EXAMEN PARCIAL – Bloque Unidades Didácticas 7 a 11 (04/06/2013)

APELLIDOS:		NOMBRE:	
DNI:		FIRMA:	

Este bloque tiene una puntuación máxima de **10 puntos** (que aportará 2.5 puntos a la nota global).

Indique, para cada una de las siguientes 50 afirmaciones, si éstas son verdaderas (V) o falsas (F). **Cada respuesta vale: correcta= 0.2, errónea= -0.2, vacía=0.** Al final de cada pregunta dispone de un espacio reservado para justificar mejor su respuesta, en caso de considerarlo necesario.

1. Un sistema distribuido:

	Es una clase de sistema concurrente; aunque no todos los sistemas concurrentes son distribuidos.
	Ofrece la imagen de un sistema coherente y único.
	Proporcionará diferentes tipos de transparencia. Entre ellos: transparencia en el rendimiento, en la escalabilidad, en su disponibilidad, en la seguridad...
	Es un sistema de tiempo real que necesitará un análisis de planificabilidad.

JUSTIFICACIÓN:

2. La comunicación basada en mensajes:

	Es un tipo particular de memoria compartida y requiere acceso en exclusión mutua.
	Será sincrónica cuando el canal pueda mantener los mensajes durante un intervalo indefinido.
	Se utiliza en su variante asincrónica para implantar el servicio de correo electrónico.
	Se utiliza en su variante sincrónica no persistente para implantar las llamadas a procedimiento remoto.

JUSTIFICACIÓN:

3. El algoritmo de Cristian:

	Proporciona la base necesaria para implantar cualquier algoritmo descentralizado.
	Permite sincronizar el reloj local de un nodo cliente con el mantenido por un nodo servidor.
	Es uno de los algoritmos de elección de líder más eficientes.
	Permite identificar los mensajes en tránsito por cada uno de los canales de comunicación de un sistema distribuido.

JUSTIFICACIÓN:

4. Sobre los relojes lógicos de Lamport:

	Son necesarios para deshacer los empates en el algoritmo de Chandy y Lamport.
	Ordenan de manera parcial los eventos que hayan ocurrido en un sistema distribuido.
	Permiten determinar en todos los casos si dos eventos de una traza han sido concurrentes o no.
	Si C(a)=C(b), entonces a||b.

JUSTIFICACIÓN:

5. Sobre los servicios de nombres en un sistema distribuido:

	Se necesitan para asegurar la exclusión mutua.
	Suelen utilizar una estructura jerárquica para facilitar su escalabilidad.
	Pueden retornar identificadores para facilitar la gestión de entidades móviles.
	LDAP es un ejemplo de servicio de directorio basado en atributos.

JUSTIFICACIÓN:

6. Un sistema "peer-to-peer":

F	Es un ejemplo de arquitectura software en niveles.
V	Emplea una arquitectura de sistema basada en distribución horizontal.
F	Es un ejemplo de arquitectura de sistema centralizada.
V	Es un ejemplo de sistema distribuido con buenas escalabilidades de tamaño y distancia.

JUSTIFICACIÓN:

7. Sobre tiempo lógico:

	Si “a ---> b”, entonces C(a) < C(b).
	Si “a ---> b”, entonces la ejecución del evento “a” siempre sucederá antes que la ejecución del evento “b”.
	Si “a||b”, entonces la ejecución de los eventos “a” y “b” siempre sucederá en el mismo instante de tiempo.
	Sean VT(a)=[3,4,4] y VT(b)=[5,4,8] los relojes vectoriales de dos eventos “a” y “b” en un sistema formado por tres procesos P1, P2 y P3. Podemos afirmar que “b” no ha sido ejecutado por el proceso P2.

JUSTIFICACIÓN:

8. Sobre el mecanismo de invocación a objeto remoto (ROI):

	Es el utilizado en Java RMI.
	Proporciona transparencia de ubicación.
	Admite paso de parámetros por referencia en sus invocaciones.
	Proporciona transparencia de fallos.

