Adaptación del formalismo de "bolas y urnas" para la generación de cadenas de Markov de tiempo discreto †

Héctor Cancela Bosi
Investigación Operativa - InCo - Facultad de Ingeniería
Universidad de la República - Montevideo - Uruguay
y
IRISA - Rennes - Francia

May 31, 1996

Resumen

En este trabajo presentamos una extensión del formalismo de descripción de modelos de "bolas y urnas" al caso de sistemas cuyo parámetro de tiempo es discreto. Este formalismo está implementado en la biblioteca **BB**, *que provee diversos servicios básicos (generación de la cadena de Markov subyacente, almacenamiento y recuperación, simulación) para permitir al usuario la descripción y el análisis cuantitativo de la performance de un sistema markoviano.*

Como aplicación, presentamos el modelado y análisis de performance de una arquitectura de conmutador ATM, tema de actualidad en el marco del diseño de redes de alta velocidad. Comprobamos experimentalmente el hecho de que la performance de estos conmutadores depende, a carga fija, de las características del proceso de llegada de los paquetes de datos.

Palabras clave: Bibliotecas de modelado, cadenas de Markov, aplicaciones a telecomunicaciones.

†Trabajo financiado por la Acción U93E03 del Programa ECOS de cooperación científica entre Francia y Uruguay, y por el CONICYT (Uruguay)

1 Introducción

La complejidad de las nuevas generaciones de sistemas informáticos y de comunicación y el hecho de que numerosas actividades dependen de su buen funcionamiento ha ocasionado una atención creciente a las técnicas de modelado y de evaluación de este tipo de sistemas. Entre los conceptos más empleados encontramos el de *performance* (cantidad de servicio proporcionada por un sistema), y el de *seguridad de funcionamiento* ("propiedad que permite a los usuarios del sistema depositar una confianza justificada en el servicio que éste les proporciona" [Lap92]), este último de particular interés en el caso de los sistemas llamados críticos, a grandes riesgos humanos o económicos: sistemas de control de centrales nucleares, sistemas de navegación espacial o aérea, etc. Entre otras medidas de interés, se encuentran la *confiabilidad* (medida del suministro continuo de un servicio correcto), la *disponibilidad* (medida del suministro de un servicio apropiado respecto a la alternancia servicio apropiado – servicio inapropiado), la *mantenibilidad* (medida del tiempo hasta la reparación a partir del último fallo acaecido), y la *performabilidad* (medida combinada performance – seguridad de funcionamiento).

Para estudiar estas medidas los sistemas son modelados como procesos estocásticos, típicamente en un cuadro markoviano. Varias técnicas numéricas han sido desarrolladas para realizar cálculos sobre procesos markovianos. El problema principal es que el tamaño del espacio de estado de los procesos crece proporcionalmente a la complejidad y el nivel de detalle del modelo, lo que hace imposible que el usuario especifique directamente el proceso subyacente. Esta tarea requiere un enfoque de modelado sistemático, con el empleo de interfaces de alto nivel, que ofrezcan una representación compacta del sistema estudiado. Lo ideal es que estas interfaces permitan al usuario describir el sistema estudiado a través de los mismos conceptos usados, por ejemplo, para diseñarlo.

Dentro del cuadro markoviano, encontramos dos grandes familias de modelos, correspondiente al empleo de un parámetro de tiempo continuo o discreto respectivamente. En el caso de los sistemas cuya evolución se efectúa según un parámetro de tiempo continuo, existen diversas herramientas basadas en formalismos que permiten dar una representación compacta del sistema, y generar posteriormente el proceso de Markov subyacente. Podemos encontrar, entre otros, herramientas basadas en las redes de Petri ([Chi91] [SO93], [BAA+93] [DTGN84], las redes de filas de espera [KS91], los modelos "bolas y urnas" [Ste91], [Can93], [CR94], los sistemas a eventos markovianos [Gra91], varios formalismos lógicos [CF86] o mixtos orientados a objetos [Bou93], [BdSM91], y herramientas basadas en lenguajes específicos del estilo "máquinas y reparadores" (machine-repairmen model, MRM) [BHL+93].

En cambio, el caso en el cual el sistema es modelado de acuerdo a una evolución en tiempo discreto ha sido mucho menos tratado en la literatura. Recientemente se ha reconocido que el empleo de un parámetro de tiempo discreto es muchas veces necesario para modelar de forma más precisa el comportamiento tanto a nivel del hardware como del software. De particular interés es el caso de la modelización de redes de transmisión a alta velocidad basadas sobre ATM, tema de actualidad y en rápida expansión. Esto ha motivado algunos trabajos recientes que presentan formalismos en tiempo discreto (autómatas finitos [PA91], redes de Petri estocásticas en tiempo discreto [Cia95], redes de filas de espera en tiempo discreto [Vu 93] [Kob83] [Hun83]) utilizados para estudiar el comportamiento de dichos sistemas. Sin embargo, la mayoría de las herramientas de modelado hoy en día existentes no incorporan por el momento estos formalismos y se limitan al tratamiento del caso de parámetro de tiempo continuo.

