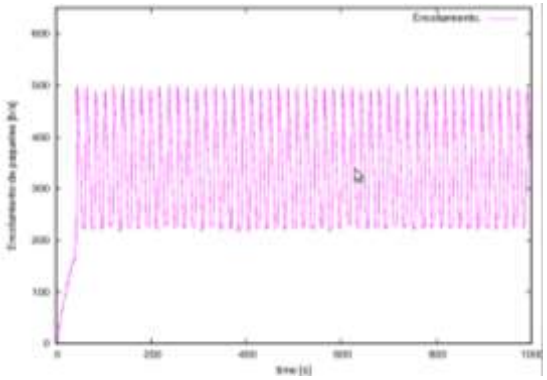


Figura 42.1 Encolamiento de los paquetes en la terminal transmisora en el script TBTP.

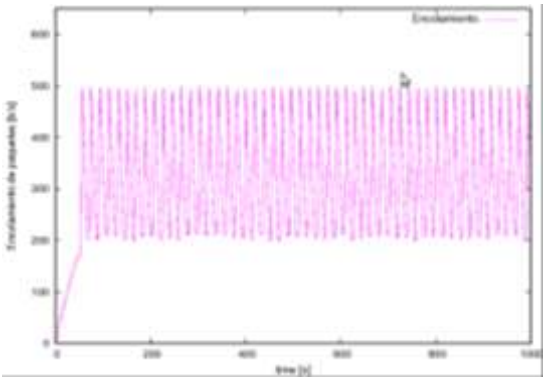


Figura 42.2 Encolamiento de los paquetes en la terminal transmisora en el script 1.5TBTP.

Mediante la figura 43 se indica el retardo en la red para los dos tiempos de planificación de asignación de recursos.

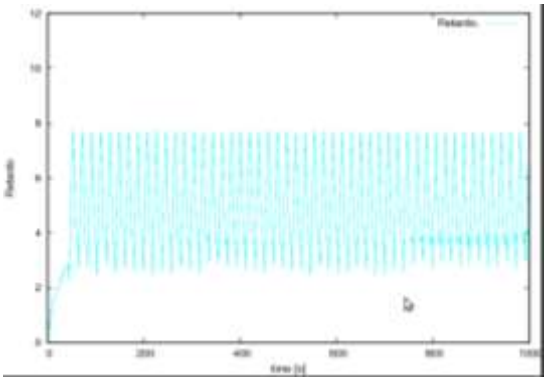


Figura 43.1 Retardo en la red para el script TBTP.

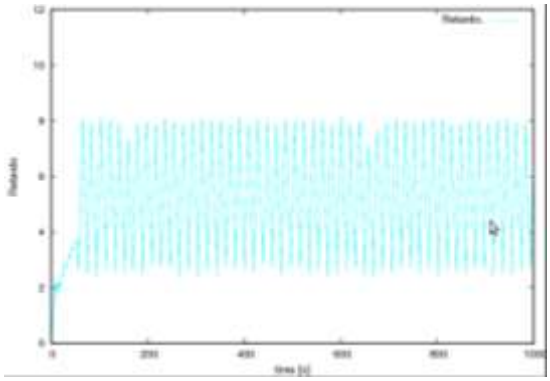


Figura 43.2 Retardo en la red para el script 1.5TBTP.

4.4. DESEMPEÑO DEL *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

Algunos de los parámetros que permiten evaluar el desempeño de la red, son el *throughput* y la utilización del canal; a continuación se indican las gráficas 44 y 45 que visualizan el comportamiento de los scripts en estudio.

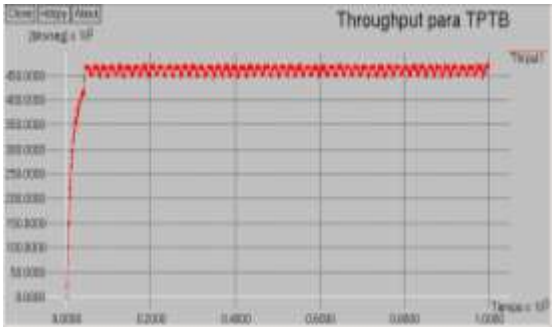


Figura 44.1 *Throughput* total en el script TBTP.

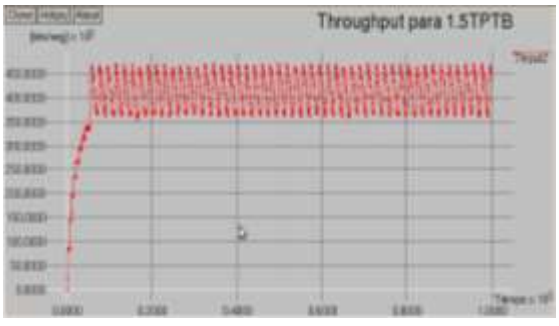
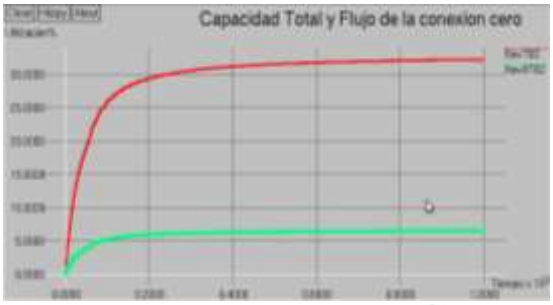


Figura 44.2 *Throughput* total en el script 1.5TBTP.

Capacidad total Conexión 0

Figura 45.1 Utilización total y de la conexión cero para el script TBTP.



Capacidad total Conexión 0

Figura 45.2 Utilización total y de la conexión cero para el script 1.5TBTP.

En el protocolo TCP para un entorno satelital no es tan necesario que se actualice frecuentemente el TBTP debido a los grandes RTTs que se presentan.

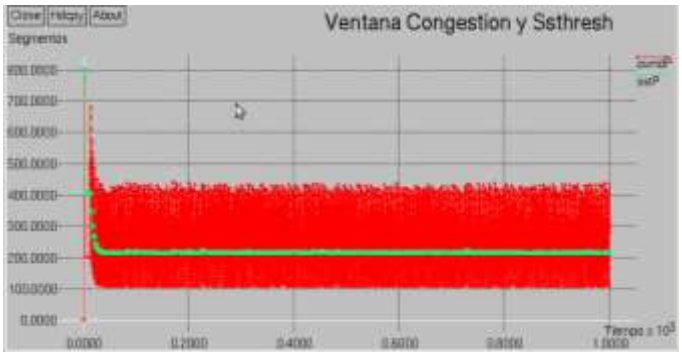
5.0. IMPACTO AL VARIAR EL TIPO DE ASIGNACIÓN DE RECURSOS.

Para esta prueba se escogió un script que presentará un buen desempeño en la red como lo fue Slot2, el único cambio que se hizo para realizar esta prueba fue utilizar otra clase que se encargue de la asignación de recursos en la NCC, la clase se llama Proporcional; a este nuevo script se le llamó Proporcional y se lo compara con el escenario Slot2. Los resultados indican que al utilizar un asignador tipo “Proporcional” se obtiene un mejor aprovechamiento de la capacidad del canal, además de aumentar el rendimiento de todas las conexiones.

En el archivo *event-test1-bod.tr* del escenario Slot2 se observa un gran desperdicio de los slots asignados, además se puede visualizar que su asignación se hace según una planificación de cola tipo FIFO, donde las asignaciones de los slots se realizan según sea el orden de petición a la NCC.

5.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

Aparentemente el ejemplo Slot2 presenta un mejor comportamiento de *cwnd* si se observa la figura 46, sin embargo durante los intervalos donde se encuentra en la fase de recuperación le cuesta mucho más tiempo obtener el mismo nivel de ventana antes de que ocurriera la pérdida**,** su ventana se caracteriza por tener un nivel más alto del *ssthresh*, y la ventana oscila sobre niveles más altos indicando que el tiempo de asignación de recursos es mayor, por lo tanto tarda un mayor tiempo recuperarse de la fase de congestión**.**

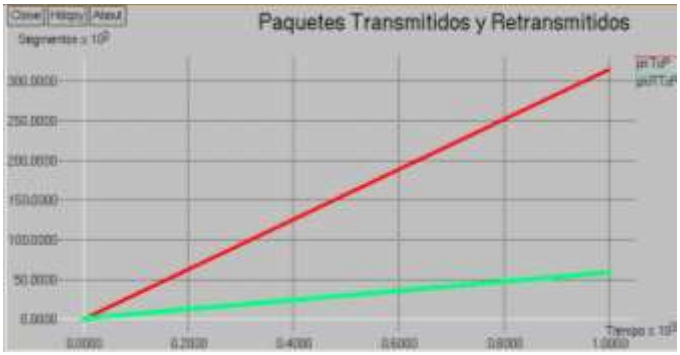


Cwnd ***ssthresh***

Figura 46. Comportamiento de la ventana de congestión y el *ssthresh* en el script Proportional.

La ventana en Proportional, trabaja sobre un nivel de *ssthresh* menor lo que indicaría que un menor número de paquetes son transmitidos, no obstante su ventana presenta un mejor desempeño ya que continuamente la NCC le asigna recursos y de esta manera su ventana no es tan grande, sin embargo se transmite en muchos más intervalos, y no está a la espera de que le asignen recursos para transmitir su información.

Los paquetes transmitidos indican que en Proportional presenta un mejor rendimiento, el número de paquetes retransmitidos es menor ya que hace uso de una ventana menor y además no presenta el inconveniente del retardo de asignación para transmitir sus datos. En las figuras 47 se visualiza el comportamiento de las variables *ndatapack* y *nrexmitpack*, para las dos configuraciones.



Paquetes Tx **Paquetes Retransmitidos**

Figura 47. Paquetes transmitidos y retransmitidos en el script Proportional.

5.2. DATOS OBTENIDOS PARA EL ACK, y RTT.

Al tener una asignación de recursos equitativa, el inicio de la ventana de congestión es más rápido lo que conlleva que más ACKs lleguen al transmisor y *cwnd* este continuamente creciendo. A continuación se indica la figura 48 que describe este procedimiento, para los dos scripts.

Figura 48. Comparación del comportamiento de los ACKs.

El RTT percibido por el escenario Proporcional es inferior al empleado por Slot2 como se observa en la figura 49, el cual es buen indicador de que la ventana de congestión para el escenario Proporcional presenta un mejor desempeño.

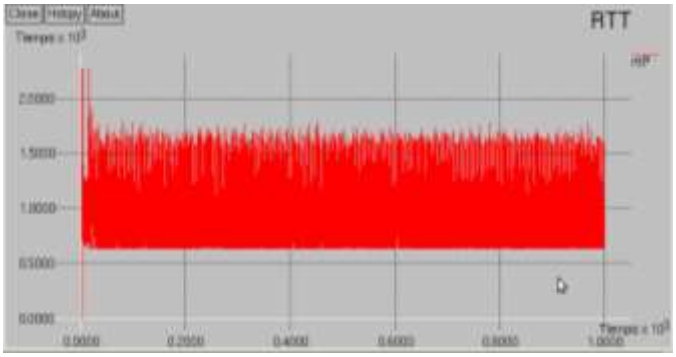


Figura 49. Comportamiento del RTT para el script Proportional.

5.3. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO DE LA RED.

El encolamiento de paquetes en Proportional es bastante inferior que para el caso Slot2, la razón es porque el terminal transmisor no tiene que esperar mucho tiempo para la asignación de recursos para poder transmitir. El encolamiento de paquetes en Slot2 es peor ya que tarda mucho más tiempo en transmitir datos. En la figura 50 se a precia el comportamiento del encolamiento para el script Porportional

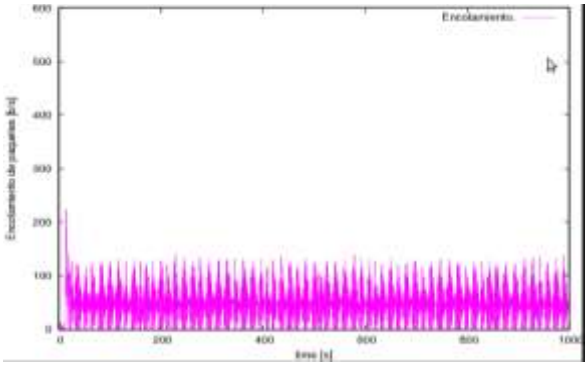


Figura 50. Comportamiento del encolamiento en Proportional.

El retardo es mucho mayor en el escenario Slot2, debido a que requiere un mayor tiempo de espera para transmitir los datos, mientras tanto el tamaño del búfer estará continuamente almacenando recursos. En la figura 51 se observa el comportamiento del retardo en Proportional.

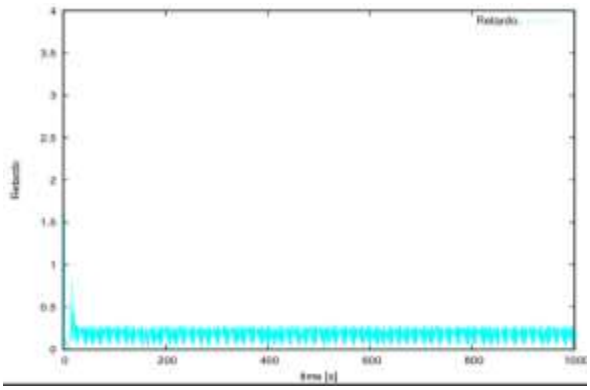
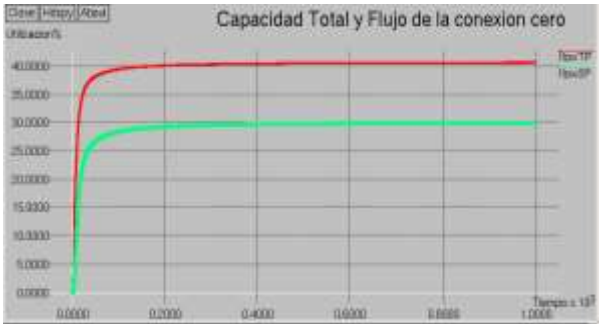


Figura 51. Retardo en la red para el script Proportional.

5.4. RESULTADOS OBTENIDOS PARA LA CAPACIDAD DEL ENLACE UTILIZADO Y LOS SLOTS ASIGNADOS.

A continuación en la figura 52 se muestra la capacidad total del enlace mediante la variable flow0T, y el de la conexión cero para los scripts en evaluación.



Capacidad Total **Conexión Cero**

Figura 52. Utilización total de la conexión y del flujo cero en Proportional.

Un parámetro que permite evaluar el impacto de utilizar una clase diferente de asignación son los slots asignados por cada estación, como se ve en las figuras 53, la capacidad asignada después de los primeros segundos permanece sobre un nivel más alto de asignación en el script Proporcional a diferencia de Slot2.

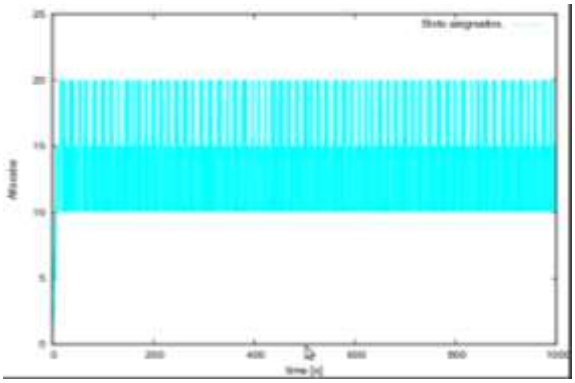


Figura 53.1 Slots asignados por la NCC en Proportional.

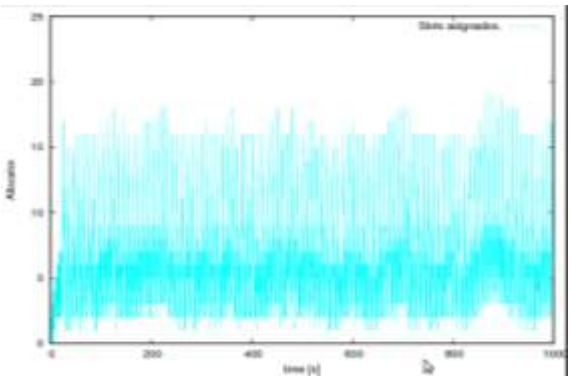


Figura 53.2 Slots asignados por la NCC en Slot2.

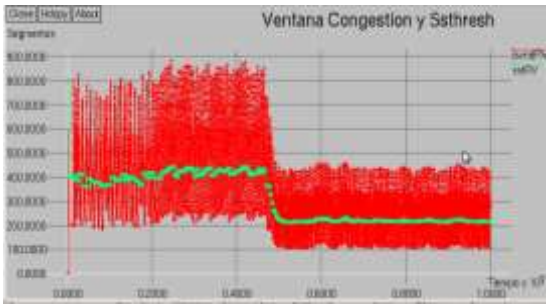
6.0. IMPACTO AL UTILIZAR LOS DIFERENTES TIPOS DE ASIGNACIÓN DE DEMANDA.

En esta prueba se quiere observar cómo puede impactar la forma en que los terminales realizan sus peticiones a la estación de control según las políticas de asignación de demanda que utiliza DBV-RCS, cómo cada una de ellas altera el retardo de asignación y cómo se ve afectado por el retardo en la red, para esta prueba cada terminal utiliza un tipo diferente para la asignación de recursos como se observa en la tabla 3, el script escogido fue *Allocator*, con las modificaciones comentadas.

Tabla 3. Política implementada por cada conexión.

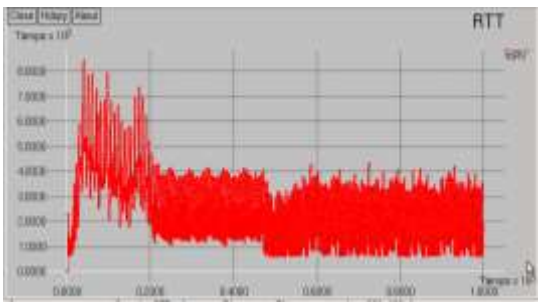
Política	Número de la conexión
RVBDC	1
RBDC	2
CRA	3
VBDC	4
RBDC	5

Para el primer caso se analizó, la asignación de demanda RVBDC la cual se implementó para la conexión cero, esta política realiza peticiones absolutas que varían en el tiempo para poder vaciar la cola del terminal. El término absoluta significa que cada solicitud es remplazada por las solicitudes previas del mismo terminal.



Cwnd ***ssthresh***

Figura 54.1 Comportamiento de la ventana de congestión en la política RVBDC.



RTT

Figura 54.2 RTT medido durante la simulación para la política RVBDC.

Como se observa en la figura 54, el desempeño del RTT está estrechamente relacionado con el comportamiento del tamaño de ventana, las gráficas presentan tres intervalos, en el primer de ellos la ventana presenta una mayor congestión, su ventana no es tan oscilante a pesar de tener un gran volumen de datos transmitiéndose, en el segundo intervalo el RTT se disminuye y esa es la razón que la ventana se recupera más rápido de las pérdidas ocasionadas por el subdimensionamiento del búfer, un factor importante de este buen desempeño es que el número de Slots solicitados y asignados se incrementa como se visualiza en la gráfica 55**,** por lo tanto implica que se transmiten más datos sin preocuparse de la congestión, en el ultimo intervalo los paquetes a transmitir son menores debido a que la asignación de recursos también ha disminuido, sin embargo el RTT baja aún más y esta es la razón porque la ventana es más oscilante que en los dos intervalos anteriores.

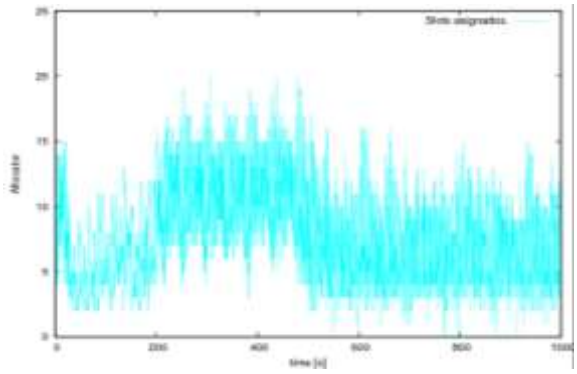


Figura 55.2 Slots solicitados con la política RBVDC a la NCC.

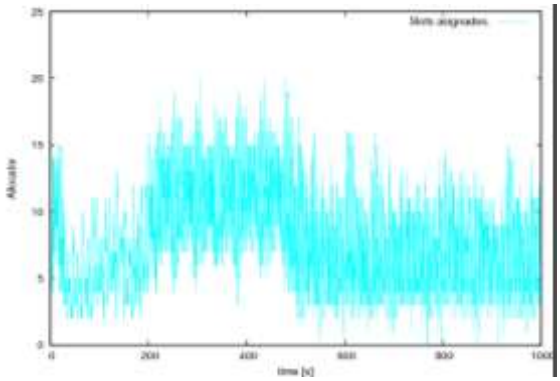


Figura 55.1 Slots asignados al terminal con la política RBVDC.

El encolamiento para esta conexión indica dos intervalos uno donde un mayor encolamiento se produce originado por un mayor volumen de datos transmitidos y un RTT más grande, en el otro las asignaciones disminuyen por lo tanto los paquetes a transmitir también, sin embargo su RTT es menor.

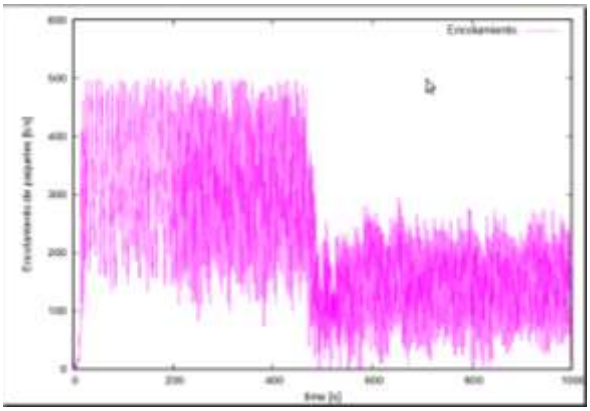
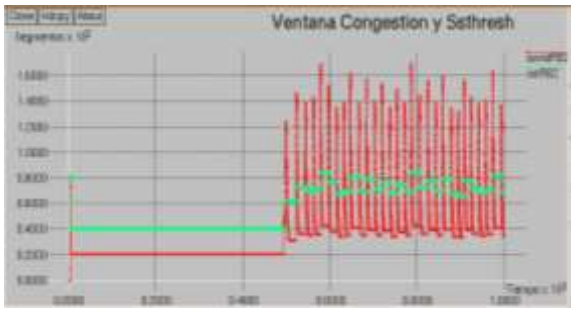


Figura 56. Encolamiento del terminal RCST con la política RVBDC.

Las solicitudes realizadas a la NCC coinciden con el estado de encolamiento en el búfer en la estación, como se ve en la figura 56; un mayor encolamiento requiere un mayor número de slots solicitados para poder vaciar el búfer, en ese intervalo las solicitudes superan el número máximo de slots fijados por la red, de esta manera está solicitando una mayor capacidad de la que el mismo enlace emplea para intentar disminuir el tamaño del búfer. Las solicitudes se

pueden ver como una estimación del próximo valor a utilizar en la ventana de congestión para su conexión [5].

Para el segundo caso se analizó, la asignación de demanda RBDC, el objetivo de esta política es enviar un mensaje de solicitud a la estación de control teniendo en cuenta el tráfico entrante, esta solicitud se negocia al inicio de la conexión y se mantendrá hasta que una nueva sobrescribe la anterior o simplemente cuando ya haya transcurrido un tiempo límite de asignación.

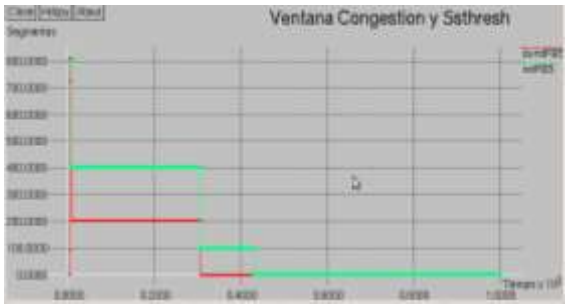


Cwnd ***ssthresh***

Figura 57.1 Ventana de congestión y *ssthresh* para la conexión dos con la política RBDC.

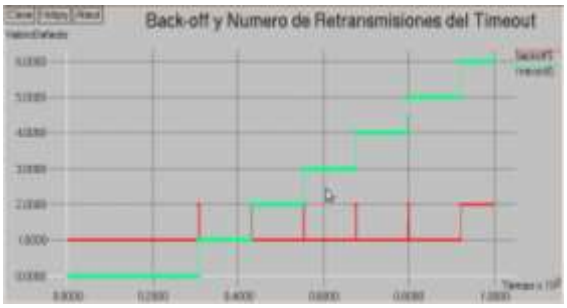
Backoff **nrexmit**

Fig 57.2 Comportamiento del *timeout* en la conexión dos bajo la política RBDC.



Cwnd ***ssthresh***

Figura 58.1 Ventana de congestión y *ssthresh* para la conexión cinco con la política RBDC.



Backoff **nrexmit**

Fig 58.2 Comportamiento del *timeout* en la conexión cinco bajo la política RBDC.

El comportamiento de la ventana de congestión observado para las dos conexiones (dos y cinco), particularmente en la conexión dos indican que su desempeño se ve muy afectado por la manera como se comparten los recursos con otras políticas, la congestión que se presenta no permite que los ACKs lleguen a tiempo ocasionando que se active el *timeout* en el transmisor, tal como se muestra en la figuras 57 y 58. Los slots asignados en el inicio de la conexión son muy pocos de esta manera la ventana no se activa. El incremento del RTT como lo indica la figura 59 es bastante y este es su principal inconveniente para obtener un buen desempeño.

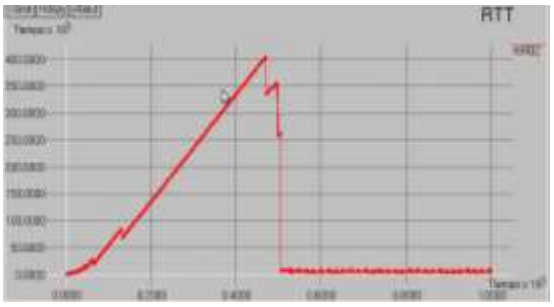


Figura 59.1 Comportamiento del RTT en la conexión dos al emplear la política RBDC.

Figura 59.2 Comportamiento del RTT en la conexión cinco al emplear la política RBDC.

Los paquetes encolados crecen hasta el momento en que se presentan las primeras pérdidas luego permanecen constantes igual que su ventana, cuando logran recuperarse de los *backoffs* su encolamiento se torna oscilante como debería ser. En las figuras 60 se indican el comportamiento del encolamiento con la política RBDC.

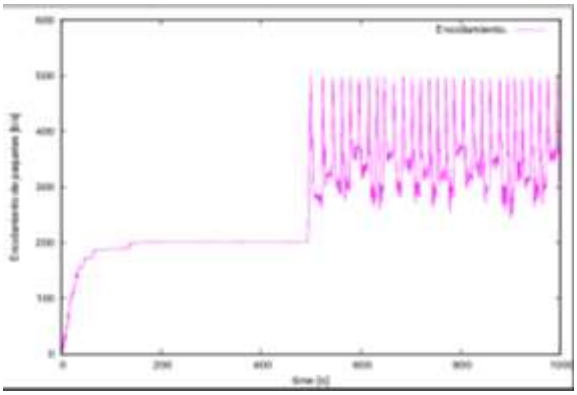


Figura 60.1 Paquetes encolados para la conexión dos al emplear la política RBDC.

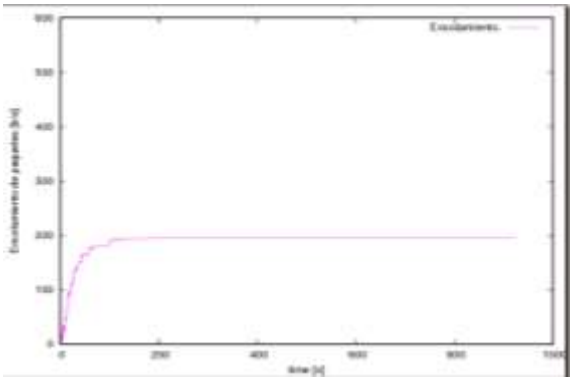
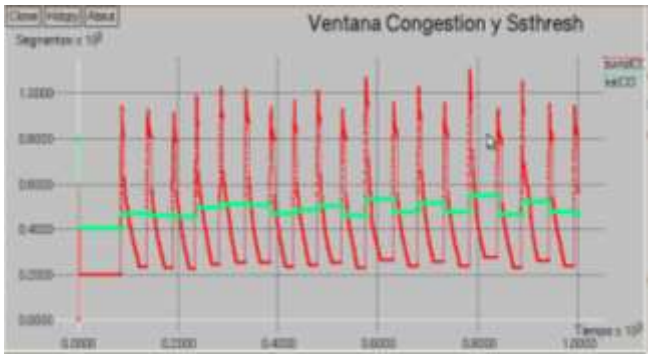


Figura 60.2 Paquetes encolados para la conexión cinco al emplear la política RBDC.

Para el tercer caso se analizó, la asignación de demanda CRA, la cual se utiliza para tipos de tráfico en tiempo real, al momento de ajustar la conexión el terminal solicita un número fijo de slots que en algunas ocasiones no serán utilizados, sin embargo si se conocen las necesidades de la red podrían ser muy útiles; puesto que reducen el retardo de asignación.

Para esta conexión se fijo un único slot, el comportamiento observado en la figura 61 indica que la ventana de congestión tarda mucho tiempo en salir de la fase de recuperación, y la red presenta mucha congestión, debido a ello, cuando la ventana es reducida por causa del sudimensionamiento del búfer, la ventana tarda mucho tiempo en recuperarse, por esta razón su comportamiento es oscilante.



Cwnd ***ssthresh***

Figura 61. Comportamiento de la ventana de congestión y el *ssthresh* con la política Constant.

Con respecto al encolamiento de los paquetes, siempre mantendrá un nivel alto porque se encuentra limitado por el número de slots que solicita a la NCC; de esta manera si el tráfico que utiliza es superior a los datos que entran al búfer, el encolamiento siempre permanecerá en niveles altos, en la figura 62 se describe este comportamiento.

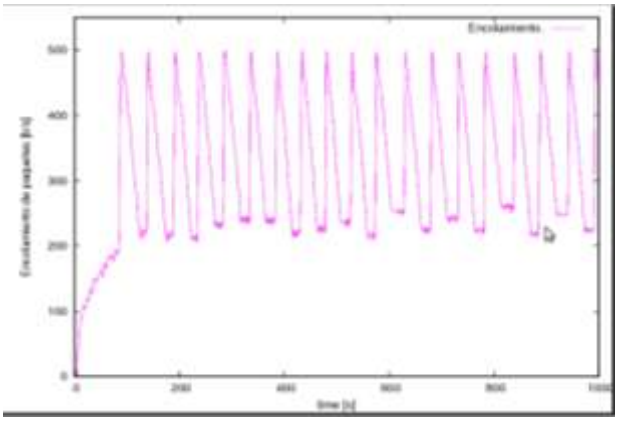
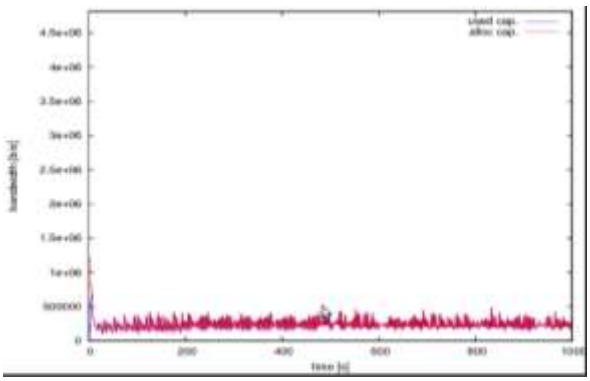


Figura 62. Paquetes encolados en la RCST con la política Constant.

La utilización en el enlace de subida es muy pobre como lo indica la figura 63, debido al único slot asignado por la NCC, sin embargo el tipo de asignación lo ajusta a un valor más adecuado a las condiciones de la red.



Capacidad usada **Capacidad asignada**

Figura 63. Utilización del enlace de subida mediante la política Constant.

Para el cuarto caso se analizó, la asignación por demanda VBDC.

VBDC es la política de asignación por demanda que utiliza por defecto DVB-RCS, sus solicitudes son acumulativas de tal forma que puedan transmitir todos los datos que se encuentran disponibles en la cola MAC, para lograrlo requieren de posteriores solicitudes. En la figura 64 se indica el comportamiento de *cwnd* y *ssthresh* en VBDC.

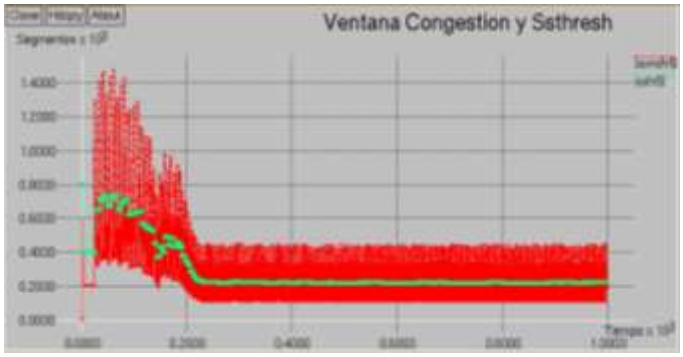


Figura 64. Comportamiento de la ventana de congestión y el *ssthresh* con la política VBDC.

VBDC presenta la mejor utilización del canal al inicio de la conexión, debido a esa gran apertura de la ventana de congestión el tamaño de ventana que alcanza es alto, y en la medida en que las demás conexiones incrementen su tasa de transmisión se disminuye el número de slots asignados y de esta manera su ventana.

El RTT mediante la política VBDC, no se ve tan afectado por la congestión y presenta valores de retardo pequeños como lo indica la gráfica 65, por lo tanto un mayor número de ACKs llegan al transmisor.

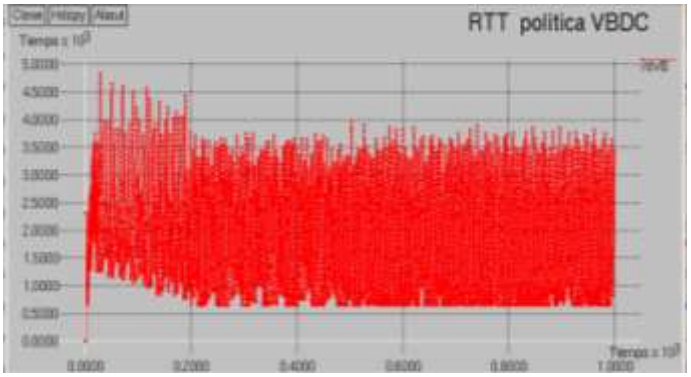


Figura 65. RTT observado en la conexión VBDC.

Al observar la gráfica 66 indica que un mayor número de solicitudes se presentan al inicio de la simulación, así mismo su encolamiento requiere más slots para disminuir la congestión del búfer.

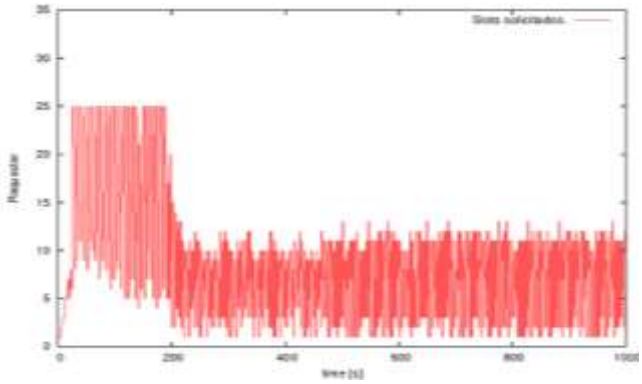


Figura 66. Slots solicitados a la NCC por la política VBDC.

En la figura 67 se gráfica la utilización del canal contra el tiempo de simulación, el desempeño de VBDC y RVBDC obtiene una buena utilización del canal si se compara con la utilización total de la red, no obstante los slots asignados para las políticas restantes están muy por debajo.

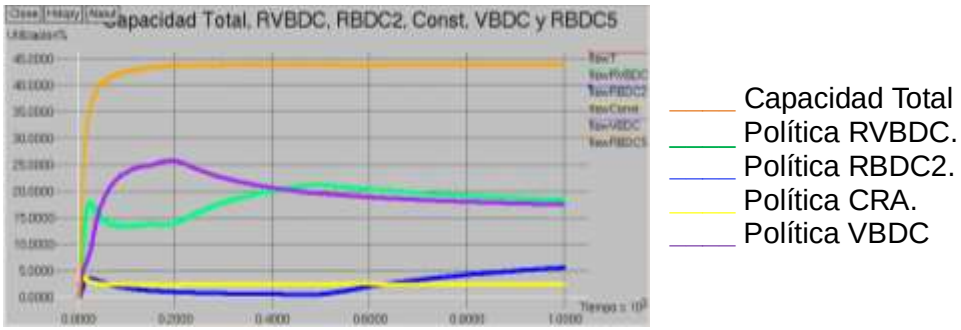


Figura 67. Utilización del canal total y para las diferentes políticas.

BIBLIOGRAFIA

[1] E. Paaby, *Evaluation of TCP Retransmission Delays*, Universidad de Oslo, May. 2006. Documento disponible en: http://heim.ifi.uio.no/~paalh/students/EspenSoegaardPaaby.pdf. Consultado 30 Enero del 2009.

[2] C. Caini y R. Firriencili, *End-to-end TCP enhancements performance on satellite links*, Universidad de Bologna, Italia, 2006. Documento disponible en: http://portal.acm.org/citation.cfm?id=1157991. Consultado el 15 Abril 2009.

[3] B. Baesjou, *TCP synchronisation effect in TCP New Reno and TCP Hybla*, Universidad de Twente, 2004. Documento disponible en: http://dacs.ewi.utwente.nl/assignments/completed/bachelor/reports/B-assignment_Baesjou.pdf. Consultado el 20 Noviembre 2008.

[4] R. Secchi, TDMA-DAMA, página web disponible en: http://ala.isti.cnr.it/wnlab/tdmadama. Consultado el 20 Mayo 2009.

[5] P.Chinin, G. Giambene, C Roseti, M. Luglio, *Dynamic resource allocation based on a TCP-MAC Cross-Layer approach for Interactive Satellite Networks*, Universidad de Roma, Tor Vergata, Italia, 2005. Documento disponible en: http://www.tlcsat.uniroma2.it/wp-content/uploads/2007/04/sat851-24-5-367-385.pdf Consultado el 20 Marzo 2009.

ANEXO D

RESULTADOS OBTENIDOS CON LA VERSIÓN TCP HIGHSPEED.

1. IMPACTO AL VARIAR EL NÚMERO DE SLOTS.

Para determinar cuál es el impacto que produce al variar el número de slots en la versión TCP HighSpeed en un entorno satelital, se utilizaron los escenarios Slot2 y Lenglsl1, la política de asignación que se empleo en la NCC es FODA, y en las 5 conexiones se utilizó la asignación por demanda VBDC.

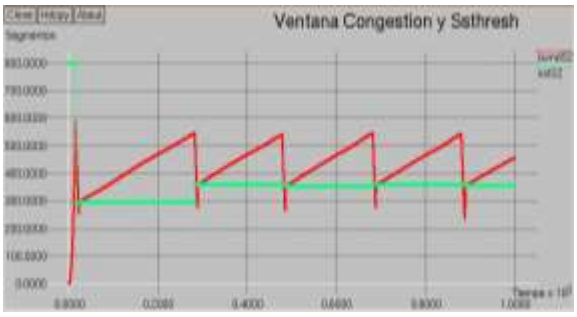
En la tabla 1, se indica las configuraciones realizadas a los scripts de simulación para las diferentes variaciones a los mecanismos de la capa de enlace de datos.

Tabla 1. Parámetros de configuración para los diferentes scripts de simulación.

Script	Lengs2	Lengs1	Slot1	Slot2	Slot3	Slot4
Duración trama	0,04	0,05	0,04	0,05	0,06	0,04
N^0 Slots	14	14	14	20	24	16
Tamaño Slot	1504	1504	184	1504	1504	1504
Ancho de Banda	4.211.200	3.368.960	515.200	4.812.800	4.812.800	4.812.800
Supertrama	25	20	25	20	17	25

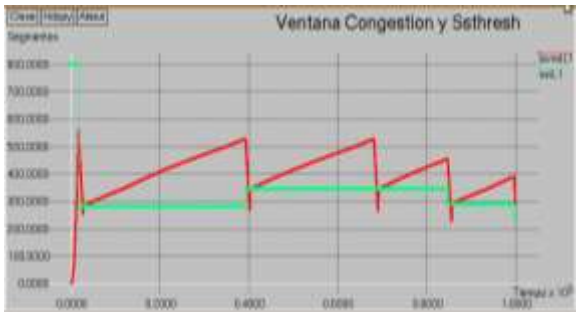
1.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

En TCP HS su ventana de congestión se actualiza según sea el número de ACKs que lleguen al transmisor y se decrementa según sea el número de pérdidas que ocurran en la ventana de congestión, Slot2 indica que un mayor nivel de pendiente se están transmitiendo como lo indica en la figura 1, se puede observar un comportamiento de diente de sierra entre más frecuentemente sea se presentará un tamaño de ventana más grande. En la figura 1.2 se infiere que el nivel del *ssthresh* en el script Lengsl1 es afectado por la congestión en la red.



Cwnd ***ssthresh***

Figura1.1 Ventana de congestión y *ssthresh* en el script Slot2.



Cwnd ***ssthresh***

Figura1.2 Ventana de congestión y *ssthresh* en el script Lenghsl1.

Los paquetes transmitidos demuestran que para la configuración Lengsl1 la variable ndatapack es menor su incremento que en Slot2, evidenciando un peor rendimiento, los paquetes transmitidos se aumentan según sea la llegada de ACKs al transmisor. En la figura 2 se compara el comportamiento de los paquetes transmitidos y retransmitidos por los scripts.

