Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

PRÁCTICAS DE MODELADO Y SIMULACIÓN

Práctica 3: Simulador de colas G/G/k

Profesores:

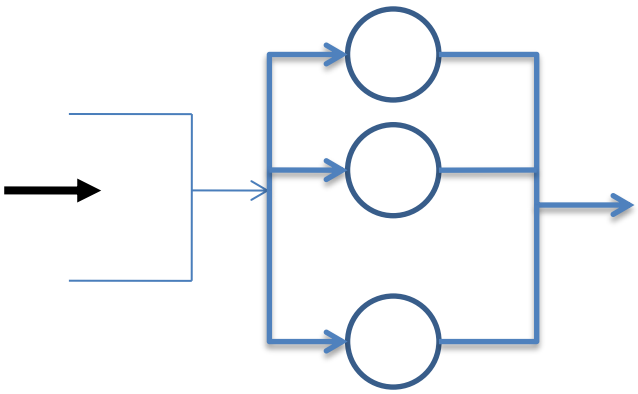
Javier Vales Alonso

1. INTRODUCCIÓN

En esta práctica se implementará en matlab un **simulador de colas G/G/k**, y se evaluará el rendimiento de dichas colas en diferentes configuraciones prácticas (ver Boletín de entrega). Auxiliarmente se construirá un generador U(0,1) mediante un **generador congruencial lineal (GCL)** y a partir de éste, generadores de variables aleatorias U(a, b), exponenciales(λ) y Bernoulli(p).

Una cola G/G/k es una estructura en la cual se dispone de k *recursos*. Cada uno de estos recursos ejecuta *tareas*. El tiempo de ejecución de una tarea es una variable aleatoria (**S** unidades de tiempo). Las tareas llegan al sistema cada **X** unidades de tiempo, donde **X** es también una variable aleatoria. Cuando la tarea llega al sistema, si no hay otras tareas esperando y alguno de los k recursos está libre, la tarea pasa a ejecutarse. Por el contrario, si los recursos están ocupados y hay otras tareas esperando, la nueva tarea pasará también a estar a la espera. En este sistema asumimos además que la disciplina de cola es tipo FIFO. Es decir, las tareas esperan en orden de llegada.

Gráficamente, se suele representar a estas colas con el siguiente esquema:



Los sistemas G/G/k modelan muchas situaciones "del mundo real" interesantes. Por ejemplo, una cola de personas esperando a ser atendidas (en tiendas, bancos, aeropuertos, etc.). También en el ámbito de las telecomunicaciones muchos sistemas encajan en este modelo. Por ejemplo, un *router* debe almacenar los paquetes a transmitir (tareas) y los enviará por orden de llegada por la línea o líneas de transmisión (recursos). También los trabajos enviados a grandes computadores de cálculo se pueden modelar con este esquema. Los trabajos serían ejecutados por CPUs (recursos).

En todas estas situaciones nos interesa estudiar fundamentalmente dos aspectos:

1. Tiempo de respuesta (o transito) de una tarea. El tiempo desde que la tarea llega al sistema hasta que termina su ejecución. Lo denotaremos por la letra T.
2. El número medio de tareas dentro el sistema. Lo denotaremos por la letra N.

Analíticamente estos sistemas pueden modelarse solamente en casos "excepcionales". Si las variables **X** y **S** son exponenciales tendríamos un sistema Markoviano (denotado M/M/k) analizable para cualquier valor de k. Si **X** es exponencial, pero **S** no, tendríamos un sistema

semi-markoviano (denotado M/G/k) para el cual sólo existen soluciones para el caso k=1 (M/G/1), denominadas fórmulas de Pollaczek-Khintchine.

Es indudable que muchas situaciones de interés caen fuera de estas dos configuraciones (M/M/k o M/G/1). En estos casos debemos recurrir a la simulación para la evaluación de estos sistemas. Este es precisamente el desafío planteado en esta práctica.

2. OBJETIVOS

Relativos a simulación:

- Comprender la estructura de los simuladores por eventos discretos
- Adquirir soltura en la manipulación de conceptos de simulación
- Comprender los mecanismos de generación de muestras de variables aleatorias

Relativos al sistema bajo estudio:

- Comprender el funcionamiento de las colas G/G/k y evaluar su rendimiento en diversas configuraciones prácticas.
- Comprender las posibilidades del estudio por simulación frente al análisis matemático
- Validar el estudio por simulación

3. DESARROLLO DE LA PRÁCTICA

La práctica se divide en tres partes:

1. Primero se implementará una función de matlab que permita generar muestras de VAs.
2. Tras esto deberá implementar el simulador de colas G/G/k donde se empleará la función anterior para parametrizar el funcionamiento del mismo.
3. Finalmente, se probará el simulador en distintas configuraciones prácticas, con el fin de obtener resultados sobre el comportamiento del sistemas