En este trabajo presentamos una extensión al caso de parámetro de tiempo discreto del formalismo "bolas y urnas", empleado previamente para el modelado de sistemas en tiempo continuo. Este formalismo permite descripciones más compactas que aquellas basadas en cadenas de Markov, conservando su misma generalidad (las clases de modelos expresables son equivalentes). Dicha extensión ha sido incorporada a la biblioteca de generación de modelos **BB** [CR94], que previamente sólo soportaba modelos en tiempo continuo. Esta biblioteca está compuesta por un conjunto de clases y métodos para la construcción y la evaluación de modelos de "bolas y urnas". En particular es posible hallar la cadena de Markov equivalente a un modelo BB específico dada por su espacio de estado y su matriz de transición), lo que permite usar la batería de métodos analíticos existentes; también es posible emplear métodos de simulación, en caso de que el tamaño del modelo haga poco práctica o imposible la primer alternativa. Como ejemplo de utilización de la biblioteca, hemos tomado un modelo en tiempo discreto de la competencia por recursos para una arquitectura específica de un conmutador ATM.

El trabajo está organizado de la forma siguiente: en la próxima sección, presentamos el formalismo de bolas y urnas en tiempo discreto. La implementación del formalismo en la biblioteca BB es discutida en la Sección 3. En la Sección 4 desarrollamos un ejemplo concreto de aplicación de la biblioteca al modelado y la evaluación de la performance de un conmutador ATM. La Sección 5 está dedicada a las conclusiones y las perspectivas de trabajo futuro.

2 Formalismo de bolas y urnas en tiempo discreto

2.1 Presentación informal y ejemplo

El formalismo llamado de *bolas y urnas* ("balls and buckets" en [Ste91]) ha sido propuesto para especificar de una forma compacta cadenas de Markov, a través de un conjunto reducido de reglas muy sencillas que permiten construir automáticamente el espacio de estado y las transiciones de la cadena.

En este formalismo el estado del sistema es representado par un conjunto de L urnas, cada una de las cuales contiene una cantidad $n[i]$ de bolas, con $0 \leq n[i] \leq nmax[i]$, $i = 1, 2, \ldots, L$. El comportamiento dinámico del sistema está dado a través del tránsito de las bolas entre las diferentes urnas. El movimiento de una bola desde una urna i hacia una urna j es llamado una *transferencia* de i a j, y puede realizarse sólo si $n[i] > 0$ y $n[j] < nmax[j]$ (condición de habilitación); el resultado de una transferencia de i a j es que $n[i]$ es decrementado en 1 y $n[j]$ es incrementado en 1. En una unidad de tiempo, una urna puede ser el origen de a lo sumo una transferencia, pero puede ser el destino de varias. Las transferencias se efectúan de acuerdo a probabilidades que pueden depender del estado global del sistema, es decir, del número de bolas en cada una de las urnas; es claro que la suma de las probabilidades de las transferencias que tienen como origen la misma urna debe ser siempre menor o igual a 1. A continuación de la transferencia puede ocurrir, con una cierta probabilidad, una modificación arbitraria del contenido de las urnas, que se efectúa de forma instantánea. Como durante una misma unidad de tiempo varias transferencias (con distintas urnas de origen) pueden efectuarse, consideraremos que estas transferencias (y las modificaciones consecuentes) se realizan según un orden de prioridad (que por simplicidad será idéntico a la posición de la urna de origen). En ese caso una transferencia sera posible si la condición de habilitación era verdadera al comienzo del ciclo y sigue siéndolo luego de considerar el efecto de las transferencias con mayor prioridad. Una *transición* es una modificación del estado del sistema, resultado de la composición de todas las transferencias y modificaciones instantáneas que tienen lugar durante una unidad de tiempo.

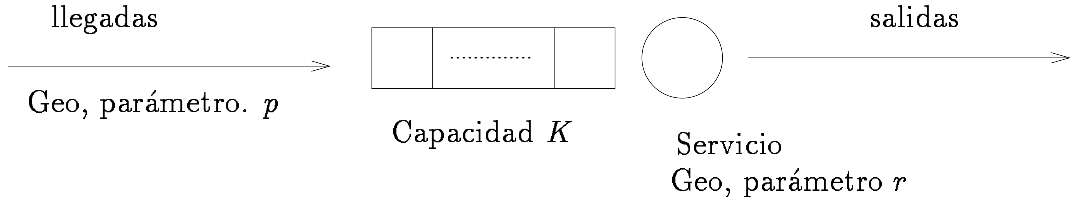

Figure 1: Fila de espera Geo/Geo/1/K

Como ilustración, consideremos la fila de espera en tiempo discreto Geo/Geo/1/K que aparece en la Figura 1. En cada unidad de tiempo puede llegar un nuevo cliente con probabilidad p (la ley de las llegadas es Bernouilli de parámetro p). El tiempo de servicio en la estación es una variable aleatoria de ley geométrica de parámetro r (notaremos $\bar{p} = 1 - p$, y $\bar{r} = 1 - r$). La fila de espera tiene una capacidad de K clientes. Suponemos que en este sistema "las llegadas ven las salidas", es decir que primero se efectúan las salidas del sistema y luego las llegadas al mismo.

