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:

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

2. La comunicación basada en mensajes:

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

3. El algoritmo de Cristian:

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

4. Sobre los relojes lógicos de Lamport:

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

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

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

6. Sobre tiempo lógico:

V	Si "a ---> b", entonces C(a) < C(b).
V	Si "a ---> b", entonces la ejecución del evento "a" siempre sucederá antes que la ejecución del evento "b".
F	Si "a||b", entonces la ejecución de los eventos "a" y "b" siempre sucedará en el mismo instante de tiempo.
V	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.

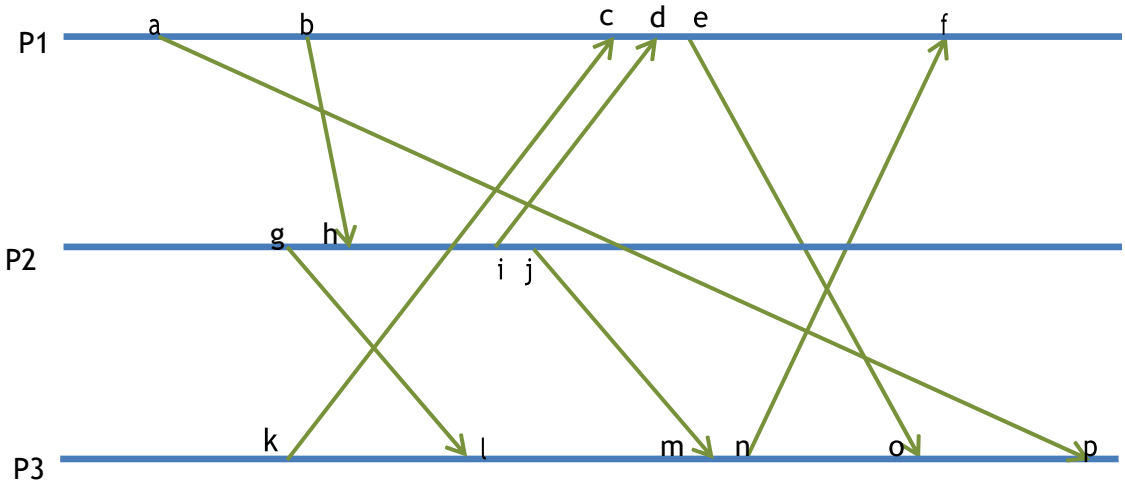
7. Sobre el mecanismo de invocación a objeto remoto (ROI):

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

8. Sobre los algoritmos descritos en el tema 9 ("Sincronización en sistemas distribuidos"):

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

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



V	El reloj vectorial de "p" es VT(p)=[5,4,6] y el de "f" es VT(f)=[6,4,4].
V	Los eventos "c" y "m" son concurrentes.
V	El reloj de Lamport de "d" es C(d)=5 y el de "m" es C(m)=6.
V	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".
V	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".

1. Sobre la práctica 4:

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

EXAMEN 2do PARCIAL – Bloque Unidades Didácticas 7 a 11 9 de Junio de 2014 Concurrencia y Sistemas Distribuidos **Modelo A**

APELLIDOS:				**NOMBRE:**	
DNI:		**GRUPO:**		**FIRMA:**	

Este bloque tiene una puntuación máxima de **10 puntos,** que equivalen a 2.5 puntos de la nota final de la asignatura. 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.**

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.

1. Sobre los sistemas distribuidos:

F	A.- Todos los sistemas de tiempo real son ejemplos de sistemas distribuidos.
V	B.- Algunos tipos de transparencia pueden comprometer la eficiencia del sistema, por ejemplo introduciendo retardos en las interacciones entre componentes de un sistema distribuido.
V	C.- Un sistema distribuido ofrece la imagen de un sistema coherente y único.
F	D.- Los sistemas distribuidos proporcionan diferentes tipos de transparencia. Entre ellos: transparencia en el rendimiento, en la escalabilidad, en su disponibilidad, en la seguridad...
F	E.- Si se desea que un sistema distribuido ofrezca escalabilidad de tamaño, no se debe hacer uso de técnicas de replicación, ya que al actualizar el estado de un componente tendremos que propagar tal actualización a todas las réplicas.

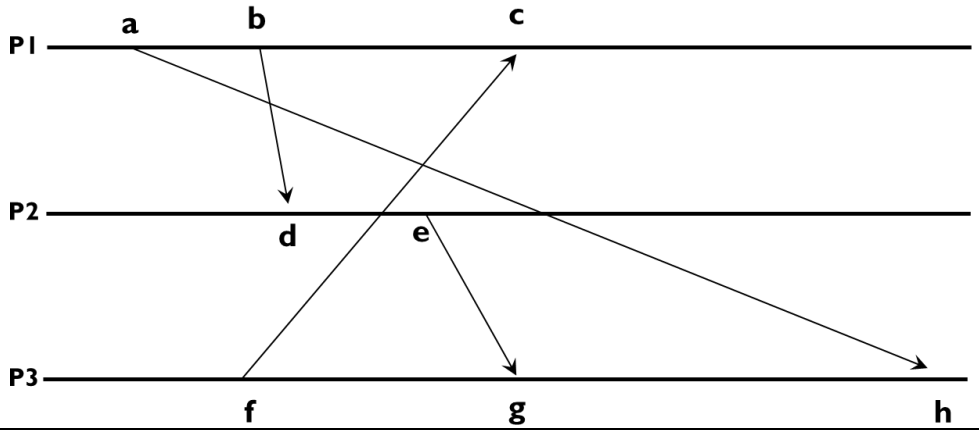
2. Sobre el mecanismo de comunicación RPC:

F	A.- Al igual que ROI, siempre requiere que se utilice un servidor de nombres.
V	B.- El stub cliente sirve para poder invocar procedimientos remotos como si fuesen locales
F	C.- Sólo podemos definir como procedimientos remotos aquellos que no utilicen paso de parámetros por referencia.
F	D.- Se basa en un modelo de invocación/respuesta con asincronía y persistencia
F	E.- El programador debe escribir el código de los stubs cliente y servidor para cada procedimiento que pueda invocarse de forma remota

