MODELO A

EXAMEN SEGUNDO PARCIAL **Concurrencia y Sistemas Distribuidos**
Unidades Didácticas 7 a 9 - Prácticas 4 y 5 **Fecha: 27 de Mayo 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 **60 afirmaciones**, si éstas son verdaderas (**V**) o falsas (**F**).

Cada respuesta vale: correcta= 10/60, errónea= -10/60, vacía=0.

Sobre los sistemas distribuidos:

1. Un sistema distribuido es un conjunto de ordenadores independientes que ofrecen a sus usuarios la imagen de un sistema coherente único.	V
2. De forma general, los fallos compuestos se tratan de igual forma a la aparición de varios fallos simples de forma consecutiva.	V
3. En un sistema distribuido, a nivel hardware las máquinas comparten entre sí los recursos (memoria, reloj, disco, etc...).	F
4. Los fallos, las tareas de mantenimiento y los ataques maliciosos son tres factores que afectan a la escalabilidad del sistema.	F
5. Para lograr transparencia de ubicación, los recursos requieren estar identificados con nombres simbólicos únicos.	V

Sobre la escalabilidad y disponibilidad de los sistemas distribuidos:

6. En general, la técnica de replicación permite aumentar tanto la escalabilidad del sistema como su disponibilidad.	V
7. La técnica de *caching* es un caso particular de la replicación donde el cliente mantiene una réplica exacta, con consistencia fuerte, de los datos que mantiene el servidor.	F
8. La distribución del procesamiento entre diferentes nodos y el particionado de los datos permite aumentar la escalabilidad del sistema.	V
9. El teorema CAP nos indica que, en un sistema a gran escala, donde las particiones ocurren, se debe sacrificar disponibilidad del sistema, o bien su consistencia.	V
10. En la replicación activa, en caso de fallo de una réplica, el trabajo de reconfiguración consiste en que la réplica primaria enviará un mensaje menos de *checkpoint*.	F

Sobre los mecanismos de comunicación ROI y Java RMI:

11. Se considera que una invocación a un objeto es una invocación local cuando los objetos invocador e invocado residen en procesos diferentes del mismo nodo.	F
12. En ROI, para el paso de objetos como argumentos por valor, se utiliza la técnica de serialización de objetos para así transmitirlos al nodo destino de la invocación.	V
13. En Java RMI, un objeto invocable de forma remota debe implementar la interfaz `java.rmi.Remote`.	V
14. Java RMI es un middleware de comunicación que proporciona comunicación asincrónica.	F
15. Java RMI proporciona la interfaz `Registry` para que tanto el cliente como el servidor interaccionen con un servidor de nombres usando métodos como `lookup`, `bind`, `rebind` …	V
16. El servidor de nombres de Java RMI almacena para cada objeto registrado, su proxy y su esqueleto.	F

Sobre la escalabilidad y disponibilidad de los sistemas distribuidos:

17. El servicio de pertenencia a grupo permite determinar quiénes son los nodos que están activos, y utiliza para ello el algoritmo de elección de líder en anillo.	F
18. Los detectores de fallos, el servicio de pertenencia a grupo y la replicación son mecanismos que permiten aumentar la disponibilidad y lograr tolerancia a fallos.	V
19. Se produce una partición en un sistema distribuido cuando se producen varios fallos en nodos o canales de comunicación que dejan al sistema dividido en dos o más subgrupos.	V
20. Tenemos un sistema con varios nodos que miden la temperatura ambiente y se la transmiten a un nodo coordinador. El coordinador puede determinar como fallo simple detectable la situación en la que un nodo tarda mucho tiempo en contestar.	V
21. Tenemos un sistema con varios nodos que miden la temperatura ambiente y se la transmiten a un nodo coordinador. El coordinador puede utilizar la replicación (con varios nodos midiendo la temperatura de la misma zona) para poder detectar los fallos bizantinos.	V
22. Los fallos de parada no pueden detectarse directamente por otros nodos. Se requiere de algoritmos de consenso o de quórum para poder tratar este tipo de fallos.	F

Sobre los mecanismos de comunicación en los sistemas distribuidos:

23. Por comunicación directa se entiende que un proceso se envía un mensaje a sí mismo directamente, y por comunicación indirecta que se lo envía a otro proceso diferente.	F
24. En el envío de un mensaje en un sistema con comunicación sincrónica en la respuesta, el emisor esperará hasta que el receptor haya procesado el mensaje y devuelva la respuesta.	V

Respecto a Java Message Service:

25. Java JMS es una API Java que permite a las aplicaciones invocar a objetos remotos mediante el envío y recepción de mensajes.	F
26. Los componentes de JMS son los proveedores, los clientes, los mensajes y los objetos administrados.	V
27. Cuando se quiere que los componentes de una aplicación distribuida no dependan de conocer las interfaces de otros componentes es preferible usar Java RMI frente a JMS.	F
28. En JMS un mismo mensaje puede ser entregado a varios clientes.	V
29. En JMS, en general, para que un cliente A pueda enviar un mensaje a otro cliente B, el cliente B debe estar activo.	F

