ANEXO A

CONCEPTOS Y FUNCIONAMIENTO DE LOS SIMULADORES DE REDES

En el presente anexo se realiza la definición de conceptos importantes para los simuladores de redes OPNET Modeler y ns-3. Además, se describen sus funcionamientos y los procesos realizados para el desarrollo de los modelos de simulación.

1 OPNET Modeler

Para el desarrollo del presente proyecto de grado, se utilizó el software de simulación de redes OPNET Modeler en su versión 11.5. A continuación se presentan los conceptos más importantes del simulador, así como la descripción detallada del funcionamiento del modelo desarrollado en el mismo.

1.1 CONCEPTOS

OPNET Modeler tiene la facilidad de poseer distintos niveles de abstracción, los cuales son [54]:

1. **Red:** En este nivel de abstracción, el usuario de OPNET Modeler es capaz de crear una red con la utilización de las distintas librerías existentes en el programa, las cuales incluyen PCs, routers, servidores, etc., como también puede utilizar nodos personalizados que no estén por defecto en las librerías del programa. Debido a que la idea de este proyecto es crear un modelo que represente fielmente el comportamiento del tráfico de un servicio de VoD junto con aplicaciones asociadas al contenido, es necesario tener un servidor el cual transmita por separado cada una de las tramas MPEG-2 de los videos, por tal motivo se creó un nodo personalizado.
2. **Nodo:** En este nivel de abstracción, OPNET Modeler permite crear un nodo con sus características hardware, como transmisores, receptores, procesadores, etc. Aquí también es posible utilizar módulos existentes en las librerías de OPNET Modeler, los cuales tienen asignados procesos por defecto, o crear módulos personalizados con sus propios procesos. Para el presente proyecto se utilizó tanto módulos existentes por defecto en OPNET Modeler, como también módulos personalizados (esto se explicará en detalle más adelante en este mismo capítulo).
3. **Proceso:** Este nivel de abstracción se utiliza para asignar procesos a los módulos creados dentro de los nodos. Como se puede apreciar en la Figura 1-1, estos procesos se basan en máquinas de estado finito, las cuales tienen estados forzados (color verde) y estados no forzados (color rojo), además de un estado inicial (flecha) y líneas de interrupción (punteadas) y de continuación (no punteadas). Cada uno de los estados de la máquina de estados, posee dos tipos de ejecuciones, una de entrada y otra de salida, la primera se ejecuta cuando el proceso entra en el estado y la segunda cuando el proceso sale del estado. Estas ejecuciones están codificadas en

lenguaje de programación C; este código corresponde al último nivel de abstracción de OPNET Modeler y será descrito en el siguiente numeral. Los estados forzados se caracterizan porque cuando el proceso entra en el estado, este ejecuta el código de entrada, el de salida y sale del estado por medio de la línea de continuación que sale del mismo. A diferencia de esto, cuando un proceso entra en un estado no forzado, este ejecuta el código de entrada del estado y se mantiene en él hasta que se activa una interrupción. Para esto, los estados no forzados poseen líneas de interrupción como salidas. Estas líneas de interrupción poseen cabeceras que sirven para asignar un tipo de interrupción a la línea (véase numeral 4).

4. **Código en C:** El último nivel de abstracción de OPNET Modeler es el código en C asignado a las ejecuciones de entrada y/o salida de los estados de las máquinas de estado. En esta parte del programa se utilizan funciones propias de OPNET Modeler, así como también funciones y algoritmos propios del lenguaje de programación C. Para que el código esté mejor organizado, existen en el programa bloques que pueden ser utilizados para poner partes de código repetitivas, como por ejemplo funciones. Estos bloques son:

   SV (*Static Variable*): En este bloque se pueden inicializar las variables de estado que van a ser utilizadas en la simulación, estas pueden ser de distintos tipos como *int, double, Packet* (variable tipo paquete), *Distribution* (variable tipo distribución, utilizada para asignar una FDP a una variable), *char,* etc. Estas variables se caracterizan porque inician automáticamente en 0 (variables no numéricas como *Packet,* son inicializadas en vacío, *null*) y cuando son alteradas en el código, estas conservan este valor hasta la próxima vez que sean alteradas.