3. Sobre el mecanismo de comunicación ROI:

V	A.- El cliente obtiene un proxy para invocar al objeto remoto.
V	B.- El proxy incluye una referencia al objeto remoto
F	C.- El mecanismo de ROI permite el paso de objetos como argumentos en las invocaciones, utilizando "paso por valor". El paso de los objetos por referencia se simula mediante el paso por valor, de forma similar a como se emplea en el mecanismo RPC.
F	D.- En Java RMI, se requiere emplear un lenguaje especial de definición de interfaces, denominado IDL (Interface Definition Language), para describir los interfaces de los objetos remotos
F	E.- El programador debe escribir el código del proxy y esqueleto para cada objeto remoto

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



V	A.- Se cumple que f $\rightarrow$ h
V	B.- Se cumple que a $\rightarrow$ g
V	C.- Se cumple que d || c
F	D.- Se cumple que a|| e
V	E.- Utilizando solamente el valor de los relojes de Lamport, no podemos determinar si los eventos "a" y "e" son concurrentes entre sí.

5. Respecto a la figura anterior:

F	A.- En "a" el reloj de Lamport C(a) = 0
V	B.- En "e" el reloj de Lamport C(e) = 4
F	C.- En "f" el reloj vectorial V(f) = [1,1,1]
V	D.- En "c" el reloj vectorial V(c) = [3,0,1]
V	E.- En "h" el reloj vectorial V(h) = [2,2,3]

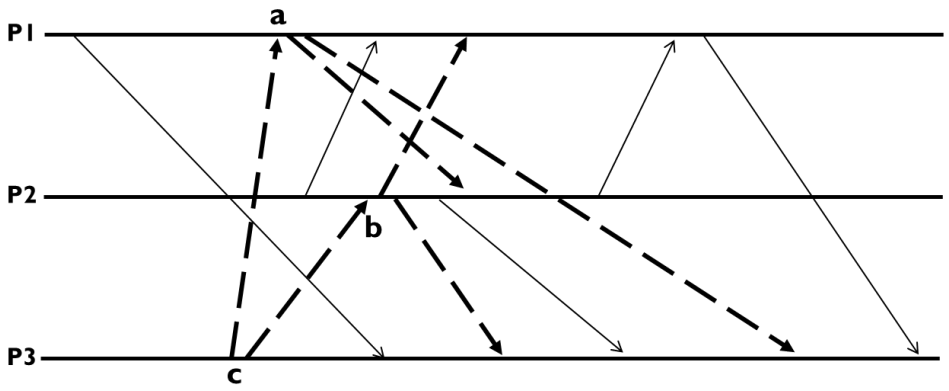
6. Respecto a los algoritmos de sincronización:

V	A.- El algoritmo de Chandy-Lamport requiere que los canales no pierdan ni desordenen mensajes
F	B.- El algoritmo de Bully falla si dos o más nodos inician el algoritmo de forma simultánea (es decir, si hay más de un iniciador)
F	C.- El algoritmo de elección de líder sobre anillos falla si dos o más nodos inician el algoritmo de forma simultánea (es decir, si hay más de un iniciador)
V	D.- El algoritmo centralizado de exclusión mutua visto en clase limita la escalabilidad y la tolerancia a fallos porque el coordinador supone un cuello de botella y un punto de fallo único
V	E.- El algoritmo distribuido de exclusión mutua visto en clase utiliza relojes lógicos para ordenar algunas solicitudes de entrada a la sección crítica.

7. Sobre el espacio de nombres y la resolución de nombres:

V	A.- Para el espacio de nombres suele utilizarse un esquema jerárquico (de tipo árbol).
V	B.- Algunos servicios de nombres permiten asociar atributos a las entidades y buscar por atributos.
V	C.- Por escalabilidad, suelen utilizarse varios servidores de nombres, que se ocupan de directorios distintos.
F	D.- La resolución de nombres iterativa supone que cada servidor debe resolver su parte y pasar el resto al siguiente servidor en la jerarquía.
F	E.- La resolución de nombres recursiva impone mayor responsabilidad al cliente, aligerando la carga sobre los servidores.

8. Respecto a la siguiente figura, suponga que P3 inicia el algoritmo de Chandy-Lamport en "c", siendo las líneas discontinuas los mensajes que se envían como consecuencia de la ejecución de este algoritmo y las líneas continuas los mensajes normales. Cuando finalice el algoritmo:



V	A.- Los estados locales de P1 en "a", P2 en "b" y P3 en "c" formarán parte del estado global
F	B.- Se habrá obtenido en este caso un corte inconsistente, pues alguno de los mensajes registrados como recibidos todavía no ha sido enviado.
V	C.- En el canal (P1,P3) se habrá registrado 1 mensaje
F	D.- En el canal (P2,P3) se habrá registrado 1 mensaje
F	E.- En el canal (P2,P1) se habrán registrado 2 mensajes

9. Sobre el servicio de localización:

V	A.- Dado un identificador, retorna una dirección.
F	B.- El modelo de difusión escala bien en distancia.
F	C.- En el modelo de punteros adelante, el servicio de recolección de residuos ("garbage-collection") permite acortar la longitud de las cadenas de punteros, y por lo tanto agilizar accesos posteriores
F	D.- En el modelo de difusión, asumiendo que el nodo A contiene la entidad a buscar, el fallo de un nodo distinto de A impide obtener la dirección de dicha entidad.
V	E.- En el modelo de punteros adelante, asumiendo que el nodo A contiene la entidad a buscar, entonces el fallo de un nodo distinto de A que forma parte de la cadena de punteros impide obtener la dirección de dicha entidad.