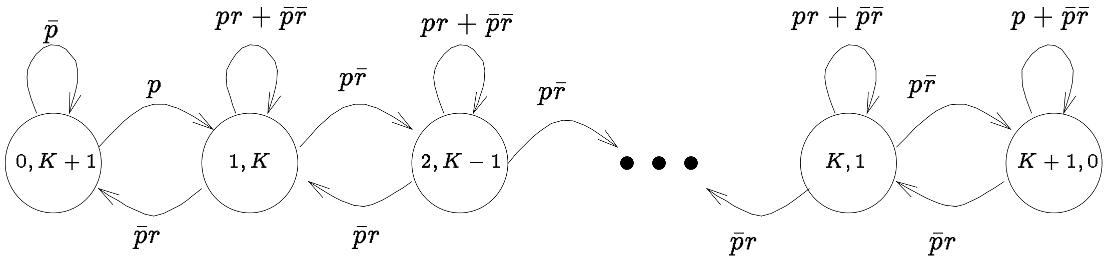

Figure 2: Cadena de Markov correspondiente a la fila Geo/Geo/1/K

Un modelo bolas y urnas equivalente es el siguiente:

- $L = 2$: dos urnas, la primera correspondiendo a la cantidad de clientes en el sistema, y la segunda representando el exterior del sistema;
- $nmax[1] = K + 1$ (capacidad del sistema);
  $nmax[2] = K + 1$;
- transferencia correspondiente a la llegada de un cliente: de la urna 2 a la urna 1, con probabilidad p;
- transferencia correspondiente a la finalización del servicio en la estación: de la urna 1 a la urna 2, con probabilidad r;

- estado inicial: $(0, K+1)$.

Las bolas de la primer urna corresponden a los clientes presentes en la estación 1 (incluyendo su fila de espera). La segunda urna es utilizada para representar el exterior del sistema, que provee las llegadas. En la Figura 2, podemos observar la cadena de Markov subyacente a este sistema. El espacio de estado $\mathcal{E}$ de esta cadena es

$$\mathcal{S} = \{(0, K+1), (1, K), (2, K-1), (3, K-2), \ldots, (K, 1), (K+1, 0)\}$$

y su matriz de transición es M, con

$$M = \begin{array}{cc}
 & \begin{array}{ccccccccc} 0 & 1 & \ldots & i-1 & i & i+1 & \ldots & K & K+1 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ \vdots \\ i-1 \\ i \\ i+1 \\ \vdots \\ K \\ K+1 \end{array} &
\left(\begin{array}{ccccccccc}
\bar{p} & p & 0 & & & & & & \\
r\bar{p} & rp+\bar{r}\bar{p} & \bar{r}p & 0 & & & & & \\
0 & \ddots & \ddots & \ddots & 0 & & & \mathbf{0} & \\
 & 0 & r\bar{p} & rp+\bar{r}\bar{p} & \bar{r}p & 0 & & & \\
 & & 0 & r\bar{p} & rp+\bar{r}\bar{p} & \bar{r}p & 0 & & \\
 & & & 0 & r\bar{p} & rp+\bar{r}\bar{p} & \bar{r}p & 0 & \\
 & & & & 0 & \ddots & \ddots & \ddots & 0 \\
 & & \mathbf{0} & & & 0 & r\bar{p} & rp+\bar{r}\bar{p} & \bar{r}p \\
 & & & & & & 0 & r\bar{p} & \bar{r}+rp
\end{array}\right)
\end{array}$$

2.2 Definición formal

Formalmente, una descripción genérica bolas y urnas en tiempo discreto está dada por una quíntupla $(L, \overrightarrow{nmax}, \vec{e_0}, \lambda, \Gamma)$, donde:

- $L \geq 1$ es el número de urnas;
- $\overrightarrow{nmax} \in \mathbb{N}^L$ es el vector de las capacidades máximas de las urnas;
- $\vec{e_0} \in \mathbb{N}^L$, $0 \leq e_0[i] \leq nmax[i]\ \forall i \in 1..L$, es el estado inicial;
- $\lambda \in (\mathbb{N}^L \rightarrow [0,1])^{L \times L}$ es la matriz de las probabilidades de las transferencias (en función del estado actual). Para cada $\vec{n}$, i, j, tal que $0 < n[i]$ y $n[j] < nmax[j]$ (o $n[j] = nmax[j]$ si $i = j$),

  $$\lambda[i,j](\vec{n}) = \text{probabilidad de la transferencia de una bola de la urna } i \text{ a la urna } j;$$

  Si $n[i] = 0$ o si $n[j] = nmax[j]$, por definición $\lambda[i,j](\vec{n}) = 0$. Además, tenemos siempre $\sum_j \lambda[i,j](\vec{n}) \leq 1$, $\forall i$, $\forall \vec{n}$.

- $\Gamma \in \left(\mathbb{N}^L \rightarrow (\mathbb{N}^L \times [0,1])^*\right)^{L\times L}$ es la matriz de las modificaciones instantáneas; para cada $\vec{n}$, resultado de una transferencia de i a j, Γ da el conjunto de las modificaciones instantáneas posibles:

$$\Gamma[i,j](\vec{n}) = \begin{cases} \{(\vec{n},1)\} & \text{si no hay ninguna modificación instantánea;} \\ \left\{(\overrightarrow{m_1},p_1),\ldots,(\overrightarrow{m_H},p_H),(\vec{n},1-\sum_1^H p_h)\right\} & \text{donde } \overrightarrow{m_1},\ldots,\overrightarrow{m_H} \text{ son los estados resultado} \\ & \text{de modificaciones instantáneas a partir de } \vec{n} \\ & \text{(luego de una transferencia de la urna } i \text{ a la} \\ & \text{urna } j\text{). La probabilidad que la modificación} \\ & \text{instantánea sea hacia } \overrightarrow{m_h} \text{ es } p_h. \end{cases}$$

