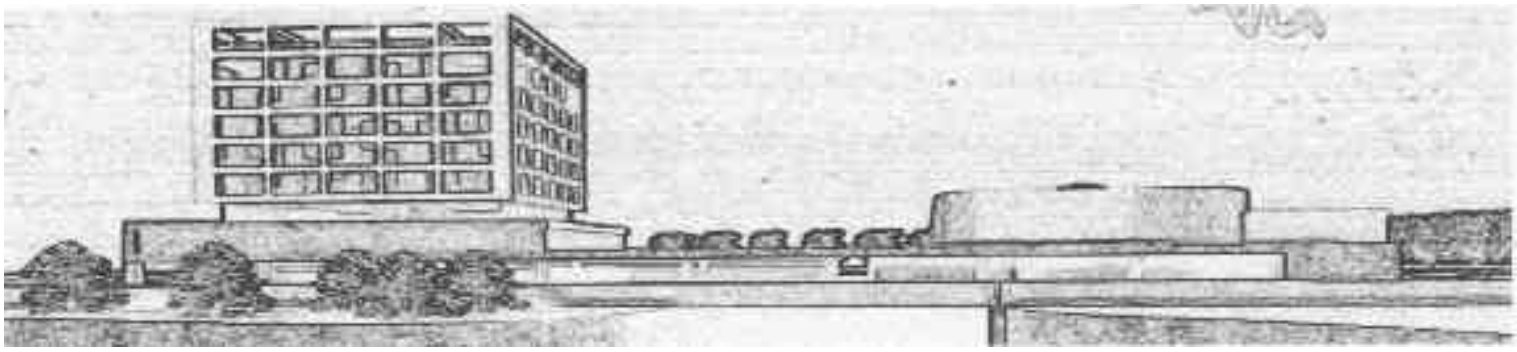
VOLUME VII, NUMBER 3



REPORTS ON SYSTEMS AND COMMUNICATIONS

Estudio comparativo de políticas de planificación de colas con aplicaciones al tráfico de tiempo real

JUAN MARTÍNEZ-ROMO
LUIS LÓPEZ-FERNÁNDEZ
ANTONIO FERNÁNDEZ
JUAN CÉSPEDES

Móstoles (Madrid), Abril 2007
Depósito Legal: M-50653-2004
ISSN: 1698-7489

Estudio comparativo de políticas de planificación de colas con aplicaciones al tráfico de tiempo real

Juan Martínez-Romo, Luis López-Fernández, Antonio Fernández, Juan Céspedes
{juaner, llopez, anto, cespedes}@gsyc.escet.urjc.es
Laboratorio de Algoritmia Distribuida y Redes
Universidad Rey Juan Carlos
Tulipán S/N
28933 Móstoles (Madrid)

Abstract *Well-known theoretical results have proven that buffer scheduling disciplines based on the traditional FIFO policy may produce unstable networks even at very low network loads. In addition, similar analysis have demonstrated the existence of novel disciplines that are stable independently of the topology, as long as the network is not fully loaded.*

In this paper, we present a comparative study of different network parameters obtained by applying FIFO and other novel scheduling disciplines. This comparison have been carried out in two different environments, one based on the J-Sim simulator, and other based on a real execution by means of the modification of a 2.6.7 linux kernel. Based on the obtained results, we conclude that some of these novel disciplines seem to present better features than FIFO, and could have an important impact on the quality of service of multimedia and other types of real-time-constrained traffic.

1. Introducción

En una red, es común que un gran número de paquetes intenten cruzar el mismo enlace en el mismo instante de tiempo. El criterio usado para elegir el paquete que pasará en primer lugar dependerá de la política o estrategia que aplique la cola asociada a ese enlace. Las políticas usadas por los routers, influyen drásticamente en como se comporta la red. Con el objetivo de estudiar esa influencia en escenarios de peor caso (WCS), se han creado un conjunto de modelos teóricos.

El primer modelo, propuesto por Cruz [1, 2], asumía que todos los paquetes en la red estaban agrupados en sesiones con rutas y tiempos de llegada asociados, y que la inyección de paquetes estaba controlada por un adversario. En este modelo se ha mostrado que existen políticas que garantizan la estabilidad [3, 4] cuando el enlace no esta totalmente cargado mientras que FIFO (First-In-First-Out) puede llegar a ser inestable [5].

En un segundo modelo, denominado Adversarial Queueing Theory (AQT) [6, 7], el adversario controla tanto la inyección como las rutas de los paquetes. Este modelo no necesita de la existencia de sesiones, aunque asume que el sistema evoluciona en pasos discretos, lo que implica que todos los paquetes deben de tener la misma longitud y todos los enlaces el mismo ancho de banda. Se ha demostrado que sobre AQT existen políticas que son estables para cualquier red no sobrecargada, y políticas como *FIFO* (entre otras) que pueden ser inestables para cualquier carga constante de la red [8].

Recientemente ha surgido Continuous AQT (CAQT) [9, 10], un tercer modelo que intenta combinar los dos anteriores evitando el comportamiento síncrono de AQT. En este modelo, los paquetes no necesitan estar agrupados en sesiones ni tener el mismo tamaño. Los enlaces pueden tener diferentes anchos de banda y retardos de propagación. En [10] se ha demostrado que algunas de las políticas estables en AQT también lo son en $CAQT$.

Estos resultados teóricos ayudan a caracterizar el problema de elegir una política de planificación. En el caso de poder optar por alguna política, sería preferible una política estable a una política potencialmente inestable. Sin embargo, en aplicaciones de tiempo real y críticas en cuestión de latencia, no es suficiente con tener asegurado un retardo máximo, sino que es necesario tener retardos pequeños y un jitter lo más bajo posible. Sobre la base de estos últimos requisitos han surgido trabajos como [11] en el que se realizan simulaciones empleando las políticas introducidas en [6], y en el que se muestran notables mejoras en relación a FIFO, pero nunca utilizando un sistema real.

