F1 – Diseño Modular del Simulador de Eventos Discretos y Resultados

18 de junio de 2008

Contestar a las siguientes preguntas

(Contestar exclusivamente lo que pregunta. Utilizar las líneas que sean necesarias para contestar, pero recordar los criterios de brevedad y claridad especificados en el enunciado de la práctica).

1. ¿De qué tipo de simulación se trata (horizonte finito o infinito)?

   La simulación es de horizonte no finito. Buscamos estimar la evolución del tiempo medio de respuesta de todo el sistema y no existe ningún acontecimiento que marque el fin de cada una de las experiencias a realizar sobre el modelo matemático.

2. ¿Qué metodo habeis utilizado para construir el intervalo de confianza?

   El método de las réplicas.

3. ¿Cómo habeis tratado el problema del transitorio?

   Hemos utilizado la misma técnica que QNAP2, utilizar longitudes de réplica muy grandes de modo que el peso del transitorio es muy reducido.

4. ¿Qué método habeis utilizado para ajustar el intervalo de confianza a los criterios de precisión?

   Dada la precisión especificada en el enunciado en forma de error relativo hemos hecho uso del método secuencial para realizar las suficientes réplicas cómo para poder dar la estimación final con ese nivel de confianza y error relativo.

   Los apartados 5, 6 y 7 deben contestarse en el supuesto de programación clásica o estructurada, aunque finalmente se haya utilizado programación orientada a objetos.

5. Especifica y describe mínimamente (una frase) cada uno de los eventos que habeis definido en vuestro simulador.

   Arrival1: llegada de un cliente a la estación número 1.
   Arrival2: llegada de un cliente a la estación número 2.
   Arrival3: llegada de un cliente a la estación número 3.
   Departure1: salida de un cliente de la estación número 1.
   Departure2: salida de un cliente de la estación número 2.
   Departure3: salida de un cliente de la estación número 3.

6. Especifica y describe mínimamente (una frase) cada una de las variables de estado que habeis definido en vuestro simulador.

   Clock: reloj del sistema.
   EventsList: lista de eventos del sistema.
   Arrivals1: cola de instantes de llegada a la estación número 1.
   Arrivals2: cola de instantes de llegada a la estación número 2.
   Arrivals3: cola de instantes de llegada a la estación número 3.
   ServerBusy1: ocupación del servidor de la estación número 1.
   ServerBusy2: ocupación del servidor de la estación número 2.
   ServerBusy3: ocupación del servidor de la estación número 3.

7. Especifica y describe mínimamente (una frase) cada uno de los contadores estadísticos que habeis definido en vuestro simulador.

   Served1: número de clientes que han recibido servicio en la estación número 1.
   Served2: número de clientes que han recibido servicio en la estación número 2.
   Served3: número de clientes que han recibido servicio en la estación número 3.
   Served: número de clientes que han recibido servicio en el sistema.
   RTSum1: suma de los tiempos de respuesta experimentados por los clientes en la estación número 1.
   RTSum2: suma de los tiempos de respuesta experimentados por los clientes en la estación número 2.
   RTSum3: suma de los tiempos de respuesta experimentados por los clientes en la estación número 3.

8. Pseudo-código de todo el simulador. Se trata de especificar en forma de pseudo-código el contenido de todo el simulador, de forma que responda al modelo de la Figura 4 de la lección L11. Esto quiere decir, por lo que respecta al bloque de simulación de trazas individuales: especificar todas las rutinas de eventos, las otras rutinas y las relaciones entre todas ellas (en forma de llamadas); por otra parte, este bloque de simulación de trazas individuales se tiene que conectar convenientemente con el bloque de control estadístico, el contenido del cual también se tiene que especificar.

