EXAMEN RECUPERACIÓN SEGUNDO BLOQUE **Concurrencia y Sistemas Distribuidos**
Unidades Didácticas 7, 8 y 9 - Prácticas 4 y 5. **Fecha: 7 de Junio de 2019**
Este examen tiene una duración total de 90 minutos.

Este examen tiene una puntuación máxima de **10 puntos**, que equivalen a **3.5** puntos de la nota final de la asignatura. Consta tanto de preguntas de las unidades didácticas como de las prácticas.
Indique, para cada una de las siguientes **55 afirmaciones**, si éstas son verdaderas (**V**) o falsas (**F**).

Cada respuesta vale: correcta= 10/55, errónea= -10/55, vacía=0.

Sobre la transparencia en los sistemas distribuidos:

1. Los detectores de fallos y la replicación son mecanismos para lograr transparencia de fallos.
2. La transparencia de concurrencia persigue ocultar la ubicación de los recursos.
3. La transparencia de persistencia trata de ocultar el hecho de que los recursos estén almacenados de forma no volátil.
4. La transparencia en sistemas distribuidos suele implicar coste para lograrla y suele implicar mayor calidad observada por los usuarios, respecto a no ofrecer tal transparencia.
5. Los principales ejes de la transparencia de distribución son tres: transparencia de ubicación, transparencia de fallos y transparencia de replicación.

Respecto a la disponibilidad en los sistemas distribuidos:

6. Los únicos factores que afectan a la disponibilidad son los fallos y las tareas de mantenimiento.
7. El principal mecanismo para lograr tolerancia a fallos es la replicación.
8. Los fallos bizantinos son fallos compuestos.
9. Las particiones son fallos detectables, pues los nodos pueden saber en todo momento si el sistema está particionado en dos o más particiones en ejecución.

Sobre los modelos de replicación:

10. En la replicación pasiva, la réplica primaria envía mensajes de actualización de estado (*checkpoint*) a las otras réplicas.
11. La replicación activa requiere de menor trabajo de reconfiguración en caso de fallos que la replicación pasiva.
12. La replicación pasiva suele ser más eficiente que la replicación activa durante la operativa del sistema en ausencia de fallos.
13. La replicación activa requiere que las réplicas se ejecuten de acuerdo a un modelo de ejecución determinista.
14. La replicación activa requiere del empleo de algoritmos de difusión ordenada de mensajes.

Sobre la escalabilidad:

15. Entre las técnicas más importantes para lograr escalabilidad, podemos mencionar replicación y distribución de la carga entre diferentes nodos.
16. Un sistema que emplee caching generalmente será menos escalable que otro sistema que no emplee dicha técnica.
17. Los sistemas altamente escalables suelen garantizar consistencia fuerte.
18. La técnica de distribuir datos mejora la escalabilidad.

Sobre los mecanismos de comunicación en los sistemas distribuidos:

19. En la comunicación asíncrona, el middleware responde al emisor tras recibir aviso del receptor de haber procesado el mensaje.	
20. JMS ofrece una comunicación fuertemente acoplada, ya que los clientes deben establecer conexión entre sí a través de las factorías de conexión.	
21. En JMS se emplea sincronización síncrona, de modo que el emisor debe esperar a que el receptor reciba el mensaje.	
22. El mecanismo de comunicación JMS resulta útil cuando los componentes del sistema no necesitan recibir una respuesta inmediata a sus mensajes para poder continuar su funcionamiento.	
23. El mecanismo de comunicación ROI proporciona transparencia de ubicación.	
24. En ROI el emisor envía mensajes a una cola y el receptor recibe mensajes de dicha cola, a la que se ha suscrito.	

Sobre los mecanismos de comunicación ROI y Java RMI:

25. Una vez el objeto remoto ha finalizado la ejecución del método invocado, el proxy empaqueta los argumentos de la llamada a dicho método.	
26. Java RMI es una API de comunicación especificada en múltiples lenguajes de programación, tales como Java, Javascript, C#, Python, etc.	
27. En una invocación remota (ROI), los objetos invocador e invocado residen siempre en procesos diferentes, independientemente de si los procesos están ubicados en el mismo nodo o en nodos diferentes.	

Sobre los algoritmos de sincronización de relojes físicos:

28. El algoritmo de Cristian para la sincronización de relojes se basa en el uso de un servidor con un reloj más exacto en el cual todos confían.	
29. El algoritmo de Cristian sólo puede aplicarse si el tiempo que necesita el servidor para procesar el mensaje es 10 milisegundos.	
30. En el algoritmo de Berkeley, no es necesario conocer a priori el retardo medio de los mensajes que intercambian los distintos nodos.	
31. El algoritmo de Berkeley permite ajustar todos los relojes a una misma hora, pero no garantiza que se trate de la hora exacta (real).	