10. Sobre las arquitecturas de un sistema distribuido:

V	A.- Las arquitecturas de capas y las arquitecturas basadas en objetos son ejemplos de estilos arquitectónicos software.
F	B.- La arquitectura de sistema define cuál es la organización lógica de los componentes, cómo están organizados y cómo deberían interactuar entre sí.
V	C.- Las arquitecturas de sistema se dividen en arquitecturas centralizadas (que emplean distribución vertical), arquitecturas descentralizadas (que emplean distribución horizontal) y arquitecturas híbridas (que combinan distribución vertical y horizontal).
V	D.- Los sistemas peer-to-peer son un ejemplo de arquitectura descentralizada con distribución horizontal.
V	E.- La distribución horizontal se da cuando un cliente o servicio puede estar físicamente dividido en partes equivalentes a nivel lógico, pero cada parte funcionando en un nodo distinto.

EXAMEN 2do PARCIAL – Prácticas 3 y 4 **9 de Junio de 2014**

Concurrencia y Sistemas Distribuidos **Modelo A**

1. Sobre la práctica de Servicios de Dominio de Active Directory AD DS:

V	A.- Se recomienda, pero no se requiere, que exista más de un controlador de dominio en cada dominio.
V	B.- La estructura de dominios de una organización puede estar compuesta por varios árboles.
F	C.- El usuario *eovic\Administrador* no puede eliminar un fichero creado por el usuario *amsterdam\Administrador* con los permisos asignados por defecto.
F	D.- Un usuario del subdominio *amsterdam* no puede a acceder a un recurso compartido de una máquina que no pertenece a dicho subdominio.
V	E.- Un usuario sólo puede pertenecer a una unidad organizativa.

PARTE TEORIA

Sobre los sistemas distribuidos:

1.	La transparencia de acceso odemos las diferencias en la representación de los datos y en cómo se accede a los recursos.	V
2.	Cuando un odemos distribuido es abierto facilita que uno de sus módulos o odemosan pueda utilizarse en otro odemos distribuido.	V
3.	Para conseguir escalabilidad de distancia debe asumirse que se está utilizando una red de área local. JUSTIFICACIÓN: *La escalabilidad de distancia permite extender el odemos por redes de área amplia (WAN), por lo que si se usan algoritmos basados en redes de área local, se debe odemosan que no se está usando una red local, sino una red WAN, por lo que hay que tener en cuenta los efectos de los retardos en la odemosant de los datos y la menor fiabilidad de las odemosantve.*	F
4.	Los algoritmos descentralizados facilitan distribuir la carga computacional entre diferentes ordenadores.	V
5.	Se odemos odemos algoritmos descentralizados para conseguir escalabilidad odemosantve, de modo que los cómputos se distribuyen entre diferentes áreas administrativas del odemos. JUSTIFICACIÓN: *Para conseguir escalabilidad odemosantve se deben odemos protocolos y mecanismos estándar de autenticación y autorización; así como implementar mecanismos para proteger a cada organización del resto y del propio odemos.*	F
6.	Para mejorar la escalabilidad de tamaño, los odemos deben delegar en el servidor tantas responsabilidades como sea odemos. JUSTIFICACIÓN: *Al contrario, contra menos se centralicen las tareas en los servidores, mayor escalabilidad de tamaño se podrá conseguir.*	F
7.	La *capa de middleware*, que se ubica bajo el nivel de aplicación, puede integrar algunos mecanismos de comunicación que faciliten la programación de aplicaciones distribuidas; por ejemplo: JMS.	V

Sobre el mecanismo de comunicación ROI:

8.	El componente denominado ORB se encarga, entre otras cosas, de identificar y odemosa a los objetos remotos.	V
9.	El middleware de un odemos con comunicación basada en ROI emplea generalmente comunicación sincrónica no persistente y direccionamiento directo.	V
10.	Cuando se pasa un objeto por valor, se copia una referencia al mismo. JUSTIFICACIÓN: *Cuando se pasa un objeto por valor, el estado del objeto se empaqueta, mediante un proceso denominado serialización.*	F
11.	El proxy empaqueta los argumentos una vez recibe la contestación del esqueleto. JUSTIFICACIÓN: *El proxy desempaqueta los argumentos al recibir la contestación, para así devolverlos al proceso cliente.*	F

Sobre el mecanismo de comunicación Java RMI:

12.	El proxy correspondiente a un objeto odemo se crea en tiempo de ejecución cuando seaccede por primera vez al objeto odemo.	V
13.	Se considera objeto local todo objeto que ode puede invocarse desde la computadora enque se define (aunque sea desde otras máquinas virtuales Java), y objeto odemo a todo objeto que puede invocarse desde otras computadoras. *JUSTIFICACIÓN: Si un objeto se invoca desde otras máquinas virtuales Java de la misma computadora, también se considera como objeto odemo.*	F
14.	No proporciona transparencia de ubicación, porque la sintaxis de invocación del método es distinta dependiendo de si el objeto es local o odemo. *JUSTIFICACIÓN: La sintaxis de invocación es la misma. En la interfaz del objeto definimos los métodos del objeto (con independencia de que vaya a ser odemo o local). Si el objeto es odemo, la interfaz debe extender java.rmi.Remote.*	F
15.	La clase de un objeto odemo debe implementar un interfaz que extienda java.rmi.Remote	V

Respecto a los servicios Web RESTful:

16.	En un servicio Web RESTFul, los mensajes constan de una cabecera formada por varios campos fijos definidos por el estándar RESTful, un conjunto de propiedades que pueden ser definidas por la aplicación y de un contenido (normalmente, en XML o JSON). *JUSTIFICACIÓN: El estilo arquitectónico REST no especifica ninguna cabecera, ni campos fijos definidos. Por tanto, los mensajes tienen estructura libre.*	F
17.	GET https:/weatherapp.com/zipcodes es un ejemplo de llamada a un servicio Web RESTFul.	V
18.	Un odemos distribuido con comunicación basada en servicios web REST emplea generalmente comunicación sincrónica no persistente y direccionamiento directo.	V
19.	Son atendidos por servidores que mantienen el estado de las peticiones de los odemos. *JUSTIFICACIÓN: Los servicios son "sin estado", de modo que los servicios no mantienen ninguna odemos odemos a las peticiones de los odemos.*	F

