EFFECT OF TRANSMISIÓN IMPAIRMENTS ON TCP WINDOW IN A WIRELESS DIFFSERV ENVIRONMENT

(Efecto de los inconvenientes de transmisión en ventana TCP en un entorno inalámbrico DIFFSERV)

En las últimas décadas los sistemas radio-móviles han alcanzado aplicaciones a nivel de usuario muy superiores a los de la telefonía fija.

Servicios como el correo electrónico, el intercambio de archivos y también el World Wide Web, que hoy son utilizados cotidianamente por millones de personas en todo el mundo a través del teléfono móvil, requieren un tipo de transmisión de datos que sea fiable desde el punto de vista de la integridad de los datos. Igualmente, dentro de unos límites razonables, esos servicios no consideran los tiempos de transmisión como un parámetro crítico para las aplicaciones.

La arquitectura de red Differentiated Services es la más prometedora para la solución del problema de la calidad de Servicio en Internet gracias a su sencilla implementación, elevada escalabilidad y posibilidad de coexistencia con las redes anteriores.

El objetivo del proyecto es estudiar los efectos del retardo de cola y la respuesta a las expectativas de la arquitectura de diferenciación de servicios para garantizar la calidad de servicio en una red UMTS con router de acceso a la red (core network) completamente IP mediante simulación en ambiente NS2.
En los siguientes apartados se describirán brevemente las características de la arquitectura de las redes Differentiated Services y el modelo teórico de los routers Diffserv. Además se expondrá un escenario de simulación elegido para la realización de dicho estudio.

El entorno de simulación

El módulo DiffServ ha sido desarrollado por Nortel Networks y ha sido añadido a NS2 el 2 de Noviembre de 2000. La funcionalidad DiffServ está compuesta por cinco módulos subclases de Queue. Las instancias de esta clase son desarrolladas en el elemento enlace (link) el interfaz de salida de un router.

Los cinco módulos son:

- La clase dsREDQueue que modela la funcionalidad de base de un router DiffServ, como la cola múltiple con la cual se pueden implementar las diversas clases de servicio; esta es la clase madre de todas las demás, y contiene todos los parámetros que son comunes a las otras.
- La clase redQueue con la cual gestionar los mecanismos de control de la congestión de cada cola.
- La clase CoreQueue que modela la funcionalidad de un router intermedio de la red (core router).
- La clase EdgeQueue que modela la funcionalidad de un router frontera (edge router).
- La clase Policy Classifier, que es una subclase de la precedente, con la cual se gestionan las funcionalidades de contar y marcar los flujos.

La estructura de la cola proporciona la clase dsREDQueue consiste en cuatro colas físicas, cada una de las cuales contiene tres colas RED “virtuales” que se refieren a los diferentes niveles de precedencia. Cada cola física corresponde a una clase de tráfico y cada combinación de una cola y de un nivel de precedencia es asociada a un codepoint: los paquetes se encolan en base al codepoint con el que están marcados.

La clase dsREDQueue contienen también una estructura de datos llamada PHB Table, que contiene un vector con tres campos: Codepoint, Class (Physical Queue) y Precedence (Virtual Queue).

La PHB Table permite a los routers de frontera e intermedios de la red mapear los codepoint con las diferentes colas y los diversos niveles de precedencia.

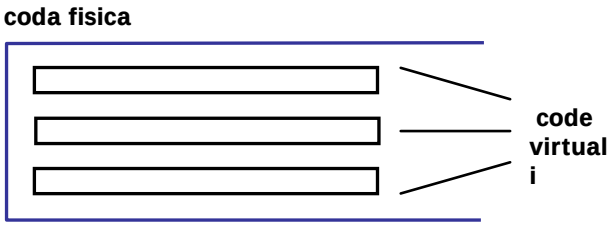

Figura 1: Ejemplo de una cola física con tres colas virtuales.

Haciendo uso del siguiente código configuramos un router frontera:

```
      $dsredq  addPolicyEntry  [$n1  id]  [$n2  id]  TokenBucket  10
1000000 10000
      $dsredq addPolicerEntry TokenBucket 10 11
```

Con el comando addPolicyEntry se añade una línea a la Policy Table, que es un vector donde son memorizadas las informaciones necesarias para la caracterización del tráfico, como los nodos fuente y destino del flujo, el tipo de contador y los parámetros con los que son medidos, los codepoint iniciales asignados y otras informaciones de estado.

Con el comando addPolicerEntry, viene añadida una línea en el array Policer Table que contiene el mapa de los codepoint iniciales y de aquellos de asignar a los paquetes cuanco exceden los límites estables de los respectivos contadores.

El entorno de simulación

El escenario de simulación en ambiente NS2 para verificar los efectos del retardo de cola y la correspondencia con las expectativas de la arquitectura Differentiated Services para garantizar la calidad de servicio de una red UMTS con red interna completamente IP será descrito en esta sección. Los servicios

de calidad diferente deberán coexistir en una misma red y compartir los recursos como, por ejemplo, el ancho de banda de un link.

En particular se ha querido demostrar cómo podrá ser garantizado un servicio con requisitos de calidad muy exigentes, por ejemplo, una videoconferencia a costa de aplicaciones insensibles al retardo como la lectura del correo electrónico.

En este trabajo de caracterización se intentará garantizar las exigencias de los servicios con requisitos más exigentes de los sercicios usando la funcionalidad de avance y acondicionamiento puesto a disposición del módulo DiffServ implementado en la versión 2.29 de NS2, descrito en el paragrafo precedente. La utilización de la arquitectura DiffServ a los fines de la QoS se verá verificada evaluando el *througput* de la red simulada y comparándolo con aquellos obtenidos recorriendo únicamente la arquitectura best effort tradicional.

Topología de la red simulada

La topología de la red utilizada para este escenario de simulación está mostrada en la figura 1 y compuesta de diecisiete nodos, de los cuales tres son router y diez host. En la figura se muestra el dominio DiffServ que está compuesto de dos nodos frontera (edge router), marcados con los números 1 y 2 y del nodo interno marcado con el número 0, que es el router interno del dominio.

