Simulación de técnicas de encolado con prioridad*[1]

[Simulation of techniques for priority queues]

Néstor Jaime CASTAÑO-PÉREZ[2], Germán William LONDOÑO-JIMÉNEZ[3]

Recibo: 20.05.2011 - Ajuste: 03.06.2011 - Ajuste: 09.12.2011 - Aprobación: 17.12.2011

Resumen

Motivados por la necesidad de generar ambientes de desarrollo donde se prueben y analicen los comportamientos de equipos y algoritmos de conmutación de paquetes en un router, se presenta una arquitectura para la gestión de paquetes con priorización de tráfico, que consta de generador de trafico de voz y de video, gestor de colas y receptor del tráfico procesado. Como método para la gestión de tráfico se usa la generación de variables aleatorias de tiempo de llegada y tamaño de paquetes de voz y video, utilizando métodos de simulación de eventos discretos. Se plantea un política de encolado basada en etiquetado de paquetes y descarte por saturación de un sistema de espera M/M/1/K. Los programas se realizaron en Java y se utilizó Matlab para el análisis y graficación de los resultados. Como resultado principal, se demostró que los tiempos de espera promedio del los paquetes etiquetados para ser procesados con prioridad alta en la cola no sufrieron cambios significativos, cuando se aumentó el tráfico de baja prioridad.

Palabras Claves: *Programación de Hilos, Simulación, Encolado, Priorización de Tráfico.*

* Modelo para citación de este reporte de caso:
CASTAÑO PÉREZ, Néstor Jaime y LONDOÑO JIMÉNEZ, Germán William (2011). Simulación de técnicas de encolado con prioridad. En: Ventana Informática. No. 25 (jul. – dic., 2011). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 25-39. ISSN: 0123-9678.

1 Artículo proveniente de reflexiones académicas realizadas en el periodo 2009-2010, en el desarrollo del proyecto de aula para la integración de los Sistemas y Telecomunicaciones en la Universidad de Manizales.

2 Ingeniero Electrónico, Especialista en Telecomunicaciones, Magister en Automatización Industrial. Docente, Universidad de Manizales, Manizales (Colombia). Correo electrónico: ncastano@umanizales.edu.co.

3 Ingeniero Electricista, Especialista en Sistemas de Transmisión y Distribución, Especialista en Informática y Computación. Docente, Universidad de Manizales, Manizales (Colombia). Correo electrónico: glondono@umanizales.edu.co.

Abstract

Due to the need to create environments where behavioral development, equipment and packet switching algorithms in a router, are tested and analyzed, the paper presents an architecture for managing packet traffic prioritization, which is composed of a traffic generator for voice and video, a queue manager and a receiver processing traffic. For this architecture, the random variables were generated arrival time and size of packet voice and video, using methods of discreet event simulation. In addition, programs were conducted in Java and Matlab was used for analysis and graphing of results. Besides, considering a policy-based sizing and discard packet tagging saturation of a queuing system M/M/1/K. The paper shows that the average waiting times labeled packets to be processed with high priority, to increase the low-priority traffic, did not suffer significant changes.

Keywords: *Threads programming, Simulation, Queuing, Prioritization of traffic.*

Introducción

Las colas son elementos fundamentales en los sistemas de telecomunicaciones modernos y nacen de la necesidad de volver las redes más eficientes por medio de la *multiplexación* estadística de datos, concepto planteado por Gross, et al (2008, 5). Las colas se dan en los momentos en los que el flujo de entrada a un dispositivo de comunicaciones es mayor que el flujo de salida, por lo tanto existe una acumulación de datos o paquetes en la entrada del elemento de red. Si el flujo de entrada es superior o igual al flujo de salida, la acumulación de datos sería infinita y se rompería el equilibrio estadístico requerido en los modelos de Le Boudec y Thiran (2001, 53). Por este motivo los diseños de redes de telecomunicaciones están concebidos para que en la mayor parte del tiempo, el flujo de salida sea mayor al flujo de entrada. Aún, con estas consideraciones, existen instantes de tiempo en los cuales, por la misma naturaleza de la *multiplexación* estadística la red tiene un mayor volumen de paquetes a su entrada de los que puede procesar y es justo en estos momentos en los que se deben aplicar diferentes algoritmos para administrar el envío de los paquetes que se encuentran en la cola, temáticas tratadas por Olifer (2009, 201).

Uno de los primeros algoritmos para la administración de las colas es conocido como *PEPS* (Primero en Entrar - Primero en Salir), Tanembaum (2003, 34), en el cual los paquetes son atendidos en la cola en

el mismo orden en el cual fueron llegando a ésta. Para el transporte de datos planos este simple algoritmo puede desempeñarse adecuadamente y brindar una buena experiencia para el usuario. El problema con este tipo de colas, es que no es posible de forma alguna diferenciar un tráfico de otro, y con la necesidad de ofrecer servicios integrados sobre redes de datos, se hace necesario dar prioridad al trato de determinado flujo de datos que requiere en determinados casos minimizar variables como el tiempo de transmisión. Para solucionar este tipo de problemas se implementaron las colas con prioridad, expuestas por García y Widjaja (2003, 340).