Paquetes transmitidos Slot2
Paquetes transmitidos Lenghsl1

Figura 2.1 Paquetes transmitidos por los scripts Slot2 y Lenghsl1.

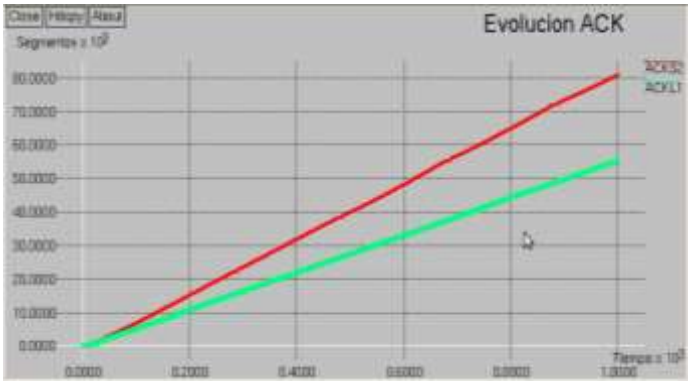
Paquetes retransmitidos Slot2
Paquetes retransmitidos Lenghsl1

Figura 2.2 Paquetes retransmitidos por los scripts Slot2 y Lenghsl1.

Los paquetes retransmitidos son imperceptibles comparados con TCP Hybla, sin embargo estos afectan demasiado el comportamiento de la ventana de congestión; para el caso Slot2 se presentan al inicio de la simulación, y durante ella con una sola pérdida le indica al transmisor que debe disminuir su ventana. En la figura 2 el número de pérdidas es mayor en Slot2 debido a que su apertura de ventana fue más agresiva.

1.1. DATOS OBTENIDOS ACK, RTT Y NÚMERO DE SECUENCIA.

El nivel de incremento en los ACKs evidencia que tan rápido ha salido de la fase de retransmisión rápida para seguir a la fase de evasión de congestión, en la medida que sigan llegando más ACKs se incrementa el valor del número de secuencia. Para Slot2 es notorio un mejor desempeño como se aprecia en la figura 3.



ACKs Slot2 ACKs Lengsl1

Figura 3 Crecimiento de los ACKs para los dos scripts.

El RTT que percibe Lengsl1 es mayor y es esa la principal razón por la cual la red se encuentra congestionada, y por ello la ventana no alcanza un mejor desempeño, como lo muestra en la figura 4.

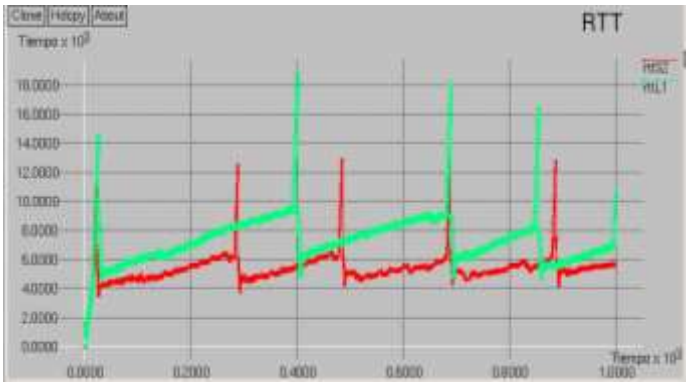


Figura 4 Comportamiento del RTT para Slot2 y Lenghsl1.

1.2. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO EN LA RED.

El tamaño del búfer configurado en las estaciones transmisores se ajusto a 500 paquetes, este valor para la versión de HS-TCP no es muy apropiado al producto por retardo por ancho de banda, sin embargo un valor más alto afecta el RTT del enlace.

La mayoría de las pérdidas que ocurren en TCP HS están relacionadas al encolamiento en el búfer. Cuando el encolamiento en el búfer es grande y llega una ráfaga de datos a la terminal RCST se presenta un sub-dimensionamiento del búfer, en ese instante se aplica la política de gestión de búfer *Drop Tail* para descartar el paquete. Al otro lado en el transmisor, la ventana se reduce porque él considera que la red se encuentra congestionada.

Al incrementar el tamaño del búfer también implica un mayor retardo en la red y por supuesto un mayor RTT, parámetros que inciden en el comportamiento de la ventana. En Slot2 el retardo es inferior que en Lengsl1 como se visualiza en la gráfica 5.

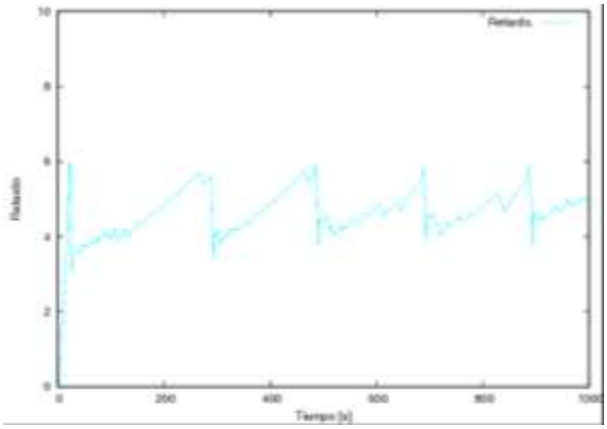


Figura 5.1 Retardo en la red para el script Slot2.

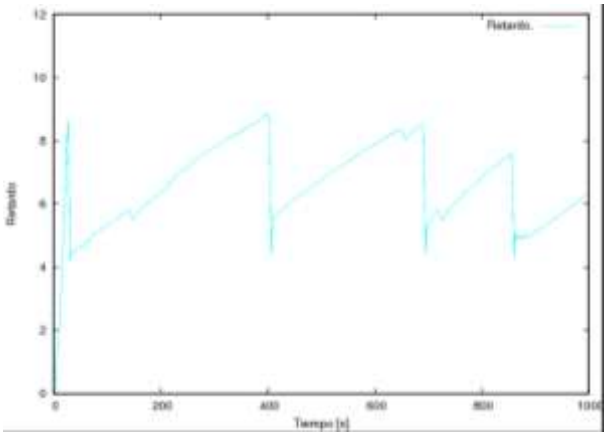


Figura 5.2 Retardo de encolamiento para el script Lenghsl1.

Cuando el transmisor decrementa su ventana los paquetes que se encuentran en vuelo llegan al receptor y este envía los ACKs para que la ventana se incremente, este proceso ocurre mucho más rápido en Slot2 lo que le permite recuperarse en un menor tiempo de la pérdida.

Cuando el encolamiento es menor que el máximo tamaño del búfer, la pérdida se le atribuye a la congestión o errores de transmisión en el canal satelital.

Según lo observado en las simulaciones realizadas para esta versión de TCP, un mayor número de slots obtiene un mejor rendimiento en la red, el número de paquetes de datos transmitidos y el número de secuencia aumenta más rápidamente**.** El encolamiento en los paquetes para Slot2 es mejor ya que presenta un mayor nivel de pendiente, lo que significa que más paquetes se tienen que transmitir por lo tanto los slots solicitados se incrementan y hay un mayor crecimiento de la ventana de congestión, sin embargo el número de paquetes retransmitidos también aumentan. En la figura 6 se indica el comportamiento del encolamiento de los paquetes para las dos configuraciones.

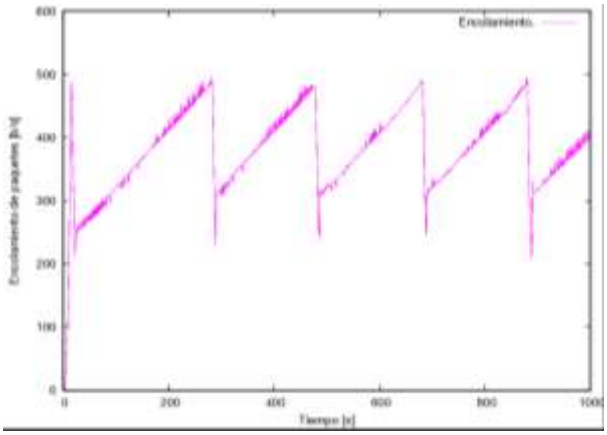


Figura 6.1 Encolamiento de paquetes en Slot2.

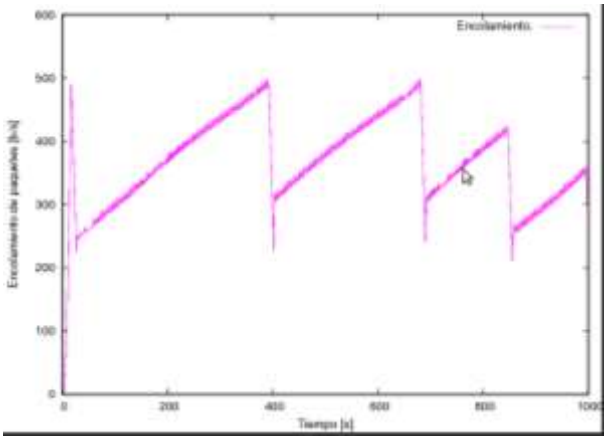


Figura 6.2 Encolamiento de paquetes en Lenghsl1.

1.3. *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

El *throughput* de la red en Lengsl1 es inferior y permanece en un valor constante; para Slot2 el rendimiento se presenta en forma de diente de sierra que indica los intervalos donde las pérdidas afectan al rendimiento total de la red. En la figura 7 se muestra el *throughput* para los dos scripts.

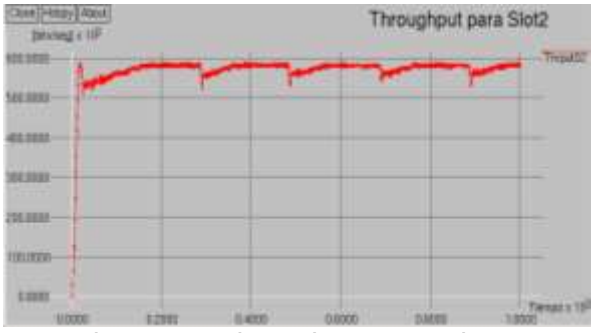


Figura 7.1 *Throughput* para Slot2.

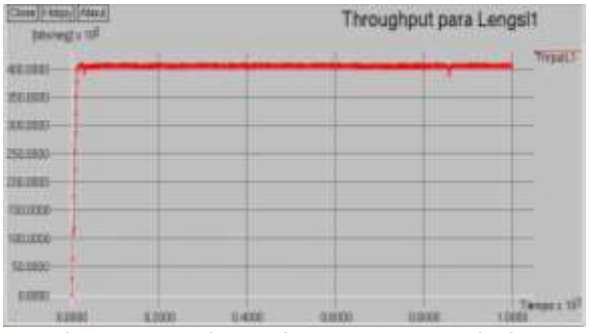


Figura 7.2 *Throughput* para Lenghsl1.

A continuación se muestra la figura 8 que indica la capacidad del canal de subida por los scripts Slot2 y Lengsl1, donde se aprecia un mayor porcentaje de red utilizado en Slot2 dado que el número de slots asignados por la NCC es mayor, lo que indica que un mayor número de paquetes han sido transmitidos.

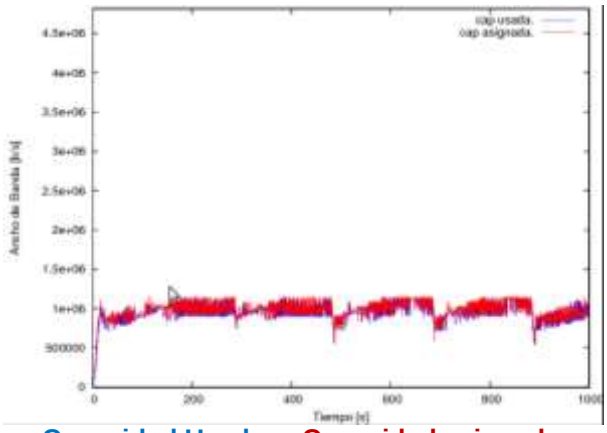


Figura 8.1 Utilización del ancho de banda *uplink* por el Slot2.

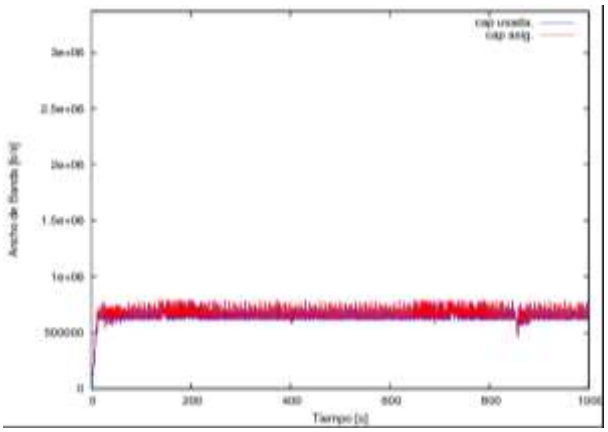


Figura 8.2 Utilización del ancho de banda *uplink* por Lenghsl1.

2.0. IMPACTO AL VARIAR EL TAMAÑO DEL SLOT.

Para esta prueba se utilizó los mismos scripts que en TCP Hybla con la diferencia de que todas las estaciones transmisoras utilizan la política de asignación de demanda VBDC.

2.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

Las ventanas de congestión en los dos scripts, se ven limitados cuando se encuentran en la fase de inicio lento causado por el tamaño del búfer de su estación transmisora, este efecto se resalta con claridad en la figura 9, sin embargo el tener un ancho de banda más grande permite recuperarse con mayor eficiencia del evento de una pérdida, tal como se observa en la figura 10.

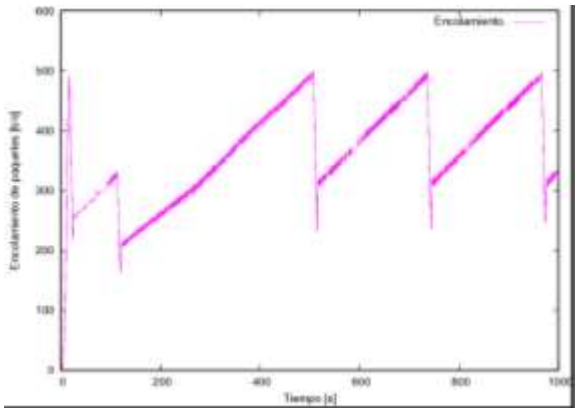


Figura 9.1 Encolamiento de paquetes en Lenghsl2.

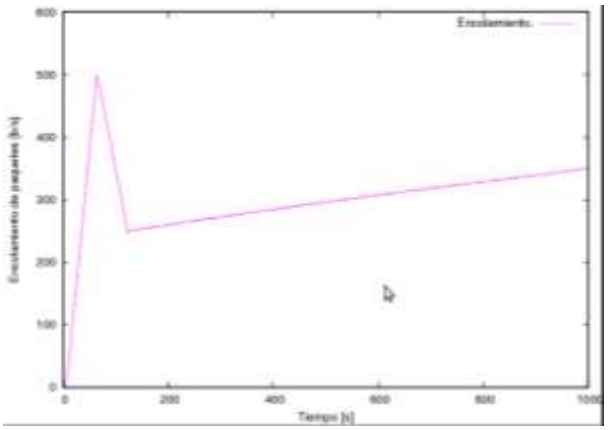
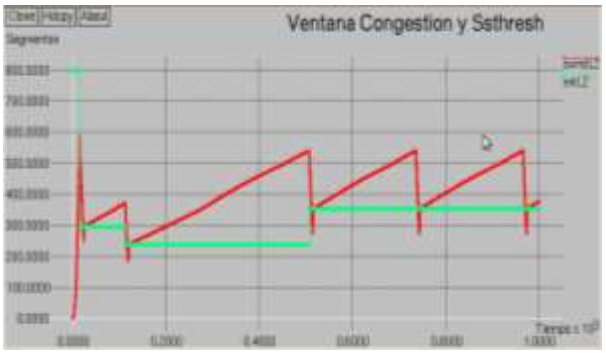


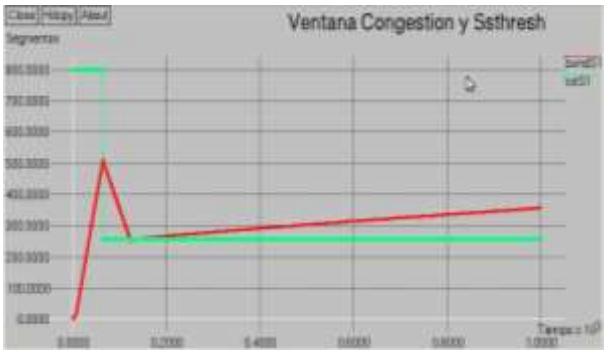
Figura 9.2 Encolamiento de paquetes en Slot1.

El script Slot1 indica el pobre desempeño de la ventana cuando esta inicia su fase de evasión de congestión, el retardo de propagación es tan grande que la ventana tiene que optar por una política de incremento muy conservadora.



Cwnd ***ssthresh***

Figura 10.1 Comportamiento *cwnd* y *ssthresh* para el script Lenghsl2.



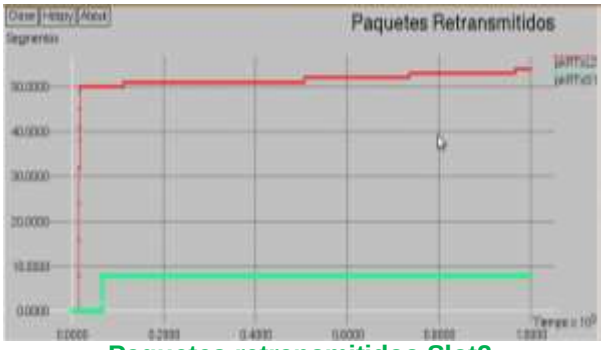
Cwnd ***ssthresh***

Figura 10.2 Comportamiento *cwnd* y *ssthresh* para el script Slot1.

Los paquetes transmitidos en el momento que ocurre una pérdida presenta una menor pendiente indicando que está en la fase de recuperación y retransmisión rápida; este comportamiento se aprecia en la figura 11. Entre mayor sea el número de paquetes transmitidos implica que la ventana presenta un buen desempeño como sucede para el script Lengsl2.

Paquetes transmitidos Slot2
Paquetes transmitidos Lenghsl2

Figura 11.1 Comparación del número de paquetes transmitidos por los dos scripts.



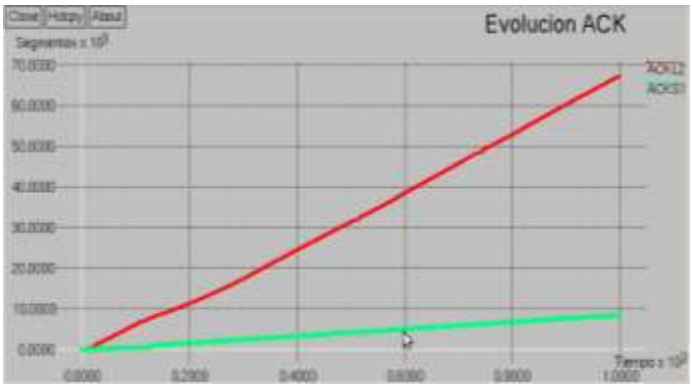
Paquetes retransmitidos Slot2
Paquetes retransmitidos Lenghsl2

Figura 11.2 Comparación del número de paquetes retransmitidos por los dos scripts.

Con respecto a los paquetes retransmitidos los resultados demuestran que al tener un tamaño de slot más pequeño y un menor ancho de banda, su ventana en la fase de inicio lento es conservadora y por esta razón las pérdidas de segmentos son menores. TCP HS en presencia de menor congestión implementa una política más agresiva, lo que también conlleva a que muchos más datos se pierdan.

2.2. DATOS OBTENIDOS DE ACK Y RTT.

El número de ACKs en Slot1 permite observar cuales son los mecanismos de congestión que la red está presentando; al principio la ventana se encuentra en inicio lento donde el incremento del número de los ACKs es mayor, para el intervalo en que ocurre una pérdida este se encuentra en recuperación rápida, donde los ACKs disminuyen y luego pasa a evasión de congestión donde los ACKs se incrementan de manera constante. En la figura 12 se observa un mayor número de ACKs para el caso Lengsl2 que en Slot1.



ACKs Slot2 ACKs Lengsl2

Figura 12 Crecimiento de los ACKs para los dos scripts.

El RTT es la principal razón porque en el script Slot1 presenta un mal desempeño, los RTTs grandes le dan a entender al transmisor que la red se encuentra muy congestionada que debe optar por una política de crecimiento más conservadora. A continuación se muestra la gráfica 13 con el comportamiento del RTT para los dos scripts en comparación.

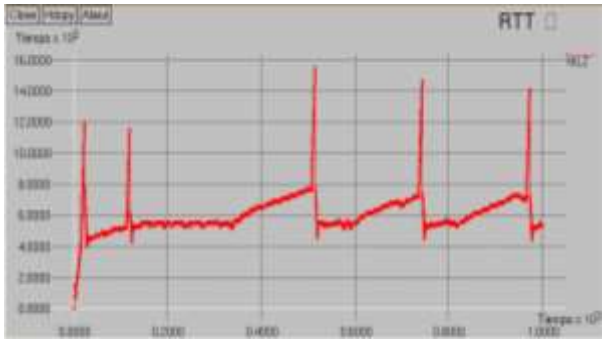


Figura 13.1 Comportamiento del RTT en Lenghsl2.

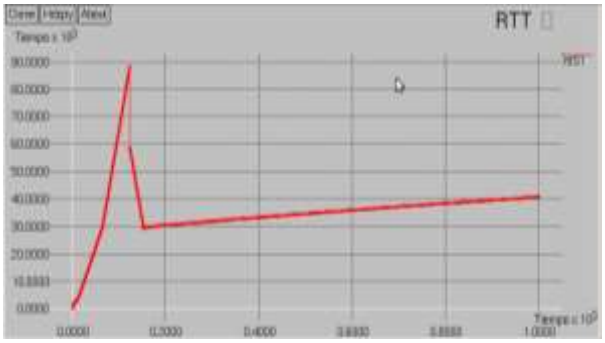


Figura 13.2 Comportamiento del RTT en Slot1.

2.3. ANÁLISIS DEL ENCOLAMIENTO Y RETARDO EN LA RED.

El encolamiento de paquetes en Lengsl2 tiene un nivel de pendiente mayor lo que deduce que más paquetes están siendo transmitidos y un mayor número de slots son solicitados a la NCC; y en consecuencia no se ven tan afectado por la congestión, como se observa en la figura 9.

Un tamaño de slot pequeño significa un menor número de paquetes por transmitir, donde las estaciones solicitadoras tienen que esperar un tiempo adicional para poder transmitir, en ese tiempo de espera el retardo se incrementa como se visualiza en la figura 14.

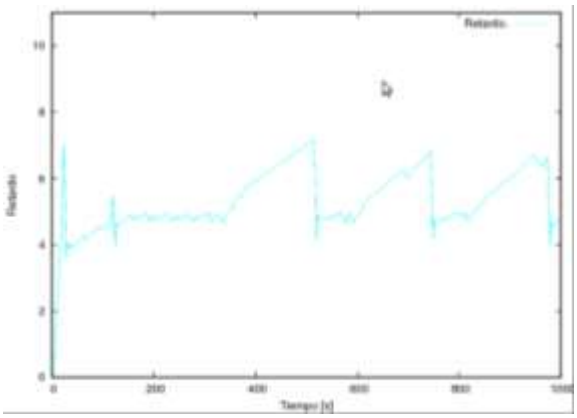


Figura 14.1 Retardo en la red para el script Lenghsl2.

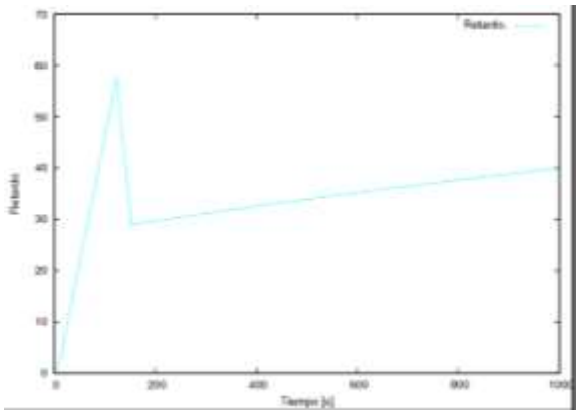


Figura 14.2 Retardo en la red para el script Slot1.

2.4. *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

Los resultados arrojados para medir el impacto al variar el tamaño del slot, indican que un mayor tamaño del slot mejora notablemente el desempeño de la red como lo indican las figuras 15 y 16. La utilización del canal para el caso Slot1 presenta un bajo porcentaje, como consecuencia del comportamiento de su ventana.

Figura 15.1 ***Throughput*** para Lenghsl2.

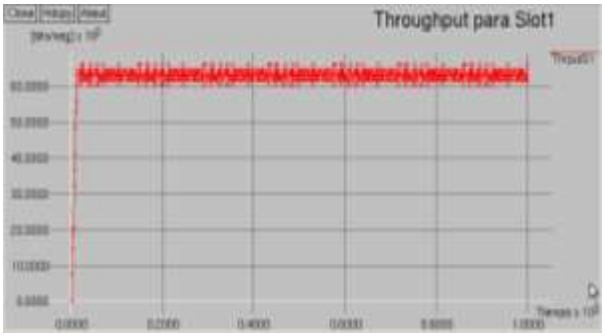
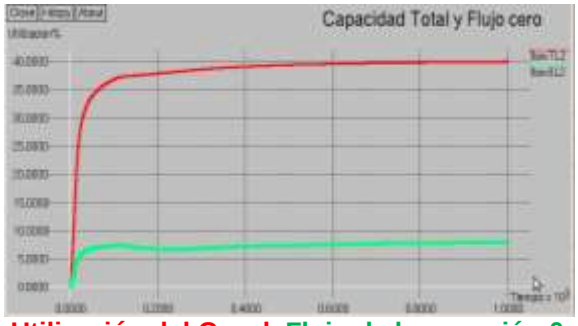
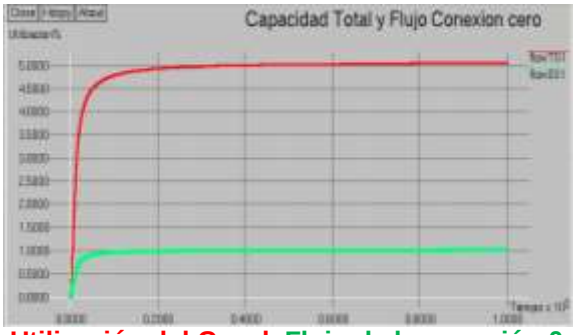


Figura 15.2 ***Throughput*** para Slot1.



Utilización del Canal **Flujo de la conexión 0**

Figura 16.1 Capacidad total y flujo de la conexión cero en Lenghsl2.



Utilización del Canal **Flujo de la conexión 0**

Figura 16.2 Capacidad total y flujo de la conexión cero en Slot1.

3.0. IMPACTO AL VARIAR LA DURACIÓN DE LA TRAMA.

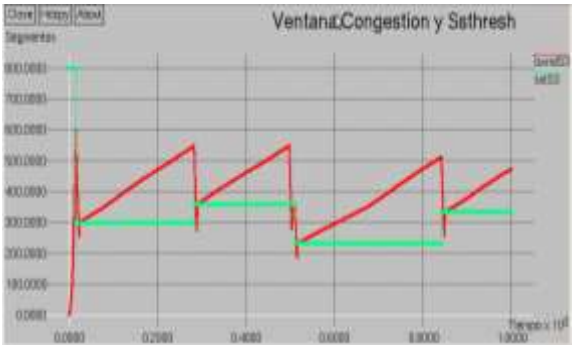
Los scripts seleccionados fueron Slot4 y Slot3, la configuración de los datos se indica en la tabla 1.

3.1. ANÁLISIS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

Al observar la gráfica 17 los scripts Slot4 y Slot3 alcanzan un comportamiento apropiado en la ventana; las pérdidas que ocurren en su gran mayoría son causados por el sub-dimensionamiento en el tamaño del búfer y no por la congestión de la red. Slot4 se encuentra levemente menos congestionado con un nivel de pendiente mayor lo que indica que más paquetes están siendo transmitidos. En ambos scripts la red se encuentra congestionada de igual forma, en la medida que el transmisor reciba mayor número de ACKs este implementa una política más agresiva en su ventana con un mayor nivel de pendiente. El nivel del *ssthresh* en las dos conexiones es un buen indicador del comportamiento de la ventana.

Cwnd ***ssthresh***

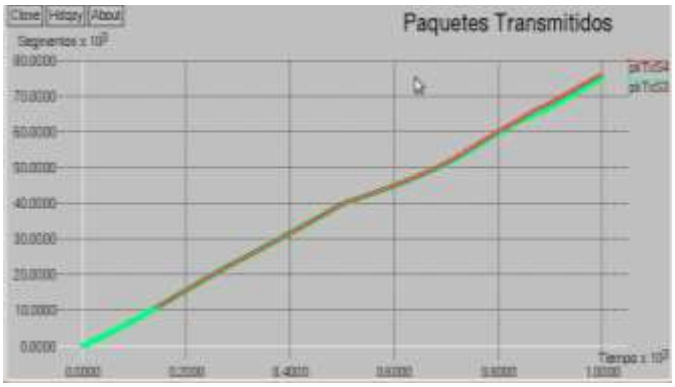
Figura17.1 Ventana de congestión y *ssthresh* en el script Slot4.



Cwnd ***ssthresh***

Figura17.2 Ventana de congestión y *ssthresh* en el script Slot3.

Los paquetes transmitidos en ambos scripts son muy similares como lo indica la figura 18, sin embargo se presenta un mayor número de paquetes transmitidos en Slot4; de esta manera el rendimiento y la utilización del canal también obtienen valores muy parecidos.



Paquetes transmitidos Slot4 y Slot3

Figura 18 Comparación de los paquetes transmitidos en los dos scripts.

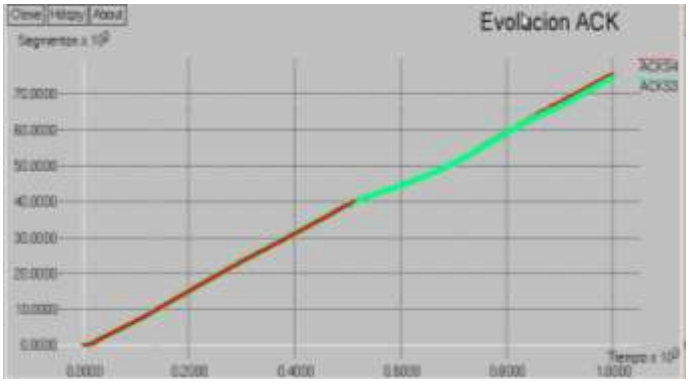
Una menor congestión al inicio de la transmisión conlleva a que el transmisor envíe mas paquetes de datos, de esta manera una ráfaga de mayor tamaño llega al búfer y el mecanismo de gestión de búfer se encarga de descartar muchos más paquetes, en la figura 19 se visualiza este comportamiento para el script Slot4.

Paquetes RTx Slot4 **Paquetes RTx Slot3**

Figura 19 Comparación de los paquetes retransmitidos en los dos scripts.

3.2. DATOS OBTENIDOS DE ACK, RTT Y NÚMERO DE SECUENCIA.

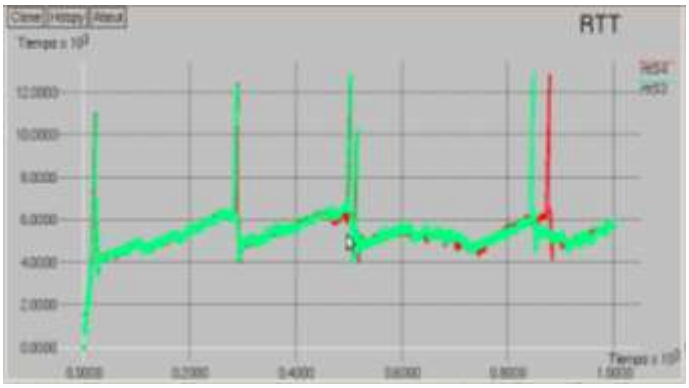
Los ACKs continuamente muestran al transmisor que política de incremento aditivo se debe implementar en este, mientras que los paquetes perdidos indican su valor a decrementar, en la figura 20 se deduce que Slot4 presenta un mejor desempeño con respecto a los ACKs recibidos.



ACKs Slot4 **ACKs Slot3**

Figura 20 Crecimiento de los ACKs para los dos scripts.

En ambos scripts es claro que tienen problemas de congestión debido al incremento del RTT; en el instante cuando ocurre una pérdida su RTT se incrementa y se estabiliza nuevamente después de unos segundos. A continuación en la figura 21 se indica el comportamiento del RTT para los dos valores de duración de trama.



RTT Slot4 **RTT Slot3**

Figura 21 Comportamiento del RTT para los scripts Slot4 y Slot3.

3.3. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO DE LA RED.

El encolamiento del script Slot4 y Slot3 son muy similares, la diferencia está en que en Slot4 implementa un incremento aditivo menos conservador que permite en un pequeño intervalo alcanzar un mayor número de paquetes transmitidos y luego verse afectado por la pérdida ocasionada por el sub-dimensionamiento del búfer, lo anterior también indica que la red en Slot4 se encuentra en un menor grado de congestión. A continuación en las figuras 22 y 23 se visualiza el comportamiento del encolamiento y su relación con los slots asignados.

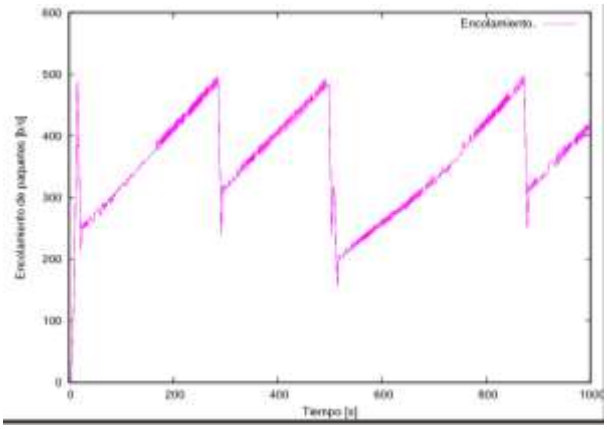


Figura 22.1 **Encolamiento** en el terminal transmisor para Slot4.

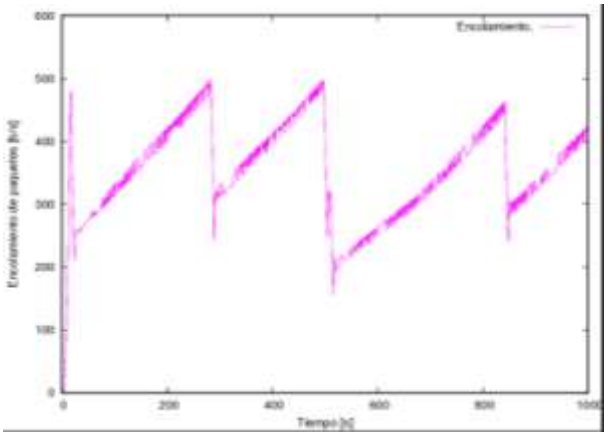


Figura 22.2 **Encolamiento** en el terminal transmisor para Slot3.

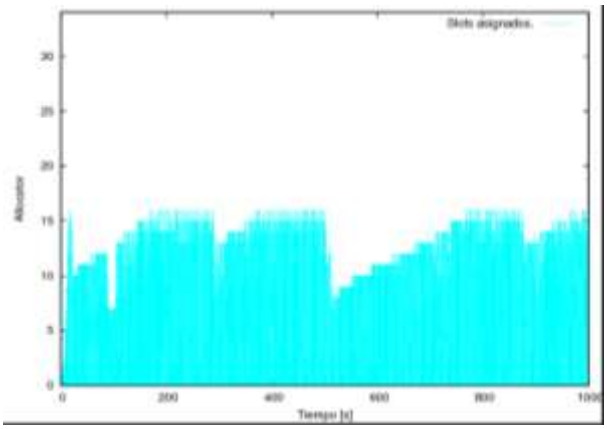


Figura 23.1 **Slots asignados** por la NCC para Slot4.

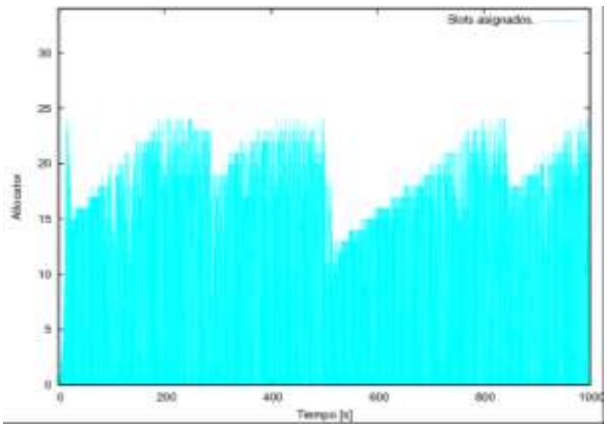


Figura 23.2 **Slots asignados** por la NCC para Slot3.

Si el retardo en la red es muy pequeño indica que los slots solicitados a la NCC se demoran menos tiempo para transmitir los recursos, un mayor retardo se presenta por la congestión en la red o simplemente por una mala asignación de recursos por la NCC. En la gráfica 24 se indica el comportamiento del retardo para los dos scripts.

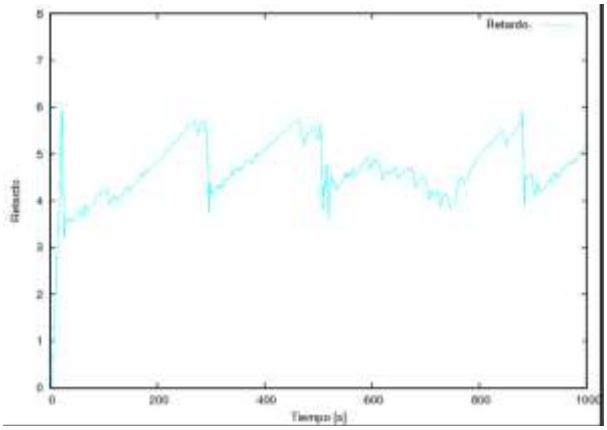


Figura 24.1 **Retardo** en la red con el Slot4.

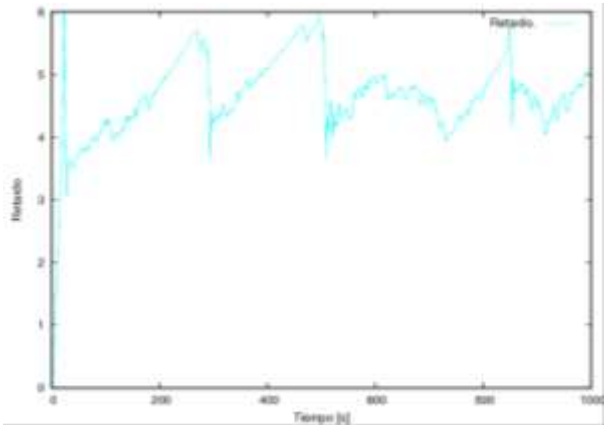


Figura 24.2 **Retardo** en la red con el Slot3.

3.4. *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

El rendimiento total alcanzado por la red y la capacidad del enlace utilizado por todas las conexiones son muy similares, incluso algunas conexiones presentan mejor desempeño al utilizar un mayor tiempo de trama, sin embargo el *throughput* total para todas las conexiones demuestra que una duración de trama menor presenta un mejor rendimiento, así mismo para la utilización del canal, como se muestra en la figuras 25 y 26.

Figura 25.1 ***Throughput*** total en el script Slot4.

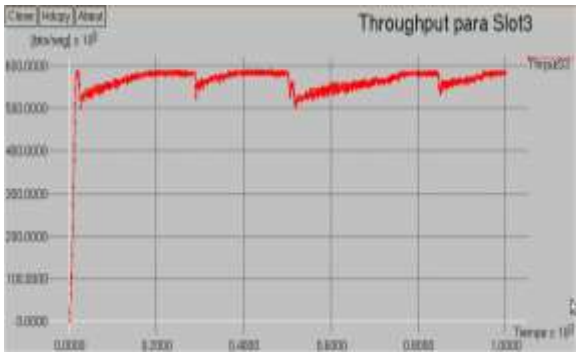
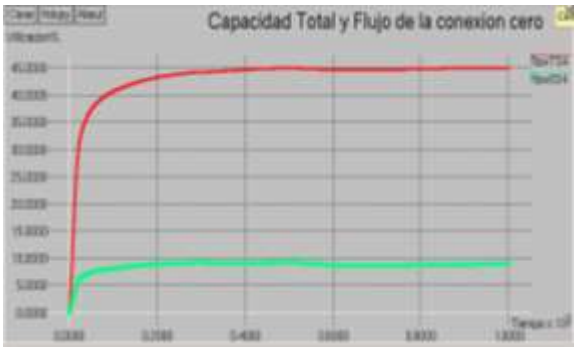
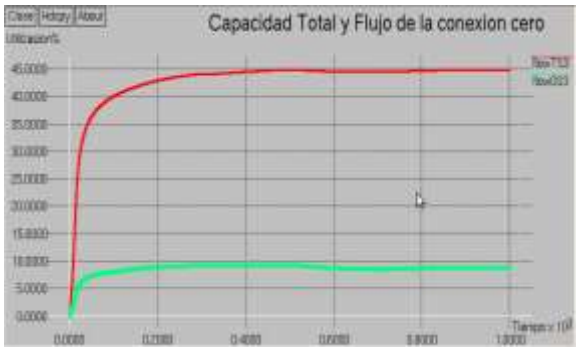


Figura 25.2 ***Throughput*** total en el script Slot3.



Utilización del Canal **Flujo de la conexión 0**

Figura 26.1 Utilización total y del flujo cero en Slot4.



Utilización del Canal **Flujo de la conexión 0**

Figura 26.2 Utilización total y del flujo cero en Slot3.