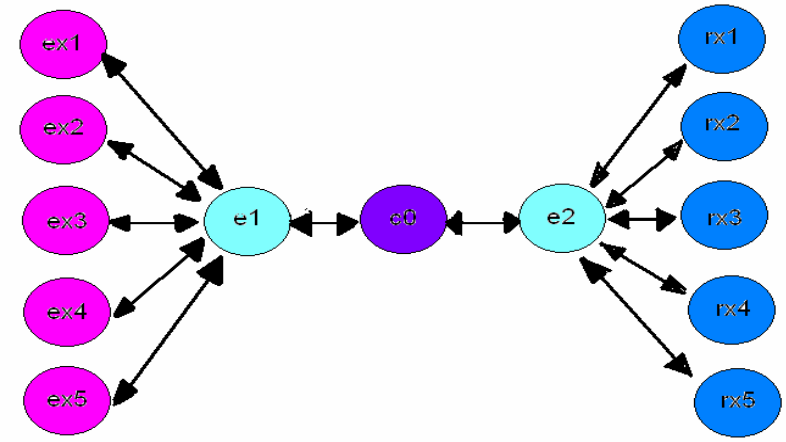

Figura 2: Esquema de la red

En esta topología es posible encontrar:

- La red UMTS, compuesta por los nodos del dominio DiffServ (nodos 0, 1 y 2).
- La red de acceso radio UMTS, constituida por los nodos 8, 9, 10, 11 y 12 unidos al router de frontera 2.
- Las fuentes, situadas en la red Internet, le sorgenti, situate nella rete Internet, por lo tanto externas a la red UMTS, representadas por los nodos 3, 4, 5, 6, 7 y unidos al router de frontera 1.

Los nodos router están conectados entre ellos con los enlaces simples y todos los nodos host estan unidos entre ellos con enlaces duplex de modo que los *acknowledgement* de las fuentes que usan el TCP como agente de transporte tienen un recorrido de retorno.

Las fuentes de tráfico

En el esquema de la red se muestra el recorrido de los flujos de tráfico presentes en la red. En base al recorrido de los flujos y a cómo están dimensionados los enlaces entre los router, esta claro que el cuello de botella de la red interna se forma entre los router 1 y 0. Entonces, serán las colas de estos dos routes aquellas que introducirán retardo y causaran la pérdida de paquetes en condiciones de sobrecarga de la red.

Analizaremos ahora con más detalle las características de las fuentes y de los flujos de tráfico elegidos para esta simulación.

Entre los nodos 3 y 8 hay un tráfico de tipo Premium de PHB *Expedited Forwarding* en ambiente DiffServ, este tráfico podría ser representativo de una videoconferencia y generado por una aplicación CBR (*Constant Bit Rate*) que

produce paquetes del tamaño de un *frame*, los cuales usan el agente de transporte RTP que implementa el homónimo protocolo de transporte para aplicaciones *real time.*

En la simulación son presentadas tres clases de tipo *Assured Forwarding* (avance garantizado) llamadas AF1, AF2 y AF3, cada una de ellas asociada a un tipo de tráfico diferente. Con un grupo de clases de este tipo puede ser implementado un modelo de servicio conocido como *Olympic service*, que consiste en tres clases con niveles de prioridad decrecientes como gold, silver y bronze.

A la clase AF1 gold está asociado un flujo de datos que suponemos sea tráfico multimedia. Los paquetes de este flujo, cada uno de 1000 bytes, son generados por una fuente CBR y usan como protocolo de transporte el UDP, generado por el agente de transporte del simulador con mismo nombre.

A la clase AF2 silver es asociado el servicio para la transmisión de archivos con grandes dimensiones. Los paquetes son generados por la aplicación FTP y emplean como agente de transporte el FullTCP, que implementa la versión Reno de TCP de modo que haya garantía sobre la efectiva asignación de los datos. Como dimensiones de los paquetes son elegidos aquellos por defecto del simulador: 576 bytes de los cuales 536 son de datos y 40 de cabecera TCP.

A la clase AF3 bronze está asociado el servicio de *web server.* El flujo de datos producido por un servicio de este tipo está simulado mediante el generador de tráfico Pareto, que usa una distribución paretiana, y es transportado también, por un agente FullTCP. En las simulaciones efectuadas se ha elegido el segundo tipo porque ha sido demostrado que la distribución paretiana aproxima mejor que la exponencial el tráfico presente en Internet.

A la clase BE, finalmente, se ha asociado el tráfico de web browsing y todo el tráfico generado por los reconocimientos (ack) de las conexiones TCP. Este

tráfico es generado con una aplicación Pareto y es transportado por un agente FullTCP.

Configuración de los routers

Para la configuración de los router, se necesita naturalmente distinguir los dos tipos de servicio, Best Effort y DiffServ, que son implementados y comparados en las simulaciones efectuadas.

El caso Best Effort es más bien simple porque todos los router son configurados con una cola FIFO y el algoritmo de descarto selectivo DropTail (al que se le indica el límite de la cola, que por defecto es de 50 paquetes), a partir de este valor todos los paquetes que lleguen serán descartados. En base a la topología y a la configuración de los enlaces elegidos, se tendrán paquetes descartados solamente en los routers 1 y 0. Notar que Network Simulator permite asignar el límite de la cola en función del límite de la cola y no, más realista, en función del número de bytes en el buffer.

La elección de los parámetros de acondicionamiento del tráfico y de la configuración de los router en el caso DiffServ está mucho más completa para el alto número de variables en juego y por la ausencia de información de las especificaciones para su configuración.

En cuanto a lo concerniente a las condiciones del tráfico se ha elegido medir todos los flujos con sus Token Bucket en modo de distinguir, marcando los paquetes con DSCP diversos, dos niveles de precedencia para los paquetes considerados "*in profile*" o "*out of profile*" de cada flujo. El fragmento de código mostrado a continuación como ejemplo, muestra la configuración de los parámetros específicos del flujo de la videoconferencia.

```
   $queue_e1_c0 addPolicyEntry [$ex1 id] [$rx1 id]  TokenBucket 36
$rrx_1 1536

   $queue_e1_c0 addPolicerEntry TokenBucket 36 38
```

```
   $queue_e1_c0 addPHBEntry 36 0 0
   $queue_e1_c0 addPHBEntry 38 0 1
```

El número de colas físicas presentadas en los router se han elegido en base al número de flujos gestionados por cada uno de ellos, mientras que el número de colas virtuales y, por tanto, los niveles de precedencia, para cada cola física es fijo para todas las colas. Para el descarto selectivo de los paquetes se ha elegido WRED para todas las colas. A modo de ejemplo se muestra la configuración para el buffer que gestiona el flujo de videoconferencia.

```
$queue_e1_c0 setMREDMode WRED 0
$queue_e1_c0 configQ 0 0 5 10 0.1
$queue_e1_c0 configQ 0 1 0 0 1.0
```

La elección del algoritmo de scheduling (programación de los eventos) para la cola está hecho siguiendo las indicaciones encontradas en literatura. El manejador de eventos prioritario (`$queue_e1_c0 setSchedularMode PRI` en el archivo de configuración) es el más indicado para garantizar al servicio premium un bajo retardo de propagación y bajas oscilaciones en el tráfico (jitter). Para evitar el bloqueo de los flujos con prioridad más baja se ha hecho una asignación estática de la banda residua, indicando para cada cola la tasa asignada gracias al comando `addQueueRate`.