Respecto al mecanismo de comunicación Java Message Service:

20.	JMS utiliza direccionamiento directo, pues en la cabecera del mensaje se especifica claramente el destino del mensaje. *JUSTIFICACIÓN: JMS emplea direccionamiento indirecto, ya que el cliente envía el mensaje a un destino (Destination), que puede ser una cola (Queue) o un tema (Topic). Y no tiene por qué conocer quién será el consumidor del mensaje.*	F
21.	Java Message Service ofrece una comunicación fuertemente acoplada, pues tanto el productor como el consumidor de un mensaje se crean utilizando la misma factoría de conexión. *JUSTIFICACIÓN: JMS ofrece una comunicación débilmente acoplada, ya que al emplear destinos, el productor y el consumidor del mensaje no necesitan conocerse. Solamente requieren conocer el destino (y estar de acuerdo en el formato del mensaje).*	F
22.	Existen dos tipos de destinos: Colas (Queues), utilizadas para enviar mensajes a un único cliente; y Temas (Topics), que permiten la publicitación/suscripción.	V
23.	Las factorías de conexiones y los destinos se crean utilizando una herramienta odemosantve ofrecida por el Proveedor de JMS.	V

Respecto a la sincronización en sistemas distribuidos:

24.	En un odemos distribuido, si varios nodos desean hacer uso de un recurso compartido, pero dicho recurso no puede ser utilizado a la vez por más de un nodo, debemos odemos un algoritmo de selección de líder para escoger al nodo que puede hacer uso del recurso. *JUSTIFICACIÓN: Para el acceso en odemosa mutua a un recurso compartido debemos odemos un algoritmo de odemosa mutua de sistemas distribuidos (por ejemplo, cualquiera de los tres algoritmos vistos en la asignatura, i.e. algoritmo centralizado, algoritmo distribuido o algoritmo para anillos).*	F
25.	El algoritmo de Chandy–Lamport permite establecer un orden total entre los eventos de un odemos distribuido. *JUSTIFICACIÓN: El algoritmo Chandy–Lamport permite obtener el estado global del odemos distribuido. Para establecer un orden total, utilizaremos los relojes lógicos de Lamport y los identificadores de los nodos.*	F
26.	Si los nodos de un odemos distribuido están dispuestos en un anillo lógico, resultaría sencillo obtener el estado global del odemos, pues pueden registrar su estado cuando reciben el token que circula por el anillo. *JUSTIFICACIÓN: El estado global incluye no ode el estado de cada nodo, sino también los mensajes en tránsito (que han sido enviados y que todavía no han llegado a su destino). Por tanto, haría falta registrar también (de algún modo) cuáles son dichos mensajes, de modo que obtengamos una instantánea consistente.*	F

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

27.	El algoritmo distribuido de odemosa mutua odemos que se utilicen relojes lógicos de Lamport e identificadores para odemos la precedencia de las solicitudes. *JUSTIFICACIÓN: Se odemos establecer un orden total de los eventos para así resolver los empates que se puedan odemos cuando dos o más nodos desean acceder "a la vez" a la sección crítica.*	V
28.	En el algoritmo Bully, un nodo sabe que será el nuevo coordinador cuando, tras enviar los mensajes de tipo ELECCIÓN, no recibe ninguna respuesta. *JUSTIFICACIÓN: Como envía ELECCIÓN a los nodos con mayor identificador, si no recibe respuesta, entonces él es el nodo odemo con mayor identificador. Por tanto, él es el nuevo líder.*	V
29.	En el algoritmo de elección de líder para anillos, el mensaje ELECCIÓN incluye dos campos: uno para registrar el iniciador y otro para informar de cúal era el líder antes de iniciar el algoritmo. *JUSTIFICACIÓN: El mensaje ELECCIÓN incluye dos campos. Uno para registrar el iniciador y el otro para registrar la lista de nodos activos.*	F
30.	En el algoritmo centralizado de odemosa mutua, cuando un nodo solicita entrar en la sección crítica y ésta está ocupada, el coordinador envía un mensaje al nodo que está en la sección crítica para que este odemo, cuando termine, se lo notifique al nodo que ha hecho la solicitud. *JUSTIFICACIÓN: El coordinador no envía ningún mensaje al nodo en la sección crítica. Cuando éste termina, notifica directamente al coordinador.*	F
31.	En el algoritmo distribuido de odemosa mutua, un nodo entra en la sección crítica cuando recibe un mensaje OK de cada uno de los demás nodos.	V

Respecto a los algoritmos de sincronización de relojes físicos.

32.	En el algoritmo de Cristian, si un cliente C pregunta al servidor su hora en el instante 20.000 (según el reloj de C), recibe la respuesta del servidor en el instante 20.010 (según el reloj de C) y calcula que el nuevo valor para su reloj debe ser 20.024, entonces se puede deducir que la respuesta del servidor habrá sido 14. *JUSTIFICACIÓN: En el algoritmo de Cristian, el servidor envía "su valor de reloj", por ejemplo, 20.014. Pero no envía "su diferencia" odemos al cliente.*	F
33.	En el algoritmo de Berkeley, el nodo que actúa como coordinador puede que tenga que ajustar también su reloj, al igual que los otros nodos que odemosant en el algoritmo.	V
34.	El algoritmo de Berkeley odemo que el odem de un mensaje desde el nodo A al nodo B consume el mismo tiempo que la respuesta desde B hasta A.	V

Respecto a los relojes lógicos de Lamport y vectoriales:

35.	Si el valor en un evento **x** del reloj lógico de Lamport es igual a C(**x**)=1 y el valor en **y** del reloj lógico de Lamport es igual a C(**y**)=6, entonces odemos afirmar que **x**→ **y**. *JUSTIFICACIÓN: Dados los valores de los relojes lógicos de Lamport, no se puede afirmar si dos eventos son concurrentes o si uno ocurre antes que el otro.*	F
36.	Si el valor en **x** del reloj vectorial es igual a V(x)=[9,0,1] y el valor en **y** del reloj vectorial es igual a [6,2,2], entonces odemos afirmar que **x** || **y**	V

