Sistemas Distribuidos de Tiempo Real

PRÁCTICAS: Distribución con Ada y CORBA

Por: J. Javier Gutiérrez gutierjj@unican.es
Héctor Pérez Tijero perezh@unican.es
http://www.ctr.unican.es/

Grupo de Computadores y Tiempo Real, Universidad de Cantabria

Práctica 1: Contadores sincronizados en monoprocesador

Escribir una aplicación sencilla con cuatro tareas periódicas con las siguientes características:

- **períodos 0.5 - 1.5 - 3.0 - 6.0 en segundos**
- **cada tarea debe llamar consecutivamente a dos funciones come tiempos que deben estar en dos paquetes separados:**

```
package Primera_Parte is
   procedure Ejecuta (Segundos : Duration);
end Primera_Parte;

package Segunda_Parte is
   procedure Ejecuta (Segundos : Duration);
end Segunda_Parte;
```

Práctica 1: Contadores sincronizados en monoprocesador (cont.)

- cada tarea debe tener un contador que inicialmente vale 0 y realizar la suma de 1 - 3 - 6 - 12 respectivamente en cada período sólo si se ha cumplido el plazo
- cada tarea debe escribir el resultado de la suma en una variable protegida
- asignar parámetros de planificación suponiendo que los plazos son iguales a los periodos

Escribir además una tarea monitor igualmente periódica de periodo 6 segundos que pinte el valor de los cuatro contadores en pantalla

El arranque de las cinco tareas debe realizarlo el programa principal y sincronizarlo con un valor concreto del reloj

Práctica 1: Contadores sincronizados en monoprocesador (cont.)

Observar el comportamiento de la ejecución monoprocesadora con distintos valores de la carga del procesador, por ejemplo: 30%, 60%, 90% y 120%

Realizar experimentos con distribuciones uniformes y no uniformes de la carga (entre tareas y entre la primera y la segunda parte)

El funcionamiento es correcto si se cumplen todos los plazos, es decir, si la tarea monitor escribe siempre valores iguales de los contadores

Anotar los valores de los experimentos y los resultados obtenidos

Práctica 2: Contadores sincronizados en distribuido

Convertir la aplicación de contadores de la práctica 1 en una versión distribuida en dos procesadores:

- categorizar el paquete `segunda_parte` como RCI y configurar dos particiones:
  - una con el procedimiento principal (y las tareas)
  - la otra con el RCI

Experimentar con los valores de ejecución correspondientes a las dos cargas mayores (90% y 120%) de la práctica 1, y después con cargas superiores

Utilizar también distribuciones uniformes y no uniformes de la carga (entre tareas y entre la primera y la segunda parte), y anotar los valores de los experimentos y los resultados obtenidos

Práctica 2: Contadores sincronizados en distribuido (cont.)

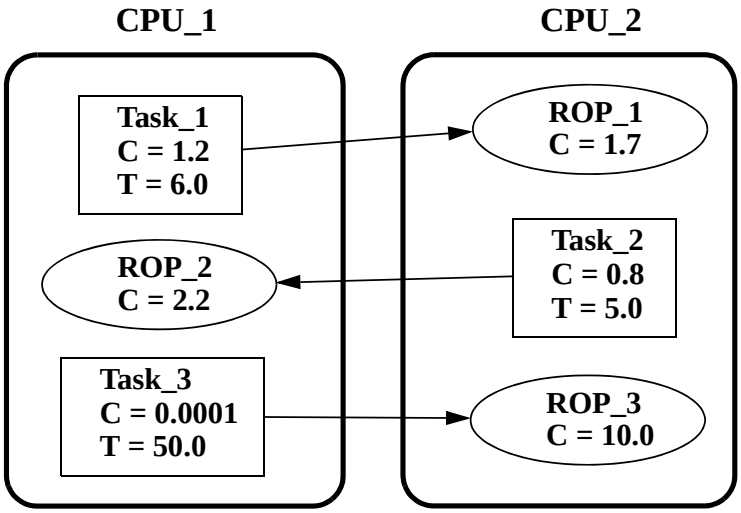
Modificar la aplicación distribuida para añadir un procesador más de manera que:

- el procesador 1 tenga el procedimiento principal y las tareas
- el procesador 2 tenga la RCI para la ejecución remota de las tareas 1 y 2
- el procesador 3 tenga otra RCI para la ejecución remota de las tareas 3 y 4

Probar con cargas del 60% y del 80% en cada procesador.