   TV (*Temporary Variable*): Este bloque sirve para inicializar variables temporales que van a ser utilizadas en el código. Estas variables, al igual que en SV, pueden ser de tipos como *int, double, Packet, char,* etc. La particularidad de este tipo de variables, es que primeramente, pueden ser inicializadas por el usuario (si no se hace esto, se inicializan automáticamente en 0 o vacío según el tipo de variable), y segundo, que cuando son alteradas, estas conservan su valor únicamente mientras el proceso está ejecutando el código en el cual fueron alteradas; cuando el proceso termina de ejecutar este código, las variables vuelven a su estado inicial.

   HB (*Header Block*): En este bloque es posible declarar las cabeceras que van a ser utilizadas en el código, como por ejemplo las de las líneas de interrupción.

   FB (*Function Block*): Este bloque se utiliza para declarar funciones que se utilizarán en el código. Esto es útil para que cuando se necesite ejecutar una función dentro del código, solo sea necesario llamarla y no tener que declararla constantemente.

1.2 FUNCIONAMIENTO

Servidor

La Figura 1-1 muestra el proceso utilizado para el módulo procesador del servidor de VoD del presente proyecto de grado. En el capítulo 4 de la monografía del mismo se describieron los distintos estados de este proceso. Para contextualizar lo anteriormente expuesto y

teniendo en cuenta la Figura 1-1, se puede decir a modo de ejemplo que cuando el proceso está en el estado init y se completa el tiempo inicial de espera para el manejador de la interrupción *TIMEOUT2*, el proceso identifica el código que posee el manejador y atiende la interrupción asignada a ese código, que en este caso es *TIMEOUT2*. En este punto el proceso ha cambiado al estado TX_P. En el código de entrada de este estado aparece nuevamente el manejador de la interrupción *TIMEOUT2*, el cual esta vez ha sido configurado para que su tiempo de espera sea un valor aleatorio de acuerdo a la FDP correspondiente a las tramas P del video.

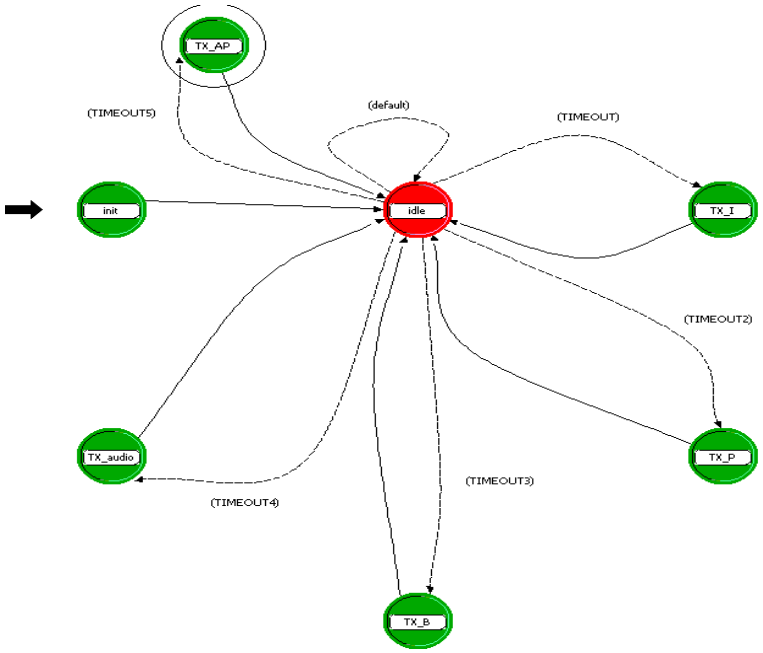


Figura 1-1. Proceso asociado al nodo servidor de VoD

Dentro del código de estos cinco estados, existen líneas que se encargan de asignar tamaños a los paquetes de acuerdo a las FDP de tamaño de paquetes de cada trama. Para lograr esto, primero hubo que crear un paquete personalizado al cual asignarle el tamaño de las tramas. En la Figura 1-2 se puede observar el editor de paquetes de OPNET Modeler y el paquete que ha sido creado. El campo llamado *header* se utiliza para asignar un código al paquete para que pueda ser reconocido por el cliente (o pueda determinar qué tipo de trama es). El campo denominado *payload* es al que se le va a asignar el tamaño del paquete.