Es posible ver que este formalismo es tan potente (en el sentido de la clase de los modelos que se pueden expresar) como el de las cadenas de Markov finitas de tiempo discreto. En efecto, cualquier cadena de Markov de espacio de estado finito puede ser descrita como un modelo bolas y urnas de tiempo discreto: si el proceso $X(t)$ posee espacio de estado $\mathcal{S} = \{a_1, a_2, \ldots, a_L\}$ y matriz de transición $M = (\lambda[i,j])$, alcanza con tomar L urnas y transferencias entre las urnas i y j con probabilidades $M[i,j]$. Una única bola se desplaza entre las urnas. El recíproco es trivialmente verdadero, dado que todo modelo bolas y urnas en tiempo discreto verifica la propiedad de Markov (ausencia de memoria).

Si generalizamos el formalismo permitiendo la existencia de urnas con una cantidad de bolas no acotada, encontramos la clase de las cadenas de Markov de espacio de estado numerable y tiempo continuo. En este caso es claramente imposible generar el espacio de estado y la matriz de transición. Pero es posible utilizar otros métodos de cálculo de medidas de seguridad de funcionamiento, que solo acceden a una parte del espacio de estado y obtienen resultados aproximados (cotas, simulación).

Si volvemos al ejemplo de la fila Geo/Geo/1/K, la descripción formal del modelo bolas y urnas correspondiente es:

- $L = 2$;
- $\overrightarrow{nmax} = (K+1, K+1)$;
- $\vec{e} = (0, K+1)$;
- $\lambda[i,j](\vec{n}) = \begin{cases} p & \text{si } i = 2 \text{ y } j = 1; \\ r & \text{si } i = 1 \text{ y } j = 2; \\ 0 & \text{en los demás casos.} \end{cases}$
- $\Gamma[i,j](\vec{n}) = \{(\vec{n},1)\}$en todos los casos.

3 Implementación

3.1 Biblioteca BB

La biblioteca **BB**, presentada en [CR94] y [Can93], está compuesta por un conjunto de clases y métodos, implementados en C++, que a partir de un modelo de seguridad de funcionamiento de un sistema construyen el proceso de Markov a utilizar ulteriormente para el cálculo de las medidas deseadas. La biblioteca está organizada en dos niveles. El primero es un núcleo genérico que no depende del formalismo elegido para describir el sistema. El segundo nivel consiste en una especialización de las clases del núcleo para el caso particular en el que la descripción del sistema se realiza en el formalismo de *bolas y urnas*.

El núcleo de la biblioteca proporciona servicios de generación, despliegue, almacenamiento y simulación de modelos. A este nivel, la descripción del sistema debe incluir una función que proporcione todas las transiciones posibles a partir de un estado con las probabilidades asociadas, y también el estado inicial del sistema. El programa construirá el conjunto $\mathcal{S}$ de los estados que pueden ser alcanzados a partir del estado inicial y el proceso de Markov equivalente X de espacio de estado $\mathcal{S}$ y matriz de transición M. El algoritmo implementado construye la matriz de las transiciones del sistema en forma incremental, a través de la búsqueda de los estados accesibles a partir del estado inicial. Esta técnica tiene diversas ventajas. Por ejemplo, solamente los estados que pueden ser alcanzados a partir del estado inicial estarán efectivamente representados en la matriz de transición resultante (lo que representa muchos menos elementos que el producto cartesiano de los estados de los objetos individuales que constituyen el sistema, tomado como espacio de estado en algunos formalismos de modelado).

El segundo nivel contiene toda la información necesaria para almacenar y manipular un estado de un sistema descrito en términos de *bolas y urnas*. Esto incluye la implementación de métodos virtuales definidos en el nivel precedente. El usuario especifica su modelo a traves de la quíntupla $(L, \overrightarrow{nmax}, \overrightarrow{e_0}, \lambda, \Gamma)$. En particular, las funciones λ y Γ se especifican como procedimientos en C++, lo que permite una gran flexibilidad de especificación. El sistema es representado por un objeto de la clase $babSystem$, cuyos métodos permiten un acceso transparente a todos los servicios de la biblioteca.

3.2 Algoritmo de generación de transiciones

Para que ambos niveles de la biblioteca puedan interactuar, es necesario que exista un algoritmo que a partir de la descripción de nivel bolas y urnas de un sistema pueda proporcionar todas las transiciones activas en cualquier estado $\overrightarrow{e}$, con sus probabilidades respectivas. Es decir, a partir de la descripción del sistema en términos de transferencias y modificaciones instantáneas, es necesario reconstruir el comportamiento de bajo nivel del mismo.

