UNIVERSIDAD NACIONAL DE INGENIERIA
Facultad de Ingeniería Eléctrica y Electrónica

Eficiente controlador interno de células en un conmutador ATM bajo tráfico auto-similar

TESIS

Para optar el grado de Maestro en Ciencias
Mención: Ingeniería Electrónica. Especialidad: Telecomunicaciones

Presentada por:

Marino Ricardo Lent Wong

Lima - Perú

Extracto

Se prevee que las redes ATM (Modo de Transferencia Asíncrono) reemplazarán en los próximos años a las redes de datos actuales y servirán como un sistema de transporte unificado de alta velocidad para soportar una gran cantidad de servicios multimedia existentes o por venir. Para soportar el amplio ancho de banda demandado, se necesitan de nodos de conmutación capaces de manejar eficientemente las altas tasas de llegada, procesamiento y salida de células.

Se han propuesto numerosos esquemas en la literatura para definir la técnica de conmutación a emplear. De los esquemas propuestos, uno de los más simples de implementar, es la arquitectura paralela *crossbar* con buffers FIFO en los puertos de entrada (donde la primera célula que entra es la primera célula que sale). Este esquema tiene la desventaja de tener un pobre rendimiento de flujo de células debido al bloqueo que ocasionan las primeras células de los buffers FIFO. Esto hace que el uso de este esquema sea poco práctico.

Una manera efectiva de resolver el problema del bloqueo causado por la primera célula en los buffers FIFO es utilizar buffers separados para cada puerto de salida en cada puerto de entrada, y construir un elemento de selección que determine la secuencia de entrega de células de los buffers. Debido a la alta velocidad que se requiere para realizar la decisión de conmutación interna, las técnicas de computación secuenciales no ofrecen una alternativa práctica porque su complejidad y lentitud

crecen con el número de puertos del conmutador. Las redes neuronales artificiales sí podrían funcionar adecuadamente como elemento de selección interna por su alta velocidad de procesamiento.

En este trabajo de tesis utilizamos la red neuronal Hopfield propuesta originalmente por Marrakchi y Troudet (1989) y la adaptamos para funcionar como elemento de decisión en un conmutador ATM crossbar con buffers FIFO en los puertos de entrada. Se propone una modificación de la red neuronal original para evitar estados no deseables en la red y aumentar así el flujo del conmutador, mejorando su función de penalización.

Para evaluar el rendimiento del conmutador se construye un modelo de simulación, y se estudia el sistema utilizando modelos de tráfico auto-similares. Como lo han demostrado estudios recientes, el uso de los procesos auto-similares para modelar el tráfico de datos permite capturar mejor las características del tráfico real que los modelos tradicionales de tráfico como Poisson. Originalmente, Marrakchi y Troudet sólo estudiaron el flujo obtenido con la red neuronal usando matrices de demanda arbitrarias. Una importante contribución de esta tesis es estudiar el comportamiento del flujo del conmutador controlado por una red neuronal Hopfield usando condiciones más reales para el tráfico de entrada.

Debido a lo reciente del tema, existe poca investigación sobre el efecto del tráfico auto-similar en los modelos de redes de comunicaciones. Con el fin de aportar conocimiento en este campo, se analiza el efecto en el rendimiento del modelo de conmutación usando tanto secuencias sintéticas de procesos auto-similares como secuencias modeladas como un proceso de Poisson, y los resultados se comparan y analizan.

Además del flujo del modelo propuesto, en este estudio también se evalúan otros parámetros importantes de la calidad de servicio: la probabilidad de pérdida de células y el retardo promedio de células en los buffers del conmutador.

El modelo de conmutación propuesto utilizando redes neuronales Hopfield opera efectivamente, elevando el rendimiento del flujo del conmutador con buffers FIFO en los puertos de entrada hasta en un 22.8% y mejorando el retardo promedio de las células hasta en un 35%.

crecen con el número de puertos del conmutador. Las redes neuronales artificiales sí podrían funcionar adecuadamente como elemento de selección interna por su alta velocidad de procesamiento.

En este trabajo de tesis utilizamos la red neuronal Hopfield propuesta originalmente por Marrakchi y Troudet (1989) y la adaptamos para funcionar como elemento de decisión en un conmutador ATM crossbar con buffers FIFO en los puertos de entrada. Se propone una modificación de la red neuronal original para evitar estados no deseables en la red y aumentar así el flujo del conmutador, mejorando su función de penalización.

Para evaluar el rendimiento del conmutador se construye un modelo de simulación, y se estudia el sistema utilizando modelos de tráfico auto-similares. Como lo han demostrado estudios recientes, el uso de los procesos auto-similares para modelar el tráfico de datos permite capturar mejor las características del tráfico real que los modelos tradicionales de tráfico como Poisson. Originalmente, Marrakchi y Troudet sólo estudiaron el flujo obtenido con la red neuronal usando matrices de demanda arbitrarias. Una importante contribución de esta tesis es estudiar el comportamiento del flujo del conmutador controlado por una red neuronal Hopfield usando condiciones más reales para el tráfico de entrada.

Debido a lo reciente del tema, existe poca investigación sobre el efecto del tráfico auto-similar en los modelos de redes de comunicaciones. Con el fin de aportar conocimiento en este campo, se analiza el efecto en el rendimiento del modelo de conmutación usando tanto secuencias sintéticas de procesos auto-similares como secuencias modeladas como un proceso de Poisson, y los resultados se comparan y analizan.

Además del flujo del modelo propuesto, en este estudio también se evalúan otros parámetros importantes de la calidad de servicio: la probabilidad de pérdida de células y el retardo promedio de células en los buffers del conmutador.

El modelo de conmutación propuesto utilizando redes neuronales Hopfield opera efectivamente, elevando el rendimiento del flujo del conmutador con buffers FIFO en los puertos de entrada hasta en un 22.8% y mejorando el retardo promedio de las células hasta en un 35%.

Universidad Nacional de Ingeniería
Facultad de Ingeniería Eléctrica y Electrónica

Efficient internal cell controller for an ATM switch under self-similar traffic

THESIS

Requirement for the degree of Master in Science
Field: Electronics Engineering. Major: Telecommunications

by:

Marino Ricardo Lent Wong

Lima - Perú

Abstract

It is likely that the ATM (Asynchronous Transfer Mode) networks will replace the current data networks in the next years and will become an unified high-speed transport system to support a large amount of existing or future multimedia services.

In order to support the large bandwidth required, it is necessary to have suitable switching nodes to handle efficiently high-speed arriving rates, cell processing and transmission.

There are several proposed schemes in the literature to define the best switching technique. One of the most simple models to implement is the *crossbar* parallel architecture with FIFO buffers in the input ports (where the first cell to enter is the first cell to leave). This architecture has one drawback, poor performance in the cell throughput because of the blocking by the first cell in a FIFO buffer. This phenomenon makes this architecture impractical.

One effective way to solve the blocking problem of the first cell in a FIFO buffer is to use separated buffers for each output port at each input port, and implement a, selection element to define the cell delivery sequence from the buffers. Because of the high internal speed required to perform the internal switching decision, the computational sequential techniques do not offer a practical solution because of their complexity and increasing delay when the number of ports in the switch grows. Artificial neural networks could work adequately as an internal selection element due to their high processing speed.

In this thesis we use a Hopfield neural network proposed by Marrakchi and Troudet (1989), adapted to work as a decision element in an ATM crossbar switch with buffers in the input ports. We propose a modification to the original network to avoid certain undesired states in order to increase the throughput of the switch, improving its penalty function.

We implement a simulation model of the switch to evaluate its performance using self-similar traffic models. Recent studies have demonstrated that self-similar process can represent more accurately the features of the real network traffic than traditional traffic models such as Poisson. In their original work, Marrakchi and Troudet only evaluated the neural network using matrices of an arbitrary form. One important contribution of this thesis is the study of the throughput behavior in a switch controlled by a neural network using realistic conditions to model the input traffic.

Because the discovery of the self-similar nature in the real traffic is very recent, there is very little research done on the study of the effects of self-similar traffic models in communication network models. To feed the knowledge on that field, we analyze the performance of the switch using both self-similar process and Poisson process, and we compare and analyze the results.

Besides the throughput in the proposed model, in this study we also evaluate other important parameters in the quality of service: the cell loss probability and the cell average delay in the buffers.

Our proposed switching model using Hopfield neural networks operates efficiently, increasing the performance of the throughput in the switch with FIFO buffers in the input ports up to 22.8%, and improving the average cell delay up to 33%.

Tabla de Contenido

Capítulo 1

Introducción

Las redes ATM (Modo de Transferencia Asíncrono) son redes de conmutación de paquetes, en donde cada paquete, celda o célula en la red, es transmitida en forma independiente. En ATM el transporte es orientado a la conexión, por lo que se requiere el establecimiento de conexiones extremo-extremo antes de que la transferencia de información se inicie.

Los nodos de conmutación ATM o conmutadores ATM transportan las células desde sus puertos de entrada a sus puertos de salida a través de una estructura de conmutación interna, usando información almacenada en sus tablas de encaminamiento e información contenida en la cabecera de cada célula. Ambos parámetros son definidos en la fase de establecimiento de la conexión.

Los conmutadores ATM están compuestos por una estructura de conmutación, un conjunto de buffers y un controlador. El diseño de estos elementos determina el rendimiento general del conmutador.

El parámetro más utilizado para medir el rendimiento de un conmutador es el flujo [1]. Otros parámetros importantes son: la probabilidad de pérdida de células y el retardo promedio que sufren las células dentro del conmutador (Ver apéndice para una discusión sobre el protocolo y los tipos de conmutadores ATM).

Un conmutador ATM puede ser modelado como un sistema de colas, y para su evaluación es preciso tener en cuenta los siguientes puntos [1]:

1. El proceso de llegada de paquetes (la estadística de llegada de los paquetes).
2. La distribución de duración de los paquetes.
3. La política de servicio.

En la teoría de colas, el proceso de llegada de paquetes se asume que es un proceso de Poisson. Ha habido mucha investigación bajo esta premisa en el diseño, control

Realmente no existe una definición estándar para el flujo. Una definición de flujo (o *throughput*) es la relación entre el número de células que salen del conmutador por unidad de tiempo. En este documento, a menos que se diga lo contrario, asumimos que el flujo es una relación relativa a la capacidad máxima del canal. Lo definimos como la relación entre el número de células que salen y el número máximo de células que puede manejar el conmutador limitado por la capacidad de sus puertos.

y estudio del rendimiento de las redes de datos. Sin embargo, en [2] se demuestra que el tráfico real no sigue un proceso de Poisson.

Podemos asumir que el efecto de la distribución de duración de los paquetes en ATM es despreciable, dado que los paquetes ATM son de longitud fija, por lo que todas las células causan el mismo efecto.

La política de servicio determina el tipo de estructura de conmutación, los tipos de buffers (FIFO, LIFO, etc), la selección interna de células, etc. El diseño de una adecuada política de servicio es fundamental para un buen rendimiento del conmutador.

1.1 Formulación del problema

La estructura de conmutación *crossbar* [3] con *buffers FIFO* en los puertos de entrada es uno de los modelos más sencillos de implementar de las estructuras propuestas para las redes ATM. Sin embargo, su uso se le considera poco práctico dado el pobre rendimiento que presenta respecto al flujo.

La razón de este pobre rendimiento radica en su política de uso de los buffers. En este esquema, de las células en cola en cada buffer, solamente es posible seleccionar la primera de cada una para su transmisión. Esto resulta en un *bloqueo por la primera célula* o *bloqueo HOL (Head of Line Blocking)*: si la primera célula está bloqueada, porque el puerto de salida está ocupado, las otras células en la cola no podrán ser transportadas a otras salidas no ocupadas. Este fenómeno degrada el flujo del conmutador para medianas y altas cargas de tráfico. En la Figura 1.1 se muestra el rendimiento de un conmutador crossbar simétrico de cuatro puertos con buffers FIFO colocados en los puertos de entrada. El rendimiento del flujo se compara con el de un conmutador crossbar con buffers en los puertos de salida, que tiene un comportamiento ideal desde la perspectiva de la conmutación interna. El uso de buffers en los puertos de salida requiere de una alta velocidad interna, proporcional al número de puertos del conmutador, lo que dificulta su crecimiento en número de puertos.

Una solución efectiva al bloqueo HOL es separar en cada puerto de entrada, colas independientes para cada puerto de salida, y crear un controlador que seleccione las células que pueden entregarse dependiendo de los recursos internos del conmutador. El caso se ilustra en la Figura 1.2. Debido a la alta velocidad en la arquitectura paralela del conmutador, y a la rápida decisión de conmutación que se requiere, las

Figura 1.1: Comparación del flujo versus el uso de la red en un conmutador ATM crossbar simétrico de 4 puertos con buffers en los puertos de entrada con otro similar pero con los buffers en los puertos de salida. Se ha modelado el tráfico de entrada como un proceso Poisson.

redes neuronales ofrecen una solución adecuada para funcionar como controlador interno de selección de células.

Los resultados mostrados en la Figura 1.1 fueron obtenidos asumiendo el tráfico entrante como un proceso de Poisson, sin embargo, podemos inferir que los resultados no pueden ser precisos dado que el tráfico real no es correctamente descrito por este modelo.

De acuerdo a. [4], el tráfico real exhibe dependencia de amplio rango similar al que presentan los modelos auto-similares (como los fractales). Esta característica del tráfico real no está presente en los procesos de Poisson, éstos exhiben dependencia de corto rango, demostrable fácilmente porque su función de autocorrelación decae exponencialmente o más rápidamente.

1.2 Trabajos previos

Los trabajos de investigación en conmutación de paquetes son numerosos en la literatura. En [5] se puede encontrar un resumen de las principales tendencias comparándolas cualitativamente. Una comparación cuantitativa puede encontrarse en [6]. [7].

La mayoría de los estudios sobre el rendimiento de conmutadores ATM se han

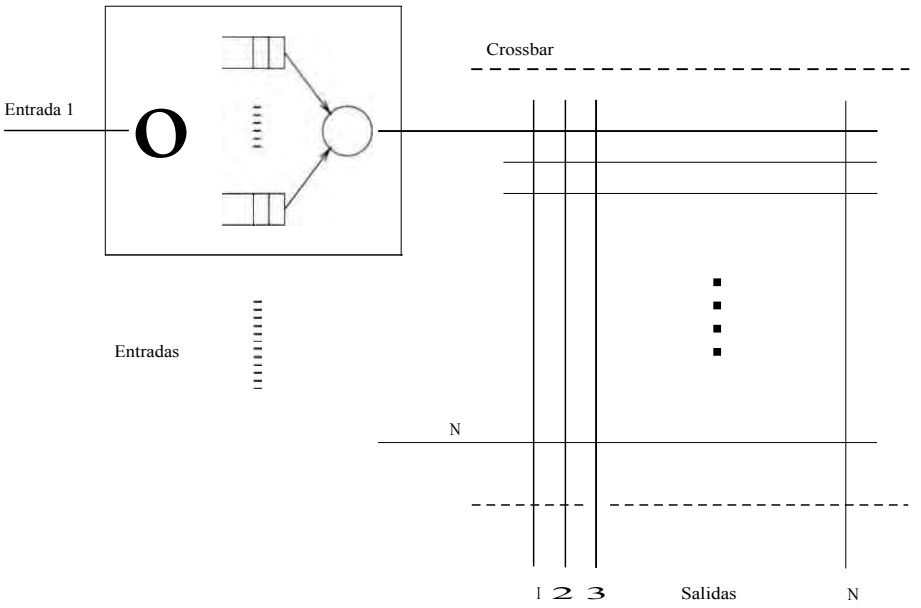


Figura 1.2: Componentes de un conmutador ATM crossbar con buffers FIFO en los puertos de entrada para cada destino.

realizado utilizando procesos de Poisson para modelar el tráfico entrante. Un estudio analítico sobre el rendimiento de un conmutador utilizando cuatro diferentes técnicas en el uso de buffers puede encontrarse en [8]. En esta investigación se determina que el fenómeno HOL en un conmutador con buffers infinitos en los puertos de entrada limita el flujo a $(2-\sqrt{2}) = 0.586$. Para resolver el problema HOL se han presentado varias alternativas en la literatura.

En [9] se propone el uso de buffers internos en cada punto de cruce de la estructura crossbar para retener las células no favorecidas en la selección interna. Utilizando programas orientados al objeto para realizar la simulación y modelos de tráfico Poisson, el autor reporta que puede lograrse un flujo de 97% en el conmutador crossbar con este método. La desventaja de esta propuesta es que es costosa, porque el número de buffers requeridos aumenta con el cuadrado del número de puertos. Además, es difícil implementar físicamente buffers dentro de la estructura de conmutación.

En [10], [11] se proponen algoritmos iterativos para la selección interna de células en un conmutador paralelo de alta velocidad crossbar con buffers en los puertos de entrada y sin bloqueo interno. Estos algoritmos están basados en una modificación del algoritmo de prioridad rotativa o *round robin*. También proponen otros algoritmos basados en la longitud de la cola en los buffers o en el tiempo en que las células están en los buffers. Con estos algoritmos se ha reportado un flujo relativo al tráfico entrante de 98% utilizando tráfico modelado como un proceso Bernoulli. Aunque no se trataran en este trabajo, los algoritmos iterativos parecen ser una

alternativa aceptable y sencilla para mejorar el flujo en un conmutador crossbar. Una comparación cuantitativa de estos algoritmos puede encontrarse en [12].

En [13] los autores presentan una arquitectura de redes neuronales Hopfield para seleccionar en tiempo real la conmutación interna de paquetes, y proponen una arquitectura VLSI para controlar un conmutador de 8 x 8 puertos. La ventaja de usar redes neuronales como elemento de decisión en la conmutación es su alta velocidad de proceso. La desventaja de este esquema es que en su diseño, acepta como una configuración válida la no conmutación interna bajo determinadas circunstancias, aunque existan células por entregar, lo que puede degradar el flujo del sistema. Los autores evaluaron el rendimiento del flujo del modelo usando 500 matrices de demanda arbitrarias, sin seguir ningún esquema de tráfico en particular. La eficiencia reportada para el flujo relativo al número de paquetes que ingresan al conmutador es del 98%. Dado que los autores no lo investigan, surge la duda de cual sería el rendimiento del sistema bajo matrices de demanda que sigan en forma más real las características del tráfico de paquetes de datos como el auto-similar. También surge incertidumbre sobre el comportamiento de otros parámetros importantes en el rendimiento del conmutador como son la probabilidad de pérdida de paquetes y el retardo promedio de los paquetes en los buffers.

En [2] los autores establecen que ni el modelo Poisson ni el Poisson compuesto pueden modelar acertadamente el tráfico en una red de área local en anillo, y proponen un modelo alternativo bautizado "trenes de paquetes".

Analizando el tráfico en Internet, en [14] se descubre que el tráfico de las redes de área extendida, no puede ser modelado exactamente en el sentido estadístico, pero que se pueden construir modelos analíticos simples con una buena aproximación. Una ampliación de este trabajo se da en [15], donde se evalúan 24 conjuntos de datos representando las llegadas de paquetes TCP de área extendida para determinar el error introducido cuando se les modela con procesos Poisson. Se demuestra que ciertos servicios pueden ser modelados aproximadamente por Poisson mientras que otros se desvían considerablemente de este modelo.

En [16], [4], [17] los autores demuestran que el tráfico de las redes de área local Ethernet es estadísticamente auto-similar y que ninguno de los anteriores modelos para representar el tráfico puede capturar su inherente característica fractal. Las conclusiones a las que llegan las fundamentan con un riguroso análisis estadístico de cientos de millones de medidas de alta calidad de tráfico Ethernet. Un trabajo similar se presenta en [18], [19] donde se comparan las medidas obtenidas en una

red de área local Ethernet con varios modelos estadísticos.

En [20], [21] los autores analizan conjuntos de datos de tasa variable (tráfico VBR) generado por diferentes codificadores de vídeo representando una gran variedad de escenas. Luego de un extenso estudio estadístico, concluyen que la dependencia de amplio rango es una característica inherente de la escena (es decir, si la fuente es vídeo teléfono, vídeo conferencia, etc.) y del codificador. Demuestran que es posible diferenciar el tipo de fuente de vídeo en base a la dependencia de amplio rango que presente la muestra estudiada.

En [22] los autores abordan el tema de modelar el tráfico real de paquetes usando tres enfoques: modelos Markovianos tradicionales, modelos auto-similares y modelos no estacionarios, y concluyen que los modelos auto-similares son consistentes con la mayoría de los conjuntos de datos analizados, por los que serán relevantes para los trabajos futuros en ingeniería de redes. Los mismos autores presentan en [23] un resumen de los avances en el modelamiento fractal del tráfico.

Un modelo sintético para generar tráfico auto-similar Ethernet que capture la autocorrelación y la dependencia de amplio rango que ha sido observada en el tráfico LAN Ethernet real se presenta en [24]. Los autores sugieren además, que el descubrimiento de la naturaleza auto-similar del tráfico traerá serias implicaciones en los modelos de conmutación ATM previamente diseñados.

Se han propuesto otras técnicas para la generación de procesos auto-similares [25], [26], [27], [28], [29], [30]. Particularmente interesante por su rápida velocidad de convergencia es el método propuesto por Paxson en [31] utilizando la transformada rápida de Fourier.

Aunque el estudio de los efectos del modelamiento auto-similar del tráfico en los sistemas de comunicaciones está en su fase inicial de desarrollo, ya se han presentado algunos trabajos de investigación en este campo [32], [33].

En [34] se estudia el flujo y la probabilidad de pérdida de células en un conmutador crossbar con sistemas de buffers en los puertos de entrada y en los puertos de salida. Se demuestra que los resultados obtenidos con modelos geométricos difieren sustancialmente de los resultados con tráfico auto-similar. En [35] se llega a similares conclusiones.

Una demostración de que la dependencia de amplio rango del tráfico real tiene un gran impacto en los sistemas de colas puede encontrarse en [36]. En este investigación, los autores llegan a dos importantes conclusiones sobre la dependencia de amplio rango del tráfico real: 1) su efecto en el comportamiento de un sistema de

colas es medible, y 2) es de crucial importancia para los problemas de ingeniería de tráfico de paquetes.

En [37] los autores analizan el comportamiento de un buffer en un conmutador ATM con buffers FIFO usando modelos analíticos y asumiendo tráfico de entrada de naturaleza auto-similar. Determinan que la probabilidad de pérdida de células del buffer decrece con el tamaño del buffer en forma algebraica y no exponencial como lo determinan los modelos Markovianos tradicionales.

Todos los trabajos de investigación recientes en tráfico real de paquetes verifican su naturaleza auto-similar, lo que cuestiona la validez de los sistemas diseñados con modelos de tráfico tradicionales dado que la auto-similaridad implica un mayor nivel de ráfagas de paquetes en la red que se manifiesta en diversas escalas de tiempo, no previsto por los modelos de tráfico tradicionales. Esto hace pensar que en el diseño de los sistemas usando estos modelos de tráfico se han asumido tamaños de buffers inferiores a lo realmente requerido para obtener la calidad de servicio esperado.

1.3 Objetivo de esta tesis

El objetivo de este trabajo de tesis es resolver el problema del bloqueo HOL (o bloqueo por la primera célula) en un conmutador ATM crossbar con buffers FIFO en los puertos de entrada para mejorar su flujo de células, y evaluar el efecto que tienen los modelos de tráfico auto-similares en el rendimiento del conmutador.

Por su alta velocidad de proceso, se han escogido las redes neuronales para diseñar el controlador que maneje la selección interna en el conmutador para incrementar el flujo. El rendimiento del conmutador propuesto se comparará con el de los modelos con buffers FIFO únicos por puerto de entrada, y con buffers FIFO en los puertos de salida.

Los parámetros escogidos, por ser los más importantes para evaluar la calidad de servicio del conmutador, son: el flujo, la probabilidad de pérdida de células y el tiempo promedio de las células en el conmutador.

1.4 Organización de la Tesis

El Capítulo 2 es una discusión sobre los recientes descubrimientos sobre la naturaleza del tráfico real de paquetes de datos y la investigación relacionada a la aplicación de los procesos auto-similares en su modelamiento. Se presentan también

métodos para verificar si una secuencia tiene dependencia de amplio rango, y métodos para generar un proceso auto-similar. En el Capítulo 3 se sustenta el diseño de una red neuronal Hopfield para usarse como controlador en un conmutador ATM sin bloqueo interno. El desarrollo de los modelos y la simulación se discute en el Capítulo 4. En este capítulo se validan los modelos construidos con resultados analíticos previamente presentados. El estudio está basado en resultados de simulación y en explicaciones intuitivas de las observaciones. El Capítulo 5 cubre el análisis de los resultados de la simulación, mostrando las diferencias entre los resultados obtenidos usando modelos de tráfico auto-similares y Poisson. Finalmente, un resumen de los resultados relevantes de esta tesis y recomendaciones para futuros trabajos de investigación se muestran en el Capítulo 6.

Capítulo 2

Naturaleza del tráfico de datos

La validez de un análisis de colas en un sistema de comunicaciones depende de la validez del modelamiento del proceso de llegada de paquetes. Para el modelamiento del proceso de llegada de paquetes se han utilizado comúnmente procesos de Poisson, que presentan dependencia de rango medio (es decir, cuyos elementos son independientes), porque este modelo tiene propiedades teóricas interesantes que facilitan el análisis del sistema de colas.

Sin embargo, como se demostró en [2] y en posteriores casos estudiados, el tráfico no es modelado correctamente con el proceso de Poisson, lo que explica porque los resultados hallados con el análisis de colas usando procesos de Poisson difieren sustancialmente de los reales [38], [39].

Recientes trabajos de investigación [4], [15] han demostrado que para algunos entornos, el tráfico real es auto-similar y no Poisson, con lo que que ponen en duda los resultados de los estudios que usan procesos de Poisson como modelos del tráfico de entrada.

La característica más importante de los procesos auto-similares, desde el punto de vista del rendimiento en una red de datos, es la. persistencia de sus agrupamientos (que aparecen como ráfagas de paquetes en el tráfico real). Con el tráfico Poisson, los agrupamientos ocurren en períodos cortos, pero tienden a suavizarse en períodos largos. Bajo esta premisa, podemos diseñar un sistema de servidores y colas, con buffers esperando que para períodos largos no se presenten significativas variaciones entre los agrupamientos. Esto implica que sólo se requieren de buffers de longitud modesta; una cola podría llenarse en períodos cortos de tiempo, pero en períodos largos los buffers deberían limpiarse. Sin embargo, con tráfico real, los buffers se llenarían más rápidamente de lo que preveen los modelos de Poisson, lo que provocaría una tasa de pérdida de paquetes más alta de lo esperado.

En este capítulo se presentan formalmente los procesos auto-similares, y se discuten técnicas para estimar y generar secuencias auto-similares. Este material se utilizará posteriormente para modelar el tráfico que se usará en la simulación de los modelos de conmutación estudiados.

2.1 Procesos auto-similares

Un fenómeno que es auto-similar parece igual o actúa igual cuando es visto con diferentes grados de "magnificación" o diferentes escalas de una dimensión. La dimensión puede ser el espacio (longitud, altura o profundidad) o el tiempo. En este capítulo discutimos series de tiempo y procesos estocásticos que presentan auto-similaridad con respecto al tiempo.

Un proceso que se repite de igual manera en diferentes escalas de tiempo puede definirse como *exactamente auto-similar,* y puede generarse a partir de una serie de tiempo determinística. Un ejemplo de un fenómeno exactamente auto-similar es el conjunto de Cantor [40].

Es posible estudiar las características del tráfico de datos viendo la secuencia de información como un proceso estocástico, cuyas estadísticas no varían con un cambio en la escala del tiempo. Este proceso podemos definirlo como *estadísticamente auto-similar* o simplemente auto-similar. El comportamiento promedio del proceso estocástico en intervalos cortos es el mismo que en intervalos largos de tiempo.

Los procesos auto-similares han sido definidos de varias maneras en la literatura [41]. A continuación se revisan los conceptos más importantes, primero en el tiempo continuo, y luego en tiempo discreto, que es más relevante para nuestra discusión sobre tráfico de paquetes de datos.

2.1.1 Definición en tiempo continuo

Una definición común de un proceso estocástico auto-similar está basada en el escalamiento directo de una variable continua en el tiempo.

Un proceso estocástico $X(t)$ es estadísticamente auto-similar con parámetro H $(0.5 < H < 1)$ si para cualquier valor real a > 0, el proceso $a^{H}X(at)$ tiene las mismas propiedades estadísticas que $X(t)$. Esta relación puede ser expresada con las siguientes tres condiciones para la media, varianza y autocorrelación:

$$\text{ni} = E[X(t)] = \frac{E[X(at)]}{a^{H}}$$

$$\sigma^{2} = Var[X(t)] = \frac{Var[X(at)]}{a^{2H}}$$

$$r(k) = R_x(t, s) = \frac{R_x(at, as)}{a^{2H}}.$$

El parámetro H es conocido como el parámetro de Hurst o el parámetro de auto-similaridad. El parámetro H es una medida de la persistencia de un fenómeno

estadístico, concretamente es una medida de la longitud de la *dependencia de amplio rango de un proceso*. Un proceso estacionario tiene dependencia de amplio rango, si su función de autocorrelación $r(k)$ es no sumable, es decir, $\sum_k r(k) = \infty$. Un valor de $H = 0.5$ indica la ausencia de auto-similaridad. Mientras H este más cercano a 1, mayor será el grado de persistencia o dependencia de amplio rango del proceso.

2.1.2 Definición en tiempo discreto

Para. analizar secuencias de tráfico reales, se utilizan formulaciones de auto-similaridad para series de tiempo que se discutirán en esta sección. La serie de tiempo está definida como puntos discretos en el tiempo de una función continua X(t).

Para $X = (X_t : t = 0, 1, \ldots)$, un proceso estocástico *covarianza estacionario o estacionario en el sentido amplio (WSS)*, definimos las series de tiempo *m-agregadas* $X^{(k)} = X_m^{(k)}$, $m = 0, 1, 2, \ldots$. sumando la serie de tiempo original en bloques no superpuestos y adyacentes de tamaño k. Esto puede ser expresado como,

$$X_m^{(k)} = \frac{1}{k} \sum_{i=mk-(k-1)}^{mk} X_i$$

por ejemplo, para $X^{(3)}$),

$$X^{(3)} = \frac{X_{3m-2} + X_{3m-1} \quad X_{3m}}{3}$$

Una forma de ver las series agregadas de tiempo es como una técnica de compresión en la escala del tiempo. Podemos considerar a $X^{(1)}$ como la máxima magnificación o mayor resolución posible para esta serie de tiempo. El proceso $X^{(3)}$ es el mismo proceso reducido en magnificación por un factor de 3. Promediando cada conjunto de tres puntos perdemos el detalle que provee la máxima magnificación. Si las estadísticas del proceso se conservan con la compresión, el proceso es auto-similar.

También podemos ver cada punto en las series $X^{(k)}$ como un promedio en el tiempo del proceso X. Para un proceso ergódico, el promedio en el tiempo debería ser equivalente al promedio de conjunto, y la varianza del promedio en el tiempo debería acercarse a cero relativamente rápido mientras crece k. Esto no ocurre en un proceso auto-similar; la varianza se acerca a cero más lentamente que en un proceso estacionario ergódico.

Entonces, un proceso X es exactamente auto-similar, con parámetro D ($0 < D < 1$) si para todo $k = 1, 2, \ldots$ tenemos que,

$$\sigma = Var[X(k)] = \frac{Var[X]}{k^D}$$

$$r(m) = R_{X(k)}(m) = R_X(m).$$

El parámetro D puede ser relacionado con el parámetro de Hurst definido anteriormente con la siguiente ecuación.

$$H = 1 - \frac{D}{2} \tag{2.1}$$

Para un proceso estacionario ergódico $\boldsymbol{D} = 1$, y la varianza del promedio en el tiempo cae a cero con una tasa de 1/k. Para un proceso auto-similar, la varianza del tiempo promediado decae más lentamente.

Una condición menos estricta es la siguiente: Un proceso X es *asintóticamente auto-similar* si para todo m lo suficientemente grande,

$$Var[X^{(k)}] = \frac{Var[X]}{k^L}$$

$$R_{X^{(k)}}(m) \to R_X(m); \boldsymbol{mientras\ k} \to \infty.$$

Así, con esta condición de auto-similaridad, la autocorrelación del proceso agregado tiene la misma forma que el proceso original. Esto sugeriría que el grado de variabilidad, o grado de ráfagas, podría ser el mismo a diferentes escalas de tiempo.

2.2 Dependencia de amplio rango

Una de las propiedades más importantes de los procesos auto-similares es su dependencia de amplio rango. Esta propiedad está definida en términos del comportamiento de la autocovarianza *C(7)* mientras τ se incrementa.

A diferencia de los procesos con dependencia de amplio rango, un proceso con *dependencia de corto rango* satisface la condición de que su autocovarianza decae al menos tan rápido como exponencialmente,

$$C(k) \quad a^{|\cdot|}; \boldsymbol{mientras} \quad |k| \to \infty, \quad 0 < \boldsymbol{a} < 1.$$

Los modelos de tráfico típicamente considerados en la literatura emplean sólo procesos con dependencia de corto rango. Usando la inecuación:

$$\sum_{k=0}^{\infty} X^k \qquad \frac{1}{1 \quad X} \quad ; \qquad <$$

se puede observar que $\sum_k C(k)$ es finita.