Las colas con prioridad son en realidad una colección de colas *PEPS,* en las que a cada una se les asigna una prioridad específica y por medio del algoritmo de servicio, se atienden las colas *PEPS* por orden de prioridad de mayor a menor.

Para utilizar las colas de prioridad es necesario distinguir los paquetes cuando llegan al dispositivo para poder saber a qué cola se deben asignar. Es por este motivo que en la mayoría de los casos los paquetes son marcados antes de ser enviados al dispositivo de encolado, el cual utilizando una marcación predeterminada está en la capacidad de distinguir el tráfico y por ende tratarlo de una forma adecuada, según señala Donahue (2008, 24).

También se pueden dar casos en los cuales no es necesario marcar los paquetes sino que se utilizan los campos ya definidos en el encabezado de los paquetes para su diferenciación, como es el caso de la dirección de origen o dirección destino de los paquetes, o protocolo de transporte, o en programación de *routers* proyecto propuesto por la Universidad de *Stamford*, documentado por Reforgiato (2011, 22).

Ahora, en cuanto al funcionamiento de las colas de prioridad se pueden identificar dos tipos se sistemas: *preemptive* y *non-preemptive*. En un sistema *non-preemptive* una vez se da inicio a la atención de una unidad de datos el sistema no se puede detener hasta que termine su servicio, mientras que en un sistema *preemptive* es posible detener una unidad que está siendo atendida para la transmisión de otra unidad de mayor prioridad, considera Daigle (2010,248).

En este documento se simulará una cola de prioridad con dos niveles distintos de servicio en los cuales se utilizarán sistemas *non-preemptive*, es decir que una vez se inicie la transmisión de un paquete, esta se debe finalizar.

Idealmente, un sistema estricto de colas debería estar en la capacidad de parar la transmisión de un paquete ante la llegada de flujo de una mayor prioridad, pero al estar tratando con sistemas discretos no es

posible detener la transmisión de un paquete una vez esta se inicia. Además de ser algo que no es posible de implementar, se podría pensar en cancelar la transmisión del paquete como un error, pero esto obligaría a la retransmisión del paquete lo que a la final haría que aumentara el retraso en la red, por la constante retransmisión de paquetes que tuvieron que ser interrumpidos a causa de flujos de mayor prioridad.

1. Marco teórico

1.1 Caracterización de las variables aleatorias

Antes de emprender cualquier proceso de simulación de eventos discretos se hace necesario caracterizar las variables aleatorias que intervienen en el sistema. Esta caracterización se realiza mediante los histogramas de cada variable. En la literatura se encuentran diferentes métodos para la generación de estas variables aleatorias, expuestos con claridad por Kay (2005,18). En este caso se encontró que el tiempo entre llegadas del tráfico de *VoIP* es exponencial (Figura 1) y el número de paquetes que llega por segundo es de tipo *Poisson* (Figura 2). La caracterización de tráfico permite generar vectores de tráfico en *Matlab* de acuerdo con las necesidades experimentales.

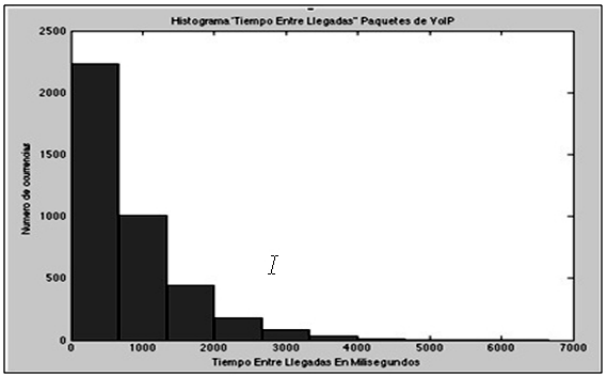

Figura 1. Histograma Tiempo entre llegada

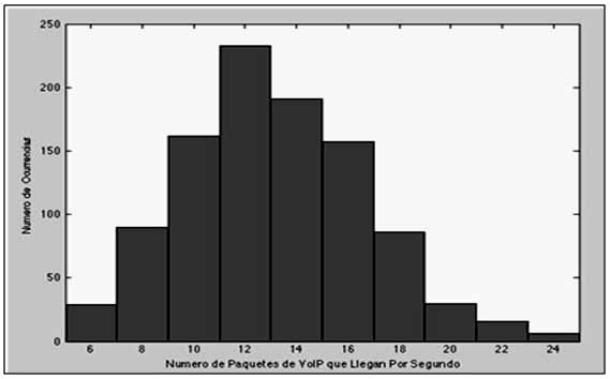

Figura 2. Número de Paquetes que llegan por segundo

1.2 Tráfico tipo Poisson

En los sistemas de teletráfico, las llegadas de paquetes pueden ser modeladas como sistemas tipo *Poisson,* si se considera que son producto de una población amplia de usuarios independientes, como lo afirma Coosbu (2002, 43).