El *throughput* obtenido por el escenario Slot4 es muy similar al del escenario Slot3 no obstante para muchos más intervalos es superior con un menor tiempo de trama. Al hacer una comparación con cada una de las conexiones se observó que debido a que todos los flujos están continuamente en competencia por los recursos, algunas conexiones se ven más favorecidas que otras en un script en particular.

4.0. IMPACTO AL VARIAR EL TIEMPO DE REPROGRAMACIÓN TPTB.

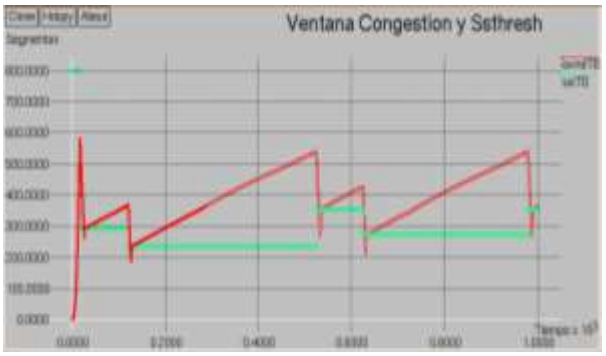
La siguiente prueba se realizó para medir el impacto que presenta variar la duración de la trama en el desempeño del protocolo TCP HS, para esta prueba se configuran los escenarios como se indica en la tabla 2. Los resultados indican que no se afecta mucho el variar el tiempo de planificación de recursos, los tiempos a analizar son 0.8 y 1.0 segundos.

Tabla 2. Parámetros de configuración en la prueba TPTB.

Script	TPTB	1.5TPTb
Duración trama	0,025	0,025
Número Slots	24	24
Tamaño Slot	534	534
Ancho de Banda	4.101.120	4.101.120
Supertrama	32	44

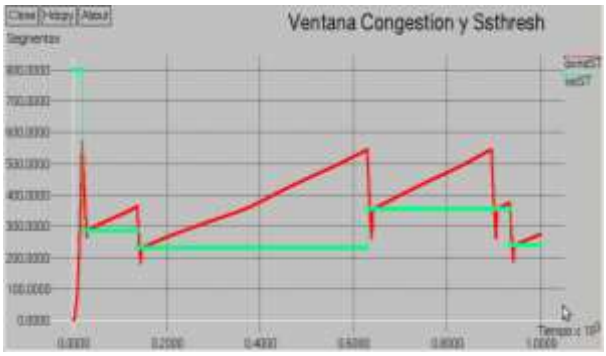
4.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

Las dos ventanas de congestión presentan comportamientos muy similares, sus inconvenientes están en que la congestión y las pérdidas que ocurren por el sub-dimensionamiento del búfer hacen que la ventana pase a la fase de evasión de congestión de manera muy conservada. El *ssthresh* se configura según sea el estado de la ventana, si se presenta alguna pérdida por congestión el valor del *ssthresh* se disminuye como se observa en la figura 27.



Cwnd ***ssthresh***

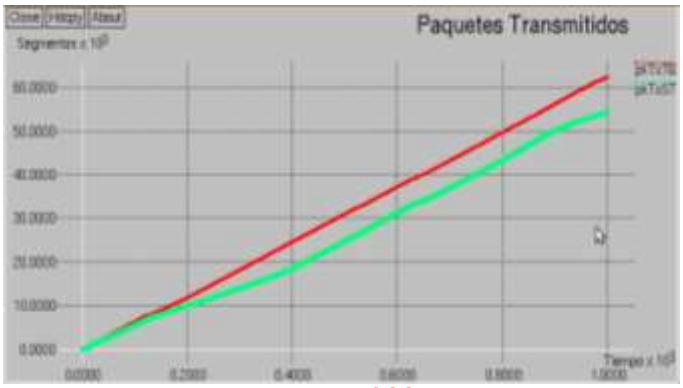
Figura 27.1 Comportamiento de la ventana de congestión y *ssthresh* en el script TPTB.



Cwnd ***ssthresh***

Figura 27.2 Comportamiento de la ventana de congestión y *ssthresh* en el script STPTB.

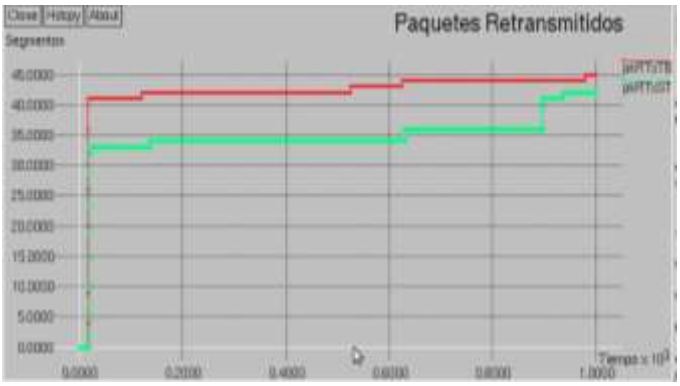
Mediante el comportamiento de los paquetes transmitidos se puede afirmar que el script TPTB manifiesta un mejor rendimiento, si se observa la figura 28 se demuestra que STPTB no puede transmitir la misma cantidad de datos que TPTB.



Paquetes transmitidos TBTP
Paquetes transmitidos STBTP

Figura 28 Paquetes transmitidos en los scripts TPTB y STPTB.

Los paquetes retransmitidos indican que en la fase de inicio lento el script TPTB presenta un mayor número de pérdidas, por lo tanto la ventana de congestión se encuentra menos congestionada que en STPTB, evidenciando un menor RTT como retardo. En la figura 29 se indica el impacto al variar el tiempo TPTB mediante el parámetro *nrexmitpack*.

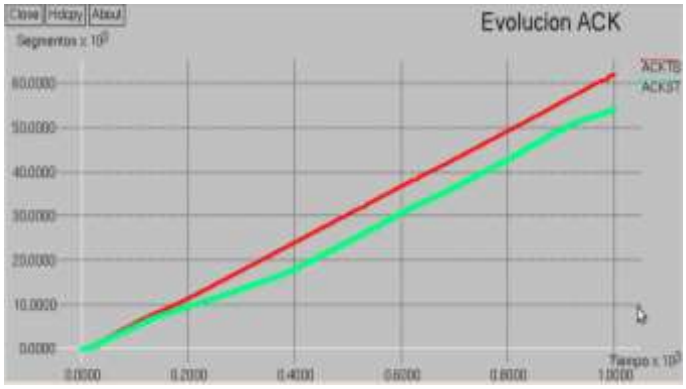


TBTP **STBTP**

Figura 29. Paquetes retransmitidos en los scripts TPTB y STPTB.

4.1.1. DATOS OBTENIDOS DEL ACK Y RTT.

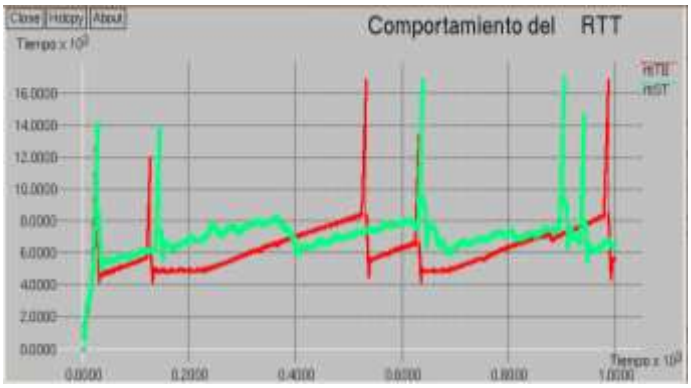
El número de ACKs que fueron recibidos por el nodo transmisor deduce que *cwnd* para el caso TPTB se ve menos afectado por la congestión y obtiene un mejor *throughput* en la red, la figura 30 muestra el comportamiento para este parámetro en los dos scripts**.**



ACKs TBTP **ACKs STBTP**

Figura 30 Comportamiento de los ACKs para los scripts TPTB y STPTB.

Para que TPTB haya presentado un mejor desempeño de ventana es porque su RTT fue menor, este beneficio le permite recibir un mayor número de reconocimientos en el transmisor, lo que implica que más paquetes han sido transmitidos. La figura 31 visualiza el comportamiento del RTT para las dos scripts de simulación.



RTT TBTP **RTT STBTP**

Figura 31 Desempeño del RTT para los dos scripts en comparación.

4.1.2. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO DE LA RED.

Si se observa la gráfica 27 en el instante después que ocurre la pérdida, la ventana de congestión entra a retransmisión rápida e inicia su incremento en inicio lento hasta que *cwnd* es mayor que el *ssthresh*, cuando entra en evasión de congestión la ventana se incrementa lentamente y el encolamiento del búfer indica que la velocidad de entrada de paquetes es mucho mayor a la que sale.
Las pequeñas oscilaciones que se observan durante el aumento y la diminución de los paquetes encolados representan un mayor volumen de datos que salen del búfer que el número de paquetes que entran. En la gráfica 32 se muestra el comportamiento del encolamiento en las RCSTs.

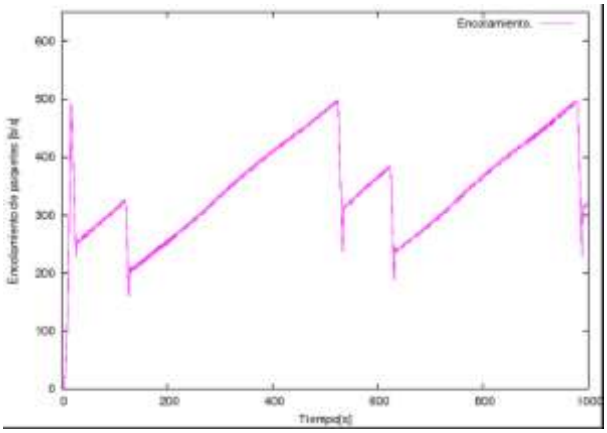


Figura 32.1 **Encolamiento** de los paquetes en la terminal transmisora en el script TPTB.

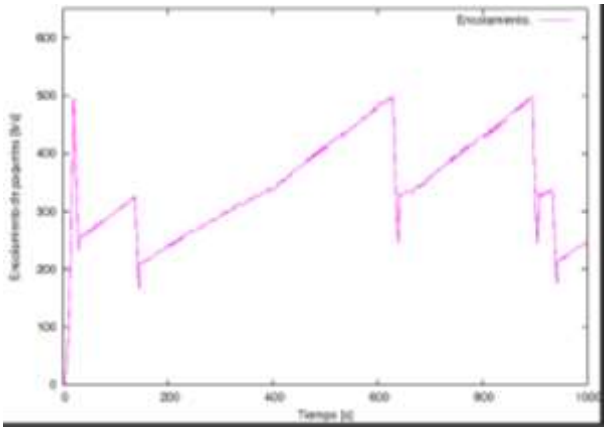


Figura 32.2 **Encolamiento** de los paquetes en la terminal transmisora en el script STPTB.

El retardo en el script TPTB, indica que los paquetes no pierden mucho tiempo en ser recibidos por su sumidero, tal y como se manifiesta en la gráfica 33.

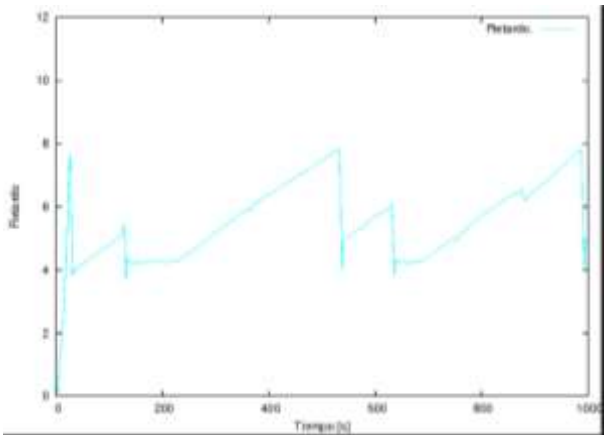


Figura 33.1 **Retardo** en la red para el script TPTB.

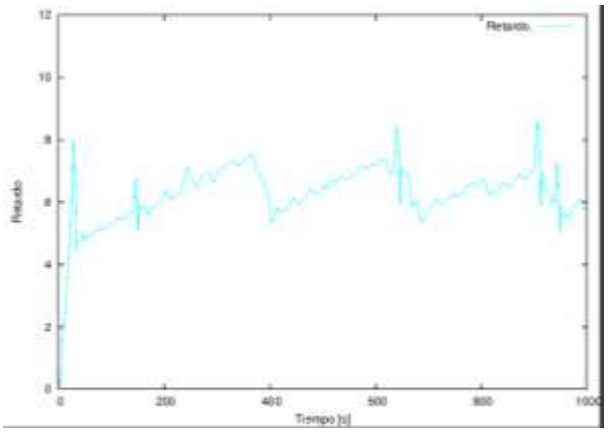


Figura 33.2 **Retardo** en la red para el script STPTB.

4.1.3. *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

El rendimiento para el caso STPTB indica el impacto que genera utilizar un mayor tiempo de planificación de asignación de recursos. A continuación se muestra las figuras 34 y 35 el *throughput* y porcentaje de utilización del canal para los tiempos 0.8 y 1.5 segundos.

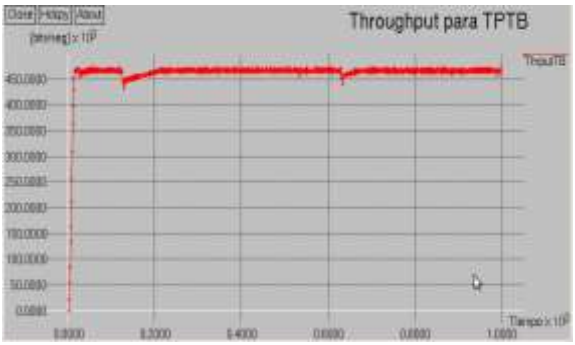


Figura 34.1 ***Throughput*** total en el script TPTB.

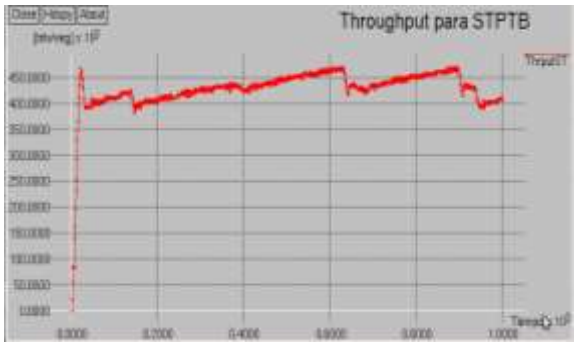
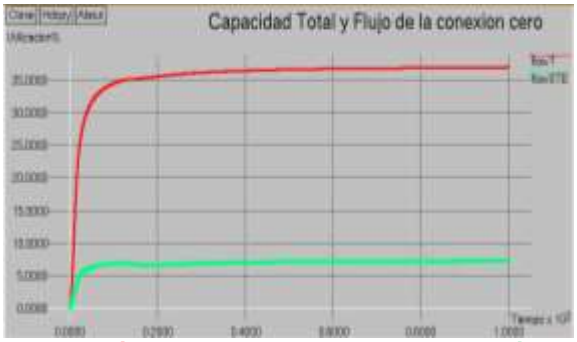
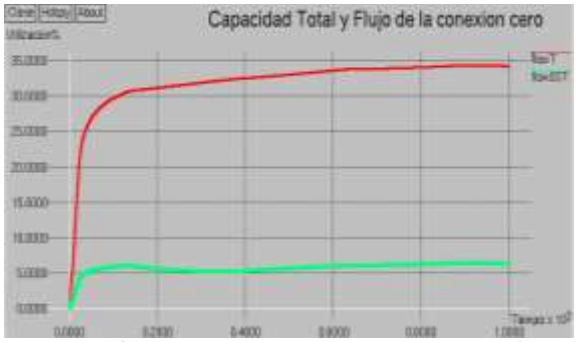


Figura 34.2 ***Throughput*** total en el script STPTB.



Utilización del Canal **Flujo de la conexión 0**

Figura 35.1 Utilización total y de la conexión cero para el script TPTB.



Utilización del Canal **Flujo de la conexión 0**

Figura 35.2 Utilización total y de la conexión cero para el script STPTB.

5.0. IMPACTO AL VARIAR EL TIPO DE ASIGNACIÓN DE RECURSOS.

Con esta prueba se observa cómo afecta el comportamiento de TCP HS, al cambiar el tipo de asignación de recursos en la NCC. El escenario que se seleccionó para esta prueba fue Slot3 (con FODA), el cual se comparará con el uso del algoritmo de asignación de recursos Proportional.

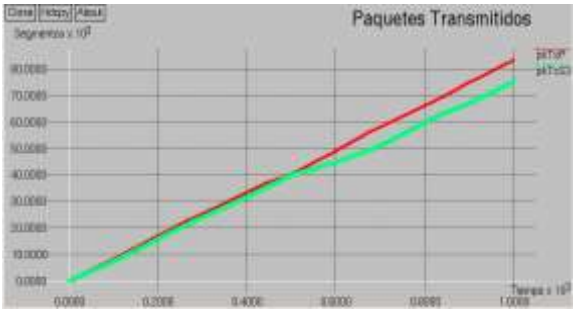
5.1. OBTENCIÓN DE RESULTADOS DE LA VENTANA DE CONGESTIÓN, *SSTHRESH*, PAQUETES TRANSMITIDOS Y RETRANSMITIDOS:

Al observar la figura 36, la ventana de congestión y el nivel del *ssthresh* infieren que al hacer uso de una política de asignación más proporcional se presentan mejores resultados. La ventana en el script Proportional tiene un inicio más rápido en el incremento de la ventana de congestión, y de esta manera aprovecha apropiadamente los recursos de la red, la ventana evidencia que no se ve tan afectado por la congestión, ya que los slots asignados permanentemente están transmitiendo datos sin monopolizar la conexión.

Cwnd ***ssthresh***

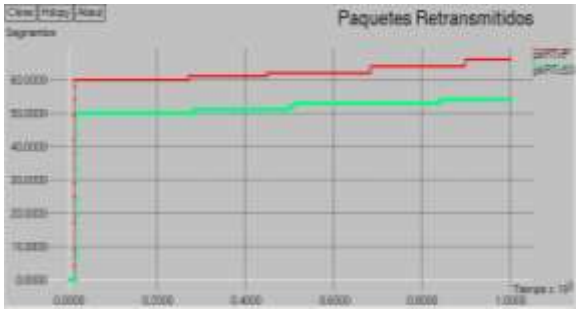
Figura 36 Comportamiento de la ventana de congestión y el *ssthresh* en el script Proportional.

Los paquetes transmitidos indican un mejor comportamiento cuando se hace uso de la política Proportional demostrando así que un mayor volumen de datos están siendo transmitidos por el transmisor, tal como se muestra en la figura 37. Los paquetes retransmitidos al inicio de la simulación representan un gran porcentaje de las perdidas del canal teniendo en cuenta que estas se producen durante la fase de inicio rápido donde su apertura es mayor.



Paquetes transmitidos Proportional
Paquetes transmitidos FODA

Figura 37.1 Comparación de los paquetes transmitidos por los dos scripts en evaluación.

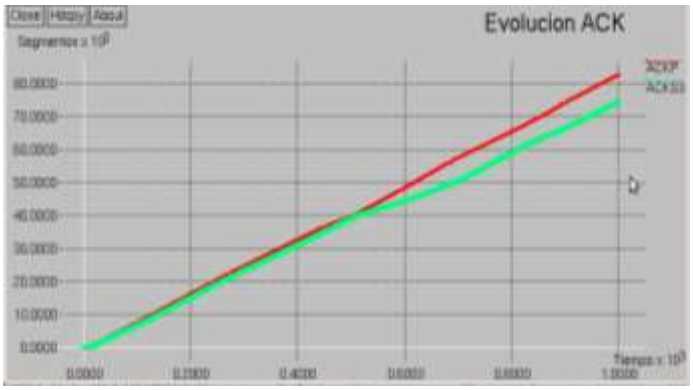


Paquetes retransmitidos Proportional
Paquetes retransmitidos FODA

Figura 37.2 Paquetes retransmitidos por el script Proportional y Slot3.

5.2. DATOS OBTENIDOS DEL ACK Y RTT.

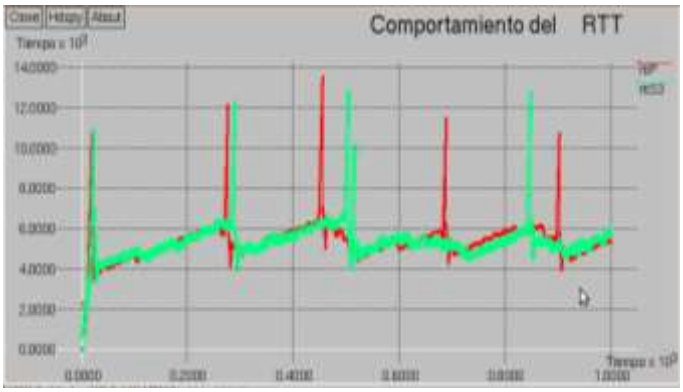
El incremento de ACKs para el caso Proporcional se realiza de una forma más lineal, con variaciones en su nivel de pendiente que evidencian los intervalos donde la congestión en la red se disminuye, en la figura 36 presenta un intervalo donde permanece mucho tiempo en evasión de congestión, cuando la red se entera que la congestión a disminuido (gracias a que arriban más ACKs al transmisor) este aplica una política más agresiva para aprovechar los recursos en la red, en la figura 38 se aprecia lo anteriormente nombrado.



ACKs Proportional **ACKs FODA**

Figura 38. Comparación del comportamiento de los ACKs.

El RTT y el retardo son parámetros muy similares en los dos casos de simulación, debido a ello el mayor impacto que presenta la red está en la manera como la NCC esta asignando recursos a la estación para transmitir. En la figura 39 se muestra el comportamiento del RTT.



RTT Proportional **RTT FODA**

Figura 39. RTT para el script Proportional y Slot3.

5.3. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO DE LA RED.

Para el escenario Slot3-Proportional, se analiza el archivo *curral.tr* el cual indica como ha sido la asignación de los slots para cada una de las estaciones teniendo en cuenta las solicitudes hechas a la NCC, se observa que al implementar el tipo de asignación Proportional se implementa un criterio más equitativo para todas las conexiones, es decir los 24 slots son repartidos a todas las conexiones en cada duración de trama de manera uniforme, y de esta manera evita que cualquier estación monopolice la capacidad de asignación, caso contrario se

observa al implementar la política FODA, además para algunos intervalos esta asignación es cero, lo que implicaría un mayor encolamiento de los paquetes, y un aumento en el retardo de los paquetes, cosas que no ocurren debido a que FODA implementa una asignación considerando el orden en que las solicitudes son realizadas, de esta manera cuando llega el turno de una estación esta hace uso de un gran porcentaje de los slots asignados, y por lo tanto su desempeño permanece constante, este tipo de asignación tiene un gran efecto según sea el tipo de tráfico que se esté utilizando, sin embargo su simplicidad a la hora de su implementación lo hacen muy útil. En la figura 40, se indica el comportamiento del encolamiento y el rango de oscilación del retardo para el script Proportional.

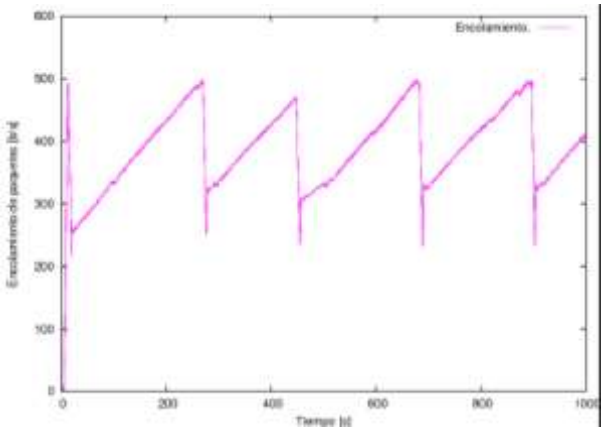


Figura 40.1 Paquetes encolados en la terminal RCST al utilizar la política de asignación Proportional.

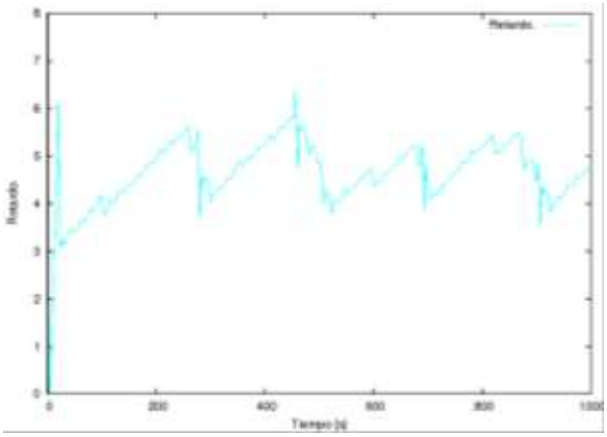
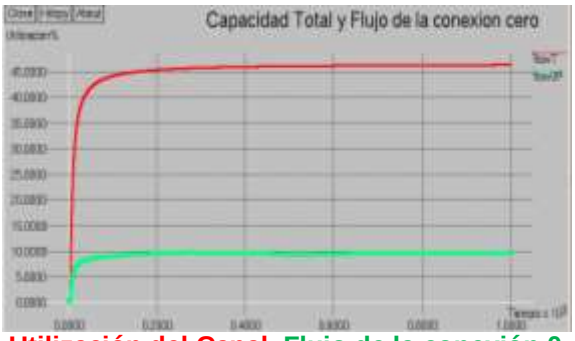


Figura 40.2 Retardo en la red al utilizar la política de asignación Proportional.

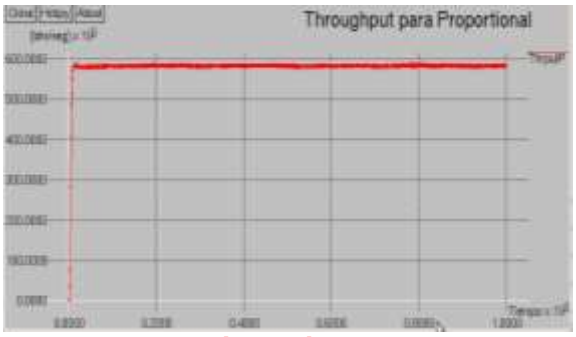
5.4. *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

Por último se muestra, la figura 41 indicando la utilización del canal y el *throughput* respectivamente para el script Proportional. En los primeros segundos la política Proportional hace una mejor utilización del canal y por lo tanto se estabiliza más rápido sobre el máximo valor.



Utilización del Canal **Flujo de la conexión 0**

Figura 41.1 Utilización total de la conexión y del flujo cero en la política Proportional.



Throughput

Figura 41.2 *Throughput* obtenido con la política Proportional.

6.0. IMPACTO AL UTILIZAR LOS DIFERENTES TIPOS DE ASIGNACIÓN DE DEMANDA.

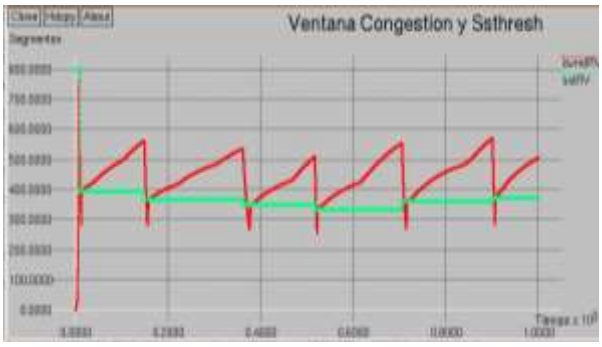
En esta prueba se quiere observar como impacta la manera en que los terminales realizan sus peticiones a la estación de control según las políticas de asignación de demanda que utiliza DBV-RCS, para TCP HS. El escenario escogido fue Slot4 con el tipo de asignación Proportional, a cada conexión se le configura un tipo diferente de asignación de demanda como se muestra en la tabla 3, a excepción del caso RBDC.

Política asignada en la RCST	Número de la conexión
RVBDC	1
RBDC	2
Constant	3
VBDC	4
RBDC	5

Tabla 3. Política implementada en cada conexión.

6.1. IMPACTO DE LA POLÍTICA RVBDC.

Para el primer caso se analizó, la asignación por demanda RVBDC.



Cwnd ***ssthresh***

Figura 42.1 Comportamiento de la ventana de congestión en la política RVBDC.

Figura 42.2 Paquetes retransmitidos mediante la política RVBDC.

La ventana de congestión en la figura 42.1 indica que en la fase de evasión de congestión el transmisor utiliza una política más conservadora y en la medida que llegan más ACKs el nivel de pendiente aumenta, la mayoría de sus retransmisiones ocurren por el sub-dimensionamiento del búfer como lo indica la figura 43, cuando esto ocurre el terminal RCST envía un mensaje ICMP al transmisor informándole que el búfer de este nodo se ha desbordado, por lo tanto el transmisor deja de transmitir y está a la espera de los ACKs para incrementar su ventana nuevamente.

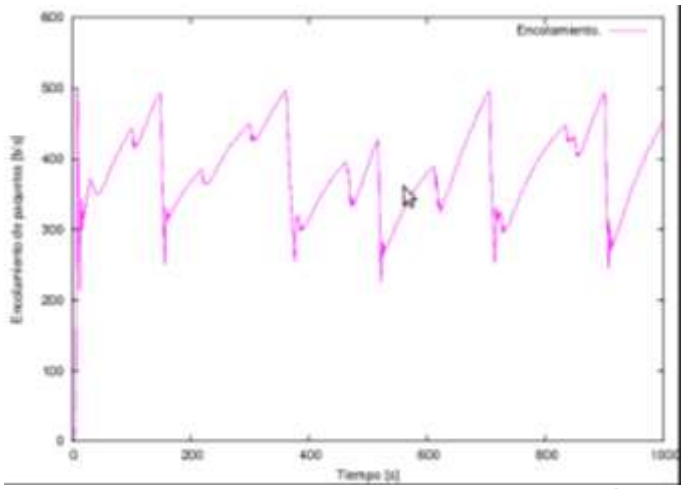


Figura 43 Comportamiento del encolamiento con la política RVBDC.

Cuando la conexión inicia la ventana presenta una apertura muy grande y por esta razón el número de paquetes a retransmitir aumenta considerablemente, como lo indica la figura 42.2. El número de paquetes retransmitidos es alto donde su comportamiento evidencia las condiciones en que se encuentra la red. El RTT de la conexión es muy bueno, en el momento que presenta mayor congestión su valor se ve incrementado.

El retardo presenta un comportamiento similar a su encolamiento en la medida que más paquetes se hayan encolado su retardo se aumenta, los picos que se observan en la figura 44 corresponden a los cambios de la ventana de congestión al implementar las políticas de crecimiento o decremento.

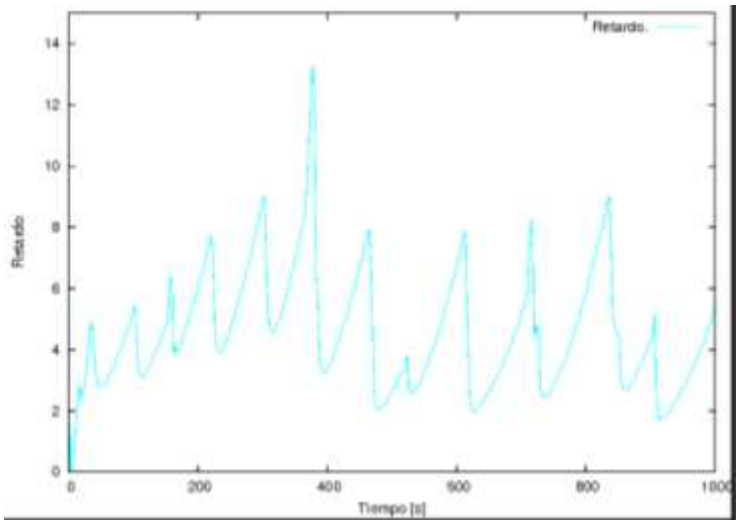


Figura 44 Retardo en la red con la política RVBDC.

Con respecto al comportamiento de las solicitudes que realiza la estación a la NCC con esta política, las solicitudes tienen en cuenta el nivel de la ventana de congestión que se muestra en el nivel de encolamiento presente en el búfer de la estación transmisora. En la figura 45 se indica el comportamiento de las solicitudes realizadas a la NCC con la política RVBDC.

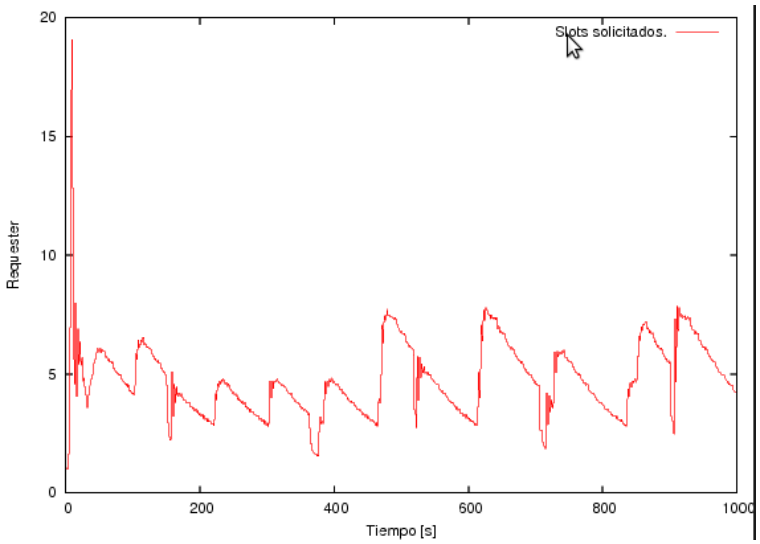
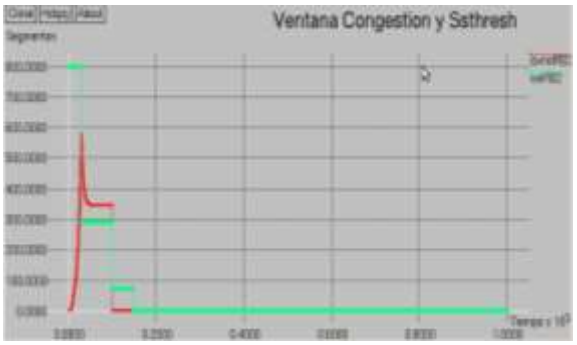


Figura 45 Slots solicitados por la política RVBDC.

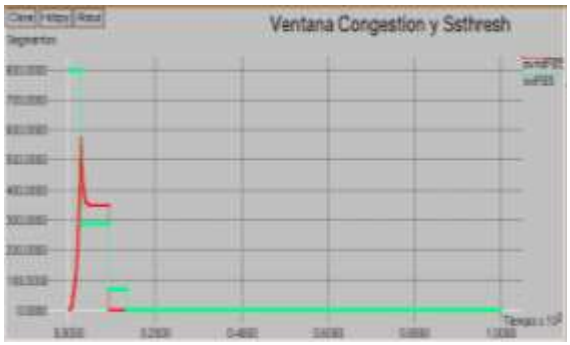
6.2. IMPACTO DE LA POLÍTICA RBDC.

Para el segundo caso se analizó, la asignación por demanda RBDC.



Cwnd ***ssthresh***

Figura 47.1 Ventana de congestión y *ssthresh* para la conexión dos con la política RBDC.



Cwnd ***ssthresh***

Figura 47.2 Ventana de congestión y *ssthresh* para la conexión cinco con la política RBDC.

Al visualizar la figura 47 se indica el pobre desempeño de las ventanas de congestión para las dos conexiones, su desempeño se ve considerablemente afectado por la congestión de la red generada por las demás conexiones y esto se manifiesta en RTTs muy altos, como lo indica la figura 48, por lo tanto el transmisor opta por activar el *timeout* en muchas circunstancias pero como la red continua congestionada la ventana permanece en su valor mínimo.

RTT conexion 2 RTT conexion 5

Figura 48. Comportamiento del RTT para las conexiones que utilizan la política RBDC.

En la figura 49 se indica el comportamiento del *back-off* y el número de retransmisiones que se presentan durante toda la simulación para las conexiones que utilizan la política RBDC. Es de notar que las retransmisiones por el *timeout* se presentan después que *cwnd* entra a la fase de evasión de congestión, luego como el RTT sigue incrementándose (ver figura 48) debido a que la velocidad de entrada de datos a la estación es nula, la estación de control no podrá asignarle recursos y de esta manera la ventana de congestión permanece constante.

Backoff nrexmit

Figura 49.1 Comportamiento del *timeout* en la conexión dos bajo la política RBDC.

Backoff nrexmit

Figura 49.2 Comportamiento del *timeout* en la conexión cinco bajo la política RBDC.

El principal inconveniente que presenta la política RBDC es la manera como realiza la asignación de recursos, al no tener slots para transmitir el tiene que esperar demasiado tiempo para recibir los ACKs que le permitan incrementar su ventana y en la medida que estos no lleguen el transmisor opta por activar el *timeout*.

6.3. IMPACTO DE LA POLÍTICA CRA O CONSTANT.

Para el tercer caso se analizó, la asignación por demanda CRA.

El inicio de la ventana de congestión es muy lento, y cuando entra en la fase de evasión de congestión los paquetes transmitidos son pocos, ya que los slots solicitados son constantes pero en un nivel muy bajo que no permiten un crecimiento más rápido de la ventana de congestión, en la figura 50 se muestra el comportamiento de la ventana y su relación con el número de slots solicitados. El nivel de pendiente es más bajo lo cual evidencia una gran congestión que se presenta en la red.

Figura 50.1 Comportamiento de la ventana de congestión y el *ssthresh* con la política CRA.

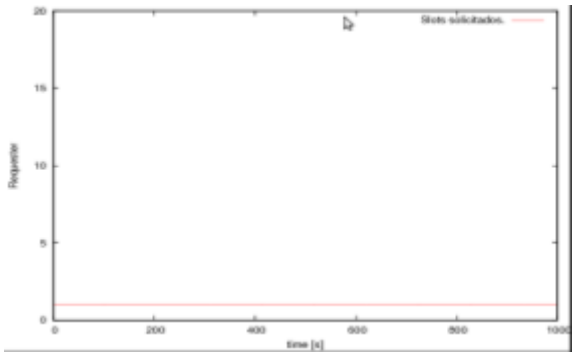


Figura 50.2 **Slots solicitados** a la RCST con la política CRA.

Los paquetes retransmitidos son pocos debido a la política conservadora de TCP HS para períodos de congestión.

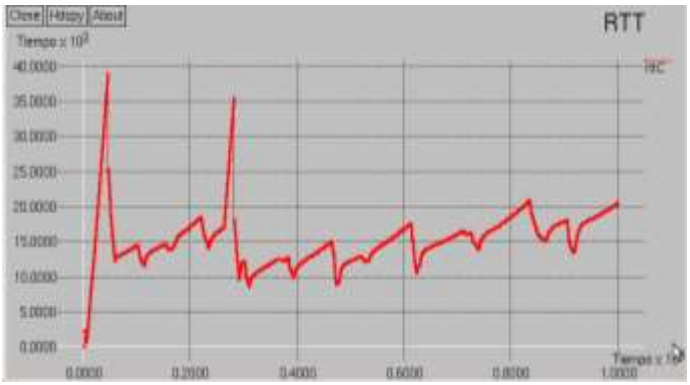


Figura 51 Comportamiento del RTT al utilizar la política CRA.

El RTT medido durante la simulación es alto como se indica en la figura 51, producto de la demora en la llegada de los ACKs al transmisor para incrementar la ventana de congestión.

6.4. IMPACTO DE LA POLÍTICA VBDC.

Para el cuarto caso se analizó, la asignación por demanda VBDC.

De igual forma que en la política RVBDC son políticas que al compartir recursos con otras presentan un buen desempeño en la red, el número de solicitudes a la NCC por cada trama son mayores que para los anteriores casos, y como son acumulativas procuran constantemente disminuir los datos encolados en la capa MAC. En la figura 52 se observa el comportamiento de las solicitudes realizadas a la NCC.

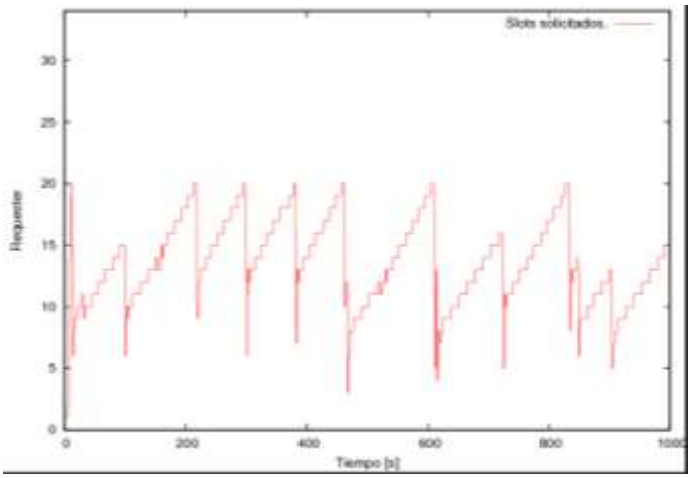


Figura 52 Slots solicitados a la NCC por VBDC.

La ventana de congestión en la figura 55.1 presenta un comportamiento excelente, coherente a la manera como se manejan las solicitudes mediante la política VBDC a la NCC, ya que constantemente se asignan slots para transmitir un gran volumen de datos sin esperar mucho tiempo en su transmisión. En la fase de inicio lento los paquetes retransmitidos se incrementan debido a que TCP HS implementa una política agresiva en su ventana.

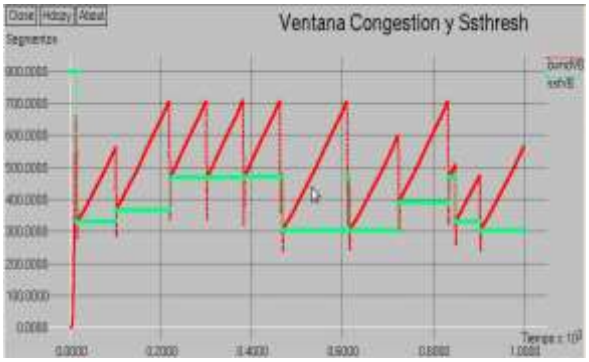


Figura 55.1 Comportamiento de la ventana de congestión y el *ssthresh* con la política VBDC.