La salida a través de internet del interfaz radio UMTS

Analizamos ahora más en detalle las características de las salidas a través del interfaz radio UTRAN y de su comportamiento cuando recibe un paquete.

El interfaz radio está simulado como un enlace entre dos nodos con encolamiento DropTail, pero añadiendo un simple modelo de error markoviano para simular el ruido en el canal. El retardo sobre la línea tiene dos componentes:

La primera componente cuenta los retardos de propagación y de elaboración y es un tiempo constante, mientras que la segunda tiene en cuenta los retardos de transmisión (ARQ, *Automatic Repeat reQuest*) a nivel RLC (*Radio Link Control*). Como se ve en el siguiente código la configuración de los enlaces entre los router de frontera, e2, y el nodo móvil, rx1, no es diferente a los casos precedentes salvo en la capacidad del canal reducida a 64 kbit/s. La variable *delayRLC_* contiene la componente fija del retardo mientras que la componente variable es introducida con el comando *dynalink* asociado al enlace entre los nodos e2 y rx1, como se ve en la segunda línea. En este modo se especifica que sobre un enlace particular se necesita tener la cuenta de los errores introducidos en el canal y del tiempo necesario para la transmisión.

```
$ns duplex-link $rx1 $e2 64kb $delayRLC_ DropTail
$ns dynalink $e2 $rx1 $cap_
```

Las modificaciones efectuadas no requieren solo el interfaz TCL del simulador, para ello se dispone del código C++. El comando TCL *dynalink* se refiere a una nueva clase: *Dynalink*, añadida al compilador C++ para ser usada

por el método *recv(Packet* p, Handler* h)* de la clase Delay. Si el paquete recibido por el router de frontera 2 tiene como destinatario un nodo con un enlace dinámico con interfaz radio a través de los nodos 8, 9, 10 ,11 e 12 se divide la información en diferentes partes llamadas PDU (Payload Data Unit) de 40 bytes. El envío de las PDU está regulado por una ventana (que indica el número de PDUs que se pueden transmitir en un time slot) y utiliza los reconocimientos (acknowledgement) ya sean positivos o negativos, para comunicar a la fuente el resultado de la transmisión. Si cualquier PDU se pierde en la transmisión, o no se recibe su correspondiente *acknowledgement*, el router reenvia esta PDU perdida en el próximo *time slot,* ocupando la posición que ocupaba en el time slot precedente. Se actúa así con el fin de que todas las PDUs del paquete sean recibidas correctamente por el destinatario. Si un paquete no ocupa todas las posiciones de un time slot para completar su transmisión, las posiciones vacías que faltan son ocupadas con las de la PDU del siguiente paquete a transmitir. En la siguiente figura se muestra el funcionamiento:

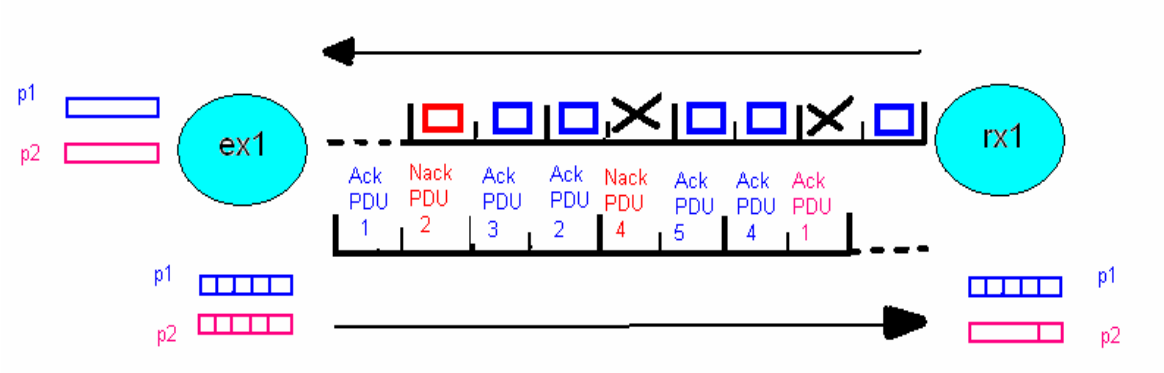

Figura 3: Funcionamiento del core router

En el simulador cada vez que se recibe una trama se llama al método *recv()* que empezaría transmitir en un time slot nuevo aunque en el anterior hayan quedado posiciones vacías generando un retardo erróneo ya que se cuentan tiempos de transmisión de PDUs que no han sido necesarios. Para solucionar este problema a la hora de llevar la cuenta de los *time slots* ocupados por una misma trama se cuentan, además de las PDUs de las que consta, las posiciones ocupadas por las PDUs fallidas y que necesitan retransmisión. Para simular el que una trama empiece a transmitirse en mitad

de un slot anterior (en las posiciones que quedarían vacías) se tiene en cuenta las posiciones que ha dejado libre la trama anterior y se crea una ventana de ese tamaño para la retransmisión de las primeras PDUs de la segunda trama, a continuación se usan ventanas con el tamaño apropiado para la capacidad del canal.

Éste tamaño de ventana se calcula dividiendo la capacidad del canal por el tamaño de una PDU (40bytes), es decir, el tamaño de la ventana define cuantas unidades pueden ser transmitidas en el mismo *time slot.*

