dit
UPM
Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación
Departamento de Ingeniería de Sistemas Telemáticos

EXAMEN FINAL EXTRAORDINARIO DE SISTEMAS OPERATIVOS
Convocatoria de Febrero. 13–2–98
PRIMERA PARTE (SIN LIBROS)

Puntuación y longitud *máxima* —puede de hecho ser más corta— de la respuesta, en cada pregunta. **Una página es una cara**.

DURACIÓN: 1 hora

Pregunta 1

En el sistema de ficheros plano visto en el capítulo 7 no existen operaciones de apertura y cierre. ¿A qué se debe esto? ¿Cuál es la razón para que resulte atractivo en sistemas distribuidos?

(*1/2 página; 1 punto*)

Se debe a que son operaciones no idempotentes, que no pueden realizarse con servidores sin estado, como es el caso del servidor de ficheros plano. El interés de los servidores sin estado y las operaciones idempotentes radica en que simplifican la recuperación en caso de caídas temporales, tanto de líneas de comunicación como de servidores (y clientes): Basta con reintentar las peticiones de servicio.

Pregunta 2

Explique por qué en el sistema de ficheros NFS se suele requerir que las operaciones de escritura que llegan al servidor no retornen hasta que los datos estén físicamente en el disco. Diga los inconvenientes que plantea y sugiera una solución física (*hardware*) para hacerlo innecesario.

(*1/2 página; 1 punto*)

Para que en caso de caída del servidor no se pierdan datos que el cliente crea que se han escrito. Es costoso en tiempo y puede soslayarse con un sistema de alimentación ininterrumpida más o menos complejo.

Pregunta 3

¿Para que sirven las hebras *threads* en los procesos servidores? ¿Y en los clientes?

(*1/2 página; 1 punto*)

En los procesos servidores, las hebras sirven para poder dar servicio a peticiones posteriores, cuando una petición anterior se queda bloqueada esperando que se complete una operación de entrada/salida.

En los clientes sirven, combinadas con buzones de comunicación, para desacoplar fácilmente a los flujos de procesamiento, de las peticiones bloqueantes que estos necesiten hacer a servidores.

Pregunta 4

Explique cómo se puede diferenciar la caída de un procesador de la caída de los enlaces que permiten comunicarnos con él.

(1/4 página; 0.5 puntos)

> *No se pueden diferenciar sin canales de comunicación alternativos.*

Pregunta 5

Explique qué problemas plantean las particiones de red a los sistemas de elección de coordinador.

(1/4 página; 0.5 puntos)

> *Pueden dar lugar a tantos coordinadores como particiones, pudiendo ocasionar un comportamiento incoherente.*

UPM
Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación
Departamento de Ingeniería de Sistemas Telemáticos

EXAMEN FINAL EXTRAORDINARIO DE SISTEMAS OPERATIVOS
Convocatoria de Febrero. 13–2–98
SEGUNDA PARTE (CON LIBROS)

Puntuación y longitud *máxima* —puede de hecho ser más corta— de la respuesta, en cada pregunta. **Una página es una cara**.

DURACIÓN: 1 hora y cuarto

Pregunta 1

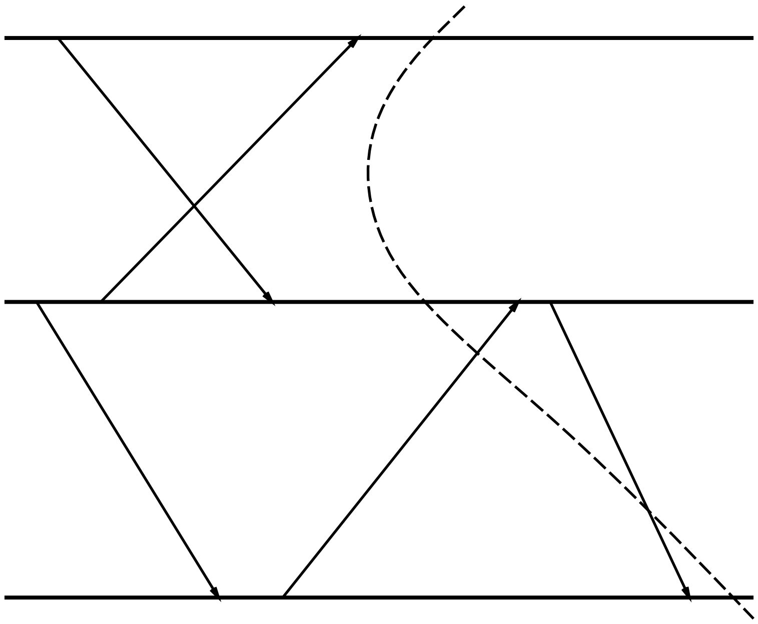

Figura 1: Computación distribuida

Para la computación distribuida que se muestra en la figura adjunta, el corte mostrado,

- ¿Es coherente? (*consistent*)

(*2 letras; 1/2 punto*)

NO

- Argumentar la respuesta anterior en base a consideraciones "geométricas" (flechas, regiones, derecha/izquierda, etc.)

(*1/4 página; 1/2 punto*)

No es coherente porque corta a un mensaje de tal forma que el origen del mismo queda a la derecha del corte, mientras que su destino (la punta de la flecha) está a la izquierda del corte.