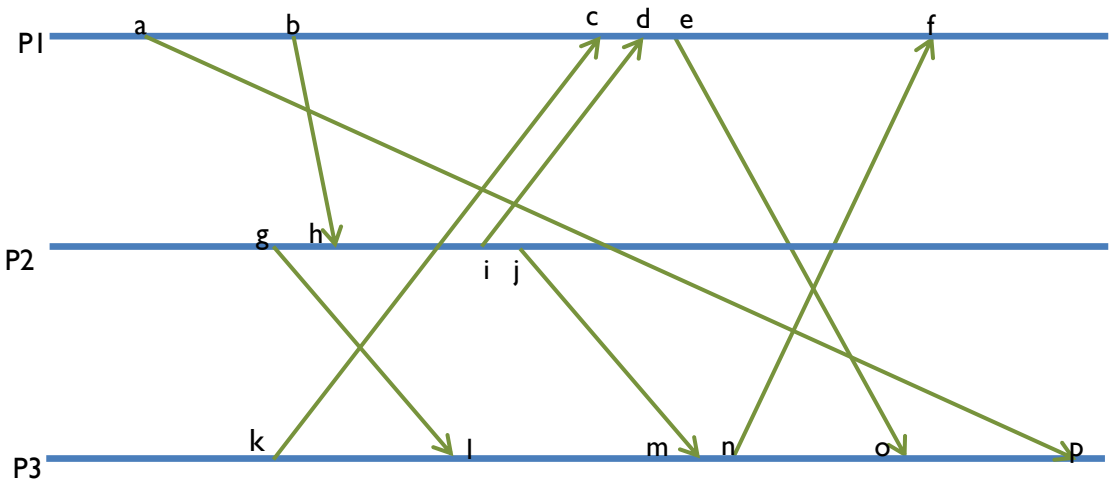
JUSTIFICACIÓN:

9. Sobre los algoritmos descritos en el tema 9 (“Sincronización en sistemas distribuidos”):

	Se describieron dos algoritmos basados en anillos: uno de exclusión mutua y otro de recolección del estado global.
	El algoritmo de Chandy y Lamport recoge el estado global del sistema.
	El algoritmo Bully resuelve el problema de la exclusión mutua en un sistema distribuido.
	Algunos algoritmos de exclusión mutua pueden utilizar relojes lógicos e identificadores para deshacer los empates que surjan.

JUSTIFICACIÓN:

10. Dado el siguiente conjunto de eventos en un sistema distribuido, asumiendo que no hay otros eventos previos:



	El reloj vectorial de “p” es VT(p)=[5,4,6] y el de “f” es VT(f)=[6,4,4].
	Los eventos “c” y “m” son concurrentes.
	El reloj de Lamport de “d” es C(d)=5 y el de “m” es C(m)=6.
	A partir de la figura podemos afirmar que “c ---> o” (“c ocurre antes que o”), pues existe un camino dirigido que va desde “c” hasta “o”.
	Si el reloj vectorial de un evento “x” fuera VT(x)=[5,4,1] y el de “f” fuese VT(f)=[6,4,4], entonces podríamos afirmar que “x ---> f”.

JUSTIFICACIÓN:

11. Sobre la gestión de recursos:

JUSTIFICACIÓN:

JUSTIFICACIÓN:

APELLIDOS:		NOMBRE:	
DNI:		FIRMA:	

Este bloque tiene una puntuación máxima de **10 puntos** (que aportará 0.5 puntos a la nota global).

Indique, para cada una de las siguientes 10 afirmaciones, si éstas son verdaderas (V) o falsas (F). **Cada respuesta vale: correcta= 1, errónea= -1, vacía=0.** Al final de cada pregunta dispone de un espacio reservado para justificar mejor su respuesta, en caso de considerarlo necesario.

1. Sobre la práctica de los filósofos comensales:

F	Se ha implementado una solución al problema, basada en la asimetría entre los filósofos que resuelve el problema rompiendo la condición de exclusión mutua.
V	Se ha implementado una solución al problema, basada en la asimetría entre los filósofos que resuelve el problema rompiendo la condición de espera circular.
F	La solución basada en la clase PhiloBothOrNone resuelve el problema de interbloqueos rompiendo la condición de no expropiación.
V	La solución basada en la clase PhiloBothOrNone resuelve el problema de interbloqueos rompiendo la condición de retención y espera.
V	Se ha desarrollado una solución al problema, basada en la limitación del número de comensales que resuelve el problema de interbloqueos rompiendo la condición de espera circular.

JUSTIFICACIÓN:

2. Sobre la práctica 4:

	Los procesos ChatServer y `rmiregistry` pueden arrancar en el mismo ordenador.
	El primer paso de todo cliente es conectarse al servidor de chat, tras lo cual contacta con el servidor de nombres.
	Podemos lanzar varios clientes en una misma máquina o en máquinas diferentes.
	Los ChatClient deben obtener la lista de canales invocando algún método del ChatServer.
	El proceso ChatClient no se registra en `rmiregistry`.

JUSTIFICACIÓN: