EXAMEN FINAL – 2º BLOQUE **Concurrencia y Sistemas Distribuidos**
Unidades Didácticas 7 a 10 - Prácticas 4, 5 y 6 **Fecha: 16 de Junio de 2017**

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 **58 afirmaciones**, si éstas son verdaderas (**V**) o falsas (**F**).

Cada respuesta vale: correcta= 10/58, errónea= -10/58, 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.

TEORÍA (49 PREGUNTAS):

Sobre los sistemas distribuidos:

1. Todos los sistemas distribuidos de tiempo real son concurrentes.
2. Todos los sistemas concurrentes son distribuidos.
3. Whatsapp es un ejemplo de sistema distribuido.
4. Los sistemas distribuidos tratan de proporcionar la imagen de sistema único, es decir que el usuario o cliente del sistema lo observe como si no estuviera distribuido.

Sobre la transparencia en sistemas distribuidos:

5. Existen múltiples clases de transparencia. Algunas de ellas son: ubicación, migración, replicación, fallos.
6. Algunas veces es imposible lograr la transparencia al 100%.
7. Los servicios de nombres ayudan a proporcionar transparencia de ubicación.

Sobre Java RMI:

8. Los objetos remotos deben implementar una interfaz que haya extendido la interfaz java.rmi.Remote.
9. Proporciona comunicación sincrónica entre clientes y servidores.
10. Permite el paso de argumentos a los métodos remotos únicamente por valor, mediante el procedimiento conocido como serialización.
11. Proporciona un servicio de nombres, llamado "rmiregistry".

Un esqueleto (“stub” servidor):

12. Permite a un único proxy conectarse al objeto remoto. De forma que necesitamos tantos esqueletos como proxies tenga el objeto.
13. Realiza la llamada al objeto destino de la invocación, de forma local.
14. Se encarga de empaquetar los argumentos de salida a la hora de construir el mensaje de respuesta hacia el cliente.

Respecto a los servicios web RESTful:

15. Los servicios Web REST son un ejemplo de comunicación con direccionamiento indirecto.
16. Para realizar cualquier llamada a un servicio web RESTful se requiere emplear el método HTTP GET, seguido de un objeto codificado en XML o JSON.
17. En la respuesta a cualquier llamada a un servicio RESTful se requiere devolver un código de terminación basado en códigos de estado del protocolo HTTP.

Sobre la comunicación basada en mensajes mediante JMS:

18. En JMS emisor y receptor no necesitan conocerse entre sí.
19. En JMS las factorías de conexiones se emplean para crear directamente los mensajes que intercambiarán los clientes JMS.
20. JMS proporciona un modelo de comunicación fundamentalmente sincrónico.
21. Los objetos administrados son exclusivamente los temas y los mensajes.
22. A diferencia de otros mecanismos, se requiere de un proceso adicional (bróker) que proporcione la comunicación.
23. Si un proceso recibe con éxito un mensaje de una cola de mensajes, esto implica necesariamente que el proceso que envía los mensajes a la cola está funcionando en ese preciso instante.

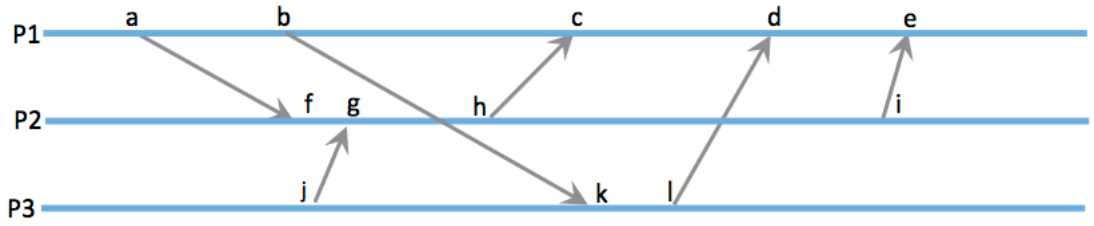
En un sistema distribuido con 4 nodos se emplea el algoritmo de Berkeley para sincronizar sus relojes. Uno de los nodos (nodo A) actúa como servidor del algoritmo, mientras que los otros nodos (nodos B, C y D) actúan como clientes. Supongamos que cada nodo tiene un reloj que indica el número de ticks transcurridos desde la misma base temporal; y que en un momento determinado tienen el siguiente valor para sus relojes: C_A=2000, C_B=2003, C_C=2005 y C_D=2004. En dicho momento el servidor (nodo A) inicia el algoritmo de Berkeley.

24. En el primer paso del algoritmo, el nodo A enviará un mensaje al resto de nodos, cuyo contenido será 2000.
25. Si la transmisión del primer mensaje desde el nodo A al B tarda *b* ticks de reloj contados desde el nodo B, cuando el nodo B reciba ese primer mensaje de A, le contestará con un mensaje cuyo contenido será *b+3*.
26. La diferencia promedio calculada por A será de 4 unidades de tiempo, por lo que el servidor ajustará su reloj incrementándolo en 4.
27. El nodo D recibe como valor de ajuste -1.