Para tal objetivo, como se observa en la Figura 1-3, se asigna a la variable paquete_pk, la cual es de tipo *Packet*, el valor de un paquete sin formato por medio de la función *op_pk_create*. Esta función tiene como parámetro el tamaño del paquete creado, en este caso es *pksize*, que no es más que una variable de tipo *double* a la que se le ha asignado un valor aleatorio proveniente de la FDP correspondiente al tamaño de las tramas P del video. Este valor es multiplicado por ocho, debido a que el simulador trabaja con bits. Una vez creado el paquete paquete_pk, se procede a crear un paquete con formato (el que se creo con el editor de paquetes), el cual se configura con el código de la trama P (2) en el campo

header, y se le asigna al campo *payload* el paquete paquete_pk creado anteriormente. De tal manera que cada vez que el proceso entre en el estado TX_P, este creará un paquete con un tamaño aleatorio correspondiente a la FDP de las tramas P del video.

Luego de haber creado y configurado el paquete, el código se encarga transmitirlo hacia el módulo transmisor por medio del flujo de señal mediante la utilización de la función *op_pk_send*. Los parámetros de esta función son el paquete a enviar y el código del flujo de señal (este código es asignado de acuerdo al número de flujos de salida y entrada unidos al módulo procesador, en este caso el código del flujo de señal es 0, véase Figura 1-3).

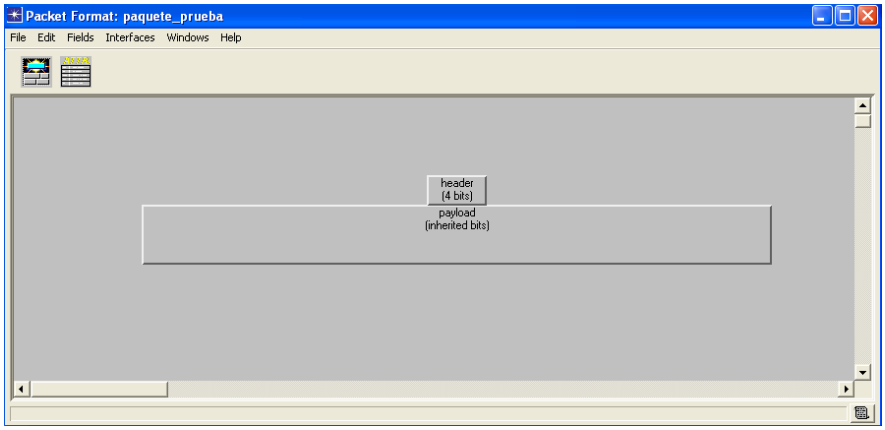


Figura 1-2. Editor de paquetes de OPNET Modeler

```
pksize = op_dist_outcome(weibull3_tamP);
pk = op_pk_create_fmt("paquete_prueba");
paquete_pk = op_pk_create(pksize*8);
op_pk_nfd_set(pk, "header", 2);
op_pk_nfd_set(pk, "payload", paquete_pk);
op_pk_send(pk, 0);
```

Figura 1-3. Fragmento de código del estado TX_P encargado de crear una trama y transmitirla al módulo transmisor del nodo servidor

Cliente

La Figura 1-4 muestra el proceso asignado al módulo procesador del nodo cliente, el cual posee tres estados de transición.

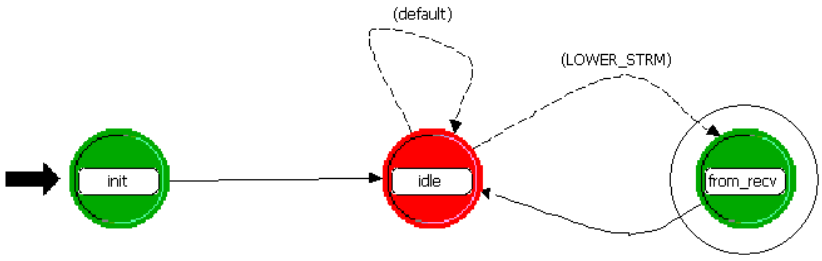


Figura 1-4. Proceso asociado al nodo cliente

El estado init es el estado inicial del proceso. Aquí se han declarado variables de estado correspondientes a las estadísticas tomadas por el cliente. Para poder declarar estas estadísticas es necesario inicializar las variables de cada una de ellas en el bloque SV. Estas variables son del tipo *Stathandle*. En la Figura 1-5 se aprecia la inicialización de la variable *intertimeP*, correspondiente a la estadística que recoge el tiempo entre tramas tipo P.