Un proceso con dependencia de amplio rango, presenta una función de autocovarianza que decae hiperbólicamente [35]:

$$C(k) \quad |k| \quad ;mientras \quad |k| \text{-4 00, } 0 < D < 1$$

donde D es el mismo parámetro definido anteriormente y que está relacionado con el parámetro de Hurst. En este caso, $\sum_k C(k)$ es infinita.

La dependencia de amplio rango intuitivamente refleja el fenómeno de persistencia en los procesos auto-similares, o dicho de otra manera, la existencia de ráfagas en distintas escalas de tiempo.

2.3 Métodos para determinar el parámetro de auto-similaridad

Esta sección está dedicada a describir métodos estadísticos heurísticos para determinar la dependencia de amplio rango en series de tiempo.

2.3.1 Gráfico de la varianza en el tiempo

Una característica de los procesos estocásticos que presentan dependencia de amplio rango es que la varianza de su media aritmética decrece más lentamente que el recíproco del tamaño de la muestra.

Sea $X^{(k)}(= X_m^{(k)}; k = 1, 2, \ldots)$ un nuevo proceso covarianza estacionario (con función de autocorrelación $r^{(k)}$) obtenido de promediar la serie X original en bloques no superpuestos de tamaño k, esto es:

$$X_m^{(k)} = {}^{1} \qquad +\cdots + X_{km}); \quad m > 1.$$

La especificación de la secuencia $Var(X^{(k)}); k > 1$, es equivalente a una especificación de la función de autocorrelación $r^{(k)}(m); m > 0$.

Para un valor constante finito positivo c, tenemos:

$$Var(X^{(k)}) \; {}^{ti} \; ck \quad ; k \to \infty, \quad 0 < D < 1. \tag{2.2}$$

Así para un valor grande de k, $X^{(k)}$ tiene una estructura de correlación fija, únicamente determinada por D.

Por otro lado, para procesos covarianza estacionarios con dependencia de corto rango (modelos Markovianos), los procesos agregados $X^{(k)}$ tienden a ruido blanco

$$Var(X^{(k)}) \sim ck \quad ; mientras\ k \to \infty.$$

Pueden utilizarse los siguientes pasos para estimar el valor del parámetro de Hurst usando el gráfico de la varianza en el tiempo.

1. Sea k un entero. Para diferentes enteros k en el rango $2 < k < n/2$, y un número suficiente (m) de sub-series de longitud k, calcular las medias de las muestras $X_1(k), X_2(k), \ldots, X_m(k)$ y la media total:

$$\bar{X}(k) = \frac{1}{m} \sum_{j=1}^{m} \bar{X}_j(k).$$

2. Para cada k, calcular la varianza de las muestras $X_j(k)$; $(j = 1, 2, \ldots, m)$ como sigue:

$$Var(X) = \frac{1}{m-1} \sum (\bar{X}_j(k) - \bar{X}(k))^2.$$

Para procesos con dependencia de amplio rango, la varianza de los procesos agregados $X^{(k)}, k = 1, 2, \ldots$ decrece linealmente si los graficamos en un plano con ambos ejes logarítmicos. Las pendientes serán arbitrariamente menores a -1. Podemos encontrar la pendiente de una línea ajustada con el método de los mínimos cuadrados a los puntos resultantes en el plano, ignorando los valores pequeños de k (período transitorio) así como los valores más altos. Luego de la ecuación 2.2 tenemos:

$$log(Var(X^{(k)})) \sim log(c) - Dlog(k). \tag{2.3}$$

De la ecuación 2.3, puede notarse que la pendiente del gráfico de la varianza en el tiempo es $-D$. Usando la relación entre D y H dada por la ecuación 2.1, puede estimarse el valor de H. Este efecto puede verse en la Figura 4.6, donde se muestra la varianza versus tiempo de una secuencia real de paquetes TCP/IP recolectados de una red de área local Ethernet donde la mayor parte del tráfico es desde y hacia Internet.

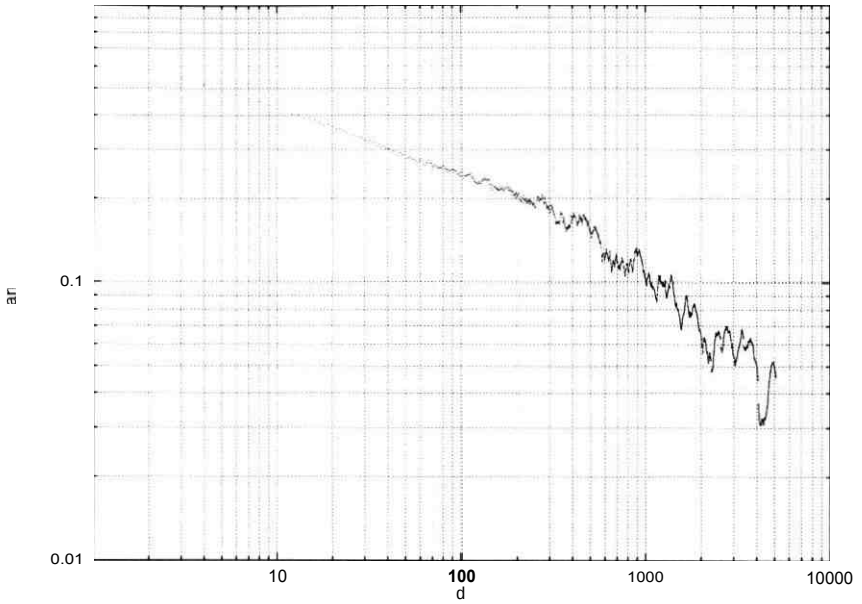


Figura 2.1: Gráfica de la varianza en el tiempo para una secuencia de tráfico real TCP/IP.

2.3.2 Gráfico R/S

El gráfico R/S es otra aproximación heurística para estimar el valor de H en un proceso estocástico. Este método relaciona la medida del rango del proceso con la desviación estándar de la muestra.

Dada una muestra de n observaciones tomadas durante el intervalo de tiempo entre t y t n, denotemos X_k el valor de la muestra tomada en el tiempo t k (o retardo *k), k* $= 1, 2 \ldots$ n.

Sea $X(n)$ la media y $S(n)$ la varianza de la secuencia X_k, $k = 1,2 \ldots$ n. El rango ajustado re-escalado $R(n)/S(n)$ se calcula con [21], [35]:

$$R(n)/S(n) = \frac{}{S(n)}[max(0, W_1, W_2, \ldots, W_n) - min(0, W1, W2, \ldots, W_n)]$$

donde $W_k = (X_1 + X2 + \quad + X_k) - kX(n)$, con $1 < k <$ n.

La estimación de H puede resumirse en:

1. Calcular $Q = (R(n)/S(n))$ para todos los posibles (o al menos para un número suficiente de diferentes) valores de t y d.

2. Graficar en el plano $log(Q)$ versus $log(d)$ para obtener el gráfico R/S también llamado *Diagrama Pox.* Este diagrama tiene generalmente k valores de Q en el eje y, correspondiente a cada valor de k en el eje x.

3. Dibujar una línea recta usando el ajuste de los mínimos cuadrados que corresponde al comportamiento actual de los datos. La pendiente de la línea $log(Q) = a$ -I- $blog(k)$ es un estimado de ***H.***

Un diagrama típico comienza con una zona de transición representando la naturaleza de la dependencia de rango corto en la muestra, pero eventualmente tiende a estabilizarse y fluctuar sobre una línea recta de pendiente ***H.*** La Figura 2.2 muestra la gráfica R/S de la misma secuencia usada anteriormente para obtener el gráfico de la varianza versus tiempo.

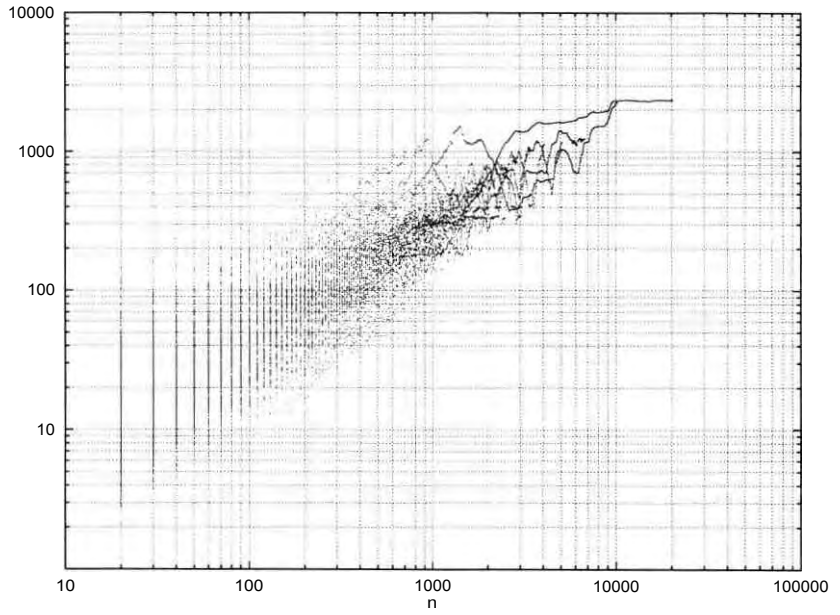


Figura 2.2: Gráfica R/S de una secuencia real TCP/IP.

2.4 Generación de secuencias auto-similares

A continuación se describen algunos de los métodos propuestos en la literatura para la generación de procesos auto-similares y asintóticamente auto-similares.

Uno de estos métodos [4], consiste en agregar un número de procesos regenerativos [42] para obtener procesos auto-similares. Un proceso regenerativo es un proceso que probabilísticamente puede asumir un estado ya asumido anteriormente. Esta aproximación tiene el problema de la lentitud para los cálculos. A mayor valor del parámetro de Hurst, mayor la cantidad de procesos regenerativos para obtener resultados aceptables. Una ventaja de este método es que puede ser implementado con computación paralela. En [4] se asegura que toma de 3 a 5 minutos generar una

secuencia de 100,000 puntos en un computador paralelo con 16,384 procesadores usando este método.

Otro método [15] utiliza una cola $M/G/\infty$ donde los clientes llegan según un proceso de Poisson, y los tiempos de servicios son tomados de una distribución de *cola pesada.* En este modelo, el valor de una muestra en un tiempo t determinado corresponde al número de clientes en el sistema en ese tiempo t. Este modelo produce muestras que son asintóticamente auto-similares. Este método también tiene la desventaja de la lentitud para realizar los cálculos.

Un tercer método llamado *Desplazamiento aleatorio del punto medio (RMD)* está descrito en [27]. Este método tiene la propiedad de ser rápido y existen reportes de que pueden generarse 260,000 observaciones en menos de 2 minutos en una SPARCstation 20. La inconveniencia de este método es que el parámetro de Hurst H de las muestras tiende a ser mayor que el buscado para $0.5 < H < 0.75$ y tiende a ser menor para $0.75 < H < 1.0$.

Un cuarto método está basado en mapas caóticos [23], [22], [29], [30]. La idea con esta aproximación es extender el espacio de estado a uno continuo, y describir la evolución de una variable continua x_n sobre el tiempo discreto usando un mapa no lineal (caótico) *F(.)*. El proceso de generación de paquetes es modelado estipulando que una fuente genera un bloque de paquetes en un momento punta (correspondientes al estado encendido) cuando la variable está sobre un umbral, y no genera paquetes cuando la variable esta bajo este umbral. Escogiendo adecuadamente la función *f(.)* se puede modelar un rango de comportamientos de las fuentes encendido/apagado.

El método descrito en [31] está basado en la transformada rápida de Fourier. La ventaja de este método sobre los otros propuestos es su velocidad de proceso. Toma 80 segundos generar una secuencia de 262,144 puntos en una SPARCstation IPX sin sufrir la alteración del parámetro de Hurst que ocasiona el método RMD. Se describe con más detalle a continuación.

2.4.1 Síntesis de Ruido Fraccional Gaussiano usando el método de la Transformada Rápida de Fourier

Supóngase que conocemos la densidad espectral de potencia $f(\lambda; H)$ de un proceso de ruido gaussiano fraccional (FGN) con la que podemos construir una secuencia de números complejos z_i correspondientes a esta densidad espectral de potencia. Usando la inversa de la transformada discreta de Fourier (IDFT) podemos obtener x_i, la contraparte en el dominio del tiempo de z_i. Dado que x_i tiene (por construc-

ción) la densidad espectral de potencia de un FGN, y dado que la autocorrelación y la densidad espectral de potencia se relacionan [43], entonces está garantizado obtener las propiedades autocorrelacionales de un proceso FGN, que es la característica más importante para la mayoría de las aplicaciones. Este método propone utilizar la transformada rápida de Fourier (FFT) para realizar los cálculos, de allí que toma este nombre.

La dificultad en esta aproximación está en el cálculo preciso de $f(\lambda; H)$, y en encontrar un z_i que corresponda a la densidad espectral de potencia de un FGN. Particularmente no hay ninguna razón a *priori* para asumir que los z_i son individualmente independientes. Además, capturar esta interdependencia puede ser difícil.

Siguiendo a [31], la densidad espectral de potencia de un proceso FGN es:

$$f(\lambda; H) = \mathrm{A}(\lambda; H)[|\mathrm{A}| \quad +\mathrm{B}(\lambda; \mathrm{H})] \tag{2.4}$$

para O $< h <$ 1 y —7i- $<$ A $<$ 7r, donde:

$$\mathrm{A}(\lambda; H) = 2sin(\pi H)\Gamma(2H = 1)(1 - cos\lambda)$$

$$B(\lambda; H) = \sum_{j=1}^{\infty} [(2\pi \quad \lambda)^{-2H-1} + (27\mathrm{i} - \lambda)^{-2H-1}].$$

La sumatoria infinita de la ecuación 2.4 la podemos aproximar por:

$$B(\lambda; H) \quad \text{-I-} \quad + a_2^d + b_2^{'} + a_3 + \cancel{b}^d + \frac{a_3^{-} \quad b_3^{'} \quad a_4^{'} \quad b_4^{'}}{8H\pi} \tag{2.5}$$

donde:

$$d = —2H - 1$$

$$dp = —2H$$

$$a_k = 2k\pi + \mathrm{A}$$

$$b_k = 2k\pi - \lambda.$$

Entonces definirnos $\tilde{f}(\lambda; H)$ como una aproximación de la ecuación 2.4 usando la ecuación 2.5 para $B(\lambda; H)$.

El procedimiento es el siguiente:

1. Construir una secuencia de valores $\mathrm{f_i}$, , $f_{n/2}$, donde $= f$; $H)$, correspondientes al espectro de potencia de un proceso FGN para frecuencias de $2\pi/n$ a 7r.

2. Alterar cada f_i multiplicándolo por una variable aleatoria exponencial independiente con media 1. Llamemos a la secuencia alterada

   Esto se hace porque cuando se estima la densidad espectral de potencia de un proceso usando el periodograma de la muestra, la potencia estimada para una frecuencia dada está distribuida asintóticamente como una variable aleatoria exponencial independiente con media igual a la potencia actual.

3. Construir $z_1, \ldots, z_{n/2}$ una secuencia de valores complejos tales que $|z_i| = f_i$ con la fase de z_i uniformemente distribuida entre O y 27r.

4. Construir $z_1, \ldots, z_{n-1}$ a partir de $z_1, \ldots, z_{n/2}$ como sigue:

$$z'_j = \begin{cases} 0 & ;si\ j = 0 \\ z_j & ;si\ 0 < j < n/2 \\ \overline{z_{n-j}} & ;si\ n/2 < j < n \end{cases} \qquad (2.6)$$

   donde $\overline{z_{n-j}}$ denota el complejo conjugado de z_{n-j}. El término conserva la densidad de potencia usada en construir z_i, pero como es simétrico alrededor de $z'_{n/2}$, ahora corresponde a la transformada de Fourier de una señal con valores reales.

5. Tomando la transformada inversa de Fourier de obtenemos muestras FGN aproximadas x_i.

Más adelante se generará una secuencia con este algoritmo, y se determinará su auto-similaridad usando los métodos discutidos en este capítulo.

Capítulo 3

Redes neuronales Hopfield

Las *redes neuronales artificiales,* o simplemente *redes neuronales,* son modelos computacionales que consisten en la conexión de un número grande de unidades de proceso simples. Estas unidades se conocen comúnmente como *neuronas artificiales* o *nodos.* La conexión entre los nodos se realiza por enlaces asociados con pesos llamados potencia de sinapsis de enlace. El entrenamiento de la red es el procedimiento de encontrar los pesos apropiados para el funcionamiento deseado. Existen varios tipos de redes neuronales; la red neuronal Hopfield que se desarrolla en este capítulo se ajusta a problemas de optimización, como es el tipo de problema que queremos resolver.

3.1 Operación de una neurona artificial

La unidad básica de las redes neuronales artificiales es la neurona. Existen varios tipos de neuronas propuestas que capturan las capacidades esenciales de procesamiento paralelo de información que tienen las neuronas biológicas. El tipo de neurona que utilizaremos realiza una correspondencia no lineal de un conjunto de entradas a una variable única de salida. La salida vi es calculada como la suma ponderada de las entradas u_j de acuerdo a una *función de transferencia* no lineal.

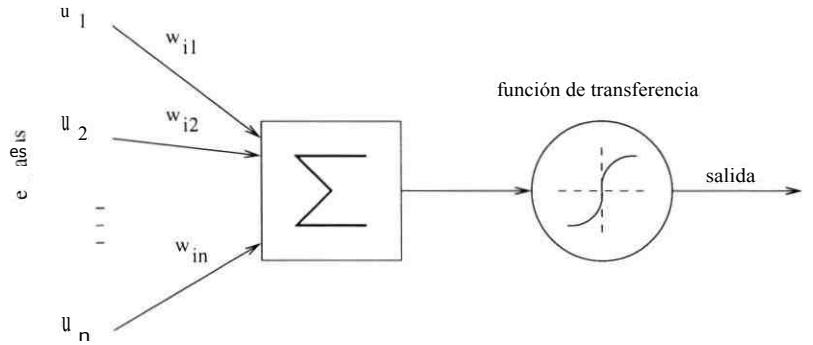


Figura 3.1: Modelo generalizado de una neurona McCulloch-Pitts.

La operación del modelo de la neurona puede ser descrito como

$$v_i = f(\sum w_{ij} u_j)$$

$$f(v_i) = \frac{1}{1 + exp(-u_i\ \tau^p)}$$

donde la función tangente hiperbólica sesgada (conocida también como función sigmoide), ha sido empleada como función de transferencia o función de activación.

La ganancia de la función de transferencia es controlada por un parámetro T. La variable de salida está limitada al intervalo y_i E [0, 1], por la acción de la función de transferencia $f(\cdot)$. El término w_{ij} es el peso para la conexión entre la neurona *i y* la neurona *j,* que representa la eficiencia sináptica del enlace que conecta estas dos neuronas.

3.2 Redes neuronales Hopfield

Las redes neuronales Hopfield se caracterizan por ser redes de una única capa donde la salida de cada neurona se conecta a las entradas de las demás neuronas sin conexión entre la salida y la entrada de una misma neurona.

Las redes neuronales Hopfield son sistemas dinámicos, que evolucionan en el tiempo discreto o continuo (redes neuronales de Gradiente), en función de su energía computacional, la que decrece continuamente hasta alcanzar un *punto de equilibrio* o *mínimo estable* en el espacio de estado.

Típicamente, la función de energía de la red se hace equivalente a una determinada función objetivo (o *penalización)* que requiere minimizarse. La búsqueda de una función de energía corresponde a la búsqueda de la solución de un problema de optimización. En nuestro caso, la función de optimización intentará maximizar el flujo interno del conmutador. Las redes neuronales de Gradiente son ejemplos de sistemas no lineales, dinámicos y asintóticamente estables.

3.2.1 Fundamento matemático

Una red Hopfield consiste de n neuronas que pueden implementarse usando amplificadores operacionales, cada uno con una entrada de voltaje u_i y una salida de voltaje v_i, y con una función de transferencia o activación $f(u_i)$. Los pesos se representan por conductancias w_{ij} que conectan la salida de la *j*ésima neurona a la entrada de la *i*ésima neurona. La red es simétrica $(w_{ij} = w_{ji})$, sin conexión de la salida de una neurona a su propia entrada $(w_{ii} = 0)$.

Cada entrada en la i-ésima neurona tiene una conductancia g_i con tierra, y representa la conductancia de entrada de la i-ésima neurona. Extrayendo esta conductancia, la neurona podría ser considerada como un amplificador libre de absorciones parásitas de corriente. Similarmente, C representa la capacitancia de entrada de la i-ésima neurona. Las capacitancias C_i, para i $= 1, 2, \ldots$ n, son responsables de la dinámica del estado transitorio de este modelo.

Figura 3.2: Implementación de la i-ésima neurona.

La Figura 3.2 representa la i-ésima neurona. La ecuación de las leyes de Kirchoff para este nodo es:

$$i_i + \sum_{j=1, j\neq i}^{N} w_{ji} v_j - \qquad + g_i] = C_i(\frac{du_i}{dt}). \tag{3.1}$$

El lado izquierdo de la ecuación 3.1 representa el total de corriente que ingresa por la capacitancia C_i, y corresponde a la suma de las corrientes de valor $(\mathrm{v_i} - u_i)w_{ij}$, para $j = 1, 2, \ldots,$ n, j i, la corriente i, y la corriente $-g_i u_i$.

Definiendo las matrices C y G como

$$C = diag[C_1, C2, \bullet\bullet\bullet, C_n]$$

$$\mathrm{G} = diag[G_1, G2 \bullet\bullet\bullet \ G_n]$$

y acomodando $u_i(t)$, $v_i(t)$ e i_i en los vectores $\mathbf{u(t)}$, $\mathbf{v(t)}$ e I, podemos reescribir la ecuación 3.1 como

$$\mathbf{C}\frac{d\mathbf{u(t)}}{dt} = \mathbf{W v(t)} - \mathbf{G u(t)} \quad \mathbf{I} \tag{3.2}$$

$$\mathbf{v}(\mathbf{t}) = f[\mathbf{u}(\mathbf{t})].$$

La estabilidad de las ecuaciones puede ser evaluada usando una función de energía computacional generalizada $E[\mathbf{v}(\mathbf{t})]$. La derivada en el tiempo de $E[\mathbf{v}(\mathbf{t})]$ puede ser obtenida de:

$$\frac{dE[\mathbf{v}(\mathbf{t})]}{dt} \quad \sum_{i=1}^{n} \frac{\partial E(\mathbf{v})}{\partial v_i}$$

donde,

$$v_i = \frac{\mathrm{d}}{dt} v_i.$$

Este resultado puede ser escrito compactamente como el producto escalar de los vectores:

$$\frac{dE[\mathbf{v}(\mathbf{t})]}{dt} = \nabla E^i(\mathbf{v})$$

donde $\nabla E(\mathbf{v})$ denota el vector gradiente

$$\nabla E(\mathbf{v}) = \begin{vmatrix} \frac{\partial E(\mathbf{v})}{\partial v_1} \\ \frac{\partial E(\mathbf{v})}{v2} \\ \\ \\ \frac{\partial E(\mathbf{v})}{\partial v_n} \end{vmatrix} \tag{3.3}$$

3.2.2 Función de energía

La función de energía para el sistema con pesos simétricos tiene la siguiente forma (función (le Liapunov) [44]:

$$E(\quad) = -\frac{1}{2}\mathbf{v}\ \mathbf{W}\mathbf{v} - \mathbf{i}\ \mathbf{v} + \sum_{i=1} G_i^{\,v_i} \qquad (z)dz \tag{3.4}$$

donde f_i^- es la inversa de la función de activación fi.

Aunque el tiempo ha sido omitido en la ecuación 3.4 por conveniencia en la notación, se entiende que el vector v representa $\mathbf{v}(t)$.

La variación de la función de energía con el tiempo para un W simétrico es:

$$\frac{dE}{dt} = (-\mathbf{W}\mathbf{v} - i \quad \mathbf{G}\mathbf{u})\ \dot{v}.$$

Con este resultado, la ecuación de estado 3.2 puede ser ahora escrita usando el símbolo de gradiente:

$$-\nabla E(\mathbf{v}) - \mathbf{C}\frac{d\mathbf{v}}{dt} \tag{3.5}$$

De la ecuación 3.5 puede verse que el gradiente negativo de la función de energía $E(\mathbf{v})$ es directamente proporcional a la velocidad del vector de estado u_i y a la capacitancia C_i.

3.2.3 Punto de equilibrio

El punto de equilibrio para la red de neuronas con funciones de activación continuas, corresponde a encontrar la solución v* de la siguiente ecuación:

$$\nabla E(\mathbf{v}) = 0. \tag{3.6}$$

En forma más precisa, un punto de equilibrio v* corresponde a un *punto estacionario* de energía que puede ser un máximo o un mínimo. Los puntos estacionarios de la función de energía pueden encontrarse usando las ecuaciones 3.4 y 3.6:

$$-\mathbf{W}\mathbf{v}(\mathbf{t}) \quad \mathbf{G}\mathbf{u}(\mathbf{t}) - \mathrm{I} = 0 \tag{3.7}$$

$$\mathbf{W}\mathbf{v}(\mathbf{t}) = -\mathrm{I}$$

$$v^* \quad -\mathbf{W} \quad \mathbf{I}.$$

Se ha asumido que se disponen de neuronas de alta ganancia ($\mathrm{v_i} >> u_i$), lo que elimina el último término de la ecuación 3.7.

La matriz Hessian del sistema usando neuronas de alta ganancia es:

$$\nabla\ E(\mathbf{v}) = \nabla(-\mathbf{W}\mathbf{v}(\mathbf{t}) - \mathrm{I})$$

que lleva a,

$$\nabla\ E(\mathbf{v}) = -\mathrm{W}.$$

De este resultado se desprende que la función de energía $E(\mathbf{v})$ posee un mínimo restringido, lo que significa que en el caso de limitar el espacio de salida del sistema a un hipercubo [-1, 1], el mínimo restringido debe de estar localizado en algún lugar de los límites del hipercubo, en otras palabras, en las aristas o caras del cubo.

Examinemos la ubicación de los puntos estacionarios de una red neuronal de alta ganancia. La función de energía en la ecuación 3.4 sin el tercer término, y evaluado en las caras y aristas se simplifica a las expresiones siguientes

$$E(\mathbf{v}) = -\frac{1}{2}(w_{ij}v_i v_j \quad w_{ji}v_j v_i) - k_i v_i - k_j v_j \tag{3.8}$$

$$E(\mathbf{v}) = \qquad C \tag{3.9}$$

donde k_i es constante. Los índices i y *j,* para i, $j = 1, 2, \ldots,$ n determinan la cara *(i, j)* o la arista (i), de un cubo $n-$dimensional. Dado que la matriz Hessian para la función $E(v_i, v_j)$ tiene una diagonal de ceros, no puede existir mínimos en las caras del cubo. Además, como la ecuación 3.9 es una función lineal y monotónica de una sola variable v_i, el mínimo, si es que existe, tampoco podría ocurrir en las aristas del cubo. De esta forma, de existir los mínimos, deberían estar localizados en los vértices del cubo.

liemos mostrado que si se asume el uso de neuronas de alta ganancia, se puede despreciar el último término de la función de energía en la ecuación 3.4, forzando a que los mínimos estén restringidos a estar en los vértices del cubo [-1,1] (de un total de 2n vértices).

Es de esperarse que la red de alta ganancia que se aproxima en el caso límite al sistema en tiempo discreto, tenga sus mínimos de energía en las esquinas del cubo.

En conclusión, la solución que alcance una red neuronal Hopfield depende de los parámetros de la red y de sus condiciones iniciales. Algunas de las soluciones producidas son útiles, por lo que son deseables. Otras soluciones pueden ser menos útiles y hasta erróneas, aunque algunas veces sean difíciles de evitar.

A continuación se analizará el uso de una red neuronal Hopfield para controlar la selección interna de células en un conmutador crossbar.

3.3 Determinación de las funciones de penalización

Las entradas de un conmutador crossbar N por N pueden ser representadas por una matriz binaria N por N de *demanda* $R = [r_{ij}]$. Si existe al menos un paquete en la entrada i para ser transmitido a la salida j del conmutador, entonces $r_{ij} = 1$, de otro modo $r_{ij} =$ O. El funcionamiento de conmutador crossbar en un

determinado intervalo de tiempo puede ser descrito por una matriz de *configuración* binaria C $= [c_{ij}]$. cuyos elementos indican que puntos de cruce deben de ser cerrados ($c_{ij} = 1$) o abiertos ($c_{ij} =$ O). Para un conmutador crossbar, debe de haber al menos un punto cerrado de cruce en cada columna y fila. Para maximizar el flujo del conmutador, una configuración matricial válida debe de ser encontrada de forma que su transposición con la matriz de demanda sea maximizada.

La red neuronal propuesta para controlar la entrega interna de células en un conmutador ATM crossbar es un arreglo de $N x N$ amplificadores inversores v_{ij} (neuronas), conectadas como se muestra en la Figura 3.3.

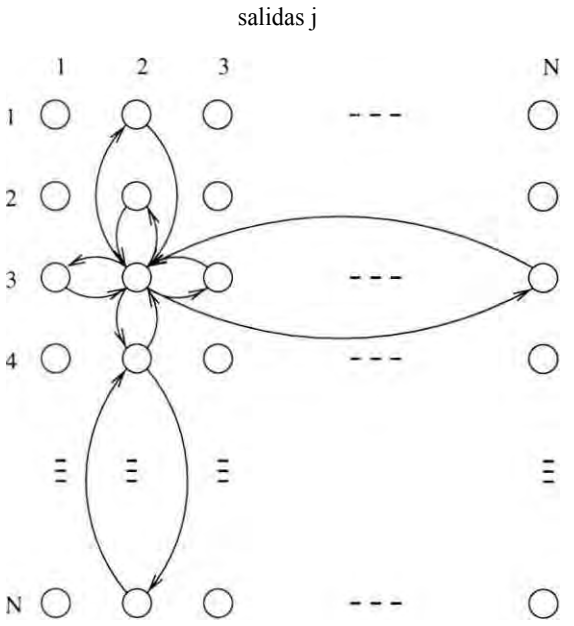


Figura 3.3: Implementación del controlador.

Cada punto de cruce es representado por una neurona simple de valores continuos. Si una neurona en particular v_{ij} está activada (es decir, su salida es cercana a 1), entonces el punto de cruce correspondiente será cerrado para permitir la transmisión de una célula desde la entrada i a la salida *j*. Similarmente, si v_{ij} está desactivada (su salida cercana a O), entonces el punto de cruce correspondiente será abierto. Para asegurar que el estado de la red neuronal represente una matriz de configuración válida para el conmutador crossbar, existen conexiones inhibitorias entre cada neurona y todas las otras neuronas en la misma fila y columna.

La siguiente función de energía de Liapunov resulta de esas conexiones inhibitorias,

$$E = E_1 + \mathbf{E2}\ \mathbf{E3}$$

$$E_1 = A\sum_x \sum \mathbf{E}\ V_{xi}V_{x3}$$

$$\mathbf{E2} = B\sum_i \sum_x \mathbf{E}_{y,x\neq y}\ V_{xi}V_{yi}$$

$$\mathbf{E3} = C(\sum\sum(V_{xi} - n)\ .$$

El primer término alcanza un mínimo cuando al menos una neurona está activada en cada fila. La potencia de la conexión entre las neuronas en la misma fila es denotada por A. Similarmente, el segundo término alcanza un mínimo cuando al menos una neurona está activada en cada columna. La potencia de la conexión entre las neuronas, en este caso en la misma columna, es denotada por B. Las funciones E_1 y **E2** fueron originalmente propuestas en [13], y aseguran que no exista más de una activación o 1 por cada fila y columna de la matriz. El término **E3** lo introducimos para asegurar de que la matriz no tenga todos sus términos desactivados o puestos en O y que globalmente existan exactamente N términos activados o puestos en 1.

Entonces, la función de energía E tiene valor mínimo de O para todas las matrices que tienen exactamente un 1 por columna y por fila. Otras configuraciones de la matriz tienen valores más altos de energía. La minimización de la función de energía E favorece una configuración adecuada para la entrega interna de células en la estructura de conmutación.

Los parámetros A y B son usualmente escogidos iguales, para dar la misma importancia a la violación de las restricciones de entrada y salida. El parámetro C se escoge dependiendo de la importancia que se quiera imponer a la condición respectiva en la función de penalización.