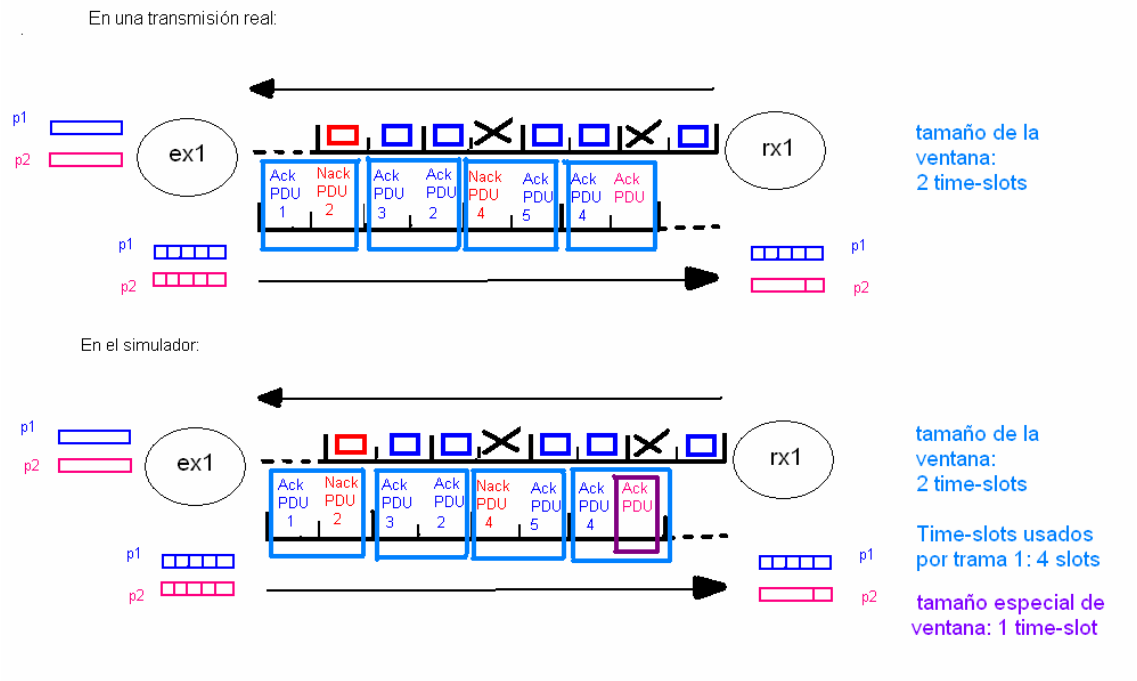

Figura 4: Funcionamiento y simulación del core-router

Como ya señalamos antes, es necesario introducir un modelo de Markov para simular un canal real en el cual existe la posibilidad de que un paquete se pierda en la transmisión. Para nuestro objetivo son suficientes dos estados, en los cuales los valores de probabilidad se encontrarán en los estados *Good*, en *Bad* o puede que la transmisión sea errada en alguno de los dos estados cuyas probabilidades son introducidas mediante los siguientes comandos:

```
$ns probState $e2 $rx1 $probState_
$ns probDelayGood $e2 $rx1 $probDelayGood_
$ns probDelayBad $e2 $rx1 $probDelayBad_
```

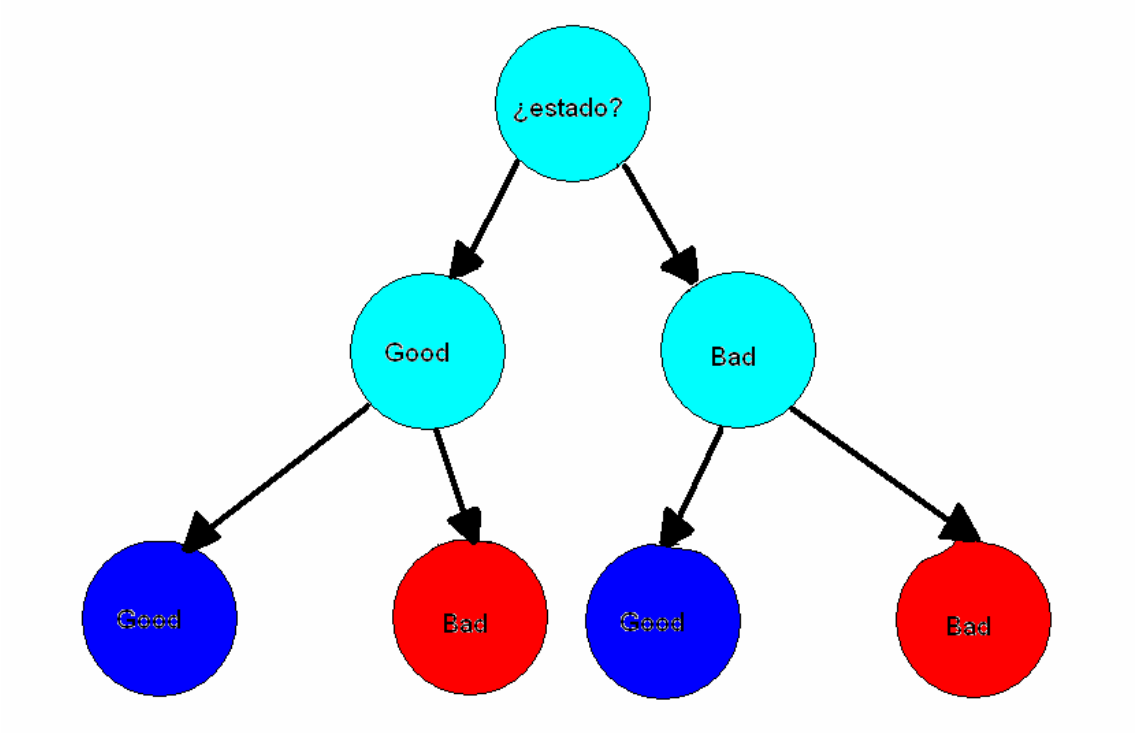

Figura 5: Modelo Markov de 2 estados

Para hacer las simulaciones en este proyecto usamos los valores de probabilidad del artículo:

Francesco Vacirca, Andrea De Vendictis, Alfredo Todini, Andrea Baiochi: “On the effects of ARQ mechanisms on TCP performance in wireless environments”.

Análisis de los resultados

El resultado de la simulación de NS2 es un archivo, de nombre *trace file*, el cual devuelve de modo esquemático todos los eventos acaecidos durante la simulación, siguiendo el orden de la siguiente tabla:

Columna	1	2	3	4	5	6	7	8	9	10	11	12
Inform.	Action	Time	n1	n2	Ptype	Dim	Flag	Fid	Src	Dst	Sn	uid
Esempio	+	0.125	6	0	Tcp	40	-----	5	6.0	11.0	0	0

Tabla 1: Disposición del trace file.

Brevemente, el significado de cada columna es:

1- Acción: Indica la entrada o salida de un paquete a una cola.

2- Tiempo: Instante de tiempo en el cual ocurre el evento especificado en la columna precedente.

3 y 4- n1 u n2, son los dos nodos que identifican el enlace que atraviesa el paquete en ese momento.

5- Ptype: tipo de paquete.

6- Dim: dimensión de paquetes, en byte.

7- Flag: bits de señalización utilizados en TCP.

8- Fid: identificador de flujo.

9- Src: Nodo fuente.

10- Dst: Nodo destino.

11- Sn: Número progresivo asignado al paquete del protocolo utilizado para el transporte.