Si se desea calcular el retraso y el *throughput* del *buffer* de un enlace de red que opera con una capacidad de N Mbps. Ésta situación se ilustra en la figura 3, donde los paquetes se representan como cajas rectangulares.

Un modelo muy popular para representar el número de paquetes que llegan en un intervalo de tiempo es el proceso de *Poisson*. En un proceso de *Poisson* con tasa λ el número de llegadas de eventos en un intervalo de tiempo [t, t+τ], denotado por N[t+τ] - N[t] está dado por:

$$P[N(t+\tau)-N(t)=K]=\frac{(\lambda\tau)^k}{K!}e^{-\lambda\tau} \quad (1)$$

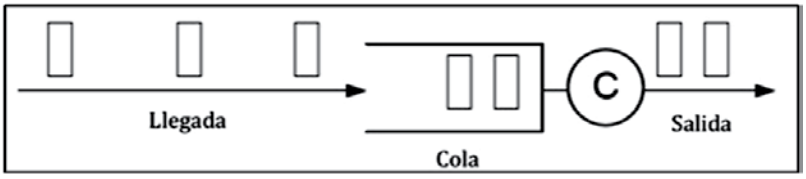

Figura 3. Llegada y Salida de paquetes en una cola

Un proceso de *Poisson* tiene algunas propiedades interesantes Youngjae (2008, 1) y útiles como:

El tiempo de llegada entre dos eventos (en este caso, paquetes) sigue una distribución tipo exponencial:

$$P[Tiempo\ Entre\ Eventos \le X] = 1 - e^{\lambda X} \quad (2)$$

Si se suman las llegadas de dos procesos tipo Poisson con tasa λ y μ, es equivalente a tener un solo proceso tipo Poisson con tasa de λ + μ.

1.3 Tráfico de video

El video digital consiste de una secuencia de tramas de video que son mostradas a una tasa de típicamente 30 cuadros por segundo. Asumiendo que un cuadro tiene un tamaño de 720 x 480 pixeles (un formato estándar utilizado para video digital) y una profundidad de 24 bits por pixel, esto resulta en una tasa de datos de 240 Mbps por línea de video. Utilizando métodos de compresión modernos, como *MPEG y*

H.264, la tasa de datos puede ser drásticamente reducida a unos pocos Mbps, como asegura Richardson (2010, 9).

Un codificador *MPEG* (y casi la gran mayoría de algoritmos de compresión) generan, según Harte (2006, 19), tres tipos de tramas: Intra-Codificadas (I), Inter-Codificadas o Predictivas (P) y Bi-direccionales (B). Las tramas I son codificadas como imágenes congeladas como *JPEG*; las tramas P codifican las diferencias entre la trama P o I más reciente; y las tramas B son interpolaciones de las tramas anteriores y posteriores P o I. Un codificador *MPEG* genera estas tramas en un patrón repetitivo, llamado Grupo de Imagen *(Group Of Picture -GOP-)*. Con 30 cuadros por segundo, el tiempo que pasa entre dos tramas es aproximadamente 33 ms. Un patrón de *GOP* típico es *IBBPBBPBB*, donde las dependencias entre las tramas están indicadas por flechas, como se muestra en la figura 4.

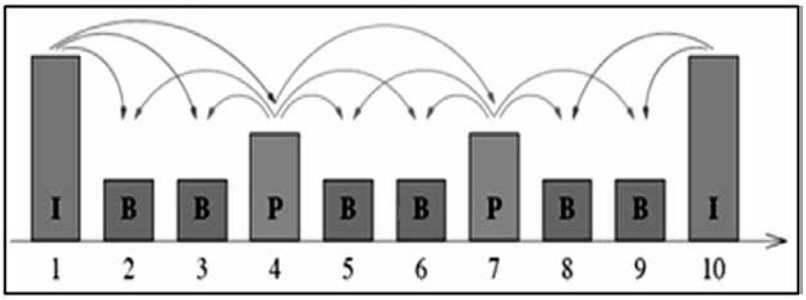

Figura 4. GOP de una trama de video (Sniderman, 2011)

Los diferentes tipos de trama tienen tamaños distintos, siendo las tramas I las más grandes y las B las más pequeñas. Se debe tener en cuenta que las tramas B son construidas a partir de dos tramas de referencia tipo I o P. Por lo tanto, para codificar o decodificar una trama tipo B es necesario que el transmisor o el receptor tengan disponible las tramas P o I correspondientes. Para cumplir con esta característica, las tramas son transmitidas en una secuencia diferente a la cual son presentadas. En la figura 4, la secuencia de transmisión y de presentación es como se describe en la tabla 1.