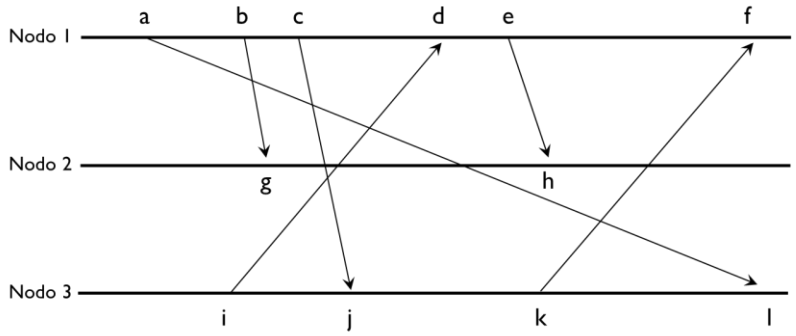


Figura 1: muestra todos los envíos de mensajes entre tres nodos

Respecto a la figura 1:

37.	Se cumple que $\mathbf{c} \rightarrow \mathbf{l}$ y que $\mathbf{e} \| \| \mathbf{l}$	V
38.	El valor en **l** del reloj lógico de Lamport es igual a 6	V
39.	El valor en **d** del reloj vectorial es igual a V(d)=[4,0,1] y el valor en **j** del reloj vectorial es igual a V(j)= [3,1,2] *JUSTIFICACIÓN: V(d)=[4,0,1] , V(j)=[3,0,2]*	F

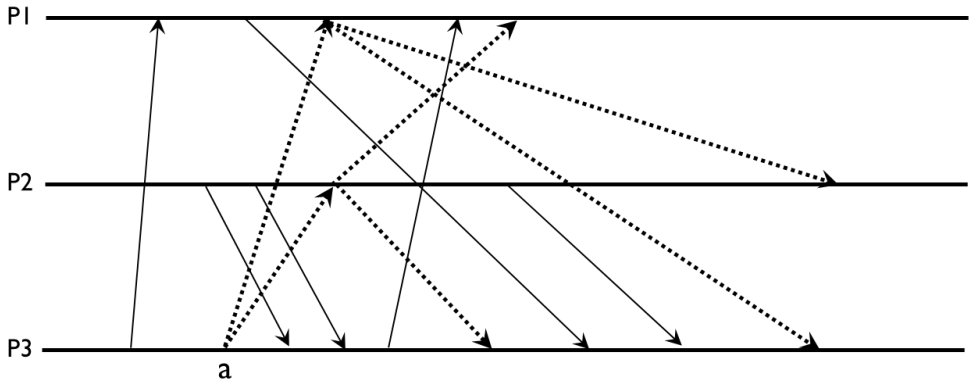


Figura 2

Respecto a la arquitectura de los sistemas distribuidos:

40.	Las arquitecturas de capas, las arquitecturas basadas en objetos y las arquitecturas basadas en eventos son ejemplos de estilos arquitectónicos de arquitecturas software.	V
41.	En una arquitectura centralizada se distinguen dos tipos de roles: servidor, responsable de la gestión de un recurso determinado y que define una serie de servicios (sobre dicho recurso); y cliente, que es un componente que utiliza dichos servicios.	V
42.	Los sistemas replicados y los sistemas peer-to-peer presentan distribución horizontal.	V

Respecto a la figura 2, suponga que P3 inicia el algoritmo de Chandy-Lamport en **a**, siendo las líneas discontinuas los mensajes que se envían como consecuencia de la ejecución de este algoritmo y las líneas continuas los mensajes normales. Cuando finaliza el algoritmo,

43.	en el canal (P3,P1) se habrán registrado 0 mensajes *JUSTIFICACIÓN: Al acabar el algoritmo, la instantánea consistente comprende los puntos donde P1 y P2 recibieron el primer MARCA (pues ahí guardaron su estado), así como el punto "a" (donde P3 guardó su estado). Si unimos esos puntos, podemos ver claramente que el algoritmo habrá registrado 1 mensaje en canal (P1,P3), 2 mensajes en canal (P2, P3), 0 mensajes en canal (P3,P1) y 0 mensajes en canal (P3,P2). Los otros mensajes, o bien pasaron antes de la instantánea (como el primer mensaje del dibujo), o bien pasarán después, en el futuro, como ocurre con el mensaje en canal (P3,P1) y el último mensaje del canal (P2, P3).*	V
44.	en el canal (P2,P3) se habrán registrado 3 mensajes	F
45.	en el canal (P1,P3) se habrá registrado 1 mensaje.	V

PARTE PRACTICAS

Respecto a la práctica del Chat distribuido en Java RMI:

9.	Todos los objetos invocables de forma remota deben estar registrados en el servidor de nombres (rmiregistry).	F
10.	Cada uno de los procesos ChatClient y ChatServer abre un puerto, porque es necesario para utilizar el servidor de nombres (rmiregistry).	F
11.	Los objetos de la clase ChatMessage son creados siempre por el ChatServer.	F
12.	ChatRobot debe utilizar la clase que proporciona la interfaz gráfica (ChatUI) para obtener las indicaciones sobre a qué servidor y canal debe conectarse.	F

Dado el siguiente fragmento de código, que corresponde al código original de la práctica deChat, donde los puntos suspensivos indican código omitido:

..... *(Omitido. Código en el enunciado del examen)*

13.	Los objetos de la clase ChatUser pueden ser accedidos remotamente.	V
14.	Los objetos de la clase ChatClient pueden ser accedidos remotamente.	F
15.	El método sendMessage del objeto ChatUser se invoca para que el usuario que representa reciba un mensaje de un canal.	V
16.	En la clase ChatClient, la variable srv es un proxy del servidor de chat (objeto remoto ChatServer).	V