12- Uid: Identificativo único asignado a cada paquete, diferente del anterior.

Usando un programa simple en C se reorganiza la información precedente y se extrae el throughput de cada flujo usando la siguiente formula:

```
throughput = (dim*8)/(time- timePrima)
```

- Dim: columna 6, es decir, dimension del paquete.
- Time: Instante en el cual se calcula el throughput.
- TimePrima: Instante de llegada del paquete precedente.

La monitorización de los parámetros se ha efectuado sobre dos escenarios con diversa capacidad de canal:

Collegamento tra		
Sorgente e Router (Kb)	Router e Router (Kb)	Router e interfaccia radio (Kb)
70	200	64
160	500	144

Como se puede ver la capacidad entre el router y el nodo móvil esla de un canal UMTS, mientras que la fuente tiene una capacidad superior a la del canal para tener siempre en el buffer un paquete para transmitir. La capacidad del *core netxork UMTS* (los enlaces centrales) es menor respecto a los recursos del canal: se desea simular el caso en el cual los recursos han sido asignados a otras comunicaciones no examinadas pero que tienen la función de tráfico sobrante. El cuello de botella se forma en el nodo DiffServ cuando se efectúa la decisión sobre cómo se dividen los recursos entre las conexiones.

Se realizarán dos simulaciones: una con el escenario DiffServ y otra sin diferenciar los servicios. En el escenario DiffServ la tabla de los Per Hop Behaviour es la siguiente:

Ex1 -> Rx1	Premium Service
Ex2 -> Rx2	AF1 (Gold)
Ex3 -> Rx3	AF2 (Silver)
Ex4 -> Rx4	AF3 (Bronze)
Ex5 -> Rx5	Best Effort

Tabla 3: PHB

CAPACIDAD DEL CANAL RADIO A 64 Kb/s

- Con *Diffserv*:

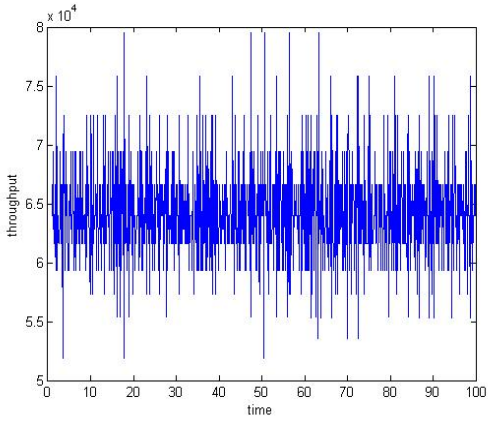

Fig. 1: Enlace entre e2-rx1 (P. S.)

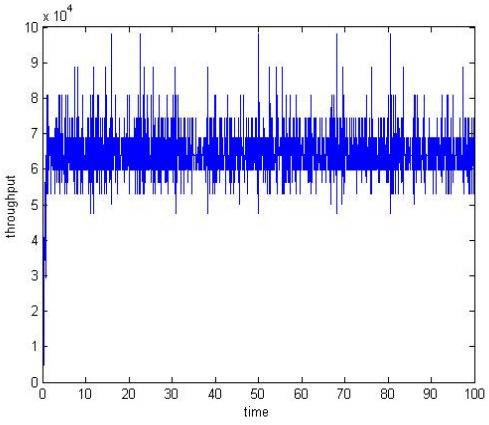

Fig. 2: Enlace entre e2-rx3 (AF2)

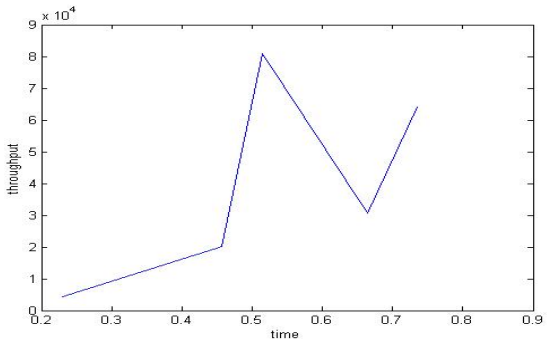

Fig. 3: Enlace entre e2-rx5 (Best Effort)

- Sin *Diffserv*:

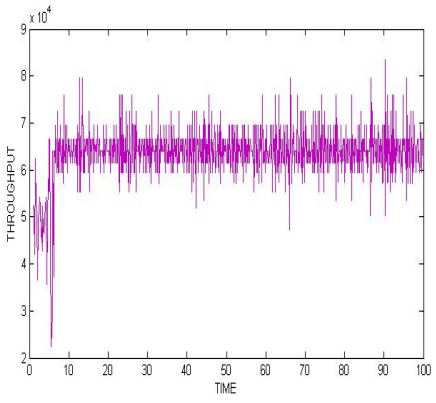

Fig. 4: Enlace entre e2-rx1

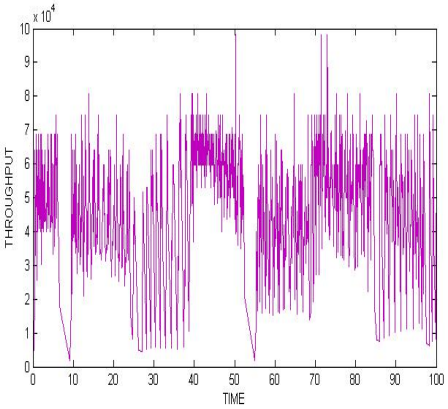

Fig. 5:Enlace entre e2-rx3

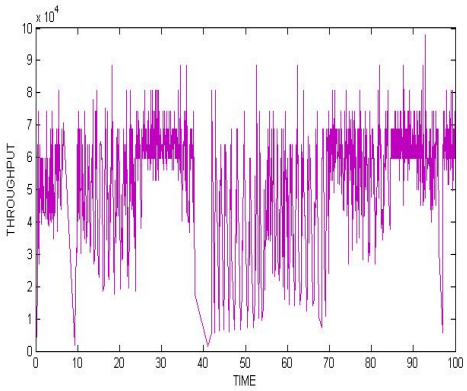

Fig. 6:Enlace entre e2-rx5

Vemos que por el enlace Premium Service se obtiene un significativo mejoramiento con respecto a los otros dos flujos, disfrutando de un uso eficiente y continuado del canal radio. También el flujo AF2 mejora tal y como se podía esperar aplicando una diferenciación de servicios. Estas mejoras se obtienen a espensas de los servicios best effort que ven reducido su throughput. En éste último caso la situación más favorable es aquella que no prevee la utilización de DiffServ porque toda la banda disponible está dividida en partes iguales entre los enlaces sobre el nodo de salida.