El *throughput*, la *latencia* y el *jitter* son tres de los parámetros más importantes de la Calidad de Servicio (QoS), cuando hablamos de una transmisión de tiempo real. El *throughput* determina la cantidad de información que puede circular por un medio físico de comunicación de datos por unidad de tiempo. Es bien conocido que existen diferentes modos de medir la *latencia* de red, pero lo más habitual es calcularla como el tiempo necesario para que un paquete de información viaje desde la fuente hasta su destino. El *jitter* por su parte, mide la dispersión en el retardo de los paquetes dentro de una misma sesión. Estos parámetros han sido

ampliamente estudiados en la literatura [12] y su impacto en el tráfico de tiempo real ha sido claramente demostrado en tecnologías ATM.

La *latencia* tiene una gran repercusión en aplicaciones que implican la interacción del usuario, como por ejemplo VoIP. El *jitter* por su parte posee una gran relevancia en la difusión de contenidos en tiempo real, tales como streaming de vídeo o audio. Por estas razones, se han concentrado las medidas en el valor de estos tres parámetros.

1.1. Contribución

En este artículo presentamos la implementación, en un escenario real y en uno simulado, de algunas de las políticas propuestas para el modelo $CAQT$ [7, 10], extrapolando este modelo a un escenario con colas acotadas, pérdidas de paquetes y sobrecarga de los enlaces. En la sección 4 podremos apreciar los resultados obtenidos en medidas como el throughput, la latencia, y el jitter, con el objetivo de evaluar el comportamiento de las distintas políticas. En base a estos resultados podremos convenir que algunas de las políticas desarrolladas presentan mejoras frente a *FIFO*, pudiendo tener un gran impacto en la calidad de servicio de las transmisiones multimedia y otros tipos de tráfico de tiempo real.

Como ya hemos mencionado anteriormente, en este artículo vamos a valorar algunas de las políticas estudiadas analíticamente en el modelo $CAQT$ [10]. Estas políticas, que serán comparadas con FIFO, son las siguientes: Longest-In-System (LIS) que otorga mayor prioridad a los paquetes más viejos en el sistema, Shortest-In-System (SIS) por el contrario proporciona mayor prioridad a los paquetes más jóvenes, y en cuanto a las políticas que se basan en el tiempo de vida (TTL), Farthest-From-Source (FFS) considera más prioritarios los paquetes que más enlaces han atravesado, y Nearest-From-Source (NFS) que permite salir antes a los paquetes que menos saltos han realizado. Además de estas cuatro nuevas políticas, hemos desarrollado un nuevo diseño, denominado *LISB*, y que es una variante de *LIS* en la que solamente se contabiliza el tiempo que los paquetes han permanecido esperando en las colas de los routers anteriores.

En la sección de resultados se puede observar la gran semejanza entre las medidas obtenidas en el escenario real y en el simulado, dotando de una gran coherencia a este trabajo. En cuanto a las medidas llevadas a cabo, el *throughput* revela que políticas como *LIS*, *SIS* o *FFS* tienen un comportamiento más ecuánime en relación a los paquetes con distinta antigüedad en el sistema. Observando la *latencia media*, *LIS* y *SIS* actúan de nuevo de una manera más equilibrada, empeorando *FFS* sus valores para los paquetes más jóvenes. Resumiendo, *FFS*, *LIS* y *SIS* poseen un mejor rendimiento que *FIFO* y claramente que *NFS*.

El resto del artículo está organizado de la siguiente manera. En la sección 2 presentaremos el modelo sobre el que se lanzarán las simulaciones o ejecuciones reales. En la sección 3 continuaremos con el entorno de ejecución, introduciendo el simulador utilizado así como las modificaciones efectuadas en el kernel de linux. En la sección 4 comentaremos los resultados obtenidos. Finalmente, en la sección 5 presentaremos nuestras conclusiones y las líneas de un trabajo futuro.

2. Escenarios

Siguiendo el modelo propuesto por $CAQT$, nuestra red ha sido modelada como un grafo dirigido, adoptando una distribución jerárquica. De la misma manera que en $CAQT$ existe un adversario que elige las rutas y los ratios de inyección, las rutas elegidas en esta red tienden a crear un cuello de botella en el que poder valorar la actuación de las políticas de planificación. Además, con el objetivo de analizar sus comportamientos en una situación próxima a la sobrecarga, se han ajustado en los escenarios distintos ratios de transmisión en torno al nivel de saturación de la red.

El modelo sobre el que se han realizado las ejecuciones y simulaciones, el cual puede verse en la Fig. 1, consta de once nodos organizados conforme a una topología jerárquica. De estos once nodos, siete emiten paquetes a través de tres routers teniendo como objetivo a un receptor común. La elección de este tipo de topología corresponde a la semejanza con una red real en la cual no existen ciclos y las conexiones de los nodos se ajustan a una *distribución de Pareto(20-80)*.

En este modelo que estamos describiendo se puede razonar la existencia de paquetes con un mayor tiempo en la red, procedentes de los emisores que se encuentran a tres saltos del receptor (nodos situados más a la izquierda en la Fig. 1), y paquetes más jóvenes procedentes de los emisores a tan solo un salto del receptor. Esta coexistencia de paquetes de distinta procedencia junto a la competencia de distintos emisores por atravesar un enlace común, han conformado un marco incomparable para observar el comportamiento de las distintas políticas.

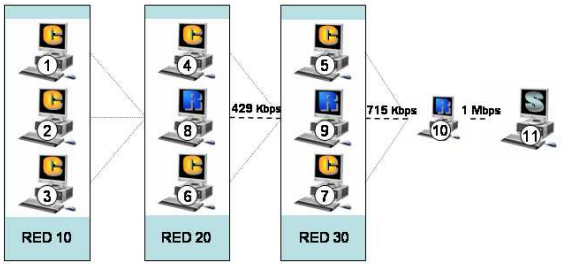


Figura 1: Topología jerárquica de once nodos.

Con el objetivo de evaluar las características de estas políticas de planificación, se han organizado una serie de escenarios dependientes de las políticas inmersas y de once cargas de la red distintas. Como ya se ha indicado anteriormente, la carga de la red estaba controlada por el número de paquetes que los emisores inyectan en la red por unidad de tiempo, siguiendo además una distribución exponencial. Para seleccionar el rango de valores para el tiempo transcurrido entre la inyección de dos paquetes, que oscila entre los cincuenta y los sesenta milisegundos, previamente se calculó el valor teórico de saturación de las colas. Una vez conocido este valor teórico se seleccionó un rango que nos permitiera observar el comportamiento de las políticas antes de la saturación y su progresión a medida que la cola eliminaba paquetes debido a la congestión.

El tamaño de los paquetes se fijó en *1058* bytes, siendo constante a lo largo de toda la emisión. De esta cantidad *1000* bytes corresponden a datos, *20* a las cabeceras *IP* (Internet Protocol), *16* a las opciones de la cabecera, *8* a la cabecera *UDP* y *14* a la cabecera *Ethernet*. El ancho de banda de los enlaces no se limitó excepto en aquellos que correspondían a la salida de los routers. Además, para asegurar que todos los nodos pudieran hacer llegar sus paquetes al objetivo, se fijaron los siguientes anchos de banda: el último enlace antes del receptor fue limitado a *1Mbps*, el enlace central se fijó en *715Kbps*, y al primero se le concedieron *429Kbps*. El tamaño de las colas se estableció en *1000* paquetes. Por último, la duración de los experimentos fueron de *6000 segundos*, de los cuales se ignoraron los *1000* primeros, asumiendo que este es un periodo de tiempo suficiente para la estabilización de la red.

3. Entornos

Como citábamos en la sección 1.1, en este artículo se presentan resultados de dos entornos totalmente distintos. Por un lado se han realizado ejecuciones en un entorno real y por otro lado se han desarrollado un conjunto de simulaciones modificando un simulador existente (J-Sim).

3.1. Entorno de ejecución

Uno de los principales puntos de interés de este artículo reside precisamente en este hecho. Hasta el momento en el que se redactó este texto, no se habían publicado resultados del comportamiento, bajo *CAQT*, de estas políticas en un entorno real. Todas las ejecuciones en este trabajo han sido llevadas a cabo sobre un conjunto de once máquinas con un procesador Pentium IV a 2.80GHz y una distribución Debian 3.0 de linux con un kernel 2.6.7. Sobre este sistema, y siguiendo el modelo presentado en la sección 2, siete de ellas realizaron la función de emisor, una la función de receptor, y las tres restantes se comportaron como routers. La interconexión del conjunto de nodos se realizó a través de un switch, dedicando una interfaz de red por cada conexión con otro nodo. De esta forma, tanto los emisores como los receptores tenían una sola interfaz, mientras que a los routers se les instalaron cuatro tarjetas de red.

Una vez construida la red, el siguiente paso debía ser sincronizar los relojes del sistema. Debido a que algunas de las políticas a evaluar (LIS y SIS) realizan su planificación confiando en la existencia de una hora global, se optó por instalar *NTP* (Network-Time-Protocol) [13] en el sistema.

Con la red montada y sincronizada, tan solo faltaba que los routers realizaran la planificación deseada y que una aplicación pusiera a prueba las políticas. Esta aplicación fue desarrollada con una arquitectura cliente-servidor. El software emisor era el encargado de construir paquetes a nivel IP, para que las cabeceras pudieran ser construidas con las opciones correspondientes, como, por ejemplo, el sello de tiempo. Estos paquetes también contienen una cabecera *UDP* de cara a poder diferenciar los flujos de información. Como hemos dicho, esta emisión se realizó según una distribución exponencial en la que el límite superior estaba incluido como un parámetro. El programa destinatario de los paquetes extraía información relevante de cada paquete, para posteriormente realizar cálculos estadísticos sobre la calidad de servicio en la red. Estos cálculos estadísticos forman parte de los resultados mostrados en la sección 4.

En cuanto a los paquetes, objeto de intercambio, debían cumplir varios requisitos. El más importante era que debían respetar el estándar *IP*. Por otro lado, si tenemos en cuenta que las políticas *SIS*, *LIS* y *LISB* planifican las colas en base a su sello de tiempo, y que en una red Ethernet los tiempos de transmisión son muy reducidos, era necesario idear una forma de conseguir una mayor precisión que los actuales milisegundos. Además, la política *LISB* necesita que su sello de tiempo comience con un valor de cero, para ir incrementándose en las colas de los routers que atraviesa. Toda esta problemática se verá resuelta en las siguientes secciones.

3.1.1. Formato de la cabecera *IP*

Para este trabajo se han utilizado dos formatos de cabecera *IP* diferentes. Por un lado se ha utilizado una de las opciones del estándar *IP*, en la que la precisión del sello de tiempo es de milisegundos. Y por otro lado se ha diseñado un nuevo formato de cabecera derivado del anterior, variando únicamente parte de las opciones del paquete *IP* y respetando los campos obligatorios. Estos dos tipos de paquetes respetan las especificaciones de la *RFC-791* [14] sobre *IP*.

