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:

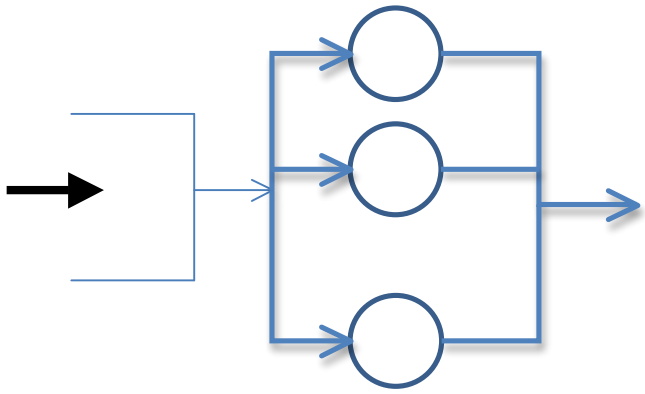
Javier Vales Alonso
Juan José Alcaraz Espín

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. Auxiliarmente se construirá un generador U(0,1) mediante un **generador congruencial multiplicativo** y a partir de este, generadores de variables aleatorias U(a, b) y exponenciales(λ).

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 cola es de 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 todos estas situaciones nos interesa estudiar fundamentalmente dos aspectos:

- 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.
- 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")
% 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 un generador congruencial multiplicativo, y se usará como base para construir muestras de las variables aleatorias deseadas.

Para obtener muestras del generador congruencial multiplicativo 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 Fishman y Moore

end
```

Observe que GCLM() devuelve un entero, y que para obtener la variable aleatoria U(0,1) debe dividir el valor retornado por $(2^{32}-1)$.

Las muestas de U(a,b) se obtienen dada u~U(0,1), como a+u*(b-a).

Las muestras de exp(λ) se obtienen tras obtener u~U(0,1), como -ln(u)/λ. ¿Qué método de los explicados en teoría sustenta este procedimiento de generación para VAs exponenciales?

(LOS CODIGOS DEBE ENTREGARLOS EN LA MEMORIA)

Verificación:

- Compruebe que la muestra Z_{10000} es 399268537 partiendo de la semilla "1"
- Calcule las dos primeras muestras de una VA exponencial de parámetro λ (elija el que desee) manualmente y compruebe que coinciden con las dadas **(RESULTADO A ENTREGAR EN LA MEMORIA)**

3.2. Implementación del simulador

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

Fundamentalmente deberá actualizar los procedimientos asociados a los eventos de LLEGADA y SALIDA de tareas. Adicionalmente deberá definir variables que permitan desde el script EsqueletoSIM definir las variables aleatorias X y S (es decir, definir tipoX, tipoS, param1X, param2X, param1S, param2S).

Debemos señalar que en un simulador G/G/k 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 solamente como:

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

Sin embargo, puede resolverse este problema fácilmente:

- Primero, 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).

- Por tanto, cuando se planifique el evento de salida puede extraerse de la FIFO de tiempos el momento de entrada de dicha tarea y guardar dicho tiempo en la propia cola de eventos (añada un tercer campo a la lista de eventos y úselo para almacenar esta información en los eventos de SALIDA)

(EL CÓDIGO DEL NUEVO ESQUELETOSIM DEBE ENTREGARLO EN LA MEMORIA)

EJERCICIO OPCIONAL: OBTENGA TAMBIÉN EL NÚMERO MEDIO DE USUARIOS EN EL SISTEMA (N)

3.3. Pruebas a realizar

En primer lugar, realizaremos varias pruebas de validación.

a) Elija una configuración M/M/1 (es decir, valores λ y μ) y compruebe que el resultado es coherente **(A ENTREGAR EN LA MEMORIA)**
b) Elija una configuración M/M/3 (es decir, valores λ y μ) y compruebe que el resultado es coherente **(A ENTREGAR EN LA MEMORIA)**
c) Elija una configuración M/D/1 (es decir, valores λ y s –s es constante-) y compruebe que el resultado es coherente **(A ENTREGAR EN LA MEMORIA)**
d) Si ha realizado el ejercicio opcional compruebe que se verifica

$N = T * \lambda$ (LEY DE LITTLE)

En todos los escenarios anteriores.
(A ENTREGAR EN LA MEMORIA)

Tras las pruebas de validación se ejecutará el simulador para las configuraciones:

1. G/G/k, con X~U[2,5], S~U[1,4], y k=1..5. Indique el promedio de T obtenido en la memoria.
2. G/M/5, con X=40 (constante) S~exp(λ), con λ=10…100 (dibuje en una gráfica de matlab el resultado)