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
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
Para lograr transparencia de ubicación, los recursos requieren estar identificadoscon 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 proxyy 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 fallosen 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 defallos.	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
Con relojes lógicos de Lamport se garantiza que dos eventos correspondientes anodos 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
En el algoritmo de Bully, todo nodo que recibe OK sabe que no será elegido comolíder.	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
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:

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
En el algoritmo de exclusión mutua distribuido, ningún nodo necesita mantener unalista de respuestas pendientes (nodos que han solicitado acceso al recurso, pero a los que todavía no se les ha contestado).	F
En el algoritmo de exclusión mutua distribuido, el protocolo de salida consiste endifundir 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
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 realizaen 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

APELLIDOS:		**NOMBRE:**	
DNI:		**FIRMA:**	

EXAMEN FINAL (18 junio 2013) – Bloque segundo parcial teoría

Este bloque tiene una puntuación máxima de **10 puntos.**
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.**

1. Un sistema distribuido:

F	Requiere que los relojes de todos sus ordenadores estén sincronizados.
F	Deberá tener siempre una actividad líder, que podrá ser seleccionada empleando algún algoritmo de elección de líder.
F	Requiere que todos sus eventos estén ordenados, empleándose generalmente los relojes lógicos de Lamport para este fin.
V	Será escalable si es capaz de mantener su capacidad de servicio cuando crezca su número de componentes (nodos, procesos, clientes, etc.).

2. Las siguientes son técnicas para aumentar la escalabilidad en sistemas distribuidos:

V	Utilizar replicación.
V	Utilizar algoritmos descentralizados.
V	Delegar parte del procesamiento a los clientes.
V	Repartir tanto las tareas como los datos entre múltiples nodos servidores.

3. El algoritmo de Berkeley:

F	Es un ejemplo de algoritmo descentralizado.
V	Sincroniza los relojes de los nodos de un determinado sistema distribuido, sin importarle la divergencia que pueda haber entre estos relojes y la hora "oficial".
F	Es uno de los algoritmos de exclusión mutua más eficientes.
F	Es el utilizado para actualizar los relojes vectoriales.

4. Los relojes vectoriales:

F	Son necesarios para implantar la transparencia de concurrencia.
F	Son los resultantes del algoritmo de Cristian.
V	Permiten determinar en todos los casos si dos eventos de una ejecución son concurrentes o no.
V	No siempre pueden ordenarse entre sí.

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

V	Se necesitan para obtener las direcciones o identificadores de ciertas entidades, cuando se conozca su nombre.
V	Pueden implantar la resolución de nombres de forma recursiva o de forma iterativa.
V	Suelen proporcionar tres operaciones: inserción (o registro), resolución y borrado.
V	LDAP es un ejemplo de servicio de directorio basado en atributos.

6. Sobre el mecanismo de llamada a procedimiento remoto (RPC):

F	Proporciona transparencia de persistencia.
F	Utiliza stubs clientes para gestionar la recepción de mensajes de petición y el envío de los mensajes de respuesta.
V	Proporciona transparencia de ubicación.
V	En su variante asincrónica no puede retornar resultados ni argumentos de salida.

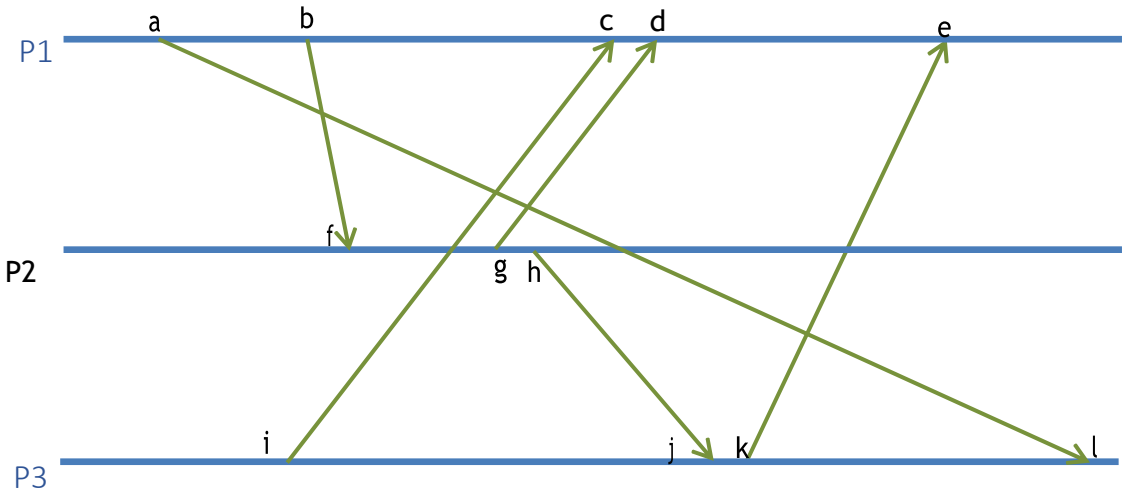
7. Sobre el mecanismo de invocación a objeto remoto (ROI):

V	Es el utilizado en Java RMI.
V	Oculta el envío y recepción de mensajes entre el nodo cliente y el nodo servidor.
F	Siempre utiliza paso de parámetros por valor.
V	Utiliza proxies y esqueletos.

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

F	El algoritmo de Cristian se utiliza para gestionar relojes lógicos.
F	El algoritmo distribuido de exclusión mutua no necesita realizar ninguna acción en su protocolo de salida.
V	El algoritmo Bully requiere comunicación fiable.
V	El algoritmo de Chandy y Lamport requiere que los canales de comunicación respeten un orden FIFO.

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



F	El reloj vectorial de “e” es VT(e)=[5,3,3] y el de “h” es VT(h)=[2,3,1].
F	Los eventos “e” y “f” son concurrentes.
F	El reloj de Lamport de “d” es C(d)=4 y el de “k” es C(k)=7.
F	El reloj de Lamport de “g” es C(g)=4 y el reloj vectorial de ese mismo evento es VT(g)=[2,2,1].
F	Los eventos “b” y “l” son concurrentes.