Según se indica en la *RFC-791* [14], las cabece-

ras de los paquetes IP están formadas por una parte obligatoria y pueden incluir otra parte opcional. Estas opciones son una serie de campos adicionales que proporcionan funciones de control necesarias o útiles en algunas situaciones pero innecesarias en las comunicaciones más comunes. En el caso que nos ocupa, vamos a utilizar un tipo de opción que corresponde a los sellos de tiempo. Esta opción, además de utilizar 32 bits para guardar una marca de tiempo expresada en milisegundos, puede registrar las direcciones de los encaminadores.

Un segundo tipo de paquete *IP* permite a la opción de sello de tiempo aumentar la precisión hasta los microsegundos. Entre los campos obligatorios de esta opción, existen cuatro bits dedicados a *flags*. En la actualidad, tan solo tres tipos de *flags* están especificados en el estándar [14], por lo que en este artículo planteamos utilizar un nuevo tipo para identificar el formato de sello de tiempo que proponemos. Este nuevo formato tan solo varía del propuesto por la *RFC-791* en el aumento de bits dedicados a la marca de tiempo. Los 32 bits originales, destinados a almacenar los milisegundos desde la medianoche, se utilizarían para los segundos, aumentando la marca de tiempo en otros 32 bits para recoger los microsegundos.

Perfilado el diseño final del paquete, como se puede observar en la Fig.2, el último detalle para la puesta a punto del entorno sería la inclusión de las políticas de planificación en los encaminadores.

3.1.2. Modificación del kernel de linux

Iproute es un conjunto de herramientas para linux que permite controlar el comportamiento de las redes *TCP/IP*. Formando parte de esta colección se encuentra *TC*. Esta herramienta, entre otras cosas, puede ser utilizada para gestionar la configuración de la cola del kernel de linux. Por este motivo, para conseguir que los routers utilizasen las políticas que estamos analizando, hemos tenido que crear nuevos módulos con las implementaciones de las nuevas planificaciones, y hemos tenido que modificar el kernel de linux para conseguir cargar estas políticas en la cola.

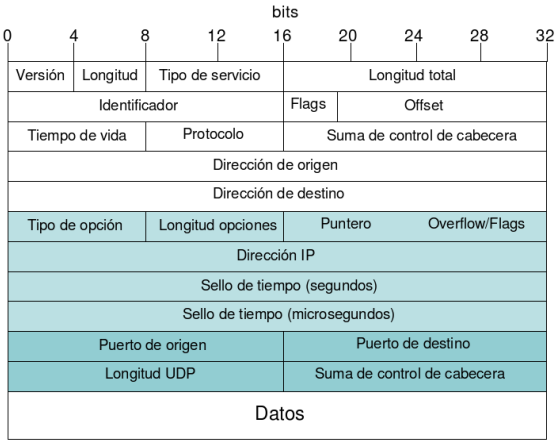


Figura 2: Propuesta de formato de cabecera IP

Una parte de la modificación del núcleo consistió en el desarrollo de nuevas disciplinas de cola. Las disciplinas de cola tienen tres funciones, por un lado dar a conocer a la herramienta *TC* la existencia de estas nuevas políticas. Además analizan los parámetros de configuración, cargando las opciones. Por último, ofrecen información sobre cada política y sus opciones.

Si atendemos a la creación de los módulos del kernel, sobre estos recae la mayor parte del peso de la planificación. Parte de su implementación se encarga del encolado, desencolado y borrado de paquetes de la cola, por lo tanto una gran parte del trabajo en las ejecuciones reales reside en el buen funcionamiento de estas tres operaciones. Estos módulos deberían identificar el formato del paquete, planificar la cola atendiendo a los criterios de cada una de las políticas, y en el caso de políticas como *LISB*, deberían darle un trato distinto al campo del sello de tiempo.

3.2. Entorno de simulación

Las simulaciones, en este trabajo, han sido llevadas a cabo usando el simulador J-Sim [15]. J-Sim, es un paquete de simulación desarrollado en la Universidad de Ohio, que implementa una arquitectura basada en componentes.J-Sim es una evolución de NS2 desarrollado completamente en Java. J-Sim ha sido diseñado para simular de una manera realista el comportamiento de la red. Esto significa que considera cualquier tipo de variable que tenga un impacto real en los escenarios, incluyendo retardos de propagación, tiempos de procesado de paquetes, etc.

Para el propósito de este trabajo hemos modificado o añadido algunos paquetes de Java para adaptar el simulador a nuestro modelo. En primer lugar se ha modificado el componente encargado de monitorizar la llegada de paquetes, de cara a capturar un mayor número de parámetros no soportados por el simulador original. En segundo lugar, se ha desarrollado un nuevo paquete con el objetivo de añadir la funcionalidad necesaria para aplicar las políticas de planificación mencionadas anteriormente.

4. Resultados

En las gráficas que se muestran a continuación, se pueden ver tres curvas que evolucionan con el tiempo, correspondientes a las medias de tres conjuntos de nodos. Según la Fig. 1 y comenzando por la izquierda, los nodos más alejados de la raíz se denominan *red 10*, siendo los más cercanos la *red 30*. Los emisores centrales corresponden por lo tanto a la *red 20*. También es importante mencionar que estos resultados que se presentan son un trabajo preliminar, permaneciendo abiertos algunos pequeños detalles sin resolver. Aunque en

algunas figuras se pueden encontrar discrepancias entre el modelo real y el simulado, es notable el paralelismo de los resultados obtenidos en ambos entornos para la mayor parte de los casos.