Práctica 3: Desarrollo de aplicación de tiempo real distribuida

Realizar la aplicación distribuida de tiempo real que se propone en la siguiente figura utilizando DSA, RT-EP y MaRTE OS



Práctica 3: Desarrollo de aplicación de tiempo real distribuida (cont.)

El sistema tiene las siguientes características:

- las transacciones de `Task_1` y `Task_2` son las que tienen requisitos temporales con plazos iguales a los periodos
- la transacción de `Task_3` tiene siempre la mínima prioridad en los recursos que utiliza
- asignar prioridades cualesquiera a los mensajes de las transacciones de `Task_1` y `Task_2`, y menor prioridad a los de `Task_3`

Verificar que las asignaciones de prioridad que se proponen en la siguiente tabla obtienen resultados temporales similares a los mostrados

Práctica 3: Desarrollo de aplicación de tiempo real distribuida (cont.)

Resultados correspondientes a la ejecución con MaRTE OS, RT-EP y RT-GLADE

Priority Schemes	Task_1 Prio.	ROP_1 Prio.	Task_2 Prio.	ROP_2 Prio.	Task_1 WCRT (s)	Task_2 WCRT (s)
RT-GLADE	High	Medium	High	Medium	4.26	4.81
Client Propagated	High	High	Medium	Medium	3.34	Unbounded
	Medium	Medium	High	High	8.17	3.44
Server Declared	High	Medium	High	Medium	16.81	6.18
	High	High	Medium	Medium	10.85	Unbounded
	Medium	Medium	High	High	Unbounded	3.44

Práctica 3: Desarrollo de aplicación de tiempo real distribuida (cont.)

Medir los tiempos en la CPU en la que se inicia la transacción:

- implementar la operación remota como una RPC síncrona
- pintar el tiempo sólo cuando se obtiene un caso peor

Modelar el sistema con MAST:

- hacer una estimación de los tiempos de peor caso de los mensajes
- realizar la asignación de prioridades (no ponerlas como preasignadas)
- comprobar si la asignación obtenida consigue mejores tiempos de peor caso en la aplicación real que los obtenidos en los experimentos previos

Práctica 4: Contadores sincronizados en PolyORB-CORBA

Convertir la aplicación de contadores de la práctica 1 en una versión distribuida en dos procesadores para PolyORB-CORBA (tomar como referencia el ejemplo de la suma):

- crear un objeto CORBA con la funcionalidad del paquete `segunda_parte`
- añadir al programa cliente el código necesario para usar el objeto remoto
- realizar el programa servidor para que arranque el objeto y lo declare en el servidor de nombres
- utilizar el servidor de nombres que se proporciona (o el po_cos_naming)
  - obtener su localización (IOR) para configurar el cliente y el servidor

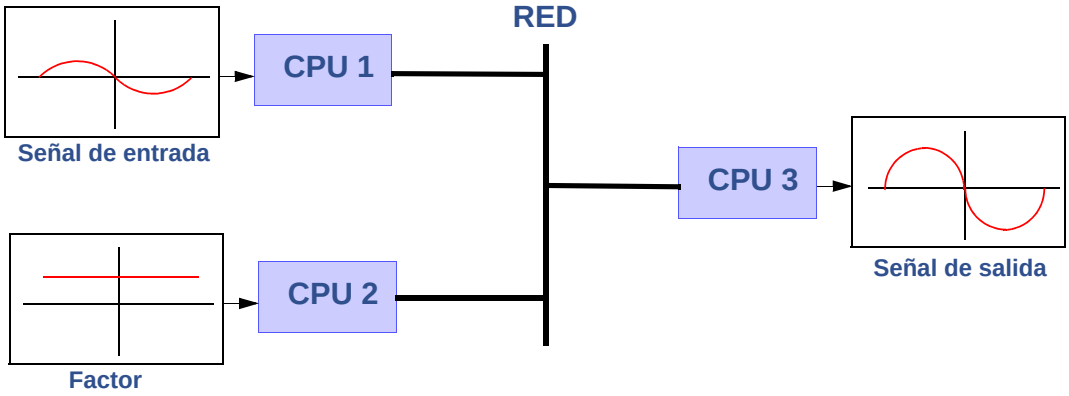
Práctica 4: Contadores sincronizados en PolyORB-CORBA

Experimentar con los valores de ejecución correspondientes a las dos cargas mayores (90% y 120%) de la práctica 1, y después con cargas superiores

Utilizar también distribuciones uniformes de la carga y anotar los valores de los experimentos y los resultados obtenidos

Práctica 5: Diseño y análisis de sistemas distribuidos de tiempo real

Se desea realizar un sistema capaz de modular la amplitud de una señal de acuerdo con un factor que viene dado por otra señal.