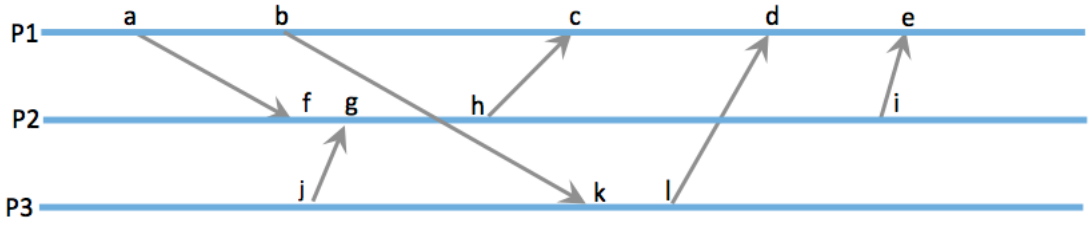
Sobre los relojes lógicos de Lamport y los relojes vectoriales:

32. El reloj lógico de Lamport asocia un valor entero a cada evento (ej. para evento x hablamos de L(x)), de forma que si para dos eventos a y b cualesquiera se cumple L(a)<L(b), entonces a->b.	
33. Haciendo uso de los relojes lógicos de Lamport y de los identificadores de los nodos se puede establecer un orden total entre los distintos eventos.	
34. Si dos eventos a y b tienen relojes vectoriales asociados V(a)=[3,2,2] y V(b)=[2,4,1], podemos afirmar que a||b.	

Sobre el algoritmo de Chandy-Lamport:

35. Por sus características, el algoritmo de Chandy-Lamport puede considerarse descentralizado.
36. En el algoritmo de Chandy-Lamport se pueden recibir varios mensajes MARCA por un mismo canal.

Dado el siguiente cronograma que muestra la ejecución de tres procesos en un sistema distribuido, cada uno en un nodo distinto:



37. Los relojes de Lamport de los eventos "f" y "b" son iguales, por lo que estos dos eventos son concurrentes.
38. El reloj de Lamport del evento "i" es 5.
39. El reloj vectorial en "h" es V(h)=[1,3,1]
40. El reloj vectorial de "k" es V(k)=[2,0,2].

Sobre los algoritmos de exclusión mutua y consenso:

41. En el algoritmo distribuido de exclusión mutua, todo nodo necesita mantener una lista de respuestas OK pendientes.
42. En el algoritmo centralizado de exclusión mutua, cuando el propietario de la sección crítica ejecuta el protocolo de salida, éste consiste en difundir "OK" a los restantes nodos.
43. En el algoritmo de exclusión mutua para anillos, si un nodo que desea entrar a la sección crítica recibe el token, utiliza relojes lógicos (Lamport) para determinar si tiene prioridad sobre otros que también desean entrar.
44. Un ejemplo de algoritmo de consenso en ausencia de fallos consiste en que cada nodo difunde su propuesta, y todos eligen la propuesta del nodo con menor identificador.

Sobre los algoritmos de elección de líder:

45. El algoritmo Bully falla si más de un proceso inicia el algoritmo de forma quasi-simultánea (o sea, antes de recibir el mensaje del otro iniciador).
46. El algoritmo Bully requiere que cada nodo disponga de un identificador único, que además debe ser conocido por los otros nodos.
47. El algoritmo de elección de líder en anillo falla si más de un proceso inicia el algoritmo de forma quasi-simultánea (o sea, antes de recibir el mensaje del otro iniciador).

Sobre la práctica de Java RMI:

48. El proceso ChatServer implementa la clase MessageListener para poder recibir las peticiones de los ChatClient para conectar a los ChatUser.	
49. El orden de lanzamiento de los procesos para iniciar el chat distribuido con un ChatRobot es el siguiente: rmiregistry, ChatRobot, ChatServer y tantos ChatClient como se desee.	
50. Los objetos que representan a los canales se registran en el servidor de nombres.	
51. Para enviar un mensaje privado entre dos clientes, es necesario que intervenga ChatServer para la comunicación entre estos dos clientes.	

Sobre la práctica de JMS:

52. La creación de una cola temporal para que el cliente (CsdMessengerClient) reciba la respuesta inicial del servidor (CsdMessengerServer) la realiza CsdMessengerServer.	
53. Ante la conexión de un nuevo usuario, CsdMessengerClient recibe un mensaje con la lista de usuarios.	
54. Las colas JMS asociadas a cada Cliente (CsdMessengerClient) las crea cada cliente una vez se ha identificado ante CsdMessengerServer.	
55. En la aplicación de chat se transmiten distintos objetos por medio de mensajes de tipo *ObjectMessage*.	