Tabla 1. Orden de secuencia de presentación y transmisión de una trama de video

Secuencia de Presentación	1	2	3	4	5	6	7	8	9	10
Secuencia de Transmisión	1	4	2	3	7	5	6	10	8	9

Debido a las diferencias en tamaño de las tramas I, P y B, la tasa de datos de un video *MPEG* cambia de trama a trama. Por esta razón, el

video comprimido como *MPEG*, es referido como video de tasa variable *(Variable BitRate-VBR-)* o video *VBR.*

1.4 Consideraciones de tráfico

Es necesario tener en cuenta que el tráfico tipo *Poisson* es posible modelarlo a partir de una distribución específica, desde la cual es posible desprender características y variables que permiten inferir el comportamiento del tráfico en un futuro, y por ende planear de acuerdo a estas perspectivas.

El video, al ser un tráfico tipo VBR, no obedece una distribución específica y para su análisis es necesario adoptar otro tipo de estrategias como es el caso de las metodologías presentadas por Dunrkin (2003, 22).

Aunque en este documento no se profundiza en temas como curvas de llegada o curvas de servicio, es importante hacer la aclaración de que los dos tipos de tráficos con los que se trata se comportan de forma distinta e independiente aunque ambos son aleatorios.

2. Implementación

En el presente artículo se presenta, a partir de Foster (2004, 23), la implementación de una cola de prioridades utilizando el lenguaje de programación *JAVA* y sus librerías para *Sockets* para la implementación de la cola, el envío de paquetes y la recepción de los mismos y *Matlab* para realizar el análisis de los resultados obtenidos por medio de la cola. Como parte del análisis, se incluirá el tiempo medio de espera de un paquete y el tamaño promedio de cada una de las colas.

Como se mencionó anteriormente, para implementar la cola de prioridad es necesario marcar los paquetes que se están enviando a la cola para que una vez lleguen a este dispositivo sea posible su ubicación en el *buffer* respectivo dependiendo de su prioridad. En el presente caso, se pretende diferenciar tráfico de video, de tráfico tipo *Poisson*. Por eso se utilizan dos fuentes de datos, cada una de las cuales marca los datos que envía colocando el byte *0* (cero) del paquete enviado en *1* en el caso del generador de datos *Poisson* y en *2* en el caso del generador de tráfico de video.

La implementación propuesta es la de una cola de prioridades, que permita la diferenciación en cuanto al tratamiento en cola del tráfico de video y tráfico tipo *Poisson*, como se muestra en la Figura 5. La solución propuesta son dos programas que generan el tráfico: - El programa representado por el rectángulo *Tráfico de voz* que se encarga de generar el tráfico tipo *Poisson* a partir de un archivo de texto que

contiene la información del tráfico, además marca los paquetes antes de ser enviados a la cola, y - el programa representado por el rectángulo *Tráfico de video* que genera tráfico de video igualmente a partir de un archivo de texto que contiene la información del tráfico.

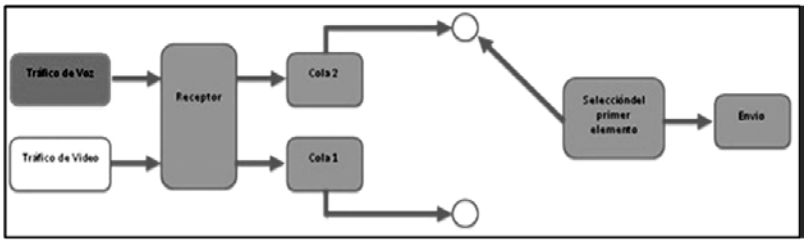

Figura 5. Diseño general de la cola de prioridad

El tercer programa implementado es la cola, la cual está compuesta por dos hilos, para que la cola pueda recibir paquetes y enviarlos de forma paralela. El primero se encarga de recibir los paquetes y clasificarlos, ubicándolos en el *buffer* correspondiente por medio de la marcación realizada en el proceso de envío, mientras el segundo se encarga de implementar la cola de prioridad, tomando los elementos del *buffer* de mayor prioridad en primera instancia, y los del *buffer* de menor prioridad en segunda medida siempre y cuando el *buffer* de alta prioridad se encuentre vacío. Luego de tomar un elemento de alguno de los *buffers*, este es enviado.

El diseño del hilo que se encarga de recibir los paquetes en la cola de prioridad se muestra en la Figura 6.