En la biblioteca hemos implementado el algoritmo siguiente, que almacena en la lista Δ dicha información a partir de las funciones λ y Γ que forman parte de la descripción del modelo bolas y urnas. Para poder obtener todas les combinaciones posibles de transferencias y modificaciones instantáneas, se consideran de forma secuencial todas las urnas, y se aplica en cada paso todas las transferencias cuyo origen es la urna fijada sobre todos los estados que fueron generados en las etapas previas, junto con las modificaciones instantáneas correspondientes. De esta manera encontramos nuevos estados accesibles (con sus probabilidades). Los estados originales siguen estando presentes, pero su probabilidad debe ser ponderada por la probabilidad de que los mismos no sean origen de ninguna transferencia durante la etapa considerada. Para la etapa siguiente, se incorporan los nuevos estados en el conjunto de origen.

Seudocódigo del algoritmo de generación

```
Δ := {(e⃗, 1)}
repetir para  todas  las urnas i
    Δ_i := ∅
    repetir para todos los pares (e⃗_k, p_k) en Δ
         Δ_ik := ∅
         repetir para todas las transferencias i → j habilitadas
                      en el estado e_0 y en el estado e_k
              p_ij := λ[i,j](e⃗_k)
              si (p_ij > 0)
                   e⃗*_k := e⃗_k
                   e*_k[i]--
                   e*_k[j]++
                   Δ_ikj := Γ[i,j](e⃗*_k)
                   repetir para todos los pares (m⃗_l, p_l) en Δ_ikj
                        p_l := p_l × p_ij × p_k
                   fin repetir
                   Δ_ik := Δ_ik ∪ Δ_ikj
              fin si
         fin repetir
         p_k := p_k × (1 − Σ_j p_ij)
         Δ_i := Δ_i ∪ Δ_ik
    fin repetir
    Δ := Δ ∪ Δ_i
fin repetir
```

4 Modelo de un conmutador ATM

4.1 Arquitectura del conmutador

Uno de los desafíos actuales en el área de las telecomunicaciones es la integración de la transmisión y ruteo de diferentes tipos de servicios (voz, datos, video) en redes de alta velocidad. Una solución que aparece entre las favoritas es el empleo de redes basadas en ATM (Asynchronous Transfer Mode). En estas redes la información está segmentada en paquetes o celdas de tamaño fijo. El tiempo está particionado en intervalos unitarios (time slots), cada intervalo es igual al tiempo necesario para transmitir un paquete. Para analizar las redes basadas en ATM, una componente fundamental es la performance de los conmutadores utilizados. Las arquitecturas de conmutadores ATM pueden clasificarse en arquitecturas de memoria compartida (shared-memory), de medio compartido (shared-medium), y de división espacial (space-division).

El sistema que estudiaremos es un modelo de conmutador ATM de arquitectura de medio compartido, presentado en [ZP94], que puede verse en la Figura 3. En este modelo, el conmutador debe rutear los paquetes que recibe de las N fuentes hacia los N destinos posibles, disponiendo de un cierto espacio de almacenamiento interno (buffers de espera) tanto en la entrada como en la salida, y de un único bus. Si un buffer de entrada está lleno, los nuevos paquetes que llegan al conmutador se pierden; si un buffer de salida está lleno, se producen bloqueos a nivel de la conmutación.

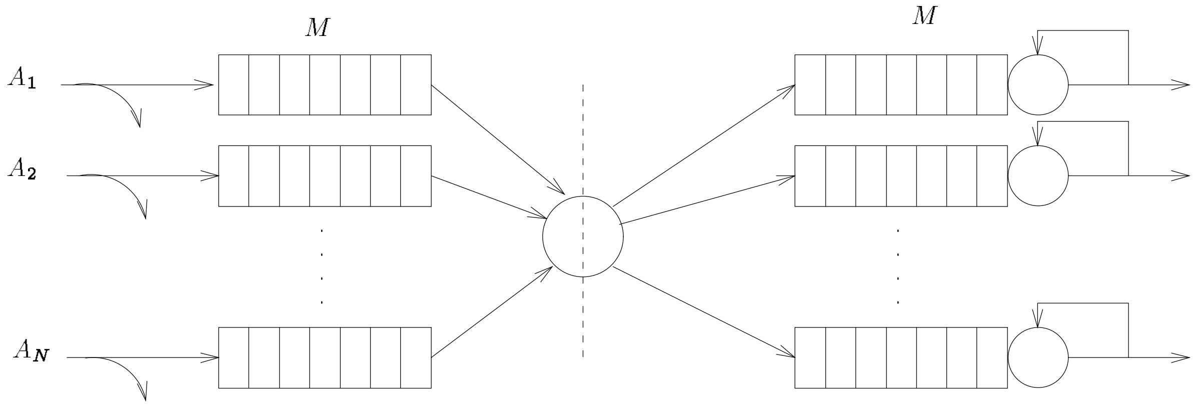

Figure 3: Arquitectura de un conmutador de tipo "shared-medium"

Las características del conmutador son las siguientes:

- **Fuentes:** existen N fuentes, cada una de las cuales genera paquetes según un proceso de Bernouilli interrumpido (IBP, Interrupted Bernoulli Process). Cada fuente tiene dos estados posibles, Activo y Pasivo. En un intervalo de tiempo unitario, la fuente i en estado Pasivo puede pasar a Activo con probabilidad $1-q[i]$; y de Activo a Pasivo con probabilidad $1-p[i]$. Cuando la fuente pasa de Pasivo a Activo, genera un paquete; mientras permanece en Activo, puede generar un paquete en cada intervalo con probabilidad $\alpha[i]$.