El sistema consta de tres procesadores (PCs) con instrumentación de I/O, que están conectados por una red Ethernet dedicada.

Práctica 5: Diseño y análisis de SDTR; requisitos

Los requisitos del sistema son:

- **La amplitud de la señal de entrada está comprendida entre 0 y 5 voltios**
- **La señal factor de modulación está comprendida entre 0 y 2 voltios**
- **Se quieren muestrear señales de entrada de hasta 50 Hz**
- **Se pretende ser sensible a cambios en el factor con un retraso máximo de 50 ms**

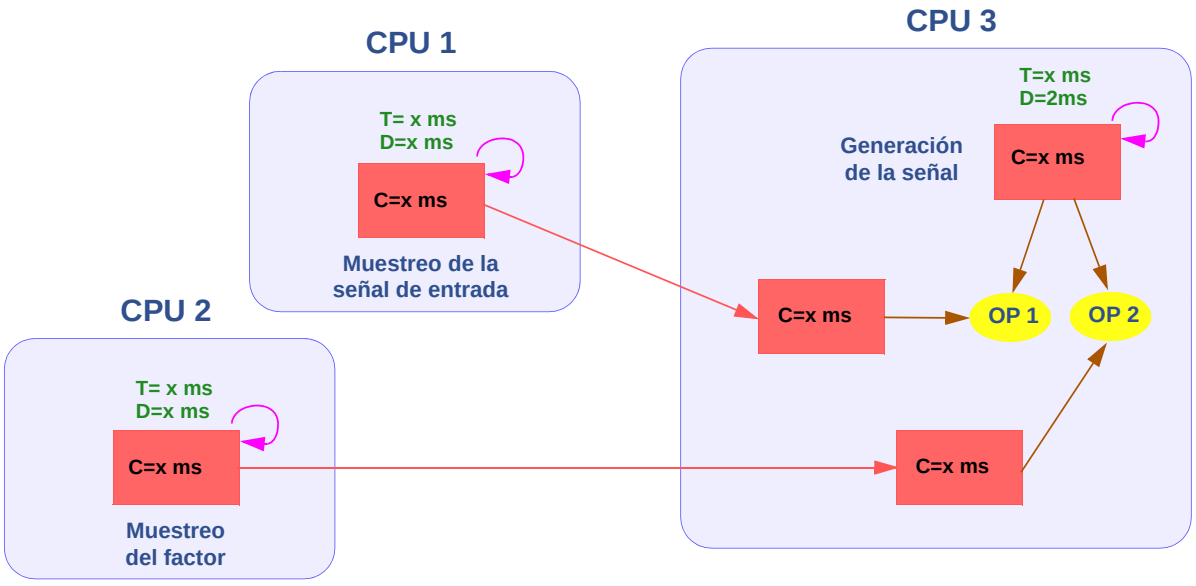
Práctica 5: Diseño y análisis de SDTR; arquitectura del sistema

El software del sistema se diseña de la manera siguiente:

- La señal de entrada se muestrea en la CPU 1 con la frecuencia adecuada, y el dato se envía a la CPU 3 y se registra en un objeto protegido PO 1
- El factor se muestrea en la CPU 2 con la frecuencia adecuada, y el dato se envía a la CPU 3 y se registra en el objeto protegido PO 2
- La generación de la nueva señal la realiza, con la frecuencia adecuada, una tarea en la CPU 3 que toma los datos de los objetos protegidos PO 1 y PO 2.

Las tres transacciones a que dan lugar y sus características se muestran en la figura siguiente.

Práctica 5: Diseño y análisis de SDTR; arquitectura del sistema (cont.)



