Ejercicio 2 SS

José Manuel Pérez Lendínez

February 24, 2021

Contents

1 Sucesos y grafos de sucesos. **3**

2 Variables Importantes. **3**

3 Rutinas de sucesos. **4**
3.1 Suceso de llegada. 5
3.2 Suceso fin A. 5
3.3 Suceso fin B. 6
3.4 Suceso fin. 6

4 Inicialización de los parámetros. **7**

5 Composición de la lista de suceso. **7**

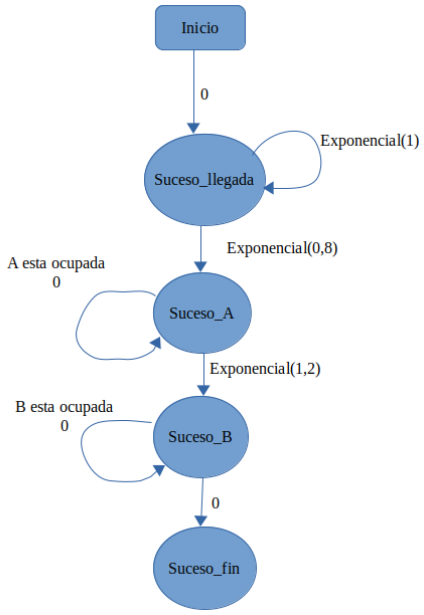
6 Ejecuciones **8**

1 Sucesos y grafos de sucesos.

Nuestro grafo de sucesos contendrá los siguientes sucesos:

1. **Suceso de Llega:** Se da cuando llega un nuevo cliente al servidor y queda en espera para ejecutar A.
2. **Suceso de A:** Cuando un cliente termina de ejecutar A y este queda libre para ejecutar a otro cliente. También se encargara de poner a la espera o ejecutar al cliente en b.
3. **Suceso de B:** Se dará cuando termine de ejecutar un cliente en el servidor B y tendremos que liberar el servidor b y añadir un nuevo cliente que este en la espera de b.
4. **Suceso Fin:** Se dará cuando se llegue al tiempo máximo que tengamos para ejecutar, en nuestro caso 480.

El grafo seria el siguiente:



2 Variables Importantes.

Las variables mas importantes son las siguientes:

1. **libreA y libreB:** Son dos booleanos que marcan si los servidores a y b están libres.

2. **colaA y colaB:** Son enteros que nos indican el numero de clientes que están en espera para el servidor a y b.
3. **parar:** Booleano que nos dirá cuando se ha llegado al suceso fin y parara el simulador.
4. **lsuc:** Lista de clientes que están por ser asistidos.
5. **llegadas:** Lista de double que indica cunado llego cada cliente
6. **numSimulaciones:** Se pasa como primer parámetro y especifica el numero de simulaciones que realizaremos.
7. **reloj:** Llevará el reloj del sistema y con el que marcaremos en que momento sucede algún suceso.
8. **nodosFinalizados:** Se llevara la cuenta de nodos que se han finalizado.
9. **acumSistema:** Se llevara la acumulación del tiempo que están en el sistema todos los nodos.

3 Rutinas de sucesos.

Primero vamos a enseñar la función generarInsertarSuceso que sera utilizada en las rutinas de todos los sucesos:

```
void generarInsertarSuceso(int suceso, double tiempo)
{
    nodo.suceso = suceso;
    nodo.tiempo = reloj + generador_exponencial(tiempo);
    insertar_lsuc(nodo);
}
```

Esta función se encarga de generar un nodo y añadirlo en la lista de nodos. Se le pasan como parámetros el tipo de suceso al que corresponde el nodo y el tiempo medio que ocupa dicho suceso para generar el tiempo que ocupara.

3.1 Suceso de llegada.

```
void llegada()
{


    generarInsertarSuceso(suceso_llegada, TIEMPO_MEDIO_LLEGADA);
    llegadas.push_back(reloj);

    if (libreA)
    {
        libreA = false;
        generarInsertarSuceso(suceso_fin_A, TIEMPO_MEDIO_A);
    }
    else
    {
        colaA++;
    }
}
```

En este se empieza generando el nodo con el suceso llegada y se añade a la lista llegadas el tiempo en el que llego. Después miramos si el servidor A esta libre para generar un nuevo nodo con el suceso para fin de A y poneos libreA en false para indicar que esta ocupado. Si A no estaba libre aumentamos la cola de A para indicar que hay otro nodo esperando.

