

## RECOPILACIÓN PREGUNTAS UNIDAD 9 (desde 2014)

**[Examen final 2º bloque junio 2020 --> examen online]**

**TEMA 9: RELOJES FÍSICOS. BATERÍA 10 preg => Selecc. 4**

Supongamos que hemos de coordinar los relojes de 3 ordenadores, a los que llamamos nodos A, B y C. Sabemos que el reloj del nodo A retrasa 1 milisegundo cada hora, el reloj del nodo B adelanta 5 milisegundos cada hora, y que el reloj del nodo C es un reloj preciso.

Inicialmente los relojes de todos los nodos tienen el mismo valor, que se corresponde con la hora real.

1. En este sistema es más adecuado ejecutar el algoritmo de Berkeley que el algoritmo de Cristian para que los relojes de los nodos del sistema marquen la hora más próxima a la real.

\*Falso

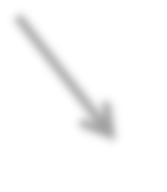
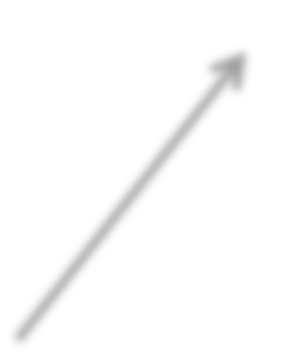
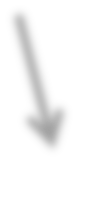
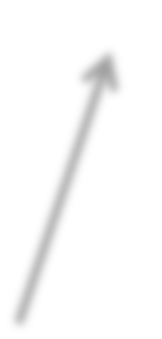
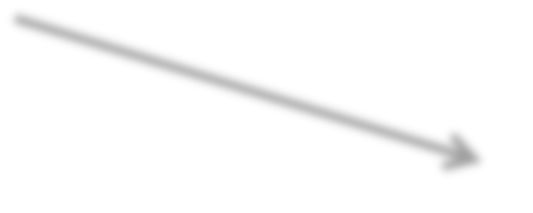
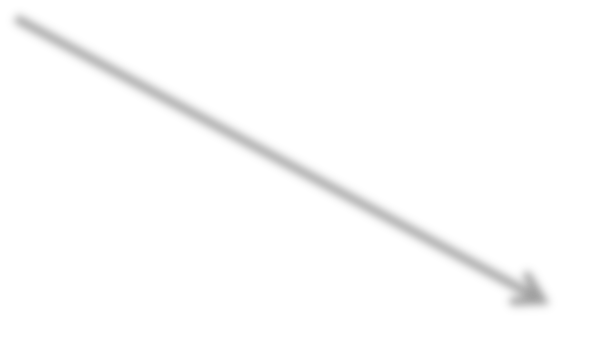
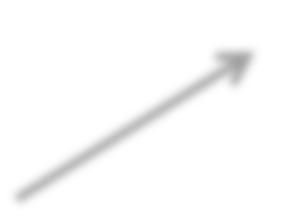
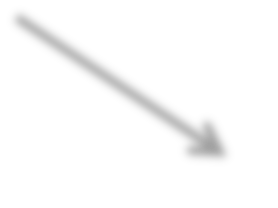
1. Si ejecutamos el algoritmo de Berkeley, siendo el nodo A el servidor, lograremos que los relojes se sincronicen mejor que si empleamos como servidor el nodo B o el nodo C, pues solamente tiene un ligero retraso de 1 milisegundo. \*Falso
2. Si ejecutamos el algoritmo de Cristian empleando como servidor al nodo C, y lo aplicamos de forma periódica cada hora sobre los relojes A y B, lograremos que todos los nodos sincronicen sus relojes aproximadamente al tiempo correcto. \*Verdadero
3. Si ejecutamos el algoritmo de Cristian empleando como servidor al nodo A, de forma periódica cada hora (tanto B como C), lograremos que todos los relojes tengan aproximadamente el mismo tiempo. \*Verdadero
4. Para el algoritmo de Cristian podemos utilizar cualquiera de los nodos como servidor del algoritmo, aunque es preferible que sea el nodo C quien actúe como servidor, al ser el nodo con el reloj más preciso. \*Verdadero
5. No podemos aplicar el algoritmo de Berkeley utilizando al nodo C como servidor, pues como su reloj es preciso, no puede utilizarse para realizar los cálculos que requiere este algoritmo. \*Falso
6. En este ejemplo, para poder aplicar el algoritmo de Cristian necesitamos conocer a priori el tiempo medio que tarda un mensaje en transmitirse desde un nodo concreto hasta el servidor.