EXAMEN FINAL DE RECUPERACIÓN – Concurrencia y Sistemas Distribuidos
Unidades Didácticas 1 a 11 **23 de Junio de 2014**

NOMBRE:		DNI:	

Este bloque tiene una puntuación máxima de **10 puntos**, que equivalen a 5 puntos de la nota final de la asignatura. 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.

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

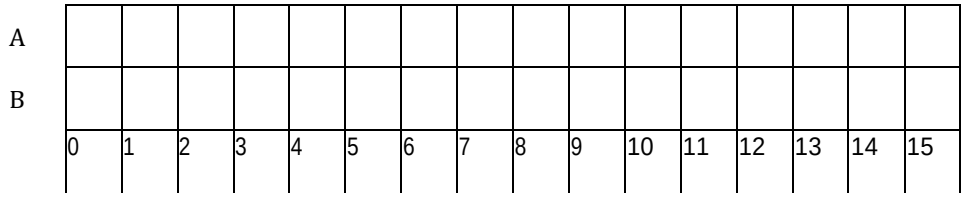
Atención: La columna V/F que aparece en este documento sólo sirve a efectos de facilitar el proceso de realización del examen, de modo que no se tendrá en cuenta lo ahí puesto en la corrección del examen. **Solamente se corregirá la hoja de respuestas, que se proporciona aparte.**

Sean dos tareas A, y B, con prioridad(A)>prioridad(B); y dos semáforos Q y R que utilizan el protocolo del techo de prioridad inmediato, tales que:

Tarea	Prioridad	Instante Activación	Patrón Ejecución
A	1	2	ERRRQQE
B	2	0	EQQRRRE

E representa una unidad de ejecución sin acceso a sección crítica alguna, Q una unidad de ejecución con bloqueo del semáforo Q, R una unidad de ejecución con bloqueo del semáforo R; cada tarea bloquea el semáforo una única vez.

Para contestar a las cuestiones puede ayudarse utilizando la cuadrícula adjunta, donde se ilustre gráficamente la ejecución de las tareas.

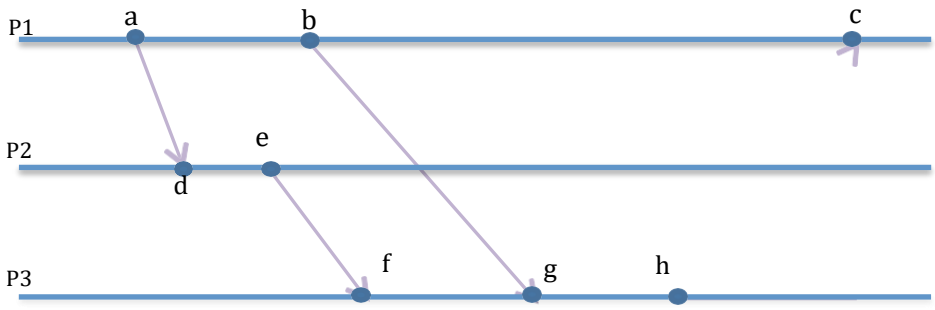


	V/F
1. La tarea B está en ejecución en el intervalo [0,2).	V
2. La tarea A está en ejecución en el intervalo [2,6).	F
3. En el instante t=6 se produce un interbloqueo entre A y B.	F
4. En el intervalo [1,6) la prioridad de la tarea B es 1.	V
5. La tarea B finaliza su ejecución en el instante 14.	V

Sobre los relojes lógicos de Lamport:

26. Si dos eventos tienen el mismo valor de reloj, entonces son concurrentes.	V
27. Si dos eventos son concurrentes, entonces tienen el mismo valor de reloj.	F
28. Si un evento ocurre antes que otro, el primero tendrá un valor de reloj menor.	V
29. Requieren que se envíen mensajes entre los nodos de forma periódica para mantener dichos relojes sincronizados.	F

Dado el siguiente diagrama temporal de eventos… (se asume que no hay eventos previos):



30. El valor del reloj lógico de Lamport del evento “g” es 5.	V
31. El valor del reloj vectorial del evento “h” es [2,2,3]	V
32. Podemos observar que los eventos “b” y “d” son concurrentes.	V
33. Podemos deducir que el valor del reloj lógico de “c” en ocasiones será 7 y en ocasiones tendrá otro valor.	F
34. Se cumple que f --> c y que e||b	V

Sobre la localización de entidades en sistemas distribuidos:

35. El mecanismo de punteros hacia delante escala peor que el mecanismo de difusiones.	F
36. ARP es un ejemplo de localización por punteros hacia delante.	F
37. Un problema del mecanismo de localización por punteros hacia delante radica en que se deben hacer difusiones para encontrar partes de la cadena.	F
38. Nunca es necesaria cuando disponemos de un servicio de nombres.	F

Sobre la invocación a objetos remotos (ROI):

39. El proxy ofrece la misma interfaz que el esqueleto.	F
40. Existe un esqueleto por cada método del objeto remoto.	F
41. El proxy empaqueta los argumentos del método llamado antes de que el esqueleto invoque al objeto remoto.	V
42. El esqueleto desempaqueta los argumentos enviados por el proxy antes de invocar al objeto remoto.	V

Respecto a los algoritmos de sincronización de relojes:

43. En el algoritmo de Berkeley no se asume que exista un ordenador con un reloj preciso.	V
44. Cuando se utiliza el algoritmo de Cristian, en la expresión Cs + (T1 ‐T0)/2 , T1 es el instante en el que el cliente pide el valor del reloj del servidor.	F
45. Permiten asignar valores numéricos a los eventos de envío y recepción de mensajes a partir de la relación “ocurre‐antes”.	F
46. En general es problemático hacer que el reloj de un ordenador retroceda.	V

Sobre los sistemas distribuidos:

47. La transparencia de ubicación oculta el hecho de que un recurso esté ubicado en memoria volátil o en memoria persistente.	F
48. Para mejorar la escalabilidad administrativa hay que conseguir que todos los ordenadores del sistema pertenezcan a una única organización.	F
49. La capa de middleware puede integrar algunos mecanismos de comunicación de alto nivel que faciliten la programación de aplicaciones distribuidas; por ejemplo: RPC.	V
50. Las arquitecturas de sistema para sistemas distribuidos representan la implementación física de los componentes software del sistema en máquinas reales.	V

EXAMEN FINAL DE RECUPERACIÓN – Concurrencia y Sistemas Distribuidos
Unidades Didácticas 1 a 11 23 de Junio de 2014

EXAMEN RECUPERACIÓN SEGUNDO BLOQUE Concurrencia y Sistemas DistribuidosUnidades 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:

Los detectores de fallos y la replicación son mecanismos para logrartransparencia de fallos.	V
2. La transparencia de concurrencia persigue ocultar la ubicación de los recursos.	F
3. La transparencia de persistencia trata de ocultar el hecho de que los recursos estén almacenados de forma no volátil.	V
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.	V
5. Los principales ejes de la transparencia de distribución son tres: transparencia de ubicación, transparencia de fallos y transparencia de replicación.	V

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.	F
7. El principal mecanismo para lograr tolerancia a fallos es la replicación.	V
8. Los fallos bizantinos son fallos compuestos.	F
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.	F

Sobre los modelos de replicación:

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

Sobre la escalabilidad:

15. Entre las técnicas más importantes para lograr escalabilidad, podemos mencionar replicación y distribución de la carga entre diferentes nodos.	V
16. Un sistema que emplee caching generalmente será menos escalable que otro sistema que no emplee dicha técnica.	F
17. Los sistemas altamente escalables suelen garantizar consistencia fuerte.	F
18. La técnica de distribuir datos mejora la escalabilidad.	V
19. En la comunicación asíncrona, el middleware responde al emisor tras recibiraviso del receptor de haber procesado el mensaje.	F
20. JMS ofrece una comunicación fuertemente acoplada, ya que los clientes deben establecer conexión entre sí a través de las factorías de conexión.	F
21. En JMS se emplea sincronización síncrona, de modo que el emisor debe esperar a que el receptor reciba el mensaje.	F
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.	V
23. El mecanismo de comunicación ROI proporciona transparencia de ubicación.	V
24. En ROI el emisor envía mensajes a una cola y el receptor recibe mensajes de dicha cola, a la que se ha suscrito.	F

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.	F
26. Java RMI es una API de comunicación especificada en múltiples lenguajes de programación, tales como Java, Javascript, C#, Python, etc.	F
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.	V

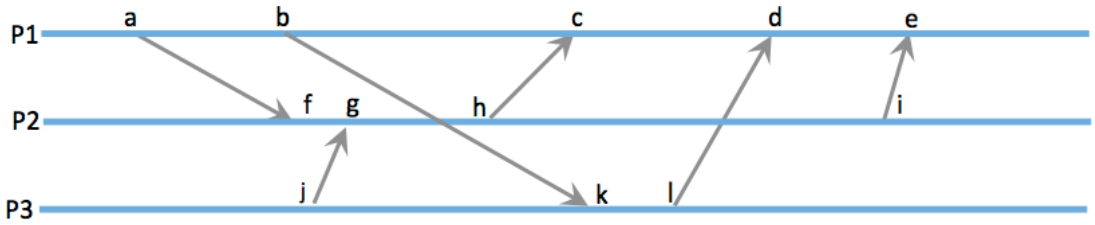
Sobre los algoritmos de sincronización de relojes físicos:

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.	V
29. El algoritmo de Cristian sólo puede aplicarse si el tiempo que necesita el servidor para procesar el mensaje es 10 milisegundos.	F
30. En el algoritmo de Berkeley, no es necesario conocer a priori el retardo medio de los mensajes que intercambian los distintos nodos.	V
31. El algoritmo de Berkeley permite ajustar todos los relojes a una misma hora, perono garantiza que se trate de la hora exacta (real).	V

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.	F
33. Haciendo uso de los relojes lógicos de Lamport y de los identificadores de losnodos se puede establecer un orden total entre los distintos eventos.	V
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.	V
35. Por sus características, el algoritmo de Chandy-Lamport puede considerarse descentralizado.	V
36. En el algoritmo de Chandy-Lamport se pueden recibir varios mensajes MARCA por un mismo canal.	F

Dado el siguiente cronograma que muestra la ejecución de tres procesos en un sistemadistribuido, 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.	V
38. El reloj de Lamport del evento "i" es 5.	V
39. El reloj vectorial en "h" es V(h)=[1,3,1]	V
40. El reloj vectorial de "k" es V(k)=[2,0,2].	V

Sobre los algoritmos de exclusión mutua y consenso:

41. En el algoritmo distribuido de exclusión mutua, todo nodo necesita mantener unalista de respuestas OK pendientes.	V
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 losrestantes nodos.	F
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 sitiene prioridad sobre otros que también desean entrar.	F
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.	V

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).	F
El algoritmo Bully requiere que cada nodo disponga de un identificador único,que además debe ser conocido por los otros nodos.	V
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).	F
48. El proceso ChatServer implementa la clase MessageListener para poder recibir las peticiones de los ChatClient para conectar a los ChatUser.	F
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.	F
50. Los objetos que representan a los canales se registran en el servidor de nombres.	F
Para enviar un mensaje privado entre dos clientes, es necesario que intervengaChatServer para la comunicación entre estos dos clientes.	F

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.	F
53. Ante la conexión de un nuevo usuario, CsdMessengerClient recibe un mensaje con la lista de usuarios.	V
Las colas JMS asociadas a cada Cliente (CsdMessengerClient) las crea cada clienteuna vez se ha identificado ante CsdMessengerServer.	F
En la aplicación de chat se transmiten distintos objetos por medio de mensajes detipo *ObjectMessage*.	V