3.2 Suceso fin A.

```
void finA()
{

    if (colaA > 0)
    {
        colaA--;
        generarInsertarSuceso(suceso_fin_A, TIEMPO_MEDIO_A);
    }
    else
    {
        libreA = true;
    }

    if (libreB)
    {
        libreB = false;
        generarInsertarSuceso(suceso_fin_B, TIEMPO_MEDIO_B);
    }
    else
    {
        colaB++;
    }
}
```

En este suceso lo primero que veremos es si hay algún nodo esperando para ser ejecutado en A. Si lo hay quitamos un nodo de la cola y lo añadimos a la lista con el suceso de fin de a. En el caso de que no lo queden nodos esperando para a pones a como libre.

Después tenemos que ver si el servidor b esta libre, si esta libre añadiremos un nodo para el suceso fin b y si no lo esta simplemente aumentamos la cola del servidor b.

3.3 Suceso fin B.

```
void finB()
{

    if (colaB > 0)
    {
        colaB--;
        generarInsertarSuceso(suceso_fin_B, TIEMPO_MEDIO_B);
    }
    else
    {
        libreB = true;
    }

    nodosFinalizados++;
    double tiempoLlegada = llegadas.front();
    llegadas.pop_front();
    acumSistema += reloj - tiempoLlegada;
}
```

Para el suceso de fin b solo tenemos que mirar si hay mas nodos en la cola de b para añadirlos y si no es el caso poner libre el servidor B. En este caso también añadiremos los datos necesarios para calcular posteriormente el tiempo de estancia del servidor. Para esto añadimos uno en la variable que nos llevara la cuenta de nodos que hemos finalizado y cogeremos el tiempo del primer nodo que tenemos en el frente de la lista de llegadas para calcular cuanto tiempo lleva en el sistema y sumarlo a acumSistema para luego calcular la media.

3.4 Suceso fin.

Este es el suceso que se encarga de finalizar la simulación.

```
void fin()
{
    parar = true;
    double estanciamedia = acumSistema / nodosFinalizados;
    cout << endl
         << "Simulacion " << numSimulacion << " Tiempo medio de estancia en el sistema " << estanciamedia;
}
```

Para finalizar la simulación ponemos parada = true. Después calculamos la estancia media y la mostramos para que se vea por cada simulación.

4 Inicialización de los parámetros.

```
void inicializacion()
{

    nodosFinalizados = 0;
    reloj = 0.0;
    acumSistema = 0.0;
    nodo.suceso = suceso_llegada;
    nodo.tiempo = reloj + generador_exponencial(TIEMPO_MEDIO_LLEGADA);
    insertar_lsuc(nodo);
    nodo.suceso = suceso_fin;
    nodo.tiempo = reloj + FINAL;
    insertar_lsuc(nodo);
    colaA = 0;
    colaB = 0;

    libreA = true;
    libreB = true;
    parar = false;
}
```

Inicializamos el numero de nodos finalizados a 0 y el reloj igual. A continuación añadimos el primer nodo que sera un nodo de llegada y también añadiremos el nodo que finalizara el simulador. Ponemos las colas para los dos servidores a 0 y también ponemos los dos servidores como libres. Después ponemos la parada a false para que se empiece la simulación.

5 Composición de la lista de suceso.

Cada vez que se añade un nuevo nodo se realiza una ordenación de la lista mediante el valor de nodo.tiempo. El código es el siguiente.

```
bool compare(const suc &s1, const suc &s2)
{
    return s1.tiempo < s2.tiempo;
}

void insertar_lsuc(suc n)
{
    lsuc.push_back(n);
    lsuc.sort(compare);
}
```

6 Ejecuciones

Vamos a realizar varias ejecuciones para ver los resultados:

Numero Simulaciones	Media de estancia
5	52.2423 ± 18.3738
10	37.4228 ± 18.652
100	46.983 ± 14.6713
1000	46.1055 ± 14.5343

Como se ve al final se estabiliza en una media de 46 aproximadamente aunque tenemos una desviación muy grande. Para conseguir un tiempo medio de estancia medio inferior a 10 se usa 0.86 como tiempo medio de espera para b. Con ese valor obtenemos un tiempo tal que 9.09592 ± 2.930469.