Comenzando por el *throughput*, en las Fig. 3 y 4, en términos relativos observamos que determinadas políticas (FFS, LIS y LISB) reparten el ancho de banda de manera más ecuánime. Esto se debe a que estas políticas ofrecen mayor prioridad a los paquetes que están más cerca de su destino, y que ya han sufrido otros cuellos de botella en la red. Se deduce por lo tanto, que acaban equilibrando los anchos de banda de los nodos de la red. Si atendemos a *SIS*, se puede observar una ligera diferencia entre los resultados de la simulación y de la ejecución real pudiendo ser debido a problemas en estos últimos en cuanto a la sincronización de los relojes [16] mediante *NTP*.

En las Fig. 5 y 6 se pueden apreciar las diferencias en cuanto a la *latencia*. Las políticas *LIS* y *LISB* muestran una mejora sustancial frente a *FIFO*, reflejando una uniformidad total y situándose la media por debajo de los valores de la *red 10* en el caso de *FIFO*. La política *FFS* también manifiesta un comportamiento muy positivo. A pesar de no exponer resultados sobre la desviación estándar, debemos mencionar que las políticas *FFS*, *LIS* y *LISB* alcanzan unas medias siempre por debajo de las correspondientes a FIFO. En el caso de *NFS* se siguen acusando los problemas en relación a la topología de la red, pudiendo corregirse este comportamiento en redes de mayor tamaño. Estos resultados ponen de relieve la importancia de algunas de estas políticas como por ejemplo *LIS*.

Por último y en cuanto al jitter se refiere, las Fig. 7 y 8 continúan evidenciando unas mejores prestaciones por parte de las políticas *LIS*, *LISB* y *FFS* frente a *FIFO*. La media del jitter en estas políticas se presenta totalmente uniforme, al mismo tiempo que la media de la desviación estándar también resulta más baja que en el caso de *FIFO*.

5. Conclusiones

En este artículo se han importado las principales ideas y resultados del modelo teórico *CAQT*, para desarrollar un conjunto de simulaciones y ejecuciones en un sistema real. Esto nos ha permitido comparar el comportamiento de FIFO, como una política de planificación, frente a otras nuevas políticas, cuyas mejores propiedades de estabilidad han sido sido probadas en el marco de este modelo. Con este objetivo, nuestra investigación se ha centrado en el diseño de estas simulaciones y ejecuciones sobre una red jerárquica, además de la medida de tres de los principales parámetros de la calidad de servicio (QoS): el throughput, la latencia y el jitter. Los resultados obtenidos sugieren que, al menos para la topología utilizada, las políticas LIS y LISB pueden proporcionar ventajas significativas frente a FIFO y otras políticas, en cuanto al tráfico de tiempo real se refiere.

Estos resultados abren un amplio campo de investigación en el que desarrollar topologías de red más grandes y complejas (toro, *scale free*, etc.), donde analizar estas nuevas políticas, incluyendo la coexistencia de diferentes políticas de planificación dentro de la misma red. Finalmente, sería deseable emplear aplicaciones reales en la generación de tráfico, para verificar los resultados aquí obtenidos.

Referencias

[1] Rene L. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[2] Rene L. Cruz. A calculus for network delay, part ii: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[3] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1(3):344–357, 1993.

[4] Abhay K. Parekh and Robert G. Gallagher. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Trans. Netw.*, 2(2):137–150, 1994.

[5] Matthew Andrews. Instability of fifo in session-oriented networks. *J. Algorithms*, 50(2):232–245, 2004.

[6] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson. Adversarial queuing theory. *Journal of the ACM*, 48(1):13–38, 2001.

[7] Matthew Andrews, Baruch Awerbuch, Antonio Fernández, Tom Leighton, Zhiyong Liu, and Jon Kleinberg. Universal-stability results and performance bounds for greedy contention-resolution protocols. *J. ACM*, 48(1):39–69, 2001.

[8] Rajat Bhattacharjee, Ashish Goel, and Zvi Lotker. Instability of fifo at arbitrarily low rates in the adversarial queueing model. *SIAM J. Comput.*, 34(2):318–332, 2004.

[9] Juan Echage, Vicent Cholvi, and Antonio Fernandez. Universal stability results for low rate adversaries in packet switched networks. *IEEE communications letters*, 7(12), December 2003.

[10] María J. Blesa, Daniel Calzada, Antonio Fernández, Luis López, Andrés L. Martínez, Agustín Santos, María J. Serna, and hristopher Thraves. Adversarial queueing model for continuous network dynamics. *Theory of Computing Systems*, in press.

[11] Agustín Santos, Antonio Fernández, and Luis

López. Evaluation of packet scheduling policies with application to real-time traffic. In *V Jornadas de Ingeniería Telemática*, Sep. 2005.

[12] A.S. Tanenbaum. *Computer Networks*. Pearson Education International, forth edition, 2003.

[13] Dave L. Mills. Network time protocol (NTP). Network Working Group Request for Comments: 958, 1985.

[14] RFC791. Internet protocol. September 1981. DARPA Internet Program Protocol Specification.

[15] Hung-Ying Tyan. J-Sim home page, 2005. Available on-line at http://www.j-sim.org.

[16] Juan Céspedes et al. Performace of scheduling policies in adversarial networks with non synchronized clocks. In *Proceedings of the IEEE Symposium on Computers and Communications*, July 2007.

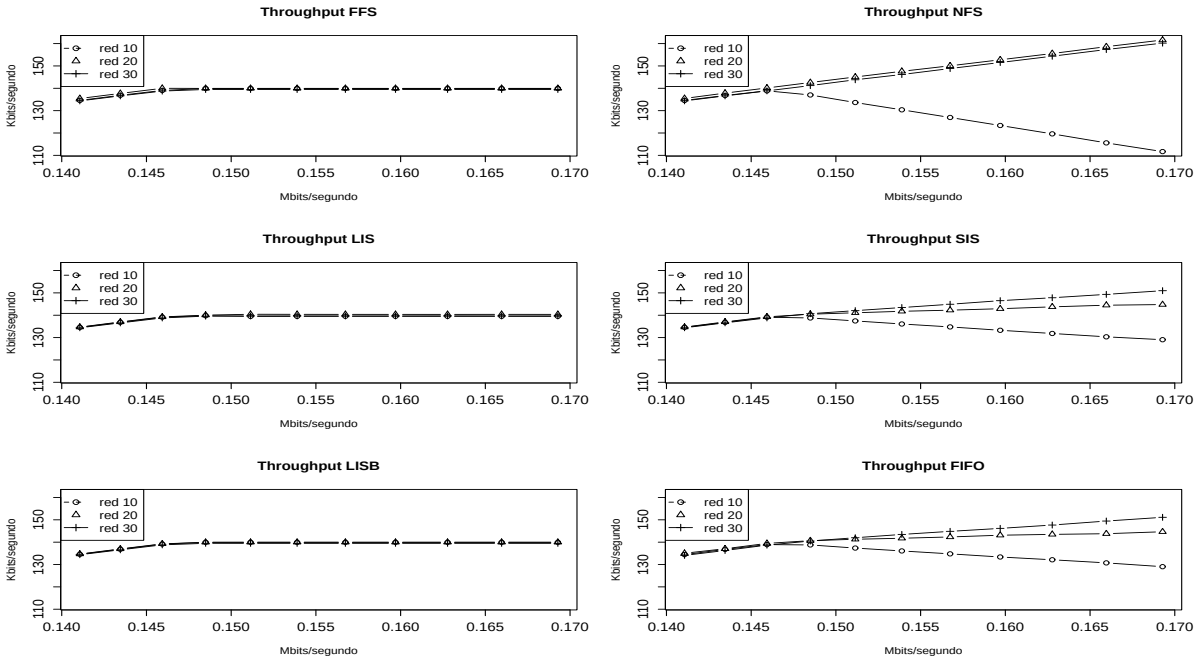


Figura 3: Media del throughput por grupos de nodos experimentado en el entorno real.

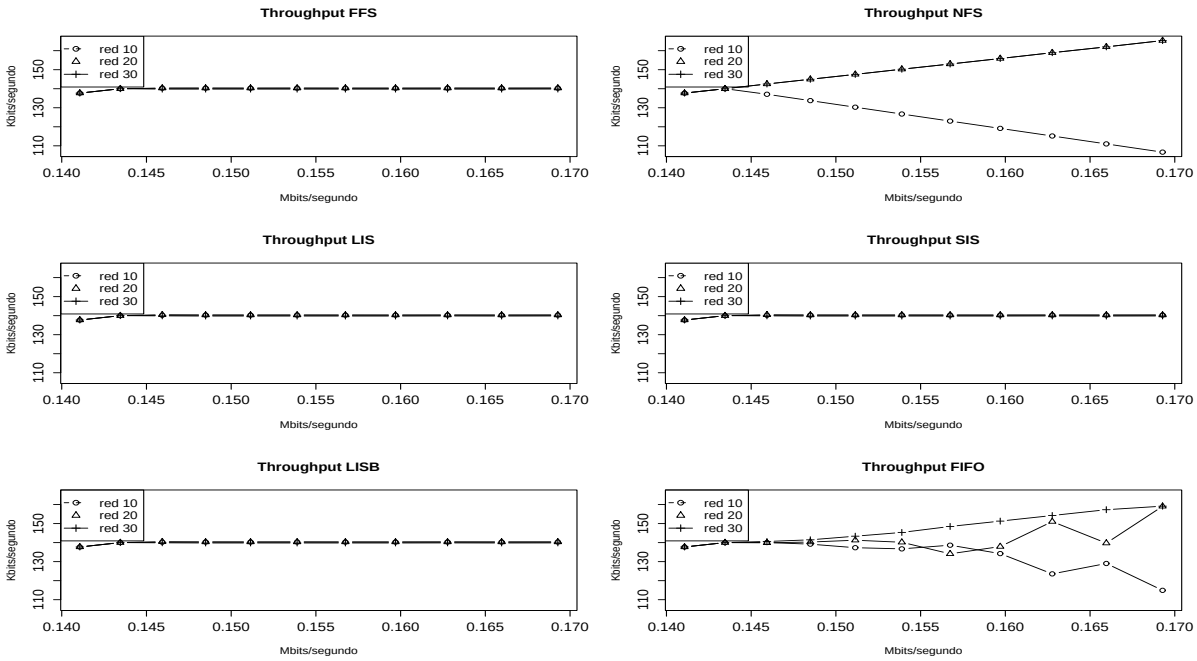


Figura 4: Media del throughput por grupos de nodos experimentado en el simulador.

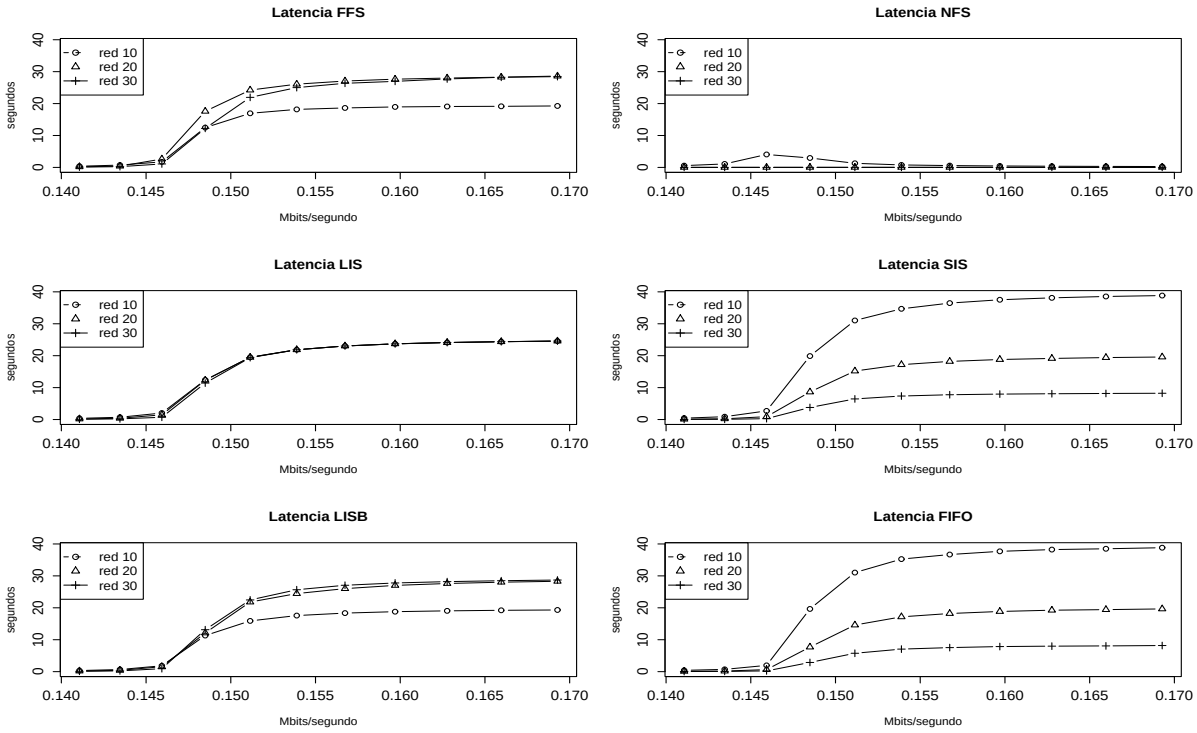


Figura 5: Media de la latencia por grupos de nodos experimentado en el entorno real.

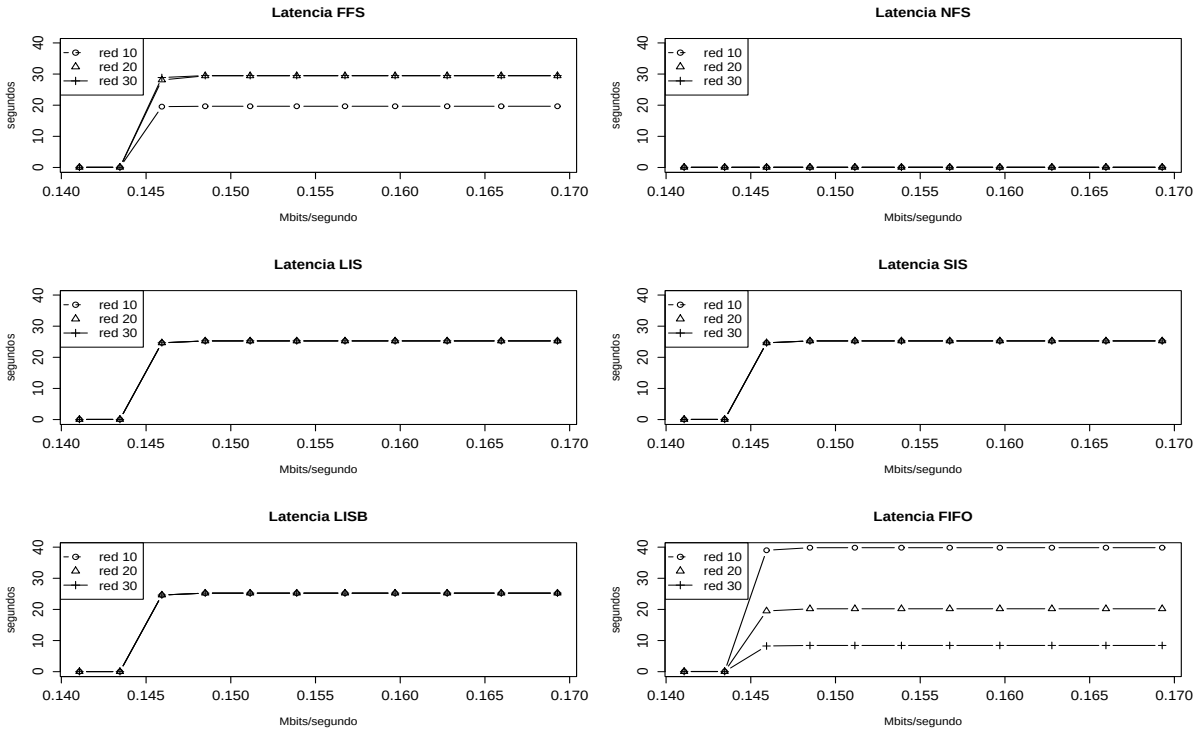


Figura 6: Media de la latencia por grupos de nodos experimentado en el simulador.