Por comparación con la función de Liapunov para la red Hopfield en la ecuación 3.4, la interconexión entre las neuronas de la matriz de dos dimensiones puede ser determinada como

$$= -2A\delta_{xy}(1 - \quad) - 2B\delta_{ij}(1 - \delta_{xy}) - 2\mathrm{C}. \tag{3.10}$$

Con estos resultados, para un conmutador crossbar simétrico de 2 puertos, la matriz de pesos sería la siguiente,

$$= \begin{vmatrix} 0 & -2\mathrm{A} - 2\mathrm{C} & -2\mathrm{B} - 2\mathrm{C} & 0 \\ -2\mathrm{A} - 2\mathrm{C} & 0 & 0 & -2\mathrm{B} - 2\mathrm{C} \\ -2\mathrm{B} - 2\mathrm{C} & 0 & 0 & -2\mathrm{A} - 2\mathrm{C} \\ 0 & -2B - 2\mathrm{C} & -2\mathrm{A} - 2\mathrm{C} & 0 \end{vmatrix} \tag{3.11}$$

Dado el signo negativo de los elementos de la matriz, la interconexión entre las neuronas de la misma fila (o columna) son inhibitorias.

En la construcción práctica de este esquema será necesario usar la salida invertida del amplificador que implementa la función sigmoide para evitar el problema de las conductancias negativas que aparecen en la ecuación 3.10.

Es claro que no se ha intentado maximizar la transposición entre las matrices de demanda y configuración incluyendo términos adecuados en la función de Liapunov de la red Hopfield, en vez de ello, sólo se ha forzado a la red a buscar un dominio limitado en el hiperespacio descrito por todos los posibles estados de la red neuronal.

Cualquier configuración válida (es decir, alguna con al menos una entrada no cero en cada fila y columna), corresponde a un mínimo absoluto de la función de energía propuesta. Sin embargo, es obvio que muchos de esos estados válidos de la matriz de configuración no llevarán a obtener el máximo flujo del conmutador (no maximizarán la transposición con la matriz de demanda).

Capítulo 4

Modelamiento y simulación

En este capítulo se desarrollan los modelos de simulación de los conmutadores con buffers FIFO en los puertos de entrada, buffers FIFO en los puertos de salida, y buffers independientes para cada salida en cada entrada con redes neuronales para controlar la selección interna de células. Los dos primeros modelos son validados comparando su eficiencia con resultados teóricos anteriormente reportados para esos tipos de conmutadores. El último modelo es verificado en condiciones similares de simulación que el reportado originalmente en [13] dado que no se disponen de resultados analíticos previos. Las suposiciones en el desarrollo de los modelos son también discutidas. Además, se propone un método para la generación de secuencias que siguen un proceso de Poisson. Las secuencias generadas de Poisson y auto-similares son validadas en este capítulo.

La técnica de simulación usada para resolver los modelos es la *simulación estocástica discreta de eventos* [45], donde los componentes del sistema (buffers, estructura de conmutación, etc.) son representados dentro de un programa de computadora. Los eventos que puedan ocurrir durante la operación del sistema (por ejemplo, la llegada de células al conmutador) son recreados durante la ejecución del programa.

4.1 Modelo general de los conmutadores

La Figura 4.1 muestra un diagrama de bloques que representa el proceso general de la simulación. El tráfico que se inyecta al conmutador se almacena temporalmente en buffers antes de su proceso por la estructura de conmutación. Las células que no han podido almacenarse en los buffers por encontrarse estos llenos, son descartadas. Un sistema de control o selección determina la conmutación interna para resolver los conflictos y permitir la entrega de células desde los puertos de entrada a los puertos de salida, para su transmisión final fuera del conmutador.

Para poder calcular los parámetros que permitan evaluar el rendimiento del conmutador, se guardan datos en diversos puntos en la simulación del modelo. Los valores que interesan para los cálculos son:

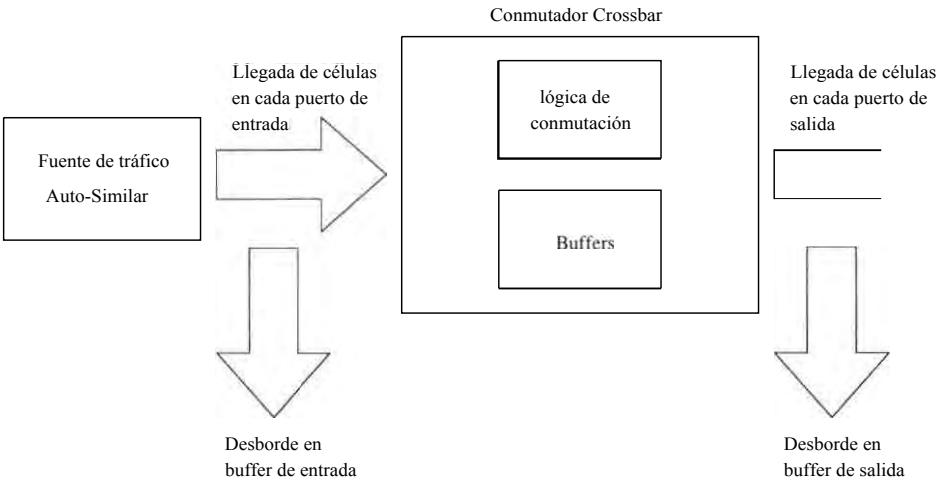


Figura 4.1: Diagrama de bloques del modelo para la simulación.

- El total de células recibidas por el modelo.
- El total de células entregadas, que son las células recibidas que han sido procesadas por el conmutador y fueron entregadas exitosamente a los puertos de salida para su transmisión.
- El total de células descartadas, que son las células recibidas y desechadas en algún punto del modelo. Dado que el sistema bajo estudio es un conmutador sin bloqueo interno, la única razón de descarte de alguna célula es por rebalse en el buffer respectivo.
- Cantidad de tiempo en que la célula está dentro del conmutador.
- Ocupación de los buffers en cada instante de tiempo.

Estos datos permitirán encontrar el flujo, la probabilidad de pérdida de células, el retardo promedio que sufren las células y la ocupación promedio de los buffers en el conmutador que se está estudiando.

4.1.1 Suposiciones en el desarrollo de los modelos

Para facilitar el desarrollo de los modelos de los conmutadores bajo las estrategias en el uso de buffers mencionadas, se hicieron las suposiciones siguientes,

1. El tiempo de simulación se divide conceptualmente en segmentos de tiempo que corresponden al tiempo requerido para la transmisión de una célula. Las células sólo pueden llegar a un puerto en el inicio de uno de estos segmentos de tiempo.

2. El tiempo de conmutación interna y transmisión de una célula de un puerto de entrada a uno de salida se realiza en un tiempo menor o igual a un segmento de tiempo.

3. Un buffer de tamaño N tiene la capacidad de almacenar hasta N células (de 53 bytes cada una). Las células que no encuentren buffers disponibles son descartadas.

4. El tráfico en cada entrada del conmutador es independiente de las otras entradas. Cada entrada tiene una fuente diferente de generación de tráfico (diferente semilla para la generación de números pseudo-aleatorios).

5. La fuente de tráfico en cada puerto se asume como asintóticamente auto-similar.

6. El número de entradas y salidas es el mismo, y de la misma capacidad. Para la simulación los puertos de entrada y salida son de 155 Mbps.

7. El destino de cada célula es escogido en forma pseudo-aleatoria con distribución de probabilidad uniforme.

8. Para resolver los conflictos internos, en los modelos generales de buffers en la entrada o salida, se ha escogido un esquema de prioridad rotativa (o *round robin*).

4.2 Tráfico de entrada

Los modelos van a ser estudiados excitándolos con secuencias de datos que forman un proceso auto-similar y que representan el tráfico de entrada. También se usarán procesos de Poisson para representar el tráfico de entrada para poder comparar y estudiar los resultados.

4.2.1 Procesos de Poisson

El proceso de llegada de Poisson es ampliamente utilizado en el diseño de sistemas de colas. Su uso ha sido muy difundido para el estudio del tráfico en redes telefónicas, así como en la evaluación del rendimiento de los sistemas de conmutación en general.

Son tres los enunciados básicos que definen un proceso de Poisson [1]. Considérese un pequeño intervalo de tiempo Δt con Δt O, entonces:

1. La probabilidad de una llegada en el intervalo Δt se define como $\lambda\Delta t \quad 0(\Delta t)$, $\lambda\Delta t << 1$, siendo una constante de proporcionalidad especificada, y $0(\Delta t)$ implica que otros términos son de mayor orden en Δt y que se aproximan a cero más rápidamente que Δt conforme $\Delta t \to 0$.

2. La probabilidad de cero llegadas en Δt es $1 - \lambda\Delta t \quad 0(\Delta t)$.

3. Las llegadas son procesos sin memoria: cada llegada (evento) en un intervalo de tiempo es independiente de eventos en intervalos previos o futuros.

Si se toma un intervalo finito T mayor, se encuentra la probabilidad $p(k)$ de k llegadas en T *(distribución de* $Poisson$) dado por:

$$p(k) = (\lambda T)^{k} \cdot \frac{}{k!}$$

donde $k = 0, 1, 2, .$

Para este proceso, el primer y segundo momento estadístico están dados por,

$$E(k) = \sum_{k=0}^{\infty} kp(k) = \lambda T \tag{4.1}$$

$$\sigma_{k} = E(k) = \lambda T. \tag{4.2}$$

Ahora considérese un intervalo de tiempo grande en donde ocurre un proceso de Poisson. El tiempo de llegada entre eventos se representa por el símbolo T. Es evidente que T es una variable aleatoria positiva con distribución continua. En la estadística de Poisson, T es una variable aleatoria con *distribución exponencial;* es decir, su función de densidad de probabilidad $f_{\tau}(\tau)$ está dada por,

$$f_{\tau}(\tau) = \mathrm{A} \quad {}^{\lambda\tau} \qquad - \geq 0.$$

En procesos de llegada de Poisson, el tiempo entre llegadas es más bien pequeño, y la probabilidad entre dos eventos (o llegadas) sucesivos disminuye en forma exponencial con el tiempo.

La media de la distribución exponencial es,

$$E(\tau) = \int_{0} \quad T f_{,}(\tau) d\tau = {}^{1}$$

y la varianza es,

$$\sigma_{\tau} = \frac{1}{\lambda^{2}}$$

Generación de secuencias de Poisson

Podemos obtener una secuencia que corresponda aproximadamente a un proceso de Poisson a partir de una secuencia de números pseudo-aleatorios con distribución de probabilidad uniforme [1] usando el *método de la transformación inversa* [42].

Usando la función acumulativa de distribución de probabilidad para una variable aleatoria exponencial k con media $1/\lambda$,

$$F_k(k) = 1 - e^{\cdot\cdot} = \text{n}$$

tenemos que,

$$x = --/\text{n}(1 - \text{n}). \tag{4.3}$$

Usando la ecuación 4.3 podemos generar valores para una variable aleatoria exponencial de media $1/\lambda$ a partir de números pseudo-aleatorios uniformemente distribuidos en el intervalo [0, 1). Luego dividiendo el tiempo en segmentos fijos de longitud $1/\lambda$ y contando el número de eventos en cada segmento de tiempo podemos generar una nueva secuencia con comportamiento similar a un proceso Poisson con media 1/A.

Verificación de la secuencia generada

Para estudiar el efecto del tráfico Poisson con respecto al tráfico auto-similar en el rendimiento de un conmutador, vamos a inyectar a los modelos bajo estudio ambos tipos de tráfico.

Un método para verificar si una secuencia dada corresponde a un proceso de Poisson es la *prueba* $\mathbf{x}^2$ *de la bondad de ajuste* [46]. Está basado en una comparación entre los datos observados y los teóricos, donde los datos teóricos se obtienen a partir de la distribución teórica que se prueba.

La prueba de la bondad de ajuste compara la frecuencia observada $f_{..}$ con la frecuencia esperada que se generaría si se supone la distribución de Poisson.

En la Figura 4.2 se muestra (con puntos) el histograma de una secuencia de Poisson generada usando el método de la transformada inversa buscando una media de 5, y el histograma teórico (con líneas) de un proceso de Poisson de media exactamente 5.

En la tabla siguiente están los valores de frecuencia encontrados.

[1] Usando la función *rand* de la librería estándar *stdlib* del lenguaje de programación C.

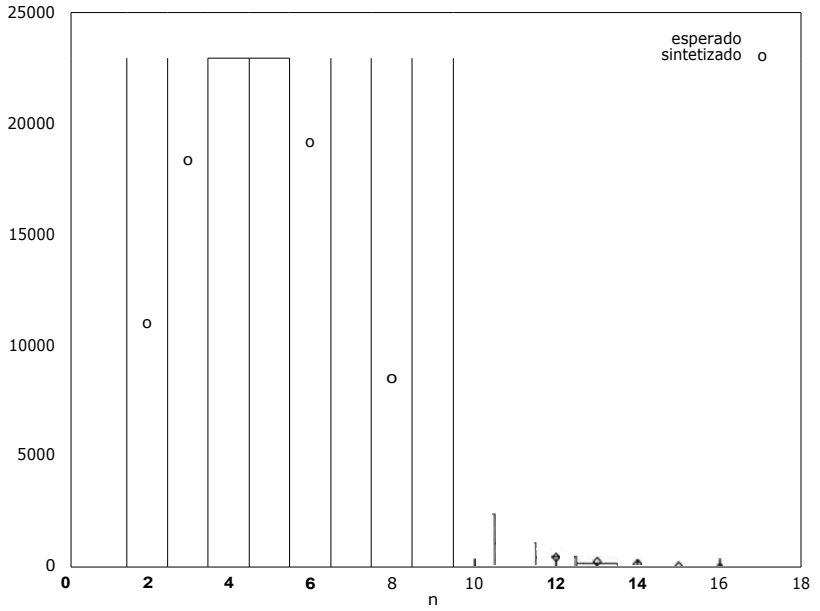


Figura 4.2: Histograma de una secuencia teórica esperada de Poisson y de una secuencia sintetizada de Poisson.

n	f_n	n	f_n	n	f_n	n	f_n	n	f_n
0	843	4	23023	8	8458	12	430	16	17
1	4489	5	22990	9	4741	13	247	17	9
2	10961	6	19147	10	2456	14	158	18	3
3	18341	7	13885	11	1061	15	60	>18	0

Para determinar la frecuencia estimada, calculamos la media de la muestra generada con media esperada de 5:

$$= \frac{\sum_{n}^{18} \cup n f_n}{\sum_{n=}^{18} 0 f_n} = 4.9996.$$

El paso siguiente consiste en determinar las probabilidades p_n para una distribución teórica de Poisson con media 4.9996. Obsérvese que,

$$p_n = \frac{4.9996^n e^{-4.9996}}{77..\quad \cdot}$$

n	f_n	n	f_n	n	f_n	n	f_n	n	f_n
0	0.0067	4	0.1754	8	0.0652	12	0.0034	16	4.9096
1	0.0337	5	0.1754	9	0.0362	13	0.0013	17	1.4438
2	0.0842	6	0.1462	10	0.0181	14	0.0004	18	4.0104
3	0.1403	7	0.1044	11	0.0082	15	0.0001	>18	0

Como tenemos 131072 observaciones, la frecuencia esperada se puede obtener como:

$$e_n = 131072 p_n.$$

El método recomienda usar valores de e_n mayores a la mitad del valor máximo, por lo tanto, será necesario combinar los valores extremos hasta lograr tal condición.

n	f_n	en	X^2
0-2	16293	16343	1.8577
3	18341	18402	1.1667
4	23023	23001	0.5962
5	22990	22999	0.2028
6	19147	19164	0.0215
7	13885	13688	0.0035
8-18	17640	17476	0.0155
Total			3.864

Ahora comparemos el valor obtenido de $\chi^{\hat{}} = 3.864$ con el valor crítico de la distribución x^2. Para esto, necesitamos especificar un nivel de significancia a, y los grados de libertad v. El valor de ν para la prueba de la bondad de ajuste está dado por y = número de intervalos de clase - número de parámetros estimados - 1, lo que nos da $= -1-1=5$.

Mediante el uso de un nivel de significancia a $= 0.05$, las tablas x^2 producen un valor crítico de $\chi^2_5 = 11.1$.

La aplicación de la prueba x^2 recomienda que se acepte la hipótesis en el nivel de significancia especificado a si el valor $x^2 <$ Como esta condición se cumple, concluimos que la secuencia generada cumple con una distribución de Poisson con media 5.

4.2.2 Procesos auto-similares

El tráfico en cada puerto de entrada modelado como un proceso auto-similar, tiene el parámetro de Hurst ajustado al valor típico de $\boldsymbol{H} = 0.79$, valor encontrado en redes de datos bajo condiciones normales de trabajo, y se genera según el método de Ruido Gaussiano Fraccional con la transformada rápida de Fourier propuesto en [31].

Dado que este método de generación sólo produce un proceso asintóticamente auto-similar, se requiere de un cierto tiempo de computación para lograr un determinado grado de auto-similaridad. Este efecto es ilustrado en la Figura 4.3, donde se grafica el promedio de los datos versus el número de muestras.

La fuente de tráfico auto-similar en cada puerto de entrada inyecta al modelo un total de 131072 valores, que corresponden al número de células en cada segmento de observación. Esto es equivalente a inyectar 1310720 segmentos de tiempo por

puerto de entrada, dado que se debe *esparcir* el número de células de cada intervalo de tiempo en n llegadas independientes, y por la capacidad del canal, pueden existir como máximo 10 células por intervalo de tiempo.

El número de segmentos de medida es un valor lo suficientemente grande como para alcanzar un estado estable en el proceso auto-similar.

Entonces la probabilidad de pérdida de células puede calcularse hasta valores por encima de 10^{-6}, siendo los cálculos inferiores a 10^{-5} no precisos estadísticamente.

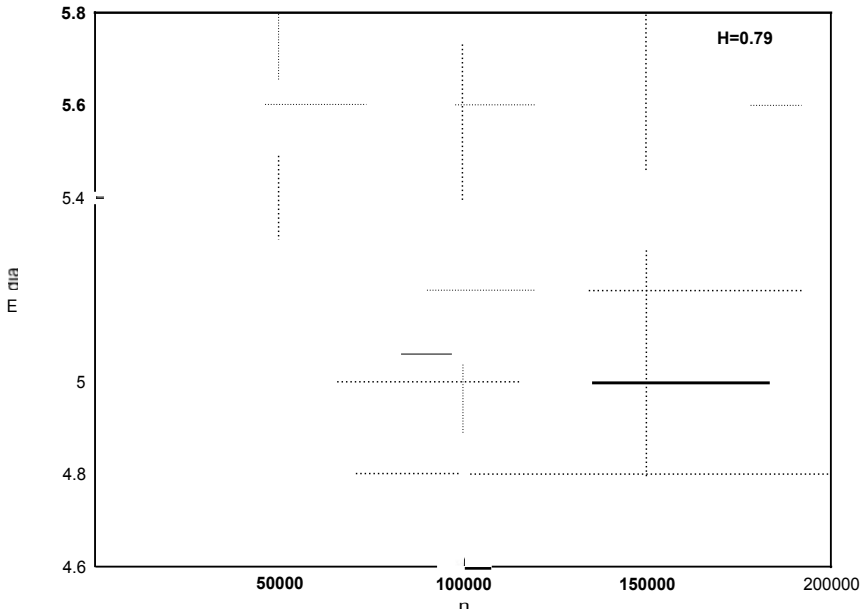


Figura 4.3: Promediador móvil para el tráfico auto-similar.

Puede modificarse la carga o uso de la red modificando la media del proceso auto-similar de entrada. Manteniendo la varianza igual a la media, tendremos similares condiciones en cantidad de células que ingresan al modelo, que en el caso de usar modelos de Poisson.

Verificación de la auto-similaridad de la secuencia generada

La Figura 4.4 muestra un conjunto de gráficos que ilustran el número de paquetes por unidad de tiempo para cuatro diferentes escalas de tiempo. Cada figura grafica aproximadamente 500 observaciones del proceso auto-similar sintéticamente generado por el método FFT-FGN descrito anteriormente.

En el primer gráfico se usó un segmento de observación de 0.0005s, y cada subsecuente gráfico es obtenido incrementando la resolución del tiempo en un factor de 10. Como se describe en [4], se observa que el perfil o distribución del tráfico es *similar* en varias escalas de tiempo.

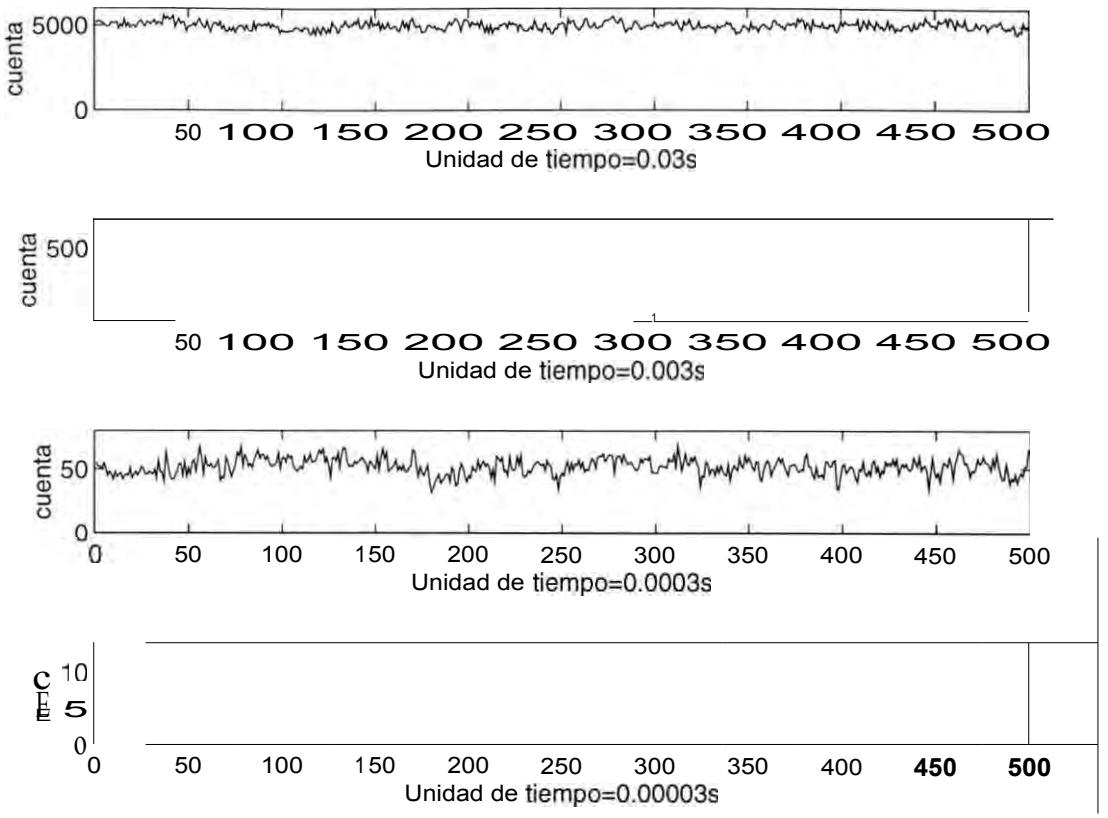


Figura 4.4: auto-similaridad visual en la secuencia generada de un proceso FFT-FGN.

La Figura 4.5 muestra una secuencia de tráfico generado como un proceso de Poisson en las mismas escalas de tiempo que el caso anterior. Como puede apreciarse, el proceso genera un perfil muy parecido al que produce el ruido blanco, y tiende a "suavizarse" cuando se incrementa el tamaño de la unidad de observación.

Entonces, la Figura 4.4 ilustra visualmente la auto-similaridad del proceso generado por el método de la FFT-FGN mostrando que a diversas escalas de tiempo el perfil del tráfico es similar.

Sin embargo, es posible que un proceso exhiba visualmente auto-similaridad en varias escalas de tiempo, pero que no presente dependencia de amplio rango. La auto-similaridad visual podría atribuirse a alguna característica no estacionaria en el proceso. La demostración de auto-similaridad visual presentada puede ser vista como un paso inicial en el establecimiento de la dependencia de amplio rango del proceso bajo estudio.

Los métodos heurísticos presentados anteriormente para determinar la auto-similaridad de un proceso serán usados enseguida para estimar el parámetro de Hurst de un proceso generado con el método de la FFT-FGN, y confirmar que corresponde a un proceso asintóticamente auto-similar.

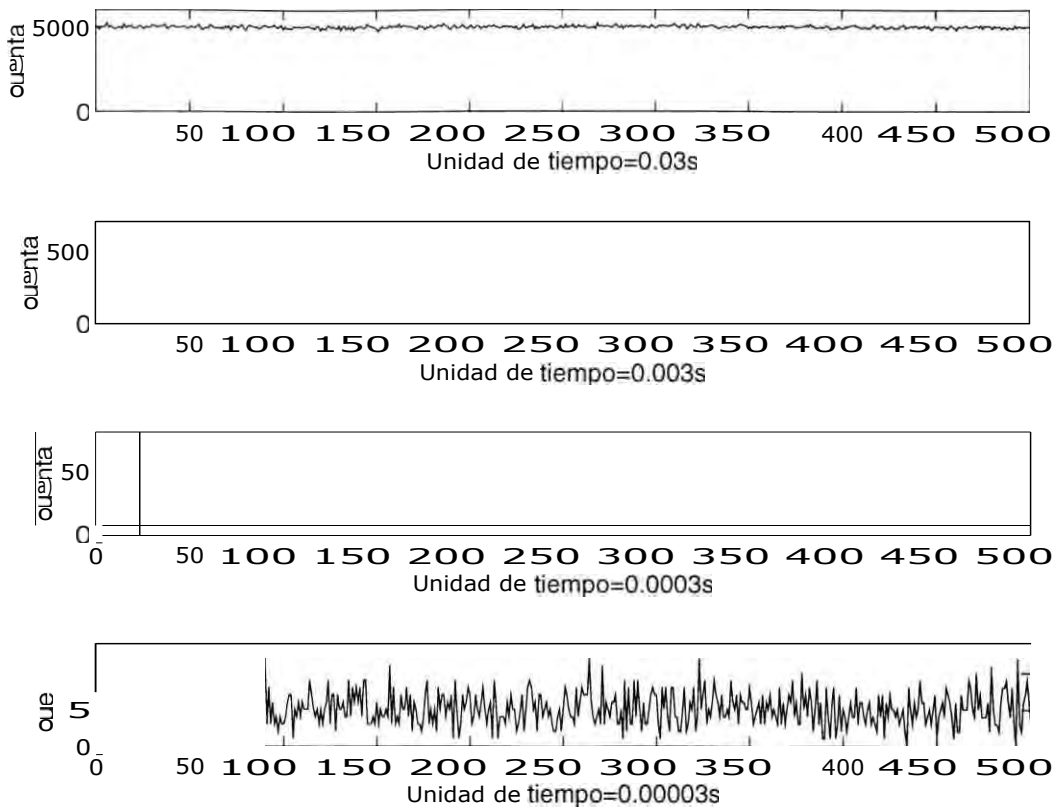


Figura 4.5: Ausencia de auto-similaridad visual en la secuencia generada de un proceso de Poisson.

En la Figura 4.3 se muestra la gráfica de la varianza en el tiempo para un conjunto de datos generados con el método de la FFT-FGN. Los extremos de la curva son ignorados: los puntos iniciales corresponden a pequeños niveles de agregación, los finales no forman un grupo lo suficientemente grande para calcular correctamente la varianza. La pendiente de la curva para este conjunto de datos es aproximadamente -0.42. Entonces de la ecuación 2.1, D, que representa la pendiente negativa de la línea que mejor se ajusta a la curva graficada, es 0.42. Con lo que el parámetro de Hurst calculado con el método de la varianza en el tiempo, es $H = 1\text{-}0.42/2 = 0.79$

El diagrama Pox, usando el análisis R/S para detectar dependencia de amplio rango para el mismo conjunto de datos se muestra en la Figura 4.7. La pendiente de la línea que mejor sigue a la curva determina el parámetro de Hurst, y corresponde a $H = 0.79$.

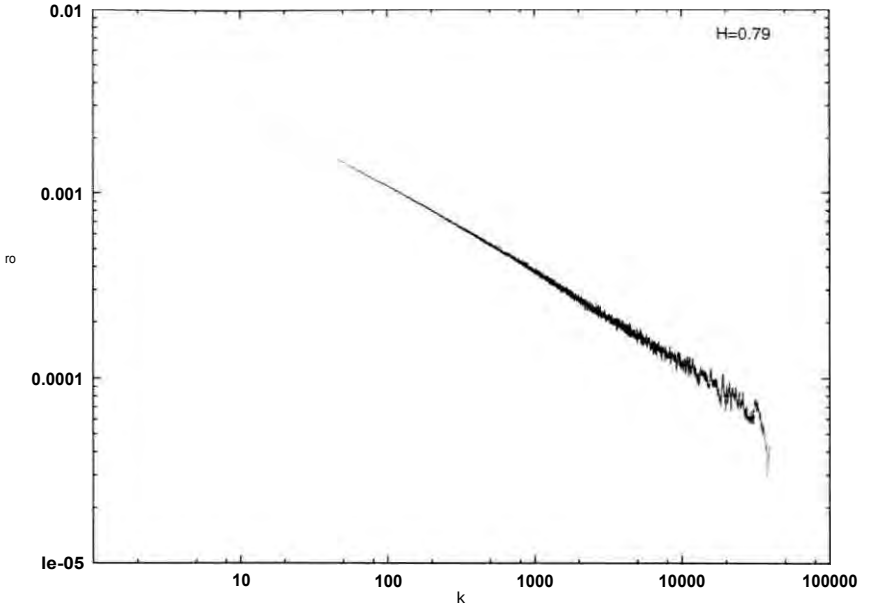


Figura 4.6: Gráfica de la varianza en el tiempo para el proceso generado con el método de la FFT-FGN.

4.3 Desarrollo del modelo de simulación de los conmutadores

4.3.1 Modelo de simulación de los conmutadores

La operación de los modelos de simulación para los conmutadores con buffers en los puertos de entrada, salida, y separados para cada salida en los puertos de entrada controlados por una red neuronal, son ilustrados en los diagramas de flujo de las Figuras 4.10, 4.11 y 4.12.

Para los tres modelos se generan secuencias de datos que representan las células que llegan por unidad de tiempo a un puerto del sistema siguiendo un proceso auto-similar o Poisson según el caso estudiado. Estas células son distribuidas en el tiempo asignándoles una dirección de un puerto de salida en forma pseudo-aleatoria con distribución de probabilidad uniforme y almacenadas en un archivo. El programa que simula el conmutador lee los archivos de datos de tráfico y durante su ejecución los procesa guardando los datos de interés que son: el número de células que ingresan al conmutador, el número de células descartadas, el tiempo en que cada célula permanece en los buffers, el número de células entregadas y el tiempo total de simulación, para poder calcular los parámetros de la calidad de servicio anteriormente mencionados. A continuación se discute el funcionamiento de cada modelo de conmutación.

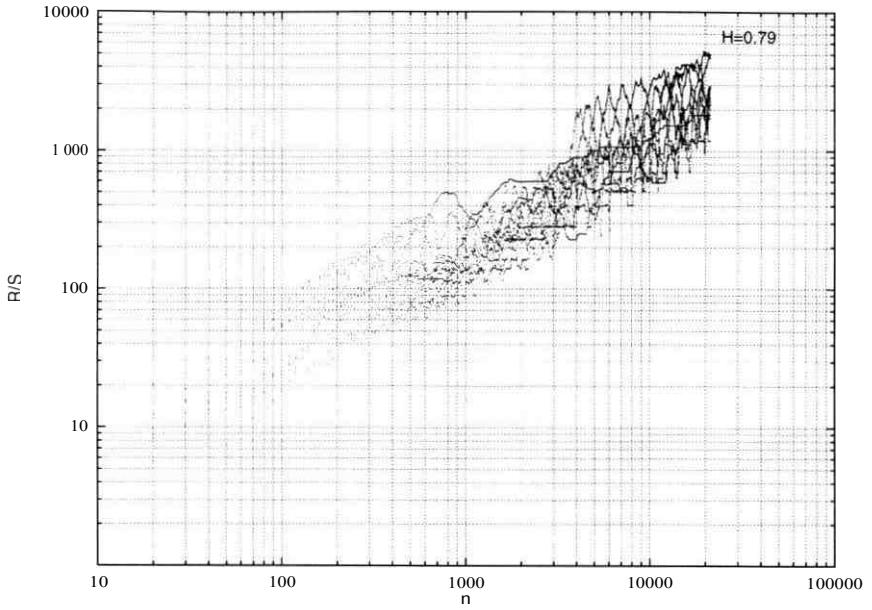


Figura 4.7: Gráfica R/S del proceso generado con el método de la FFT-FGN.

Modelo de simulación del conmutador crossbar con buffers en los puertos de entrada

Este modelo mantiene un buffer independiente por cada puerto de entrada. Una célula que llega al conmutador, entra momentáneamente a un buffer en el puerto de entrada correspondiente, donde la célula espera su acceso a la estructura de conmutación interna. Si el buffer se encuentra totalmente ocupado, se descarta la célula.