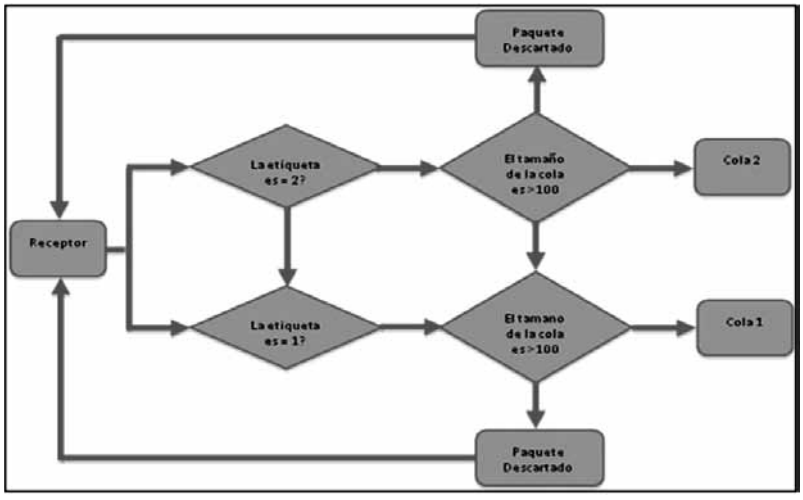

Figura 6. Diseño General del hilo para recibir los paquetes.

En este caso, los paquetes se reciben y se consulta el valor del primer byte del paquete. Si éste es 1, se mira el tamaño del *buffer* de la cola de baja prioridad, y si la cola tiene un tamaño superior a 100 Kbytes, entonces el paquete se descarta, de lo contrario se agrega a la cola de baja prioridad. Si el primer *byte* del paquete es 2, se toma el tamaño de la cola de alta prioridad y si éste excede igualmente los 100Kbytes se descarta el paquete, si no, se ubica en la cola de alta prioridad.

Ahora, el funcionamiento de la cola de prioridad se muestra en la Figura 7. El hilo para enviar está en un ciclo, el cual siempre verifica el estado de la cola de prioridad alta. Si este *buffer* tiene algún elemento, se remueve el primer ítem del arreglo y se envía. Solo en el caso en que el *buffer* de la cola de prioridad alta esté vacío se monitorea el *buffer* de la cola de baja prioridad y si tiene elementos, se remueve el primer ítem del arreglo y es enviado. Una vez un paquete sea enviado, el hilo vuelve a su estado inicial, que es monitorear el *buffer* de la cola de prioridad alta.

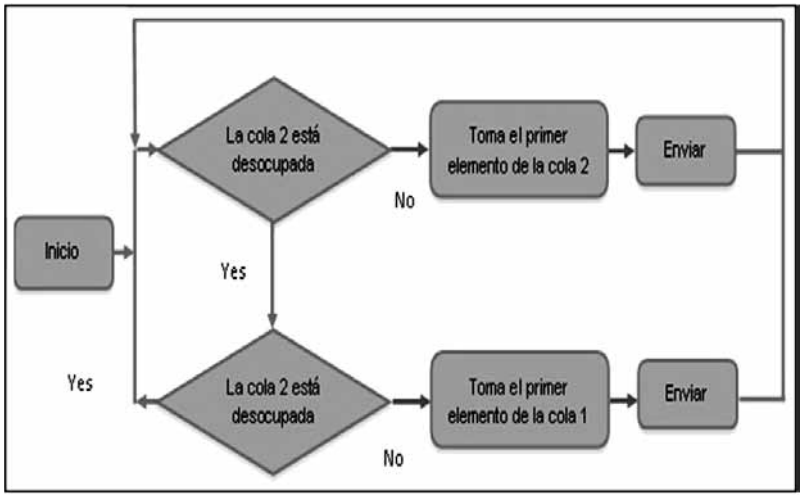

Figura 7. Diseño General del hilo para implementar la cola de prioridad.

La cola de prioridad no atenderá la cola de prioridad baja a no ser que la cola de prioridad alta este vacía. Aunque este es el comportamiento programado, en ocasiones puede no ser el deseado, ya que si el tráfico de la cola de alta prioridad es *alto* se pueden dar casos en los que el retraso de la cola de baja prioridad es grande, lo que se convierte en una denegación de servicio para el tráfico de baja prioridad, como lo plantean García y Widjaja (2003, 221).

Es por este motivo que las colas de prioridad, se utilizan para procesar tráfico que requiere transmisión en tiempo real, como es el caso de

voz en las redes de datos. Este tipo de colas son implementadas en equipos de comunicaciones como enrutadores *CISCO* que proporcionan opciones para la calidad de servicio bajo el nombre de *LLQ* (*Low Latency Queueing*).

En el caso en el que se desee diferenciar tráfico sin necesidad de recurrir a las colas de prioridad, se tienen otro tipo de algoritmos como el *WFQ, GPS o EDF* los cuales no serán considerados en el presente documento, pero se mencionan como referencia para el lector. Con el objetivo de observar el comportamiento de la cola en cuanto la diferenciación en el trato del tráfico se decide crear una variable en el generador de tráfico *Poisson* que permita la variación de la intensidad o tasa promedio de transmisión de tráfico entre 0 y 10Mbps. Esta variable se le asigna el nombre de N, y varía entre 0 y 10; por lo tanto, para un valor de N=5 se tiene un tráfico tipo *Poisson* con una tasa de transmisión promedio de 5 Mbps.

Lo que se quiere observar es cómo se comporta la cola de prioridad para diferentes valores de N y cómo afecta el aumento en el tráfico de baja prioridad el comportamiento del *buffer* de alta prioridad. También se puede observar que sucede con el *buffer* de baja prioridad a medida que se aumenta el tráfico tipo *Poisson*.

El programa diseñado para la simulación de la cola, fue pensado para emular el comportamiento de tráfico a diferentes niveles de carga, motivo por el cual el programa encargado de generar el tráfico de video posee una variable *N* que se introduce en el momento de ejecución, la cual permite generar datos con velocidad promedio desde 0 hasta 10 Mbps. Es necesario aclarar que la tasa de transmisión de datos correspondientes al tráfico de video es la misma, sin importar el valor de *N* que se esté utilizando para el transmisor de datos tipo *Poisson*.

3. Resultados

3.1 Tiempo de espera

Es interesante conocer el tiempo de espera de los paquetes en la cola debido a que éste es un factor que está estrechamente relacionado con el retraso en el que los paquetes incurren al atravesar la red. Éste retraso de los paquetes afecta de una forma directa la interactividad de la experiencia de tiempo real (audio o video) por lo cual se debe tratar de minimizar su impacto sobre el tráfico de datos.

Para medir el tiempo de espera de los paquetes en la cola de prioridad se implementó un arreglo de tipo *flotante*, el cual guarda el tiempo en el cual un paquete ha llegado al *buffer* (de alta o de baja prioridad según sea el caso). Una vez se vaya a enviar un paquete específico se toma

el tiempo en el que ocurre la transmisión y el tiempo asociado a ese paquete contenido en el arreglo de valores *flotantes*. La diferencia entre los dos tiempos da como resultado el tiempo durante el cual el paquete estuvo esperando a ser servido. Una vez el paquete es enviado, el tiempo asociado a éste es borrado del arreglo de tiempos.

Adicionalmente, en cuanto al tiempo de espera se presentan dos métricas: *Tiempo de Espera Máximo* y *Tiempo de Espera Promedio* como se puede apreciar en la Figura 8, donde se muestran como puntos en el plano los valores arrojados por la simulación de la cola, acompañados de una línea de proyección producto de una interpolación cúbica de los valores de Tiempo de Espera Máximo y Tiempo de espera promedio obtenidos para N=1, N=5 y N=9 realizada en *Matlab*.

Respecto al desempeño del *Tiempo de Espera Promedio* de los paquetes en el *buffer* de alta prioridad se observa que es prácticamente estable y cercano a los 0.8 ms. Este tipo de comportamiento era el esperado, pues sin importar la intensidad del tráfico tipo *Poisson*, los paquetes presentes en el *buffer* de alta prioridad van a ser atendidos, y para el caso de la simulación no se modificó la cantidad de tráfico de video que llega a la cola. El tiempo de diferencia entre el tiempo promedio de espera de los paquetes de la cola de alta prioridad presenta variabilidad por factores como la no dedicación de un procesador a la ejecución de ésta tarea. Las funciones de la CPU se redujeron al mínimo para evitar la interferencia con el procesamiento de la información concerniente a la Cola de Prioridades, pero de todas formas es imposible garantizar la dedicación total del procesador a realizar estos cálculos.

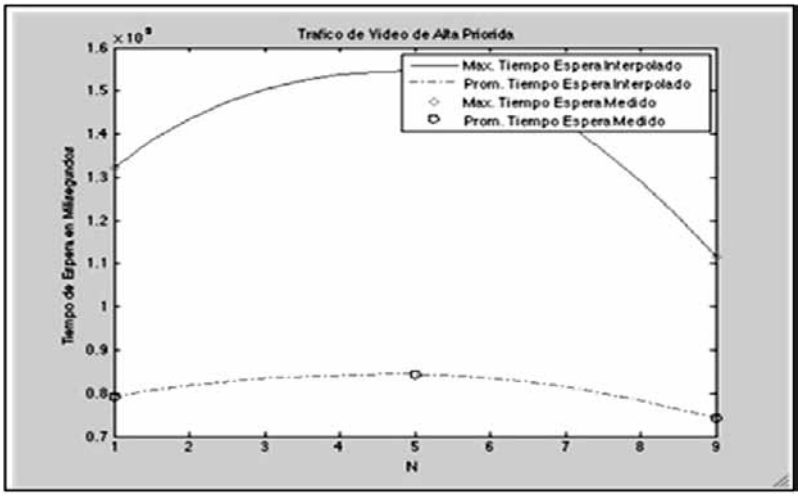

Figura 8. Resultados de la simulación de la cola de prioridad

Debido a que el canal por el cual se transmite la información no ha aumentado su tamaño, y considerando que se ha incrementado la tasa de transmisión de datos de baja prioridad, es de esperarse que para garantizar la transmisión de la información de alta prioridad, deban existir peritadas o descartes de paquetes pertenecientes al grupo de prioridad menor, tal como se puede apreciar en la Figura 9. También es interesante notar que como se ha demostrado en la teoría de redes, el comportamiento de un sistema de comunicaciones cuando alcanza su punto de saturación deja de ser lineal. Es decir que para el caso anterior, el punto de quiebre de la linealidad a la no-linealidad se da aproximadamente para valores de N=5.