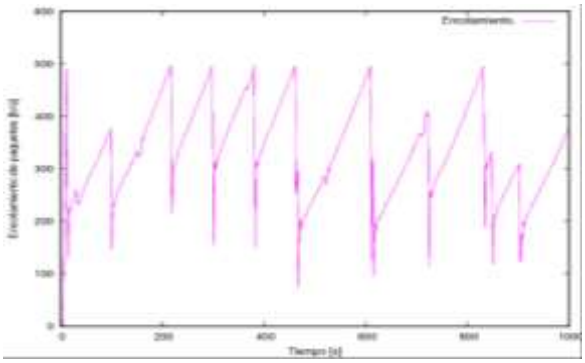


Figura 55.2 Comportamiento de los paquetes encolados en la RCST con la política VBDC.

Las pérdidas que ocurren son por el pequeño tamaño del búfer en la estación, sin embargo la ventana se recupera muy rápidamente de las pérdidas como lo indica la figura 55.2, donde se observa un nivel de pendiente pronunciado en la ventana para las fases de evasión de congestión.

Con respecto al encolamiento de los paquetes el número de paquetes disminuye, lo que da entender que al utilizar la política VBDC se realizan peticiones continuamente a la NCC para disminuir el tamaño del búfer al mínimo, tal como se visualiza en la figura 55.2.

El buen rendimiento de esta política se debe a que mantiene un RTT pequeño, y el retardo en la red es bajo, como se observa en la figura 56.

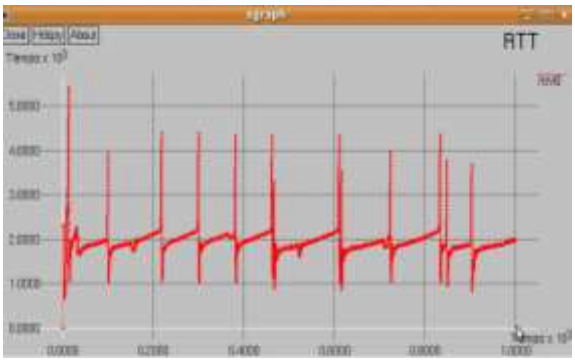


Figura 56.1 Comportamiento del RTT con VBDC.

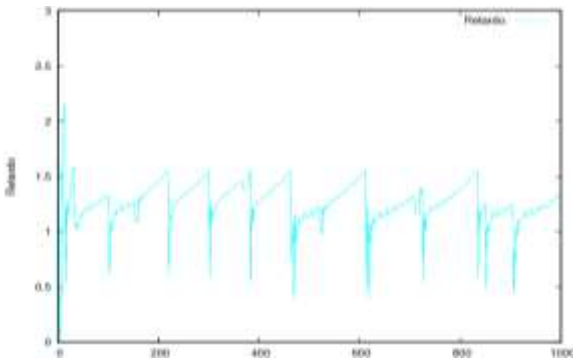


Figura 56.2 Comportamiento del retardo de encolamiento con VBDC.

En la gráfica 57 se observa la utilización del canal con respecto a la capacidad total del enlace, para cada una de las políticas, el desempeño de VBDC y RVBDC son muy buenos, no obstante los slots asignados para las políticas restantes es muy bajo debido ello el rendimiento de la política CRA y RBDC se ven afectados.

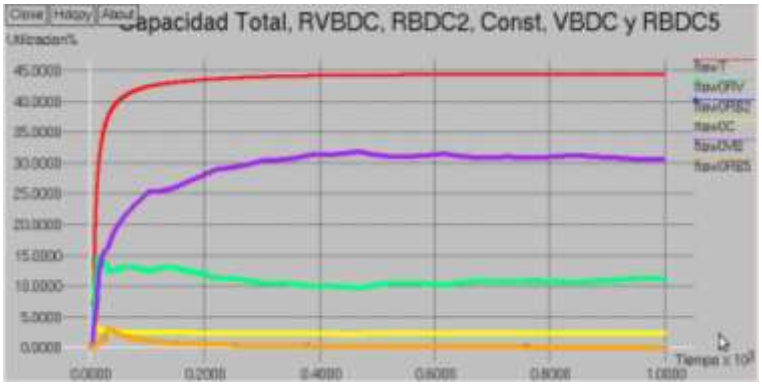


Figura 57. Porcentaje de utilización del canal total y para las diferentes políticas.

ANEXO E

RESULTADOS OBTENIDOS CON LA VERSIÓN TCP WESTWOOD.

En esta sección se muestran los resultados obtenidos del escenario uno, para determinar cuál es impacto que tienen los mecanismos de la capa de enlace de datos en el desempeño del protocolo TCP Westwood (TCPW).

Antes de empezar a analizar los diferentes protocolos de acceso al medio y las políticas de asignación de recursos, se escogerá un tamaño del slot y un tiempo de duración de trama que presente el mejor comportamiento en los parámetros de TCP, como también un buen rendimiento en los parámetros de la red. Para determinar lo dicho anteriormente se analizan cuatro casos donde se varia el tamaño del slot, el número de slots por trama, el número de tramas por supertramas y la duración de la trama, los casos analizar se muestran a continuación.

1.0. IMPACTO AL VARIAR EL NÚMERO DE SLOTS.

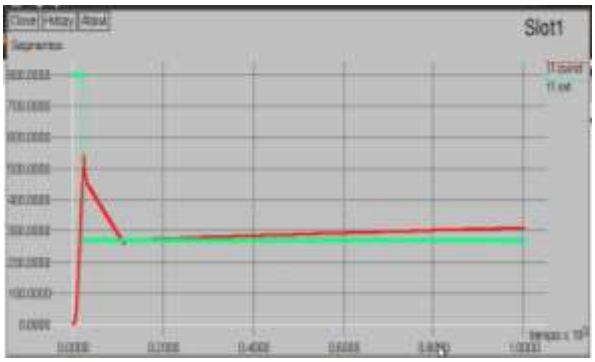
CASO 1: Para este caso se analiza el escenario 1, dejando fijo la duración de la trama, el valor de la supertrama y el número de slots por tramas, se varían los parámetros ancho de banda del enlace satelital y el tamaño del slot, el objetivo de este caso es medir el impacto que tiene estas variaciones en el protocolo TCP Westwood; en la tabla 1 se muestran los datos.

Tabla 1. Parámetros para el caso 1.

Script de simulación	**SlotS1**	**legentL2**
Duración de trama	0,04	0,04
N° de Slots por trama	14	14
Tamaño del slot	184	1504
BW del satélite *uplink*	512200	4219200
N° tramas por supertramas	25	25

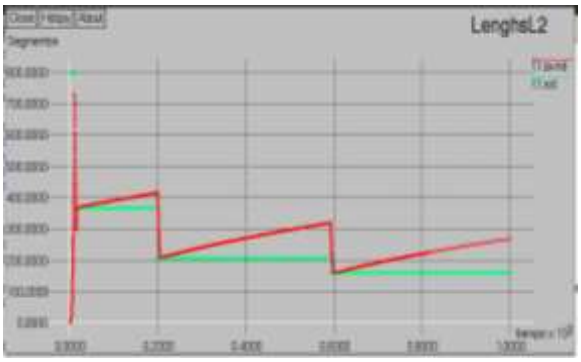
1.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, EL *SSTHRESH* Y PAQUETES RETRANSMITIDOS.

Analizando los datos arrojados por el simulador se observa el impacto que se tiene el tener un tamaño de slot más grande, es un incremento del tamaño de la ventana de congestión de manera más rápida, pero a cambio de esto se produce más congestión en la red, lo que implica presencia de mas ACKs duplicados y por tanto más retransmisiones de paquetes tal y como se visualiza en las figuras 1.1 y 1.2. A continuación se describe lo dicho para una conexión en particular:



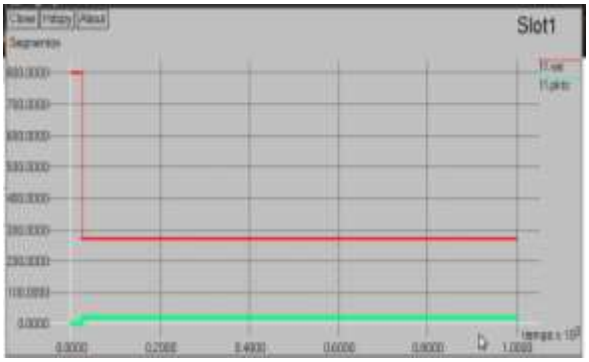
Cwnd ***ssthresh***

Figura 1.1 Ventana de congestión en Slot1.



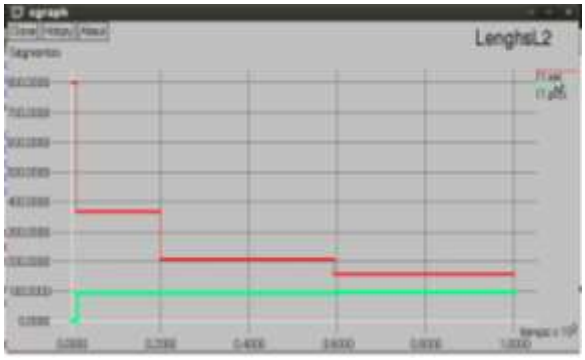
Cwnd ***ssthresh***

Figura 1.2 Ventana de congestión en legentL2.



Ssthresh **Paquetes Retransmitidos**

Figura 1.3 Paquetes transmitidos y *ssthresh* en Slot1.



Ssthresh **Paquetes Retransmitidos**

Figura 1.4 Paquetes transmitidos y *ssthresh* en legentL2.

Para el script Slot1 la ventana de congestión se ve afectada por la presencia de ACKs duplicados en t= 24s tal y como lo muestra la figura 1.1 para el intervalo de tiempo (0 - 24)s *cwnd* crece de manera exponencial de acuerdo al algoritmo de TCPW, cuando este detecta la presencia de 3 ACKs duplicados, *cwnd* decrece de manera adaptativa de acuerdo a los parámetros de congestión, el valor del *ssthresh* se ajusta a un nuevo valor y se entra al estado de evasión de congestión, el cual se mantiene en este estado por el resto de la simulación, *cwnd* crece de manera lineal alcanzando un valor de 310 para t=1000s.

En la figura 1.3 también se observa que cuando hay presencia de ACKs duplicados, los mecanismos de control de flujo de TCPW entran en acción, calculando un nuevo valor de *cwnd* y *ssthresh* los cuales dependen del RTTmin y el ancho de banda disponible en la red calculado por TCPW, cuando hay un decrecimiento de *cwnd*, se observa que hay presencia de paquetes retransmitidos (pktx), su valor alcanza los 22 paquetes, hasta que *cwnd* sea igual al *ssthresh*.

En el script legentL2, en la figura 1.2 se visualiza que *cwnd* alcanza un tamaño más grande (*cwnd*=735) en un tiempo menor, esto gracias a que se tiene un ancho de banda más grande y el tamaño del slot también lo es; pero durante el tiempo de simulación el valor de *ssthresh* cambia de estado tres veces, lo que indica que hay presencia de ACKs duplicados y por tanto hay presencia de paquetes retransmitidos para los cuales pktx llega a un valor de 97 paquetes como lo muestra la figura 1.4.

1.2. DATOS OBTENIDOS DE ACK, RTT Y NÚMERO DE SECUENCIA.

En la figura 2.1 se observa el comportamiento de los ACKs que llegan al transmisor y el número de secuencia de los mismos, durante cortos periodos de tiempo el valor de los ACKs permanece constante y el número de secuencia crece de manera muy lenta, esto indica la presencia de pérdida de paquetes donde se presenta un decremento de *cwnd* (ver figura 1.1) ya que TCPW depende de los ACKs que llegan al transmisor para estimar el ancho de banda disponible o tasa elegible estimada (ERE) para incrementar *cwnd*; durante el resto de tiempo los ACKs y el número de secuencia crecen de manera lineal con una pendiente grande y en la misma proporción

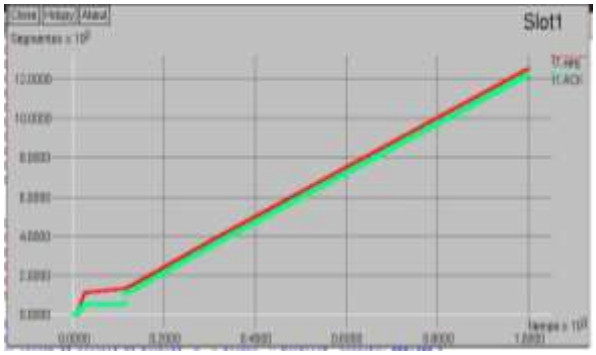


Figura 2.1Evolución del ACK y el número de secuencia en Slot1.

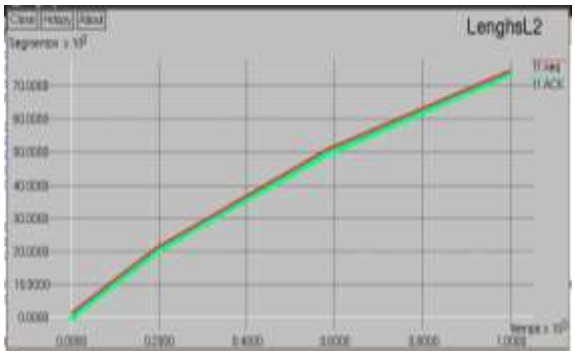


Figura 2.2Evolución del ACK y el número de secuencia en legentL2.

Se observa también que durante los intervalos de tiempo donde los ACKs permanecen constante, el RTT crece muy rápido, debido a la no llegada de ACKs al transmisor que permita calcular el RTT, el cual está indicando perdida de paquetes por congestión de la red, luego *cwnd* y el *ssthresh* es fijado de acuerdo al valor de la ERE, se observa también que durante las etapas de inicio lento TCPW tiene una recuperación de *cwnd* mas rápido que en la etapa de evasión de congestión, por cada ACK recibido por el transmisor, este calcula el RTT y la ERE para incrementar o decrementar *cwnd*; en la figura 2.1 y la figura 3.1 se puede verificar lo dicho anteriormente.

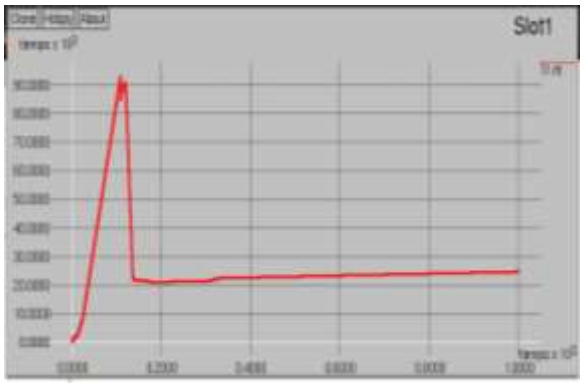


Figura 3.1 Comportamiento del RTT en Slot1.

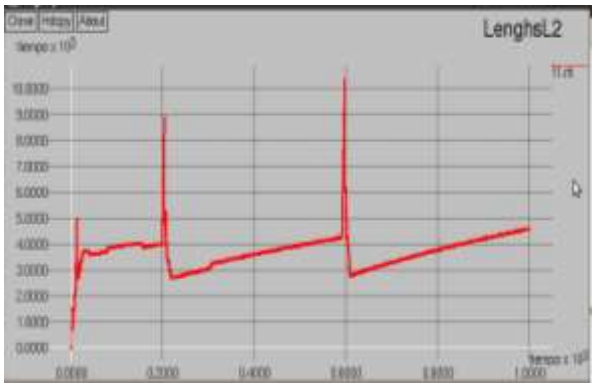


Figura 3.2 Comportamiento del RTT en legentL2.

Por otro lado se tiene que para el script legentL2 el comportamiento de los AKCs y el número de secuencia tiene el mismo desempeño pero para más intervalos de tiempo, esto debido a la presencia de más paquetes perdidos ver figura 2.2, en cuanto al comportamiento del RTT hay presencia de mas picos (ver figura 3.2) donde el valor del RTT está muy por encima de su valor promedio, esto es consecuencia de la no presencia de ACKs en el transmisor, el cual se ve traducido en una reducción de *cwnd* y el *ssthresh*, de acuerdo al valor de ERE calculado por TCPW.

El parámetro *backoff* no se ve afectado al usar un tamaño de slot más grande o pequeño, lo cual indica que no hay presencia de retransmisiones del *timeout*.

Ahora al comparar el número de paquetes transmitidos (ndatapack) en los dos scripts evaluados, la figura 4.1 muestra que para un tamaño de slot de 184 el número de paquetes crece linealmente con una pendiente menos pronunciada durante casi toda la simulación, mientras que para los valores en legentL2 ndatapack crece más rápido con pendientes más pronunciadas, esto debido a que se tiene un ancho de banda más grande y un tamaño de slot más grande, lo que permite tener un crecimiento de *cwnd* mas rápido que el caso anterior. Por lo tanto el impacto de tener un número de slot más grande se ve reflejado en más paquetes trasmitidos, pero a cambio de esto se presenta más retrasmisiones de paquetes debido a la presencia de más ACKs duplicados.

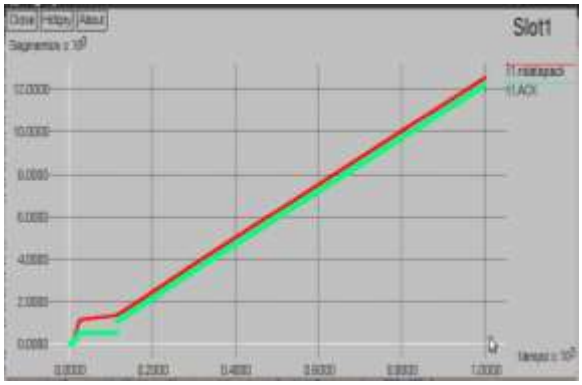


Figura 4.1 Comportamiento de los paquetes transmitidos en Slot1.

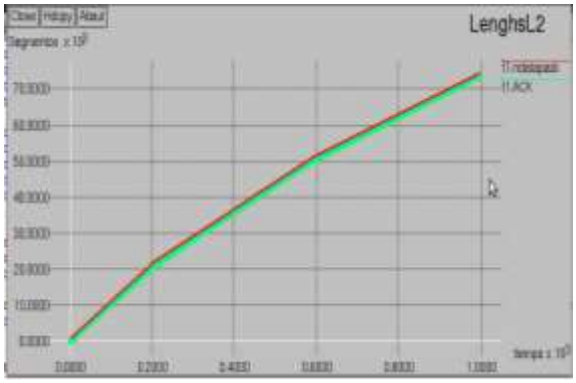


Figura 4.2 Comportamiento de los paquetes transmitidos en Lengh2.

1.3. DESEMPEÑO DEL *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

Al analizar la figura 5 donde se compara en *throughput* de la red se observa que en la gráfica 5.1 correspondiente a los datos de la columna dos de la tabla1 presenta un mejor desempeño que el thp7L2 correspondiente al *throughput* con un tamaño de slot de 1504, mientras que si se analiza la capacidad del enlace utilizado se observa que para un tamaño de slot de 1504 se tiene una mejor utilización llegando hasta un 21%, mientras que para el slot de 184 solo se llega a una utilización de canal de 3.9%, este se observa en las figuras del 5.1 a 5.4, por lo que se puede concluir que para un slot de tiempo más grande se tiene una mejor utilización del canal satelital.

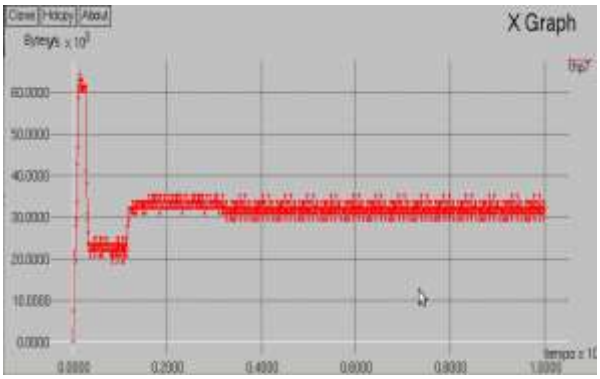
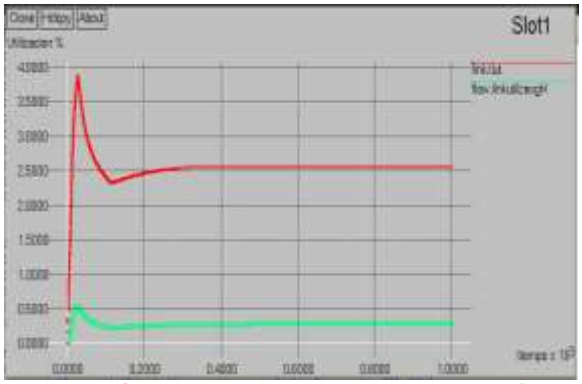


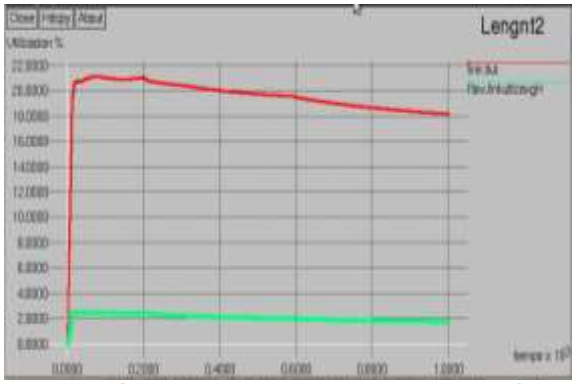
Figura 5.1*Throughput* de la red en Slot1.

Figura 5.2*Throughput* de la red en Lenght2.



Utilización del Canal **Flujo de la conexión 0**

Figura 5.3 Utilización del canal en Slot1.



Utilización del Canal **Flujo de la conexión 0**

Figura 5.4 Utilización del canal en Lenght2.

1.4. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO EN LA RED.

Al inicio de la transmisión de datos se tiene un tamaño de encolamiento en el búfer máximo (500 paquetes) para los datos de la tabla 1, esto es consecuencia a que TCPW durante su etapa de inicio lento transmite de manera muy agresiva para determinar el nivel de congestión de la red, después de un corto periodo de tiempo, el encolamiento en búfer para un tamaño de slot más grande es menor que para un tamaño de slot pequeño, lo que conlleva a un menor tiempo de encolamiento, esto se aprecia en las figuras 6 y en las figura 7.

De lo expuesto anteriormente se puede decir que para un tamaño de slot grande se va a tener un retardo y un encolamiento en búfer más pequeño el cual se ve reflejado en una mejor utilización del canal de transmisión tal y como lo muestran la figura 5.3 y 5.4.

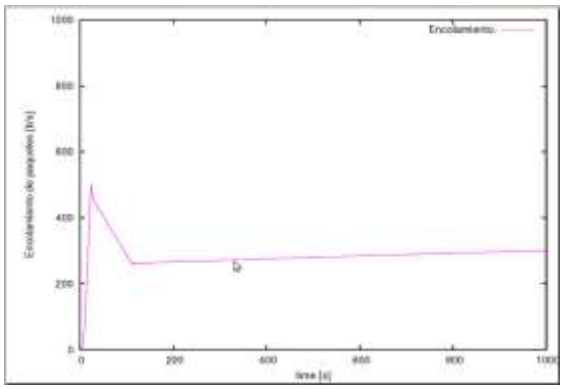


Figura 6.1 Paquetes encolados en el búfer de la RCST en Slot1.

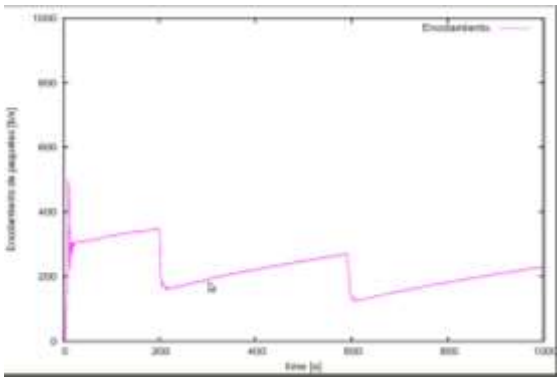


Figura 6.2 Paquetes encolados en el búfer de la RCST en Lenght2.

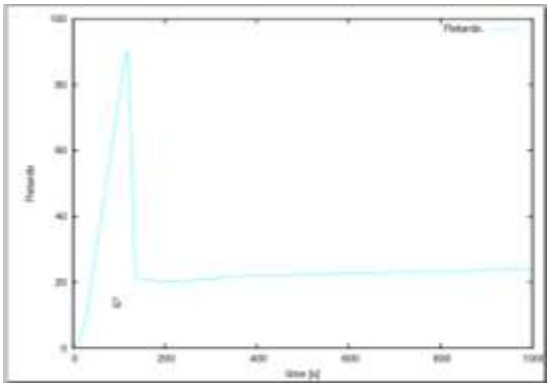


Figura 7.1 Retardo en la red para el script Slot1.

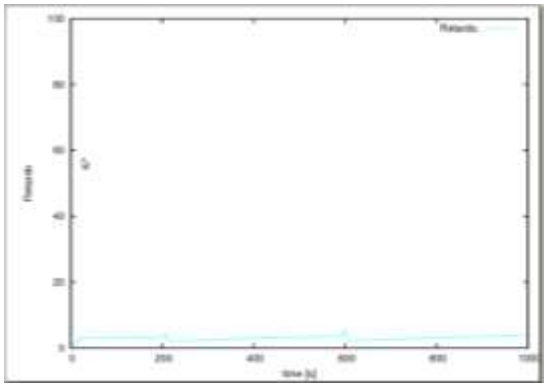


Figura 7.2 Retardo en la red para el script Lenght2.

2.0. IMPACTO AL VARIAR EL TAMAÑO DEL SLOT.

CASO 2: Para este caso se utiliza el escenario de simulación 1 al cual se deja fijo el tamaño del slot y se varía el número de slots por tramas, el objetivo de este caso es poder medir el impacto que tiene el variar el número de slots por trama en el desempeño del protocolo TCP Westwood (TCPW). En la tabla 2 se muestran los parámetros que se tomaron en la simulación.

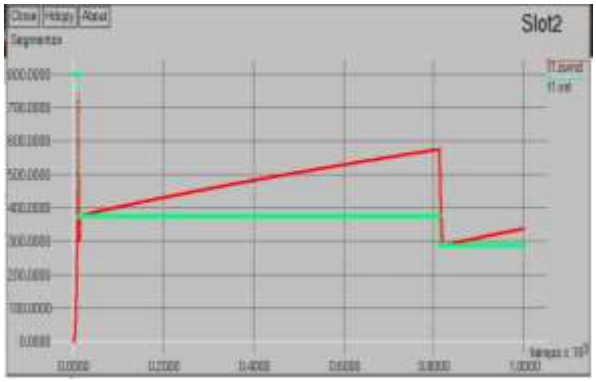
Tabla 2. Parámetros para el caso 2.

Script de simulación	**SlotS2**	**legentL1**
Duración de trama	0,05	0,05
N° de Slots por trama	20	14
Tamaño del slot	1504	1504
BW del satélite *uplink*	4812800	3368960
N° tramas por supertramas	20	20

2.1. DATOS OBTENIDOS DE LA VENTANA DE CONGESTIÓN, EL *SSTHRESH* Y PAQUETES RETRANSMITIDOS.

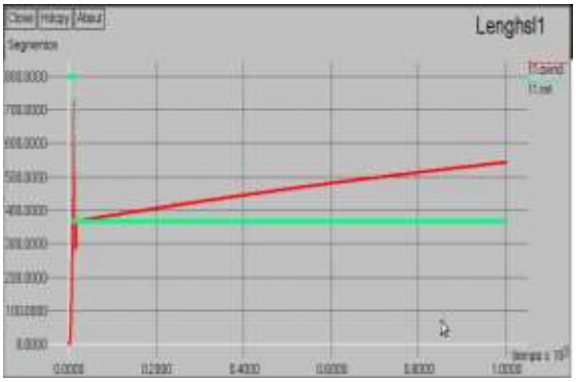
Los datos arrojados por la tabla 2 indican que un mayor número de slots por trama el comportamiento de *cwnd* es más agresivo, es decir crece de manera lineal pero con una pendiente más pronunciada y el *ssthresh* mantiene su umbral en valores superiores que para el caso de un menor número de slots por trama; donde se tiene un crecimiento de *cwnd* más conservado y lineal, en cuanto a los paquetes retransmitidos para ambos casos alcanzan la misma cantidad.

En SlotS2 hay un mayor número de paquetes que para un número menor de slots por trama, pero su diferencia no supera los 4 paquetes, como lo indica las figuras 8.3 y 8.4.



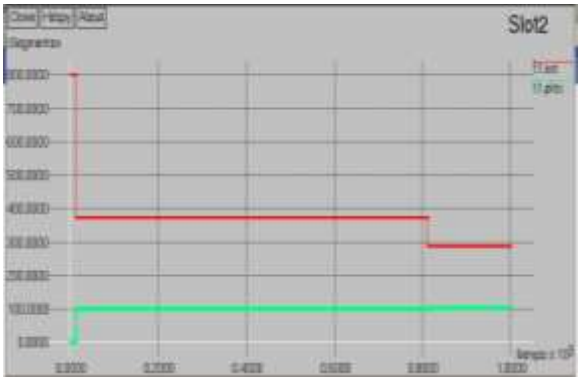
Cwnd ***ssthresh***

Figura 8.1 Ventana de congestión y *ssthresh* con el script SlotS2.



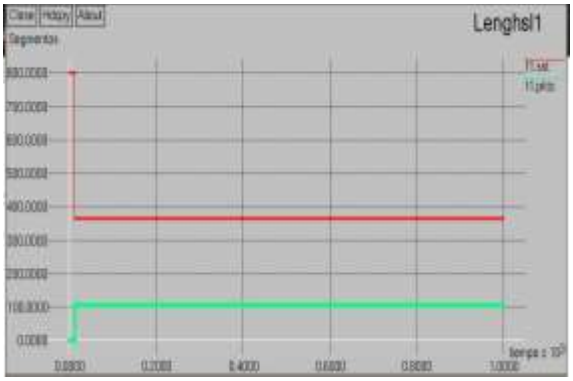
Cwnd ***ssthresh***

Figura 8.1 Ventana de congestión y *ssthresh* con el script legentL1.



Ssthresh **Paquetes Retransmitidos**

Figura 8.3 Paquetes retransmitidos por SlotS2.



Ssthresh **Paquetes Retransmitidos**

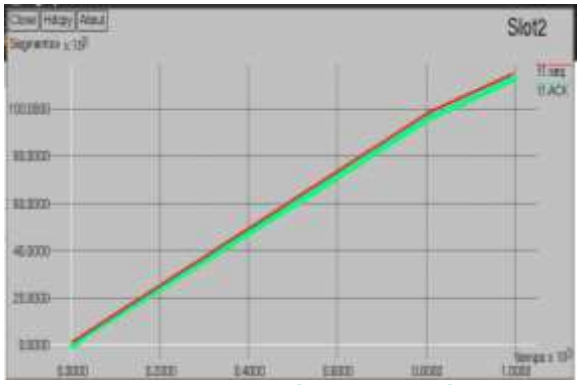
Figura 8.4 Paquetes retransmitidos en legentL1.

Se concluye entonces que el impacto que se tiene el tener más números de slots por trama, es el crecimiento de *CWND* más rápido en un tiempo más corto durante la etapa de evasión de congestión; pero a cambio de esto se presentan más intervalos de tiempo donde se tiene que retransmitir paquetes, que es el inconveniente que menos se desea tener en una red.

2.2. ANÁLISIS DE ACK, RTT Y NÚMERO DE SECUENCIA.

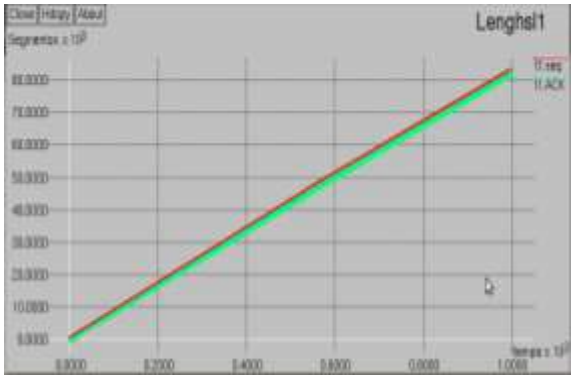
Para estos parámetros de TCPW se tiene que para un mayor número de slots por trama el crecimiento de los ACKs y el número de secuencia tienen una mayor pendiente, lo que indica una mayor llegada de ACKs al transmisor, lo que confirma un mayor crecimiento de *cwnd*, mientras que para un menor número de slots por trama el crecimiento de los ACKs y seq es menor, el cual se ve reflejado en un crecimiento de *cwnd* más conservado y menos paquetes transmitidos. Lo anterior se puede ver en las figuras 9 .1 y 9.2.

Por otro lado se tiene que el comportamiento del RTT en el script Slot2 presenta un mejor comportamiento que en lengtL1.



ACKs **N° de secuencia**

Figura 9.1 Crecimiento del número de secuencia y del ACK en Slot2.



ACKs **N° de secuencia**

Figura 9.2 Crecimiento del número de secuencia y del ACK en lengtL1.

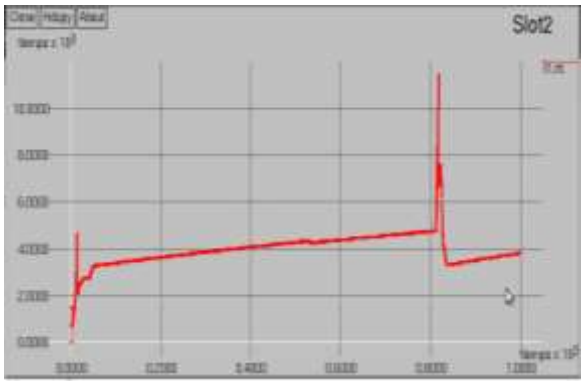


Figura 9.3 Comportamiento del RTT para el script SlotS2.

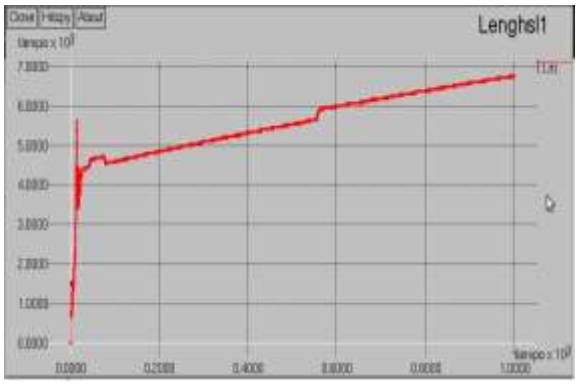


Figura 9.4 Comportamiento del RTT para el script lengtL1.

Por lo tanto se puede decir que el impacto de tener un mayor número de slots por trama se ve reflejado en un mejor comportamiento del RTT; un mayor crecimiento de los ACKs, lo que indica que se han recibido más paquetes con éxito como se muestra en las figuras 10.

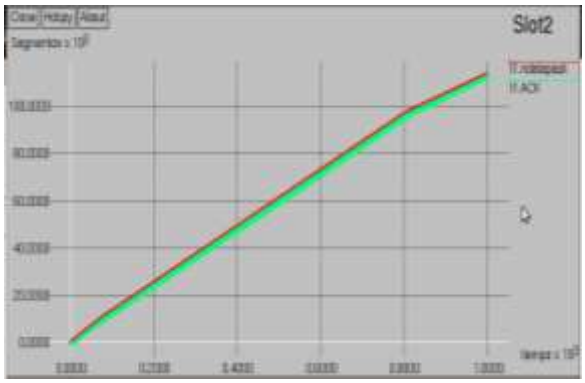


Figura 10.1 Paquetes transmitidos en SlotS2.

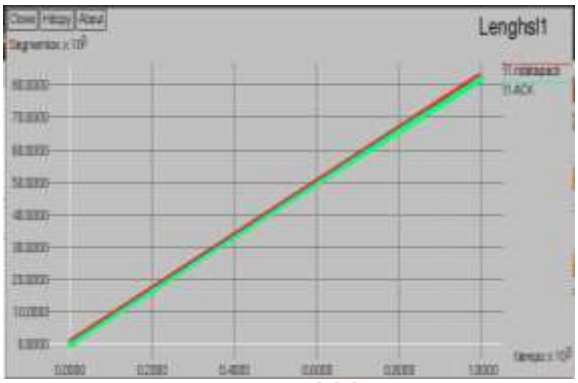


Figura 10.1 Paquetes transmitidos en lengtL1.

2.3. DATOS OBTENIDOS DEL *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

Al observar la figura 11 el script SlotS2 presenta un mejor rendimiento con respecto al t*hroughput* del script lengL1, mientras que si se analiza los resultados de la utilización del canal se observa que para un número de slots por trama de 20 se obtiene una mejor utilización alcanzando un valor hasta del 23%, mientras que si se utiliza 14 slots se obtiene una utilización de canal del 17%, esto se indica en la figura 12; donde se concluye que un mayor número de slots por trama tiene una mejor utilización del canal.

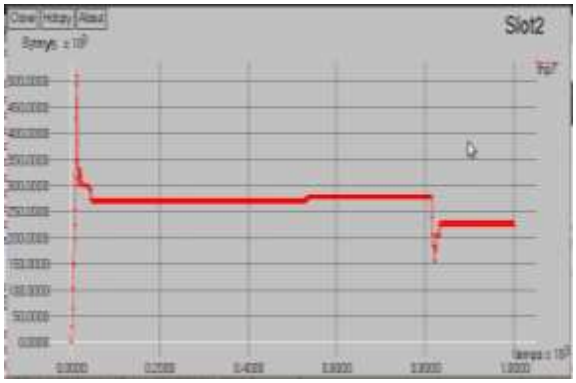


Figura 11.1*Throughput* de la red en Slot2.

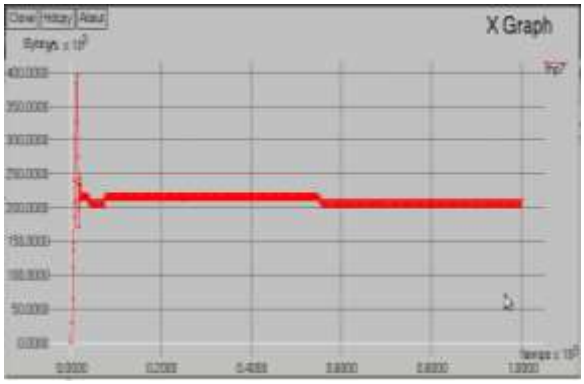


Figura 11.2 *Throughput* de la red en lengtL1.

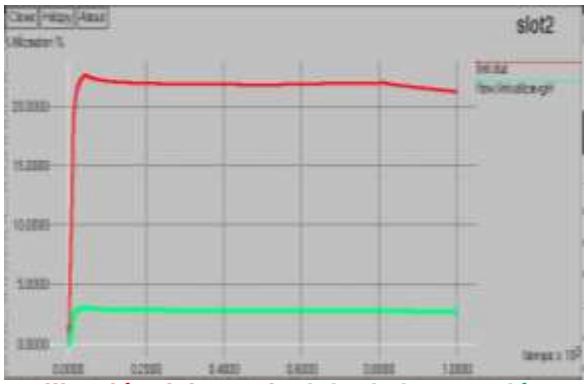


Figura 12.1 Utilización del canal en Slot1.

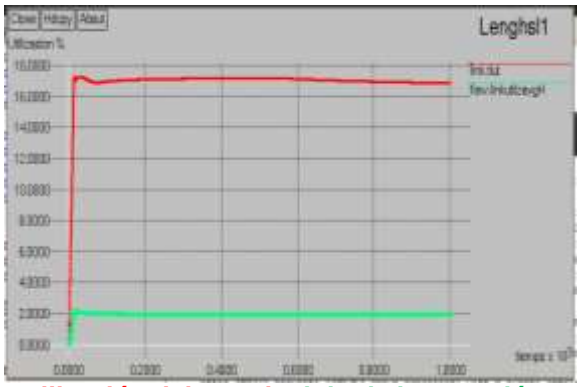


Figura 12.2 Utilización del canal en lengtL1.

2.4. DATOS OBTENIDOS DEL ENCOLAMIENTO Y EL RETARDO EN LA RED.

Se tiene que en SlotS2 el retardo en la red es menor que en lengtL1, como se muestra en la figuras 13.3 y 13.4.

Por tanto un número de slots grande por trama, va a tener un retardo más pequeño y un encolamiento en búfer más grande que se ve reflejado en una mejor utilización del canal de transmisión como se indica en las figuras 12.

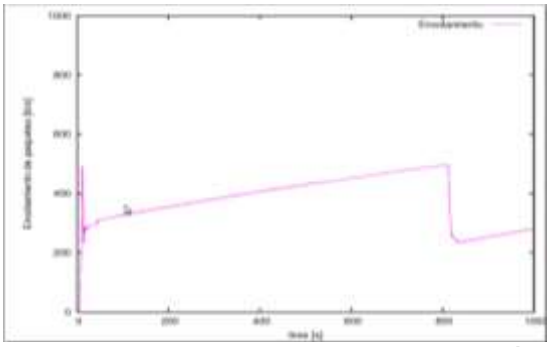


Figura 13.1 Paquetes encolados en el búfer de la RCST en SlotS2.

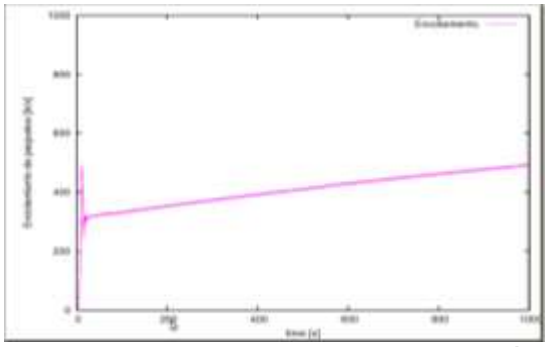


Figura 13.2 Paquetes encolados en el búfer de la RCST en lengtL1.

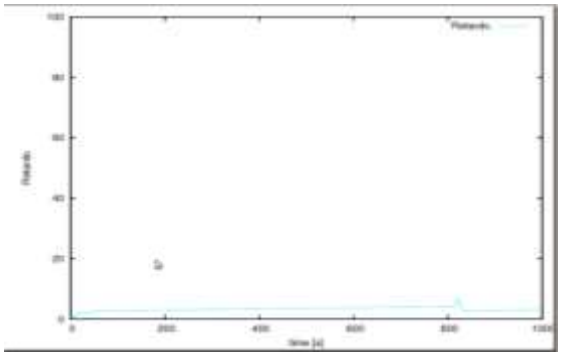


Figura 13.3 Retardo en la red para el script Slots2.

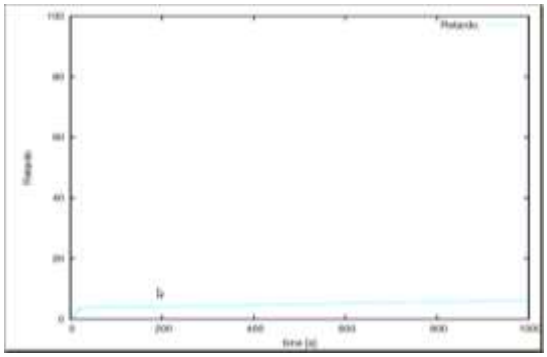


Figura 13.4 Retardo en la red para el script lengtL1.

3.0. IMPACTO AL VARIAR LA DURACIÓN DE LA TRAMA.

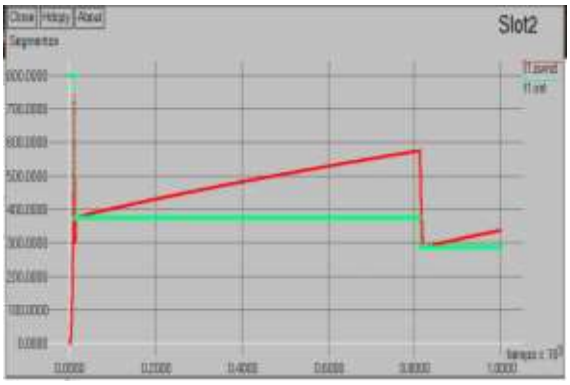
CASO 3: Para esta prueba se deja fijo el tamaño del slot y se varía el número de slots por trama, el número de tramas por supertrama y la duración de la trama; el objetivo de este caso es medir el impacto que tiene la duración de la supertrama en el desempeño del protocolo TCP Westwood (TCPW). En la tabla 3 se muestran los parámetros que se configuraron para la simulación.

Tabla 3. Parámetros para el caso 3.

Script de simulación	**SlotS2**	**SlotS3**
Duración de trama	0,05	0,06
N° de Slots por trama	20	24
Tamaño del slot	1504	1504
BW del satélite *uplink*	4812800	4812800
N° tramas por supertramas	20	17

3.1. ANÁLISIS DE LA VENTANA DE CONGESTIÓN, EL *SSTHRESH* Y PAQUETES RETRANSMITIDOS.

Se tiene que para una duración de trama menor se va a tener un mejor comportamiento de *cwnd* con respecto a su crecimiento y recuperación ante la presencia de ACKs duplicados, caso que no se presenta para una duración de trama más grande, donde *cwnd* crece de manera suave o permanece largos periodos de tiempo en un valor constante, lo que implica menos paquetes transmitidos, en cuanto a los paquetes retransmitidos se tiene que para una duración de trama más pequeña hay mas retransmisiones debido al rápido crecimiento de *cwnd*, mientras que para una duración de trama más grande hay pocas retransmisiones debido al poco crecimiento de *cwnd*. Lo anterior se observa en las figura 14.



Cwnd ***ssthresh***

Figura 14.1 Ventana de congestión y *ssthresh* con el script SlotS2.

Cwnd ***ssthresh***

Figura 14.2 Ventana de congestión y *ssthresh* con el script SlotS3.

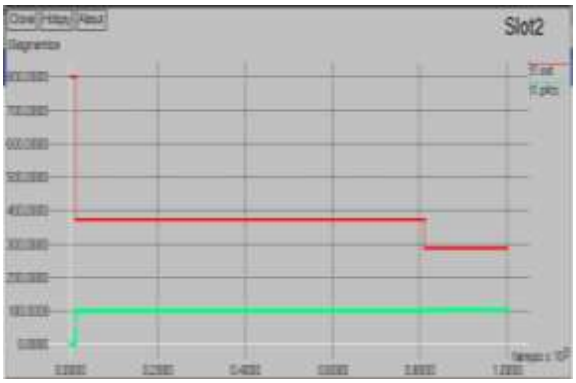


Figura 14.3 Paquetes retransmitidos por SlotS2.

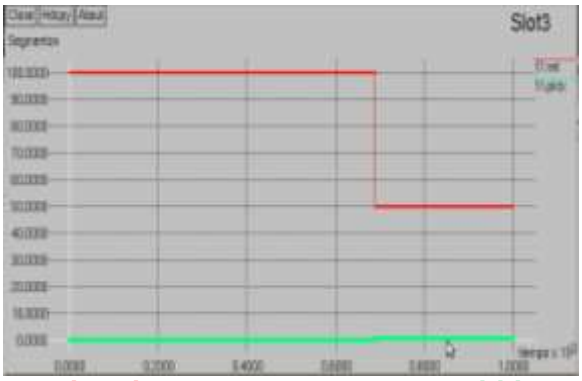


Figura 14.4 Paquetes retransmitidos en SlotS3.

Se concluye entonces que el impacto de tener una duración de trama más grande, se obtiene un crecimiento menos agresivo de la ventana de congestión y en muchos intervalos *cwnd* permanece constante o crece de manera muy suave, lo que implica que se van a tener menos perdidas y en consecuencia menos retransmisiones de paquetes.

Cabe aclarar que si se compara este resultado con la afirmación del caso de estudio 2, se pensaría que existe una contradicción, ya que en el caso dos se tiene que para un mayor número de slots por trama se presenta un mejor desempeño en los parámetros TCP y de la red, pero en este caso ocurre lo contrario, un menor número de slots por trama tiene un mejor comportamiento; la razón de estos resultados esta dado por la ecuación 1, donde indica que el numero de slots por trama, es directamente proporcional al ancho de banda del enlace de subida y a la duración de la trama, mientras que el número de slots por trama es inversamente proporcional al tamaño del slot.

$$Slotframe = \frac{(BW\ UPLink) * Duracion\ Trama}{8 * Tamaño\ del\ Slot} \quad (1)$$

3.2. DATOS OBTENIDOS POR EL ACK, RTT Y EL NÚMERO DE SECUENCIA.

El análisis de estos parámetros es el mismo para el caso 2 de la sección 2, además se tiene que el crecimiento de los parámetros ACK y seq para el script SlotS2 es más lineal, con una pendiente mayor respecto al crecimiento de los ACKs en SlotS3, lo que resulta que en el receptor recibieron menos paquetes como lo indica las figuras 9 y15.

Por otro lado se tiene que el comportamiento del RTT en el script SlotS2 tiene un promedio de 4.7s, mientras que en SlotS3 es de 1.2 segundos.

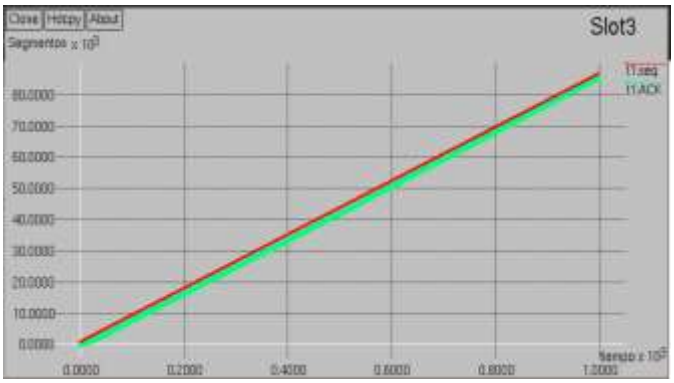


Figura 15. Comportamiento del número de secuencia y ACKs para el SlotS3.

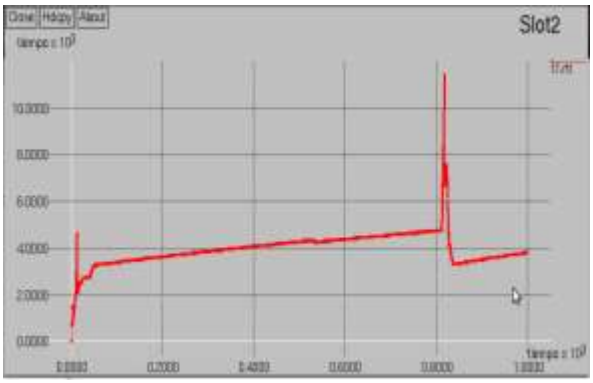


Figura 16.1 Comportamiento del RTT para el script SlotS2.

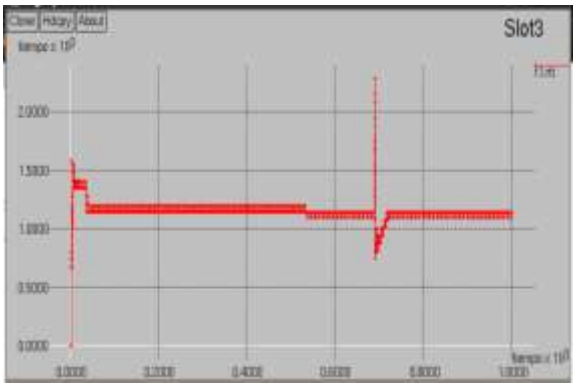


Figura 16.2 Comportamiento del RTT para el script SlotS3.

Por tanto el impacto de tener una duración de trama más grande con un número de tramas por supertrama mas pequeño, el cual permite conservar el mismo TBTP es tener un rtt mucho menor, pero a cambio de esto se tiene menos ACKs en el transmisor lo que influye notablemente en el desempeño de *cwnd* y por ende en la utilización de recursos de la red.

3.3. ANÁLISIS DEL *THROUGHPUT* Y CAPACIDAD DEL ENLACE UTILIZADO.

Al observar la figura 17 se confirma que Slots2 alcanza un mejor rendimiento ya que son más los bits recibidos por segundo que para el script SlotS3, mientras que si se analiza la capacidad del enlace utilizado se infiere que una duración de trama más grande se tiene una menor utilización obteniendo un valor promedio del 16.5% comparado al del script Slots3 que alcanza una utilización del 22%.

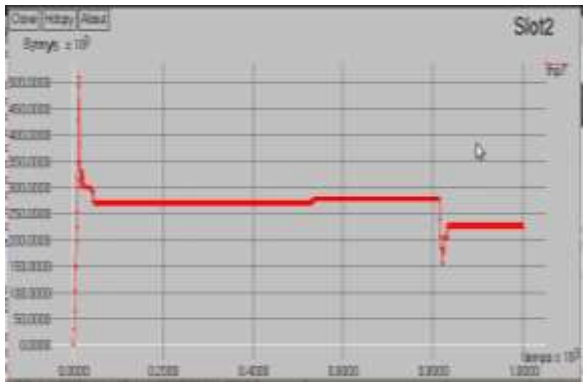


Figura 17.1*Throughput* de la red en SlotS2.

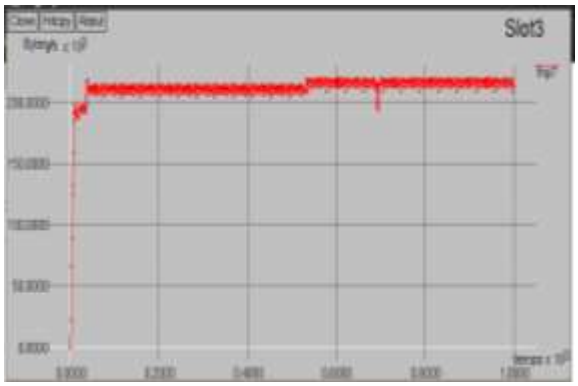
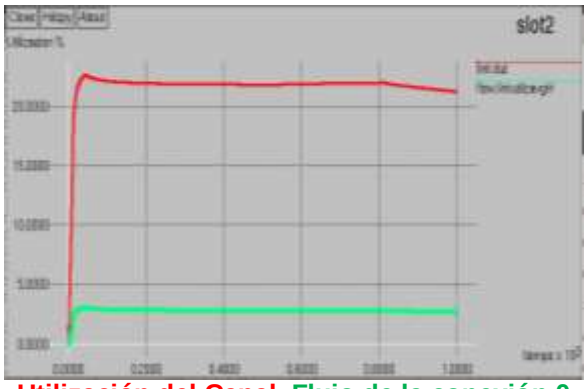
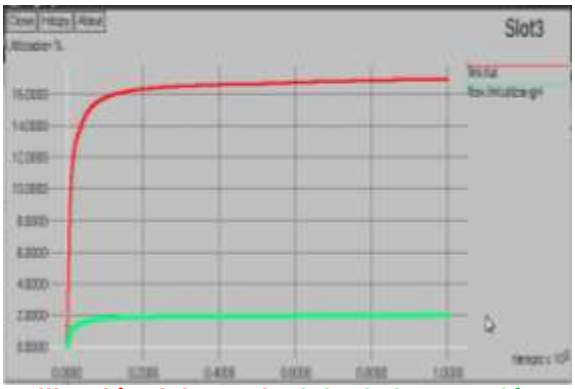


Figura 17.2*Throughput* de la red en SlotS3.



Utilización del Canal Flujo de la conexión 0

Figura 17.3 Utilización del canal en SlotS2.



Utilización del Canal Flujo de la conexión 0

Figura 17.4 Utilización del canal en SlotS3.

3.4. ANÁLISIS DEL ENCOLAMIENTO Y RETARDO DE LA RED.

En la figura 18 se visualiza que para una duración de trama mayor se tiene un menor encolamiento lo que implica también un menor retardo en la red, esto se debe al crecimiento tan lento de *cwnd* que en ocasiones permanece constante, mientras que para una duración de trama menor se tiene un tiempo de encolamiento mayor, debido que mucho más paquetes están siendo transmitidos y los recursos de la red no son suficientes para transmitirlos teniendo que esperar una nueva asignación de recursos.

Se puede decir entonces que el impacto que genera el tener una duración de tramas más grande pero conservando el TBTP de un segundo es un encolamiento y un retardo mucho menor que los casos anteriores, pero a cambio de esto se tiene una utilización de canal menor, y un *throughput* bastante pobre al igual que el crecimiento de *cwnd* y de los ACKs.

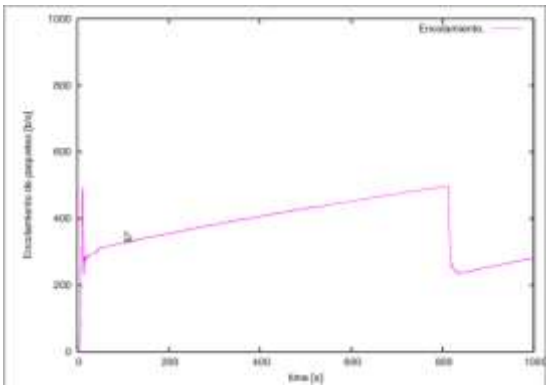


Figura 18.1 Paquetes encolados en el búfer de la RCST en SlotS2.

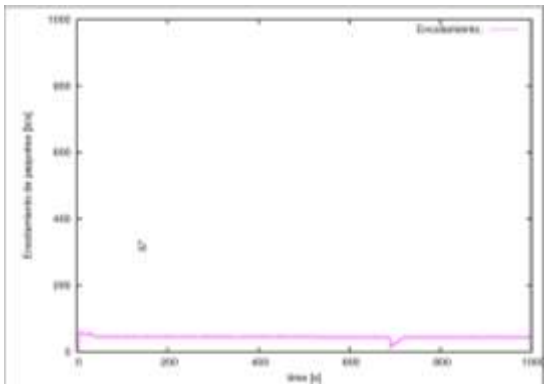


Figura 18.2 Paquetes encolados en el búfer de la RCST en SlotS3.

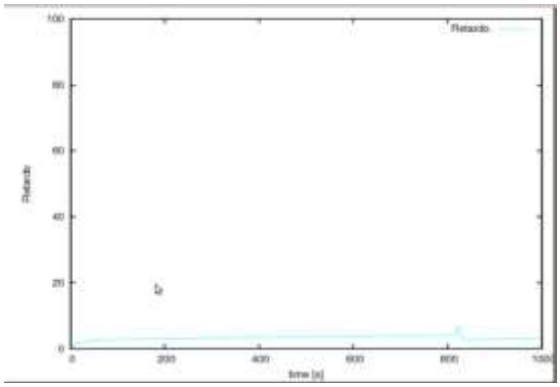


Figura 19.1 Retardo en la red para el script SlotS2.

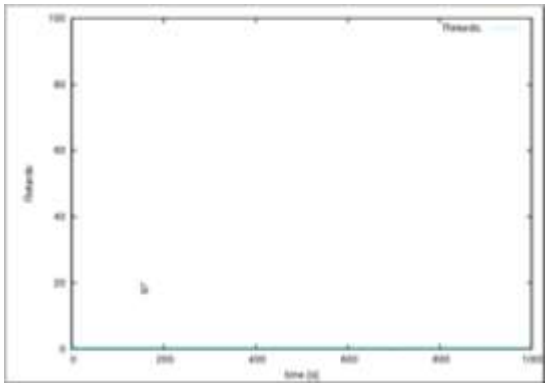


Figura 19.2 Retardo en la red para el script SlotS3.

4.0. IMPACTO AL VARIAR EL TIEMPO DE REPROGRAMACIÓN TPTB.

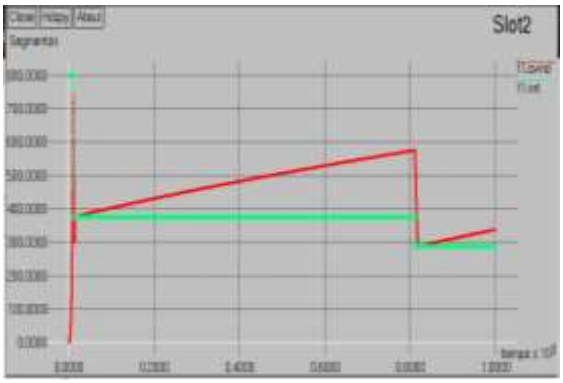
CASO 4: Ahora se describe el impacto que tiene el variar el TBTP por encima y por debajo de un segundo para los parámetros de TCP y del desempeño de la red, los parámetros han sido configurados como lo indica la tabla 5.

Tabla 5. Valores a analizar en TBTP.

Script de simulación	**Slot2=1.0s**	**TBTP1=0.05s**	**TBTP2=1,5s**
Duración de trama	0,05	0,05	0,05
N° de Slots por trama	20	20	20
Tamaño del slot	1504	1504	1504
BW del satélite *uplink*	4812800	4812800	4812800
N° tramas por supertramas	20	10	30

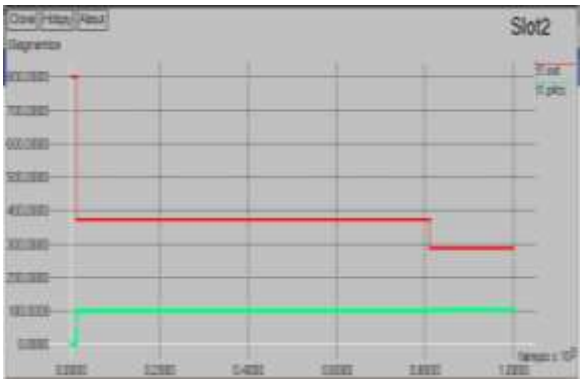
4.1. DATOS OBTENIDOS POR LA VENTANA DE CONGESTIÓN, *SSTHRESH* Y PAQUETES RETRANSMITIDOS.

Al comparar los datos de la tabla 4 entre el slot2 (analizado en el caso dos), TBTP1 y TBTP2 que el comportamiento de *cwnd* es casi similar, se diferencia en la manera de cómo crece durante las etapas de inicio lento y evasión de congestión, mientras que para un TBTP más grande *cwnd* crece de manera más conservada, para un TBTP más pequeño *cwnd* crece de manera lineal pero más rápido, a cambio de esto se presentan más paquetes retransmitidos para un TBTP más pequeño, debido al rápido crecimiento de *cwnd*, esto se ve reflejado en los demás parámetros de TCP, como en la llegada de más ACKs al transmisor, más paquetes transmitidos y un incremento del número de secuencia; el parámetro *backoff* no se ve afectado ya que no se presenta ninguna retransmisión del *timeout*, esto se observa en las figuras 20, 21 y 22.



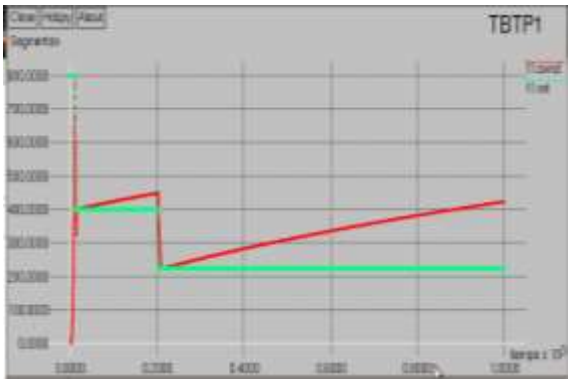
Cwnd ***ssthresh***

Figura 20.1 Ventana de congestión y *ssthresh* con el script SlotS2.



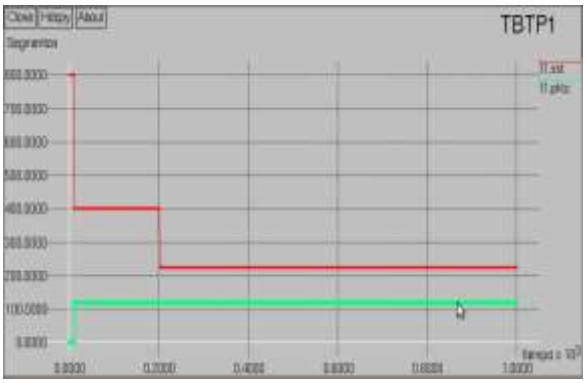
Ssthresh **Paquetes Retransmitidos**

Figura 20.2 Paquetes retransmitidos en SlotS2.



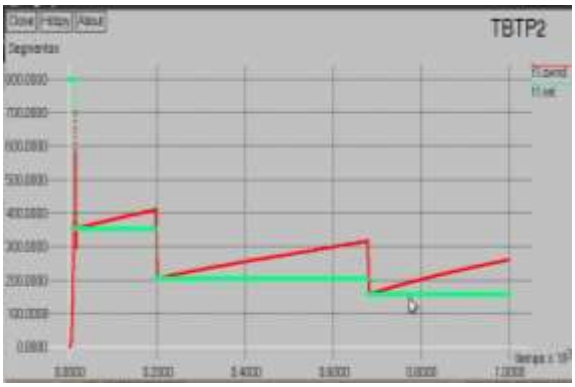
Cwnd ***ssthresh***

Figura 21.1 Ventana de congestión y *ssthresh* con el script TPTB1.



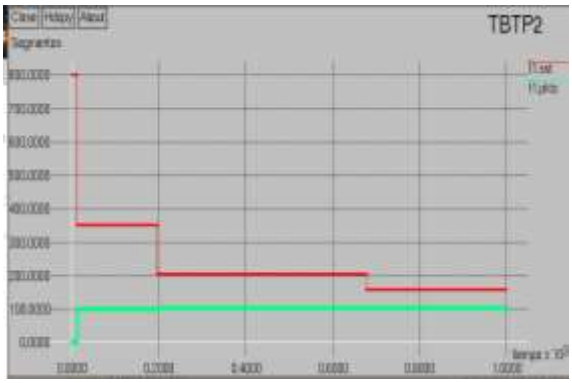
Ssthresh **Paquetes Retransmitidos**

Figura 21.2 Paquetes retransmitidos en TPTB1.



Cwnd **ssthresh**

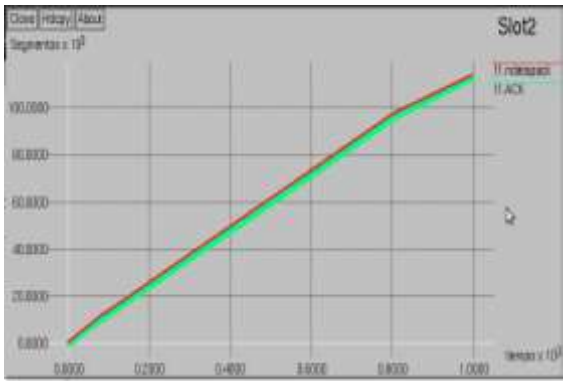
Figura 22.1 Ventana de congestión y *ssthresh* con el script TBTP2.



Ssthresh **Paquetes Retransmitidos**

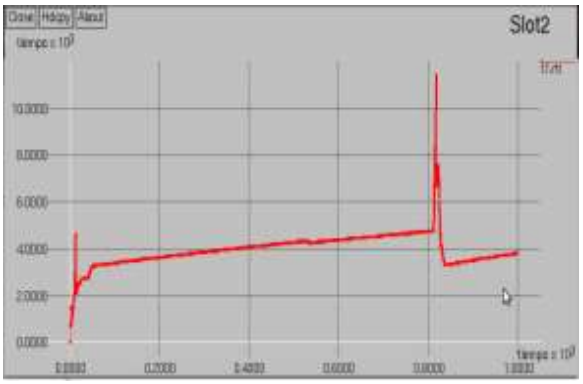
Figura 22.2 Paquetes retransmitidos en. TBTP2.

Luego el impacto que se tiene el variar el TBTP en los parámetros de TCPW está en que la ventana de congestión crece de manera lineal más rápida o más lenta, lo que influye en la llegada de ACKs al transmisor, en los paquetes transmitidos y retransmitidos; por otro lado se tiene que los parámetros *backoff* y RTT no se ven afectados ya que presentan el mismo comportamiento como se indica en las figuras 23 y 24 y 25.



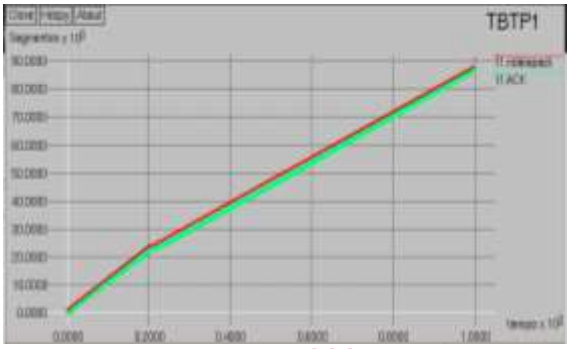
Paquetes Transmitidos **ACKs**

Figura 23.1 Comportamiento de los paquetes transmitidos en SlotS2.



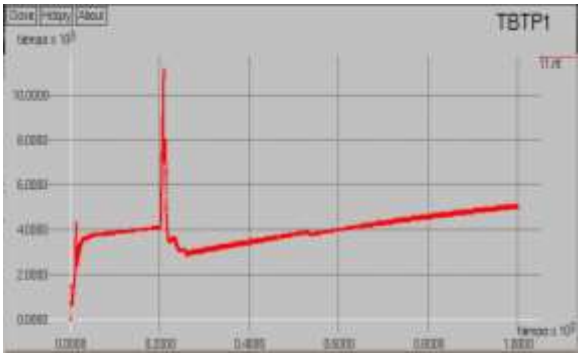
RTT

Figura 23.2 Comportamiento del RTT en SlotS2.



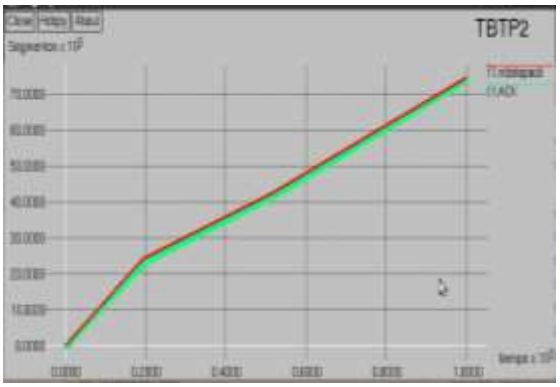
Paquetes Transmitidos **ACKs**

Figura 24.1 Comportamiento de los paquetes transmitidos en TBTP1.



RTT

Figura 24.2 Comportamiento del RTT en TBTP1.



Paquetes Transmitidos ACKs

Figura 25.1 Comportamiento de los paquetes transmitidos en TBTP2.

RTT

Figura 25.2 Comportamiento del RTT en TBTP2.

4.2. ANÁLISIS DE *THROUGHPUT* Y LA CAPACIDAD DEL ENLACE UTILIZADO.

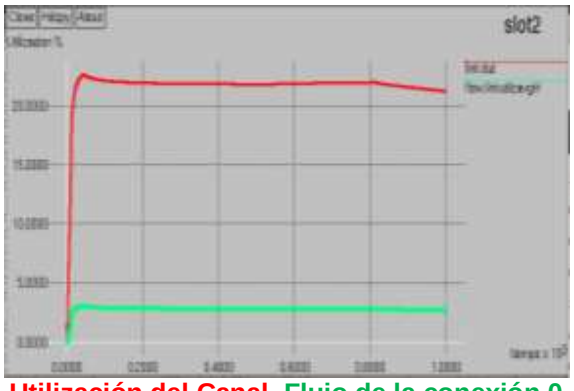
Al observar las figuras 26, 27 y 28 indican que un TBTP que se encuentre alrededor de un segundo tiene el mejor *throughput* y utilización de canal que para los otros casos, en la tabla 6 se observan algunos datos.

Tabla 6. *Thorughput* y utilización del canal promedio.

TPBT(s)	***Thput max***	***Thput promedio***	***Utilización del canal máximo.***	***Utilización del canal promedio***
0,5	552000	245000	21,8%	18%
1	502000	273000	22,5%	22%
1,5	452000	200000	21%	17%

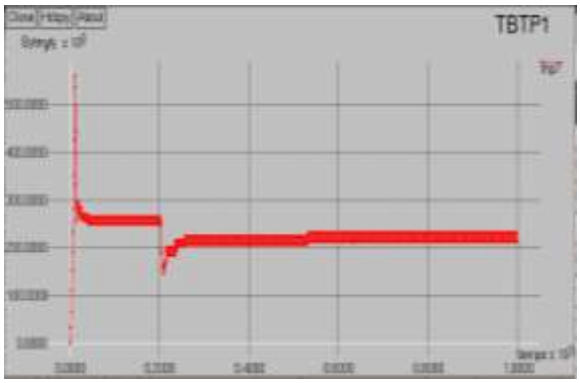
Throughput

Figura 26.1 *Throughput* de la red en SlotS2.



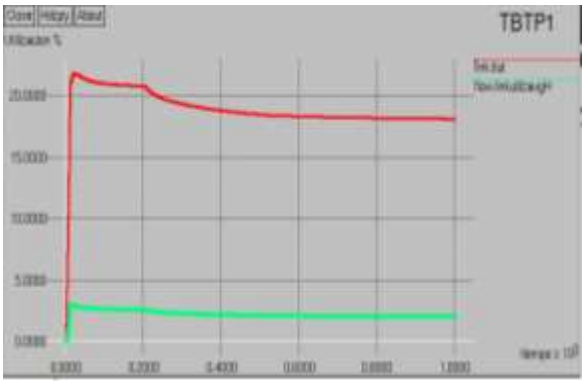
Utilización del Canal Flujo de la conexión 0

Figura 26.2 Utilización del canal en SlotS2.



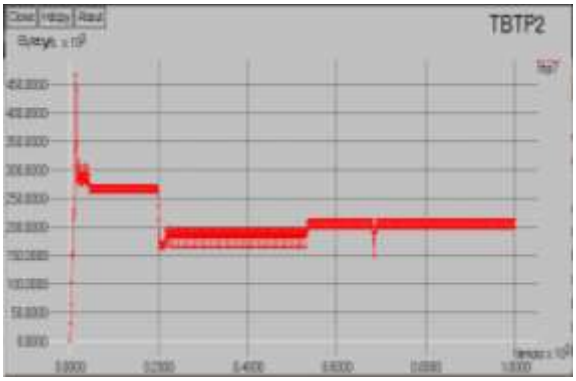
Throughput

Figura 27.1 *Throughput* de la red en TBTP1.



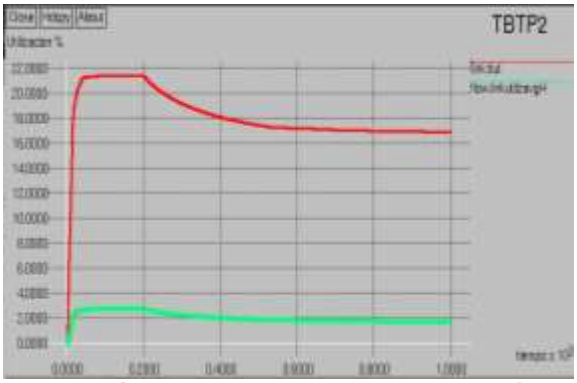
Utilización del Canal **Flujo de la conexión 0**

Figura 27.2 Utilización del canal en TBTP1.



Throughput

Figura 28.1 *Throughput* de la red en TBTP2.



Utilización del Canal **Flujo de la conexión 0**

Figura 28.2 Utilización del canal en TBTP2.

Por lo tanto se puede concluir que una duración de TBTP que se encuentre alrededor de un segundo, presenta la mejor utilización del canal y alcanza el mejor desempeño de la red.

4.3. DATOS OBTENIDOS DEL ENCOLAMIENTO Y RETARDO EN LA RED.

Al observar las figuras 29, 30 y 31 se puede concluir que el encolamiento es muy similar para los tres casos, por lo tanto se afirma que una variación del TBTP no afecta de manera significativa este parámetro.

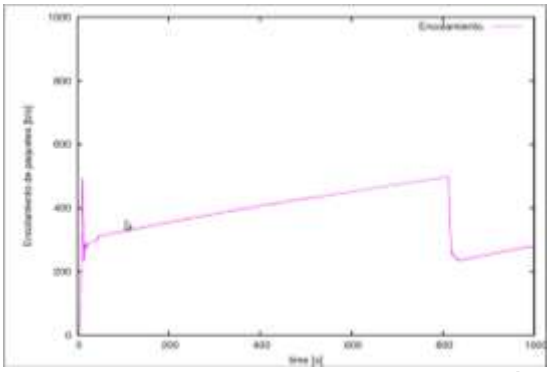


Figura 29.1 Paquetes encolados en el búfer de la RCST en SlotS2.

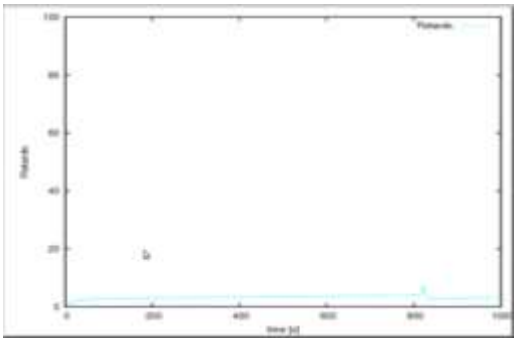


Figura 29.2 Retardo en la red en SlotS2.

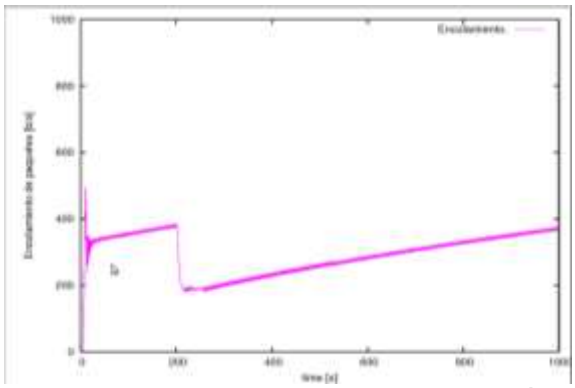


Figura 30.1 Paquetes encolados en el búfer de la RCST en TBTP1.

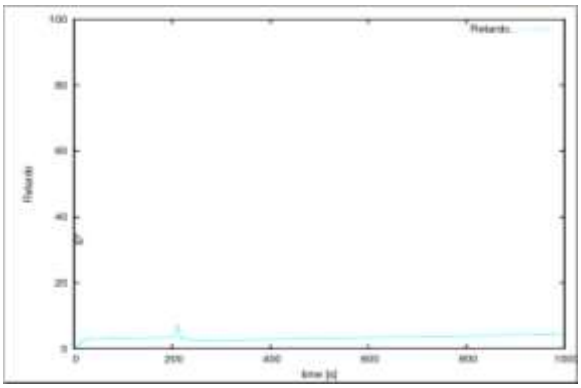


Figura 30.2 Retardo en la red para TBTP1.

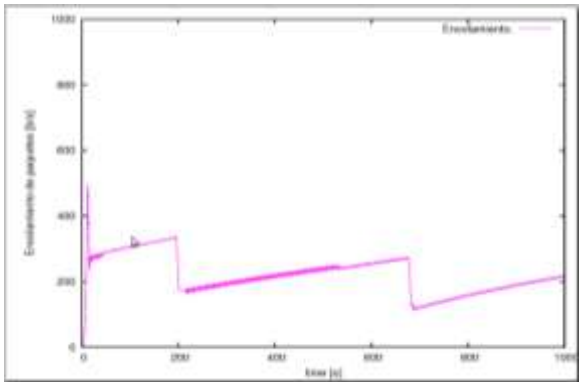


Figura 31.1 Paquetes encolados en el búfer de la RCST en TBTP2.

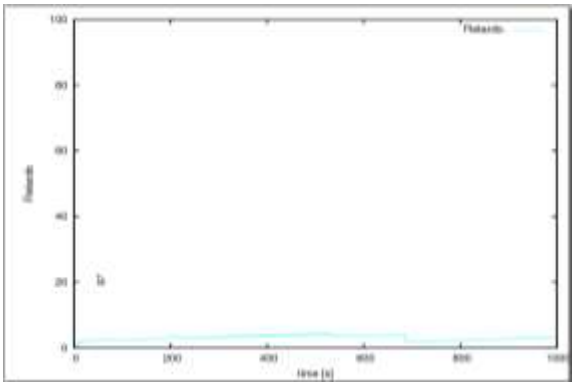


Figura 31.2 Retardo en la red para TBTP2.

Comparando los cuatro casos de estudio y los resultados obtenidos para esta prueba, se llega a la conclusión que SlotS2 presentó el mejor comportamiento en TCP.

5.0. IMPACTO DE LAS POLÍTICAS DE ASIGNACIÓN DE RECURSOS PARA TDMA-DAMA.

En esta sección se describen los resultados obtenidos al implementar las políticas de asignación de recursos en la NCC como lo son FODA y Proportional. Mediante esta prueba se pretende determinar cuál es la política de asignación de recursos que mejor asignación presenta al evaluar los parámetros de desempeño y del protocolo TCP, el escenario escogido fue SlotS2.

En la tabla 7 se muestra el comportamiento de la ventana de congestión, y el *ssthresh* para las políticas de asignación de recursos FODA y Proportional.

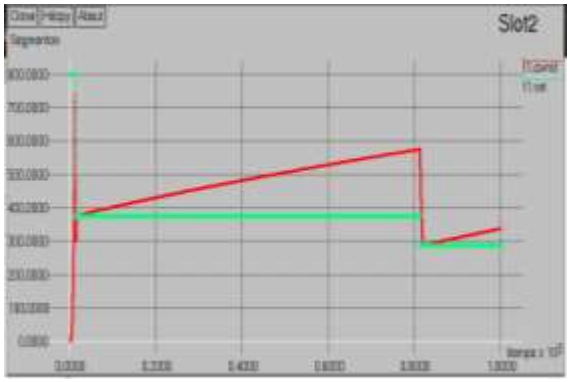
Tabla 7. Comportamiento de *cwnd* y *ssthresh*.

Asignación Proporcional			**Asignación FODA**		
Tiempo (s)	***CWND***	***Ssthresh***	**Tiempo (s)**	***CWND***	***ssthresh***
(0 - 10)	1-800	800	(0 – 11,75)	1 - 748	800
(10,1 – 12,75)	800 - 322	800 – 400	(11,8 – 14,8)	748 - 300	800 – 372
(13 - 99)	400 - 459	400	(15 – 812,3)	300 - 573	372
(100,6 – 263,7)	231 - 388	230	(812,4 - 1000)	287 - 337	287
(264,9 - 466)	204 - 403	194			
(467 - 1000)	Comportamiento diente de sierra (204 - 340)	150-190			

A partir de los datos de la tabla 7 se aprecia que el comportamiento de *cwnd* es más variable en FODA como se observa en la figura 32, para el caso de asignación Proporcional *cwnd* crece de manera más rápida, en intervalos de tiempo más cortos, lo que se ve reflejado en más paquetes transmitidos, una mayor utilización del canal y un mayor *throughput*, como se indica en las figuras 33 a 36.

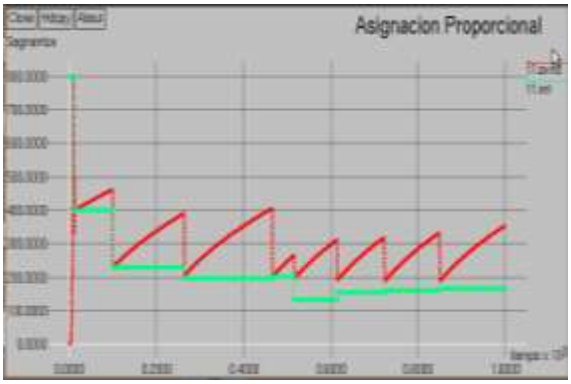
Por otro lado se tiene que si se analiza el archivo *curral.tr* generado por la simulación este indica la forma en que la NCC está haciendo la asignación de recursos, en la fase de inicio lento TCPW transmite de una manera más agresiva y *cwnd* alcanza valores más grandes.

Para el caso de FODA no hay una justa asignación de recursos, se caracteriza por un desperdicio en el ancho de banda disponible al no utilizar un gran número de slots asignados, para la conexión cero se hace una mayor asignación de los slots que para las demás conexiones; mientras que para el caso Proporcional la asignación es más equitativa para todas las estaciones, lo cual se ve reflejado en una mejor utilización de los recursos de la red. Para los demás intervalos de tiempo *cwnd* entra en evasión de congestión donde FODA continua inutilizando un gran número de slots, lo que se traduce en recursos perdidos de la red.



Cwnd ***ssthresh***

Figura 32.1 Ventana de congestión con la política de asignación FODA.



Cwnd ***ssthresh***

Figura 32.2 Ventana de congestión con la política de asignación Proportional.

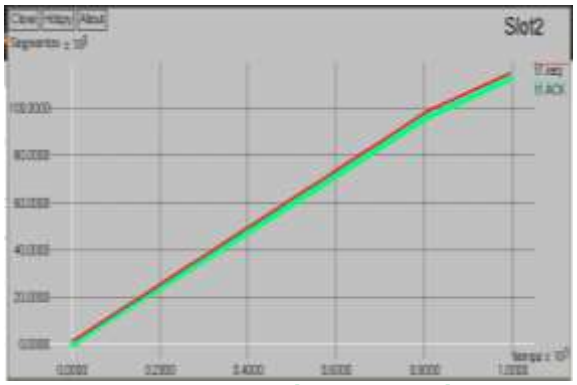


Figura 33.1Evolución del ACK y el número de secuencia con la política de asignación FODA.

Figura 33.2 Evolución del ACK y el número de secuencia con la política de asignación Proportional.

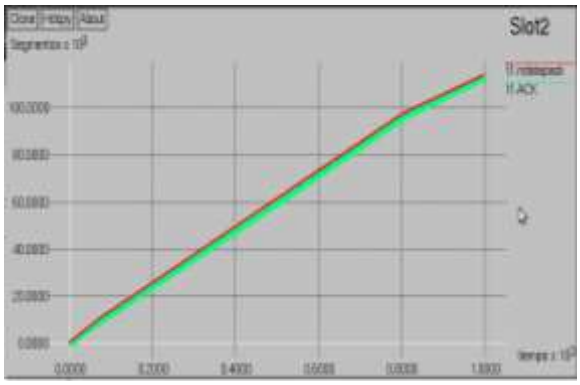


Figura 34.1 Paquetes de datos transmitidos con la política de asignación FODA.

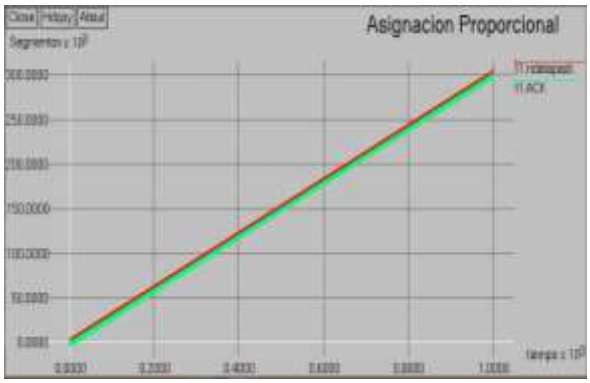


Figura 34.2 Paquetes de datos transmitidos con la política de asignación Proportional.

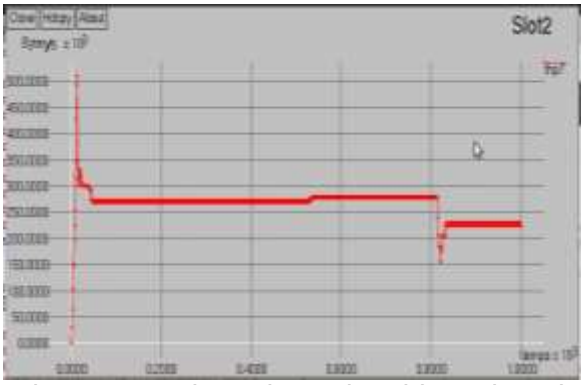


Figura 35.1 *Throughput* obtenido en la red con la política de asignación FODA.

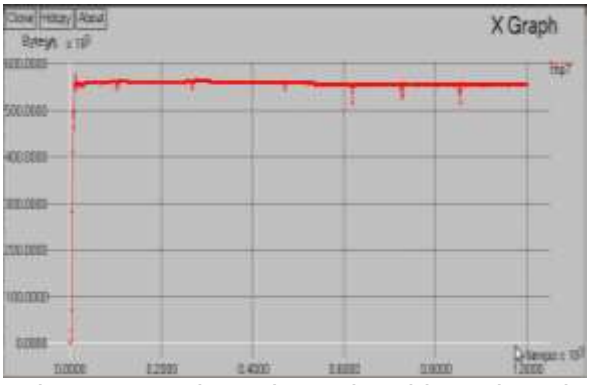
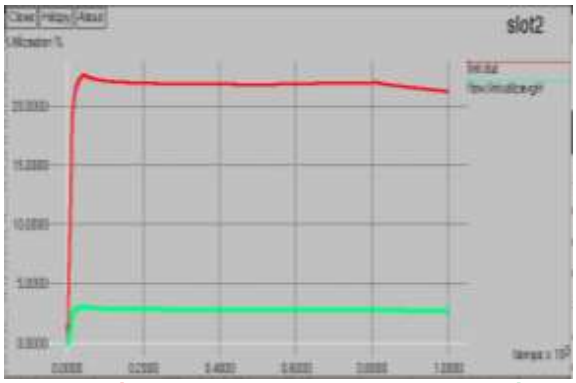
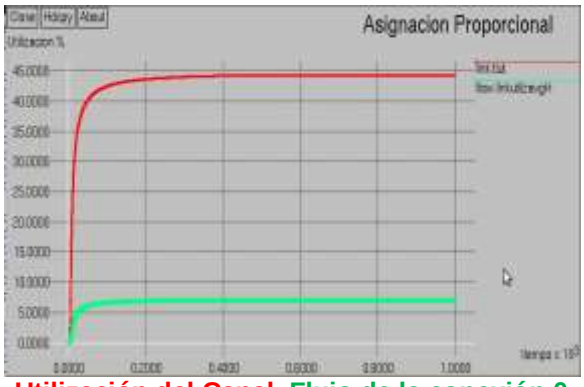


Figura 35.2 *Throughput* obtenido en la red con la política de asignación Proportional.



Utilización del Canal Flujo de la conexión 0

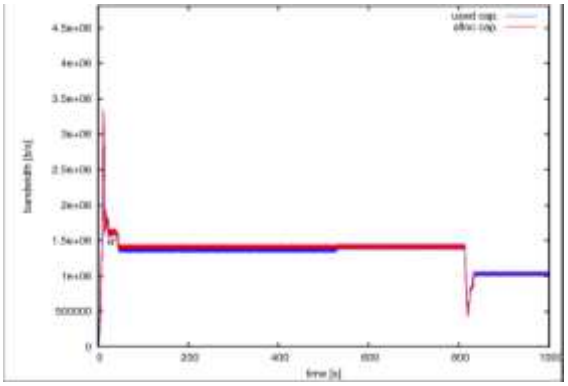
Figura 36.1 Utilización del canal para la política de asignación FODA.



Utilización del Canal Flujo de la conexión 0

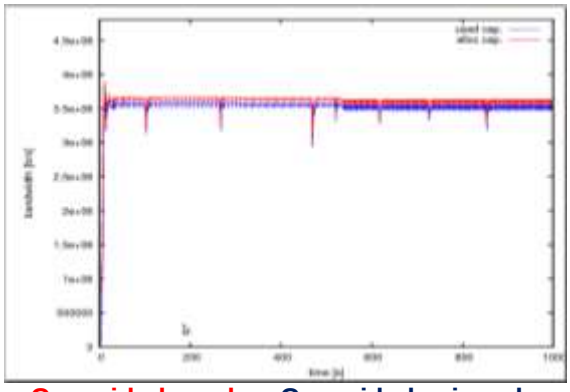
Figura 36.2 Utilización del canal para la política de asignación Proportiona.

Lo dicho anteriormente se puede verificar en las figuras 37 donde se muestran la capacidad asignada y utilizada por la terminal RCST. En la figura 37 se aprecia también el *throughput* y la utilización del canal que se duplica con el uso de la política proporcional comparado con FODA, el cual se refleja en más ACKs recibidos por parte del transmisor, más paquetes transmitidos y un menor RTT.



Capacidad usada Capacidad asignada

Figura 37.1 Utilización del canal en el enlace de subida con la política FODA.



Capacidad usada Capacidad asignada

Figura 37.2 Utilización del canal en el enlace de subida con la política FODA.

Se puede concluir que según el tipo de asignación de recursos que se utilice en la estación de control, esta impacta notablemente los parámetros de TCP (*cwnd*, ACKs, pktx, RTT) y en consecuencia en su desempeño; el único parámetro que no se ve afectado es el *backoff*, ya que en ningún momento ocurren retransmisiones a causa del *timeout*.

Al analizar el encolamiento que se produce en el búfer se observa que en las figuras 38 y 39 la política de asignación proporcional presenta un menor número de paquetes encolados que en la asignación FODA, lo que se ve reflejado en un menor retardo, un menor RTT, y más ACKs recibidos en el transmisor.

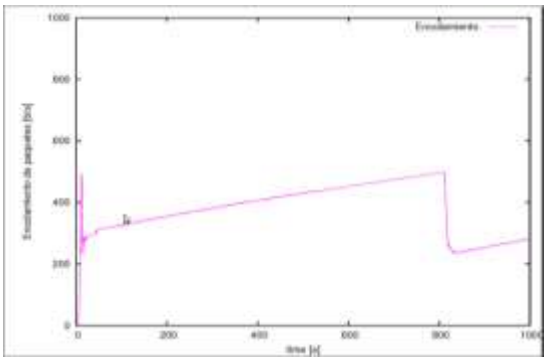


Figura 38.1 Encolamiento producido con el uso de la política FODA.

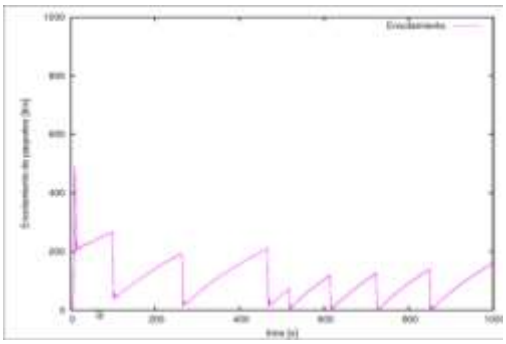


Figura 38.2 Encolamiento producido con el uso de la política Proportional.

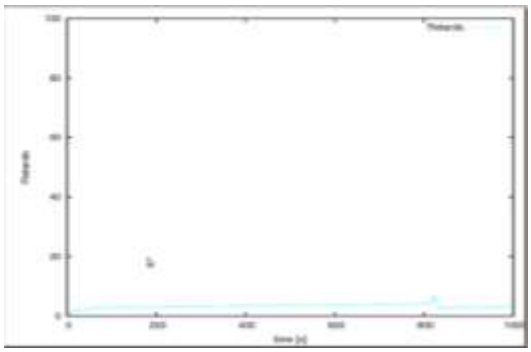


Figura 39.1 Retardo en la red con el uso de la política FODA.

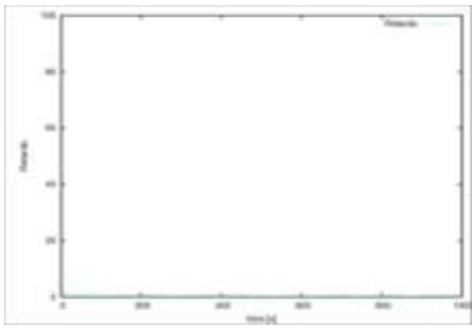


Figura 39.2 Retardo en la red con el uso de la política Proportional.

Es de resaltar que la asignación FODA presenta un bajo desempeño respecto a la utilización de recursos como se indica en la figura 38.

ANEXO F

SCRIPTS DE SIMULACIÓN Y ARCHIVOS GENERADOS POR EL SIMULADOR.

1.0. INSTALACIÓN DE LA HERRAMIENTA NS-2.31 BAJO UN ENTORNO LINUX.

1.1. INSTALACIÓN DE LA MAQUINA VIRTUAL Y SU SISTEMA OPERATIVO.

Para la instalación de la máquina virtual se utiliza el programa de Sun MircroSystems VirtualBox, el cual puede descargarse de la siguiente página **http://dlc.sun.com/virtualbox/vboxdownload.html#windows**, para este caso se seleccionó la plataforma de Windows (32-bit/64-bit), al tener ya instalada la maquina virtual se selecciona la pestaña Nueva en la maquina virtual y se escoge el tipo de sistema operativo a instalar (Linux) como su versión a instalar (Ubuntu). A continuación se configura los parámetros de memoria RAM, tamaño en el disco duro, entre otros, realizando los 7 pasos de configuración de la máquina, luego se da clic en la pestaña que dice Dispositivos y luego se monta el CD la versión de Linux a instalar.

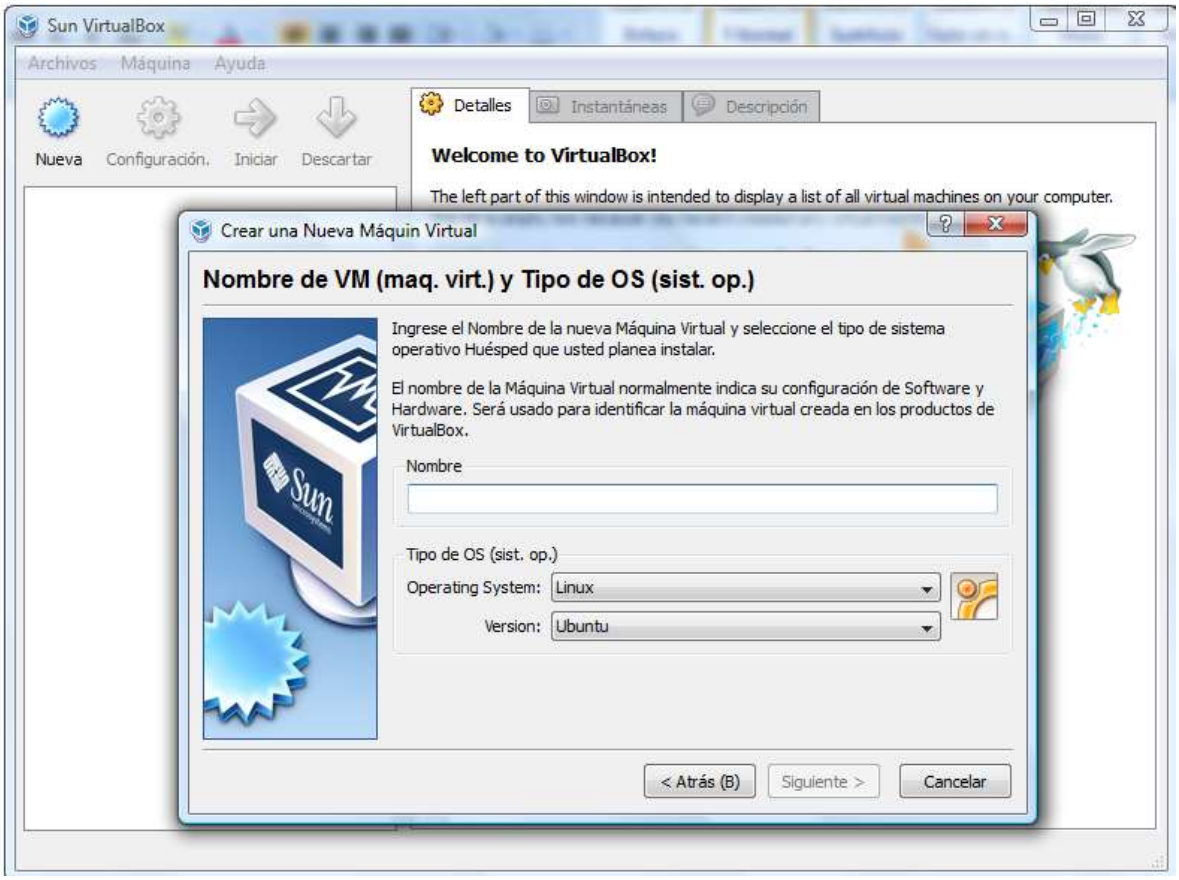


Figura 1. Creación de la máquina virtual.

1.2. INSTALACIÓN NETWORK SIMULATOR (NS-2.31).

Existen dos formas de instalar NS-2 por paquetes o “todo en uno”, la última forma es la más recomendable ya que incluyen todos los paquetes del simulador, para este caso se selecciono descargar todo el paquete.

1. Se descarga la versión NS-2.31 de la siguiente página mediante el comando:
$ wget http://nchc.dl.sourceforge.net/sourceforge/nsnam/ns-allinone-2.31.tar.gz.

2. Se descomprime el archivo según sea la ruta en el directorio donde se desee descomprimirlo.

$ tar -xzvf ns-allinone-2.31.tar.gz

3. Se abre la carpeta, mediante el siguiente comando cd ns-allinone-2.31, dentro de esa carpeta, se escribe en el terminal el siguiente comando:

$ sudo apt-get install build-essential autoconf automake libxmu-dev

4. Para la instalación se escribe en el terminal $./install

Después que termine la instalación debe presentar un mensaje como se muestra a continuación, en el que indica el estado de la instalación de los paquetes, además indica la información de la ruta donde se configurarán las variables de entorno.

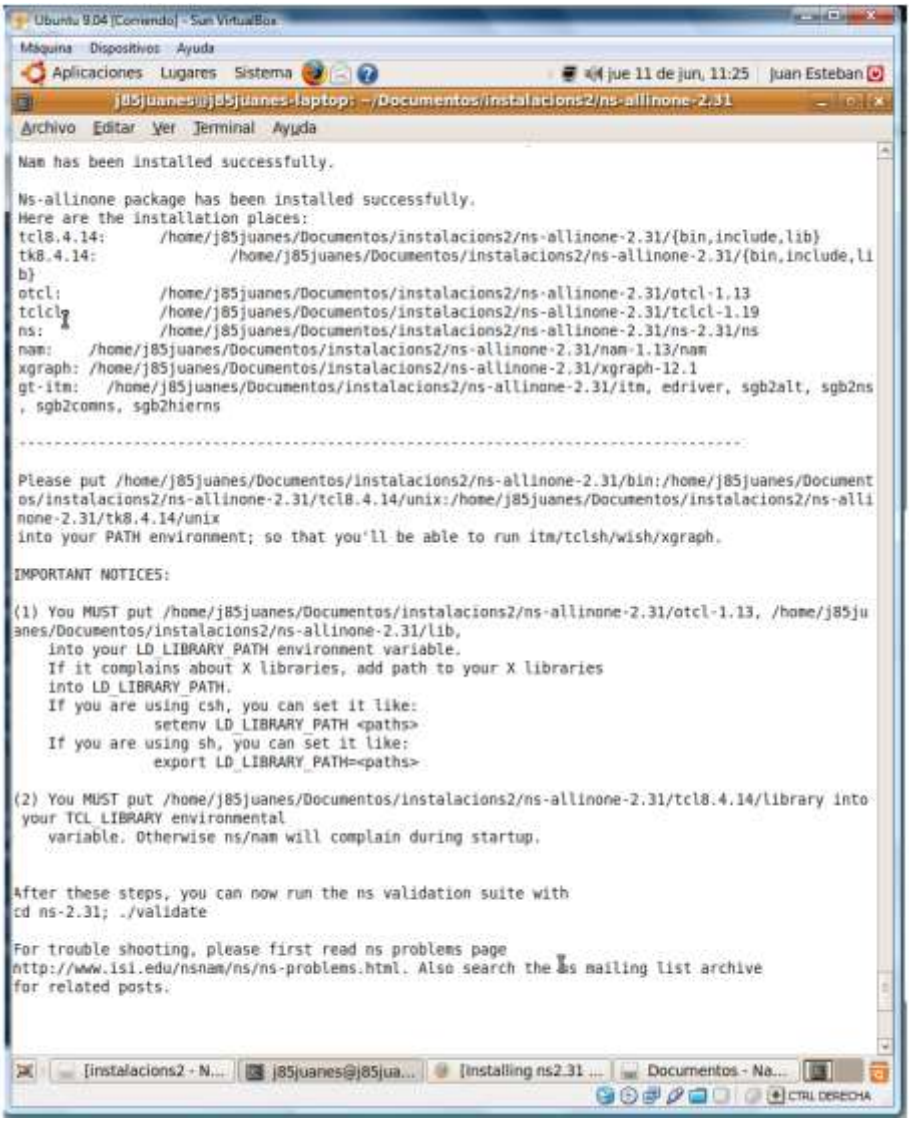


Figura 2. Finalización de la instalación de NS-2.31.

1.3. CONFIGURACIÓN DE LAS VARIABLES DE ENTORNO.

Se edita el archivo *.bashrc* mediante el siguiente comando

5. *$ gedit ~/.bashrc*

Luego se adiciona el siguiente código al final del archivo, donde se sustituye la ruta */your/patch/* por la ruta donde se haya instalado la herramienta Ns-2, en este caso sería /home/j85juanes/Documentos/instalacions2/.

```
# LD_LIBRARY_PATH
OTCL_LIB=/your/path/ns-allinone-2.31/otcl-1.13
NS2_LIB=/your/path/ns-allinone-2.31/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:
$USR_LOCAL_LIB

# TCL_LIBRARY
TCL_LIB=/your/path/ns-allinone-2.31/tcl8.4.14/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

# PATH
XGRAPH=/your/path/ns-allinone-2.31/bin:/your/path/ns-allinone-
2.31/tcl8.4.14/unix:/your/path/ns-allinone-2.31/tk8.4.14/unix
NS=/your/path/ns-allinone-2.31/ns-2.31/
NAM=/your/path/ns-allinone-2.31/nam-1.13/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

6. Se recarga el archivo *.bashrc* con el siguiente comando: ***:***

   $ source ~/.bashrc

7. Luego se reinicia el equipo, y después se ejecuta el comando**:**

   $ ns

1.4. VALIDACIÓN.

Para la validación se entra al directorio ns-2.31 y se coloca en consola *./validate*.

Para comprobar la instalación se escribe en consola *ns* y debe aparecer en esta el símbolo *%*, para salir se escribe exit, por último se escribe en consola *nam*, sin embargo al ejecutarlo se presenta el siguiente error:

```
nam:
[code omitted because of length]
: no event type or button # or keysym
    while executing
"bind Listbox <MouseWheel> {
%W yview scroll [expr {- (%D / 120) * 4}] units
}"
```

```
    invoked from within
"if {[tk windowingsystem] eq "classic" || [tk windowingsystem] eq "aqua"} {
bind Listbox <MouseWheel> {
%W yview scroll [expr {- (%D)}] units
}
bind Li..."
```

Para corregir este error se abre el archivo *tkBind.c* que se encuentra en la carpeta ns-allinone-2.31/tk8.4.14/generic, dentro de ese archivo buscar a la línea 586 y adicionar las siguientes líneas.

```
#ifdef GenericEvent
   /* GenericEvent */ 0,
   #endif
```

El archivo debería haber que dado de la siguiente manera.

```
   /* ColormapNotify */ COLORMAP,
   /* ClientMessage */ 0,
   /* MappingNotify */ 0,
   #ifdef GenericEvent
   /* GenericEvent */ 0,
   #endif
   /* VirtualEvent */ VIRTUAL,
   /* Activate */ ACTIVATE,
   /* Deactivate */ ACTIVATE,
   /* MouseWheel */ KEY
};
```

Luego se ubica en la carpeta ns-allillone-2.31y se ejecuta en el terminal *./install* para reconfigurar *ns* nuevamente, para verificar su funcionamiento se escribe nam y si este es correcto debería aparecer un mensaje similar al de la figura 3.

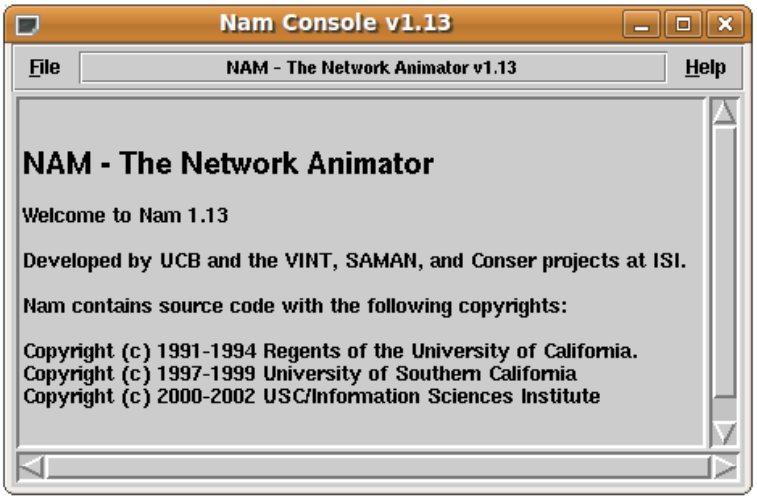


Figura 3. Nam 1.13.

1.5. INSTALACIÓN DEL PARCHE TDMA-DAMA VERSIÓN NS-2.33.

Para la instalación del parche TDMA-DAMA se realizó los siguientes pasos:

Se descarga el parche de la siguiente dirección: http://ala.isti.cnr.it/wnlab/tdmadama.

Se descomprime el archivo, mediante el siguiente comando:

tar -xzvf tdmadama-ns-2.33.tar

Luego se entra a la carpeta ns-2.31, que se encuentra en la ruta:
$ cd Documentos/instalation/ns-allinone-2.31/ns-2.31
Se copia la información del archivo defaults-tdmadama.tcl (que se encuentra en la carpeta descomprimida) para pegarla dentro del archivo *ns-sat.tcl*, este utlimo archivo se encuentra en la siguiente ruta*:*

$ cd Documentos/instalation/ns-allinone-2.31/ns-2.31/tcl/lib/

Se abre el archivo *ns-sat.tcl y* se busca lo siguiente "methods to add tracing support to Mac/Sat", en ese lugar se copia la información del archivo *defaults-tdmadama.tcl.*

Copiar los archivos *mac-tdmadama.cc* y *mac-tdmadama.h* para pegarlos en la siguiente ruta.

$ cd Documentos/instalation/ns-allinone-2.31/ns-2.31/ns2.31/mac/

Luego se edita el archivo *Makefile.in* que se encuentra en la ruta ns-2.31, se busca la palabra OBJ_CC y se escribe mac/mac-tdmadama.o \

En consola se escribe ./install, en la siguiente ruta.

$ cd Documentos/instalation/ns-allinone-2.31/

Dentro de la carpeta ns2.31, se ejecuta el comando *./configure* y por último *make*.

2.6 VERIFICACIÓN.

Para comprobar el correcto funcionamiento del parche se corre el ejemplo que trae por defecto el archivo de instalación mediante el comando *ns example-tdmadama.tcl bod* ó *ns example-tdmadama.tcl scpc*.
Al correr el parche se presenta un error, donde indica que el archivo *analy* no existe, para eliminarlo se realiza lo siguiente:
Prime que todo se descarga de esta página el archivo *analy.*

http://ala.isti.cnr.it/wnlab/start"

Luego se copia el archivo *analy* en esta ruta:
$ cd Documentos/instalation/ns-allinone-2.31/ns-2.31/ns2.31/mac/

En consola se compila este archivo mediante el siguiente comando

gcc analy.c -o analy

Por último se lo configura dentro de las variables de entorno, para ello se edita el archivo *~/.bashrc* y al final del código se adicciona lo siguiente.

ANALY=/home/juanesteban/Documentos/instalacions2/ns-allinone-2.31/ns-2.31/mac/
PATH=$PATH: $XGRAPH:$NS:$NAM:$ANALY

Luego se reconfigura el archivo, con el comando *$ source ~/.bashrc*.

Ahora, si se puede compilar el archivo y los archivos que generan deben ser iguales a los que viene por defecto el parche, si se presenta algún error descomentar la siguiente línea.

puts "mean trans.time [exec echo $function_ci | octave -q]"
EL archivo *analy* puede ser utilizado en cualquier tipo de redes, donde la información suministrada es muy útil para la evaluación del desempeño de la red.

2.0. SCRIPT BÁSICO DE CONFIGURACIÓN UNA RED SATELITAL.

A continuación se describe paso a paso como crear y configurar una red satelital mediante la herramienta de simulación NS-2, la figura 4 indica la arquitectura de red empleada.

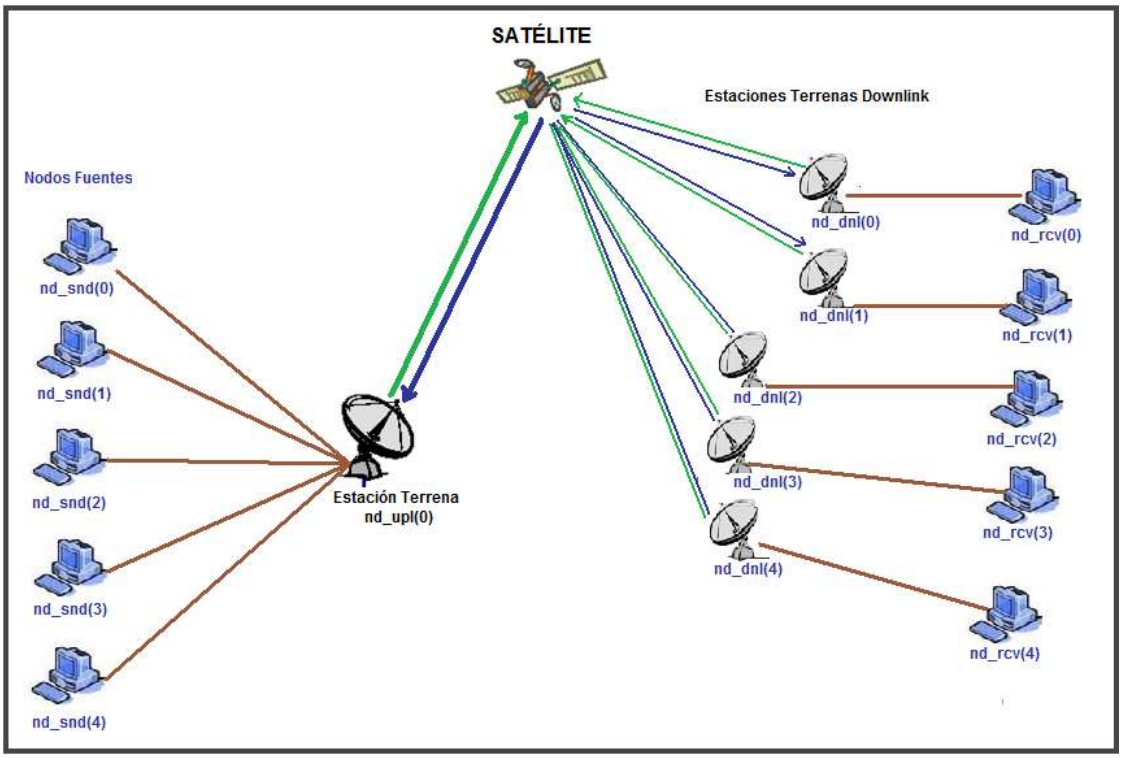


Figura 4. Arquitectura de red utilizada por el script ejemplo.

Como se observa en la figura 4 la red está compuesta por las siguientes partes:

- Cinco fuentes de tráfico llamadas nd_snd(i) dentro del código .tcl donde *i* varia de cero a cuatro.
- Seis estaciones terrenas o terminales, una para el enlace de subida nd_upl (0) y cinco para el enlace de bajada llamadas nd_dnl (i).
- Un nodo satelital denominado n1 el cual será el encargado de interconectar las estaciones terrenas.
- Cinco fuentes destino llamadas nd_rcv (i) los cuales son las encargadas de recibir todo el tráfico de los nodos fuentes.

A continuación se describe detalladamente la configuración del script para la creación de la red que se muestra en figura 4.

```
global ns
```

```
set ns [new Simulator]
set delayTER     10ms; # Se asigna el valor del retardo a la estación transmisora.
set delayRCST    10ms; # Se asigna el valor del retardo a la terminal RCST.
set capTER       100Mb
```

La primera línea indica la creación de una variable global llamada ns que instancia un objeto de la clase *simulator,* las demás líneas asignan valores de retardo y ancho de banda a las estaciones terrenas.

El paso siguiente es definir los parámetros globales del nodo satelital y las estaciones terrenas.

```
# Configuración de los parámetros globales del satélite.

set opt(chan)           Channel/Sat         # Canal satelital.
set opt(bw_up)          2Mb                 # Ancho de banda de 2Mb uplink.
set opt(bw_down)        2Mb                 # Ancho de banda de 2Mb downlink.
set opt(phy)            Phy/Sat             # Capa física satelital.
set opt(mac)            Mac/Sat             # Capa mac tipo de acceso al medio.
set opt(ifq)            Queue/DropTail      # Tipo de encolamiento.
set opt(qlim)           50                  # Tamaño de la cola.
set opt(ll)             LL/Sat              # Tipo de enlace satelital.
set opt(wiredRouting) ON                    # Enrutamiento activado.
```

Ahora se crea el archivo de traza denominado *outsat.tr*, el cual es el encargado de guardar todos los datos del comportamiento de la red, las siguientes líneas de comando lo definen.

```
set outfile [open outsat.tr w]
$ns trace-all $outfile
```

El siguiente paso es crear y configurar los parámetros de las estaciones terrenas y del nodo satelital haciendo uso de la variable *opt* que se definió anteriormente.

- Configuración del nodo satelital, este nodo se ajusta como un repetidor, para ello se utiliza el comando *phyType Phy/Repeater.*

```
$ns node-config -satNodeType geo-repeater \
                -phyType Phy/Repeater \
                -channelType $opt(chan) \
                -downlinkBW $opt(bw_down)  \
                -wiredRouting $opt(wiredRouting)
```

- Creación del nodo satelital *n1*, ubicado a -95 grados de longitud Oeste.

```
set n1 [$ns node]
$n1 set-position -95
```

- Configuración de las estaciones terrenas o nodos terminales de la red satelital.

```
$ns node-config -satNodeType terminal \
            -llType $opt(ll) \
            -ifqType $opt(ifq) \
            -ifqLen $opt(qlim) \
            -macType $opt(mac) \
```

```
-phyType $opt(phy) \
-channelType $opt(chan) \
-downlinkBW $opt(bw_down) \
-wiredRouting $opt(wiredRouting)
```

- Creación de las terminales o estaciones terrenas, con su respectiva posición.

```
#Creación de las estaciones terrenas uplink y downlink.
set nd_upl(0) [$ns node];              #Creación del nodo uplink.
$nd_upl(0) set-position 40.9 -73.9; #Coordenadas del nodo uplink.
set nd_dnl(0) [$ns node];              #Creación de la estación receptora 0.
$nd_dnl(0) set-position 37.8 -122.4
set nd_dnl(1) [$ns node]
$nd_dnl(1) set-position 34.8 -137.4;
set nd_dnl(2) [$ns node]
$nd_dnl(2) set-position 31.8 -152.4;
set nd_dnl(3) [$ns node]
$nd_dnl(3) set-position 28.8 -167.4;
set nd_dnl(4) [$ns node];      # Creación de la estación receptora 4.
$nd_dnl(4) set-position 55.8 -140.4;
```

Ahora se configura el enlace satelital entre las estación terrenas nd_upl(0) y el satélite para ello se agrega las siguientes líneas de comandos, cabe resaltar que el satélite funciona como únicamente en modo repetidor.

```
$nd_upl(0) add-gsl geo $opt(ll) $opt(ifq) $opt(qlim) $opt(mac) $opt(bw_up) \
        $opt(phy) [$n1 set downlink_] [$n1 set uplink_]
```

Configuración de los enlaces entre el satélite y las estaciones *nd_dnl($i)*, para ello se hace uso de un ciclo for.

```
set connections  5; # La variable connections define el número de enlaces a crear.

for {set i 0} {$i < $connections} {incr i} {
    $nd_dnl($i) add-gsl geo $opt(ll) $opt(ifq) $opt(qlim) $opt(mac) $opt(bw_up) \
        $opt(phy) [$n1 set downlink_] [$n1 set uplink_]
}
```

Ahora se agrega un modelo de error el cual simula la pérdida de paquetes dentro de cada enlace satelital. Para ello se agregan las siguientes líneas de código:

```
set em_ [new ErrorModel]
$em_ unit pkt
$em_ set rate_ 0.00002   #Probabilidad de pérdida.
$em_ ranvar [new RandomVariable/Uniform]
$nd_dnl($i) interface-errormodel $em_
```

Para esta topología de red se utilizó un enrutamiento centralizado que se adiciona con las siguientes líneas de comando.

```
set satrouteobject_ [new SatRouteObject]
$satrouteobject_ compute_routes
```

Ahora se crean los nodos destino *$nd_rcv($i)* y los nodos fuentes *nd_snd($i)* para ello se hace uso de un ciclo for para crear los respectivos enlaces; la línea *$ns unset satNodeType* sirve para borrar configuraciones anteriores de otros nodos (los satelitales) para no tener problemas con la configuración de los nodos terrestres; la manera de realizarlo es con las siguientes líneas de código:

```
$ns unset satNodeType_

#Enlaces entre la estación terrena uplink y los nodos fuentes.

for {set i 0} {$i < $connections} {incr i} {

        set nd_snd($i)[$ns node]      # Creación de los nodos fuentes.
        $ns duplex-link $nd_snd($i) $nd_upl(0) $capTER $delayTER DropTail
        # Tamaño máximo de la cola.
        $ns queue-limit $nd_snd($i) $nd_upl(0) 100000
}

#Enlaces entre las estaciones terrestres downlink y los nodos destinos.

for {set i 0} {$i < $connections} {incr i} {
        set nd_rcv($i) [$ns node]
        $ns duplex-link $nd_rcv($i) $nd_dnl($i) $capTER $delayRCST DropTail
}
```

Las variables *capTER* y *delayTER* son el ancho de banda y el tiempo de retardo respectivamente de cada enlace, de la misma manera la variable *delayRCST* es el tiempo de retardo de los enlaces entre las estaciones terrenas y los nodos fuentes.

```
        set record_f [open networkSat1.data w]

        # Trazas generadas para el enlace satelital
        $ns trace-all-satlinks $outfile
```

El paso siguiente es definir en cada nodo fuente y destino los agentes TCP, además se crea el agente de tráfico que para el caso es FTP y se lo incorpora a cada agente TCP. Para ello se hace uso de un ciclo for de la siguiente manera:

```
for {set i 0} {$i < $connections} {incr i} {

                #Agente TCP de los nodos fuentes.
                set ag_tcp_snd($i) [new Agent/TCP/Reno]
                #Agente TCP de los nodos destino.
                set ag_tcp_rcv($i) [new Agent/TCPSink]

                #Parámetros del agente TCP de los nodos fuente.
                $ag_tcp_snd($i) set window_             20
                $ag_tcp_snd($i) set packetSize_         536
                $ag_tcp_snd($i) set cwnd_               10
                $ag_tcp_snd($i) set ssthresh_           5
                #$ag_tcp_snd($i) set tcprexmtthresh_    3

                # Se adjunta cada agente TCP a su respectivo nodo.
                $ns attach-agent $nd_snd($i) $ag_tcp_snd($i)
                $ns attach-agent $nd_rcv($i) $ag_tcp_rcv($i)
```

```
        #Creación de los agentes FTP para luego ser adjuntados a los agentes TCP.
        set ftp($i) [new Application/FTP]
        $ftp($i) attach-agent $ag_tcp_snd($i)
        #Conexión del agente fuente con el agente destino.
        $ns connect $ag_tcp_snd($i) $ag_tcp_rcv($i)
}
```

A continuación se define un procedimiento *record* el cual es el encargado de ir grabando los datos en los archivos *.tr* para cada intervalo.

```
proc record {} {

#Creación de las variables globales
global ns ag_tcp_snd record_f

#En la línea siguiente, la variable 'l' indica el agente TCP que se quiere evaluar donde su número relaciona la conexión entre el nodo destino y la estación terrena.
set l 0

# Se establece el tiempo en que se estarán almacenando los datos.
set now [$ns now]
set time 0.1

#Se defienn los parámetros TCP a evaluar como los son el tamaño de ventana de congestión (cwnd), el ssthresh, la ventana ofrecida por la fuente (win), el número actual de secuencia transmitido (seq), el RTT, el backoff, los ACKs duplicados (dupACK), el RTT estimado (srtt), y la varianza del RTT (rttvar).

set cwin        [$ag_tcp_snd($l) set cwnd_]
set ssthresh    [$ag_tcp_snd($l) set ssthresh_]
set win         [$ag_tcp_snd($l) set window_]
set seq         [$ag_tcp_snd($l) set t_seqno_]
set rtt         [$ag_tcp_snd($l) set rtt_]
set srtt        [$ag_tcp_snd($l) set srtt_]
set rttvar      [$ag_tcp_snd($l) set rttvar_]
set backoff     [$ag_tcp_snd($l) set backoff_]
set dupACK      [$ag_tcp_snd($l) set dupacks_]
set ACK         [$ag_tcp_snd($l) set ack_]
#La línea siguiente almacena toda la información de los parámetros TCP.
puts $record_f "$now $cwin $ssthresh $win $seq $rtt $srtt $rttvar $backoff $dupACK $ACK"

# Se reprograma el procedimiento.
$ns at [expr $now+$time] "record "
}
```

Por último se crea un procedimiento de finalización llamado *proc finish {},* el cual es el encargado de cerrar los archivos trazas *(.tr)* y es donde se definen los comandos para graficar en Xgraph, el código es el siguiente:

```
proc finish {} {
        global ns outfile nf record_f
        $ns flush-trace
```

```
        # Se cierran lo ficheros utilizados.
        close $record_f
        close $outfile

        exec awk { { print $1, $2 } } networkSat1.data > t1.cwnd
        exec awk { { print $1, $3 } } networkSat1.data > t1.sst
        exec awk { { print $1, $4 } } networkSat1.data > t1.wnd
        exec awk { { print $1, $5 } } networkSat1.data > t1.bwe
        exec awk { { print $1, $6 } } networkSat1.data > t1.seq
        exec awk { { print $1, $7 } } networkSat1.data > t1.rtt
        exec awk { { print $1, $8 } } networkSat1.data > t1.srtt
        exec awk { { print $1, $9 } } networkSat1.data > t1.rttvar
        exec awk { { print $1, $10 } } networkSat1.data > t1.backoff
        exec awk { { print $1, $11 } } networkSat1.data > t1.dupACK
        exec awk { { print $1, $12 } } networkSat1.data > t1.ACK
        exec xgraph t1.cwnd t1.sst -m -x Tiempo -y Paquetes -geometry 600x200 &
        exec xgraph t1.wnd -m -x Tiempo -y Paquetes -geometry 600x200 &
        exec xgraph t 1.seq -m -x Tiempo -y Paquetes -geometry 600x200 &
        exec xgraph t1.rtt -m -x tiempo -y Paquetes -geometry 600x200 &
        exec xgraph t1.srtt -m -x tiempo -y Paquetes -geometry 600x200 &
        exec xgraph tt1.rttvar -m -x tiempo -y Paquetes -geometry 600x200 &
        exec xgraph tempsat1.backoff -m -x tiempo -y Paquetes -geometry 600x200 &
        exec xgraph tempsat1.dupACK -m -x tiempo -y Paquetes -geometry 600x200 &
        exec xgraph tempsat1.ACK -m -x tiempo -y Paquetes -geometry 600x200 &

 exit 0
}
```

La línea *exec awk { { print $1, $3 } } networkSat1.data > t1.sst* hace lo siguiente:

Mediante *awk* se extraen los valores que fueron almacenados en el archivo *networkSat1.data*, donde el número uno indica el tiempo en que se almacena los datos, el número tres el valor del *ssthresh*; estos datos se graban en otro archivo que para el caso es *t1.sst*.

La línea *exec xgraph t1.cwnd t1.sst -m -x Tiempo -y Paquetes -geometry 600x200 &* es la encargada de mostrar gráficamente los datos en pantalla, para realizarlo se utiliza los archivos que han sido extraídos previamente para ser graficados en el eje *x* como en el eje *y*, en este ejemplo son el comportamiento de *cwnd* y el *ssthresh*, el comando –m se utiliza para que las líneas salgan más gruesas y por último se indican las unidades de los ejes, que para el script son *tiempo* y *paquetes*.

En las siguientes líneas de comando se inicializan todos los procedimientos que se hayan definido en el script, como también los agentes FTP de cada conexión.

```
        $ns at 0.0 "record"
        $ns at 10.0 "$ftp(0) start"
        $ns at 10.0 "$ftp(1) start"
        $ns at 10.0 "$ftp(2) start"
        $ns at 10.0 "$ftp(3) start"
        $ns at 10.0 "$ftp(4) start"
        $ns at 100.0 "finish"
        $ns run
```

Para ejecutar el script se abre una terminal y se ejecuta la siguiente línea de comando *ns nombre_del_archivo.tcl*. Al ejecutarlo deben visualizarse nueve gráficas, una de ellas igual a la figura 5.

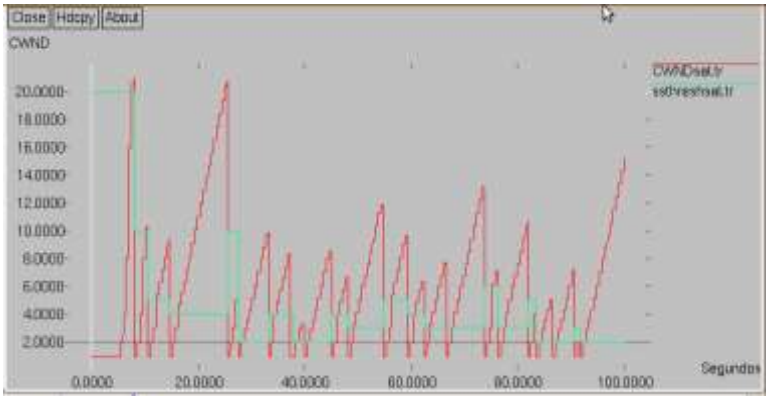


Figura 5. Comportamiento de *cwnd* y el *ssthresh* para el script base.

2.1. CONFIGURACIÓN DE LOS PARÁMETROS TCP Y DEL ACCESO AL MEDIO.

2.1.1. Protocolo de Transporte.

Para este escenario de simulación se definió como protocolo de transporte el protocolo TCP, las versiones a simular son TCP Highspeed, TCP Hybla y TCP Westwood, la manera como se define dentro del script tcl, es mediante la utilización de un procedimiento llamado *tcp_conn* el cual es el encargado de crear y conectar los agentes TCP y FTP, este procedimiento tiene unos parámetros de entrada que son:

Idx:	Parámetro que identifica el agente TCP trasmisor.
Src:	Parámetro que indica el nodo fuentes de tráfico.
dst:	Parámetro que indica el nodo destino, receptores de tráfico .
start:	Parámetro que indica el inicio de envió de tráfico.
stop:	Parámetro que indica la finalización de envío de tráfico.

La manera de crear un procedimiento es muy similar al que se describió en el ejemplo base.

```
proc create_tcp_conn { idx src dst start stop} {
        global ns rcst ncc tcp ftp sink data

        set time_start [$ns now]
        set tcp($idx) [new Agent/TCP/Linux] ;# Creación del agente TCP.
        $tcp($idx) set timestamps_ true
        $tcp($idx) set window_ 800          ;# Tamaño de la ventana de congestión
        $tcp($idx) set packetSize_ 1460       ;# Tamaño del segmento TCP.
        $ns at $time_start "$tcp($idx) select_ca hybla";# Versión de TCP.
        $tcp($idx) set fid_ $idx
        set sink($idx) [new Agent/TCPSink/Sack1]; #Creación agente sumidero.
        $ns attach-agent $dst $sink($idx);
        $ns attach-agent $src $tcp($idx);
        set ftp($idx) [new Application/FTP];
        $ftp($idx) attach-agent $tcp($idx);
        set satrouteobject_ [new SatRouteObject];
        $satrouteobject_ compute_routes;

        $ns connect $tcp($idx) $sink($idx)
```

```
        $ns at $start "$ftp($idx) start";
        $ns at $stop  "$ftp($idx) stop";
}
```

La manera de llamar a este procedimiento es a través de las siguientes líneas de código, donde se crean 5 conexiones entre el nodo transmisor y los nodos receptores:

```
create_tcp_conn 0 $wl(0) $hq(0) 0.01 $duration
create_tcp_conn 1 $wl(0) $hq(1) 0.01 $duration
create_tcp_conn 2 $wl(0) $hq(2) 0.01 $duration
create_tcp_conn 3 $wl(0) $hq(3) 0.01 $duration
create_tcp_conn 4 $wl(0) $hq(4) 0.01 $duration
```

2.1.2. Tdma-Dama.

Para los diferentes escenarios de simulación se tomaron varios tipos de acceso al medio. La manera como se define para el sistema TDMA es mediante las siguientes líneas de comandos:

Primero se definen unos parámetros generales los cuales se colocan al inicio del script tcl, antes de definir el objeto new simulator.

```
set frame_duration    0.04     # Duración de la trama en segundos.
set k_frame_          1.0
set rcst_buffer_      500      # Tamaño del búfer de la rcst.
set hub_buffer_       500      # Tamaño del búfer del hub

set slotsize_         184      # Tamaño del slot de tiempo en bytes.
set state_period      0.0417
set superframe_       25       # Tamaño de la supertrama.

Mac/TdmaDama set fixed_frame_ 0
```

Las 4 líneas de código que se muestran a continuación establecen los parámetros del agente asignador, donde utiliza una asignación por demanda ordenada tipo FIFO.

```
Allocator/FODA set ignore_loss_ 1
Allocator/FODA set delay_ 0.797
Allocator/FODA set def_slots_ 0
Allocator/FODA set xpercent_ 1.0
```

Las cinco líneas siguientes establecen los parámetros del agente solicitador, que para el caso se utiliza la política de asignación y petición de recursos RVBDC.

```
Requester/RVBDC set alpha_ 1.0
Requester/RVBDC set k_ 1.0
Requester/RVBDC set min_rvbdc_ 1.0 # Número mínimo de slots permitidos.
Requester/RVBDC set max_rvbdc_ 0

Queue/DropTail set drop_front_ true          # Tipo de encolamiento.
```

Configuración de parámetros dentro de la capa mac para tdma-dama.

```
# Número de slots por trama.
Mac/TdmaDama set max_slot_num_ $slotframe_
#Tamaño del slot en bytes.
Mac/TdmaDama set slot_packet_len_ $slotsize_
## Capacidad del enlace satelital.
Mac/TdmaDama set bandwidth_ $capSAT
# Número de tramas en una supertrama.
Mac/TdmaDama set num_frame_ $superframe_
Mac/TdmaDama set aligned_ 1

Allocator/FODA set k_frame_ $k_frame_
Allocator/FODA set maxg_ $slotframe_
Allocator/FODA set ming_ 1

set opt(mac)        Mac/TdmaDama
```

La última línea indica el tipo de acceso al medio que tiene la terminal hacia el satélite. El siguiente paso es configurar TDMA/DAMA de cada estación terrena receptora con sus propios parámetros, para hacer esto se agregan las siguientes líneas de comandos que puede ser después de la creación del modelo de error del ejemplo base que se describió.

```
# create node requester object
set streq       [$hub install-requester Requester/Constant]
set dtreq(0)    [$rcst(0) install-requester Requester/RVBDC]
set dtreq(1)    [$rcst(1) install-requester Requester/RVBDC]
set dtreq(2)    [$rcst(2) install-requester Requester/RVBDC]
set dtreq(3)    [$rcst(3) install-requester Requester/RVBDC]
set dtreq(4)    [$rcst(4) install-requester Requester/RVBDC]
```

Las líneas anteriores significan que después de haber definido el nodo satelital y las estaciones terrenas, estas deben tener instalados los objetos Allocator (asignador) y Requester (peticion); es importante tener en cuenta que el objeto Requester debe estar en cada terminal, mientras que el objeto Allocator sólo es necesario para una terminal, que generalmente se le asigna a la estación de control o sincronización, las terminales tienen la política de asignación de recursos VBDC [3].

Por último se define la estación de control NCC con su respectiva configuración, que para este escenario es una de las terminales transmisoras.

```
# Network Control Center
set ncc [$rcst(0) install-allocator Allocator/FODA]
```

2.1.3. Acceso al Medio con Aloha.

Para utilizar este tipo de acceso al medio se conserva el mismo código de este escenario, pero sin la parte de TDMA y se remplaza por las siguientes líneas de comando después de definir el objeto simulador set ns [new Simulator] con sus respectivos parámetros.

```
# Inicio del bacokff exponencial, tiempo en segundos.
Mac/Sat/UnslottedAloha set mean_backoff_ 0.5s
# Número máximo de intentos de retransmisión de ACKs.
Mac/Sat/UnslottedAloha set rtx_limit_ 3
# Tiempo de reenvió si no hay respuesta.
```

#Mac/Sat/UnslottedAloha set send_timeout_ 270ms;

Dentro de los parámetros de configuración general de debe modificar la siguiente línea set opt(mac) por la que se muestra a continuación:

set opt(mac) Mac/Sat/UnslottedAloha

3.0. SCRIPT DE CONFIGURACIÓN UTILIZADO PARA EVALUAR EL DESEMPEÑO DE LAS DISTINTAS POLÍTICAS DE ASIGNACIÓN DE DEMANDA.

```
set flavor TdmaDama
append scenario $flavor
file mkdir $scenario

set testing          1
set myseed           1
set duration       1000.0
set s_int           1.0

# Satellite Dependent

set frame_duration    0.05
set delayTER         10ms
set delayRCST        10ms
set capTER           10Mb
set cap              10
set capSAT           4812800
set k_frame_         1.0
set rcst_buffer_     500
set hub_buffer_      500
set slotsize_        1504
set state_period     0.0417
set superframe_      20

##### Radio Resource Management ##############

Mac/TdmaDama set fixed_frame_      0
Allocator/FODA set ignore_loss_    1
Allocator/FODA set delay_          0.797
Allocator/FODA set def_slots_      0
Allocator/FODA set xpercent_       1.0
Requester/RVBDC set alpha_         1.0
Requester/RVBDC set k_             1.0
Requester/RVBDC set min_rvbdc_     1.0
Requester/RVBDC set max_rvbdc_     0
Queue/DropTail set drop_front_     true

############### Traffic Flows ##############

# Data Flows
set data(dir)          upload
set data(next)         3.0
set data(bunch)        5
set data(packet)       1500

set record_f [open networkSat1.data w]
set record_q [open net.data w]
set record_l [open lnk.data w]
```

```
set record_T [open flowT.data w]

set h0 [open trafico0.tr w]
set h1 [open trafico1.tr w]
set h2 [open trafico2.tr w]
set h3 [open trafico3.tr w]
set h4 [open trafico4.tr w]

set connections  5

for {set i 0} {$i < $connections} {incr i} {
        set fl_tr_tcp_snd($i)           $scenario/tcp_snd_$i.tr
        set hn_tr_tcp_snd($i)           [open $fl_tr_tcp_snd($i) w]
}

##########  loading alternative configurations ############
set cmdline [lindex $argv 0]
if { $cmdline != "scpc" && $cmdline !="bod" } {
        puts "usage: ./test1.tcl \[bod|scpc\]"
        exit 1
}

ns-random $myseed
expr srand($myseed)

#################################################################

set slotframe_   [expr round(($capSAT*$frame_duration)/(8*$slotsize_))]
set data(load)   [expr $data(bunch)*$data(packet)*8/($capSAT*$data(next))]
puts "data(load) [format %.4f $data(load)]"
puts "slotframe_ [format %.4f $slotframe_]"

if { $cmdline == "scpc" } {
        Mac/TdmaDama set fixed_frame_ 1
        Mac/TdmaDama set slots1_ $slotframe_
        Mac/TdmaDama set offset_ -1
}

#################################################################33set connections  5 ;
## Número de conexiones ...

set traced_vars [list cwnd_ ssthresh_ ack_ t_seqno_ rtt_ seqno_]

proc create_tcp_conn { idx src dst start stop} {
        global ns rcst ncc tcp ftp sink data
        global traced_vars hn_tr_tcp_snd

        set time_start [$ns now]

        set tcp($idx) [new Agent/TCP/Linux]
        $tcp($idx) set window_ 800
        $tcp($idx) set packetSize_ 1420
        $tcp($idx) set timestamps_ true
        $ns at $time_start "$tcp($idx) select_ca highspeed"
        set sink($idx) [new Agent/TCPSink/Sack1]

        $ns attach-agent $dst $sink($idx);
        $ns attach-agent $src $tcp($idx);

        #$tcp($idx) set class_ [expr $idx + 1]
```

```
        $tcp($idx) set fid_ $idx

        #FTP application is used to generate TCP traffic
        set ftp($idx) [new Application/FTP];
        $ftp($idx) attach-agent $tcp($idx);

        set satrouteobject_ [new SatRouteObject];
        $satrouteobject_ compute_routes;

        $ns connect $tcp($idx) $sink($idx)

        $ns at $start "$ftp($idx) start";
        $ns at $stop  "$ftp($idx) stop";

        $tcp($idx) attach-trace $hn_tr_tcp_snd($idx)

        for {set j 0} {$j < [llength $traced_vars]} {incr j} {
                        $tcp($idx) trace [lindex $traced_vars $j]
        }
}


Mac/TdmaDama set max_slot_num_ $slotframe_
Mac/TdmaDama set slot_packet_len_ $slotsize_
Mac/TdmaDama set bandwidth_ $capSAT
Mac/TdmaDama set num_frame_ $superframe_
Mac/TdmaDama set aligned_ 1

Allocator/FODA set k_frame_ $k_frame_
Allocator/FODA set maxg_ $slotframe_
Allocator/FODA set ming_ 1

# Creating scenario  ############################

set ns [new Simulator]
set stats_con [open stats.tr w]
if { $testing == 1 } {
        # Tracing enabling must
        # precede link and node creation
        set f0 [open out.tr w]
        $ns trace-all $f0
}

# nodes

set opt(chan)           Channel/Sat
set opt(bw_up)          $capSAT;# uplink bandwidth
set opt(bw_down)        $capSAT# downlink bandwidth
set opt(phy)            Phy/Sat
set opt(mac)            Mac/TdmaDama
set opt(ifq)            Queue/DropTail
set opt(qlim)           230;# queue size (pkts)
set opt(ll)             LL/Sat
set opt(wiredRouting)   ON

# Configure bent-pipe satellite
$ns node-config -satNodeType geo-repeater \
                -llType $opt(ll) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(qlim) \
```

```
                    -macType $opt(mac) \
                    -phyType Phy/Repeater \
                    -channelType $opt(chan) \
                    -downlinkBW $opt(bw_down)  \
                    -wiredRouting $opt(wiredRouting)

# GEO satellite at 13 degrees longitude East (Hotbird 6)
set sat [$ns node]

$sat set-position 13

$ns node-config -satNodeType terminal \
                -llType $opt(ll) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(qlim) \
                    -macType $opt(mac) \
                    -phyType $opt(phy) \
                    -channelType $opt(chan) \
                    -downlinkBW $opt(bw_down) \
                    -wiredRouting $opt(wiredRouting)

set hub [$ns node];             #HUB
$hub set-position 40.9 -73.9; # NY


set rcst(0) [$ns node]
$rcst(0) set-position 37.8 -122.4; # RCST 0
set rcst(1) [$ns node]
$rcst(1) set-position 34.8 -137.4; # RCST 1
set rcst(2) [$ns node]
$rcst(2) set-position 31.8 -152.4; # RCST 2
set rcst(3) [$ns node]
$rcst(3) set-position 28.8 -167.4; # RCST 3
set rcst(4) [$ns node]
$rcst(4) set-position 55.8 -140.4; # RCST 4

$hub add-gsl geo $opt(ll) $opt(ifq) $hub_buffer_ $opt(mac) \
          $opt(bw_up) $opt(phy) [$sat set downlink_] [$sat set uplink_]

for {set i 0} {$i < $connections} {incr i} {

      $rcst($i) add-gsl geo $opt(ll) $opt(ifq) $rcst_buffer_ $opt(mac) \
          $opt(bw_up) $opt(phy) [$sat set downlink_] [$sat set uplink_]
}

set em_ [new ErrorModel]
$em_ unit pkt
$em_ set rate_ 0.00002
$em_ ranvar [new RandomVariable/Uniform]
$hub interface-errormodel $em_

set gat_mac [$hub set mac_(0)]
set ter_mac(0) [$rcst(0) set mac_(0)]
set ter_mac(1) [$rcst(1) set mac_(0)]
set ter_mac(2) [$rcst(2) set mac_(0)]
set ter_mac(3) [$rcst(3) set mac_(0)]
set ter_mac(4) [$rcst(4) set mac_(0)]

$gat_mac set hub_ 1
```

```
if { $testing == 1 } {
        set ev_file [open event.tr w]
        $hub trace-event $ev_file
        $rcst(0) trace-event $ev_file
        $rcst(1) trace-event $ev_file
        $rcst(2) trace-event $ev_file
        $rcst(3) trace-event $ev_file
        $rcst(4) trace-event $ev_file
        $ns trace-all-satlinks $f0
}

# create node requester object
set streq [$hub install-requester Requester/Constant]
set dtreq(0) [$rcst(0) install-requester Requester/VBDC]
set dtreq(1) [$rcst(1) install-requester Requester/VBDC]
set dtreq(2) [$rcst(2) install-requester Requester/VBDC]
set dtreq(3) [$rcst(3) install-requester Requester/VBDC]
set dtreq(4) [$rcst(4) install-requester Requester/VBDC]


# Network Control Center
set ncc [$rcst(0) install-allocator Allocator/FODA]

# We use centralized routing
set satrouteobject_ [new SatRouteObject]
$satrouteobject_ compute_routes


#Se crean los nodos y sus respectivos enlaces

set nd_rcv(0)  [$ns node]
$ns duplex-link $nd_rcv(0) $hub $capTER $delayTER DropTail
$ns queue-limit $nd_rcv(0) $hub 12


for {set i 0} {$i < $connections} {incr i} {
        set nd_snd($i)    [$ns node]
        $ns duplex-link $nd_snd($i) $rcst($i) $capTER $delayTER DropTail
        $ns queue-limit $nd_snd($i) $rcst($i) 9

}

set qmon(0) [$ns monitor-queue $hub $nd_rcv(0) ""]


set fmon(0) [$ns makeflowmon Fid]
$ns attach-fmon [$ns link $hub $nd_rcv(0)] $fmon(0)

set start 1.0

### Drawing diagrams

Agent/TCP/Sack1 instproc done {} {
        global ns stats_con
        set init [$self set time_start]
        set dur [expr [$ns now] - $init]
        puts $stats_con [format "%10.2f %8.4f" $init $dur]

}
```

```
set function_ci {
	function A = autocorr(x)
		A = x - mean(x);
		A = fftconv(A, rot90(A,2));
		A = A(length(x) : 2*length(x)-1);
		A /= length(x);
		A /= A(1);
	end

	function t = corrlen(x)

	 if (rows(x)==1)
	   x = x';
	 endif
	 [l,n] = size(x);
	 weights = 1 - [0.5, [1:l-1]/(l-1)];
	 clen = zeros(1,n);
	 for i=1:n
	   clen(i) = weights * autocorr(x(:,i));
	 endfor
	 t = max(1,clen);

	end

	input = load("stats.tr");
	input = input(:,2);

	step = ceil(corrlen(input));
	data = input(1:step:length(input));

	confidence = 0.95;

	m = mean(data);
	s = std(data);
	p = abs(tinv( (1-confidence)/2 , length(data) - 1));
	p = p * s / sqrt(length(data));

	ci = [m ; p];

	h = reshape(ci, 1, size(ci)(1) * size(ci)(2));

	printf("%10.5g %6.4f",h);
	printf("\n");
}

set plot_res1 {
	set xlabel "Tiempo [s]"
	set ylabel "Ancho de Banda [b/s]"

	set xrange [0:1000]
	set yrange [0:4812800]

	set term postscript color solid eps
	set output "result1.eps"

	plot "q_up.dat" u 1:4 t 'cap usada.' w l 3, \
	    "curall.tr" u (floor($2)):($5*4812800/20) smooth unique \
	    t 'cap asignada.' w l 1
}
```

```
set plot_res2 {
        set xlabel "Tiempo [s]"
        set ylabel "Encolamiento de paquetes [b/s]"

        set xrange [0:1000]
        set yrange [0:600]

        set term postscript color solid eps
        set output "Encolamiento.eps"

        plot "q_up.dat" u 1:7 t 'Encolamiento.' w l 4

}

set plot_res3 {
        set xlabel "Tiempo [s]"
        set ylabel "Retardo"

        set xrange [0:1000]
        set yrange [0:10]

        set term postscript color solid eps
        set output "Retardo.eps"

        plot "q_up.dat" u 1:10 t 'Retardo.' w l 5

}


set plot_ReqAll {
        set xlabel "Tiempo [s]"
        set ylabel "Allocator"

        set xrange [0:1000]
        set yrange [0:25]

        set term postscript color solid eps
        set output "ReqAll.eps"

        plot "curall.tr" u (floor($2)):5 t 'Slots asignados.' w l 5

}

set plot_Req {
        set xlabel "Tiempo [s]"
        set ylabel "Requester"

        set xrange [0:1000]
        set yrange [0:30]

        set term postscript color solid eps
        set output "Req.eps"

        plot "reques.tr" u (floor($2)):5 t 'Slots solicitados.' w l 1
#
}

# Se define un procedimiento 'record'

proc record {} {
```

```
        global ns tcp record_f
        set l 0
        set now [$ns now]
        set time 0.1
        set cwin        [$tcp($l) set cwnd_]
        set ssthresh    [$tcp($l) set ssthresh_]
        set ndatapack   [$tcp($l) set ndatapack_]
        set nrexmit     [$tcp($l) set nrexmit_]
        set seq         [$tcp($l) set t_seqno_]
        set rtt         [$tcp($l) set rtt_]
        set srtt        [$tcp($l) set srtt_]
        set rttvar      [$tcp($l) set rttvar_]
        set backoff     [$tcp($l) set backoff_]
        set pktx        [$tcp($l) set nrexmitpack_]
        set ACK         [$tcp($l) set ack_]

puts $record_f "$now [expr $cwin*1] $ssthresh $ndatapack $nrexmit $seq $rtt $srtt $rttvar
$backoff $pktx $ACK"

$ns at [expr $now+$time] "record "
}

proc record1 {} {
      global sink h0 h1 h2 h3 h4
        global opt bw_down ;
        set ns [Simulator instance]
        set time 1.0

         set bw0 [$sink(0) set bytes_]
        set bw1 [$sink(1) set bytes_]
        set bw2 [$sink(2) set bytes_]
        set bw3 [$sink(3) set bytes_]
        set bw4 [$sink(4) set bytes_]

        #Get the current time
        set now [$ns now]
        #Calculate the bandwidth (in MBit/s) and write it to the files

        puts $h0 "$now [expr $bw0/$now*8]"
        puts $h1 "$now [expr $bw1/$now*8]"
        puts $h2 "$now [expr $bw2/$now*8]"
        puts $h3 "$now [expr $bw3/$now*8]"
        puts $h4 "$now [expr $bw4/$now*8]"

        #Reset the bytes_ values on the traffic sinks
        $sink(0) set bytes_ 0
        $sink(1) set bytes_ 0
        $sink(2) set bytes_ 0
        $sink(3) set bytes_ 0
        $sink(4) set bytes_ 0

        #Re-schedule the procedure
        $ns at [expr $now+$time] "record1"
}

proc recordqueue {} {

    global qmon ns record_q

        set now [$ns now]
```

```
            set time 0.1
            set size [$qmon(0) set size_]
            set pkts [$qmon(0) set pkts_]
            set parrivals [$qmon(0) set parrivals_]
            set barrivals [$qmon(0) set barrivals_]
            set pdepartures [$qmon(0) set pdepartures_]
            set bdepartures [$qmon(0) set bdepartures_]
            set pdrops [$qmon(0) set pdrops_]
            set bdrops [$qmon(0) set bdrops_]
            set pmarks [$qmon(0) set pmarks_]

             puts $record_q "[format %.2f [$ns now]] $size $pkts $parrivals $barrivals
        $pdepartures $bdepartures $pdrops $bdrops $pmarks"

        $ns at [expr $now+$time] "recordqueue"
        }

        proc reclink { } {

        global fmon cap ns record_l

           set now [format "%.1f" [$ns now]]
           set now [$ns now]
           set time 0.1

           set  bytes              [$fmon(0) set bdepartures_]
           set  bytesDbl           [ns-int64todbl $bytes]
           set  Kbytes             [expr $bytesDbl / 1000 ]
           set  bandwidthToKBytes [expr 125* [$ns now] ];#  es 1000 * $time / 8, but time=30
           set  possibleKBytes  [format %.2f [expr $cap* $bandwidthToKBytes]]
           set  drops_t            [$fmon(0) set pdrops_]
           set  marks_t            [$fmon(0) set pmarks_]
           set  packets_t          [$fmon(0) set pdepartures_]
           set  tlu_t              [format %.2f [expr 100* $Kbytes / $possibleKBytes ]](%)

           puts $record_l "[format %.2f [$ns now]] $drops_t $marks_t $packets_t $Kbytes
        $possibleKBytes $tlu_t"
           $ns at [expr $now+$time] "reclink"
        }

proc recflow { } {
         global ns cap connections qmon record_T fmon

          set now [format "%.1f" [$ns now]]
          set now [$ns now]
          set time 0.1
          # calculus of possible bytes to sent
          set bandwidthToKBytes [expr 1000 * [$ns now] / 8 ];### 125
          set possibleKBytes [expr $cap * $bandwidthToKBytes ]
          set auxH 0
          set linkutilzH     0
          set TOTparrivalsH  0
          set TOTpdropsH     0


          # HSTCP Flows
          for {set fid 0} {$fid < 1} {incr fid} {
              # extract data for this particular flow
              set fcl [$fmon(0) classifier]; # flow classifier
              set flow [$fcl lookup auto 0 0 $fid]
```

```
        if {$flow != "" } {
           set bytes [$flow set bdepartures_]
           set bytesDbl [ns-int64todbl $bytes]
           set Kbytes [expr $bytesDbl / 1000 ]

              set auxH [expr $linkutilzH + [expr $Kbytes * 100 / $possibleKBytes] ]
           set linkutilzH $auxH
              set auxH [ns-int64todbl [expr $TOTpdropsH + [$flow set pdrops_] ] ]
              set TOTpdropsH [ns-int64todbl $auxH]

              set auxH [ns-int64todbl [expr $TOTparrivalsH + [$flow set parrivals_] ] ]
              set TOTparrivalsH [ns-int64todbl $auxH]
           }
        }

            if { $connections != 0 } {
                    set linkutilzavgH [expr $linkutilzH / $connections]
                }

          if {$TOTparrivalsH != 0} {
                    set lossrateH [expr $TOTpdropsH/$TOTparrivalsH]
                } else {
                     set lossrateH 0.0
                }


       puts $record_T "[format %.2f [$ns now]] $linkutilzH $linkutilzavgH $lossrateH"

$ns at [expr $now+$time] "recflow"

}

create_tcp_conn 0 $nd_snd(0) $nd_rcv(0) 0.01 $duration
create_tcp_conn 1 $nd_snd(1) $nd_rcv(0) 0.01 $duration
create_tcp_conn 2 $nd_snd(2) $nd_rcv(0) 0.01 $duration
create_tcp_conn 3 $nd_snd(3) $nd_rcv(0) 0.01 $duration
create_tcp_conn 4 $nd_snd(4) $nd_rcv(0) 0.01 $duration

proc finish-sim {} {
          global ns rcst hub stats_con cmdline
          global function_ci plot_res1
          global plot_ReqAll plot_Req
          global plot_res2 plot_res3

          global ns record_f h0 h1 h2 h3 h4 record_q record_l record_T
          global hn_tr_tcp_snd connections
          $ns flush-trace
          $ns halt

          close $record_l
          close $record_q
          close $record_f
          close $record_T
          close $h0
          close $h1
          close $h2
          close $h3
          close $h4
          close $stats_con
          for {set i 0} {$i < $connections} {incr i} {
```

```
            close $hn_tr_tcp_snd($i)

        }

        exec grep " [$rcst(0) id] [$hub id] " out.tr | \
        cut -d " " -f 1,2,3,4,5,6,7,8,9,10,11,12 > q_up

        exec analy q_up 1.0
        exec grep curall event.tr > curall.tr
        exec grep reques event.tr > reques.tr

        exec echo $plot_ReqAll | gnuplot
        exec echo $plot_Req | gnuplot
        exec echo $plot_res1 | gnuplot
        exec echo $plot_res2 | gnuplot
        exec echo $plot_res3 | gnuplot
        exec mv result1.eps result1-${cmdline}.eps
        exec mv stats.tr stats-test1-${cmdline}.tr
        exec mv q_up.dat q_up-test1-${cmdline}.dat
        exec mv out.tr out-test1-${cmdline}.tr
        exec mv event.tr event-test1-${cmdline}.tr
        #exec rm q_up reques.tr curall.tr; # procedimi var requester


        exec perl throughput.pl out-test1-bod.tr 7 1 > ThrputS2
        #exec xgraph ThrputS2 -t "   Throughput para Slot2    " -m -x Tiempo -y #bits/seg -
        geometry 800x400 &


        exec awk { { print $1, $2 } } networkSat1.data > cwndS2
        exec awk { { print $1, $3 } } networkSat1.data > sstS2
        exec awk { { print $1, $4 } } networkSat1.data > pkTxS2
        exec awk { { print $1, $5 } } networkSat1.data > nrexmitS2
        exec awk { { print $1, $6 } } networkSat1.data > seqS2
        exec awk { { print $1, $7 } } networkSat1.data > rttS2
        exec awk { { print $1, $8 } } networkSat1.data > srttS2
        exec awk { { print $1, $9 } } networkSat1.data > rttvarS2
        exec awk { { print $1, $10 } } networkSat1.data > backoffS2
        exec awk { { print $1, $11 } } networkSat1.data > pkRTxS2
        exec awk { { print $1, $12 } } networkSat1.data > ACKS2

exec xgraph cwndS2 sstS2 -t  "Ventana Congestion y Ssthresh " -m -x Tiempo -y Segmentos -
geometry 800x400 &
exec xgraph pkTxS2 -t "Paquetes Transmitidos" -m -x Tiempo -y Segmentos -geometry 800x400 &
exec xgraph pkRTxS2 -t "Paquetes Retransmitidos" -m -x Tiempo -y Segmentos -geometry 800x400 &
exec xgraph seqS2 -t "Numero de Secuencia" -m -x Tiempo -y Segmentos -geometry 800x400 &
exec xgraph backoffS2 nrexmitS2 -t  "Back-off y Numero de Retransmisiones del Timeout" -m -x
Tiempo -y ValorxDefecto -geometry 800x400 &
exec xgraph rttS2 -t "  RTT    " -m -x Tiempo -y Tiempo -geometry 800x400 &
exec xgraph ACKS2 -t "    ACK      " -m -x Tiempo -y Segmentos -geometry 800x400 &


#ok      exec xgraph trafico0.tr Grafica14 -geometry 800x400 &
#ok      exec xgraph trafico1.tr trafico2.tr -geometry 800x400 &
#ok      exec xgraph trafico3.tr trafico4.tr -geometry 800x400 &

exec awk { { print $1, $2 } } net.data > queue.size
exec xgraph queue.size -m -x tiempo -y Grafica16 -geometry 800x400 &

exec awk { { print $1, $7 } } lnk.data > flowTS2 &
```

```
exec awk { { print $1, $2 } } flowT.data > flow0S2
exec awk { { print $1, $3 } } flowT.data > flow.linkutilzavgH

exec xgraph flowTS2 flow0S2 -t  " Capacidad Total y Flujo de la conexion cero " -m -x Tiempo -y Utilizacion% -geometry 800x400 &

        exit 0
}


#$ns at 1.0 "new-data"
$ns at 1.0 "record"
$ns at 1.1 "reclink"
$ns at 1.1 "recflow"
$ns at 1.1 "recordqueue"
$ns at 1.1 "record1"
$ns at $duration "finish-sim"

$ns run
```

4.0. SCRIPT DE CONFIGURACIÓN UTILIZADO PARA EVALUAR EL DESEMPEÑO DE ALOHA.

```
set flavor Aloha
append scenario $flavor
file mkdir $scenario

global ns
set ns [new Simulator]

set delayTER        10ms
set capTER          10Mb

set delayRCST       10ms


Mac/Sat/UnslottedAloha set mean_backoff_ 1.0s; # mean exponential backoff time(s)
Mac/Sat/UnslottedAloha set rtx_limit_ 3; # numero maximo de intentos de retransmision

set connections  5

for {set i 0} {$i < $connections} {incr i} {
        set fl_tr_tcp_snd($i)           $scenario/tcp_snd_$i.tr
        set hn_tr_tcp_snd($i)           [open $fl_tr_tcp_snd($i) w]
}

# Global configuration parameters
# We'll set these global options for the satellite terminals

set record_f [open networkSat1.data w]
set h0 [open trafico0.tr w]
set h1 [open trafico1.tr w]
set h2 [open trafico2.tr w]
set h3 [open trafico3.tr w]
set h4 [open trafico4.tr w]
set h5 [open traficoTot w]

global opt
```

```
set opt(chan)           Channel/Sat
set opt(bw_up)          2Mb
set opt(bw_down)        2Mb
set opt(phy)            Phy/Sat
set opt(mac)            Mac/Sat/UnslottedAloha
set opt(ifq)            Queue/DropTail
set opt(qlim)           230
set opt(ll)             LL/Sat
set opt(wiredRouting)   ON

set outfile [open out.tr w]
$ns trace-all $outfile

# Set up satellite and terrestrial nodes

# Configure the node generator for bent-pipe satellite
# geo-repeater uses type Phy/Repeater
$ns node-config -satNodeType geo-repeater \
                -phyType Phy/Repeater \
                -channelType $opt(chan) \
                -downlinkBW $opt(bw_down)  \
                -wiredRouting $opt(wiredRouting)

# GEO satellite at 95 degrees longitude West
set n1 [$ns node]
$n1 set-position -95

# Configure the node generator for satellite terminals
$ns node-config -satNodeType terminal \
          -llType $opt(ll) \
          -ifqType $opt(ifq) \
          -ifqLen $opt(qlim) \
          -macType $opt(mac) \
          -phyType $opt(phy) \
          -channelType $opt(chan) \
          -downlinkBW $opt(bw_down) \
          -wiredRouting $opt(wiredRouting)

# Three terminals: one in NY and one in SF
set nd_upl(0) [$ns node]
$nd_upl(0) set-position 40.9 -73.9; # NY
set nd_upl(1) [$ns node]
$nd_upl(1) set-position 50.9 -83.9; # NY
set nd_upl(2) [$ns node]
$nd_upl(2) set-position 60.9 -93.9; # NY
set nd_upl(3) [$ns node]
$nd_upl(3) set-position 70.9 -103.9; # NY
set nd_upl(4) [$ns node]
$nd_upl(4) set-position 89.9 -113.9;

set nd_dnl(0) [$ns node]
$nd_dnl(0) set-position 37.8 -122.4; # RCST 1
set nd_dnl(1) [$ns node]
$nd_dnl(1) set-position 34.8 -137.4; # RCST 2
set nd_dnl(2) [$ns node]
$nd_dnl(2) set-position 31.8 -152.4; # RCST 3
set nd_dnl(3) [$ns node]
$nd_dnl(3) set-position 28.8 -167.4; # RCST 4
set nd_dnl(4) [$ns node]
$nd_dnl(4) set-position 25.8 -107.4; # RCST 5
```

```
# Add GSLs to geo satellites
set connections  5

for {set i 0} {$i < $connections} {incr i} {
      $nd_upl($i) add-gsl geo $opt(ll) $opt(ifq) $opt(qlim) $opt(mac) $opt(bw_up) \
            $opt(phy) [$n1 set downlink_] [$n1 set uplink_]
}

for {set i 0} {$i < $connections} {incr i} {
      $nd_dnl($i) add-gsl geo $opt(ll) $opt(ifq) $opt(qlim) $opt(mac) $opt(bw_up) \
            $opt(phy) [$n1 set downlink_] [$n1 set uplink_]

    set em_ [new ErrorModel]
    $em_ unit pkt
    $em_ set rate_ 0.000002
    $em_ ranvar [new RandomVariable/Uniform]
    $nd_dnl($i) interface-errormodel $em_

}


# We use centralized routing
set satrouteobject_ [new SatRouteObject]
$satrouteobject_ compute_routes
#ojo $satrouteobject_ set wiredRouting_ true

#---------------------------------------------
$ns unset satNodeType_

for {set i 0} {$i < $connections} {incr i} {
          set nd_snd($i)   [$ns node]
          $ns duplex-link $nd_snd($i) $nd_upl($i) $capTER $delayTER DropTail
          $ns queue-limit $nd_snd($i) $nd_upl($i) 12
}

for {set i 0} {$i < $connections} {incr i} {
          set nd_rcv($i) [$ns node]
          $ns duplex-link $nd_rcv($i) $nd_dnl($i) $capTER $delayRCST DropTail
          $ns queue-limit $nd_snd($i) $nd_upl($i) 12
}


#---------------------------------------------

# Trace all queues
$ns trace-all-satlinks $outfile

for {set i 0} {$i < $connections} {incr i} {

                    set ag_tcp_snd($i) [new Agent/TCP/Linux]
                    $ag_tcp_snd($i) set timestamps_ true
                    $ag_tcp_snd($i) set window_ 500
                    $ag_tcp_snd($i) set packetSize_ 1400
                    $ns at 1 "$ag_tcp_snd($i) select_ca westwood"
                    $ns attach-agent $nd_snd($i) $ag_tcp_snd($i)

                    $ag_tcp_snd($i) set fid_ $i
                    set ag_tcp_rcv($i) [new Agent/TCPSink/Sack1]
                    #$ag_tcp_rcv($i) set ts_echo_rfc1323_ true
```

```
                $ns attach-agent $nd_rcv($i) $ag_tcp_rcv($i)
                set ftp($i) [new Application/FTP]
                $ftp($i) attach-agent $ag_tcp_snd($i)
                $ns connect $ag_tcp_snd($i) $ag_tcp_rcv($i)
}


proc record {} {
        global ns ag_tcp_snd record_f
        set l 0
        set pktenviados 0
        set now         [$ns now]
        set time        49; # set time 49
        set cwin        [$ag_tcp_snd($l) set cwnd_]
        set ssthresh    [$ag_tcp_snd($l) set ssthresh_]
        set ndatapack   [$ag_tcp_snd($l) set ndatapack_]
        set nrexmit     [$ag_tcp_snd($l) set nrexmit_]
        set seq         [$ag_tcp_snd($l) set t_seqno_]
        set rtt         [$ag_tcp_snd($l) set rtt_]
        set srtt        [$ag_tcp_snd($l) set srtt_]
        set rttvar      [$ag_tcp_snd($l) set rttvar_]
        set backoff     [$ag_tcp_snd($l) set backoff_]
        set pktx        [$ag_tcp_snd($l) set nrexmitpack_]
        set ACK         [$ag_tcp_snd($l) set ack_]

        set pkt1        [$ag_tcp_snd(1) set ndatapack_]
        set pkt2        [$ag_tcp_snd(2) set ndatapack_]
        set pkt3        [$ag_tcp_snd(3) set ndatapack_]
        set pkt4        [$ag_tcp_snd(4) set ndatapack_]
        set pktenviados [expr $ndatapack + $pkt1 + $pkt2 + $pkt3 +$pkt4]

puts $record_f "$now [expr $cwin*1] $ssthresh $ndatapack $nrexmit $seq $rtt $srtt $rttvar $backoff
$pktx $ACK $pktenviados"