Los buffers se asumen como FIFO, de forma que, al principio de cada intervalo de tiempo, sólo la primera célula de cada buffer FIFO, puede competir por los puertos de salida del conmutador. Si cada célula está direccionada a un puerto distinto de salida, la estructura de conmutación sin bloqueo interno permite pasar todas las primeras células de cada buffer FIFO de entrada a las salidas correspondientes. Si k células en el inicio de los buffers FIFO de entrada están direccionadas a una misma salida, sólo una es permitida de pasar a través de la estructura de conmutación interna, mientras que las $k-1$ células deben de esperar hasta el siguiente intervalo de tiempo para una nueva selección entre las células que aguardan. Mientras una célula esté esperando acceso a una salida, las otras células estarán retenidas en el buffer, y consecuentemente bloqueadas de alcanzar posibles salidas libres en el conmutador (bloqueo HOL). Cuando se entrega una célula a su puerto de salida de destino, se libera la posición HOL de la cola FIFO y esta es ocupada por la siguiente célula en espera si es que existe alguna. El proceso continua hasta agotar

los archivos de tráfico. Luego de cada bucle, la variable de tiempo se incrementa en el sistema.

Modelo de simulación del conmutador crossbar con buffers en los puertos de salida

El funcionamiento de este modelo de conmutación es similar al anterior, con la diferencia de que los buffers **FIFO** de tamaño b se ubican en los puertos de salida del sistema y no en los puertos de entrada. Además, internamente esta arquitectura de conmutación debe de funcionar proporcionalmente N veces más rápido que los puertos de entrada y salida, de forma que si k células llegan en un intervalo de tiempo en diferentes puertos de entrada direccionadas a la misma salida, todas las k células puedan ser entregadas al buffer del puerto de salida correspondiente dentro del mismo intervalo de tiempo. Como sólo una célula puede ser transmitida de cada buffer por puerto de salida, las $(k-1)$ células restantes deben de esperar en la cola para su transmisión en los próximos intervalos de tiempo. Cuando se transmite un célula ocurre el correspondiente movimiento de células (si existen) en el buffer de salida. El conmutador opera en forma síncrona, de forma que las células son recibidas y servidas en intervalos fijos de tiempo. La operación del conmutador es sincronizada con los intervalos de tiempo. El proceso continua hasta agotar los archivos de tráfico. Luego de cada bucle, la variable de tiempo se incrementa en el sistema.

Modelo de simulación del conmutador crossbar controlado por redes neuronales

En este modelo ubicamos buffers FIFO independientes para cada puerto de salida en cada puerto de entrada y construimos un controlador interno basado en una red neuronal Hopfield que dependiendo de la ocupación de los buffers determina la mejor conmutación interna.

La primera parte del programa de simulación calcula el valor de la matriz de pesos usando la ecuación 3.10. Las constantes de los pesos se han hecho iguales a A=3, B=3 y C=2. Estos valores han sido ajustados junto con el valor de la ganancia de la función de activación para asegurar una adecuada y rápida convergencia a un mínimo de la red neuronal. Experimentalmente se encontró recomendable que la ganancia de la función de activación tenga un valor inferior al valor de los pesos para incrementar la velocidad de convergencia de la red.

El programa lee los archivos de tráfico, y almacena las células que llegan en los buffers correspondientes o las descarta si no encuentra espacio disponible en los buffers. La ocupación de uno de estos buffers por una o más células se refleja en una matriz de demanda, cuyos elementos reflejan si el buffer correspondiente se encuentra ocupado (en cuyo caso 2v señalan este estado), o vacío (indicado con $0v$). Esta matriz de demanda corresponde a la matriz de entrada $N x N$ de la red neuronal Hopfield que controla la conmutación interna.

La red neuronal Hopfield evolucionará en función a la matriz de entrada y a la matriz de pesos previamente calculada, bajo la regla siguiente:

$$v_i^{k+} = f(W v \quad u_i) \tag{4.4}$$

para Vi = 1, 2, . . . , n, y $k = 0, 1, \ldots,$ donde el superíndice k denota el número de veces que se va haciendo la recursión.

El esquema de actualización es asíncrono, dado que sólo opera para un valor de i a la vez. Este algoritmo es conocido como la *recursión estocástica asíncrona* de una red neuronal Hopfield, y en forma real representa retardos de propagación aleatorios y factores aleatorios como el ruido y el *jitter* presentes en las redes reales. Tales fenómenos podrían ocurrir en una red neuronal real usando funciones de activación de alta ganancia cercanas a $sgn(\cdot)$.

Podemos aproximar el funcionamiento de una red Hopfield en tiempo continuo a una red en tiempo discreto, usando una función de activación con ganancia muy alta. Esta función se muestra en la Figura 4.8.

El signo negativo en la matriz de pesos, provoca que las conductancias sean negativas. En la práctica, este problema se soluciona usando una función sigmoide invertida.

En la red neuronal Hopfield propuesta, es posible alcanzar estados de equilibrio no deseados desde el punto de vista de la conmutación interna óptima. Para ilustrar este caso y en general para mostrar como evoluciona una red neuronal Hopfield hasta sus puntos de equilibrio tomamos como ejemplo una red diseñada para controlar un conmutador simétrico de dos puertos con los pesos calculados usando la ecuación 3.10.

$$= \begin{vmatrix} 0 & -10 & -10 & 0 \\ -10 & 0 & 0 & -10 \\ -10 & 0 & 0 & -10 \\ 0 & -10 & -10 & 0 \end{vmatrix} \tag{4.5}$$

El estado de la red está dado en cualquier momento por un vector cuyos elementos son las entradas (o salidas) de las neuronas ($a = v_{00}$, $b = v_{01}$, $c = v_{10}$, $d = v_{11}$) agrupados en la matriz:

$$V = \begin{matrix} & b \\ c & d \end{matrix} \tag{4.6}$$

Dado que la red evoluciona asincrónicamente, sólo uno de los elementos de la matriz de salida puede cambiar a la vez. Esto provoca que la red se encuentre ya sea en el mismo estado en que comenzó, o en un nuevo estado que es una distancia Hamming del anterior. Como son cuatro los elementos de la matriz en este caso, habrían tres posibles nuevos estados por cada cambio en un elemento de la matriz. La Figura 4.9 muestra todos los posibles estados iniciales, y las rutas posibles que sigue la red hasta la estabilización del sistema en los puntos de menor energía. Los estados están definidos en la tabla siguiente, donde aparecen con "0" lógico los elementos de conmutación que deben ser abiertos, y con "1" lógico los que deben cerrarse. También se muestran los niveles de energía asociados en cada estado (aquí tenemos un valor de $a = 2$ dado que el "1" lógico corresponde a 2v).

Etiqueta	Estado				nivel de energía
	a	b	c	d	
A	0	0	0	0	4C a
B	1	0	0	0	Ca
C	0	1	0	0	Ca
D	0	0	0	1	Ca
E	0	0	1	0	Ca
F	0	0	1	1	2A α
G	1	0	1	0	2B α
H	1	1	0	0	2A α
I	0	1	0	1	2B α
J	0	1	1	1	(2A+2B+C) a
K	1	0	1	1	(2A+2B+C) a
L	1	1	1	0	(2A+2B+C) a
M	1	1	0	1	(2A+2B+C) a
N	1	0	0	1	0
O	0	1	1	0	0
P	1	1	1	1	(4A+4B+4C) a

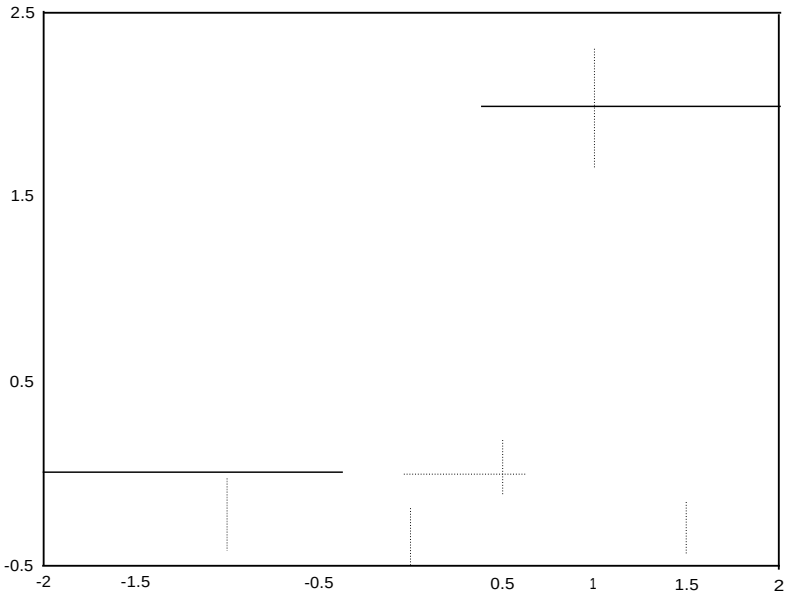


Figura 4.8: Función de activación sigmoide.

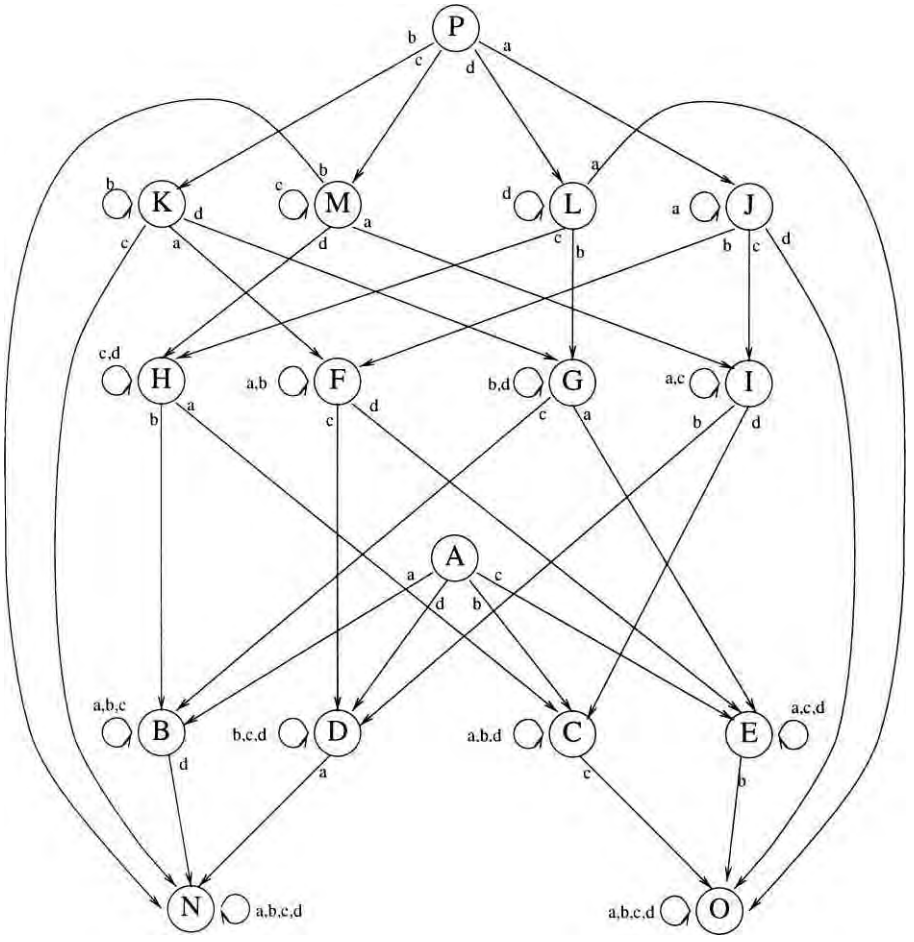


Figura 4.9: Transiciones de la red neuronal hasta sus puntos de equilibrio.

Los puntos N y O son los puntos de menor energía de la red para constantes con valores positivos, como puede notarse en la tabla anterior, y son los puntos de equilibrio a los que se puede llegar desde cualquier configuración inicial de la red, dependiendo de la secuencia de evolución de los componentes de la red.

En la Figura 4.9 puede notarse con claridad que, para determinados casos se puede llegar a un punto de equilibrio que corresponde a un estado de conmutación no deseado. Por ejemplo, si partimos del nodo K, la siguiente fase de la red nos llevará a F o G, y de allí a D o E y posteriormente a los puntos de equilibrio N u O, dependiendo del orden de actualización de los elementos de la matriz. Para obtener la conmutación interna óptima, si partimos del nodo K, deberíamos llegar al nodo N. Sin embargo, como se puede observar, también existe la posibilidad de alcanzar O bajo ciertas circunstancias, lo que no provocaría la conmutación apropiada para obtener un flujo óptimo.

En el modelo, el orden de actualización de la matriz es pseudo-aleatorio, y pueden requerirse hasta N ciclos para llegar a los puntos de equilibrio. Una vez obtenida una matriz de salida estable, esta se usa como matriz de configuración para activar la estructura de conmutación interna. Los valores iguales o superiores a 1v ("1" lógico) en la matriz de salida indican que se debe de cerrar el elemento de conmutación correspondiente, valores inferiores a 1v ("0" lógico) indican que el elemento de conmutación debe de abrirse.

Los buffers seleccionados liberan la primera célula, que es entregada al puerto de salida correspondiente para su transmisión. Como en los casos anteriores, el proceso continua hasta agotar los archivos de tráfico. Luego de cada bucle, la variable de tiempo se incrementa en el sistema.

4.3.2 Verificación de los modelos

Para verificar los modelos, usaremos la difundida técnica de comparación de los resultados obtenidos con los resultados teóricos.

El análisis teórico de los modelos de conmutación de paquetes de alta velocidad estudiados es difícil dado que la caracterización de sus espacios de estado es una tarea compleja. Como cada paquete transporta una dirección de destino, la caracterización completa de cada estado debería incluir la dirección de destino de los paquetes en las colas. Para el análisis teórico, las direcciones destino de todos los paquetes, excepto los paquetes HOL, se asumen como independientes. Las direcciones de las células HOL que han sido bloqueadas anteriormente, no pueden asumirse como

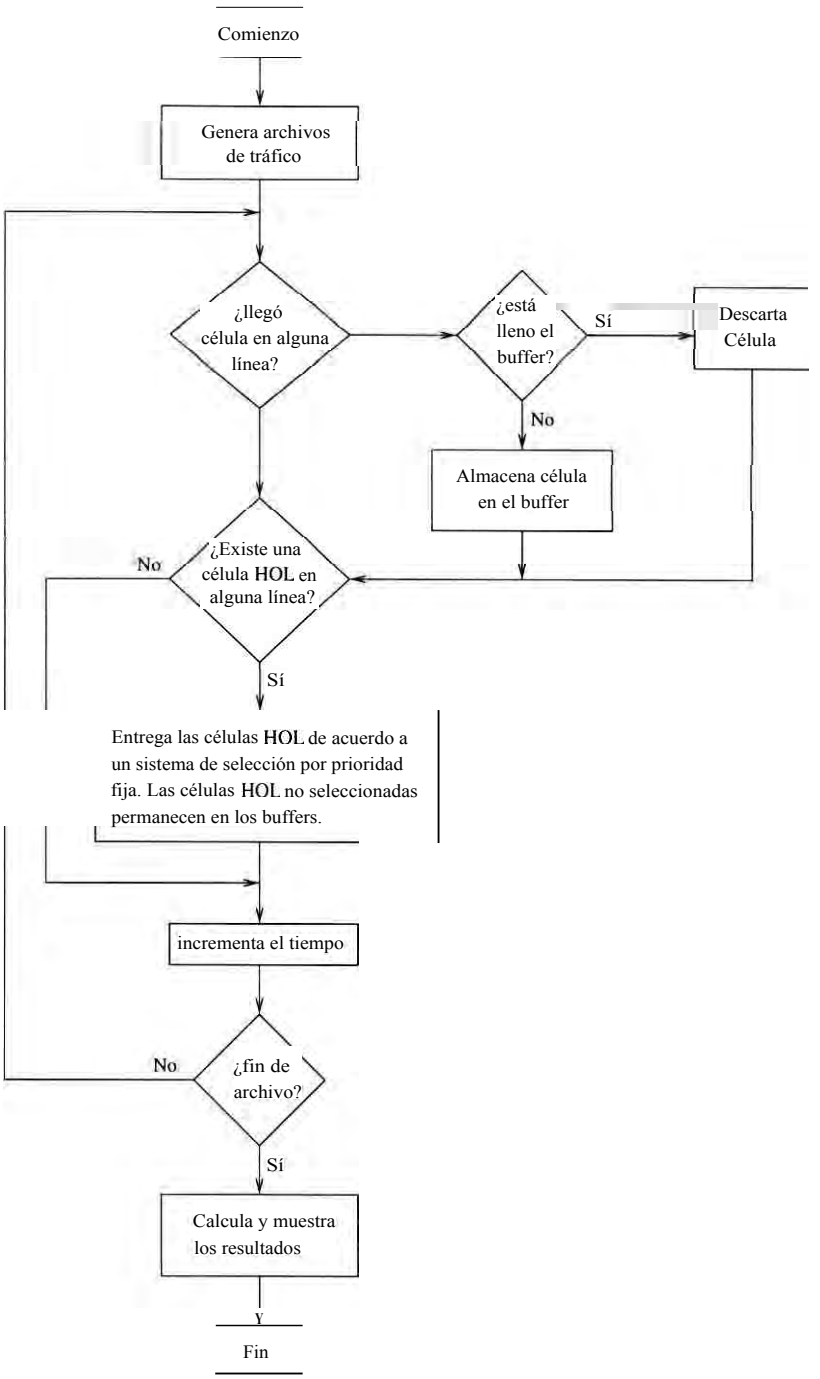


Figura. 4.10: Diagrama de flujo del modelo del conmutador con buffers en las entradas

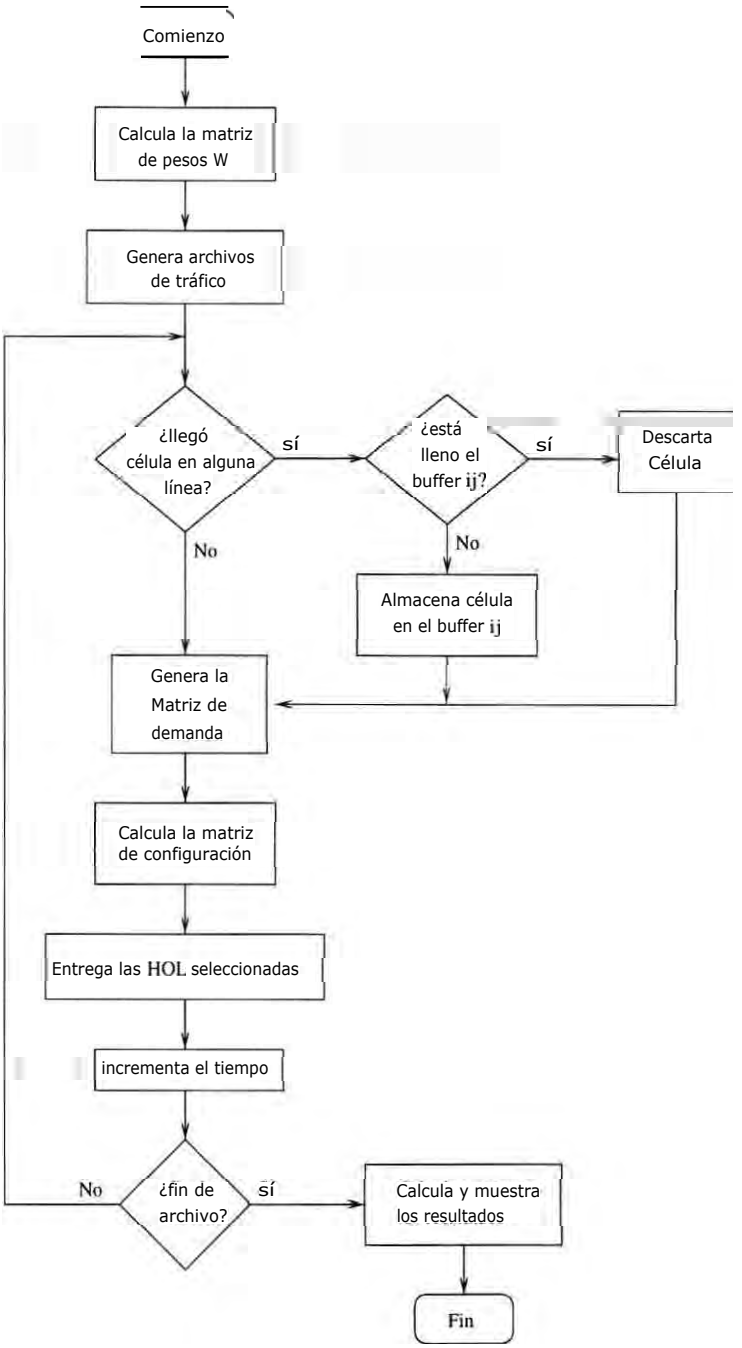


Figura 4.12: Diagrama de flujo del modelo del conmutador controlado por una red neuronal Hopfield

independientes. Además, existen dos tipos de paquetes HOL en una cola de entrada justo antes de la fase de selección: los paquetes HOL que perdieron en la selección anterior, y los paquetes HOL "frescos", que han sido recientemente movidos al HOL. Lo segundo también podría representar a nuevas llegadas en el intervalo de tiempo previo. Los paquetes HOL recientemente llegados tienen direcciones de destino independientes, mientras que las direcciones de los paquetes HOL bloqueados no son independientes dado que ha habido una selección previa.

En [8] se estudia el máximo flujo alcanzable en conmutadores con buffers FIFO en los puertos de entrada para conmutadores con diferentes números de puertos. El análisis asume un proceso de llegada Bernoulli en todos los puertos de entrada. Una célula llega independientemente en cada intervalo de tiempo con probabilidad *p,* con igual probabilidad de ser destinada a cualquier salida. Para valores pequeños de *N* se puede realizar un análisis de cadenas de Markov para determinar el flujo del sistema. La saturación del flujo en un conmutador 4 x 4 ha sido determinado analíticamente en 0.6553. Dado que las células tienen igual probabilidad de estar destinadas a las líneas de salida, entonces el flujo obtenido será el mismo para cualquier puerto de salida con igual probabilidad.

La Figura 4.13 muestra los resultados de la simulación del modelo construido del conmutador crossbar con buffers FIFO en los puertos de entrada para varios tamaños de buffers usando procesos de Poisson para modelar el tráfico de entrada. El valor de saturación encontrado es de 0.6557, valor muy próximo al encontrado analíticamente en [8] y similar al encontrado por [34] también con simulación.

En [8] también se presenta un estudio analítico del modelo con buffers FIFO de tamaño *b* ubicados en los puertos de salida del conmutador.

Definiendo una variable aleatoria A, como el número de llegadas de células destinadas a un puerto en particular en un determinado intervalo de tiempo. Tenemos que

$$a_k = P(A = k) = \binom{N}{k} \left(\frac{p}{N}\right) \qquad p \qquad ;k = 0,1,\ldots,N$$

donde N es el número de líneas de salida, y p es la probabilidad de una llegada en un intervalo de tiempo. Para N y b finitos, se define una cadena de Markov en estado finito y tiempo discreto, y se obtiene una cola de tamaño fijo directamente del balance de ecuaciones de la cadena de Markov. Se descarta una célula si, saliendo de la estructura de conmutación, se encuentra que la cola de salida contiene *b* células. La probabilidad de pérdida de células está dada por:

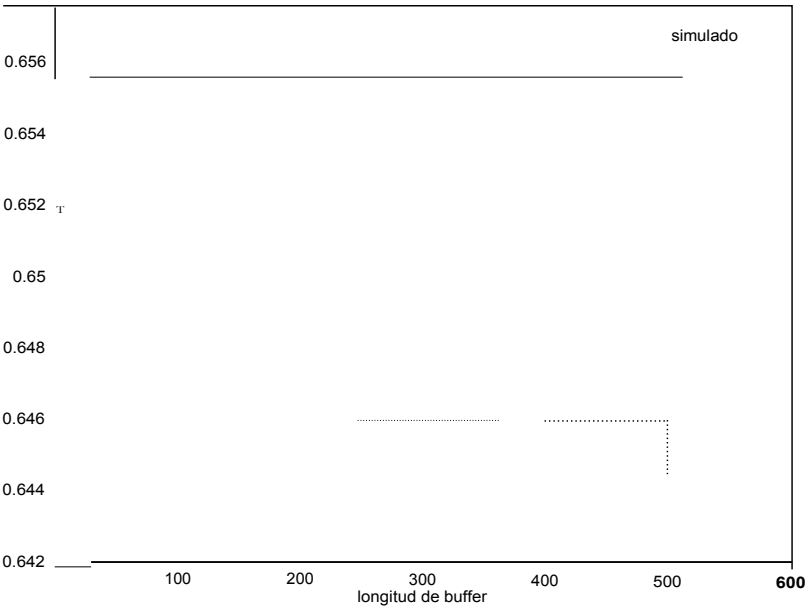


Figura 4.13: Saturación del flujo en un conmutador crossbar 4 x 4 con buffers en las entradas

$$P_{loss} = \frac{(1 - q_0 a_0)}{p}$$

donde

$$q_0 = \frac{1}{1 + \sum_{n=1} q_0}$$

$$q = \frac{(a_0 - a_1)}{a_0} q_0$$

$$q_n = \frac{(1 - a_1}{a_0} \quad \frac{a_k}{k=2\ a0} \quad ; 2 < n < b.$$

La Figura 4.14 muestra los resultados de la simulación de la probabilidad de pérdida de células para varios tamaños de buffers en un conmutador 4 x 4 con buffers en las salidas usando procesos de Poisson con media 9 para modelar el tráfico de entrada. Los valores encontrados en la simulación se acercan a los encontrados analíticamente en [8].

El modelo del conmutador ATM crossbar controlado con redes neuronales propuesto es una variación del sistema de conmutación de paquetes presentado en [13], usando una función de penalización mejorada para incrementar el flujo interno del conmutador.

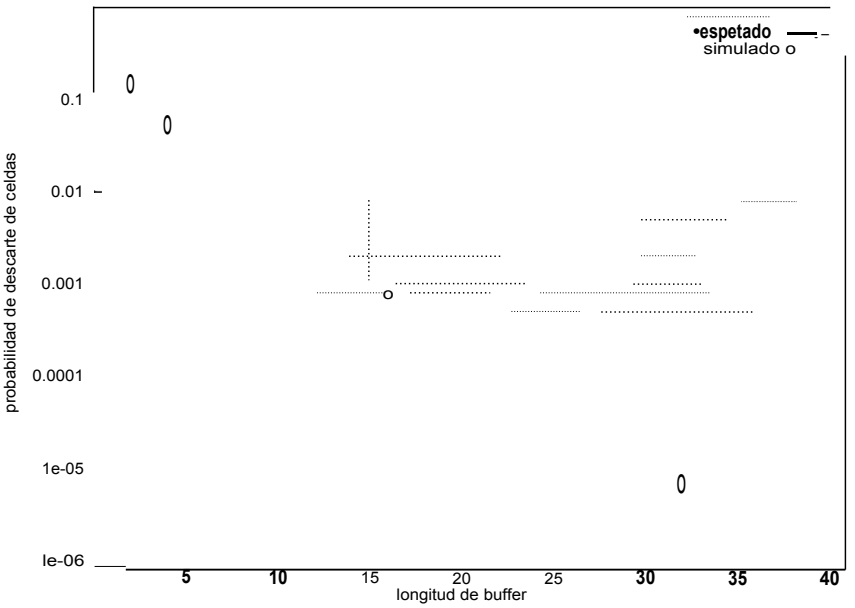


Figura 4.14: Probabilidad de pérdida de células para un conmutador crossbar 4 x 4 con buffers en los puertos de salida. Se comparan los resultados teóricos esperados con los obtenidos por simulación.

Dado que no existen modelos teóricos para verificar la validez del sistema, comparamos los resultados de este modelo con los presentados en [13] usando similares condiciones de simulación.

Inyectando tráfico con distribución de probabilidad uniforme al modelo de un conmutador de 8 puertos y longitud de buffers 16 controlado por redes neuronales, la simulación presenta un flujo relativo respecto al número de células que ingresan al conmutador muy aproximado al reportado en [13] (ver Figura 4.15), lo que se prueba, aunque parcialmente dado que los modelos no son exactamente iguales, la validez del modelo construido.

En la simulación de la red neuronal propuesta, se asume que la actualización de los nodos de la red se realiza en forma asíncrona, uno cada vez. Aleatoriamente se escoge el orden en que las neuronas actualizan sus estados. Debido a que la red puede alcanzar mínimos de energía que corresponden a arreglos en la estructura de conmutación que podrían ser no óptimos desde el punto de vista del flujo máximo, el rendimiento de la red no siempre es el mismo para una misma secuencia de tráfico de entrada. El rendimiento final depende de la secuencia aleatoria de actualización de los nodos de la red.

En el siguiente capítulo se muestran los resultados de la simulación realizada para los modelos de conmutación con buffers en los puertos de entrada, con buffers en los puertos de salida, y nuestro modelo propuesto controlado por una red neuronal

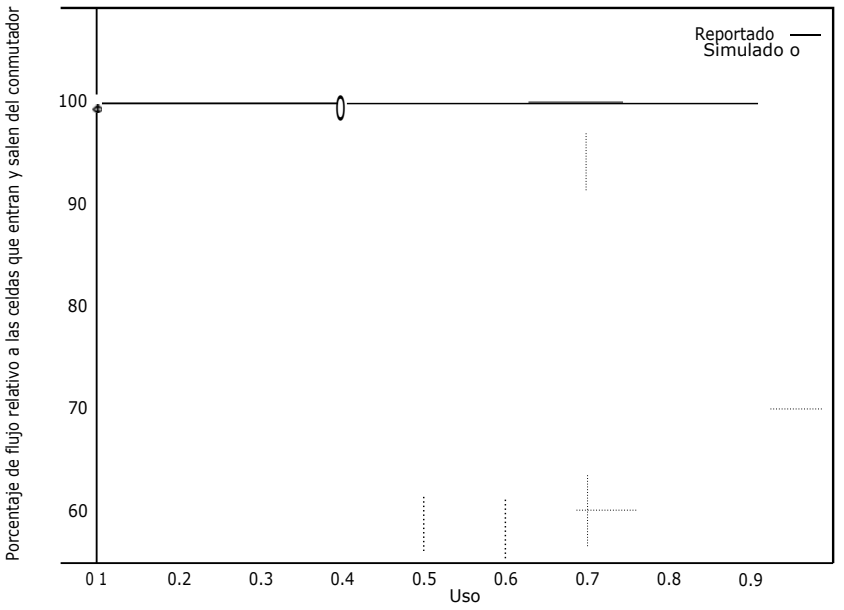


Figura 4.15: Probabilidad de pérdida de células en un conmutador crossbar 4 x 4 controlado por una red neuronal Hopfield.

Hopfield. Para este último caso se muestran los resultados promedio de 10 intentos para hallar el rendimiento, con una precisión de +10%.

Capítulo 5

Análisis de los resultados

Los modelos de conmutación y tráfico descritos en los capítulos anteriores junto con la red neuronal Hopfield propuesta, se implementan en programas de computadora y se realizan varios experimentos para obtener resultados que permitan evaluar y comparar el comportamiento de los sistemas. En este capítulo se muestran y discuten los resultados de la simulación de los modelos.

5.1 Conmutadores con buffers FIFO en los puertos de entrada o salida

La Figura 5.1 grafica el flujo versus la utilización de las líneas que se conectan a los puertos de entrada, para un conmutador 4 x 4 con capacidad de almacenar 4 células en cada buffer FIFO de cada puerto. Se comparan gráficamente los casos de conmutadores con buffers colocados en los puertos de entrada, y conmutadores con buffers colocados en los puertos de salida. El tráfico de entrada ha sido modelado como un proceso auto-similar. [1]

Puede observarse un flujo muy cercano al ideal en el modelo con buffers en los puertos de salida del conmutador. Este resultado es el esperado, dado que este modelo tiene un comportamiento ideal en lo que se refiere a la conmutación interna, sólamente limitado por la capacidad de almacenamiento de los buffers en los puertos de salida para retener las células hasta su transmisión. Intuitivamente, se lograría un flujo perfecto si se amplia la capacidad de los buffers. Este caso se estudiará más adelante.

Tal como se habia comentado anteriormente, al colocar los buffers FIFO en los puertos de entrada de un conmutador, obtenemos un rendimiento severamente degradado si lo comparamos con el caso anterior, principalmente debido al bloqueo que ocasionan las primeras células de los buffers (efecto HOL). Este fenómeno, combina-

[1] Para abreviar el rotulado de las gráficas, se usará *entrada* para indicar el uso de buffers FIFO en los puertos de entrada del conmutador, *salida,* cuando se tiene los buffers FIFO en los puertos de salida del conmutador, *y neuronal* cuando se utilizan buffers FIFO separados para cada salida en cada entrada con un controlador basado en redes neuronales para controlar la conmutación interna de células.

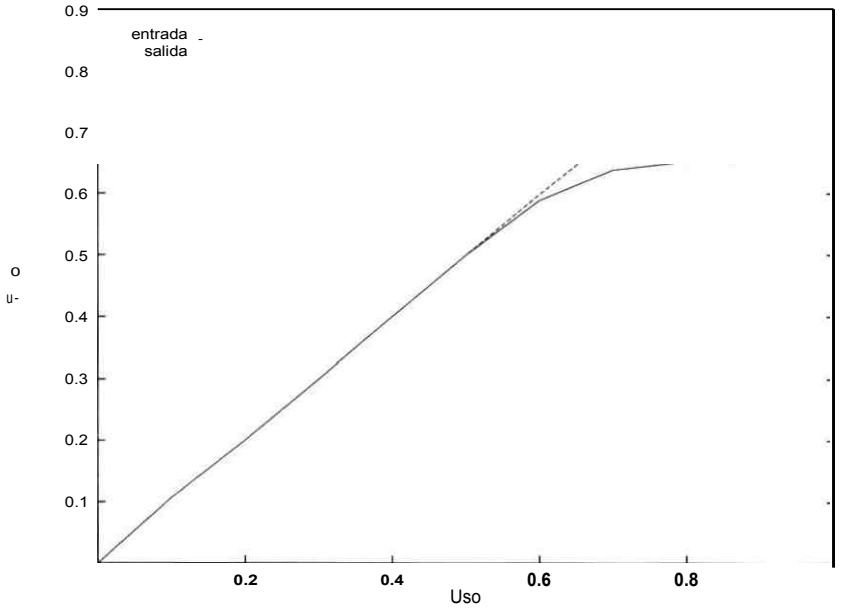


Figura 5.1: Flujo versus utilización en un conmutador 4x4 bajo dos estrategias en el uso de los buffers.

do con la dependencia de amplio rango del tráfico de entrada, ocasiona que muchas células ocupen los buffers, lo que provoca una alta tasa de descarte de células y un flujo pobre.

El incremento en el tamaño de los buffers debería mejorar hasta cierto punto el rendimiento del flujo. La Figura 5.2 grafica el flujo versus la utilización de las líneas para el modelo con buffers FIFO en los puertos de entrada con capacidad de almacenar 2, 4 y 8 células respectivamente. En la Figura 5.3 se muestra un caso similar para el modelo de conmutación con buffers en los puertos de salida.

El mejoramiento del flujo en el primer caso no es sustancial con el incremento del tamaño de los buffers. La explicación de este fenómeno es que el efecto HOL provoca que los buffers se mantengan permanentemente ocupados, lo que genera pérdida de células y un flujo pobre. El segundo caso si muestra un mejoramiento, tal como se previó intuitivamente. Su tendencia es llegar a una línea recta con pendiente 1, que sería el caso ideal.

La probabilidad de pérdida de células versus la utilización de las líneas de los puertos de entrada se muestran en las Figuras 5.4 y 5.5, para conmutadores con buffers FIFO en los puertos de entrada y salida respectivamente.

Se puede observar una alta probabilidad de pérdida de células en el primer esquema, a partir de una mediana utilización de las líneas, que no mejora sustancialmente ampliando la capacidad de los buffers. Esto confirma la idea de que los buffers en este esquema están permanentemente ocupados, y generan una alta tasa de descarte

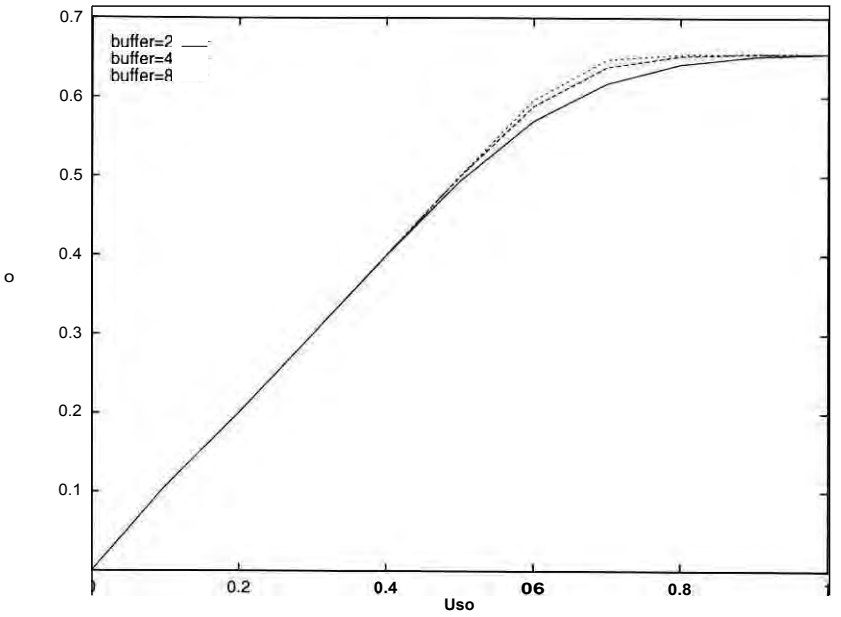


Figura 5.2: Flujo versus utilización de las líneas en un conmutador 4x4 con buffers FIFO en los puertos de entrada.

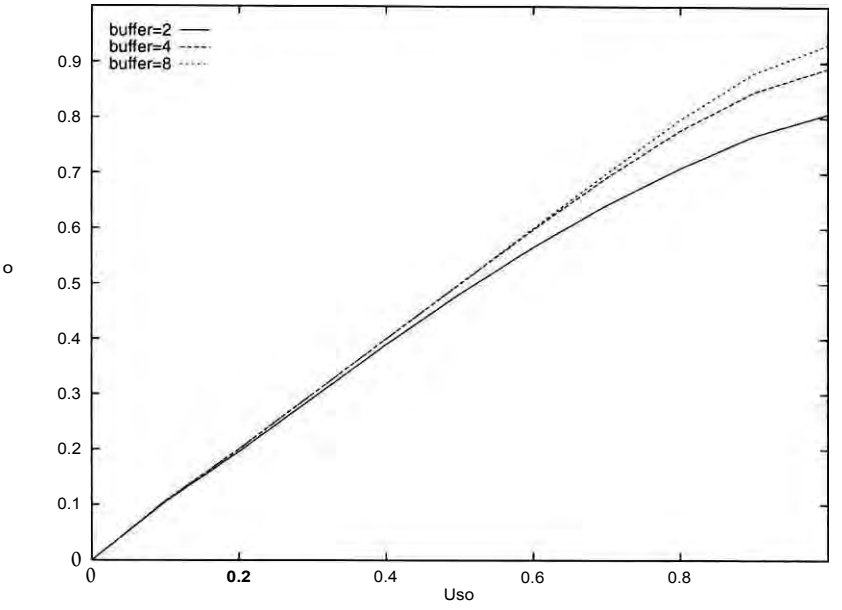


Figura 5.3: Flujo versus utilización de las líneas en un conmutador 4x4 con buffers FIFO en los puertos de salida.

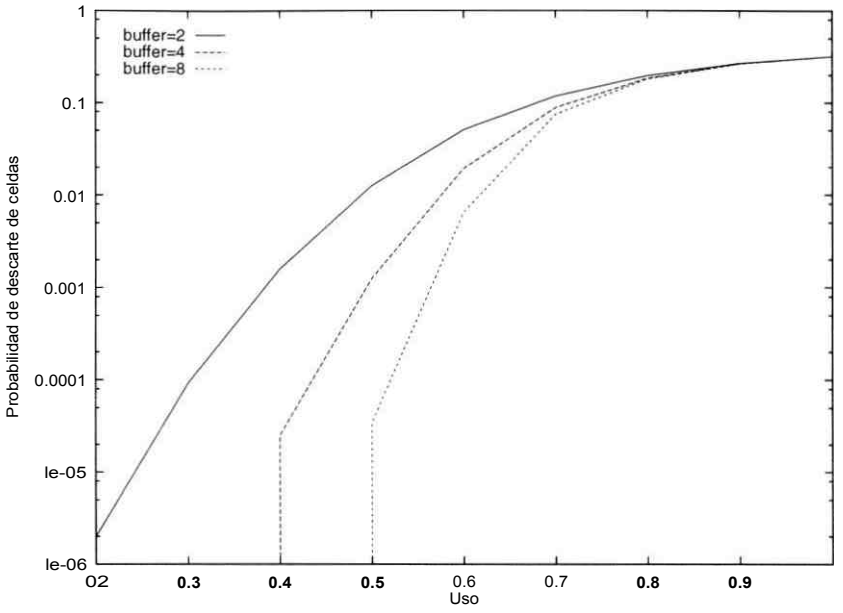


Figura 5.4: Probabilidad de pérdida de células versus utilización de las líneas en un conmutador 4x4 con buffers FIFO en los puertos de entrada.

de células.

La probabilidad de pérdida de células sí mejora ampliando la capacidad de los buffers en el caso de tener buffers FIFO en las salidas del conmutador. Dado que no existen problemas de conflicto interno en este esquema, si se amplia la capacidad de los buffer para poder retener a todas las células antes de su transmisión, se podría lograr un rendimiento del flujo muy cercano al ideal. Por lo tanto, debería de existir un punto crítico en el tamaño de los buffers que permita en este esquema, retener todas las células que llegan para lograr una probabilidad de pérdida de células nula. Es claro que este punto crítico dependerá de la forma en que se modele el tiempo de llegada entre células.

5.2 odelo de conmutación usando redes neuronales Hopfield

Los resultados que se muestran para el modelo de conmutación controlado por redes neuronales, son los valores promedio de 10 intentos con una presición de ±10%. Los valores que se obtienen para una misma secuencia de tráfico de entrada no siempre son iguales porque dependen de la actualización aleatoria (con distribución uniforme) de las neuronas de la red neuronal. Como se discutió anteriormente, la red neuronal puede alcanzar estados estables que podrían corresponder a esquemas

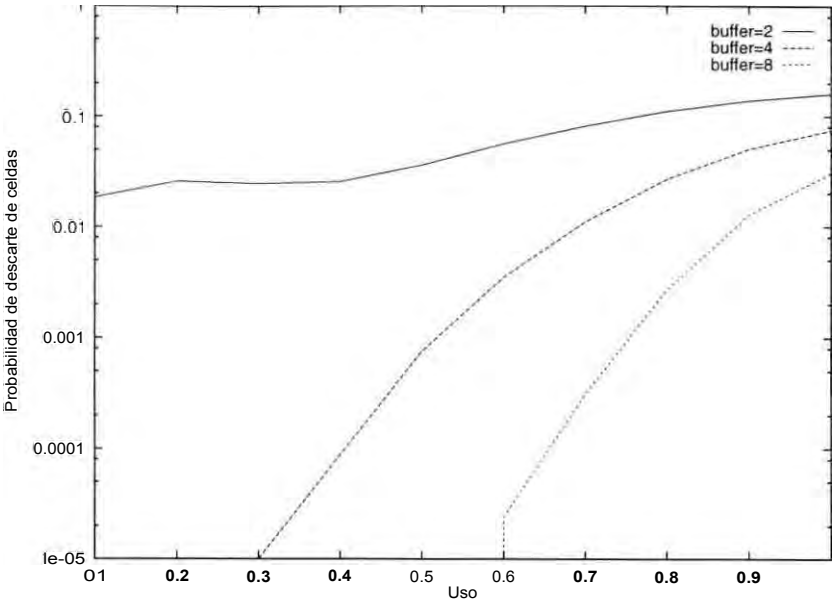


Figura 5.5: Probabilidad de pérdida de células versus utilización de las líneas en un conmutador 4x4 con buffers FIFO en los puertos de salida.

de conmutación interna no óptimas para el rendimiento.

Las Figuras 5.6, 5.7 y 5.8 comparan el flujo versus la utilización de las líneas de entrada para los modelos con buffers FIFO en los puertos de entrada, salida y el propuesto usando una red neuronal Hopfield.

Puede apreciarse que la separación de las células según su destino en los puertos de entrada, combinado con el uso de la red neuronal Hopfield para seleccionar la conmutación interna ayuda a mejorar el flujo del conmutador respecto al esquema general con un buffer FIFO por puerto de entrada. Su rendimiento mejora con el incremento del tamaño de los buffers, sin embargo, en forma no tan notoria como el que experimenta el modelo con buffers en los puertos de salida. Aparentemente el modelo presenta un punto de saturación del flujo como el caso del modelo con un buffer FIFO por puerto de entrada, que no ha sido reportado anteriormente. Este caso se ampliará más adelante.

Las Figuras 5.9, 5.10 y 5.11 comparan la probabilidad de pérdida de células versus la utilización de las líneas de entrada, para los tres esquemas de conmutación tratados.

El modelo con buffers FIFO en los puertos de salida presenta una mayor tasa de pérdida de células que los otros dos esquemas, cuando el tamaño de los buffers es muy pequeño, y para una utilización baja de las líneas de entrada. Sin embargo, su rendimiento mejora notablemente ampliando la capacidad de los buffers.

En general, para cualquier longitud de los buffers, el modelo con un controlador

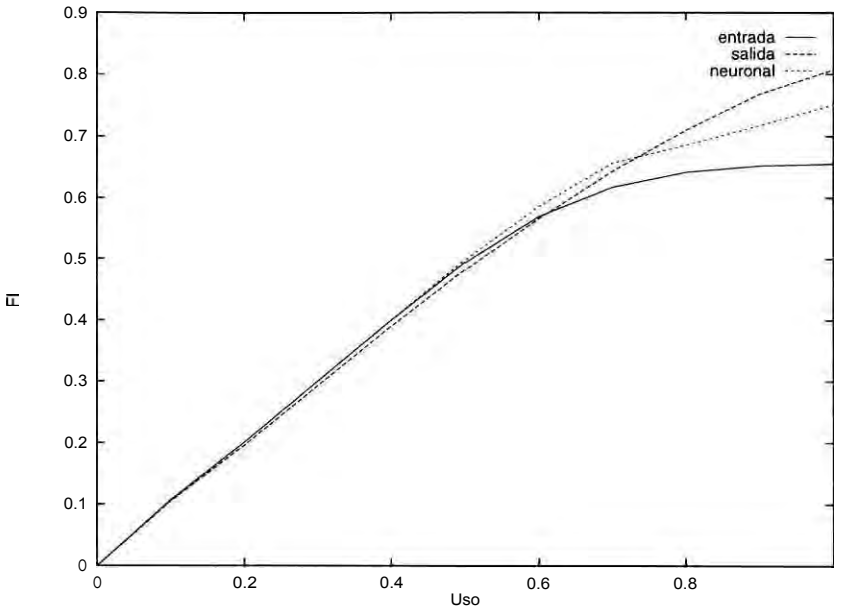


Figura 5.6: Flujo versus utilización en un conmutador 4x4 con 2 buffers por puerto bajo tres estrategias en el uso de los buffers.

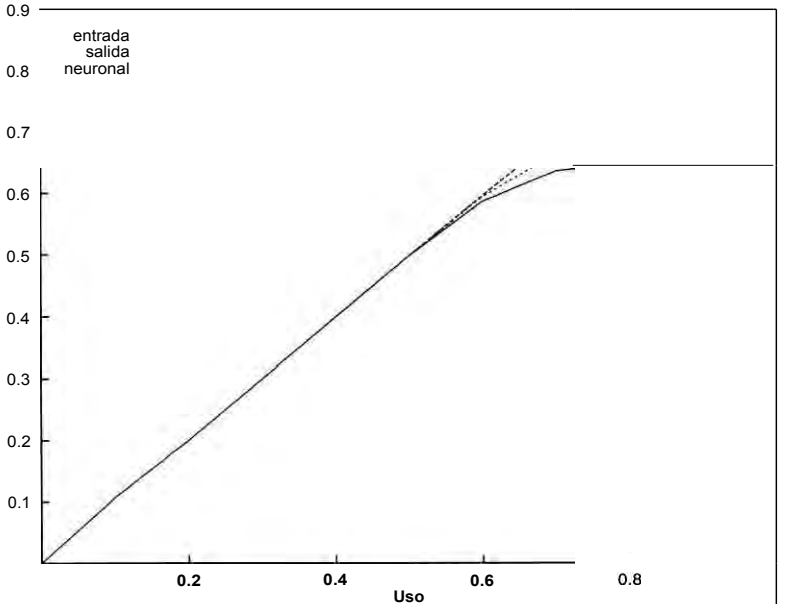
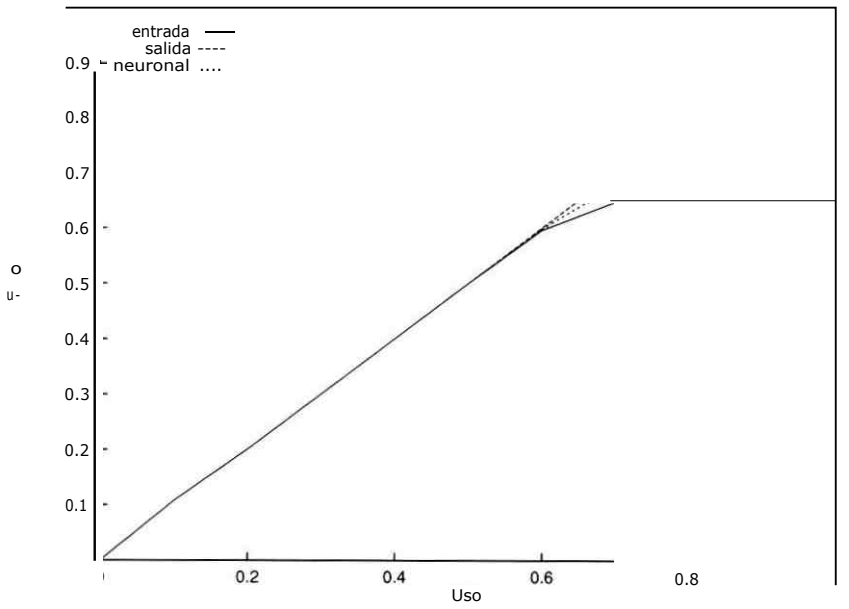


Figura 5.7: Flujo versus utilización en un conmutador 4x4 con 4 buffers por puerto bajo tres estrategias en el uso de los buffers.



Figura, 5.8: Flujo versus utilización en un conmutador 4x4 con 8 buffers por puerto bajo tres estrategias en el uso de los buffers.

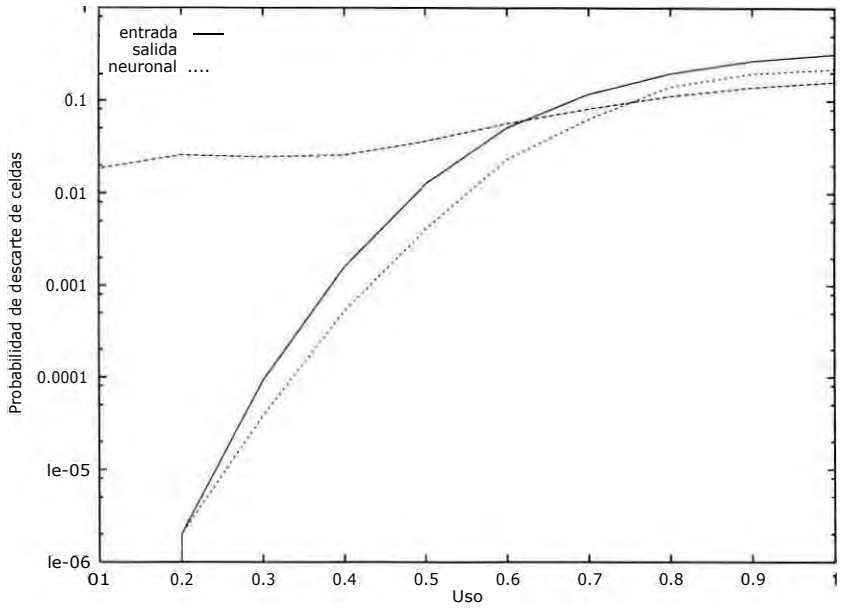


Figura 5.9: Probabilidad de pérdida de células versus utilización en un conmutador 4x4 con 2 buffers por puerto bajo tres estrategias en el uso de los buffers.

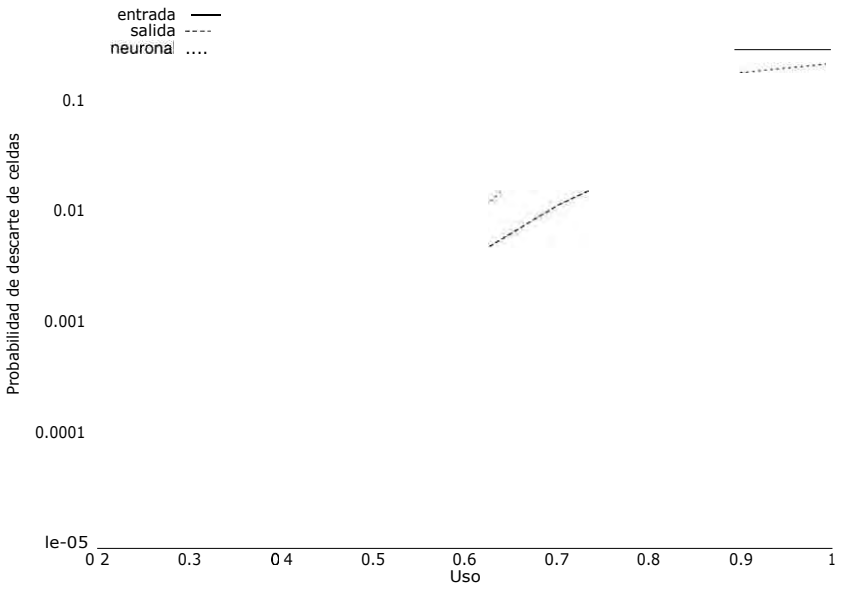


Figura 5.10: Probabilidad de pérdida de células versus utilización en un conmutador 4x4 con 4 buffers por puerto bajo tres estrategias en el uso de los buffers.

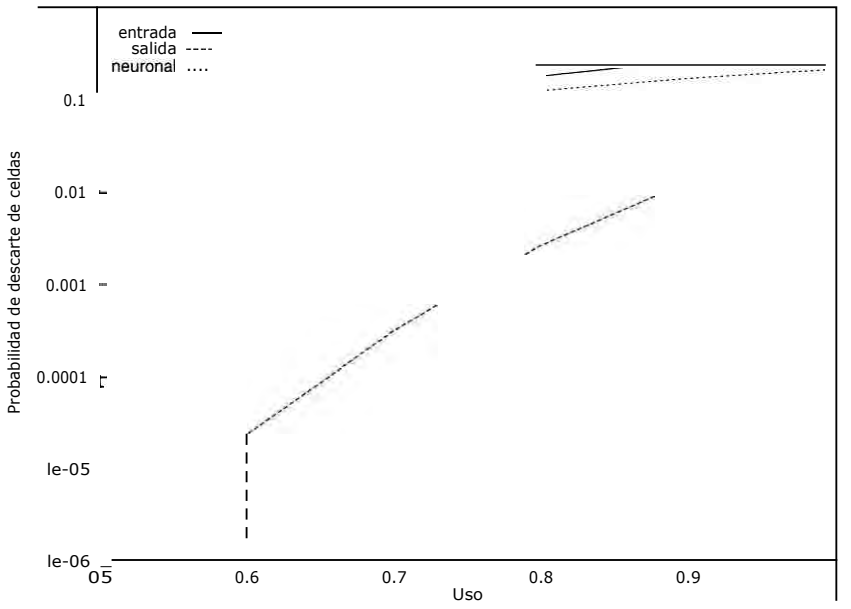


Figura 5.11: Probabilidad de pérdida de células versus utilización en un conmutador 4x4 con 8 buffers por puerto bajo tres estrategias en el uso de los buffers.

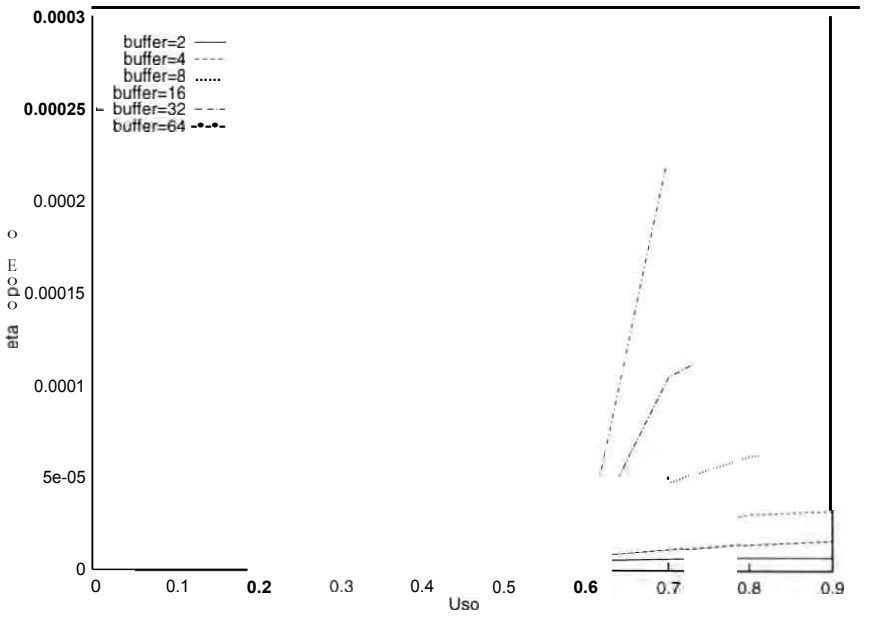


Figura 5.12: Retardo promedio de células en los buffers versus utilización en un conmutador 4x4 con buffers en los puertos de entrada.

basado en redes neuronales presenta una menor probabilidad de pérdida de células que el modelo con buffers FIFO en los puertos de entrada. Estos dos modelos presentan pequeñas mejoras en el rendimiento de la probabilidad de pérdida de células con la ampliación de la capacidad de los buffers, sobre todo para cargas bajas y medianas del tráfico de entrada.

Una de las causas que determina que el modelo que usa un controlador basado en redes neuronales no tenga un notorio mejoramiento con la ampliación de la capacidad de los buffers, es que la red neuronal puede alcanzar un estado estable que provoque una conmutación interna no óptima, lo que ocasiona que los buffers no se liberen satisfactoriamente, con lo que aumenta la probabilidad de descarte para las nuevas células que lleguen al conmutador.

5.3 Estudio del retardo promedio de las células

Las Figuras 5.12, 5.13 y 5.14 muestran el retardo promedio que sufren las células bajo estructuras de conmutación con buffers de diferentes tamaños en los puertos de entrada, salida, y controlados por redes neuronales respectivamente.

Las células que son procesadas en el esquema de conmutación con buffers FIFO en los puertos de entrada sufren un retardo muy fuerte para buffers de longitud grande y para intensidades de tráfico mayores a 0.6. Para el esquema con buffers en los puertos de salida, el retardo apreciable se presenta para buffers de tamaño

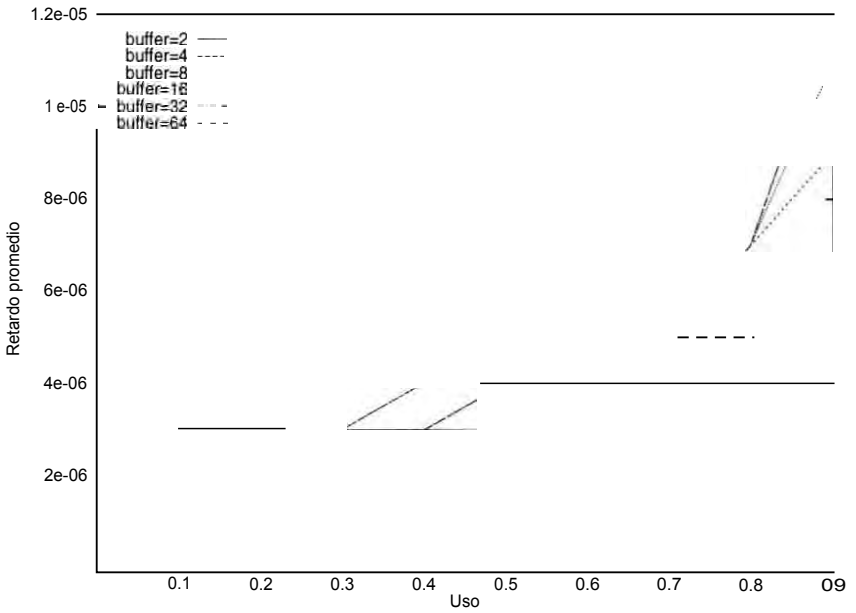


Figura 5.13: Retardo promedio de células en los buffers versus utilización en un conmutador 4x4 con buffers en los puertos de salida.

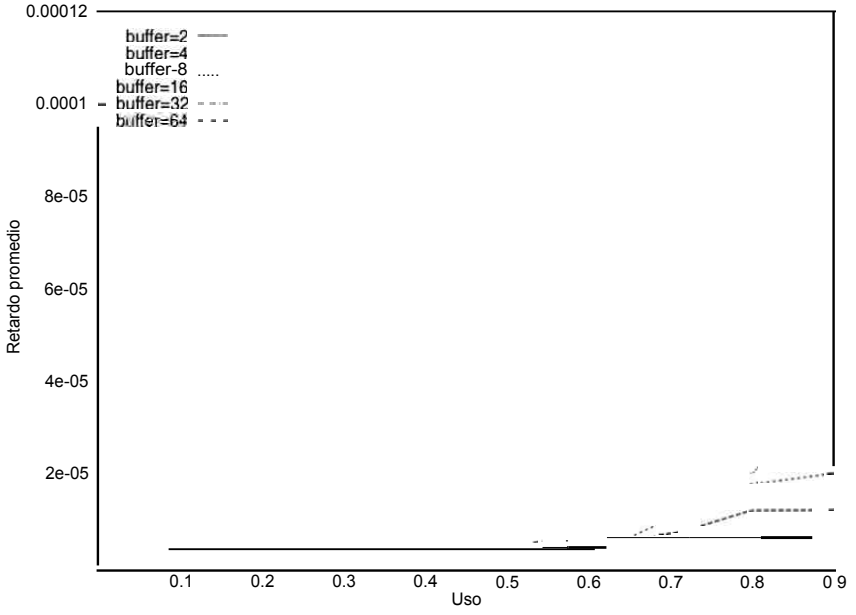


Figura 5.14: Retardo promedio de células en los buffers versus utilización en un conmutador 4x4 controlado con redes neuronales.

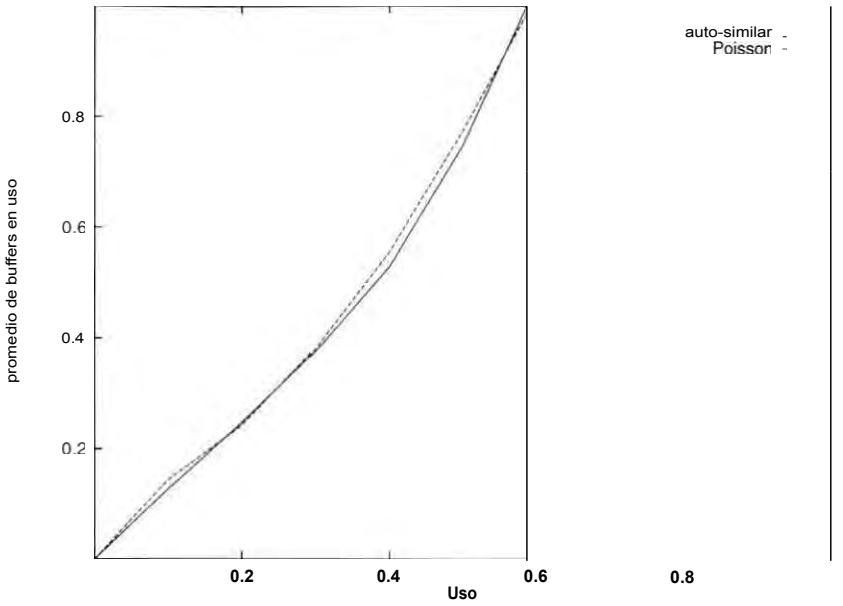


Figura 5.15: Comparación de la ocupación promedio de los buffers para un conmutador con colas de longitud 2 en los puertos de entrada, bajo tráfico modelado con procesos auto-similares y Poisson.

grande en las zonas de tráfico alto en las que prácticamente todos los intervalos de tiempo están ocupados por alguna célula. El retardo en el esquema de conmutación controlado por redes neuronales tiene un comportamiento similar al que presenta el modelo de buffers FIFO en los puertos de entrada, pero el retardo es menor en un 40%.

El retardo promedio que sufren las células es mayor cuando el tráfico de entrada se modela como un proceso auto-similar que cuando se modela como un proceso Poisson. El retardo está directamente relacionado a la ocupación promedio de los buffers. En las Figuras 5.15, 5.16 y 5.17 se muestra la ocupación promedio de los buffers para los tres casos estudiados cuando la longitud máxima del buffer es 2. Las Figuras 5.18, 5.19 y 5.20 muestran un caso similar con buffers de longitud 8.