- **Buffers de entrada:** existen N buffers de entrada de tamaño M, cada uno de los cuales corresponde a una de las fuentes. Si hay una llegada de la fuente i y el buffer correspondiente está completo, el paquete se pierde.

- **Buffers de salida:** existen N buffers de salida de tamaño M, cada uno de los cuales corresponde a una de los destinos de salida. El tiempo de servicio en cada uno de los buffers de salida es geométrico de parámetro $r[i]$ (es decir, en cada intervalo de tiempo existe probabilidad $1-r[i]$ de que el primer paquete de la fila salga del sistema).

- **Conmutación**: existe un único bus que realiza la conmutación de paquetes, cuya velocidad es N veces más grande que la de las fuentes. Supondremos una política de asignación del bus TDM (Time Division Multiplexing), en la que a cada buffer de entrada se le asigna un intervalo de servicio predeterminado; de esta forma, en un intervalo unitario es posible procesar un paquete de cada uno de los buffers de entrada. Los paquetes del buffer de entrada i tienen como destino el buffer de salida j con probabilidad $D[i,j]$. Si cuando se está tratando un paquete de la entrada i, resulta que el buffer de salida j elegido está completo, se bloquea el tratamiento de i hasta el próximo turno.

4.2 Modelo bolas y urnas

Los parámetros necesarios para completamente caracterizar el sistema que hemos visto son: N el número de fuentes y destinos; $\vec{p}$, $\vec{q}$, $\vec{\alpha}$ los parámetros de los procesos IBP de llegada de paquetes; $\vec{r}$ los parámetros de los procesos de salida; D la matriz de ruteo; M el tamaño de los buffers. Veremos ahora un modelo bolas y urnas paramétrico en esos valores. Informalmente, podemos dividir el modelo de acuerdo a los distintos componentes del conmutador:

- **Fuentes:** para cada fuente usaremos dos urnas, una para representar el estado Inactivo y otra para representar el estado Activo. Las transiciones entre estas urnas corresponden a los cambios de estado de la fuente. Además, en estado activo tenemos una transición para representar la generación de paquetes.
- **Buffers de entrada:** para cada buffer de entrada se emplean dos urnas, una para representar el número de paquetes en el buffer y otra para indicar la salida correspondiente en caso de que exista bloqueo por estar el buffer de salida completo. Las transiciones corresponden al pasaje de paquetes desde los buffers de entrada hacia los buffers de salida.
- **Buffers de salida:** para cada buffer de salida alcanza con una sola urna, para representar el número de paquetes en el buffer. Las transiciones (de la urna a si misma, seguidas de una modificación instantánea que elimina una bola) corresponden a las salidas del sistema.

La descripción formal y completa del modelo bolas y urnas que proponemos es la siguiente:

- **Número de urnas:** $L = 5*N$. Para facilitar la notación, emplearemos los índices $f_1, \ldots, f_N$ para las urnas correspondientes a las fuentes activas, los índices $p_1, \ldots, p_N$ para las urnas correspondientes a las fuentes pasivas, los índices $e_1, \ldots, e_N$ para las urnas correspondientes a los paquetes almacenados en los buffers de entrada, los índices $t_1, \ldots, t_N$ para las urnas correspondientes al destino de los paquetes bloqueados, y los índices $s_1, \ldots, s_N$ para las urnas correspondientes a los buffers de salida.
- **Capacidades máximas:** $\overrightarrow{nmax}$, con $nmax[f_k] = 1$, $nmax[p_k] = 1$, $nmax[e_k] = m$, $nmax[t_k] = m$, $nmax[s_k] = m + 1$ para $0 < k \leq N$.
- **Estado inicial:** $\vec{e_0}$, con $e_0[f_k] = 0$, $e_0[p_k] = 1$, $e_0[e_k] = 0$, $e_0[t_k] = 0$, $e_0[s_k] = 0$ para $0 < k \leq N$.
- **Transferencias y modificaciones instantáneas:**
  - Pasaje de fuente de pasivo a activo (y generación de un paquete):
    $\lambda[p_k, f_k](\vec{n}) = 1 - q[k]$
    $\Gamma[p_k, f_k](\vec{n}) = \{(\vec{m}, 1)\}$ con $m[e_k] = min(n[p_k] + 1, m)$ y $m[i] = n[i]$ para todo $i \neq e_k$.
  - Pasaje de fuente de activo a pasivo:
    $\lambda[f_k, p_k](\vec{n}) = 1 - p[k]$
    $\Gamma[f_k, p_k](\vec{n}) = \{(\vec{n}, 1)\}$.
  - Generación de un paquete en estado activo:
    $\lambda[f_k, f_k](\vec{n}) = \alpha[k] \times p[k]$
    $\Gamma[f_k, p_k](\vec{n}) = \{(\vec{m}, 1)\}$ con $m[e_k] = min(n[e_k] + 1, m)$ y $m[i] = n[i]$ para todo $i \neq e_k$.

