MODELO A

EXAMEN DE RECUPERACIÓN – SEGUNDO BLOQUE **Concurrencia y Sistemas Distribuidos**

Unidades Didácticas 7,8,9 y 11 - Prácticas 3 y 4 **23 de Junio de 2015**

Este examen tiene una duración total de 2 horas.

Este examen tiene una puntuación máxima de **10 puntos**, que equivalen a **4** puntos de la nota final de la asignatura. Indique, para cada una de las siguientes **60 afirmaciones**, si éstas son verdaderas (**V**) o falsas (**F**).

Cada respuesta vale: correcta= 1/6, errónea= -1/6, vacía=0.

Importante: Los **primeros 3 errores** **no penalizarán**, de modo que tendrán una valoración equivalente a la de una respuesta vacía. A partir del 4º error (inclusive), sí se aplicará el decremento por respuesta errónea.

Sobre los sistemas distribuidos:

1.	La disponibilidad, una de las características de los sistemas distribuidos, sirve para ocultar las diferencias de los mecanismos de comunicación. *JUSTIFICACIÓN: La disponibilidad implica que los servicios deben estar siempre disponibles (está directamente relacionada con el fallo del sistema, no con la transparencia de acceso).*
2.	La transparencia de replicación oculta la coordinación entre las actividades que gestionan un conjunto de recursos para mantener su consistencia. *JUSTIFICACIÓN: La definición dada se corresponde con la transparencia de transacción.*
3.	Los sistemas distribuidos basados en Active Directory proporcionan escalabilidad administrativa.
4.	La capa de middleware ayuda a conseguir los cuatro objetivos principales de los sistemas distribuidos: descentralización, replicación, transparencia de acceso y escalabilidad. *JUSTIFICACIÓN: Los objetivos indicados son incorrectos. Los cuatro objetivos son: facilitar acceso a los recursos remotos, transparencia de distribución, sistema abierto y escalabilidad.*

Respecto a los algoritmos descentralizados:

5.	Si hay un nodo que mantiene toda la información relevante de un algoritmo, pero este nodo toma decisiones basadas en su conocimiento local, entonces el algoritmo es descentralizado. *JUSTIFICACIÓN: Para que sea descentralizado, ningún nodo debe mantener toda la información completa que necesite el algoritmo.*
6.	La utilización de algoritmos descentralizados permite ofrecer transparencia de acceso y de ubicación, al estar distribuida la carga del algoritmo entre diferentes nodos. *JUSTIFICACIÓN: Esta distribución de la carga del algoritmo por sí misma no ofrece los tipos de transparencia indicados.*
7.	La utilización de algoritmos descentralizados permite mejorar la escalabilidad de distancia.
8.	Para mejorar la escalabilidad de tamaño se suele utilizar distribución de responsabilidades, replicación, caching y algoritmos descentralizados.

Sobre el mecanismo de comunicación ROI:

9.	Se pueden pasar objetos remotos por referencia.
10.	Cuando un cliente referencia por vez primera a un objeto remoto, obtiene el esqueleto para dicho objeto. *JUSTIFICACIÓN: Obtiene el proxy.*
11.	El esqueleto ofrece la misma interfaz que el objeto remoto. *JUSTIFICACIÓN: El proxy es quien ofrece la misma interfaz que el objeto remoto.*

Sobre el mecanismo de comunicación Java RMI:

12.	Si un objeto que se pasa como argumento implementa la interfaz *Remote*, entonces dicho objeto se pasa por referencia.
13.	Para implementar un objeto remoto en Java RMI, la clase de los objetos remotos debe implementar la interfaz remota y extender *java.rmi.server.UnicastRemoteObject*, para así poder registrar los objetos en el ORB de Java.
14.	El servidor de nombres de Java RMI guarda, para cada objeto remoto, su nombre simbólico y la dirección (host, puerto) de su proxy. *JUSTIFICACIÓN: El servidor de nombre guarda el nombre simbólico y la referencia.*
15.	El servidor de nombres de Java RMI puede residir en cualquier nodo (incluso en el nodo cliente) y es accedido usando la interfaz Registry.
16.	Java RMI es un ejemplo de middleware de mensajería, donde cliente y servidor deben suscribirse al middleware para así poder enviarse mensajes entre sí. *JUSTIFICACIÓN: Cliente y servidor no requieren subscripción al middleware. Esto se realiza en Java JMS.*

Respecto a los servicios Web RESTful y al mecanismo de comunicación Java Message Service

17.	Resulta interesante utilizar servicios Web RESTful cuando no es necesario que todos los componentes de la aplicación estén simultáneamente en ejecución. *JUSTIFICACIÓN: La comunicación no es persistente, por lo que los componentes deben estar simultáneamente en ejecución.*
18.	Los servicios Web RESTful utilizan métodos del protocolo http para indicar el tipo de operación.
19.	En JMS, los mensajes no son estructurados y se envían en texto plano en XML. *JUSTIFICACIÓN: Sí son estructurados, con cabecera, contenido y campos predefinidos.*
20.	JMS generalmente utiliza comunicación indirecta.
21.	Los objetos que implementan la interfaz JMSContext se crean a partir de una factoría de conexiones.