CAPACIDAD DEL CANAL RADIO 144 Kb/s

- Con *Diffserv*:

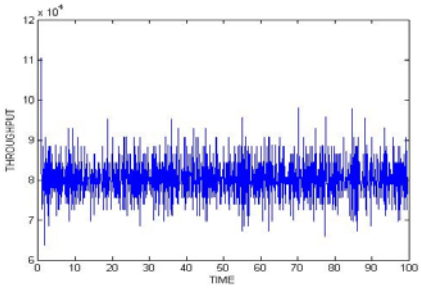

Fig.7: Enlace entre e2-rx1 (P. S.)

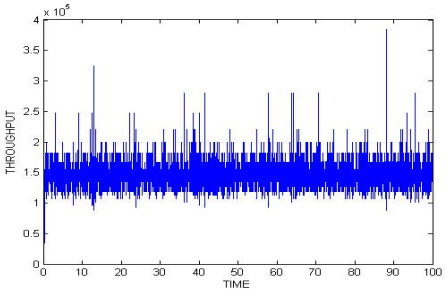

Fig.8: Enlace entre e2-rx3 (AF2)

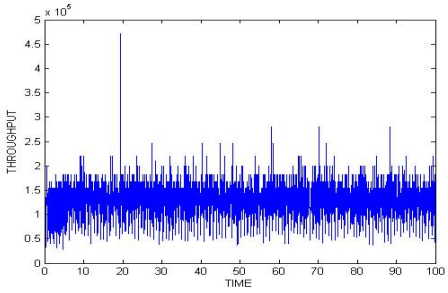

Fig.9: Enlace entre e2-rx5 (Best Effort)

- Sin *Diffserv*:

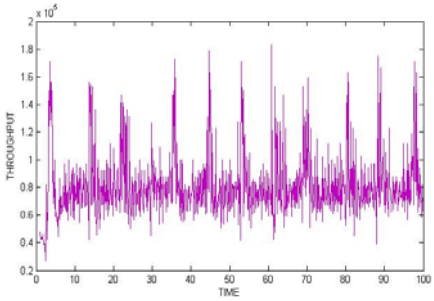

Fig.10: Enlace entre e2-rx1

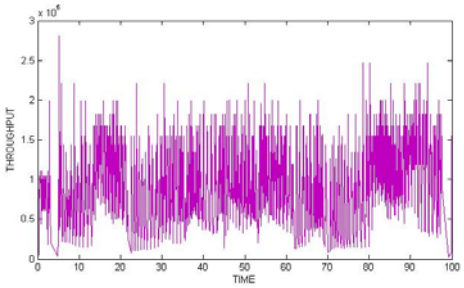

Fig.11:Enlace entre e2-rx3

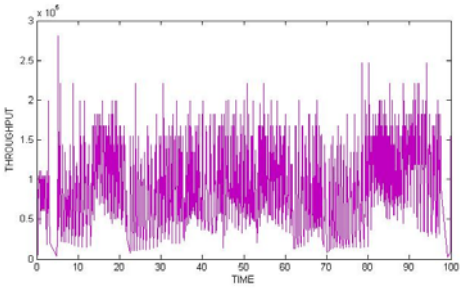

Fig.12:Enlace entre e2-rx5

Escalando todas las capacidades del escenario con los valores vistos en la tabla. Los resultados como era previsible son los mismos que en el caso precedente.

ENLACES CON DIFERENTE CAPACIDAD

Se elige usar cuatro fuentes a 140 kb/s y 4 canales radio a 144 kb/s. El dominio DiffServ proporciona solo dos clases de servicio: Best Effort y Premium Service, y tiene una capacidad de 350 kb/s, por lo que los recursos son escasos para garantizar todos los flujos.

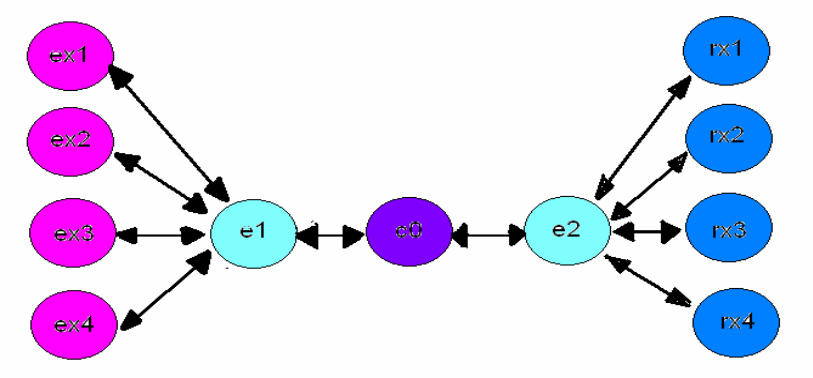

Fig.13: Esquema de la red

A los flujos se les asignan varias clases siguiendo el siguiente esquema:

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
RX1	Premium Service	Best Effort	Best Effort	Best Effort
RX2	Premium Service	Best Effort	Best Effort	Premium Service
RX3	Premium Service	Premium Service	Premium Service	Premium Service
RX4	Premium Service	Best Effort	Premium Service	Premium Service

- Escenario 1:

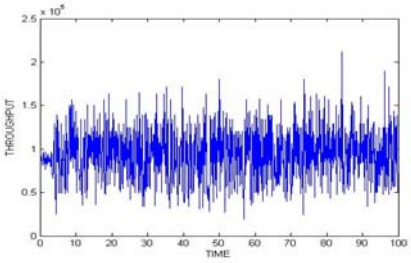

Fig.14: Enlace entre e2-rx1

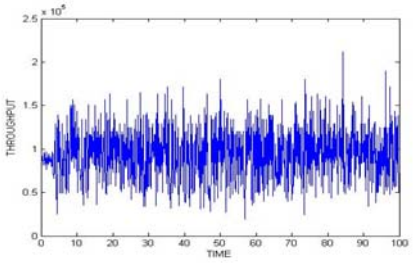

Fig.15: Enlace entre e2-rx3

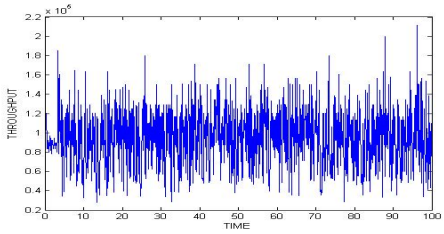

Fig.16: Enlace entre e2-rx5

- Escenario 2:

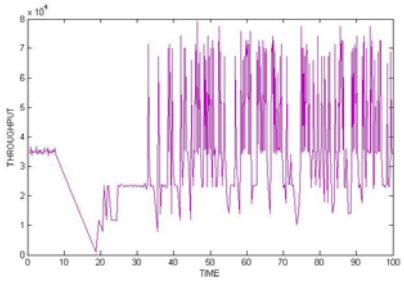

Fig.17: Enlace entre e2-rx1

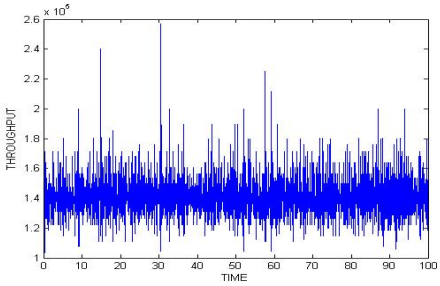

Fig.18: Enlace entre e2-rx3

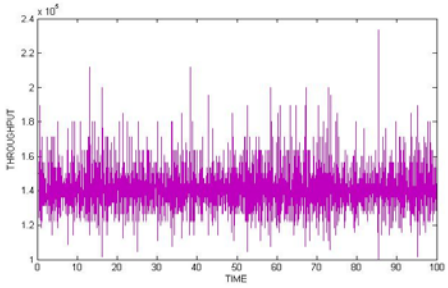

Fig.19: Enlace entre e2-rx5

- Escenario 3:

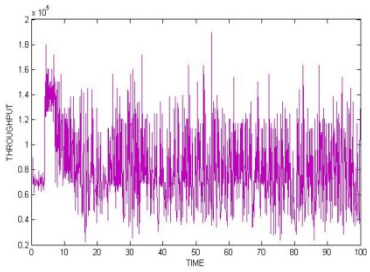

Fig.20: Enlace entre e2-rx1

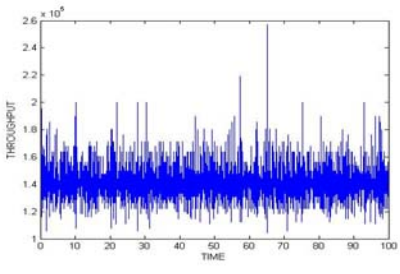

Fig.21: Enlace entre e2-rx3

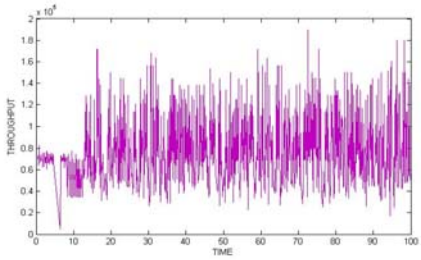

Fig.21: Enlace entre e2-rx5

- Escenario *4*:

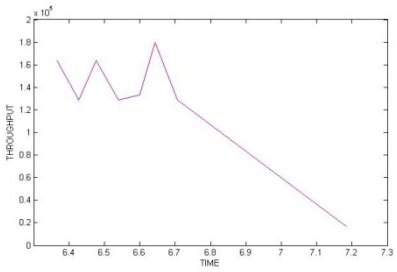

Fig.22: Enlace entre e2-rx1

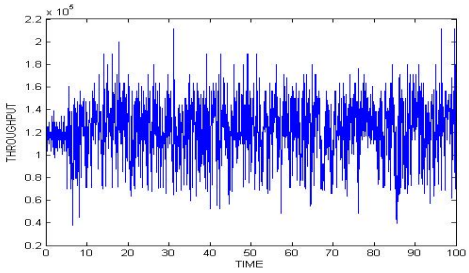

Fig.23: Enlace entre e2-rx3

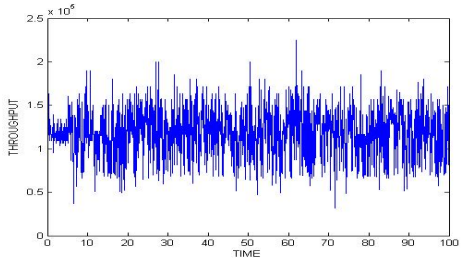

Fig.24: Enlace entre e2-rx5

Como se puede ver es diferente el caso en el cual se busca servir todos los flujos con la maxima prioridad del cas en el cual se da prioridad a un solo flujo. Si un solo tráfico es clasificado como Premium Service (escenario 2), las consideraciones son análogas a aquellas vistas en el páragrafo precedente. Si se busca servir todos los flujos con prioridad máxima (escenario 1) la diferenciación y se obtiene el efecto de una red best effort, es decir, la división de los recursos a partes iguales.

En el caso intermedio (escenarios 3 y 4) se nota la diferenciación de los servicios. Cuanto mayor es el tráfico best effort con respecto al tráfico total, mejores son las prestaciones del flujo Premium Service dado que éste último puede utilizar los recursos del tráfico con prioridad más baja. En conclusión el DiffServ es capaz de introducir prioridad pero la capacidad de proporcionar garantías depende mucho de cómo esté compuesto el tráfico en ingreso.

DIFERENTES PROBABILIDAD DE ERROR EN LAS TRANSMISIONES

En el escenario 144 Kb intentamos saber qué sucede cuando la probabilidad de paquete erróneo en el estado markoviano Bad es modificado. Para ello estudiamos las siguientes posibilidades:

- PB (*Probabilidad de Error en el estado Bad*): 0.2

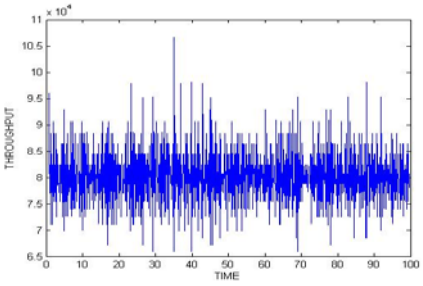

Fig.25: Enlace entre e2-rx1 (P. S.)