– Conmutación de un paquete de la entrada k a la salida j:
$\lambda[e_k, s_j](\vec{n}) = F(k, j, \vec{n})$, con

$$F(k,j,\vec{n}) = \begin{cases} 1 & \text{si } n[p_k] = j \text{ (bloqueo previo en dirección a } j\text{)}, \\ 0 & \text{si } n[p_k] > 0 \text{ y } n[p_k] \neq j \text{ (bloqueo previo en otra dirección)}, \\ D[k,j] & \text{si } n[p_k] = 0 \text{ (no hay bloqueo previo)}. \end{cases}$$

$\Gamma[f_k, s_j](\vec{n}) = \{(\vec{m}, 1)\}$, con

$$\vec{m} = \begin{cases} \vec{n} & \text{si } n[s_j] \leq M; \\ m[f_k] = n[f_k] + 1, m[s_j] = n[s_j] - 1, m[t_k] = j & \text{si } n[s_j] = M + 1 \text{ (bloqueo)}. \end{cases}$$

– Salida de paquetes del buffer k:
$\lambda[s_k, s_k](\vec{n}) = r[k]$
$\Gamma[s_k, s_k](\vec{n}) = \{(\vec{m}, 1)\}$, con $m[s_k] = n[s_k] - 1$, y $m[l] = n[l]$ para todo $l \neq k$.

4.3 Evaluación de la performance

Dado que los buffers de entrada son finitos, los paquetes que llegan cuando una fila de entrada está llena se pierden. La probabilidad de pérdida P_L (proporción de los paquetes generados que se pierden) es uno de los parámetros de performance más importantes de un conmutador de paquetes.

Para cada fuente, podemos calcular su probabilidad de pérdida a partir de la probabilidad estacionaria del sistema de la forma siguiente:

$$P_L[i] = \frac{\sum_{\vec{n} \in F_I} Prob(\vec{n})(1 - q[i]) + \sum_{\vec{n} \in F_A} Prob(\vec{n}) p[i] \alpha[i]}{\sum_{\vec{n} \in G} Prob(\vec{n})(1 - q[i]) + \sum_{\vec{n} \in H} Prob(\vec{n}) p[i] \alpha[i]}$$

donde $Prob(\vec{n})$ es la probabilidad estacionaria de que el sistema se encuentre en el estado $\vec{n}$, G es el conjunto de estados en los que la fuente está en estado Pasivo, H es el conjunto de estados en los que la fuente está en estado Activo, F_I es el conjunto de estados en los que la fuente está en estado Pasivo y la fila de entrada correspondiente está llena, F_A es el conjunto de estados en los que la fuente está en estado Activo y la fila de entrada correspondiente está llena.

El denominador es igual a $\rho[i]$ tasa media de generación de paquetes de la fuente i, y puede calcularse con la siguiente fórmula:

$$\rho[i] = \frac{\alpha[i](1 - q[i])}{2 - p[i] - q[i]}.$$

El numerador se puede calcular a partir de la distribución estacionaria si el tamaño del sistema lo permite, o sino a través de simulación.

Estudiaremos el efecto del tipo de tránsito sobre la probabilidad de pérdida $P_L = \sum_{i=1}^{N} P_L[i]$, manteniendo una tasa de generación de paquetes fija. Para esto fijaremos los siguientes parámetros: $N = 4$, $\alpha[i] = 0.5 \; \forall i$, $D[i,j] = 1/4 \; \forall i, j$, y haremos variar $p[i]$ de 0.1 a 0.9 y concomitantemente $q[i]$ de 0.82 a 0.98 para mantener una carga constante ($\rho[i] = 1/12$). Los valores bajos de $p[i]$ equivalen

p	q	P_L estimado
0.10	0.82	6.91×10^{-5}
0.20	0.84	1.49×10^{-4}
0.30	0.86	2.80×10^{-4}
0.40	0.88	3.90×10^{-4}
0.50	0.90	5.48×10^{-4}
0.60	0.92	9.90×10^{-4}
0.70	0.94	1.35×10^{-3}
0.80	0.96	1.95×10^{-3}
0.90	0.98	2.34×10^{-3}

Table 1: Resultados de simulación para el conmutador ATM a carga constante

a flujos de datos donde el largo de las ráfagas de paquetes es pequeño, en cambio los valores altos de $p[i]$ indican que los datos suelen llegar en largas ráfagas.

La Tabla 1 muestra los distintos valores obtenidos por simulación. Se puede apreciar como la probabilidad de pérdida aumenta en forma considerable a medida que el tráfico tiende a contener mayor cantidad de ráfagas, aún cuando la carga total que soporta el conmutador es la misma. Este es un fenómeno importante que se presenta en las redes ATM, y que debe ser tenido en cuenta ya que el tráfico en las mismas tiende a poseer estas características, especialmente en el caso de transmisión de video, sonido, así como en la transferencia de grandes cantidades de información.

5 Conclusiones

En este trabajo hemos presentado una extensión al formalismo de bolas y urnas que permite el modelado de sistemas cuyo parámetro de tiempo es discreto. Esta extensión ha sido incorporada en la biblioteca **BB**, dando acceso al conjunto de servicios que la misma provee (generación de la cadena de Markov subyacente, almacenamiento y recuperación, simulación).

Como ejemplo de aplicación, hemos tomado una arquitectura de conmutador ATM, desarrollando en detalle su modelo en el formalismo de bolas y urnas. La performance de esta arquitectura fue evaluada (para ciertos valores de sus parámetros) a través de las primitivas de simulación ofrecidas por la biblioteca **BB**; mostrando como a carga fija la performance del sistema depente de las características del proceso de llegadas (si existen ráfagas largas, la autocorrelación del proceso es mayor, y las performances disminuyen).