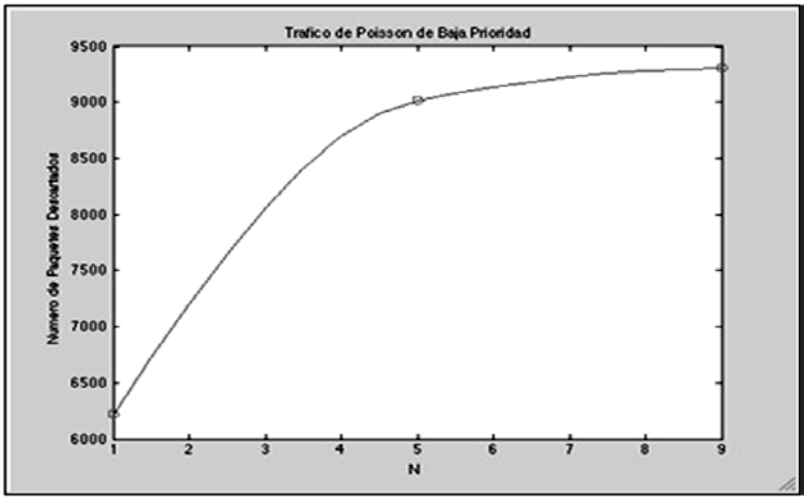

Figura 9. Paquetes Descartados en Cola de Baja Prioridad.

Luego de haber observado el comportamiento de la cola de alta prioridad y de la cola de baja prioridad, también pueden verse los resultados de la simulación, los cuales se muestran en la Tablas 2 y 3.

Tabla 2. Resultados para cola de baja prioridad

Parámetros	N=1	N=5	N=9
Tiempo Máximo de Espera	31302.76 ms	36553.04 ms	25672.84 ms
Tiempo Promedio de Espera	22343.7 ms	19910.28 ms	17460.3 ms
Tamaño Máximo de la Cola	99993 Bytes	100000 Bytes	99990 Bytes
Tamaño Promedio de la Cola	85766 Bytes	88201 Bytes	89570 Bytes
Numero de Paquetes Descartados	4170	7651	7965
Porcentaje de Paquetes descarados	41.7%	76.51%	79.65%

Tabla 3. Resultados para cola de alta prioridad

Parámetros	N=1	N=5	N=9
Tiempo Máximo de Espera	662.15 ms	828.2 ms	652.75 ms
Tiempo Promedio de Espera	100.15 ms	131.13 ms	154.99 ms
Tamaño Máximo de la Cola	14875 Bytes	16651 Bytes	14432 Bytes
Tamaño Promedio de la Cola	3118 Bytes	3328 Bytes	3056 Bytes
Numero de Paquetes Descartados	0	0	0
Porcentaje de Paquetes descarados	0%	0%	0%

En el caso de la cola de baja prioridad, se observa como el tamaño promedio de la cola se incrementa a medida que aumenta la carga o tasa de transmisión de la información, así mismo como el número de paquetes que son descartados por el sistema. Para esto se recuerda que los programas realizados en *JAVA* fueron configurados de forma tal que las colas no aceptan paquetes si éstas ya tienen su *buffer* 100.000Bytes.

Ahora, teniendo en cuenta los datos arrojados por el simulador para el flujo de datos de alta prioridad es posible apreciar que no existen paquetes descartados, debido a que todo paquete que entra en la cola siempre es servido y además se muestra que en promedio la tasa de llegada de los paquetes a su cola es inferior a la tasa de salida de estos. Por otro lado, también es posible apreciar como el tamaño, tanto promedio como máximo de la cola de alta prioridad posee valores muy interiores a los arrojados por el simulador para el caso de la cola de baja prioridad.

4. Conclusiones

- Luego de realizar las simulaciones en *JAVA* y analizar los resultados en *Matlab* fue posible demostrar de una forma clara que las colas de prioridad son altamente confiables cuando se trata de aislar un tipo de tráfico de otro. Esto se demuestra en el caso en el que el tráfico de alta prioridad se comportó de una forma estable ante la presencia de diferentes cargas de tráfico de baja prioridad. Simulaciones con generación de tráfico como las mostradas, se constituyen en una herramienta fundamental para futuros experimentos en netFPGA, en donde se necesite probar algoritmos de gestión o tiempos de retardo en LAN o WAN.
- Por ser colas que permiten un gran aislamiento del tráfico son altamente importantes para el transporte de información en tiempo real, tales como audio y video en vivo.