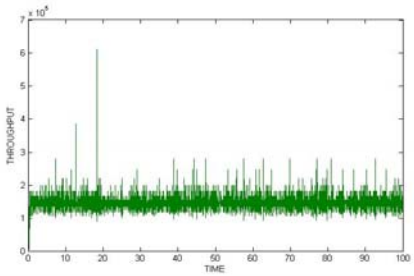

Fig. 26: Enlace entre e2-rx3 (AF2)

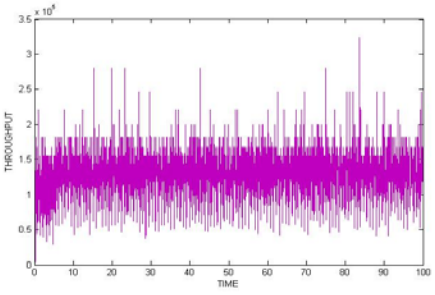

Fig.27: Enlace entre e2-rx5 (BE)

- PB (*Probabilidad de Error en el estado Bad*): 0.4

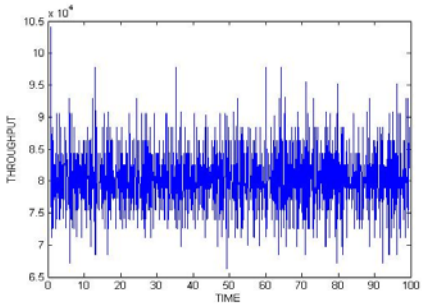

Fig.28: Enlace entre e2-rx1 (P. S.)

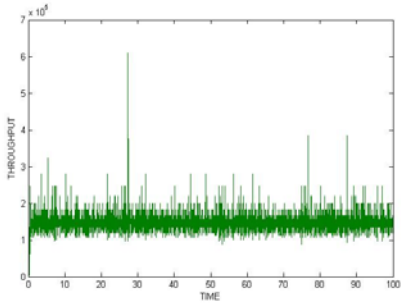

Fig. 29: Enlace entre e2-rx3 (AF2)

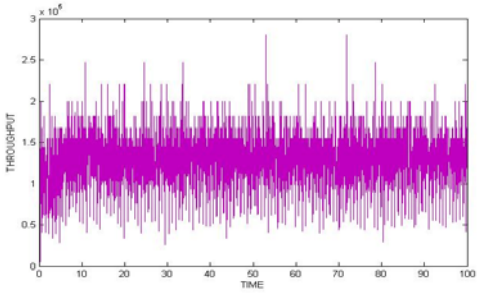

Fig.30: Enlace entre e2-rx5 (BE)

- PB (*Probabilidad de Error en el estado Bad*): 0.6

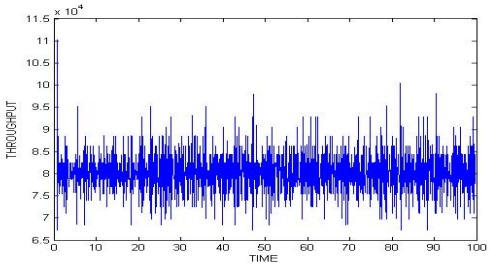

Fig.31: Enlace entre e2-rx1 (P.S.)

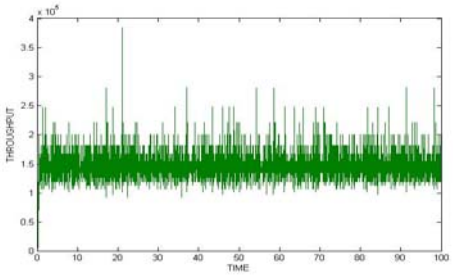

Fig.32: Enlace entre e2-rx3 (AF2)

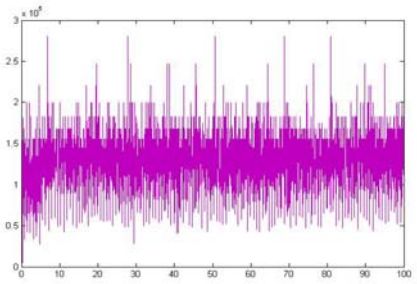

Fig.33: Enlace entre e2-rx5 (BE)

- PB (*Probabilidad de Error en el estado Bad*): 0.8

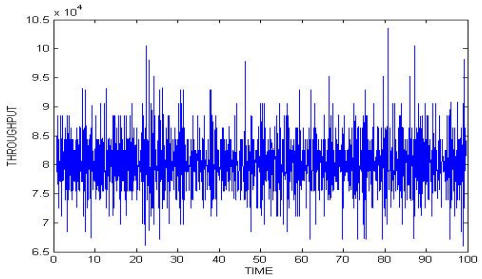

Fig.34: Enlace entre e2-rx1 (P. S.)

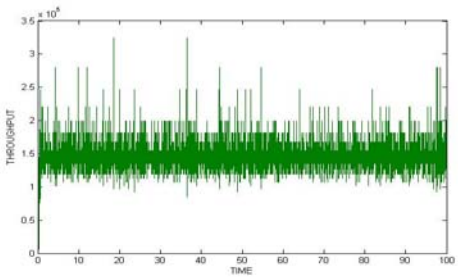

Fig.35: Enlace entre e2-rx3 (AF2)

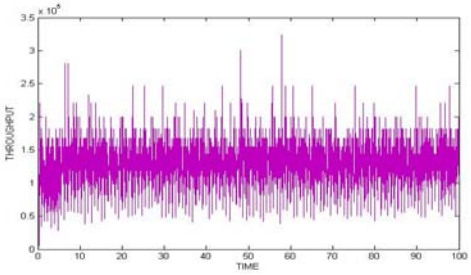

Fig.36: Enlace entre e2-rx5 (BE)

Un canal muy ruidoso necesita un mayor número de retransmisiones que inciden negativamente sobre el retardo total.

Conclusiones

La simulación desarrollada en éste proyecto ha demostrado que la arquitectura de redes Differentiated Services es capaz de realizar una eficaz diferenciación de los servicios con respecto a los efectos del retardo en la cola.

Comparando los resultados obtenidos por el escenario de servicio best effort con aquel relativo al escenario DiffServ se puede ver como el *throughput* por los enlaces que tienen mayor prioridad es mejor que el obtenido sin Diffserv porque toda la banda disponible es dividida a partes iguales entre los enlaces que hay sobre el router de salida.

Se puede ver también que, cuando todos los flujos tienen la misma prioridad el DiffServ se comporta igual que una red best effort, es decir, se dividen los recursos en partes iguales. De hecho, cuanto mayor es el peso de tráfico best effort respecto al tráfico total, mejores son las prestaciones del flujo Premium Service dado que éste último puede utilizar los recursos asignados al tráfico con prioridad más baja. En consecuencia, se puede concluir que el DiffServ es capaz de introducir prioridad, pero la capacidad de proporcionar garantía depende mucho de cómo esté compuesto el tráfico de ingreso.

También vemos modificando la probabilidad de error que en un canal más ruidoso se necesita un mayor número de retransmisiones que inciden negativamente sobre el retardo total.

Para terminar, se muestra que cuando más ruidoso es un canal necesita un mayor número de retransmisión que inciden negativamente sobre el retardo total.