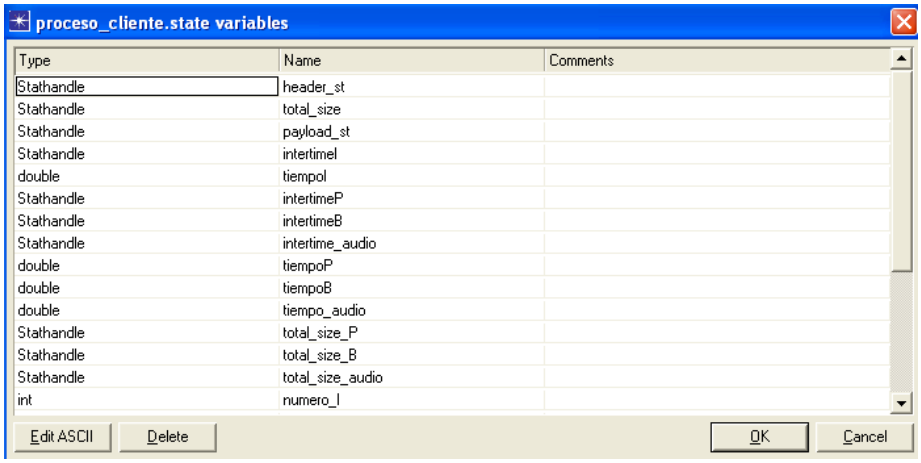


Figura 1-5. Ventana de *State Variables*

Además de ser inicializadas y declaradas, las estadísticas en OPNET Modeler también deben ser configuradas para ser estadísticas locales del nodo en cuestión. En la Figura 1-6, se aprecia que la declaración de la estadística *intertimeP* por medio de la función *op_stat_reg* tiene como parámetro un nombre. Este nombre corresponde a la denominación local de la estadística. Para poder declarar la estadística como una estadística local, es necesario ir a la pestaña *Local Interfaces* del proceso del nodo, e ingresar en la opción *Local Statistics*. En la Figura 1-7 se observa una captura de pantalla de la ventana *Local Statistics* con la configuración para declarar a la estadística *intertimeP* como una estadística local.

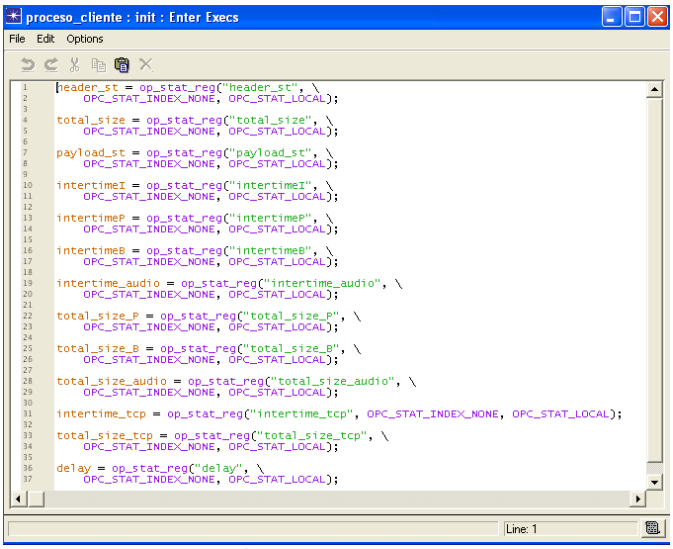
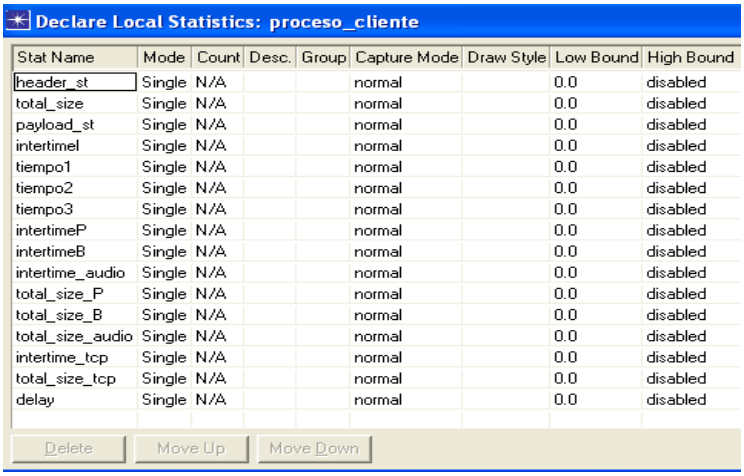