- En cuanto a los resultados arrojados por el simulador, para su análisis también es necesario considerar que se trata de tráfico *VBR – Variable BitRate* – lo cual, al interactuar con la diferencia de tasas de transmisión y por ende diferente funcionamiento de la cola de baja prioridad presenta resultados con leves diferencias en números pero que son concordantes con el comportamiento general esperado de la cola de alta prioridad.
- La variabilidad de los datos arrojados por la simulación es debida a la no dedicación exclusiva del procesador a realizar las tareas asignadas por la Cola de prioridad, y un próximo paso hacia la comprobación de resultados, sería la implementación del algoritmo en un dispositivo de dedicación exclusiva como un *FPGA*.
- Es necesario tener en cuenta que la sincronización en las colas juega un papel preponderante, debido a que se pueden dar casos en los que un elemento esté ingresando a la cola mientras otro esté siendo retirado. Si no se pone cuidado a este tipo de eventos se incurre en errores de cálculo y pérdidas de paquetes. Para evitar esto, se utilizó la clase "Vector" en *java*, la cual por herencia ya se encuentra sincronizada.

Bibliografía

COOSBU, Raúl (2002) Simulación: Un enfoque práctico. México: Limusa, 157 p. ISBN: 9681815068.

DAIGLE, John N. (2010). Queueing Theory with Applications to Packet Telecommunication. 2 ed. Mississippi (USA): Springer, 315 p. ISBN: 0-387-22857-8

DONAHUE, Gary A. (2008) Network Warrior: Everything you need to know that wasn't on the CCNA exam. Sebastopol (USA): O'Reilly, 545 p. ISBN: 978-0-596-10151-0.

DUNRKIN, James F. (2003) Voice-enabling The Data Network: H.323, MGCP, SIP, QOS, SLAS, AND SECURITY. Indianapolis (USA): Cisco Press. 227 p. ISBN: 1-58705-014-5.

FOSTER, James C. (2004) Sockets, Shellcode, Porting & Coding: Reverse Engineering Exploits and Tool Coding for Security Profesionals. Rockland (USA): Syngress Publishing Inc, 653 p. ISBN: 1-597490-05-9.

GARCÍA, Alberto and WIDJAJA, Indra. (2003) Communication Networks: Fundamental Concepts and Key Architectures. 2 ed. New York (USA): McGraw-Hill, 871 p. ISBN: 0-07-246352-X.

GROSS, Donal; SHORTLE, John; THOMPSON, Jim and HARRIS, Carl (2008). Fundamentals of Queueing Theory. New York (USA): John Wiley & Sons, 600 p. ISBN: 978-0470547830.

KAY, Steven (2005) Intuitive Probability and Random Processes using MATLAB. Kingston (USA): Springer, 851 p. ISBN: 978-0387241579

LAWRENCE, Harte (2006). Introduction to MPEG; MPEG-1, MPEG-2 and MPEG-4. Fuquay Varina (NC, USA): Althos, 92 p. ISBN: 978-1932813531

LE BOUDEC, Jean-Yves and THIRAN, Patrick (2001) Network Calculus: A Theory of Deterministic Queuing Systems for the Internet. New York (USA): Springer Verlag. 274 p. ISBN: 354042184X.

NEMBAUM, Andrew (2003) Redes de Computadoras. 5 ed. México (México). Prentice- Hall, 34 p. ISBN: 970-26-0162-2

OLIFER, Natalia (2009). Redes de Computadoras. México (México): McGraw-Hill, 751 p. ISBN: 9789701072493

REFORGIATO, Diego (2011). A Traffic Monitor on NetFPGA platform: Comparisons against a Software Implementation on the Click Modular Router. Rome (Italy): Lambert, 84 p. ISBN: 978-384338140

RICHARDSON, Iain (2010). The H.264 Advanced Video Compression Standard. London (United Kingdom): Wiley, 256 p. ISBN: 978-0-470-51692-8

SNIDERMAN, Andrew (2011). VGA Conferencing in Lync Server 2010 [on line]. Redmond (WA, USA): Markmonitor. <http://blogs.technet.com/b/nexthop/archive/2011/06/22/vga-conferencing-in-lync-server-2010.aspx> [consult: 30/06/2011].

TANEMBAUM, Andrew (2003). Redes de Computadoras. 5 ed. México (México): Prentice- Hall, 912 p. ISBN: 970-26-0162-2

TSUHAN, Chen (2005). Multimedia Systems Standards and Networks. Pittsburgh (USA): Marcel Dekker, 631 p. ISBN: 0-8247-9303-X

YOUNGJAE, Lee (2008). Characterization of Large-Scale SMTP Traffic: the Coexistence of the Poisson Process and Self-Similarity. In: IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems. MASCOTS. p. 1-10. ISSN: 1526-7539.