   Rutina principal

Algorithm 1 Rutina principal

```
inicializacionSimulador()
// ejecutar cómo mínimo 2 réplicas para poder aplicar el método secuencial
replicar()
replicar()
mientras (hayQueEjecutarOtraReplica()) {
    replicar()
}
generarInformeEstadistico()
```

   Rutina de control estadístico

Algorithm 2 Rutina de control estadístico

CIRT1 ← calcular intervalo de confianza para el tiempo de respuesta de la estación 1
CIRT2 ← calcular intervalo de confianza para el tiempo de respuesta de la estación 2
CIRT3 ← calcular intervalo de confianza para el tiempo de respuesta de la estación 3
CIRTS ← calcular intervalo de confianza para el tiempo de respuesta del sistema
MEANRT1 ← calcular el tiempo de respuesta medio de la estación 1 según las réplicas ejecutadas
MEANRT2 ← calcular el tiempo de respuesta medio de la estación 2 según las réplicas ejecutadas
MEANRT3 ← calcular el tiempo de respuesta medio de la estación 3 según las réplicas ejecutadas
MEANRTS ← calcular el tiempo de respuesta medio del sistema según las réplicas ejecutadas
si (CIRT1/MEANRT1 >RELATIVEERROR)
    retornar verdadero
si (CIRT2/MEANRT2 >RELATIVEERROR)
    retornar verdadero
si (CIRT3/MEANRT3 >RELATIVEERROR)
    retornar verdadero
si (CIRTS/MEANRTS >RELATIVEERROR)
    retornar verdadero
// si se cumplen los criterios de error relativo no hay que ejecutar más réplicas
retornar falso

Rutina generadora del informe estadístico

Algorithm 3 Rutina generadora del informe estadístico

mostrar valores medios por estaciones y del sistema
mostrar intervalos de confianza de las estimaciones anteriores

Rutina de inicialización del simulador

Algorithm 4 Inicialización del simulador

// iniciar variables necesarias para el control estadístico
RT1 ← 0
RT2 ← 0
RT3 ← 0
RTS ← 0
SQRTRT1 ← 0
SQRTRT2 ← 0
SQRTRT3 ← 0
SQRTRTS ← 0

Rutina de inicialización de la réplica

Algorithm 5 Inicialización de la réplica

// inicializar reloj y lista de eventos
CLOCK ← 0
EVENTSLIST ← lista vacía
// inicializar contadores estadísticos
SERVERBUSY1 ← falso
SERVERBUSY2 ← falso
SERVERBUSY3 ← falso
SERVED1 ← 0
SERVED2 ← 0
SERVED3 ← 0
SERVED ← 0
RTSUM1 ← 0
RTSUM2 ← 0
RTSUM3 ← 0
// inicializar las colas de cada estación
ARRIVALS1 ← cola vacía
ARRIVALS2 ← cola vacía
ARRIVALS3 ← cola vacía
// configurar la primera llegada de un cliente a la estación 1
EVENTSLIST ← ARRIVAL1

Rutina de temporización

Algorithm 6 Rutina de temporización

EVENTO ← eventoMasProximo(EVENTSLIST)
CLOCK ← InstanteDe(EVENTO)

Rutina de simulación de réplicas

Algorithm 7 Rutina de simulación de trazas individuales

REPLICAS ← REPLICAS + 1
inicializacionReplica()
mientras (CLOCK <MAXREPLICATIONTIME) {
    EVENTO ← determinar siguiente evento a procesar
    llamada a la rutina de eventos correspondiente
}
// acumular contadores estadísticos
RT1 ← RT1 + (RTSUM1/SERVED1)
RT2 ← RT2 + (RTSUM2/SERVED2)
RT3 ← RT3 + (RTSUM3/SERVED3)
RTS ← RTS + ((RTSUM1+RTSUM2+RTSUM3)/SERVED)
SQRTRT1 ← SQRTRT1 + $(RTSUM1/SERVED1)^2$
SQRTTR2 ← SQRTRT2 + $(RTSUM2/SERVED2)^2$
SQRTRT3 ← SQRTRT3 + $(RTSUM3/SERVED3)^2$
SQRTRTS ← SQRTRTS + $((RTSUM1 + RTSUM2 + RTSUM3)/SERVED)^2$

Rutina de eventos

Algorithm 8 Rutina de eventos. Llegada de un cliente a la estación 1

```
si (!SERVERBUSY1) {
    SERVERBUSY1 ← verdadero
    EVENTLIST ← marcar instante de salida del cliente
} sino {
    ARRIVALS1 ← poner el cliente en cola
}
EVENTSLIST ←añadir próximo evento Arrival1
```

Algorithm 9 Rutina de eventos. Salida de un cliente de la estación 1

```
RTSUM1 ← añadir tiempo de respuesta experimentado por el cliente
SERVED1 ←SERVED1 + 1
// determinar si otro cliente entra en el servidor
si (vacia(ARRIVALS1)) {
    SERVERBUSY1 ← falso
} sino {
    EVENTSLIST ← añadir próximo evento Departure1
    RTSUM1 ← añadir tiempo de espera (cola)
    ARRIVALS1 ← eliminar cliente de la cola
}
// encaminar el cliente hacia la siguiente estación (2 ó 3)
ESTACION ← determinar estación de destino (probabilidades de ramificación)
si (ESTACION == 2) {
    si (!SERVERBUSY2) {
        SERVERBUSY2 ← verdadero
        EVENTSLIST ← añadir próximo evento Departure2
    } sino {
        ARRIVALS2 ← añadir cliente a la cola
    }
} sino {
    si (!SERVERBUSY3) {
        SERVERBUSY3 ← verdadero
        EVENTSLIST ← añadir próximo evento Departure3
    } sino {
        ARRIVALS3 ← añadir cliente a la cola
    }
}
```

Algorithm 10 Rutina de eventos. Llegada de un cliente a la estación 2

```
si (!SERVERBUSY2) {
    SERVERBUSY2 ← verdadero
    EVENTSLIST ← marcar instante de salida del cliente
} sino {
    ARRIVALS2 ← poner el cliente en cola
}
EVENTSLIST ←añadir próximo evento Arrival2
```

Algorithm 11 Rutina de eventos. Salida de un cliente de la estación 2

```
RTSUM2 ← añadir tiempo de respuesta experimentado por el cliente
SERVED2 ←SERVED2 + 1
// determinar si otro cliente entra en el servidor
si (vacia(ARRIVALS2)) {
    SERVERBUSY2 ← falso
} sino {
    EVENTSLIST ← añadir próximo evento Departure2
    RTSUM2 ← añadir tiempo de espera (cola)
    ARRIVALS2 ← eliminar cliente de la cola
}
// encaminar el cliente hacia la estación 1
si (!SERVERBUSY1) {
    SERVERBUSY2 ← verdadero
    EVENTSLIST ← marcar instante de salida del cliente
} sino {
    ARRIVALS1 ← poner el cliente en cola
}
```

Algorithm 12 Rutina de eventos. Llegada de un cliente a la estación 3

```
si (!SERVERBUSY3) {
    SERVERBUSY3 ← verdadero
    EVENTSLIST ← marcar instante de salida del cliente
} sino {
    ARRIVALS3 ← poner el cliente en cola
}
EVENTSLIST ←añadir próximo evento Arrival3
```

Algorithm 13 Rutina de eventos. Salida de un cliente de la estación 3

```
RTSUM3 ← añadir tiempo de respuesta experimentado por el cliente
SERVED3 ←SERVED3 + 1
// determinar si otro cliente entra en el servidor
si (vacia(ARRIVALS3)) {
    SERVERBUSY3 ← falso
} sino {
    EVENTSLIST ← añadir próximo evento Departure3
    RTSUM3 ← añadir tiempo de espera (cola)
    ARRIVALS3 ← eliminar cliente de la cola
}
// encaminar el cliente hacia la siguiente estación (1) o hacia fuera del sistema
ESTACION ← determinar estación de destino (probabilidades de ramificación)
si (ESTACION == OUT) {
    SERVED ← SERVED + 1
} sino {
    si (!SERVERBUSY1) {
        SERVERBUSY1 ← verdadero
        EVENTSLIST ← añadir próximo evento Departure1
    } sino {
        ARRIVALS1 ← añadir cliente a la cola
    }
}
```

9. Mostrar la curva de tiempo de respuesta solicitada. Comentar los resultados obtenidos.

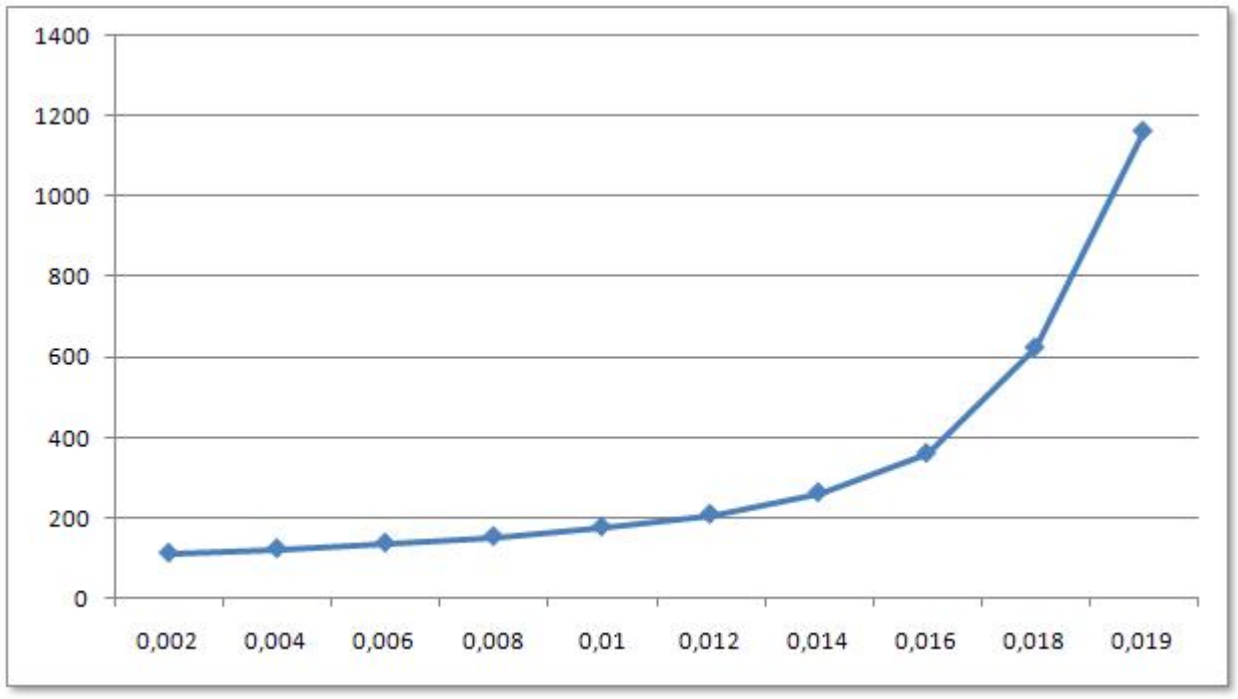

Figura 1: Curva de evolución del tiempo medio de respuesta del sistema en fución de diferentes valores del flujo medio de llegadas

λ	ω
0.002	112.32
0.004	123.54
0.006	137.33
0.008	153.88
0.010	177.83
0.012	208.99
0.014	262.90
0.016	360.80
0.018	623.39
0.019	1161.89

Cuadro 1: Datos obtenidos para 10 simulaciones. Diferentes valores de tiempo medio de respuesta del sistema (ordenadas) frente a diversos valores del tráfico global de entrada (abcisas).

Atendiendo a los resultados obtenidos podemos observar lo siguiente.

- Cómo es lógico el tiempo medio de respuesta aumenta a medida que aumenta el tráfico global de entrada. En el apartado F0 concluimos que $0 < \lambda < 0{,}02$ y en la gráfica vemos cláramente que al acercarnos a ese valor límite el tiempo medio de respuesta crece exponencialmente.