- Argumentar lo mismo, pero esta vez en base a razonamientos basados en horas vectoriales (antes elegir los sucesos de interés y calcular sus horas vectoriales).

(1/2 página; 1,5 puntos)

Los sucesos de la "frontera" del corte tiene horas vectoriales, respectivamente, de (2, 2, 0), (1, 3, 0) y (1, 5, 3). Y sucede que el segundo componente del tercer vector es mayor que el segundo componente del segundo vector, lo cuál es una incoherencia.

- El tipo de la segunda argumentación ha sido mucho más costoso de aplicar que el primero y, sin embargo, tiene mucho más interés práctico que el primero. ¿A qué se debe esto?

(1/4 página; 1/2 punto)

Los razonamientos geométricos son difíciles de automatizar para que los realice un computador, pero, en todo caso, requieren que exista una visión global de la computación, la cuál sólo se puede conseguir mediante algoritmos numéricos del tipo de los que utilizan relojes vectoriales.

Pregunta 2

Cierto sistema operativo mantiene en el núcleo dos variables relacionadas con el tiempo:

T Número de nanosegundos transcurridos desde el comienzo de la Hégira.

i Número de nanosegundos a incrementar T cada vez que llega una interrupción. Este número debe ser 1.000.000 si el cuarzo está perfectamente calibrado, pero puede ajustarse para tener en cuenta su deriva.

Un proceso de sincronización consulta un servidor de tiempos y puede modificar cualquiera de las dos variables. Supongamos que inicialmente se pone la máquina en hora con la del servidor de tiempos y se ajusta $i = 1.000.000$. Desde ese momento el proceso de sincronización sondea cada 10 segundos al servidor de tiempos y actúa en consecuencia. Supongamos que después de la primera consulta se descubre que nuestro reloj está adelantado 10 milisegundos respecto al servidor de tiempos.

1. ¿Cuántas interrupciones por segundo tenemos realmente?
2. ¿Qué sucede si actuamos directamente sobre T, forzándolo al valor del servidor?
3. ¿Qué sucede si decrementamos i en 1000 si hay adelanto y lo incrementamos en 1000 si hay retraso?
4. ¿Qué sucede si decrementamos i en 100 por cada milisegundo de adelanto (y viceversa)?
5. ¿Qué deberíamos hacer si paramos la máquina y la encendemos al cabo de varias horas, si queremos tener el reloj lo más disciplinado posible en valor y frecuencia?
6. Si utilizamos el algoritmo de Cristian, ¿cuál debe ser el retardo máximo de ida y vuelta de una consulta al servidor para que no se añada al error más de 2 milisegundos?

(1 página; 3 puntos)

1. $\Delta T = i \times 10^{-9} \times f \times \Delta t \Longrightarrow f = \frac{\Delta T}{i \times 10^{-9} \times \Delta t} = \frac{10.00}{10^6 \times 10^{-9} \times 9.99} \approx 1001$ *int/seg*
2. *Se produce una inversión de tiempo de 10 milisegundos cada 10 segundos. El tiempo local tiene forma de diente de sierra, el reloj local está siempre adelantado un máximo de 10 milisegundos, justo antes de la resincronización.*

3. *Obviamente el método tiende a corregir el error, pero no corrige i con exactitud, por lo que el sistema tiene que oscilar permanentemente. En nuestro caso particular, pasados 10 segundos después de la primera corrección, tenemos*
   $\Delta T = 10 = i \times 10^{-9} \times f \times \Delta t \Longrightarrow \Delta t = \frac{10}{i \times 10^{-9} \times f} = \frac{10^{10}}{i \times 1001} = \frac{10^{10}}{999000 \times 1001} \approx 10.00001$ *seg*
   Es decir, hemos ajustado casi la frecuencia a la del patrón, con lo que el error se mantiene y ocasiona una segunda corrección al cabo de otros 10 *mseg:*
   $\Delta t = \frac{10^{10}}{998000 \times 1001} \approx 10.01$ *seg*
   Ahora hay un retraso de unos 10 *ms, lo que hará que al cabo de otros* 10 *ms los tiempos casi coincidan. En este caso o no se produce corrección o, aleatoriamente, se corregirá en cualquiera de los dos sentidos. Si no se corrige, al cabo de* 10 *ms el error será de unos* 10 *ms y habrá que corregir. Si se corrige equivocadamente, puede llegar a doblarse. No hay pendientes negativas y, a grosso modo, el error de mantendrá dentro de* 20 *ms por arriba o por abajo.*

4. *Corrigiendo proporcionalmente al error, la frecuencia debería irse disciplinando y llegar a tener un error muy bajo. Sin embargo, en nuestro caso estamos rozando la inestabilidad y el comportamiento asemeja al del caso anterior (sin corrección al cruzar). La corrección debería hacerse bastante menor que 100 por milisegundo; el sistema tardaría en disciplinar la frecuencia, pero lo conseguiría.*

5. *Deberíamos escribir i en un fichero al apagar. Al encender, hay que consultar al servidor de tiempos para ajustar T y leer el fichero para obtener i.*

6. *El doble del error, es decir, 4 milisegundos.*