3.1. Implementación de la función *aleatorio*

En la primera sesión de prácticas empleó un simulador que hacía uso de la función *aleatorio_exp()*. Se comprobó que el simulador funcionaba también sustituyendo esta función por otra que generase muestras de una VA uniforme. No obstante, es engorroso estar cambiando el código cada vez que cambiemos los tipos de variables aleatorias. Para evitar esto vamos a construir una nueva función de matlab, con la siguiente estructura:

```
function [Z, muestra] = aleatorio(Z, tipo, param1, param2)
% TIPO = 0 -> VA uniforme [0,1]
% TIPO = 1 -> VA uniforme [param1, param2]
% TIPO = 2 -> VA exponencial lambda=param1
% TIPO = 3 -> Devuelve siempre param1 (VA “degenerada”)
% TIPO = 4 -> VA Bernoulli (devuelve 1 con probabilidad dada por param1, %
sino 0)
% SE PUEDEN AÑADIR MAS TIPOS

end
```

Es decir, según el parámetro tipo, devolverá una muestra de un tipo u otro. Las variables param1 y param2 parametrizarán la variable aleatoria (nótese que no siempre son necesarios). La variable Z se corresponde a la última muestra proporcionada por el GCL, y se usará como base para construir muestras de las variables aleatorias deseadas.

Para obtener muestras del GCL deberá programar otra función matlab:

```
function  nuevoZ  = GCLM( Z )
% Usando Z como muestra previa del generador, crea la nueva muestra.
% El GCLM debe usar los parámetros de referencia de Kobayashi

end
```

Observe que GCLM() devuelve un entero, y que para obtener la variable aleatoria U(0,1) debe dividir el valor retornado por su parámetro m.

Empleando el método de la transformada inversa se obtiene que:

- Las muestras de U(a,b) se obtienen, dada u~U(0,1), como a+u(b-a).
- Las muestras de exp(λ) se obtienen, dada u~U(0,1), como -ln(u)/λ.

Para la obtención de muestras Bernoulli(p), debe emplear el método de generación para VA de tipo discreto explicado en el Tema 5.

3.2. Implementación del simulador

Ahora debe implementar el simulador G/G/k partiendo del modelo M/M/1 proporcionado en el fichero esqueletoSIM.m.

En primer lugar deberá añadir a este fichero los parámetros de configuración del simulador G/G/k (de la misma forma que lambda y mu nos servían para configurar el simulador M/M/1):

- `Z`: semilla del GCL
- `k`: número de recursos
- `tipoX`: tipo VA para **X** (según los tipos definidos en la función aleatorio)
- `tipoS`: tipo VA para **S** (según los tipos definidos en la función aleatorio)
- `param1X`: param1 para **X**
- `param2X`: param2 para **X**
- `param1S`: param1 para **S**
- `param2S`: param2 para **S**

Empleando esta parametrización podrá cambiar rápidamente la configuración del simulador. Para poder trabajar con las variables aleatorias así definidas, cambie las llamadas a la función `aleatorio_exp` por su función `aleatorio`.

A continuación deberá actualizar los procedimientos asociados a los eventos de LLEGADA y SALIDA de tareas. La operación del simulador M/M/1 falla en un caso general con k recursos ya que **el orden de salida de tareas no tiene porque ser el orden de llegada**, por lo tanto ya no se podrán obtener muestras de T como:

```
[fifoTiempos, tentrada] = popFIFO(fifoTiempos);
summuestrasT = summuestrasT + (t_simulacion - tentrada);
```

Sin embargo, puede resolverse este problema de modo simple. Observe que aunque el orden de salida no sea el de entrada al sistema si **se verifica que el orden de entrada a los recursos es el "natural"** (el primero que llega, accede antes a un recurso que se ha vaciado). A partir de esta observación se plantea este procedimiento para solucionar el problema de la obtención de las muestras del tiempo de respuesta:

1. Guarde el tiempo en la FIFO solamente si la tarea espera.
2. Cuando planifique una SALIDA añada como información adicional del evento **el momento de entrada de dicha tarea** (para ello, añada un tercer campo `tllegadatarea` a la lista de eventos modificando las funciones **`encolarEvento`** y **`sgteEvento`** para añadir y devolver este campo adicional). Ese tiempo será `t_simulacion` si el evento de SALIDA es de una tarea que no ha tenido que esperar. Si se está planificando la SALIDA de una tarea que está en la sala de espera, el tiempo de llegada de esa tarea se obtiene de la FIFO.
3. Cuando procese el evento de SALIDA la muestra de T es simplemente:

```
t_simulacion – tllegadatarea
```

4. EJERCICIO OPCIONAL

Modifique el simulador para poner obtener el número medio de tareas en el sistema (N) a partir de la obtención de muestras de dicha variable. Para realizarlo debe crear un nuevo evento de MUESTREO que a intervalos aleatorios se ejecute y obtenga dichas muestras.