5.4 Efecto del tráfico auto-similar

En esta sección se estudia el efecto que tienen los procesos auto-similares en el rendimiento de los conmutadores tratados, con respecto a los resultados que preveen los modelos Poisson de tráfico de datos. Anteriormente se comentó que tradicionalmente se han usado modelos estadísticos que presentan dependencia de corto rango, donde no existe relación entre los elementos de la secuencia de datos para modelar el tráfico de entrada de las redes de paquetes. Uno de los objetivos de esta tesis es

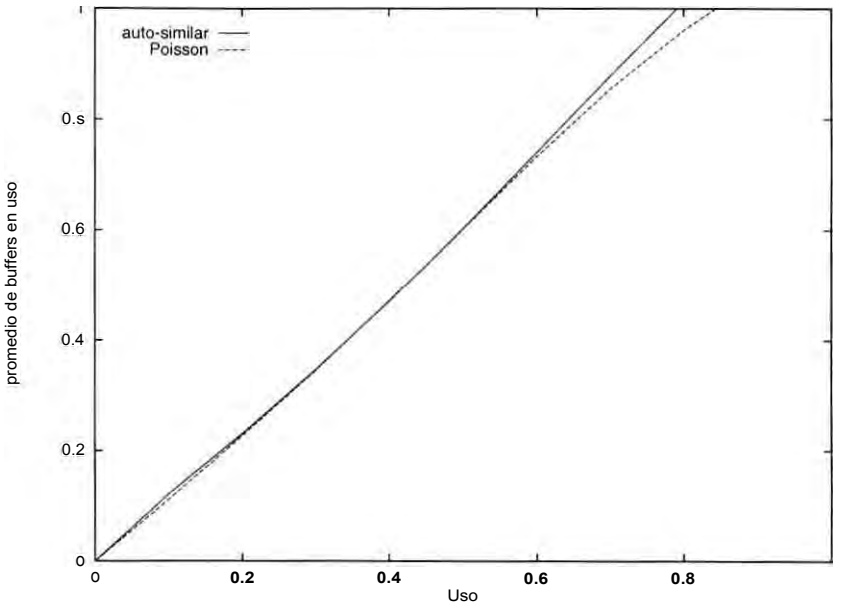


Figura 5.16: Comparación de la ocupación promedio de los buffers para un conmutador con colas de longitud 2 en los puertos de salida, bajo tráfico modelado con procesos auto-similares y Poisson.

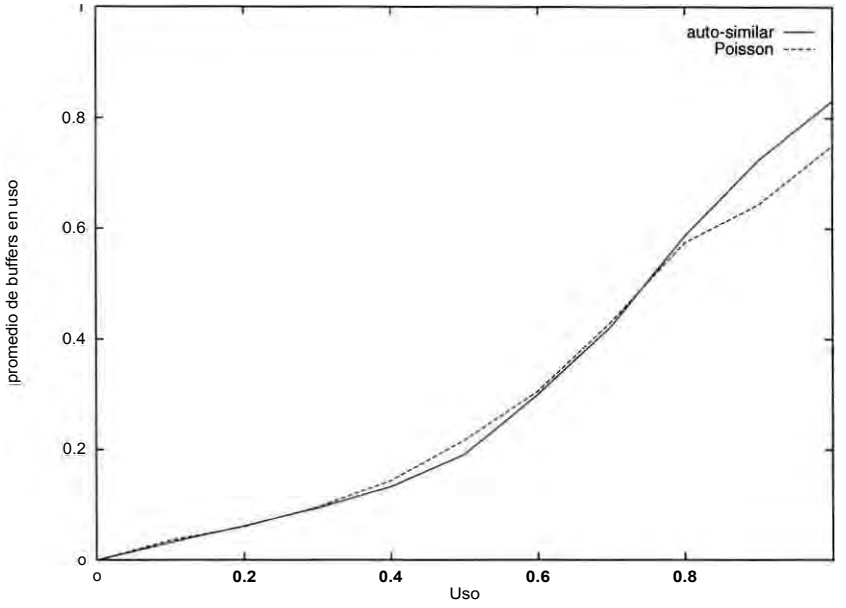


Figura 5.17: Comparación de la ocupación promedio de los buffers para un conmutador con colas de longitud 2 en los puertos de entrada controlado por redes neuronales, bajo tráfico modelado con procesos auto-similares y Poisson.

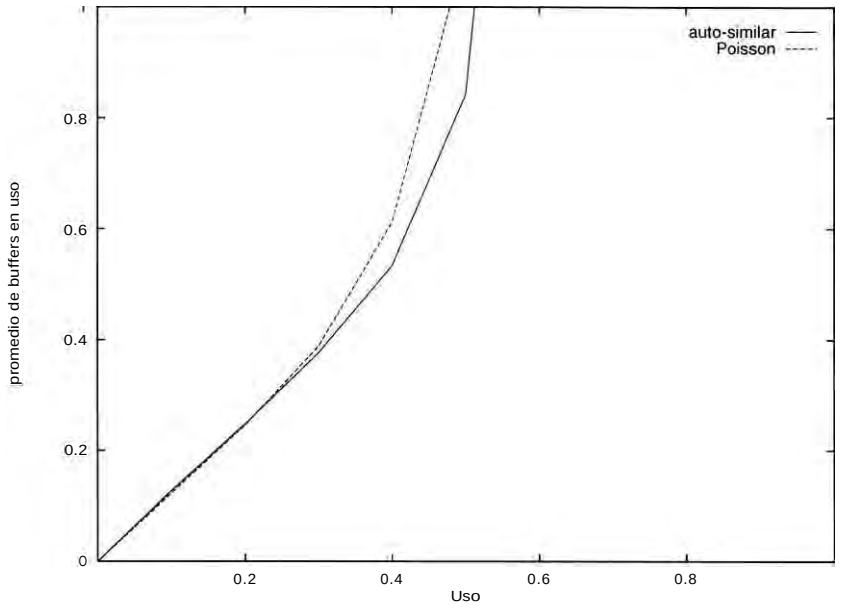


Figura 5.18: Comparación de la ocupación promedio de los buffers para un conmutador con colas de longitud 8 en los puertos de entrada, bajo tráfico modelado con procesos auto-similares y Poisson.

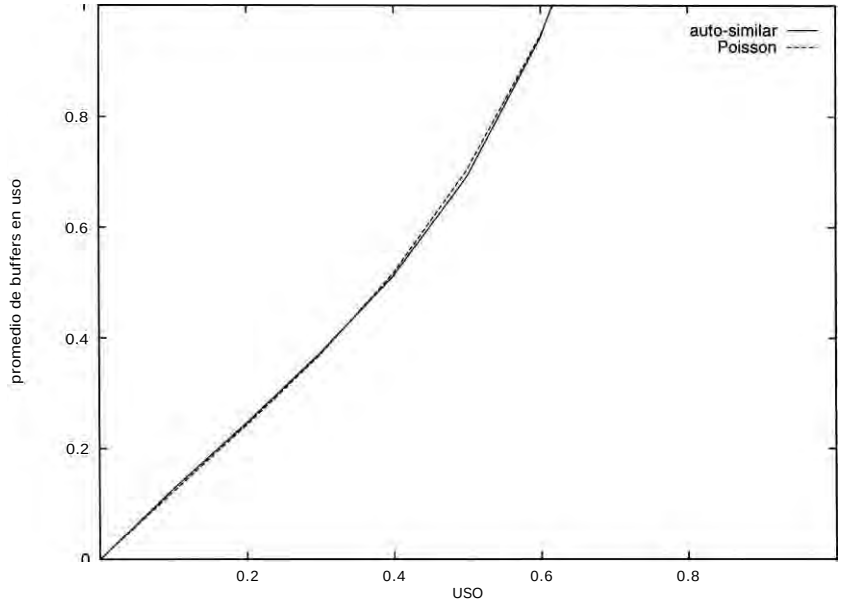


Figura 5.19: Comparación de la ocupación promedio de los buffers para un conmutador con colas de longitud 8 en los puertos de salida, bajo tráfico modelado con procesos auto-similares y Poisson.

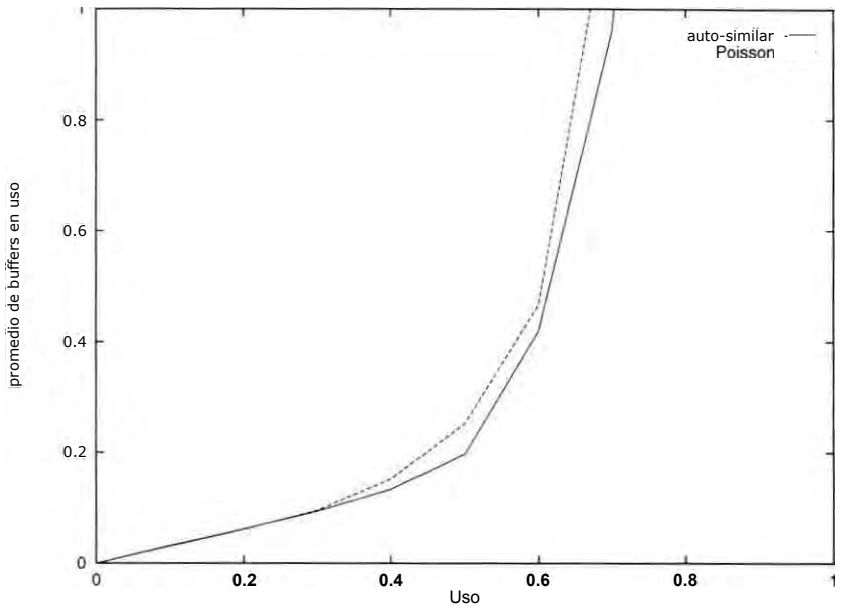


Figura 5.20: Comparación de la ocupación promedio de los buffers para un conmutador con colas de longitud 8 en los puertos de entrada controlado por redes neuronales, bajo tráfico modelado con procesos auto-similares y Poisson.

presentar resultados basados en procesos auto-similares, debido a que estos modelos capturan mejor la dependencia de amplio rango del tráfico real. Lo que sigue es una comparación de los resultados de ambos esquemas en los modelos de conmutación construidos.

Las Figuras 5.21, 5.22 y 5.23 muestran el flujo, con tráfico de excitación modelado como un proceso Poisson o auto-similar, para los tres esquemas de conmutación tratados. La explicación del mayor flujo que presentan los modelos excitados por procesos auto-similares respecto a los modelos excitados por procesos Poisson, puede hallarse en la menor varianza que presenta el modelamiento del tráfico con procesos auto-similares que la prevista por Poisson.

Los resultados para altas cargas de tráfico deben de tender a ser similares para ambos tipos de modelamiento de tráfico, dado que prácticamente todos los intervalos de tiempo están ocupados por células.

En las Figuras 5.24, 5.25 y 5.26 puede comprobarse que la probabilidad de pérdida de células en los modelos de conmutación estudiados, es mayor cuando el tráfico es modelado como un proceso auto-similar que cuando se modela como un proceso Poisson, cuando el uso de las líneas de entrada es alto. Para tráfico mediano y ligero, se tiene un fenómeno inverso. Esto puede explicarse dado que la varianza que los procesos Poisson preveen para el tráfico es alta, lo que provoca altos picos en el número de paquetes en ciertos intervalos de tiempo aún con tráfico ligero, que

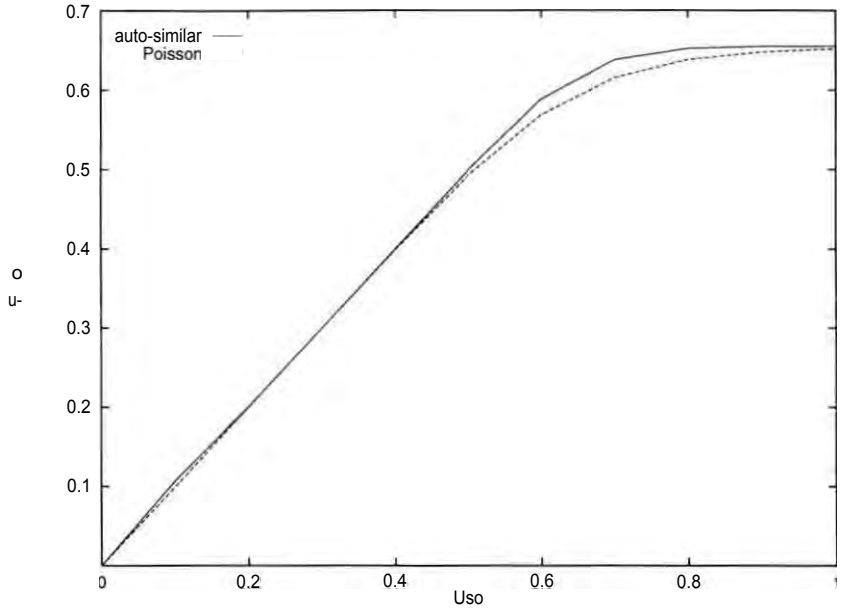


Figura 5.21: Flujo versus utilización en un conmutador 4x4 con 4 buffers por puerto de entrada bajo tráfico modelado con procesos Poisson y auto-similares.

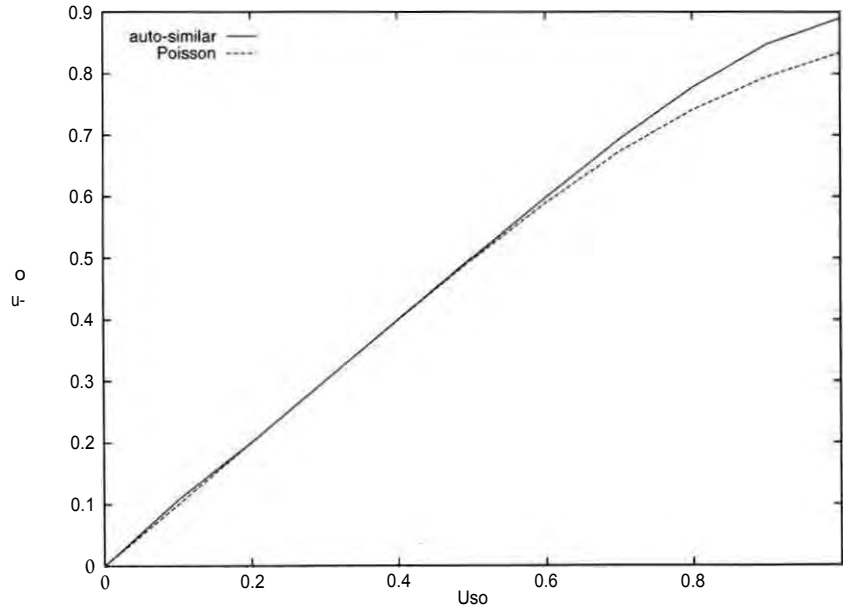


Figura 5.22: Flujo versus utilización en un conmutador 4x4 con 4 buffers por puerto (le salida bajo tráfico modelado con procesos Poisson y auto-similares.

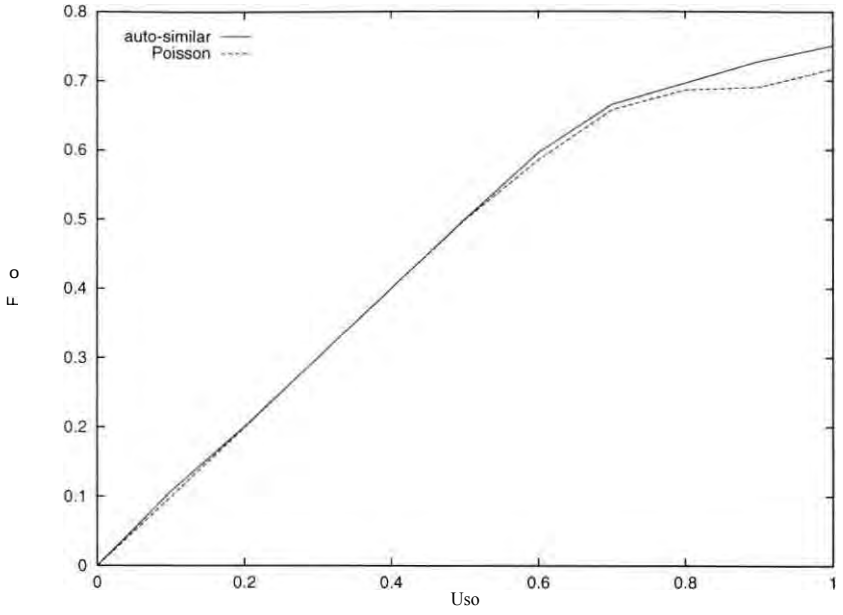


Figura 5.23: Flujo versus utilización en un conmutador 4x4 controlado por redes neuronales bajo tráfico modelado con procesos Poisson y auto-similares.

generan pérdida de células.

Los procesos auto-similares provocan un mayor retardo promedio en las células que los procesos Poisson. Este efecto puede apreciase en las Figuras 5.27, 5.28 y 5.29. Los conmutadores mostrados tienen buffers de longitud 64.

5.4.1 Efecto de la varianza del proceso auto-similar de entrada

Para este estudio, se ha considerado una varianza fija de 1 para todos los valores de la media del proceso auto-similar que modela el tráfico de entrada a los puertos de los modelos de los conmutadores.

En el tráfico real, la relación entre la media y la varianza del proceso no ha sido aún estudiada completamente. Es de esperarse, que la varianza decrezca según se incrementa la carga de tráfico, pero a un valor muy inferior al que prevee el proceso de Poisson. La prueba de esta aseveración puede encontrarse heurísticamente en las gráficas de la varianza para las diferentes agregaciones de la secuencia de tráfico real mostrada anteriormente. Pero no existe una fórmula simple que relacione la varianza con la media del tráfico. En general existe una inmensa variación entre estos dos momentos estadísticos cuando se evalúa tráfico real.

Debido a esto, incluimos un comentario sobre del efecto de diferentes valores de varianza en los resultados.

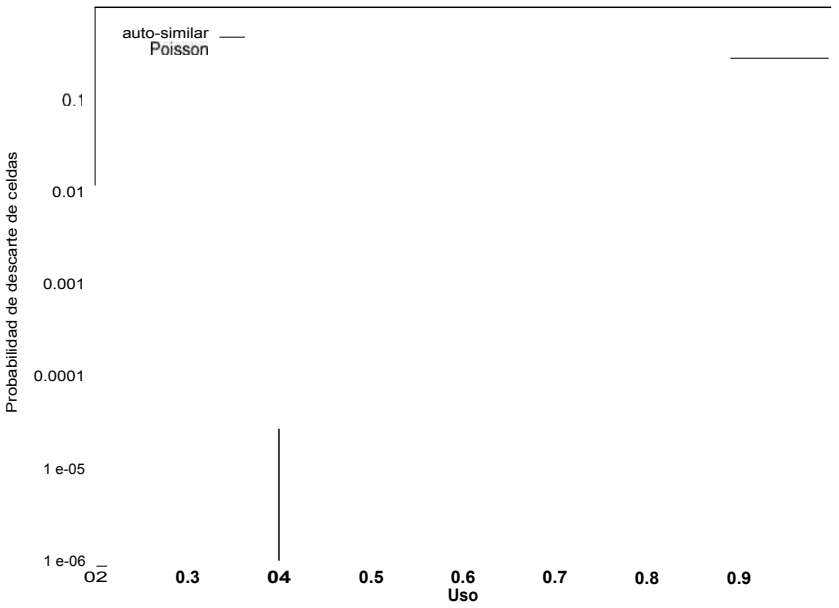


Figura 5.24: Probabilidad de pérdida de células versus utilización en un conmutador 4x4 con 4 buffers por puerto de entrada bajo tráfico modelado con procesos Poisson y auto-similares.

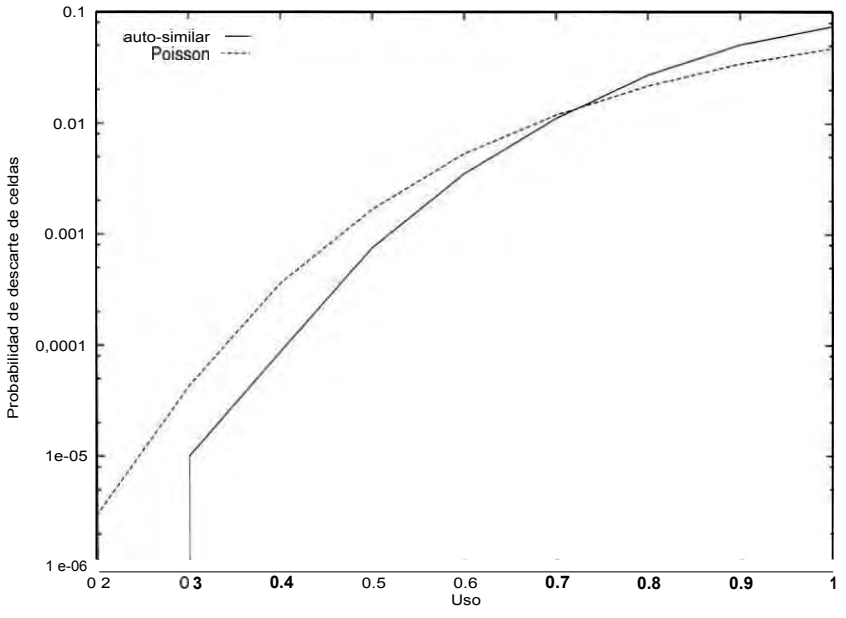


Figura 5.25: Probabilidad de pérdida de células versus utilización en un conmutador 4x4 con 4 buffers por puerto de salida bajo tráfico modelado con procesos Poisson y auto-similares.

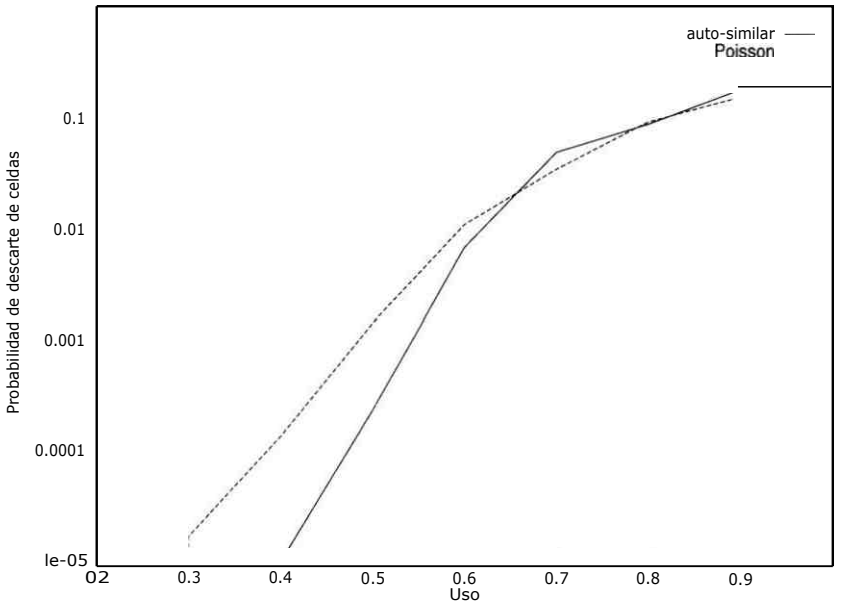


Figura 5.26: Probabilidad de pérdida de células versus utilización en un conmutador 4x4 controlado con redes neuronales bajo tráfico modelado con procesos Poisson y auto-similares.

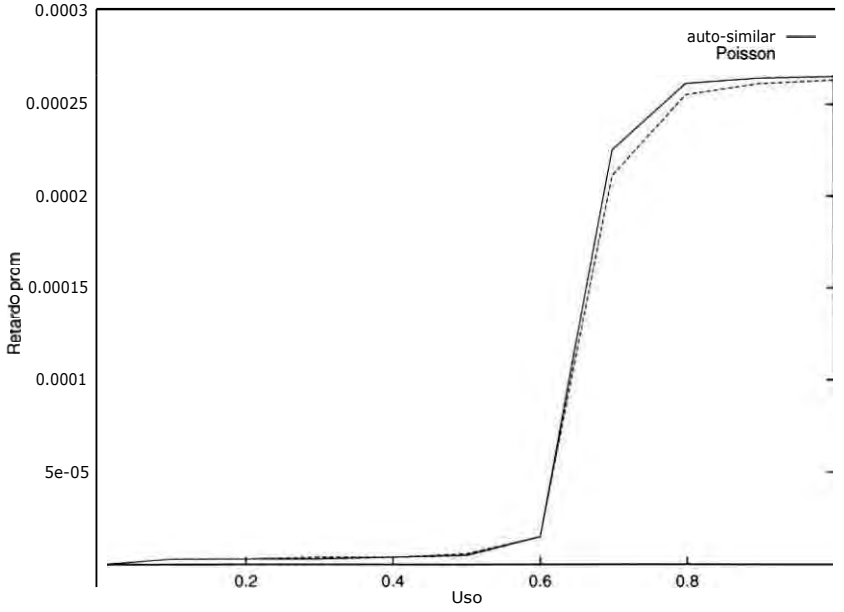


Figura. 5.27: Retardo promedio de células en los buffers de longitud 64 versus utilización en un conmutador 4x4 con buffers en los puertos de entrada.

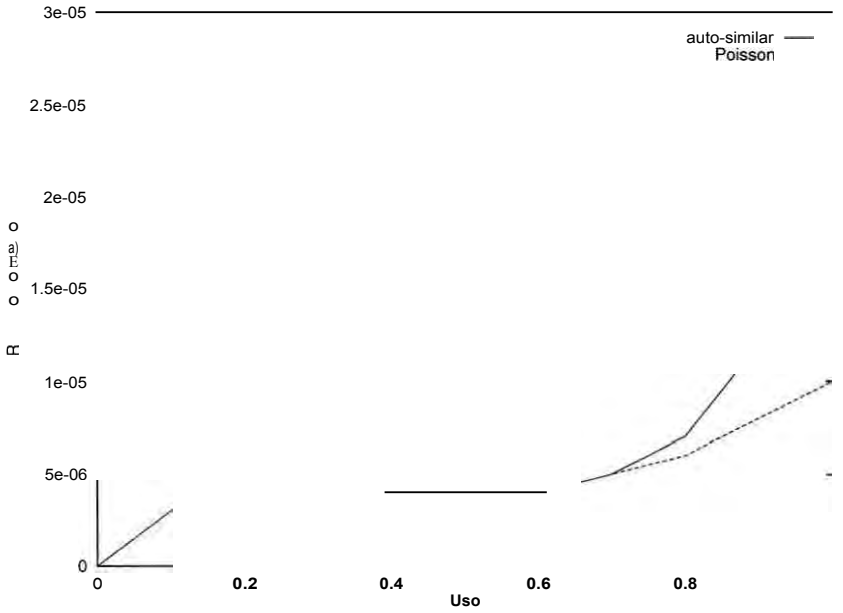


Figura 5.28: Retardo promedio de células en los buffers de longitud 64 versus utilización en un conmutador 4x4 con buffers en los puertos de salida.

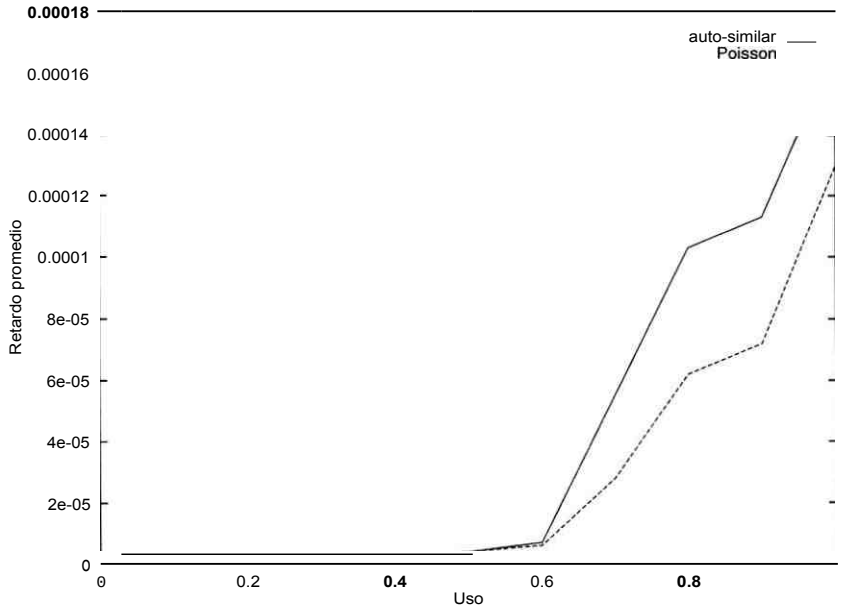


Figura 5.29: Retardo promedio de células en los buffers de longitud 64 versus utilización en un conmutador 4x4 controlado por redes neuronales.

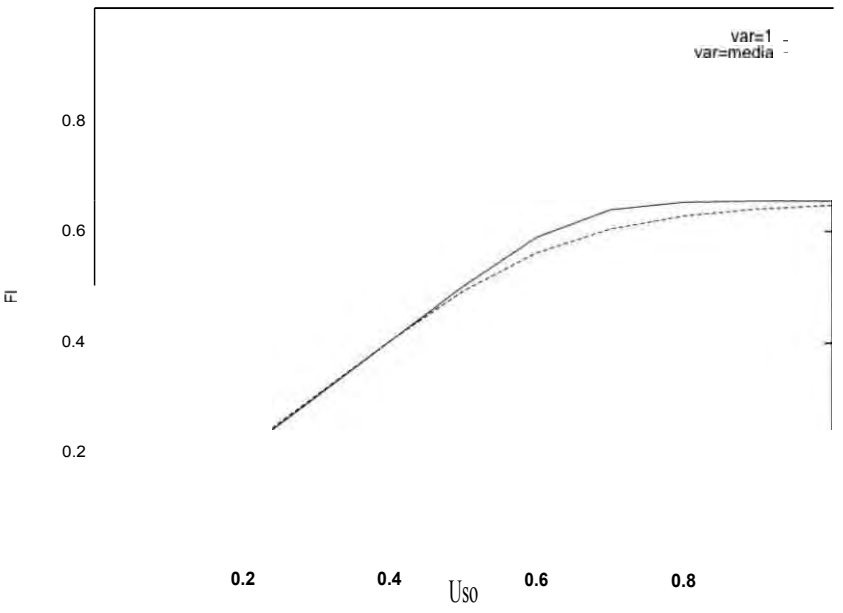


Figura 5.30: Comparación del flujo en el modelo del conmutador con buffers de longitud 4 en los puertos de entrada, bajo tráfico auto-similar con varianza 1 y varianza igual a la media.

En las Figuras 5.30, 5.31 y 5.32 se muestra el flujo para los conmutadores estudiados con buffers de longitud 4.

Se han tomado dos valores para la varianza: el primero es el valor de uno, que es el caso de los resultados presentados anteriormente, y el segundo es el valor límite igual a la media, que lo acerca a un proceso de Poisson.

Puede comprobarse de las figuras, que existe un rango en que podrían variar los resultados dependiendo de la varianza del tráfico auto-similar de entrada. Es interesante comentar que en el caso límite de tener un proceso auto-similar con media y varianza iguales, el rendimiento de los modelos de conmutación tratados son parecidos a los obtenidos con procesos Poisson.

5.5 Efecto del tamaño de los buffers

Sería interesante observar el máximo flujo que puede obtenerse en cada esquema de conmutación incrementando el tamaño de los buffers. El incremento de la capacidad de los buffers debería mejorar la tasa de pérdida de células en cada modelo de conmutación, excepto para el caso de buffers FIFO en los puertos de entrada, debido al efecto HOL como se discutió anteriormente.

En las Figuras 5.33, 5.34 y 5.35 se muestra el flujo máximo alcanzable con las tres estructuras de conmutación tratadas según se incrementa la longitud de los buffers.

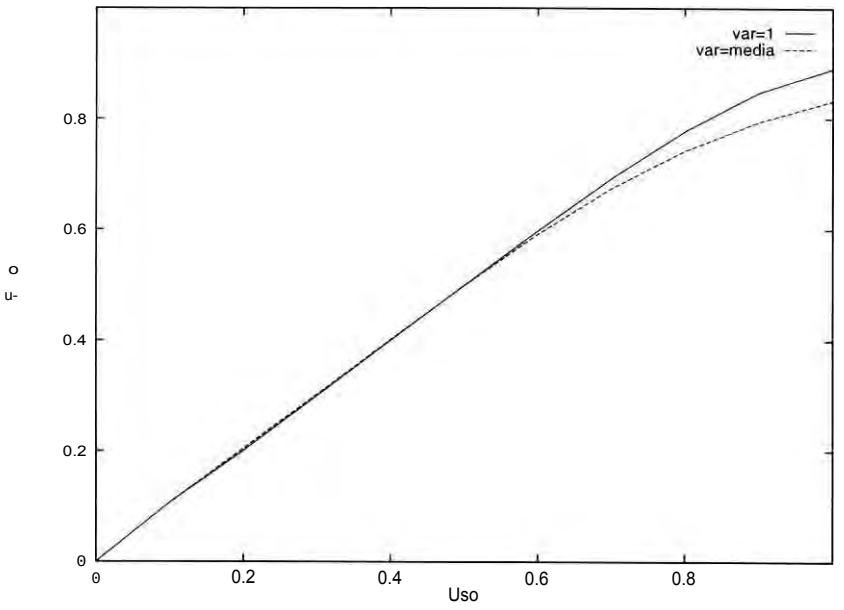


Figura 5.31: Comparación del flujo en el modelo del conmutador con buffers de longitud 4 en los puertos de salida, bajo tráfico auto-similar con varianza 1 y varianza igual a la media.