Sobre los algoritmos de exclusión mutua en sistemas distribuidos, vistos en clase:

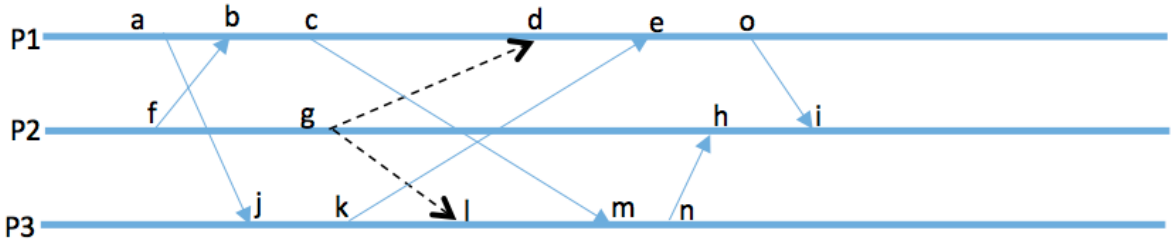
28. En el algoritmo de exclusión mutua centralizado, para que un nodo entre en la sección crítica, primero el resto de nodos deben notificar al coordinador que están de acuerdo.
29. En el algoritmo de exclusión mutua para anillos, cuando un nodo sale de la sección crítica devuelve el token al nodo coordinador, para que éste se lo dé eventualmente al siguiente nodo interesado en entrar en la sección crítica.
30. En el algoritmo distribuido, todo nodo necesita mantener una lista de respuestas pendientes (nodos que le han solicitado acceso al recurso, pero a los que todavía no se les ha contestado).
31. En el algoritmo distribuido, en caso de empate (varios nodos solicitan acceso al mismo tiempo), se desempata utilizando un algoritmo de elección de líder.

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



32. Los relojes de Lamport de los eventos "c ", "i" son iguales.
33. Teniendo en cuenta solamente los valores de los relojes de Lamport podemos afirmar que el evento “g” ocurre antes que el evento “l”
34. El reloj vectorial en "h" es V(h)=[1, 2, 1]
35. En esta figura todos los eventos que tienen el mismo valor de reloj lógico de Lamport son concurrentes.
36. El reloj vectorial de "e" es V(e)=[5, 4, 3].
37. Con relojes lógicos de Lamport se garantiza que dos eventos correspondientes a nodos distintos no pueden tener asociado el mismo valor lógico.

Dada la siguiente figura, donde se tienen 3 nodos en un sistema que cumple con las condicione impuestas por el algoritmo de Chandy-Lamport. Suponga que P2 inicia dicho algoritmo en el evento "g", siendo las líneas discontinuas los mensajes que envía P2 como consecuencia de la ejecución de este algoritmo.



38. Cuando el nodo P1 envíe el mensaje de MARCA al nodo P2, éste deberá llegar antes del evento i.
39. Cuando el nodo P3 envíe el mensaje de MARCA al nodo P2, éste deberá llegar después del evento h y antes del evento i.
40. Cuando finalice el algoritmo, en total se habrán registrado 2 mensajes en tránsito.
41. El nodo P3 registrará su estado local en el evento l.

Respecto a las arquitecturas de los sistemas distribuidos:

42. La arquitectura de sistema para sistemas distribuidos representa la conexión lógica de los componentes software del sistema.
43. En las arquitecturas de sistema descentralizadas se emplea la distribución vertical, emplazando los diferentes componentes lógicos en máquinas distintas.
44. La organización en capas plantea una distribución horizontal de los servicios, de tal forma que todos los nodos ejecutan las mismas capas.
45. En el modelo cliente/servidor, los clientes y servidores pueden estar distribuidos en diferentes máquinas.

Respecto a los sistemas peer-to-peer, los sistemas Grid y los sistemas Cloud:

46. En los sistemas peer-to-peer, cuando se utiliza una arquitectura puramente descentralizada, todos los nodos realizan las mismas tareas y no hay ningún nodo que actúe como servidor de localización.
47. Los sistemas peer-to-peer pueden clasificarse según la estructura de la red lógica, atendiendo a cómo está el contenido relacionado con la topología de la red.
48. Los sistemas Grid hacen uso de recursos computacionales ofrecidos como un servicio a través de plataformas PaaS.
49. Globus Toolkit, que es un middleware que proporciona la imagen de un sistema único, es un ejemplo típico de Infraestructura como Servicio (IaaS).

PREGUNTAS DE PRÁCTICAS

Sobre la práctica 4 (Servicios de dominio de Active Directory AD DS):

1. El usuario 'eovic\Administrador', por defecto, puede iniciar sesión solamente en edc1, edc2 y adc1, que es donde están los DC.
2. La información relativa a los usuarios del dominio *eovic* se encuentra replicada en las máquinas edc1 y edc2.
3. El equipo 'am1' se ha agregado como primer controlador del dominio amsterdam.eovic.csd.

Sobre la práctica 5 (Chat distribuido orientado a objetos basado en RMI):

4. La clase ChatMessage se puede invocar de forma remota.
5. Cuando un usuario se une a un canal, el servidor de chat (objeto ChatServer) lo notifica a los demás usuarios conectados al canal, invocando el método doJoinChannel de los clientes.
6. Una vez implementado, ChatRobot tendrá asociado un usuario de chat, que recibe por parte del canal al que se une el mismo tratamiento que el resto de usuarios.

Sobre la práctica 6 (Java Message Service):

7. El orden recomendado de lanzamiento de los procesos para ejecutar nuestra aplicación de chat con JMS es: CsdMessengerServer – Artemis – CsdMessengerClient(s).
8. Tanto en JMS como en Java RMI se emplea el proveedor de servicio RMIRegistry para establecer el mecanismo de comunicación entre los procesos.
9. Los mensajes que llegan a la cola "users-NOMBREDEUSUARIO" son consumidos por el cliente CSDMessengerClient destinatario del mensaje (NOMBREDEUSUARIO).