Sobre los algoritmos de consenso:

30. El algoritmo de consenso tolerante a fallos no soporta fallos bizantinos.	V
31. En el algoritmo de consenso en ausencia de fallos, el valor que deciden los nodos como decisión final es el promedio de los valores propuestos por los distintos nodos.	F
32. El algoritmo de consenso tolerante a fallos con N nodos requiere un total de N rondas.	F
33. En cada ronda del algoritmo de consenso tolerante a fallos deben ajustarse los *timeouts* para reducir la cantidad de situaciones detectadas como fallo cuando en realidad no lo son.	V

Sobre los algoritmos de sincronización de relojes físicos:

34. En el algoritmo de Cristian, para fijar la hora del cliente se calcula el promedio entre el reloj del cliente y el del servidor.	F
35. En el algoritmo de Berkeley cada cliente puede sincronizar su reloj con independencia del resto de clientes.	F
36. El reloj de un nodo nunca debe adelantarse.	F

Sobre los relojes lógicos de Lamport y los relojes vectoriales:

37. Asumiendo relojes lógicos de Lamport, si a->b implica que C(a)<C(b), donde C(x) representa el valor de contador asociado con el evento x.	V
38. Con relojes lógicos vectoriales, si V(a)=[0,0,1] y V(b)=[2,2,0], entonces a || b	V
39. Con relojes lógicos vectoriales, V(a)<V(b) implica que a->b	V
40. En el algoritmo vectorial, para cualquier par de vectores distintos V(a) V(b), se cumple que V(a)<V(b) o V(b)<V(a)	F
41. Con relojes lógicos de Lamport se garantiza que dos eventos correspondientes a nodos distintos no pueden tener asociado el mismo valor lógico.	F

Sobre los algoritmos de elección de líder vistos en clase:

42. En el algoritmo de elección de líder con topología en anillo, el líder es aquel nodo que posee el token.	F
43. En el algoritmo de Bully, todo nodo que recibe OK sabe que no será elegido como líder.	V

Sobre el algoritmo de Chandy-Lamport:

44. Chandy-Lamport sólo funciona si hay conectividad total y todos los canales son Fiables y FIFO.	V
45. El algoritmo de Chandy-Lamport garantiza que la instantánea que se obtiene es consistente.	V
46. Chandy-Lamport falla si más de un proceso inicia el algoritmo de forma quasi-simultánea (o sea, antes de recibir el mensaje marca del otro iniciador).	F

Sobre los algoritmos de exclusión mutua:

47. En el algoritmo centralizado, el coordinador debe mantener una lista de respuestas pendientes (nodos que han solicitado acceso al recurso, pero a los que todavía no se les ha contestado).	V
48. En el algoritmo centralizado, en caso de que falle el coordinador, podemos aplicar el algoritmo de Bully para elegir a otro.	V
49. En el algoritmo de exclusión mutua distribuido, ningún nodo necesita mantener una lista de respuestas pendientes (nodos que han solicitado acceso al recurso, pero a los que todavía no se les ha contestado).	F
50. En el algoritmo de exclusión mutua distribuido, el protocolo de salida consiste en difundir ok a todos los nodos.	F
51. En el algoritmo de exclusión mutua en anillo, sólo puede acceder a la SC el nodo que tiene el token.	V
52. En el algoritmo de exclusión mutua en anillo, si un nodo que no desea entrar a la SC crítica recibe el token, se limita a pasarlo al siguiente.	V

Sobre la práctica 4 (Chat distribuido orientado a objetos basado en RMI):

53. El proceso `ChatRobot`, al no necesitar la interfaz gráfica, no requiere de un método `main` para lanzar el propio proceso.	F
54. Los mensajes recibidos por el usuario se almacenan en dos colas diferentes, la cola de mensajes privados y la cola de mensajes del canal.	F
55. Respecto a la interfaz `MessageListener` y su método `messageArrived`(), `ChatClient` recibe los mensajes debido a que implementa dicha interfaz, y el tratamiento de los mensajes se realiza en dicho método.	V
56. Los procesos `ChatClient` reciben notificaciones de los mensajes que se envían, de esta forma pueden realizar las peticiones necesarias al servidor de nombres para obtener su contenido, destinatario y origen.	F

Sobre la práctica 5 (Java Message Service):

57. Para enviar un mensaje privado, un usuario usa la cola JMS asociada al usuario del destinatario.	V
58. `CsdMessengerServer` devuelve en la cola temporal una lista con los usuarios y sus colas asignadas ante la conexión de un nuevo usuario.	F
59. La recepción de mensajes de cada usuario en la cola que tiene asignada "users-nombreusuario" implica la creación de un nuevo contexto JMS, dado que se realiza en un hilo distinto.	V
60. `NewChatMessage` es un objeto de la clase definida en el API de JMS que se usa para transmitir mensajes entre usuarios.	F