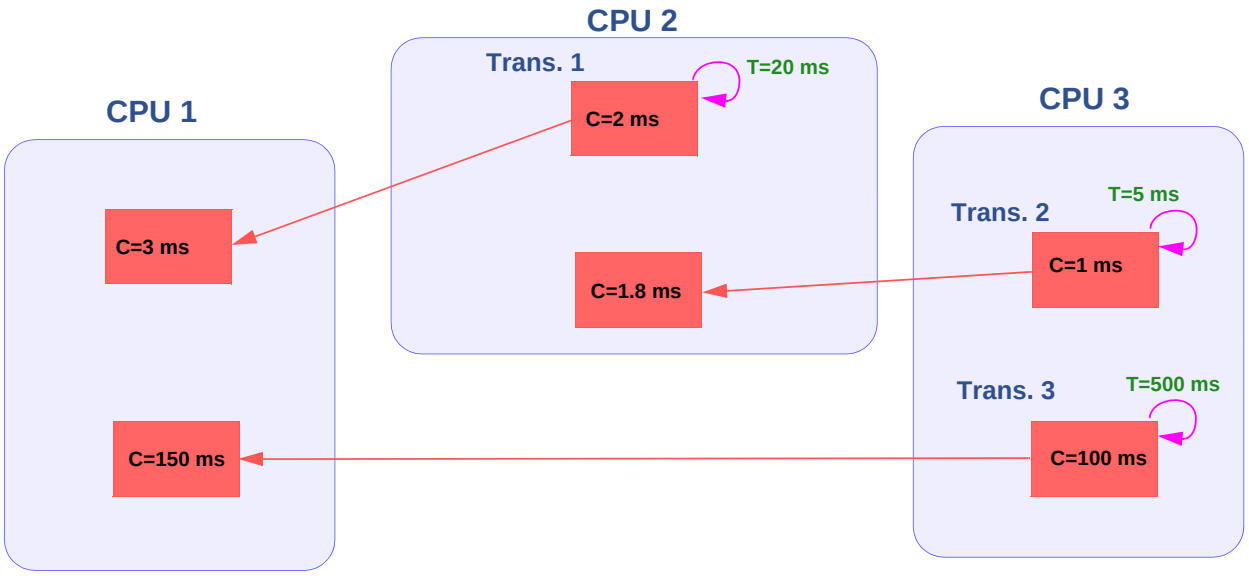
Práctica 5: Diseño y análisis de SDTR; arquitectura del sistema (cont.)

Además los procesadores ejecutan código legado, que se deberá simular, y que se corresponde con las tres transacciones de la figura siguiente:

- **Los plazos *end-to-end* de estas transacciones son iguales a los periodos.**

Suponemos que los mensajes tienen todos menor longitud que la trama Ethernet máxima y se envían en un paquete (considerar para el análisis una longitud de 1500 bytes por ejemplo).

Práctica 5: Diseño y análisis de SDTR; arquitectura del sistema (cont.)



Práctica 5: Diseño y análisis de SDTR; resolución

Se pide:

1. Establecer los parámetros de diseño que quedan por determinar: períodos y plazos en las transacciones de muestreo y generación de las señales
2. Estimar los tiempos de ejecución de peor caso implicados en dichas transacciones mediante la medida del código de las operaciones que se realizan
3. Realizar el modelo MAST del sistema
4. Asignar prioridades:
   - primero manualmente (a criterio de cada uno): analizar
   - después con la asignación de MAST: analizar

Práctica 5: Diseño y análisis de SDTR; resolución (cont.)

5. Realizar la implementación del código de la aplicación a partir del software proporcionado
6. Probar el sistema con diferentes frecuencias de la señal de entrada variando el factor y comprobar que se verifican los requisitos
7. Verificar si el sistema puede responder a frecuencias mayores de la señal de entrada:
   - sin modificar el sistema propuesto
   - modificando los parámetros de diseño (períodos y prioridades de las tareas) para tratar de buscar el límite de la frecuencia de funcionamiento

Práctica 5: Estimación de WCETs de las operaciones

Las transacciones T1, T2 y T3 del código legado disponen de toda la información para el análisis.

Las otras tres transacciones son:

- T4: muestreo de la señal
- T5: muestreo del factor
- T6: generación de la señal de salida

Los tiempos estimados de las operaciones de T4, T5 y T6 que se pueden usar en el análisis inicial (antes de medir los tiempos reales) se muestran en la tabla siguiente.

Práctica 5: Estimación de WCETs de las operaciones (cont.)

Operación	WCET (ms)	Transacción que la usa y tipo
T4_1	1.0	T4, Simple
T4_2	0.5	T4, Enclosing
T5_1	1.0	T5, Simple
T5_2	0.5	T5, Enclosing
T6_1	0.5	T6, Enclosing
OP_1	0.01	T4 y T6, operaciones del OP1, Simple
OP_2	0.01	T5 y T6, operaciones del OP2, Simple

Suponer que las operaciones de los mensajes tienen un tamaño de 1500 bytes (sólo es necesario hacer una operación Message_Transmission)