Figura 1-6. Fragmento de código del estado init del proceso del cliente



Declare Local Statistics: proceso_cliente

Stat Name	Mode	Count	Desc.	Group	Capture Mode	Draw Style	Low Bound	High Bound
header_st	Single	N/A			normal		0.0	disabled
total_size	Single	N/A			normal		0.0	disabled
payload_st	Single	N/A			normal		0.0	disabled
intertimeI	Single	N/A			normal		0.0	disabled
tiempo1	Single	N/A			normal		0.0	disabled
tiempo2	Single	N/A			normal		0.0	disabled
tiempo3	Single	N/A			normal		0.0	disabled
intertimeP	Single	N/A			normal		0.0	disabled
intertimeB	Single	N/A			normal		0.0	disabled
intertime_audio	Single	N/A			normal		0.0	disabled
total_size_P	Single	N/A			normal		0.0	disabled
total_size_B	Single	N/A			normal		0.0	disabled
total_size_audio	Single	N/A			normal		0.0	disabled
intertime_tcp	Single	N/A			normal		0.0	disabled
total_size_tcp	Single	N/A			normal		0.0	disabled
delay	Single	N/A			normal		0.0	disabled

Delete | Move Up | Move Down

Figura 1-7. Ventana de *Local Statistics*

Con el fin de recibir los paquetes provenientes del módulo receptor del nodo cliente, se utilizó la función *op_pk_get* en el estado *from_recv*, la cuál tiene como parámetro el código del flujo de la señal del cual se quiere recibir el paquete. Esta función asigna el paquete que ha producido la interrupción a una variable del tipo paquete. Una vez ha sido recibido el paquete, el código se encarga de verificar de qué tipo de trama es, por medio del campo *header* de cada paquete, el cual contiene un número que está entre 1 y 5 correspondientes a cada tipo de trama transmitida por el servidor. Es necesario identificar a qué tipo de trama corresponde cada paquete para poder escribir en las variables estadísticas por medio de la función *op_stat_write*. Para cada tipo de trama, existe un código que ha sido asignado dependiendo de ello (véase Figura 1-8).

```
pk = op_pk_get(LOWER_IN_STRM_INDEX);
op_pk_nfd_get(pk, "header", &header);
op_pk_nfd_get(pk, "payload", &payload);
//op_pk_nfd_get(pk, "timestamp", &timestamp);
timestamp = op_pk_stamp_time_get(pk);

if(header == 1){
op_stat_write(total_size, op_pk_total_size_get(payload));
op_stat_write(intertimeI, op_sim_time() - tiempoI);
tiempoI = op_sim_time();
op_pk_destroy(pk);
numero_I = numero_I +1;
op_stat_write(delay, op_sim_time() - timestamp);
//sprintf(msg, "paquetes recibidos tipo I: %d \npaquetes recibidos tipo P: %d \npaquetes
//op_sim_message("", msg);
}

if(header == 2){
op_stat_write(total_size_P, op_pk_total_size_get(payload));
op_stat_write(intertimeP, op_sim_time() - tiempoP);
tiempoP = op_sim_time();
op_pk_destroy(pk);
numero_P = numero_P +1;
op_stat_write(delay, op_sim_time() - timestamp);
//sprintf(msg, "paquetes recibidos tipo I: %d \npaquetes recibidos tipo P: %d \npaquetes
//op_sim_message("", msg);
}

if(header == 3){
op_stat_write(total_size_B, op_pk_total_size_get(payload));
op_stat_write(intertimeB, op_sim_time() - tiempoB);
tiempoB = op_sim_time();
op_pk_destroy(pk);
numero_B = numero_B +1;
op_stat_write(delay, op_sim_time() - timestamp);
//sprintf(msg, "paquetes recibidos tipo I: %d \npaquetes recibidos tipo P: %d \npaquetes
//op_sim_message("", msg);
}
```

Figura 1-8. Fragmento de código del estado *from_recv*

En la Figura 1-8 se muestra una fracción del código del estado *from_recv*, en el cual se puede apreciar cómo se reciben los mensajes y cómo se escriben las estadísticas. Entre estas estadísticas se ha medido el tiempo entre paquetes de llegada y el tamaño de los

mismos. En la Figura 1-9 se observa el resultado de la estadística tiempo entre paquetes (*intertime_P*) de las tramas tipo P del video1.