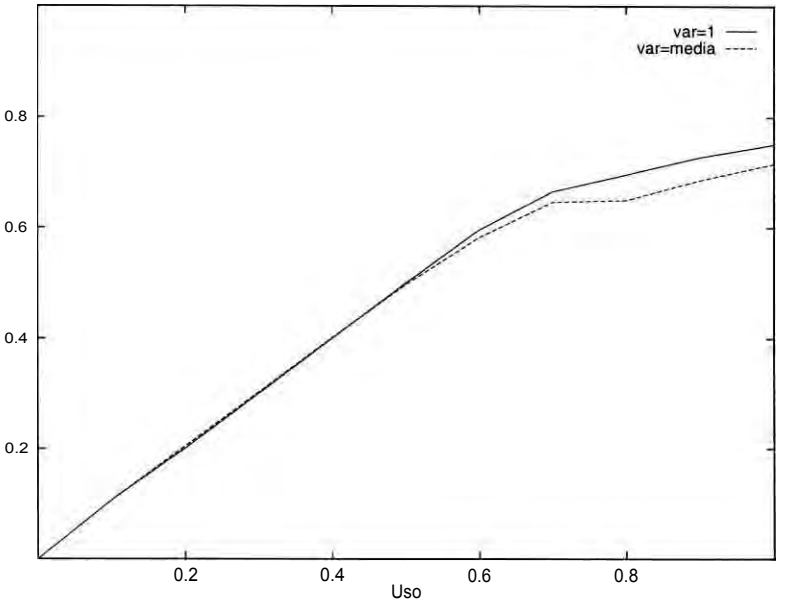


Figura 5.32: Comparación del flujo en el modelo del conmutador con buffers de longitud 4 en los puertos de entrada controlados por redes neuronales, bajo tráfico auto-similar con varianza 1 y varianza igual a la media.

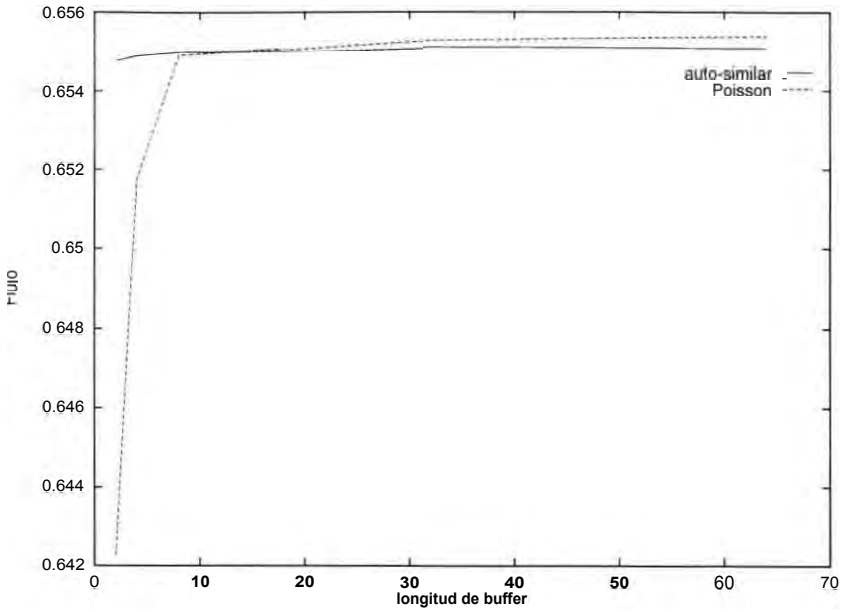


Figura 5.33: Retardo promedio de células versus longitud del buffer en un conmutador 4x4 con buffers en los puertos de entrada bajo tráfico modelado con procesos Poisson y auto-similares.

Para la estructura con un buffer por puerto de entrada, el límite está en 65.5%, para el esquema con buffers en los puertos de salida en 96.2% y para el esquema controlado por redes neuronales en 84.8%. El valor hallado para el esquema de conmutación controlado por redes neuronales propuesto es el máximo de una serie de intentos en un conmutador con un buffer de tamaño 512. Como se explicó anteriormente, el rendimiento para una misma secuencia de tráfico de entrada no siempre es el mismo debido a que depende del orden de actualización de las neuronas de la red, que es aleatorio.

Se demuestra entonces que, con cada estructura, se llega a un punto de flujo máximo que no puede ser rebasado incrementando el tamaño de los buffers. También se demuestra el efecto que tiene el tráfico auto-similar sobre estos resultados.

En las Figuras 5.36, 5.37 y 5.38 se muestra la probabilidad de pérdida de células. Mientras que para las estructuras de conmutación con buffers generales en los puertos de entrada, separados y controlados por redes neuronales, no puede mejorarse la tasa de pérdida de células, la estructura de conmutación si mejora drásticamente incrementando el tamaño de sus buffers hasta un punto crítico, en que es posible manejar sin pérdida todas las células que llegan.

En las Figuras 5.39, 5.40 y 5.41 se muestra el retardo promedio que sufren las células. Para las estructuras de conmutación con buffers generales en los puertos de entrada, separados y controlados por redes neuronales, la relación entre el tamaño del

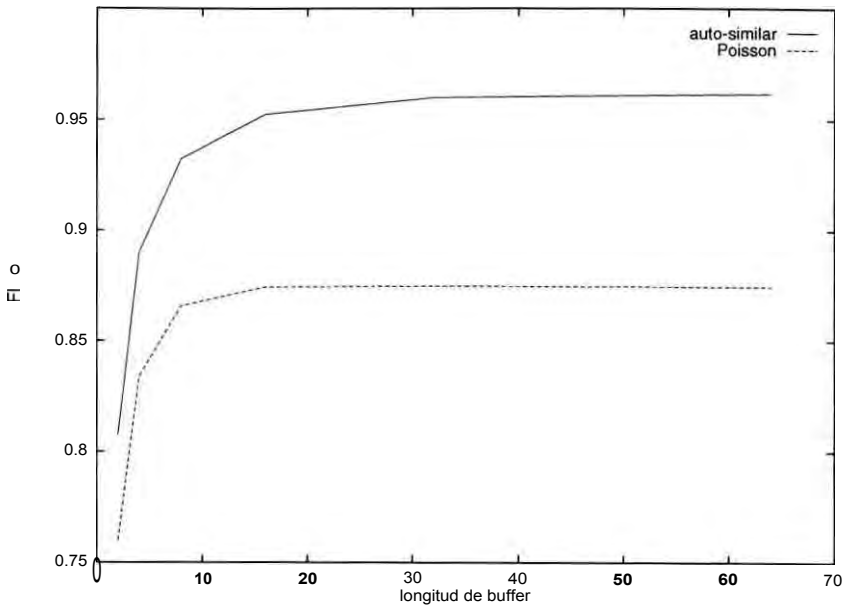


Figura 5.34: Retardo promedio de células versus longitud del buffer en un conmutador 4x4 con buffers en los puertos de salida bajo tráfico modelado con procesos Poisson y auto-similares.

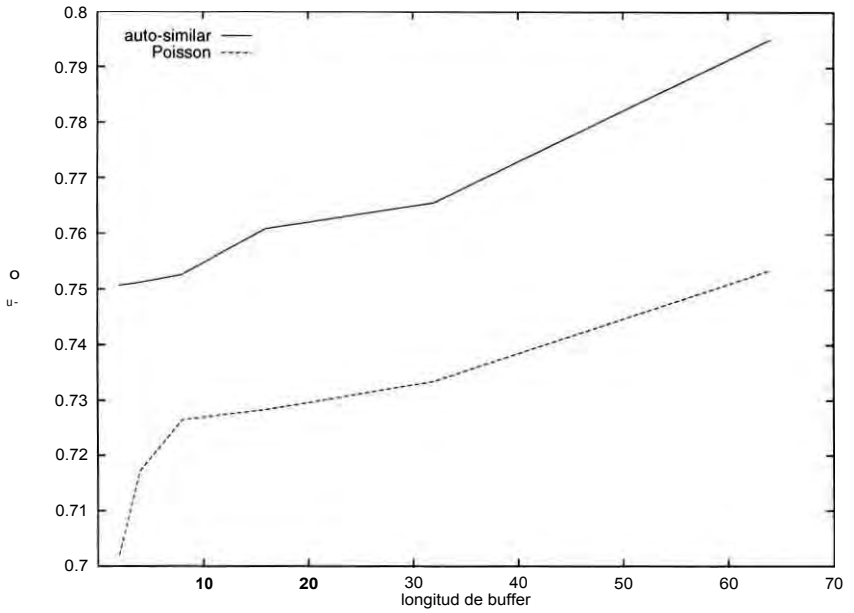


Figura 5.35: Retardo promedio de células versus longitud del buffer en un conmutador 4x4 controlado por redes neuronales bajo tráfico modelado con procesos Poisson y auto-similares.

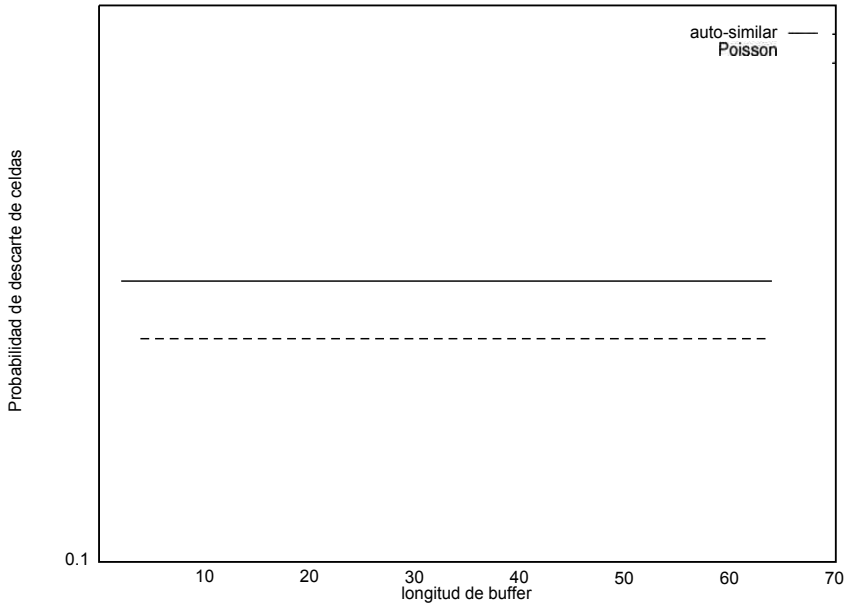


Figura 5.36: Probabilidad de pérdida de células versus longitud del buffer en un conmutador 4x4 con buffers en los puertos de entrada bajo tráfico modelado con procesos Poisson y auto-similares.

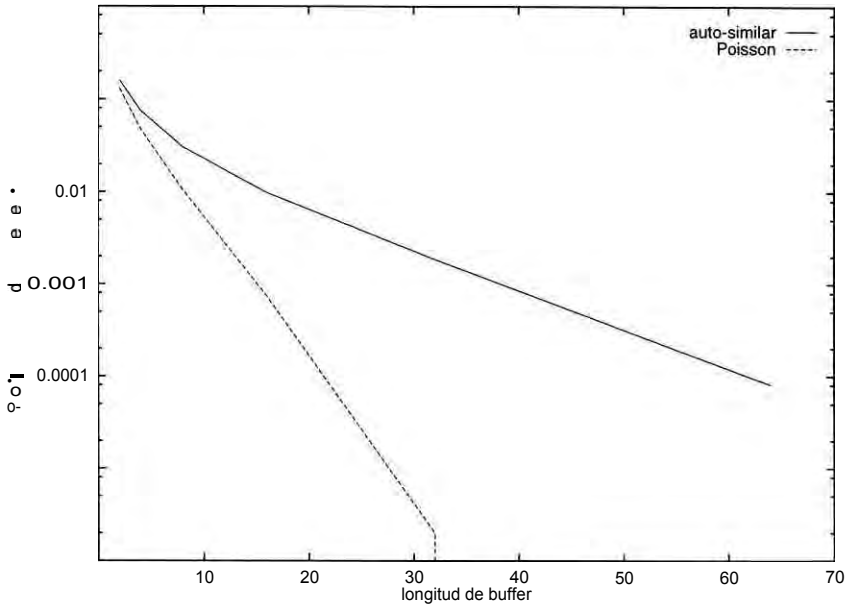


Figura. 5.37: Probabilidad de pérdida de células versus longitud del buffer en un conmutador 4x4 con buffers en los puertos de salida bajo tráfico modelado con procesos Poisson y auto-similares.

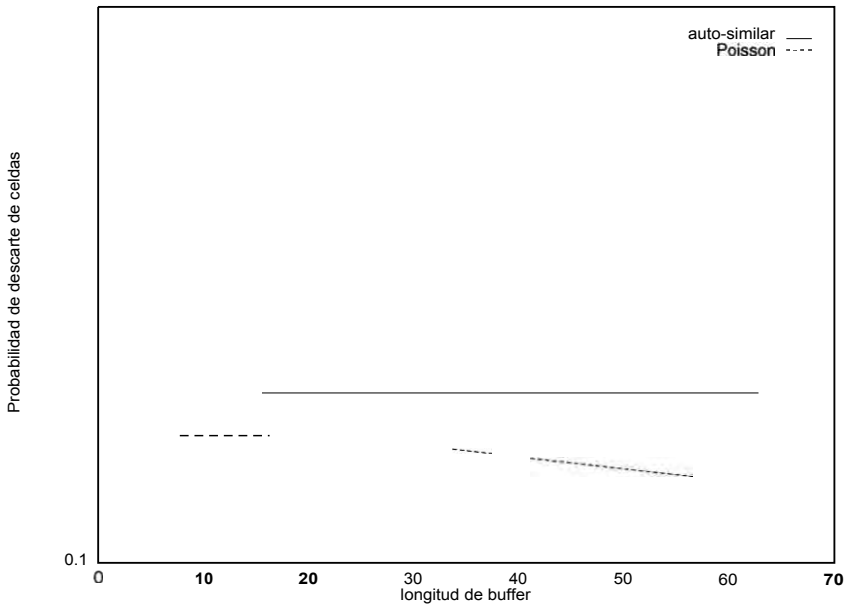


Figura 5.38: Probabilidad de pérdida de células versus longitud del buffer en un conmutador 4x4 controlado por redes neuronales bajo tráfico modelado con procesos Poisson y auto-similares.

buffer y el retardo promedio es proporcionalmente lineal. A mayor tamaño de buffer, mayor el tiempo promedio que tendría que esperar una célula en ser entregada. En cambio para la estructura con buffers en los puertos de salida, la relación llega a ser una constante.

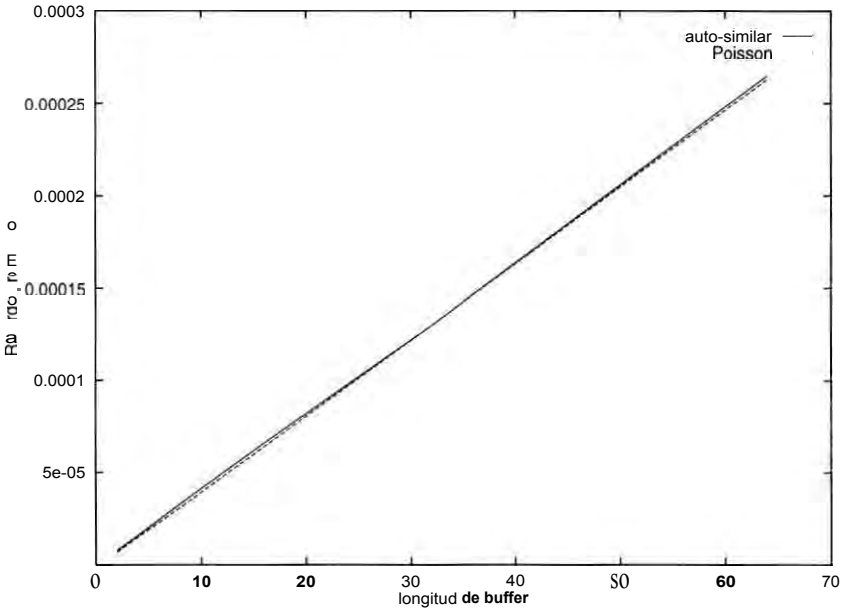


Figura 5.39: Retardo promedio de células versus longitud del buffer en un conmutador 4x4 con buffers en los puertos de entrada bajo tráfico modelado con procesos Poisson y auto-similares.

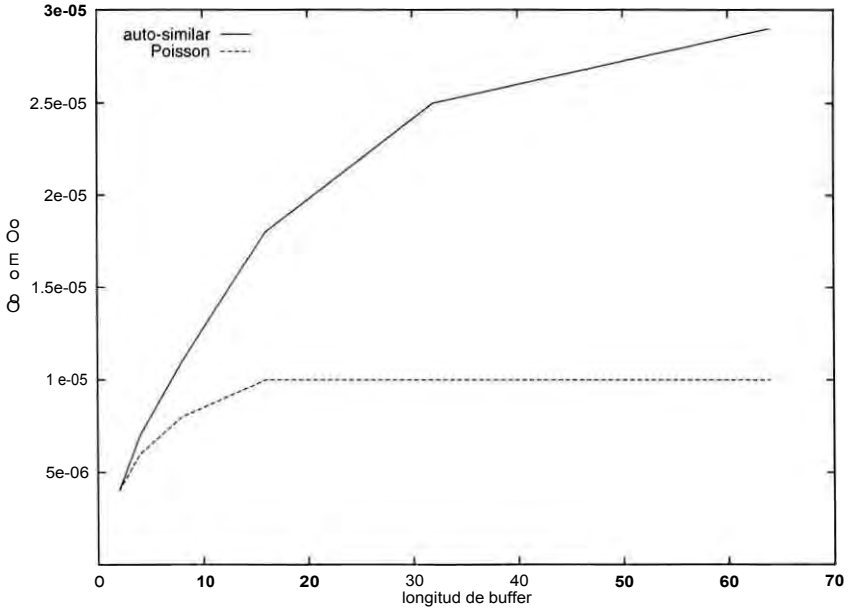


Figura 5.40: Retardo promedio de células versus longitud del buffer en un conmutador 4x4 con buffers en los puertos de salida bajo tráfico modelado con procesos Poisson y auto-similares.

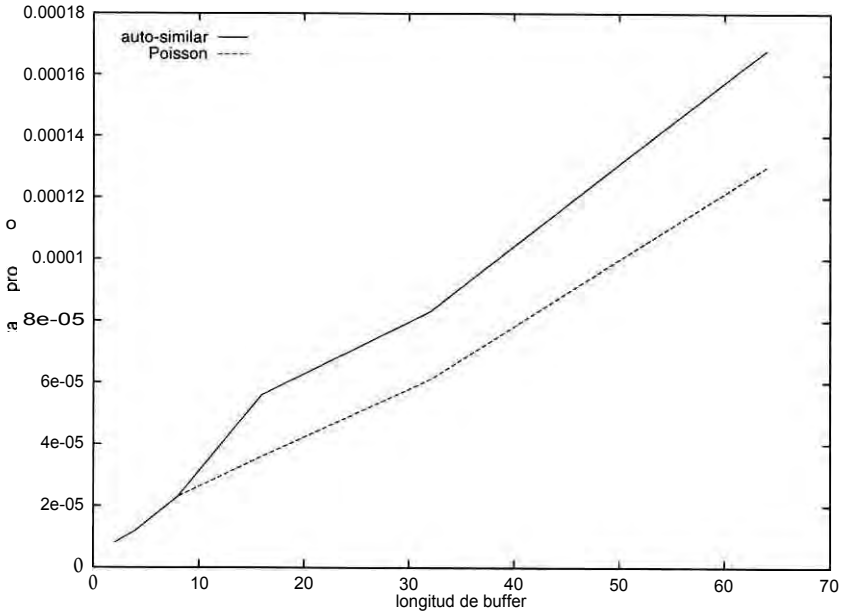


Figura 5.41: Retardo promedio de células versus longitud del buffer en un conmutador 4x4 controlado con redes neuronales bajo tráfico modelado con procesos Poisson y auto-similares.

Capítulo 6

Conclusiones

6.1 Resumen de los resultados y sus implicancias

Los tres principales objetivos de este trabajo de tesis han sido: resolver el problema del bloqueo originado por la primera célula en los buffers de entrada **FIFO** (bloqueo HOL) en un conmutador ATM crossbar para elevar su flujo de células, usando una versión mejorada de la red neuronal Hopfield originalmente propuesta por Marrakchi y Troudet; evaluar el comportamiento del flujo y otros parámetros importantes en la calidad de servicio de un conmutador como son la probabilidad de pérdida de células y el retardo promedio que sufren las células en los buffers del sistema; y evaluar el efecto de los modelos de tráfico auto-similares en el rendimiento del conmutador. Como se discutió anteriormente los modelos auto-similares permiten capturar con mayor precisión, que los procesos de Poisson y otros modelos tradicionales de tráfico, las características que posee el tráfico real.

Para lograr el primer objetivo, se modificó la función de penalización original de Marrakchi y Troudet que representa la energía computacional de una red neuronal Hopfield que controla la selección de células, para evitar el estado en la red que inhibe la conmutación interna, y así lograr un mayor flujo en el conmutador. El rendimiento del modelo construido se ha comparado con el del modelo de conmutación con buffers únicos por puerto de entrada y, con el de buffers en los puertos de salida que tiene un comportamiento ideal respecto a la conmutación interna.

En general, para altas cargas de tráfico, el uso de la red neuronal Hopfield para controlar la conmutación interna en el conmutador crossbar permite mejorar el flujo respecto al modelo general de buffers únicos por puerto de entrada hasta en un 22.8%.

Usando procesos auto-similares para modelar el tráfico de entrada, se encontró que los tres modelos exhiben un punto máximo de saturación, que no puede ser superado ampliando la capacidad de los buffers del sistema. Para el modelo de conmutación con buffers únicos por puerto de entrada, el punto de saturación del flujo está en 65.5% del flujo ideal, en el modelo de conmutación con buffers en los puertos de salida la saturación está en 96.2%. Nuestro modelo propuesto alcanza su

saturación en 84.8%.

Los resultados encontrados para la probabilidad de pérdida de células y el retardo promedio de células en el conmutador, cuyo estudio constituyen nuestro segundo objetivo, nos permiten afirmar lo siguiente:

1. El modelo propuesto tiene un mejor comportamiento respecto a la probabilidad de pérdida de células y al retardo promedio que sufren las células que el modelo de buffers únicos por puerto de entrada. En promedio, en nuestro modelo las células sufren un retardo 35% menor al que sufrirían en el esquema de conmutación con buffers únicos por puerto de entrada.

2. En general, el esquema con buffers en los puertos de salida tiene un mejor comportamiento frente a la probabilidad de pérdida y al retardo promedio de las células que los otros dos esquemas excepto para conmutadores con tamaños de buffers muy reducidos y bajo cargas muy ligeras de tráfico. Además, incrementando el tamaño de los buffers se mejora dramáticamente su rendimiento general.

3. En todos los casos, un incremento en el tamaño de los buffers disminuye la probabilidad de pérdida de células pero aumenta el tiempo promedio que pasan las células en los buffers. El mejoramiento que se logra en el flujo y en la probabilidad de pérdida de células en los modelos con buffers en los puertos de entrada es mínimo. Con el incremento en el tamaño de los buffers, el retardo promedio de las células en estos dos modelos crece drásticamente, lo que puede traer congestión en la red, debido a las retransmisiones que ocurren cuando se sobrepasa el tiempo máximo de espera de una célula.

Para cumplir el tercer objetivo, se comparó la eficiencia del conmutador usando modelos de tráfico asintóticamente auto-similares y Poisson. Se encontró que los procesos auto-similares recrean las ráfagas de paquetes que tiene el tráfico real, produciendo un mayor flujo de células que el modelo Poisson en los modelos de conmutadores estudiados. Esto se refleja también en una mayor pérdida de células y en una mayor ocupación promedio de los buffers cuando se usan este tipo de procesos en el modelamiento del tráfico de entrada.

Las consecuencias de este trabajo, basados en los resultados obtenidos, se resumen en los siguientes puntos:

1. Al presentar el tráfico real dependencia de amplio rango, las medidas de la calidad de servicio de un conmutador son subestimadas con el uso de los procesos de Poisson para modelar el tráfico de entrada. Podemos inferir que ocurrirá una situación similar con otros sistemas de conmutación de paquetes.

2. El modelo Poisson estima un tamaño de buffers para un dispositivo de comunicaciones inferior al que estiman los modelos que presentan dependencia de amplio rango. Debido a esto, los diseños basados en modelos de tráfico tradicionales requieren ser revisados y evaluados con modelos más precisos.

3. Las redes neuronales permiten elevar eficientemente el rendimiento en un conmutador ATM crossbar con buffers **FIFO** en los puertos de entrada, a pesar de que la red neuronal usada puede alcanzar mínimos estables no deseables desde el punto de vista de la conmutación interna óptima. Debido a su simplicidad y a su alta velocidad de procesamiento, el uso de las redes neuronales es muy recomendable para resolver problemas como el de la selección interna de células en un conmutador.

6.2 Recomendaciones para trabajos futuros

Existe un amplio espectro de investigación en este campo, dado que el análisis de redes con tráfico de entrada auto-similar está en su fase inicial de desarrollo, y que son necesarios mejores sistemas de conmutación para soportar el desarrollo de las redes actuales y futuras.

Unos pocos ejemplos de extensiones de este trabajo se mencionan a continuación:

1. Este estudio sólo utilizó tráfico balanceado, asumiendo que cada célula escoge su puerto de salida en forma aleatoria con igual probabilidad, lo que puede no ser válido en la realidad. Puede estudiarse el caso de tener enlaces de diferentes velocidades, y llegadas de células correlacionadas o ráfagas de células destinadas a una misma salida siguiendo alguna distribución de probabilidad.

2. Para la simulación se consideró que todas las líneas de entrada excitadas con tráfico auto-similar, tenían el mismo valor del parámetro de Hurst (0.79). Puede estudiarse el efecto de variar el parámetro de Hurst en todas o algunas líneas.

3. Aún no se ha obtenido una relación práctica entre la varianza y la media en el tráfico real. Puede determinarse esta relación analizando muestras de tráfico.

4. Las redes neuronales Hopfield presentan puntos de equilibrio no deseados dependiendo de la secuencia de actualización de sus nodos. Estos puntos de equilibrio crecen con el número de puertos del conmutador estudiado. Puede estudiarse el efecto del tamaño del conmutador y los puntos de equilibrio no deseados, y los efectos en el rendimiento del modelo.

5. Las funciones de penalización utilizadas para determinar los pesos de la red neuronal Hopfield no han intentado maximizar directamente el flujo del conmutador, sino que determinan que la red llegue a puntos de equilibrio que son permutaciones de la matriz unitaria. Puede investigarse funciones adicionales de penalización que aseguren un máximo flujo interno.

6. El retardo es un parámetro importante en el diseño de una red de comunicaciones. En esta tesis sólo se ha evaluado su comportamiento pero no se ha intentado mejorarlo. Pueden estudiarse mecanismos para mejorar el retardo promedio de las células dentro del conmutador.

Existe un amplio campo de investigación en las aplicaciones de las redes neuronales artificiales en los sistemas de comunicaciones por su simplicidad y alta velocidad de convergencia. Por otra parte, el descubrimiento relativamente reciente de la dependencia de amplio rango en el tráfico de las redes reales ofrece un amplio espectro de investigación en el área de análisis de redes. Los resultados obtenidos usando procesos con dependencia de amplio rango tendrán un gran impacto en el diseño de las redes de comunicaciones futuras.

Apéndice A

Modo de Transferencia Asíncrono

Las redes de telecomunicaciones del presente, tales como la red telefónica, la televisión por cable, etc., son redes dependientes y limitadas a un sólo tipo de servicio. Aún con recursos disponibles en la red, estos recursos no pueden ser aprovechados por otros servicios.

B-ISDN surge como una red independiente del servicio para grandes velocidades de transmisión, en que todos los servicios comparten los mismos recursos.

ATM es una técnica de conmutación y multiplexación para ser usada en B-ISDN [47], [48], donde la información del usuario es transmitida entre entidades usando pequeños paquetes de longitud fija (células ATM). Una célula ATM contiene 53 bytes, de los cuales 48 son información del usuario y 5 son de cabecera. En la cabecera está la información de como entregar la célula a su destino, particularmente en los campos VPI/VCI (en el identificador del camino virtual, y en el identificador de conexión virtual).

Las células para llegar a su destino deben de ser procesadas por nodos de conmutación, los que seleccionan la entrega en su estructura de conmutación interna en base a la información contenida en la cabecera de cada célula, para hacer llegar la célula desde un puerto de entrada a un puerto de salida.

En una red ATM, las conexiones entre los conmutadores se realizan con las interfases de nodos de red (NNI) y las conexiones entre un conmutador y un usuario se hace mediante el interfase de usuario (UNI).

A continuación sigue una breve descripción de la estructura de las células ATM y los métodos de conmutación más resaltantes.

A.1 Estructura de la cabecera ATM

Se utiliza una cabecera diferente en las células ATM para las interfases de red del usuario (UNI) y las interfases de nodos de red (NNI).

Los siguientes campos están presentes en la cabecera ATM:

- GFC (Generic Flow Control): es un campo de 4 bits que proporciona control

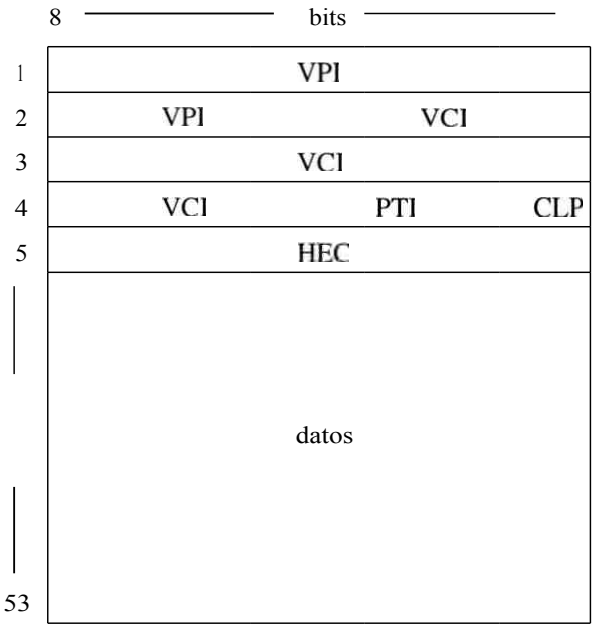


Figura A.1: Estructura de las células NNI ATM

de flujo en el UNI para el tráfico originado en el equipamiento del usuario dirigido a la red, pero no controla el tráfico en la otra dirección (es decir, el flujo red-usuario). Este campo no tiene uso dentro de la red, pero es usado por mecanismos de acceso para implementar diferentes niveles de acceso y prioridad. Por consiguiente, este campo es usado como parte del VPI en los NNIs generando una mejor capacidad de identificación de caminos. Se han definido dos tipos de operación para el campo GFC: acceso no controlado, y acceso controlado. El primero es usado en implementaciones prematuras y no tiene efecto en el tráfico que el usuario envía a la red. En el segundo caso, el flujo de células que el usuario envía a la red es controlado por el UNI.

- VCI/VPI (Virtual Connection Identifier/Virtual Path Identifier): El encaminamiento de células se realiza en cada nodo para cada célula que llega. Para esto se usan los campos VCI (8 o 12 bits) y VPI (16 bits) que contienen la información de encaminamiento de la célula.

- PTI: Son 3 bits que definen el tipo de información que contiene la célula.

- CLP: Es un campo de 1 bit usado para dar prioridad a la pérdida de células. Dada la naturaleza de multiplexación estadística que tienen las conexiones, es inevitable que ocurran pérdidas de células. Una célula con el bit CLP puesto a '1 'tienen mayor probabilidad de ser descartadas durante la congestión,

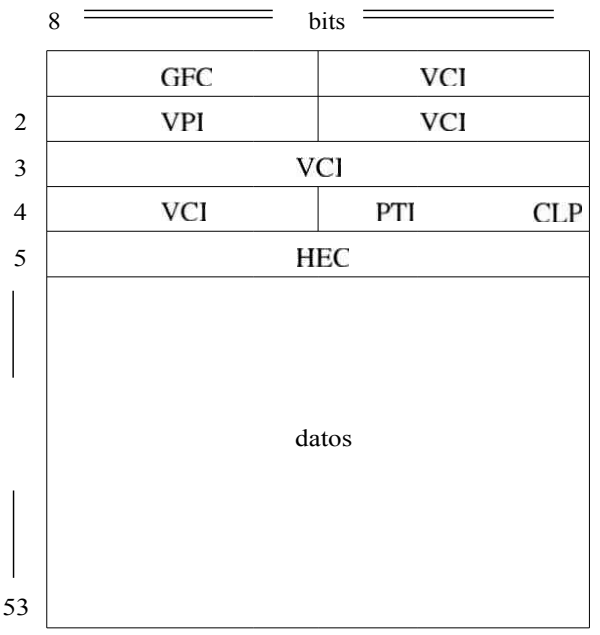


Figura A.2: Estructura de las células UNI ATM

mientras que las células con el bit CLP en '0'tienen mayor prioridad y no deben de ser descartadas de ser posible.