Como líneas de trabajo futuro, encontramos tanto aquellas basadas en ampliar la biblioteca a nuevas clases de modelos (semi-markovianos, por ejemplo), como aquellas basadas en la aplicación de la biblioteca existente al análisis y evaluación de distintos sistemas informáticos y de telecomunicaciones.

References

[BAA+93] C. Béounes, M. Aguéra, J. Arlat, S. Bachmann, C. Bourdeau, J.E. Doucet, K. Kanoun, J.-C. Laprie, S. Metge, J. Moreira de Souza, D. Powell, and P. Spiesser. SURF-2:

A program for dependability evaluation of complex hardware and software systems. In *FTCS-23 Digest of Papers*, pages 668–673, Toulouse, France, June 1993. IEEE Computer Society Press.

[BdSM91] Steven Berson, Edmundo de Souza e Silva, and Richard R. Muntz. A methodology for the specification and generation of Markov models. In William J. Stewart, editor, *Numerical Solution of Markov Chains*, volume 8, chapter 2, pages 11–36. Marcel Dekker, Inc, 1991.

[BHL+93] Alvin M. Blum, Philip Heidelberger, Stephen S. Lavenberg, Marvin K. Nakayama, and Perwez Shahabuddin. System availability estimator (SAVE) language reference and user's manual version 4.0. Research Report RA 219 S (82923), T. J. Watson Research Center - IBM, Yorktown Hts, NY 10598, June 1993.

[Bou93] Marc Bouissou. The FIGARO dependability evaluation workbench in use: Case studies for fault-tolerant computer systems. In *FTCS-23 Digest of Papers*, pages 680–685, Toulouse, France, June 1993. IEEE Computer Society Press.

[Can93] H. Cancela. Construction de modèles pour l'évaluation de la sûreté de fonctionnement. Rapport de DEA, IRISA - Université de Rennes 1, Campus de Beaulieu 35042 Rennes Cedex, September 1993.

[CF86] J. A. Carrasco and J. Figueras. METFAC: Design and implementation of a sofware tool for modelling and evaluation of complex fault-tolerant computing systems. In 16^{th} *International Symposium on Fault-Tolerant Computing. FTCS 16*, pages 424–429, Vienna, Austria, July 1986. IEEE C.S. Technical Comittee in Fault-Tolerant Computing, IEEE Computer Society Press.

[Chi91] Giovanni Chiola. GreatSPN 1.5 software architecture. In *5th. Modelling Techniques and Tools for Computer Performance Evaluation*, pages 117–132, Torino, Italy, February 1991.

[Cia95] G. Ciardo. Discrete-time Markovian stochastic Petri nets. In *Proceedings of the 2nd. International Workshop on Numerical Solutions for Markov Chains*, pages 339–358, Raleigh, North Carolina, January 1995.

[CR94] H. Cancela and G. Rubino. Construcción de modelos para la evaluación de la seguridad de funcionamiento. In *CLAIO 94*, Santiago, Chile, July 1994.

[DTGN84] J.B. Dugan, K.S. Trivedi, R.M. Geist, and V.F. Nicola. Extended stochastic Petri nets: applications and analysis. In *Performance'84*, pages 507–519, 1984.

[Gra91] Winfried K. Grassmann. Finding transient solutions in markovian event systems through randomization. In William J. Stewart, editor, *Numerical Solution of Markov Chains*, volume 8 of *Probability: Pure and Applied*, chapter 18, pages 37–61. Marcel Dekker, Inc, 1991.

[Hun83] J. Hunter. *Mathematical Techniques of Applied Probability*, volume 2, Discrete Time Models. Academic Press, 1983.

[Kob83] H. Kobayashi. Discrete time queuing systems. In *Probability Theory and Computer Sciences.* Academic Press, 1983.

[KS91] R.L. Klevans and W.J. Stewart. Xmarca: User's manual version 1.0. Technical report, Dept. of Computer Science, North Carolina State University, Raleigh, North Carolina, 27695-8206, March 1991.

[Lap92] Jean-Claude Laprie, editor. *Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese.* Springer-Verlag, 1992.

[PA91] B. Plateau and K. Atif. Stochastic automata network for modeling parallel systems. *IEEE Trans. on Software Engineering*, 17(10):1093–1108, October 1991.

[SO93] William H. Sanders and W. Douglas Obal, II. Dependability evaluation using UltraSAN. In *FTCS-23 Digest of Papers*, pages 674–679, Toulouse, France, June 1993. IEEE Computer Society Press.

[Ste91] William J. Stewart. MARCA: Markov chain analyzer, a software package for Markov modeling. In William J. Stewart, editor, *Numerical Solution of Markov Chains*, volume 8 of *Probability: Pure and Applied*, chapter 3, pages 37–61. Marcel Dekker, Inc, 1991.

[Vu 93] C. Vu Dui. Discrete time open queueing networks with feedback. Research Report MASI 93.13, Institut Blaise Pascal, MASI, 4, Place Jussieu - 75252 Paris Cedex 05, France, March 1993.

[ZP94] A.O. Zaghloul and H.G. Perros. Approximate analysis of a shared-medium ATM switch under bursty arrivals and nonuniform destinations. *Performance Evaluation*, 21:111–129, 1994.