$ns at [expr $now+$time] "record "
}

proc record1 {} {
         global ag_tcp_rcv h0 h1 h2 h3 h4 h5
        set bwt 0
        #Get an instance of the simulator
        set ns [Simulator instance]
        #Set the time after which the procedure should be called again set time 49.0
        set time 49.0
        #How many bytes have been received by the traffic sinks?

        set bw0         [$ag_tcp_rcv(0) set bytes_]
        set bw1         [$ag_tcp_rcv(1) set bytes_]
        set bw2         [$ag_tcp_rcv(2) set bytes_]
        set bw3         [$ag_tcp_rcv(3) set bytes_]
        set bw4         [$ag_tcp_rcv(4) set bytes_]
        set bwt         [expr $bw0 + $bw1 +$bw2 + $bw3 +$bw4]

        #Get the current time
        set now [$ns now]
        #Calculate the bandwidth (in MBit/s) and write it to the files

        puts $h0 "$now $bw0"
        puts $h1 "$now $bw1"
        puts $h2 "$now $bw2"
```

```
        puts $h3 "$now $bw3"
        puts $h4 "$now $bw4"
        puts $h5 "$now $bwt"

        #Reset the bytes_ values on the traffic sinks
         # $ag_tcp_rcv(0) set bytes_ 0
        #$ag_tcp_rcv(1) set bytes_ 0
        #$ag_tcp_rcv(2) set bytes_ 0
        #$ag_tcp_rcv(3) set bytes_ 0
        #$ag_tcp_rcv(4) set bytes_ 0

        #Re-schedule the procedure
    $ns at [expr $now+$time] "record1"
}


proc finish {} {
        global ns outfile nf record_f nd_upl nd_dnl plot_res1
        global plot_res2 plot_res3
        global connections hn_tr_tcp_snd
        $ns flush-trace
        close $record_f
        close $outfile

        for {set i 0} {$i < $connections} {incr i} {
                close $hn_tr_tcp_snd($i)

        }

        exec grep " [$nd_upl(0) id] [$nd_dnl(0) id] " out.tr | \
        cut -d " " -f 1,2,3,4,5,6,7,8,9,10,11,12 > q_up

        exec grep "1 6 tcp" out.tr > con1-6
        exec grep "2 7 tcp" out.tr > con2-7
        exec grep "3 8 tcp" out.tr > con3-8
        exec grep "4 9 tcp" out.tr > con4-9
        exec grep "5 10 tcp" out.tr > con5-10

        # Recordar que si se hace algun cambio toca deshabilitar exec awk y luego   #deshabilitarlo y
        volver a correr
        #exec awk -f retardo.awk -v FID=0 DST=16.0 con1-6 > retardo.conx3
        exec xgraph retardo.conx3 -m -x tiempo -y Tiempo -t Westwood -geometry 800x400 &

        exec perl throughput.pl out.tr 16 1 > thp16
        exec xgraph thp16 -m -x tiempo -y Bytes/s -t Westwood -geometry 800x400 &

        exec awk { { print $1, $2 } } con1-6 > coliAloha1
        exec grep "c" coliAloha1 > coliAloha1-6

        exec awk { { print $1, $2 } } con2-7 > coliAloha2
        exec grep "c" coliAloha2 > coliAloha2-7

        exec awk { { print $1, $2 } } con3-8 > coliAloha3
        exec grep "c" coliAloha3 > coliAloha3-8

        exec awk { { print $1, $2 } } con4-9 > coliAloha4
        exec grep "c" coliAloha4 > coliAloha4-9

        exec awk { { print $1, $2 } } con5-10 > coliAloha5
        exec grep "c" coliAloha5 > coliAloha5-10
```

```
        exec awk { { print $1, $2 } } con1-6 > coliAloha
        exec grep "c" coliAloha > coliAloha1-6

        exec awk { { print $1, $2 } } networkSat1.data > t1.cwnd
        exec awk { { print $1, $3 } } networkSat1.data > t1.sst
        exec awk { { print $1, $4 } } networkSat1.data > t1.ndatapack
        exec awk { { print $1, $5 } } networkSat1.data > t1.nrexmit
        exec awk { { print $1, $6 } } networkSat1.data > t1.seq
        exec awk { { print $1, $7 } } networkSat1.data > t1.rtt
        exec awk { { print $1, $8 } } networkSat1.data > t1.srtt
        exec awk { { print $1, $9 } } networkSat1.data > t1.rttvar
        exec awk { { print $1, $10 } } networkSat1.data > t1.backoff
        exec awk { { print $1, $11 } } networkSat1.data > t1.pktx
        exec awk { { print $1, $12 } } networkSat1.data > t1.ACK
        exec awk { { print $1, $13 } } networkSat1.data > t1.pkEnviTo

exec xgraph t1.cwnd t1.sst -m -x tiempo -y Segmentos -t Westwood -geometry 800x400 &
exec xgraph t1.pktx t1.seq t1.ACK -m -x tiempo -y Segmentos -t Westwood -geometry 800x400 &
exec xgraph t1.ACK t1.ndatapack -m -x tiempo -y Segmentos -t Westwood -geometry 800x400 &
exec xgraph t1.nrexmit t1.backoff -m -x tiempo -y time_out -t Westwood -geometry 800x400 &
exec xgraph t1.rtt -m -x tiempo -y Tiempo -t Westwood -geometry 800x400 &
#exec xgraph t1.cwnd t1.sst t1.seq t1.ACK  -m -x tiempo -y Grafica4 -geometry 800x400 &
#exec xgraph t1.cwnd t1.sst t1.seq t1.pktx -m -x tiempo -y Grafica6 -geometry 800x400 &
#exec xgraph t1.rttvar t1.ACK t1.seq -m -x tiempo -y Grafica7 -geometry 800x400 &
#exec xgraph t1.rttvar t1.seq t1.ndatapack -m -x tiempo -y Grafica8 -geometry 800x400 &
#exec xgraph t1.ACK t1.seq t1.pktx t1.ndatapack -m -x tiempo -y Grafica10 -geometry 800x400 &
#exec xgraph t1.cwnd t1.sst t1.ndatapack -m -x tiempo -y Grafica11 -geometry 800x400 &

#ok      exec xgraph trafico0.tr -geometry 800x400 &
#ok      exec xgraph trafico1.tr trafico2.tr -geometry 800x400 &
#ok      exec xgraph trafico3.tr trafico4.tr -geometry 800x400 &

 exit 0
}

$ns at 1.0 "$ftp(0) start"
$ns at 1.0 "$ftp(1) start"
$ns at 1.0 "$ftp(2) start"
$ns at 1.0 "$ftp(3) start"
$ns at 1.0 "$ftp(4) start"
$ns at 1.1 "record"
$ns at 1.1 "record1"
$ns at 300.0 "finish"
$ns run
```

5.0. SCRIPT DE CONFIGURACIÓN UTILIZADO PARA EVALUAR EL DESEMPEÑO DE SNOOP.

```
# Se debe indicar la ruta de los archivos vlan.tcl, ns-mac.tcl y ns-ll.tcl
source ~/Documentos/instalacions2/ns-allinone-2.31/ns-2.31/tcl/lan/vlan.tcl
source ~/Documentos/instalacions2/ns-allinone-2.31/ns-2.31/tcl/lan/ns-mac.tcl
source ~/Documentos/instalacions2/ns-allinone-2.31/ns-2.31/tcl/lan/ns-ll.tcl

set flavor TdmaDama
append scenario $flavor
file mkdir $scenario
```

```
set testing             1
set myseed              1
set duration            1000.0
set s_int               1.0

# Satellite Dependent

set frame_duration    0.05
set delayTER          10ms
set delayRCST         10ms

set capTER            10Mb
set cap               10
set capSAT            4812800

set k_frame_          1.0
set rcst_buffer_      500
set hub_buffer_       500

set slotsize_         1504
set state_period      0.0417
set superframe_       20

##### Radio Resource Management ##############

Mac/TdmaDama set fixed_frame_ 0

Allocator/FODA set ignore_loss_ 1
Allocator/FODA set delay_ 0.797
Allocator/FODA set def_slots_ 0
Allocator/FODA set xpercent_ 1.0

Requester/RVBDC set alpha_ 1.0
Requester/RVBDC set k_ 1.0
Requester/RVBDC set min_rvbdc_ 1.0
Requester/RVBDC set max_rvbdc_ 0

Queue/DropTail set drop_front_ true

############## Traffic Flows ##############

# Data Flows
set data(dir)           upload
set data(next)          3.0
set data(bunch)         5
set data(packet)        1500


set record_f [open networkSat1.data w]
set record_q [open net.data w]
set record_l [open lnk.data w]
set record_T [open flowT.data w]

set h0 [open trafico0.tr w]
set h1 [open trafico1.tr w]
set h2 [open trafico2.tr w]
set h3 [open trafico3.tr w]
set h4 [open trafico4.tr w]

set f16 [open trafico5.tr w]
```

```
set f17 [open trafico6.tr w]

set connections  5

for {set i 0} {$i < $connections} {incr i} {
        set fl_tr_tcp_snd($i)           $scenario/tcp_snd_$i.tr
        set hn_tr_tcp_snd($i)           [open $fl_tr_tcp_snd($i) w]
}

##########  loading alternative configurations ############
set cmdline [lindex $argv 0]
if { $cmdline != "scpc" && $cmdline !="bod" } {
        puts "usage: ./test1.tcl \[bod|scpc\]"
        exit 1
}

ns-random $myseed
expr srand($myseed)

#################################################################

set slotframe_   [expr round(($capSAT*$frame_duration)/(8*$slotsize_))]
set data(load)   [expr $data(bunch)*$data(packet)*8/($capSAT*$data(next))]
puts "data(load) [format %.4f $data(load)]"
puts "slotframe_  [format %.4f $slotframe_ ]"

if { $cmdline == "scpc" } {
        Mac/TdmaDama set fixed_frame_ 1
        Mac/TdmaDama set slots1_ $slotframe_
        Mac/TdmaDama set offset_ -1
}

# scheduling data flows
#set data(index) 0

set traced_vars [list cwnd_ ssthresh_ ack_ t_seqno_ rtt_ seqno_]

proc create_tcp_conn { idx src dst start stop} {
        global ns rcst ncc tcp ftp sink data
        global traced_vars hn_tr_tcp_snd

        set time_start [$ns now]

        set tcp($idx) [new Agent/TCP/Linux]
        $tcp($idx) set window_ 800;
        $tcp($idx) set packetSize_ 1420
        $tcp($idx) set timestamps_ true
        $ns at $time_start "$tcp($idx) select_ca hybla"
        set sink($idx) [new Agent/TCPSink/Sack1]

        $ns attach-agent $dst $sink($idx);
        $ns attach-agent $src $tcp($idx);

        #$tcp($idx) set class_ [expr $idx + 1]
        $tcp($idx) set fid_ $idx


        #FTP application is used to generate TCP traffic
        set ftp($idx) [new Application/FTP];
        $ftp($idx) attach-agent $tcp($idx);
```

```
        set satrouteobject_ [new SatRouteObject];
        $satrouteobject_ compute_routes;

        $ns connect $tcp($idx) $sink($idx)

        $ns at $start "$ftp($idx) start";
        $ns at $stop  "$ftp($idx) stop";

        $tcp($idx) attach-trace $hn_tr_tcp_snd($idx)

        for {set j 0} {$j < [llength $traced_vars]} {incr j} {
                $tcp($idx) trace [lindex $traced_vars $j]
        }
}


Mac/TdmaDama set max_slot_num_ $slotframe_
Mac/TdmaDama set slot_packet_len_ $slotsize_
Mac/TdmaDama set bandwidth_ $capSAT
Mac/TdmaDama set num_frame_ $superframe_
Mac/TdmaDama set aligned_ 1

Allocator/FODA set k_frame_ $k_frame_
Allocator/FODA set maxg_ $slotframe_
Allocator/FODA set ming_ 1

set ns [new Simulator]
set stats_con [open stats.tr w]
if { $testing == 1 } {
        # Tracing enabling must
        # precede link and node creation
        set f0 [open out.tr w]
        $ns trace-all $f0
}

# nodes

set opt(chan)           Channel/Sat
set opt(bw_up)          $capSAT ;# uplink bandwidth
set opt(bw_down)        $capSAT ;# downlink bandwidth
set opt(phy)            Phy/Sat
set opt(mac)            Mac/TdmaDama
set opt(ifq)            Queue/DropTail
set opt(qlim)           230;# queue size (pkts)
set opt(ll)             LL/Sat
set opt(wiredRouting)   ON

# Configure bent-pipe satellite
$ns node-config -satNodeType geo-repeater \
                -llType $opt(ll) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(qlim) \
                    -macType $opt(mac) \
                    -phyType Phy/Repeater \
                    -channelType $opt(chan) \
                    -downlinkBW $opt(bw_down)  \
                    -wiredRouting $opt(wiredRouting)
```

```
# GEO satellite at 13 degrees longitude East (Hotbird 6)
set sat [$ns node]

$sat set-position 13


$ns node-config -satNodeType terminal \
                -llType $opt(ll) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(qlim) \
                    -macType $opt(mac) \
                    -phyType $opt(phy) \
                    -channelType $opt(chan) \
                    -downlinkBW $opt(bw_down) \
                    -wiredRouting $opt(wiredRouting)

set hub [$ns node];                 #HUB
$hub set-position 40.9 -73.9; # NY

set rcst(0) [$ns node]
$rcst(0) set-position 37.8 -122.4; # RCST 0
set rcst(1) [$ns node]
$rcst(1) set-position 34.8 -137.4; # RCST 1
set rcst(2) [$ns node]
$rcst(2) set-position 31.8 -152.4; # RCST 2
set rcst(3) [$ns node]
$rcst(3) set-position 28.8 -167.4; # RCST 3
set rcst(4) [$ns node]
$rcst(4) set-position 55.8 -140.4; # RCST 4


$hub add-gsl geo $opt(ll) $opt(ifq) $hub_buffer_ $opt(mac) \
        $opt(bw_up) $opt(phy) [$sat set downlink_] [$sat set uplink_]


for {set i 0} {$i < $connections} {incr i} {

    $rcst($i) add-gsl geo $opt(ll) $opt(ifq) $rcst_buffer_ $opt(mac) \
        $opt(bw_up) $opt(phy) [$sat set downlink_] [$sat set uplink_]

}

set em_ [new ErrorModel]
$em_ unit pkt
$em_ set rate_ 0.00002
$em_ ranvar [new RandomVariable/Uniform]
$hub interface-errormodel $em_

set gat_mac [$hub set mac_(0)]
set ter_mac(0) [$rcst(0) set mac_(0)]
set ter_mac(1) [$rcst(1) set mac_(0)]
set ter_mac(2) [$rcst(2) set mac_(0)]
set ter_mac(3) [$rcst(3) set mac_(0)]
set ter_mac(4) [$rcst(4) set mac_(0)]
$gat_mac set hub_ 1

if { $testing == 1 } {
        set ev_file [open event.tr w]
        $hub trace-event $ev_file
        $rcst(0) trace-event $ev_file
```

```
        $rcst(1) trace-event $ev_file
        $rcst(2) trace-event $ev_file
        $rcst(3) trace-event $ev_file
        $rcst(4) trace-event $ev_file
        $ns trace-all-satlinks $f0
}

# create node requester object
set streq [$hub install-requester Requester/Constant]
set dtreq(0) [$rcst(0) install-requester Requester/RVBDC]
set dtreq(1) [$rcst(1) install-requester Requester/Constant]
set dtreq(2) [$rcst(2) install-requester Requester/Constant]
set dtreq(3) [$rcst(3) install-requester Requester/Constant]
set dtreq(4) [$rcst(4) install-requester Requester/Constant]


# Network Control Center
set ncc [$rcst(0) install-allocator Allocator/FODA]

############################## Set up S n o o p ##########################

$ns unset satNodeType_

set opt(bw)             10Mb
set opt(delay)          3ms
set opt(ll)             LL
set opt(ifq)            Queue/DropTail
set opt(mac)            Mac/802_3
set opt(chan)           Channel

set naux(0) [$ns node]
set naux(1) [$ns node]
set naux(2) [$ns node]
set naux(3) [$ns node]
set naux(4) [$ns node]

lappend nodelist0 $naux(0)
lappend nodelist1 $naux(1)
lappend nodelist2 $naux(2)
lappend nodelist3 $naux(3)
lappend nodelist4 $naux(4)

# Creamos nuestra LAN, utilizando las variables de red y la lista de nodos creados
set lan0 [$ns make-lan $nodelist0 $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac) $opt(chan)]
set lan1 [$ns make-lan $nodelist1 $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac) $opt(chan)]
set lan2 [$ns make-lan $nodelist2 $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac) $opt(chan)]
set lan3 [$ns make-lan $nodelist3 $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac) $opt(chan)]
set lan4 [$ns make-lan $nodelist4 $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac) $opt(chan)]

set opt(ll) LL/LLSnoop
set opt(ifq) Queue/DropTail
$opt(ifq) set limit_ 200; #ver1
set nsnoop(0) [$ns node]
set nsnoop(1) [$ns node]
set nsnoop(2) [$ns node]
set nsnoop(3) [$ns node]
set nsnoop(4) [$ns node]
# AÃ±ade a nuestra LAN el nodo snoop

$lan0 addNode [list $nsnoop(0)] $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac)
```

```
$lan1 addNode [list $nsnoop(1)] $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac)
$lan2 addNode [list $nsnoop(2)] $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac)
$lan3 addNode [list $nsnoop(3)] $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac)
$lan4 addNode [list $nsnoop(4)] $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac)

#Crea los nodos origen y destino

for {set i 0} {$i < $connections} {incr i} {
        set nd_snd($i)   [$ns node]
        $ns duplex-link $nd_snd($i) $nsnoop($i) 10Mb 3ms DropTail
        $ns queue-limit $nd_snd($i) $nsnoop($i) 100000
}

for {set i 0} {$i < $connections} {incr i} {

        $ns duplex-link $naux($i) $rcst($i) 10Mb 4ms DropTail
        $ns queue-limit $naux($i) $rcst($i) 9; }


for {set i 0} {$i < $connections} {incr i} {
        set nd_rcv(0)    [$ns node]
        $ns duplex-link $nd_rcv(0) $hub $capTER $delayTER DropTail;
        $ns queue-limit $nd_rcv(0) $hub 12
}

        set qmon(0) [$ns monitor-queue $hub $nd_rcv(0)  ""]
        set fmon(0) [$ns makeflowmon Fid]
        $ns attach-fmon [$ns link $hub $nd_rcv(0)] $fmon(0)

set satrouteobject_ [new SatRouteObject]
$satrouteobject_ compute_routes


#Se crea nodos y sus respectivos enlaces

set start 1.0

### Drawing diagrams

Agent/TCP/Sack1 instproc done {} {
        global ns stats_con
        set init [$self set time_start]
        set dur [expr [$ns now] - $init]
        puts $stats_con [format "%10.2f %8.4f" $init $dur]

}

set function_ci {
        function A = autocorr(x)
                A = x - mean(x);
                A = fftconv(A, rot90(A,2));
                A = A(length(x) : 2*length(x)-1);
                A /= length(x);
                A /= A(1);
        end

        function t = corrlen(x)

         if (rows(x)==1)
           x = x';
```

```
        endif
        [l,n] = size(x);
        weights = 1 - [0.5, [1:l-1]/(l-1)];
        clen = zeros(1,n);
        for i=1:n
          clen(i) = weights * autocorr(x(:,i));
        endfor
        t = max(1,clen);

      end

      input = load("stats.tr");
      input = input(:,2);

      step = ceil(corrlen(input));
      data = input(1:step:length(input));

      confidence = 0.95;

      m = mean(data);
      s = std(data);
      p = abs(tinv( (1-confidence)/2 , length(data) - 1));
      p = p * s / sqrt(length(data));

      ci = [m ; p];

      h = reshape(ci, 1, size(ci)(1) * size(ci)(2));

      printf("%10.5g %6.4f",h);
      printf("\n");
}

set plot_res1 {
      set xlabel "time [s]"
      set ylabel "bandwidth [b/s]"

      set xrange [0:1000]
      set yrange [0:4812800]

      set term postscript color solid eps
      set output "result1.eps"

      plot "q_up.dat" u 1:4 t 'used cap.' w l 3, \
          "curall.tr" u (floor($2)):($5*4812800/20) smooth unique \
          t 'alloc cap.' w l 1
}

set plot_res2 {
      set xlabel "time [s]"
      set ylabel "Encolamiento de paquetes [b/s]"

      set xrange [0:1000]
      set yrange [0:600]
      set term postscript color solid eps
      set output "Encolamiento.eps"
      plot "q_up.dat" u 1:7 t 'Encolamiento.' w l 4

}
```

```
set plot_res3 {
        set xlabel "time [s]"
        set ylabel "Retardo"
        set xrange [0:1000]
        set yrange [0:4]

        set term postscript color solid eps
        set output "Retardo.eps"
        plot "q_up.dat" u 1:10 t 'Retardo.' w l 5

}


set plot_ReqAll {
        set xlabel "time [s]"
        set ylabel "Allocator"

        set xrange [0:1000]
        set yrange [0:30]

        set term postscript color solid eps
        set output "ReqAll.eps"

        plot "curall.tr" u (floor($2)):5 t 'Slots asignados.' w l 5
}

set plot_Req {
        set xlabel "time [s]"
        set ylabel "Requester"

        set xrange [0:1000]
        set yrange [0:40]

        set term postscript color solid eps
        set output "Req.eps"

        plot "reques.tr" u (floor($2)):5 t 'Slots solicitados.' w l 1
#
}

SimpleLink instproc attach-snooper-in { SrcNode DstNode } {
  global ns

  $self instvar Snooper Entry Link TtlPointer Monitor

  #Se crea un objecto de tipo SnoopQueue/In
  set Snooper [new SnoopQueue/In]
  puts stdout "Creado Snooper $Snooper"

  set Entry [$DstNode entry]

  set Link [$ns link $SrcNode $DstNode]
  set TtlPointer [$Link set ttl_]

  $TtlPointer target $Snooper
  $Snooper target $Entry

  # Se crea un objeto QueueMonitor para recibir los datos desde el Snooper.
  set Monitor [new QueueMonitor]
  puts "Monitor creado : $Monitor"
```

```
  $Snooper set-monitor $Monitor

  #Retorna el objeto Monitor
  return $Monitor

 }

# Se define un procedimiento 'record'

proc record {} {
        global ns tcp record_f
        set l 0
        set now [$ns now]
        set time 0.1
        set cwin        [$tcp($l) set cwnd_]
        set ssthresh    [$tcp($l) set ssthresh_]
        set ndatapack   [$tcp($l) set ndatapack_]
        set nrexmit     [$tcp($l) set nrexmit_]
        set seq         [$tcp($l) set t_seqno_]
        set rtt         [$tcp($l) set rtt_]
        set srtt        [$tcp($l) set srtt_]
        set rttvar      [$tcp($l) set rttvar_]
        set backoff     [$tcp($l) set backoff_]
        set pktx        [$tcp($l) set nrexmitpack_]
        set ACK         [$tcp($l) set ack_]

puts $record_f "$now [expr $cwin*1] $ssthresh $ndatapack $nrexmit $seq $rtt $srtt $rttvar $backoff
$pktx $ACK"

$ns at [expr $now+$time] "record "
}

proc record1 {} {
      global sink h0 h1 h2 h3 h4
         global MonitorIn1 f16 MonitorIn2 f17
         #Get an instance of the simulator
         set ns [Simulator instance]
         #Set the time after which the procedure should be called again
      set time 1.0
         #How many bytes have been received by the traffic sinks?

      set bw0 [$sink(0) set bytes_]
      set bw1 [$sink(1) set bytes_]
      set bw2 [$sink(2) set bytes_]
      set bw3 [$sink(3) set bytes_]
      set bw4 [$sink(4) set bytes_]

     set long_cola_pkts [$MonitorIn1 set pkts_]
     set pkts_llegados [$MonitorIn1 set parrivals_]
     set pkts_partidos [$MonitorIn1 set pdepartures_]
     set pkts_perdidos [$MonitorIn1 set pdrops_]

    set long_cola_pkts2 [$MonitorIn2 set pkts_]
    set pkts_llegados2 [$MonitorIn2 set parrivals_]
    set pkts_partidos2 [$MonitorIn2 set pdepartures_]
    set pkts_perdidos2 [$MonitorIn2 set pdrops_]

         #Get the current time
        set now [$ns now]
         #Calculate the bandwidth (in MBit/s) and write it to the files
```

```
        puts $h0 "$now [expr $bw0/$now*8]"
        puts $h1 "$now [expr $bw1/$now*8]"
        puts $h2 "$now [expr $bw2/$now*8]"
        puts $h3 "$now [expr $bw3/$now*8]"
        puts $h4 "$now [expr $bw4/$now*8]"
        puts $f16 "$now $pkts_llegados $pkts_partidos $pkts_perdidos $long_cola_pkts"
        puts $f17 "$now $pkts_llegados2 $pkts_partidos2 $pkts_perdidos2 $long_cola_pkts2 "


            #Reset the bytes_ values on the traffic sinks
            $sink(0) set bytes_ 0
            $sink(1) set bytes_ 0
            $sink(2) set bytes_ 0
            $sink(3) set bytes_ 0
            $sink(4) set bytes_ 0

            $MonitorIn1 set pkts_ 0
            $MonitorIn2 set pkts_ 0

            #Re-schedule the procedure
        $ns at [expr $now+$time] "record1"
}

proc recordqueue {} {

      global qmon ns record_q

            set now [$ns now]
            set time 0.1
            set size            [$qmon(0) set size_]
            set pkts            [$qmon(0) set pkts_]
            set parrivals       [$qmon(0) set parrivals_]
            set barrivals       [$qmon(0) set barrivals_]
            set pdepartures [$qmon(0) set pdepartures_]
            set bdepartures [$qmon(0) set bdepartures_]
            set pdrops          [$qmon(0) set pdrops_]
            set bdrops          [$qmon(0) set bdrops_]
            set pmarks          [$qmon(0) set pmarks_]


    puts $record_q "[format %.2f [$ns now]] $size $pkts $parrivals $barrivals $pdepartures
$bdepartures $pdrops $bdrops $pmarks"

$ns at [expr $now+$time] "recordqueue"
}

proc reclink { } {

global fmon cap ns record_l

    set now [format "%.1f" [$ns now]]
    set now [$ns now]
    set time 0.1

    set  bytes                  [$fmon(0) set bdepartures_]
    set  bytesDbl               [ns-int64todbl $bytes]
    set  Kbytes                 [expr $bytesDbl / 1000 ]
    set  bandwidthToKBytes      [expr 1000 * [$ns now] / 8 ]
    set  possibleKBytes         [format %.2f [expr $cap* $bandwidthToKBytes]]
    set  drops_t                [$fmon(0) set pdrops_]
```

```
    set  marks_t                    [$fmon(0) set pmarks_]
    set  packets_t                  [$fmon(0) set pdepartures_]
    set  tlu_t                       [format %.2f [expr $Kbytes * 100 / $possibleKBytes ]]

    puts $record_l "[format %.2f [$ns now]] $drops_t $marks_t $packets_t $Kbytes     $possibleKBytes
$tlu_t"
    $ns at [expr $now+$time] "reclink"
}

proc recflow { } {
global ns cap connections qmon record_T fmon

  set now [format "%.1f" [$ns now]]
  set now [$ns now]
  set time 0.1
  # calculus of possible bytes to sent
  set bandwidthToKBytes [expr 1000 * [$ns now] / 8 ]
  set possibleKBytes [expr $cap * $bandwidthToKBytes ]
  set auxH 0
  set linkutilzH     0
  set TOTparrivalsH  0
  set TOTpdropsH     0

  # HSTCP Flows
  for {set fid 0} {$fid < 1} {incr fid} {
      # extract data for this particular flow
      set fcl [$fmon(0) classifier]; # flow classifier
      set flow [$fcl lookup auto 0 0 $fid]
      if {$flow != "" } {
         set bytes [$flow set bdepartures_]
         set bytesDbl [ns-int64todbl $bytes]
         set Kbytes [expr $bytesDbl / 1000 ]

            set auxH [expr $linkutilzH + [expr $Kbytes * 100 / $possibleKBytes] ]
           set linkutilzH $auxH
            set auxH [ns-int64todbl [expr $TOTpdropsH + [$flow set pdrops_] ] ]
            set TOTpdropsH [ns-int64todbl $auxH]

            set auxH [ns-int64todbl [expr $TOTparrivalsH + [$flow set parrivals_] ] ]
            set TOTparrivalsH [ns-int64todbl $auxH]
        }
     }

          if { $connections != 0 } {
                  set linkutilzavgH [expr $linkutilzH / $connections]
            }

       if {$TOTparrivalsH != 0} {
                  set lossrateH [expr $TOTpdropsH/$TOTparrivalsH]
             } else {
                   set lossrateH 0.0
              }

     puts $record_T "[format %.2f [$ns now]] $linkutilzH $linkutilzavgH $lossrateH"

$ns at [expr $now+$time] "recflow"

}

# Se llama al procedimiento create_tcp_conn para configurar los parámetros de entrada
```

```
create_tcp_conn 0 $nd_snd(0) $nd_rcv(0) 0.01 $duration
create_tcp_conn 1 $nd_snd(1) $nd_rcv(0) 0.01 $duration
create_tcp_conn 2 $nd_snd(2) $nd_rcv(0) 0.01 $duration
create_tcp_conn 3 $nd_snd(3) $nd_rcv(0) 0.01 $duration
create_tcp_conn 4 $nd_snd(4) $nd_rcv(0) 0.01 $duration

set Link1  [$ns link $nd_snd(0) $nsnoop(0)]
set MonitorIn1 [$Link1 attach-snooper-in $nd_snd(0) $nsnoop(0)]
puts stdout "Link > $Link1"

set Link2  [$ns link $nd_snd(1) $nsnoop(1)]
set MonitorIn2 [$Link2 attach-snooper-in $nd_snd(1) $nsnoop(1)]
puts stdout "Link > $Link2"

proc finish-sim {} {
        global ns rcst hub stats_con cmdline
        global function_ci plot_res1
        global plot_ReqAll plot_Req
        global plot_res2 plot_res3
        global f16 f17
        global hn_tr_tcp_snd connections
        global ns record_f h0 h1 h2 h3 h4 record_q record_l record_T
        $ns flush-trace
        $ns halt

        close $record_l
        close $record_q
        close $record_f
        close $record_T
        close $h0
        close $h1
        close $h2
        close $h3
        close $h4
        close $f16
        close $f17

        close $stats_con

        for {set i 0} {$i < $connections} {incr i} {
                close $hn_tr_tcp_snd($i)

        }
        # Se coloca la estación TX y luego la receptorasi se quiere ver el canal de datos
        exec grep " [$rcst(0) id] [$hub id] " out.tr | \
        cut -d " " -f 1,2,3,4,5,6,7,8,9,10,11,12 > q_up

        exec analy q_up 1.0
        exec grep curall event.tr > curall.tr
        exec grep reques event.tr > reques.tr

        exec echo $plot_ReqAll | gnuplot
        exec echo $plot_Req | gnuplot
        exec echo $plot_res1 | gnuplot
        exec echo $plot_res2 | gnuplot
        exec echo $plot_res3 | gnuplot
        exec mv result1.eps result1-${cmdline}.eps
        exec mv stats.tr stats-test1-${cmdline}.tr
        exec mv q_up.dat q_up-test1-${cmdline}.dat
        exec mv out.tr out-test1-${cmdline}.tr
```

```
        exec mv event.tr event-test1-${cmdline}.tr
        #exec rm q_up reques.tr curall.tr;

        exec perl throughput.pl out-test1-bod.tr 31 1 > ThrputSHY

        exec awk { { print $1, $2 } } networkSat1.data > cwndSHY
        exec awk { { print $1, $3 } } networkSat1.data > sstSHY
        exec awk { { print $1, $4 } } networkSat1.data > pkTxSHY
        exec awk { { print $1, $5 } } networkSat1.data > nrexmitSHY
        exec awk { { print $1, $6 } } networkSat1.data > seqSHY
        exec awk { { print $1, $7 } } networkSat1.data > rttSHY
        exec awk { { print $1, $8 } } networkSat1.data > srttSHY
        exec awk { { print $1, $9 } } networkSat1.data > rttvarSHY
        exec awk { { print $1, $10 } } networkSat1.data > backoffSHY
        exec awk { { print $1, $11 } } networkSat1.data > pkRTxSHY
        exec awk { { print $1, $12 } } networkSat1.data > ACKSHY

exec xgraph cwndSHY sstSHY -t  "Ventana Congestion y Ssthresh " -m -x Tiempo -y Segmentos -
geometry 800x400 &
exec xgraph pkTxSHY -t  "Paquetes Transmitidos " -m -x Tiempo -y Segmentos -geometry 800x400 &
exec xgraph pkRTxSHY -t  "Paquetes Retransmitidos  " -m -x Tiempo -y Segmentos -geometry
800x400 &
exec xgraph seqSHY -t " Numero de Secuencia " -m -x Tiempo -y Segmentos -geometry 800x400 &
exec xgraph backoffSHY nrexmitSHY -t  "Back-off y Numero de Retransmisiones del Timeout" -m -x
Tiempo -y ValorxDefecto -geometry 800x400 &
exec xgraph rttSHY -t "  RTT   " -m -x Tiempo -y Tiempo -geometry 800x400 &
exec xgraph ACKSHY -t  " ACK   " -m -x Tiempo -y Segmentos -geometry 800x400 &

exec awk { { print $1, $2 } } net.data > queue.size
exec xgraph queue.size -m -x tiempo -y Grafica16 -geometry 800x400 &
exec awk { { print $1, $7 } } lnk.data > flowTSHY
exec awk { { print $1, $2 } } flowT.data > flowSHY
exec xgraph flowTSHY flowSHY -t  " Capacidad Total y Flujo de la conexion cero " -m -x Tiempo -y
Utilizacion% -geometry 800x400 &

        exit 0
}

$ns at 1.0 "record"
$ns at 1.1 "reclink"
$ns at 1.1 "recflow"
$ns at 1.1 "recordqueue"
$ns at 1.1 "record1"
$ns at $duration "finish-sim"

$ns run
```

6.0. ARCHIVOS GENERADOS AL CORRER EL SCRIPT.

A continuación se describirán los archivos que se generan en la simulación, con sus respectivos datos, para luego ser analizados y comparados.

6.1. ARCHIVO *OUTPUT*-TEST1.TR.

Este archivo contiene el registro de los eventos generados durante el proceso de simulación, la figura 6 muestra la forma en que se presentan los datos, la manera de crear este archivo es:

set f0 [open out.tr w]
$ns trace-all $f0

```
+ 130.1437 1 2 ack 56 ------- 0 7.0 8.0 16 15671 40.90 -73.90 37.80 -122.40
- 130.1437 1 2 ack 56 ------- 0 7.0 8.0 16 15671 40.90 -73.90 37.80 -122.40
r 130.144275 4 10 ack 56 ------- 2 7.2 10.0 16 15654
+ 130.144275 10 4 tcp 1510 ------- 2 10.0 7.2 725 15673
- 130.144275 10 4 tcp 1510 ------- 2 10.0 7.2 725 15673
r 130.146693 6 12 ack 56 ------- 4 7.4 12.0 16 15656
+ 130.146693 12 6 tcp 1510 ---A--- 4 12.0 7.4 808 15674
- 130.146693 12 6 tcp 1510 ---A--- 4 12.0 7.4 808 15674
r 130.1467 5 1 tcp 1510 ------- 3 11.0 7.3 604 14069 28.80 -167.40 40.90 -73.90
+ 130.146711 1 7 tcp 1510 ------- 3 11.0 7.3 604 14069
- 130.146711 1 7 tcp 1510 ------- 3 11.0 7.3 604 14069
r 130.1522 1 3 ack 56 ------- 1 7.1 9.0 16 15659 40.90 -73.90 34.80 -137.40
```

Figura 6. Datos arrojados por el archivo out.tr.

Este archivo está conformado por varias columnas tal y como se observa en la figura 6, donde la primera columna muestra el tipo de evento donde: '+' indica entrada en el búfer, '-' la salida del búfer, 'r' indica paquete recibido, 'c' muestra una colisión, y 'd' revela un paquete perdido.

La segunda columna muestra el tiempo (en segundos) en que sucedió dicho evento, la tercera y cuarta columna indican los dos nodos entre los que ocurre el evento, la quinta y sexta muestran el tipo de paquete y el tamaño del paquete tcp respectivamente; las siguientes columnas contienen las banderas, en este caso se utiliza la letra A, la columna siguiente muestra el identificador de flujo IP para Ipv6. Las dos columnas subsecuentes indican la fuente del paquete y el nodo destino respectivamente. El campo que le sigue indica el número de secuencia del paquete entre los dos nodos y la última columna es un identificador único que se le asigna a cada paquete [4].

Una variante de este archivo de traza se utiliza para las redes de tipo satelital. A este archivo se le agregan 4 campos al final de cada línea, los dos primeros indican la posición (latitud y longitud) de la estación fuente, y los siguientes dos campos las posición de la estación destino ver figura 7.

```
+ 130.146693 12 6 tcp 1510 ---A--- 4 12.0 7.4 808 15674
- 130.146693 12 6 tcp 1510 ---A--- 4 12.0 7.4 808 15674
r 130.1467 5 1 tcp 1510 ------- 3 11.0 7.3 604 14069 28.80 -167.40 40.90 -73.90
+ 130.146711 1 7 tcp 1510 ------- 3 11.0 7.3 604 14069
- 130.146711 1 7 tcp 1510 ------- 3 11.0 7.3 604 14069
r 130.1522 1 3 ack 56 ------- 1 7.1 9.0 16 15659 40.90 -73.90 34.80 -137.40
+ 130.152211 3 9 ack 56 ------- 1 7.1 9.0 16 15659
- 130.152211 3 9 ack 56 ------- 1 7.1 9.0 16 15659
```

Figura 7. Datos arrojados por el archivo *out.tr* en una red satelital.

6.2. ARCHIVO EVENT.TR.

Este archivo contiene información de cómo se está haciendo la solicitud y asignación de recursos por parte de las estaciones terrenas y la estación de control, la figura 8 muestra los datos donde el primer campo representa el terminal identificador, el segundo el tiempo

transcurrido y el tercero está conformado por una serie de etiquetas que especifican un tipo de entrada, el archivo se origina mediante las siguientes líneas de comandos [4].

```
if { $testing == 1 } {
        set ev_file [open event.tr w]
        $rcst trace-event $ev_file
        $hub(0) trace-event $ev_file
        $hub(1) trace-event $ev_file
        $hub(2) trace-event $ev_file
        $hub(3) trace-event $ev_file
        $hub(4) trace-event $ev_file
        $ns trace-all-satlinks $f0
}
```

```
<5> 2.005714 strslt 5
<5> 2.005714 s-frag 184
<5> 2.039857 curslt [1][2][3][4][5][0][0][0][0][0][0][0][0][0]
<5> 2.039857 curall 9 1 1 1 1 1
<1> 2.040000 strslt 1
<1> 2.040000 s-pack 40
<5> 2.042857 reqpar 45380 338240.000000
<0> 2.042857 reqpar 0 640.000000
<1> 2.042857 reqpar 0 640.000000
<2> 2.042857 reqpar 0 320.000000
<2> 2.042857 strslt 2
<2> 2.042857 s-pack 40
<3> 2.042857 reqpar 0 0.000000
<4> 2.042857 reqpar 0 0.000000
<5> 2.042857 reques 1.017 1.017 1.009 1.000 1.000 1.000
<5> 2.042857 nxtslt [0][0][0][0][0][0][1][2][3][4][5][0][0][0]
<5> 2.042857 nxtall 9 1 1 1 1 1
```

Figura 8. Datos arrojados por el archivo event.tr.

Las etiquetas de entrada *currall* y *curslt* muestran el número de slots asignados por la estación y la estructura de la trama actual respectivamente. Las etiquetas *nxtall* y *nxtslt* muestran el número de slots asignados por la estación y la estructura de la trama después de un retardo de asignación.

La etiqueta *request* muestra las solicitudes hechas por las estaciones terrenas, con respecto a la etiqueta *reqpar* esta indica la velocidad de entrada en la cola de la estación y el estado del búfer, además este valor se actualiza según sea el tiempo de planificación de recursos TBTP [5].

BIBLIOGRAFIA

[1] S. Fernández y A. Feliu, Diseño de un Entorno para el Estudio de los Parámetros de Funcionamiento del Protocolo TCP, Universidad Politécnica Cataluña, Jul. 2006. Documento disponible en: http://upcommons.upc.edu/pfc/bitstream/2099.1/3761/1/54363-1.pdf. Consultado 12 Enero del 2009.

[2] E de Souza, *A simulation-based study of HighSpeed TCP and its deployment, Universidad de California, USA, May 2003.* Documento disponible en: http://www.escholarship.org/uc/item/623581jj Consultado 20 Junio del 2009.

[3] R. Secchi, TDMA-DAMA, página web disponible en: http://ala.isti.cnr.it/wnlab/tdmadama.

[4] C. Carreón, V. Gatica, Evaluacion del Desempeño del Protocolo Aloha, Instituto Politécnico Nacional Escuela superior de Ingeniería Mecánica y Eléctrica, Mexico D.F., Noviembre del 2008. Documento disponible en: http://itzamna.bnct.ipn.mx:8080/dspace/handle/123456789/1714. Consultado el 20 Noviembre del 2009.

[5] A.Gotta, F. Potorti, R.Secchi, *Simulating Dynamic Bandwidth Allocation on Satellite Links*, Genova Italia 2005. Articulo disponible en: http://portal.acm.org/citation.cfm?id=1190462. Consultado el 20 Mayo del 2009.