- HEC: Es usado principalmente para dos propósitos: descartar las células con cabeceras corruptas y para delinear células. Es un campo de 8 bits que se usa para verificar errores en la cabecera de la célula y provee corrección para un solo error (CRC32).

A.2 Modelo de Referencia para B-ISDN

La Figura A.3 muestra el modelo de referencia B-ISDN por la ITU-T. ATM ocupa la capa de modo de transferencia.

- Capa física: Se encarga del transporte de las células ATM entre dos entidades ATM. Adicionalmente esta capa garantiza dentro de un límite dado, la integridad de las células, y fusiona las células con información de control para la transmisión, de modo que se genere una contínua corriente de bits en el medio físico.

- Capa ATM: Provee de capacidad de transferencia de células. Sus funciones son independientes del medio físico usado. Esta capa provee de multiplexación de células, demultiplexación y encaminamiento usando los campos VPI y

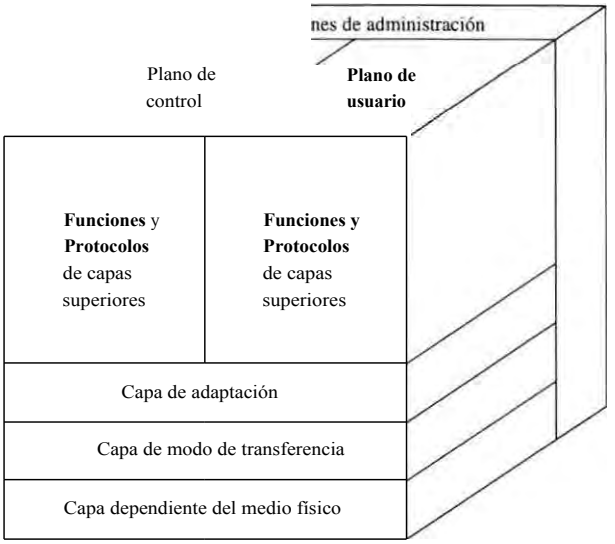


Figura A.3: Modelo de referencia B-ISDN

VCI de la cabecera de las células. Además, supervisa el flujo de células para asegurar que las conexiones mantendrán sus límites negociados en la fase de establecimiento de la llamada, y toma acciones correctivas en caso contrario. Esta capa también es responsable de mantener la integridad de la secuencia de células para cada fuente. En esta capa no se realizan retransmisiones por pérdida de células o por células corruptas.

- Capa de Adaptación (AAL): El rol de esta capa es proveer a cada clase de servicio, las funcionalidades requeridas para alcanzar una determinada calidad de servicio. La capa AAL soporta funciones de capas altas de los planos de usuario y control, y soporta conexiones entre interfases ATM y no ATM. La información que recibe AAL es segmentada o reunida, para ser insertada dentro de las células ATM. Las células recibidas por los AAL de la capa ATM son reensambladas para formar la información.

  Se han definido cinco clases de AAL por la ITU-T para diferentes tipos de servicio: AAL0 (para células puras), AAL1 (para secuencias de datos constantes), AAL2, (para secuencias de datos variables), AAL3/4 y AAL5 (para servicios orientados a la no conexión).

A.3 Conmutación de células ATM

A.3.1 Tipos de conmutadores ATM

Existen varios tipos de conmutadores propuestos en la literatura. A continuación se clasifican los principales esquemas [5], [7].

División de tiempo

En un conmutador basado en división del tiempo, todas las células fluyen a través de una única vía rápida de comunicación compartida por todos los puertos de entrada y salida. Pueden ser de dos tipos:

- Medio compartido

  En la arquitectura de medio compartido, las células que llegan son multiplexadas en un medio común, que es típicamente un bus o un anillo. La velocidad del medio es por lo general, mayor o igual a la suma de las tasas de transmisión de los enlaces entrantes. Así, un pequeño FIFO con capacidad de retener pocas células es suficiente para almacenar las células que llegan hasta que puedan acceder al medio.

  En esta arquitectura no puede ocurrir competencia de salida dado que dos o más células no pueden llegar a la vez a un mismo puerto de salida. Sin embargo, la tasa de llegada a un puerto de salida en particular, puede exceder el ancho de banda del enlace por un corto período de tiempo. Para esto, se utilizan buffers de salida de modo que se puedan almacenar las células que llegan más rápido de las que se pueden servir.

  A cada puerto de salida se le asigna una dirección única. Una vez que el enlace de salida de una célula que llega es determinado, se le agrega la dirección del puerto de salida a esta célula antes de que sea pasada al medio. Esta dirección es decodificada por cada interfase de puerto de salida y filtrada para determinar si la célula debe de copiarse al puerto de salida o no. Las células direccionadas a un puerto en particular son entonces copiadas al buffer de salida para ser enviado por el enlace de transmisión.

  Las arquitecturas de medio compartido soportan de manera natural el *broadcast* y *multicast*, y funcionan bien cuando la velocidad del medio es mayor o igual a la suma de las tasas de llegadas de los enlaces entrantes. Si aumentamos

el número de enlaces entrantes y/o aumentamos la velocidad de estos enlaces, este tipo de arquitectura se vuelve tecnológicamente no práctica dado que el medio se vuelve un *cuello de botella.* Por lo general, este tipo de arquitectura sólo soporta un número pequeño de puertos, sin embargo, pueden ser usados como elementos de conmutación para conmutadores de mayor tamaño en que cada unidad es conectada a las otras de acuerdo a alguna topología.

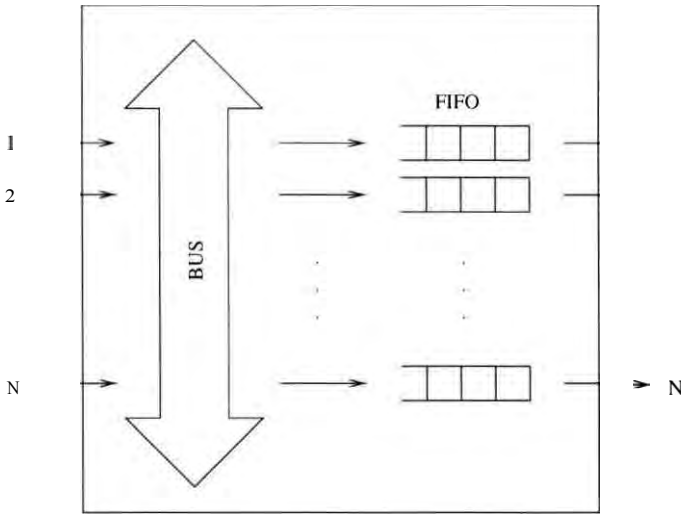


Figura A.4: Conmutador NxN con arquitectura de medio compartido

- Memoria compartida

Un conmutador de memoria compartida consiste en un único módulo de memoria con dos puertos, compartido por todos los puertos de entrada y salida. Las células entrantes son multiplexadas en una trama única y son grabadas en la memoria compartida. La memoria está organizada como colas lógicas, una para cada puerto de salida. Las células de las colas de salida también son multiplexadas en una trama única, leídas, demultiplexadas y transmitidas en las líneas de salida. En esta arquitectura el principal cuello de botella es el tiempo de acceso de la memoria para soportar tanto el tráfico de entrada como el de salida.

La memoria puede ser lógicamente organizada, ya sea como: compartida completamente o partición completa. En el primer caso, la memoria completa es compartida por todos los puertos de salida, y una célula que llega es descartada sólamente si la memoria está totalmente llena. Un límite superior es impuesto en el segundo caso para el número de células que esperan en la cola de cada puerto de salida, y una célula es descartada cuando este límite es alcanzado

en un puerto determinado aunque existe espacio libre en otras partes de la memoria.

La compartición completa da una menor probabilidad de pérdida de células que la partición completa, dado que utiliza más eficientemente la memoria, sin embargo, este esquema no es tan razonable cuando se presentan ráfagas de tráfico en un puerto en particular, lo que produciría una disminución del espacio disponible y una eventual degradación de la calidad de servicio en los otros puertos.

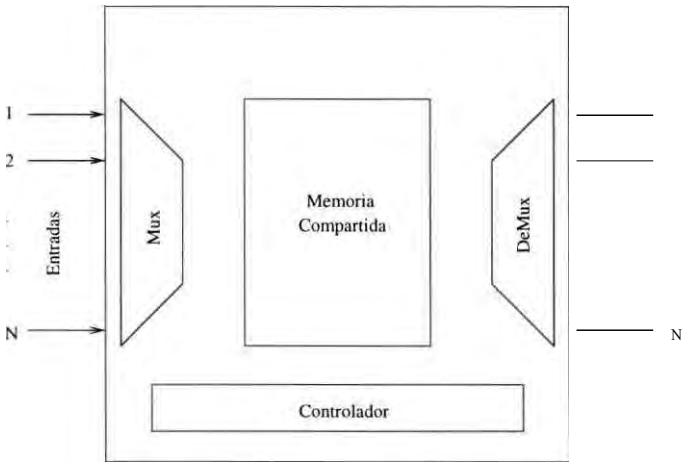


Figura A.5: Conmutador NxN con arquitectura de memoria compartido

División de espacio

Existen dos desventajas en las arquitecturas de conmutación de medio y memoria compartida. Primero, se requiere de multiplexación en la entrada y demultiplexación a la salida del conmutador, lo que restringe el crecimiento del conmutador para soportar mayor número de puertos. Segundo, el manejo de los buffers y las funciones de control son generalmente centralizadas, lo que incrementa la complejidad del conmutador.

En la conmutación por división de espacio, células múltiples de diferentes puertos de entrada pueden ser transmitidas a la vez en enlaces múltiples. Cada transferencia de células requiere del establecimiento de un enlace físico dedicado a través del conmutador de una entrada a una salida. Estos conmutadores permiten que el control este distribuido dentro del conmutador reduciendo su complejidad.

La unidad básica de construcción, en un conmutador por división de espacio, es un punto de cruce que puede estar habilitado o deshabilitado por la unidad de

control. Como se muestra en la Figura A.6 cada punto de cruce tiene dos entradas y dos salidas que permiten la activación simultánea de dos caminos separados.

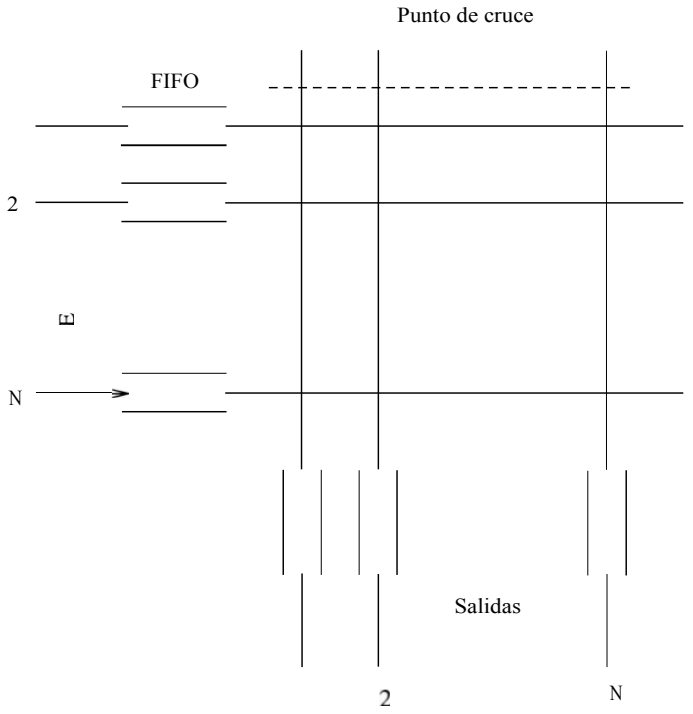


Figura A.6: **Conmutador cruzado** (crossbar).

La competencia por la salida en un punto de cruce ocurre cuando ambas líneas de entrada simultáneamente intentan acceder a la misma línea de salida. Si esto ocurre, una de las dos células en competencia logra acceso al puerto de salida, mientras que la otra puede ser descartada o puesta en un buffer temporalmente hasta que el puerto este disponible. Esta decisión se hace en base a una determinada política de **resolución de contenciones.**

Los buffers, para el almacenamiento temporal de células, pueden ser colocados en los puertos de entrada o dentro del punto de cruce. En cualquier caso, el tamaño del buffer es finito por los que su uso no resuelve totalmente el problema de la competencia por el puerto de salida. Particularmente, es posible que los buffers se llenen causando que las células nuevas sean descartadas dada la falta de espacio para almacenarlas.

Dada la pluralidad de caminos en el conmutador, se requiere de **una función de encaminamiento para seleccionar el camino a la salida adecuada. Existen dos técnicas de encaminamiento:** *self-routing* ***y*** *label-routing.*

En el enfoque self-routing **cada controlador de entrada en el conmutador** coloca una marca de encaminamiento o ***routing*** *tag* **a cada célula que llega usando** el mis-

mo mecanismo de tablas que usa para la translación de VCI [49]. Esta marca de encaminamiento especifica el número de puerto de salida al que debe de entregarse la célula. Esto permite que cada elemento de conmutación realice una decisión de encaminamiento muy rápida, simplemente mirando el contenido de la marca de encaminamiento. Cada célula llegará al puerto de salida requerido sin importar por cual puerto de entrada llegó. Este mecanismo es el más popular entre los conmutadores de división de espacio que se han diseñado.

En el caso de label-routing, el VCI es usado para indexar tablas de encaminamiento entre los elementos de conmutación. Este esquema de encaminamiento no se basa en las propiedades regulares de la red interconectada, de modo que arbitrariamente cualquier interconexión de red puede ser usada. Esta técnica permite una implementación simple de multicast, pero requiere de un número grande de tablas de encaminamiento y translación dentro del conmutador.

La interconexión de redes para un conmutador de división de espacio puede ser dividido en dos clases: redes de camino único y redes de caminos múltiples.

- Crossbar

  La Figura A.6 corresponde a un conmutador crossbar N por N, donde cada intersección corresponde a un punto de cruce. En general se utilizan N^2 puntos de cruces. La conexión desde cualquier puerto de entrada i a la salida j es realizada activando el punto de cruce correspondiente (i, *j)* en la matriz N x N. Se permiten hasta N conexiones simultáneas permitiendo pasar N células concurrentemente, de modo que se trata de un conmutador sin bloqueo interno. La mayor desventaja de este tipo de conmutador es que su complejidad crece con N^2. Además, existe un único camino entre cualquier entrada a cualquier salida lo que no lo hace tolerante a fallas [50].

  Una variación de este tipo de conmutador llamado *Knockout* se discute en [6].

- Banyan

  Las redes Banyan cubren una gran cantidad de redes que tienen un sólo camino entre cualquier entrada y cualquier salida. Actualmente el término "banyan"se aplica a la familia de redes self-routing construidas a partir de elementos de conmutación 2 x 2 con un sólo camino entre cualquier par entrada-salida.

  Están basadas en una topología de árbol donde cada entrada constituye la raíz del árbol que se ramifica sobre un número de elementos de conmutación

intermediarios y con los puertos de salida como hojas. Así, un bosque de N árboles que comparten todos los enlaces y elementos de conmutación a excepción de las raíces forman una red de conmutación Banyan de N por N. Existen muchas maneras de formar estos árboles que resultan en las diferentes topologías de las redes Banyan.

A.3.2 Problemas en los conmutadores

En un conmutador general ATM existen básicamente dos inconvenientes que pueden degradar significativamente el rendimiento [9]:

1. Bloqueo Interno: Puede ocurrir bloqueo interno cuando existen dos o más peticiones a algún componente interno del conmutador. Esto pasa cuando existen peticiones de múltiples puertos de entrada a una misma salida. El problema radica en que sólo una configuración puede ser realizada a la vez por el conmutador, es decir, sólo se puede aceptar una de las conexiones. Cuando surgen estos problemas, se utilizan algoritmos o técnicas para seleccionar cual de los puertos tiene prioridad, mientras que las demás células en los otros puertos deben de permanecer en los buffers hasta un siguiente ciclo.

   En general existen tres formas de resolver los conflictos internos:

   - Usando buffers internos.
   - Aumentando la velocidad interna con respecto a la externa.
   - Buscar caminos alternativos.

2. Bloqueo Externo: Ocurre cuando dos o más células llegan al mismo puerto de salida a la vez.

   Existen tres formas de resolver este problema:

   - Usar buffers externos de salida.
   - Colocar accesos múltiples a los puertos de salida.
   - Aumentar la velocidad externa de los puertos con respecto a la velocidad interna del conmutador.

A.3.3 Resolución de contenciones

Como se mencionó anteriormente, en un conmutador con conflictos internos, una de las formas de manejar la contención es colocar buffers en los puntos de conflicto. En un conmutador con buffers externos se requiere de un mecanismo de resolución de contenciones.

Pueden usarse tres métodos básicos, una vez que se detecta una contención en un conmutador con buffers externos: *backpressure*, desviación y pérdida.

En los diseños de conmutadores con buffers internos típicamente se usa back-pressure desde el punto de contención hasta los buffers de entrada.

Los diseños con buffers externos puros usan un mecanismo de pérdida donde las células que no pueden ser manejadas son descartadas en el punto de la contención.

Los conmutadores diseñados en base a recirculación usan tanto los mecanismos de desviación y pérdida. El mecanismo de desviación encamina las células que pierden la contención, por otro camino que no sea el más corto al destino especificado. Este camino puede ser un camino recirculante o un camino alternativo adelante del punto de contención.

La decisión de cuales células se aceptan y cuales se rechazan son realizadas por un árbitro. La decisión de arbitraje puede estar basada en las prioridades de la célula, el tiempo de permanencia de la célula o simplemente en forma aleatoria. El árbitro puede estar centralizado e implementado externamente al conmutador, o distribuido e implementado internamente dentro del mismo.

Se han propuesto tres mecanismos básicos de arbitraje: reservación de anillo, ordenamiento y arbitraje, y encaminamiento y arbitraje. En el método de reservación de anillo, los puertos de entrada están interconectados mediante un anillo que es usado para hacer las peticiones a los puertos de salida.

En los conmutadores que emplean un mecanismo de ordenamiento en el conmutador, todas las células que hacen peticiones a un mismo puerto de salida aparecerán adyacentes una de la otra luego del ordenamiento. Así, un mecanismo de arbitraje puede ser implementado comparando los destinos de cada célula con la de sus vecinos en la salida de la red de ordenamiento.

En el enfoque de encaminamiento y arbitraje, las células son encaminadas en el conmutador, y los árbitros detectan y resuelven las contenciones en los puntos de conflicto.

A.3.4 Estrategias en el uso de buffers en los conmutadores ATM

La clasificación más general para las diferentes estrategias en el uso de los buffers, es su colocación. Se tienen dos tipos: internos (dentro del conmutador) y externos (fuera del conmutador).

Buffers externos

Con esta arquitectura, las colas se localizan cerca a los puertos del conmutador a los que sirve. Cada puerto del conmutador puede verificar su cola y realizar una verificación de la carga para soportar control de congestión.

Además, cada cola puede ser separada fácilmente en múltiples clases de servicio, y cada puerto puede implementar una política de horarios basado en la ocupación de la cola para controlar mejor los requerimientos de retardo y pérdida de células para cada clase de tráfico.

La ausencia de colas dentro del conmutador ATM facilita el soporte de múltiples niveles de prioridad para las diferentes clases de tráfico.

Además, si se está empleando un conmutador de caminos múltiples, se debe de implementar un mecanismo de selección aleatorio entre los caminos posibles, para distribuir el tráfico en todo el conmutador. De este modo, el conmutador no necesita mantener un registro de la carga estimada del tráfico en cada enlace interno dentro del conmutador. Esto simplifica considerablemente el proceso de establecimiento de la llamada. Las células de la misma conexión virtual no sufrirán errores de pérdida de secuencia aunque tomen diferentes caminos dentro del conmutador dado que no existen buffers dentro del conmutador.

En un conmutador con este esquema en el uso de buffers, las operaciones de multicast para grandes anchos de banda, están generalmente soportadas con una estructura de copiado, colocada antes del conmutador. Esta estructura de copiado replica el número requerido de cada célula por el multicast, y el conmutador encamina a los puertos requeridos de salida. Con este mecanismo se requiere de una translación de etiquetas entre la estructura de copiado y el conmutador.

Buffers de entrada

Pueden usarse buffers en los puertos de entrada para evitar tanto el bloqueo interno como la competencia por el puerto de salida. En general, un buffer de

entrada es implementado como un FIFO. Una célula que no puede ser conmutada a su puerto de salida durante el ciclo actual ocupa la posición HOL (*Head of Line)* en el FIFO y reintenta durante los ciclos consecutivos. En este caso, las células detrás de la HOL están forzadas a esperar en la cola FIFO hasta que alcancen la posición HOL en la cola. En general si k células están direccionadas al mismo puerto de salida, se pueden emplear las siguientes políticas de selección [34]:

- Prioridad aleatoria: Se escoge uno de los k paquetes en forma aleatoria, donde cada paquete tiene igual probabilidad 1/k.
- Prioridad fija: Las N entradas tienen niveles de prioridad fija, luego el puerto de mayor prioridad gana la disputa.
- Prioridad cíclica: La prioridad para cada uno de los N puertos varía cíclicamente según el tiempo *(round robin)*.
- Selección por la cola más larga: El controlador envía la célula de la cola de entrada más larga.

En particular durante un ciclo, un camino del puerto de entrada al puerto de salida de células no HOL que esperan en la cola puede estar disponible, pero no puede ser usado dado que las células deben de esperar hasta convertirse en HOL.

Buffers de salida

Como se mencionó anteriormente, un conmutador sin bloqueo interno aún puede sufrir problemas por la competencia por los puertos de salida. Si no se utiliza buffers de salida, sólo una de las células en competencia podrá ser transmitida, y dependiendo del esquema usado, las otras células contendientes pueden ser descartadas (perdidas) o guardadas, ya sea en buffers de entrada o buffers internos. Con buffers de salida, todas las células que compiten por un puerto de salida son almacenadas en estas colas hasta que puedan ser transmitidas. Esto permite un mejor rendimiento que con buffers en la entrada.

En este esquema, los buffers de salida deben de ser capaces de recibir y manejar las células que llegan simultáneamente. Esta no es una tarea fácil, sobretodo porque la velocidad de los enlaces (y de conmutación) es elevada, y tiende a incrementarse por la demanda. Por ejemplo, un conmutador 8 por 8 con enlaces OC-3 requeriría una velocidad de memoria mayor a 1.2 Gbps. De aquí que cuando se usan buffers de salida, se compromete la facilidad de crecimiento del conmutador.

Buffers internos

Pueden usarse buffers en cada enlace interno para almacenar temporalmente las células contendientes por el mismo puerto de salida de un elemento de conmutación. Los buffers pueden ser colocados en los puertos de entrada, salida o pueden ser una memoria compartida para un elemento de conmutación 2 por 2.

Como se discutió previamente, los buffers en la entrada causan bloqueo por la HOL. Los buffers en la salida por otro lado requieren de un gran ancho de banda en la memoria (el doble de la velocidad de los enlaces internos cuando se usan elementos de conmutación 2 por 2).

La memoria compartida elimina el bloqueo HOL y no requiere la compleja interconexión adicional que necesitan los buffers de salida. Sin embargo, la memoria compartida no puede ser más implementada como un FIFO y requiere incluir en el elemento de conmutación una lógica de manejo de buffers que no es necesariamente menos compleja que implementar los buffers de salida, particularmente a nivel de chip.

Aunque el uso de buffers internos reduce la probabilidad de bloqueo interno, tiene varias desventajas. Dado que el número de células que llegan a un elemento de conmutación puede ser hasta $\sqrt{N}$ (esto para el caso de un conmutador Banyan), se necesitan buffers que soporten esta capacidad, lo que aumenta la complejidad del chip. También los buffers internos introducen retardos aleatorios dentro de la estructura del conmutador, causando una variación indeseable del retardo de las células en la red. Además, los buffers internos son más efectivos cuando el tráfico es uniformemente distribuido en los puertos de salida, y cuando la secuencia de células en un enlace de entrada es no correlacionado en cualquier enlace. Por lo general, ninguna de estas suposiciones se da en las redes ATM.

Aunque las células pertenecientes a diferentes cuadros son interpuestas, la secuencia de células en un enlace está altamente correlacionada, de modo que la probabilidad de que células consecutivas en un mismo enlace estén destinadas a un mismo puerto de salida es alta.

Por otro lado, si las colas están localizadas dentro de la estructura del conmutador, todas las células que pertenecen a la misma conexión virtual deben de usar el mismo camino a través del conmutador si se desea evitar errores de pérdida de secuencia [5]. Esto implica que si se está usando un conmutador de caminos múltiples, es necesario seleccionar un camino a través de la estructura del conmutador en la colocación de la llamada.

Para realizar esto, el conmutador debe de mantener un registro de la carga estimada del tráfico en cada enlace dentro del conmutador, y basar su decisión de aceptación de la llamada en base a la carga estimada a través de cada posible camino.

Esto complica significativamente el proceso de establecimiento de la llamada para un conmutador grande. La alterativa es dejar que cada célula tome cualquier camino a través del conmutador y luego ordene las células en la salida del mismo.

Dentro de un conmutador con buffers internos se puede soportar *multicast* sobre todo cuando se emplea *label-routing* [49].

Bibliografía

[1] Mischa Schwartz. Redes de Telecomunicaciones. Addison Wesley, 1994.

[2] Shawn A. Routhier, Raj Jain. Packet trains - measurements and a new model for computer network traffic. IEEE Journal on Selected Areas in Communications, (6):986-995, September 1986.

[3] Anshul Gupta, Vipin Kumar, Ananth Grama. Introduction to Parallel Computing. Design and analysis of Algorithms. The Benjamin/Cummings Publishing Company, Inc.

[4] Will E. Leland. On the self-similar nature of ethernet traffic (extended version). IEEE/ACM Transactions on Networking, 2(1):1-15, February 1994.

[5] Peter Newman. Atm technology for corporate networks. IEEE Communications Magazine, pages 90-101, apr 1992.

[6] Ellen E. Witte. A quantitative comparison of architectures for atm switching systems. Technical report, oct 1991.

Johan Karlsson, Daniel S0birk. Atm switching structures. a performance comparison.

[S] Mark J. Karol, Michael G. Hluchyj. Queueing in high-performance packet switching. IEEE Journal on Selected Areas in Communications, 6(9):1587-1597, December 1988.

[9] Liwei Li. Performance evaluation of atm switching fabrics. Master's thesis, Kent State University, aug 1995.

[10] Pravin Varaiya, Nicholas McKeown. Scheduling cells in an input-queued switch. Electronics Letters, Dec 1994.

[11] Nicholas William McKeown. Scheduling Algorithms for Input-Queued Cell Switches. PhD dissertation, University of California at Berkeley, Department of Electrical Engineering and Computer Science, 1995.

[12] Thomas E. Anderson, Nicholas McKeown. A quantitative comparison of scheduling algorithms for input-queued switches.

[13] T. Troudet, A. Marrakchi. A neural net arbitrator for large crossbar packet-switches. IEEE Transactions on Circuits and Systems, 36(7):1039-1041, jul 1989.

[14] Vern Paxson. Empirically derived analytic models of wide-area tcp connections. IEEE/ACM Transactions on Networking, 2(4):316-336, August 1994.

[15] Sally Floyd, Vern Paxson. Wide area traffic: The failure of poisson modeling. IEEE/ACM Transactions on Networking, 3(3):226-244, June 1995.

[16] Will E. Leland SIGComm 93 San Francisco C.A. On the Self-Similar Nature of Ethernet Traffic, September 1993.

[17] Walter Willinger. Self-similarity through high-variability: Statisticas analysis of ethernet lan traffic at the source level. Proccedings of the ACM/Sigcomm '95, 1995.

[18] I. Ziedins R. Alexander, N. Brownlee. Modelling self-similar network traffic. Technical report, University of Auckland, jul 1995.

[19] Ross Alexander. Monitoring, analysis and simulation of packed switched network traffic. Master's thesis, Auckland University, jan 1995.

[20] Mark W. Garrett, Walter Willinger. ACM SIGComm 94 London. Analysis, Modeling and Generation of Self-Similar VBR Video Traffic, September 1994.

[21] Murad S. Taqqu, Jan Beran, Robert Sherman. Long-range dependence in variable-bit-rate video traffic. IEEE Transactions on Communications, 43(2/3/4):1566-1579, feb/mar/apr 1995.

[22] Walter Willinger, Ashok Erramilli, Parag Pruthi. Self-similarity in high-speed network traffic measurementes: Fact or artifact? Technical report, jul 1995.

[23] Walter Willinger, Ashok Erramilli, Parag Pruthi. Recent developments in fractal traffic modeling. Bell Communications Research, Royal Institute of Technology (Sweden).

[24] Carey L. Williamson, Ying Chen, Zhiang Deng. A model for self-similar ethernet lan traffic: Design, implementation and performance implications.

[25] Wing Cheong Lau. Traffic Characterization, Quality of Service and System Design in Multimedia Broadband Neworks. PhD dissertation, The University of Texas at Austin, 1995.

[26] Jonathan L. Wang, Walter Willinger, Wing-Cheong Lau, Ashok Erramilli. Self-similar parameter estimation: A semi-parametric periodogram-based algorithm.

[27] Jonathan L. Wang, Wing-Cheong Lau, Ashok Erramilli. Self-similar traffic generation: The random midpoint displacement algorithm and its properties.

[28] Ioannis Lambadaris, Roger Kaye, Changheng Huang, Michael Devetsikiotis. Fast simulation for self-similar traffic in atm networks. IEEE ICC '95 Seattle, June 1995.

[29] Ashok Erramilli, Parag Pruthi. Heavy-tailed on/off source behavior and self-similar traffic. Royal Institute of Technology (Stockholm, Sweden), Bellcore (NJ USA).

[30] Parag Pruthi, Ashok Erramilli, R.P. Singh. Modeling packet traffic with chaotic maps. 1994.

[31] Vern Paxson. Fast approximation of self-similar network traffic. Wishful research result, Lawrence Berkeley Laboratory, University of California, April 1995.

[32] Nicolas D. Georganas, Yanhe Fan. On merging and splitting of self-similar traffic in high-speed networks. Multimedia Communications Research Laboratory. Dept EE University of Ottawa, 1995.

[33] Nicolas D. Georganas. Self-similar (fractal) traffic in atm networks.

[34] Surya Kiran Pappu Performance analysis of input and output queuing strategies of an atm switch under self-similar traffic. Master's thesis, Clemson University, aug 1996.

[35] **Patrick R.** Morin. **The** impact of self-similarity on network **performance** analysis. **Technical report,** Carleton **University, 1995.**

[36] **Walter** Willinger, Ashok Erramilli, Narayan Onuttom. **Experimental** queueing analysis **with** long-range dependence packet traffic. IEEE/ACM Transactions on **Networking, 4(3):209-223,** apr **1996.**

[37] Nicolas **D.** Georganas, Nikolai Likhanov, **Boris** Tsybakov. Analysis of an **atm buffer with** self-similar **("fractal") input** traffic. **Technical report,** Institute **for** Problems of **Information** Transmission. Russian Academy of **Science., 1995.**

[38] **Daniel V.** Wilson, **Will E.** Leland. High time-resolution measurement and analysis of lan traffic: Implications **for** lan interconnection.

[39] **Will E.** Leland, Henry **J. Fowler. Local** area network traffic characteristics, **with** implications **for** broadband network congestion **management.**

[40] **James** Gleick. Chaos, making **a** new science. Penguin **Books.**

[41] **William** Stallings. High-Speed Networks: TCP/IP and ATM Design Principles. Prentice **Hall, 1998.**

[42] **William Ferreira** Giozza, **José** Antão Beltrão, **Jacques** Philippe Sauve. Redes **Locales de Computadoras.** Protocolos de **Alto Nivel y Evaluación de Prestaciones.** McGraw-Hill, **1990.**

[43] Athanasios Papoulis. Probability, Random **Variables and** Stochastic Processes. McGraw-Hill, **1991.**

[44] Jacek M. Zurada. Introduction to Artificial Neural Systems. West Publishing **Company.**

[45] **Hussein T.** Mouftah, **James F.** Kurose. Computer-aided modeling, analysis, **and** design of communication networks. IEEE Journal on Selected Areas in **Communications, 6(1):130-145,** January **1988.**

[46] Hamdy **A.** Taha. **Investigación de** Operaciones. Alfaomega, 1991.

[47] Raj Jain, Kai-Yeung Siu. **A** brief overview of **atm: Protocols** layers, lan emulation, **and** traffic **management.**

[48] Dhadesugoor Vaman, Xiaomei Qian, Shukri Wakid. Architectures for bisdn networks: **A performance** study.

[49] Raif **O.** Onvural. **Asynchronous Transfer Mode** Networks: Performance Issues. Artech **House, 1994.**

[50] Daniel Gordon. The 14 by 14 and 16 by 16 switch fabrics design and implementation.