Sobre los algoritmos vistos en esta asignatura de exclusión mutua y elección de líder para sistemas distribuidos:

22.	El tiempo de propagación de los mensajes debe ser conocido en el algoritmo Bully para decidir hasta cuándo se esperará que el token regrese al nodo iniciador. *JUSTIFICACIÓN: En el algoritmo Bully no se utiliza ningún token.*
23.	En el algoritmo centralizado de exclusión mutua un nodo accede a la sección crítica cuando han dado su permiso todos los demás nodos. *JUSTIFICACIÓN: La explicación se corresponde con el algoritmo distribuido de exclusión mutua.*
24.	En el algoritmo de exclusión mutua para anillos no se admite que haya simultáneamente más de un token para una misma sección crítica.
25.	Los algoritmos de elección de líder gestionan correctamente la situación en la que los relojes físicos de los nodos que participan en el algoritmo no están sincronizados. *JUSTIFICACIÓN: En estos algoritmos solamente se requiere conocer el identificador de los nodos. No se utilizan relojes de ningún tipo.*

26.	Alguno de los algoritmos de exclusión mutua vistos en la asignatura requiere que el sistema utilice relojes vectoriales para ordenar sus eventos. *JUSTIFICACIÓN: Se requiere el uso de relojes lógicos de Lamport y de los identificadores de los nodos para poder establecer un orden total de los eventos.*	]
27.	En el algoritmo de exclusión mutua para anillos, cuando un nodo quiere entrar en la sección crítica, envía un mensaje TRY al resto de nodos para que el propietario actual del token se lo envíe una vez haya terminado su sección crítica. *JUSTIFICACIÓN: El token se va pasando por el anillo y quien tiene el token puede entrar en la sección crítica.*	]

Sobre los algoritmos vistos en esta asignatura de sincronización de relojes físicos:

28.	El algoritmo de Cristian requiere un nodo con un reloj más exacto que el resto.	
29.	Una vez finalizado el algoritmo de Cristian, se ajusta la hora del nodo que actúa como servidor. JUSTIFICACIÓN: Esto ocurre en el algoritmo de Berkeley.	
30.	Resulta adecuado utilizar el algoritmo de Berkeley para conseguir que todos los relojes de los nodos de un sistema distribuido se sincronicen entre sí en un momento determinado.	

Respecto a los relojes lógicos:

31.	Supongamos dos eventos `a,b` en un mismo nodo, etiquetados respectivamente con valores lógicos `C(a)` y `C(b)`: si `C(a) < C(b)` podemos afirmar que `a->b`	
32.	Supongamos dos eventos `a,b` en nodos distintos, etiquetados respectivamente con valores lógicos `C(a)` y `C(b)`: si `C(a) < C(b)` podemos afirmar que `a->b`	
33.	Supongamos dos eventos `a,b` en nodos distintos, etiquetados respectivamente con valores lógicos `C(a)` y `C(b)`: si `a->b` podemos afirmar que `C(a) < C(b)`	

Dados 3 eventos `a,b,c` etiquetados respectivamente con relojes vectoriales con valores `Va=[4,1,2], Vb=[3,0,5], Vc=[3,1,6]:`

34.	Podemos afirmar que `a || c` *JUSTIFICACIÓN: No se cumple que Va<Vc ni tampoco que Vc<Va, por lo que ambos eventos son concurrentes.*	
35.	Aunque los eventos `b` y `c` ocurran en nodos distintos, podemos afirmar que `b->c` *JUSTIFICACIÓN: Se cumple que Vb<Vc por lo que b->c*	
36.	A diferencia de los relojes lógicos de Lamport, no se cumple la transitividad (si `m->n` y `n->p`, no implica `m->p`).	

Respecto al algoritmo de Chandy-Lamport para la obtención del estado global del sistema:

37.	Un corte se considera inconsistente si incluye mensajes enviados y todavía no recibidos. *JUSTIFICACIÓN: La explicación dada se corresponde con un corte consistente.*	
38.	Se requiere conectividad total: entre cada par de nodos a,b deben existir canales unidireccionales en sentido a-b y b-a.	
39.	Únicamente envía mensajes MARCA el nodo iniciador. *JUSTIFICACIÓN: Cuando un nodo recibe un mensaje MARCA y no ha guardado aún su estado local, entonces envía MARCA al resto de nodos.*	
40.	Cuando un proceso recibe un mensaje MARCA, debe guardar su estado local sólo si es la primera vez que recibe dicho mensaje.	

Respecto a la práctica del Chat distribuido en Java RMI:

54.	Los procesos *ChatClient* registran su objeto *ChatUser* mediante el servidor de nombres (*rmiregistry*). De esa manera el proceso *ChatServer* puede obtener referencias de los usuarios conectados
55.	En la aplicación de chat hay invocaciones remotas en las que los dos procesos involucrados son procesos *ChatClient*.
56.	Cuando un usuario se une a un canal, el canal lo notifica a todos los usuarios conectados al canal, invocando sobre ellos un método (invocación remota).
57.	*rmiregistry* es un componente middleware de java RMI que permite en esta práctica encontrar a todos los usuarios de un canal
58.	Los proxy de los objetos los podemos obtener haciendo una operación *lookup* en un objeto de tipo *Registry*.
59.	En la práctica, desde el lado del cliente se pueden crear nuevos canales con la interfaz del usuario.
60.	Para localizar el objeto *ChatServer* hay que hacer una búsqueda en la que se suministra la dirección IP y el puerto del ordenador donde se ejecuta el *ChatServer*.