La Figura 1-9 corresponde al resultado obtenido en el cliente, de la simulación correspondiente al video1. En la estadística llamada *intertimeP* se han escrito los datos correspondientes al tiempo entre paquetes mediante un algoritmo matemático. En este algoritmo se asigna a una variable de estado del tipo *double*, llamada *tiempoP* para las tramas tipo P, el tiempo actual de la simulación por medio de la función *op_pk_sim_time*, cuando el proceso entra en el estado *from_recv* debido a la llegada de una trama tipo P. Una vez el proceso ha salido de este, la variable de estado conserva su valor. Cuando el proceso vuelve a entrar en el estado *from_recv* debido a la llegada de una trama tipo P, se realiza una comparación del tiempo actual con el tiempo que había sido asignado anteriormente a la variable de estado, de tal manera que se consigue obtener el tiempo entre la llegada de una trama P y la llegada de la siguiente trama P. Luego este tiempo es asignado a la variable de estado de tipo *Stathandle*, *intertime_P*, por medio de la función *op_stat_write*. De tal manera que se consigue tener en esta estadística el tiempo entre cada par de tramas tipo P. Para los otros tipos de tramas el procedimiento es el mismo.

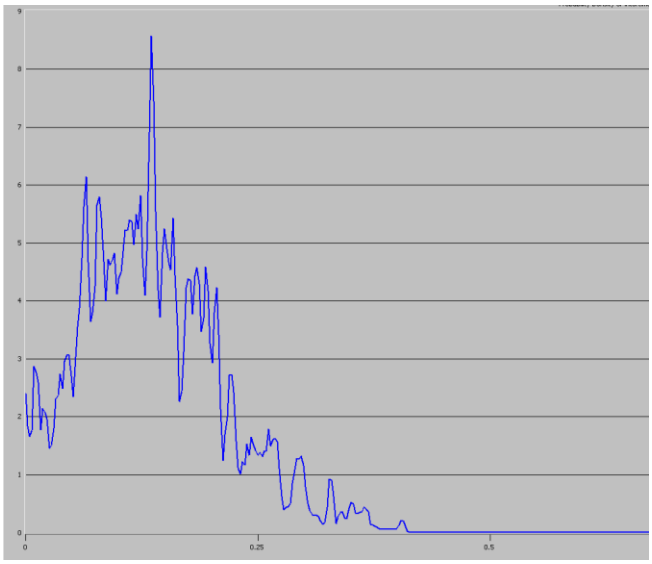


Figura 1-9. Estadística *intertimeP*, tomada del cliente, de los paquetes pertenecientes a las tramas tipo P del video1

Una vez guardado el paquete proveniente del campo *payload* en la variable de tipo *Packet*, se procede a obtener el tamaño del mismo por medio de la función *op_pk_get_size*, la cual asigna el tamaño del paquete requerido a una variable que puede ser de tipo *int* ó *double*. Esta función tiene como parámetros: la variable de tipo *Packet*, la cual contiene el paquete al que se le desea sustraer el tamaño. Después de obtener el tamaño del paquete, se procede a escribirlo en la estadística *total_size_P* por medio de la función *op_stat_write*. En la Figura 1-10 se puede apreciar la gráfica de tipo FDP resultante de esta estadística. Esta gráfica

muestra la FDP del tamaño de los paquetes pertenecientes a las tramas tipo P del video1, es decir, muestra el tamaño de las tramas tipo P que llegan al cliente para el video1.

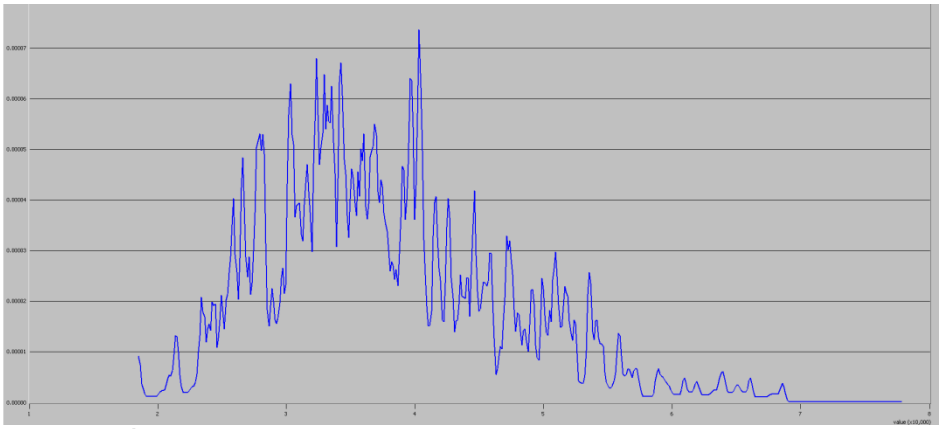


Figura 1-10. Estadística *total_size_P* tomada del cliente, de los paquetes pertenecientes a las tramas tipo P del video1.

2 ns-3

2.1 CONCEPTOS

A continuación se revisan algunos términos que son comúnmente usados en la ingeniería de redes pero que tienen un significado específico en ns-3 [2].

- **Nodo**

  ns-3 es un simulador de redes, y no precisamente un simulador de internet, por eso no se usa exclusivamente el término "*host*", dado que este término está cercanamente asociado con el concepto de internet y sus protocolos. En lugar de eso, se asume un término más genérico que es también usado por otros simuladores; el nodo.

  Se pensaría en un nodo como un computador en el cual se puede adicionar funcionalidades. Se pueden adicionar cosas como aplicaciones, pilas de protocolos y tarjetas de periféricos con sus *drivers* asociados, para permitir que el computador haga un trabajo útil. Se usa el mismo modelo básico en ns-3.

  En la construcción de topologías aparecen tipos de nodos con funciones específicas como son el nodo switch y el nodo router. Si bien estos dos tipos de nodos son abstracciones del mundo real, estos no guardan ninguna relación directa con las características físicas o de procesamiento de datos de algún dispositivo de un fabricante en especial.

- **Aplicación**

  En ns-3 no existe un concepto real de sistema operativo y tampoco el concepto de niveles de privilegio o llamadas de sistema. Sin embargo se tiene la idea de aplicación. Solo aplicaciones software corren en los computadores para desarrollar tareas en el "mundo real", las aplicaciones de n-3 corren en los nodos de ns-3 para manejar las simulaciones en el mundo simulado.

- **Canal**

  En el mundo real se puede conectar un computador a una red. Frecuentemente el medio sobre el cual los datos fluyen en estas redes se llama Canal. En el mundo simulado de ns-3, se puede conectar un nodo con un objeto representando un canal de comunicación. Los canales son especializados dependiendo de la tecnología en la que se trabaje.

- **Dispositivo de red**

  En ns-3 la abstracción de dispositivo de red cubre ambos, *drivers* y *hardware* simulado. Un dispositivo de red se instala para habilitar la comunicación de un nodo con otros, a través del canal. Los dispositivos de red son también especializados dependiendo de la tecnología en la que se trabaje.

- **Ayudantes de topologías**

  Se podrían emplear varias instrucciones de C y algunas más, propias de ns-3 para conectar dispositivos de red a los nodos, dispositivos de red a los canales, asignar direcciones IP, conectar múltiples dispositivos con canales multipunto y después conectar redes individuales a otras redes más grandes, etc. En redes a gran escala, realizar estas tareas solamente con instrucciones de esta naturaleza no es conveniente ni práctico. Por estos motivos ns-3 provee ayudantes de topologías (*topology helpers*) que combinan todas esas instrucciones C y de ns3 en un modelo fácil de usar.

- **Fuente de trazas y cernidero de trazas**

  Las fuentes de trazas (*TraceSources*) son entidades que pueden señalizar eventos que ocurren y que dan acceso a datos interesantes. Por ejemplo una fuente de trazas puede indicar que un paquete se recibió por un dispositivo de red o que un paquete se perdió.

  Las fuentes de trazas no son útiles por sí mismas, estas deben estar conectadas a funciones que puede hacer algo realmente útil con la información que la fuente de trazas brinda. Estas funciones donde se consume la información de la fuente de trazas se llaman cernideros de trazas (*TraceSinks*). Los cernideros de trazas son generadores de eventos. Esta división explícita permite que un número amplio de

fuentes de trazas sean puestas en partes del sistema de donde se puede obtener algún tipo de información útil, según sea el caso.

2.2 PLANTEAMIENTO DE LA SOLUCIÓN PARA EL MODELO DE TRÁFICO

Como ya se dijo antes, para el desarrollo de este trabajo de grado se decidió implementar una simulación de la red que modelara el comportamiento del tráfico de los servicios existente en el servidor del ST-CAV en el laboratorio de televisión digital de la Universidad del cauca.

Antes de haber llegado a una solución definitiva para la implementación de un modelo de red que sea satisfactorio para este trabajo, se habían contemplado anteriormente dos opciones más; que parecían bastante llamativas e innovadoras y que por motivos técnicos se descartaron. A continuación se da una corta descripción de estas dos alternativas y también la razón por la cual no fueron tomadas como alternativas definitivas. Inmediatamente después de se explica la opción que se tomo como solución definitiva.

- **Red emulada utilizando contenedores virtuales (maquinas virtuales de Linux)**

  Este escenario consistía en modelar una red de ns-3, a la que se conectaban maquinas virtuales. Así, estas podrían realizar las tareas de envió y recepción de datos dentro de esta red. En primer lugar se realizaron pruebas conectando dos maquinas virtuales a una red de ns3, en la cual las dos maquinas estaban en el mismo dominio. El resultado de esa primera prueba fue exitoso.

  Pero cuando las maquinas virtuales se encontraban en dos dominios diferentes los paquetes no pudieron pasar a través del enrutador para alcanzar el destino en la otra máquina. Para descartar cualquier mal funcionamiento con la red, se reemplazaron las maquinas virtuales por nos nodos de ns-3 y se generó tráfico desde estos. De este modo se comprobó que la red no tenía ningún problema.

  Al final se concluyó que los contenedores no son compatibles cien por ciento con las emulaciones en ns-3 en términos de enrutamiento.

- **Red simulada utilizando la aplicación *UdpTraceClientServerApplication***

  Después de haber descartado la anterior opción, se trabajó sobre la aplicación *UdpTraceClientServer*. Esta aplicación ha sido desarrollada específicamente para modelar el envió de MPEG-2 y MPEG-4 sobre una red de ns-3. Lo que hace esta aplicación es generar tráfico como si se tratara de un servidor de VoD, leyendo un archivo de texto organizado en columnas. Este archivo tiene información sobre el tipo de trama que se envía (trama I, P, B o de audio), el tiempo en que fueron codificadas, tamaño y tiempo de llegada. Por este motivo esta parecía ser una opción bastante atractiva para el desarrollo de este trabajo de grado.

Posteriormente se realizaron algunas pruebas de funcionamiento con archivos de texto que están disponibles en la página web de la Universidad Técnica de Berlín. Estos archivos se obtuvieron de algunos programas de televisión conocidos y películas famosas.

Para comprobar el correcto funcionamiento de la aplicación, se capturó al tráfico en la red ns-3 y se comprobó la correspondencia entre el tráfico capturado y el tráfico que debía ser generado a partir de los archivos de texto.

Entonces se procedió a analizar las trazas de tráfico de los 3 videos propuestos para este trabajo, y generar los archivos de texto con las características requeridas para ser usados por la aplicación *UdpTraceClientServer* en la simulación de ns-3.

Posteriormente se introdujeron en el archivo, se capturó y analizó el tráfico dentro de la simulación para comprobar la correspondencia del tráfico capturado con el que debía ser generado por aplicación; pero se descubrió que no había correspondencia. Al mirar detalladamente las capturas se observo un mal funcionamiento de la aplicación cuando las tramas de video son demasiado grandes. Este mal funcionamiento era básicamente una demora en llegar al estado estable causado por el tamaño de las trama. Esto al final introdujo mucho tráfico inservible que alteraba el comportamiento de la simulación y perdía relación con el tráfico capturado y analizado. Por esta razón también se descarto esta opción.

Finalmente se propuso la tercera y última opción como solución definitiva, que se describe a continuación.

- **Red simulada con la aplicación *OnOffApplication***

  Esta Aplicación es un generador de tráfico que sigue un patrón *OnOff*. Después de que la aplicación empieza a correr, los estados *On* y *Off* se alternan. La duración de cada uno de estos estados está determinada por el valor de las variables *OnTime* y *OffTime*. Durante el estado *Off* no hay tráfico generado, por otro lado, durante el estado *On* si se genera tráfico.

REFERENCIAS

[1] http://www.opnet.com/solutions/network_rd/modeler.html Sitio web del simulador OPNET Modeler

[2] http://www.nsnam.org Sitio web del simulador ns-3