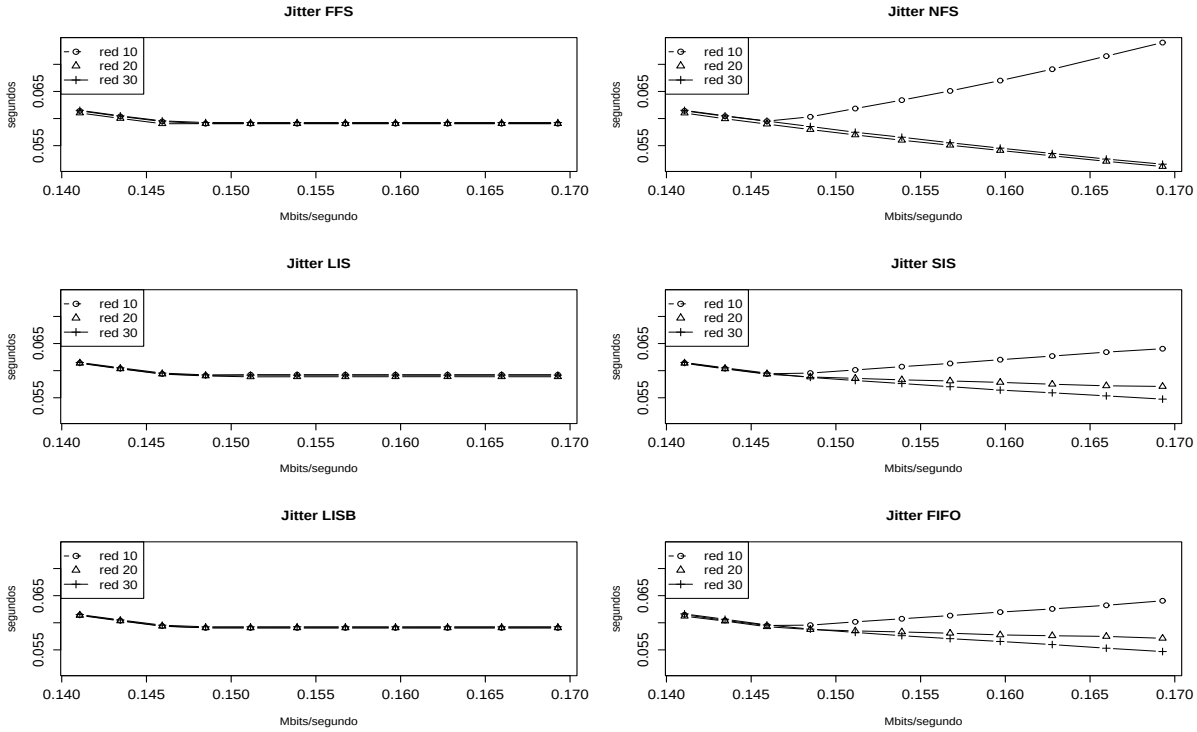


Figura 7: Media del jitter por grupos de nodos experimentado en el entorno real.

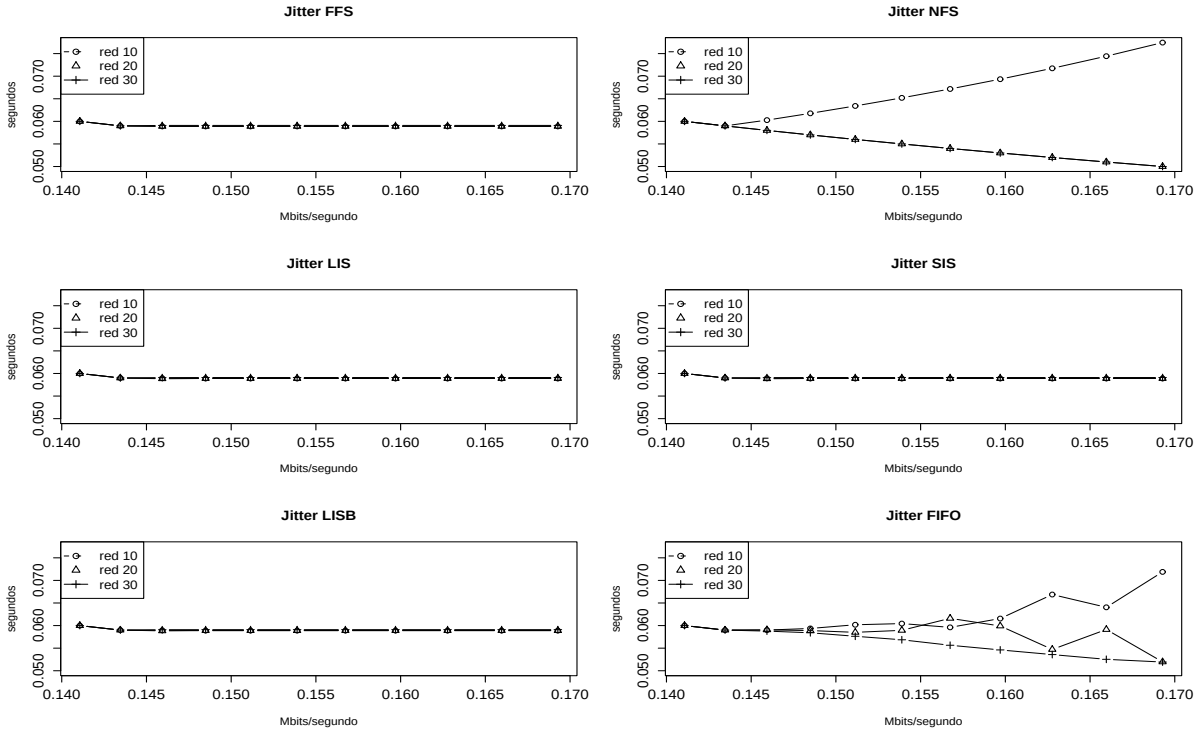


Figura 8: Media del jitter por grupos de nodos experimentado en el simulador.