\*Falso

1. En este ejemplo, para poder aplicar el algoritmo de Berkeley necesitamos conocer a priori el retardo medio de los mensajes que se intercambian entre el nodo servidor y el resto de nodos. \*Falso
2. En este ejemplo, si ejecutamos el algoritmo de Cristian empleando como servidor al nodo C, para fijar la hora del nodo B se calcula el promedio entre el reloj del nodo C y el reloj del nodo B. \*Falso
3. En este ejemplo, si ejecutamos el algoritmo de Berkeley empleando como servidor al nodo A, podríamos tener que el nodo B sincronizara su reloj respecto al nodo A, pero que el nodo C mantuviera su señal de reloj, al ser un reloj preciso. \*Falso

**TEMA 9: RELOJES LÓGICOS Y VECTORIALES. BATERÍA 20 preg => Selecc. 6**

**Figura 9.1**



P1

e11

e12

e13

e14 e15

e16

P2

e21 e22 e23

e24

e25

e26

P3

e31

e32 e33

e34

e35



Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Solución:

relojes lógicos:

e11: 1, e12: 2, e13: 7, e14: 8, e15: 9, e16:

e21: 2, e22: 3, e23: 4, e24: 5, e25: 10, e26: 11

e31: 1 , e32:5, e33: 6, e34: 7, e35: 8

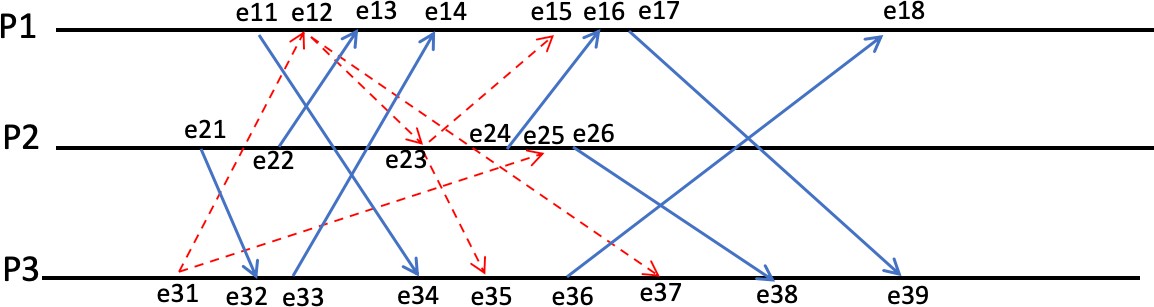
Observando la figura 9.1, podemos afirmar:

1. Los eventos e24 y e32 son concurrentes. \*Verdadero
2. Los eventos e12 y e22 son concurrentes. \*Verdadero
3. Los eventos e12 y e23 son concurrentes. \*Verdadero
4. Los eventos e32 y e14 son concurrentes. \*Falso
5. El reloj vectorial del evento e14 es [4, 3, 3]. \*Verdadero
6. El reloj vectorial del evento e16 es [6, 3, 5]. \*Verdadero
7. El reloj vectorial del evento e23 es [1, 3, 1] \*Verdadero.
8. El reloj vectorial del evento e34 es [2, 3, 4]. \*Verdadero
9. El reloj lógico del evento e22 es 2. \*Falso
10. El reloj lógico del evento e15 es 9. \*Verdadero
11. El reloj lógico del evento e33 es 3. \*Falso
12. El reloj lógico del evento e35 es 8. \*Verdadero
13. El reloj lógico del evento e26 es 11. \*Verdadero
14. El reloj lógico del evento e14 es igual al reloj lógico del evento e35. \*Verdadero
15. Se cumple que e11 --> e33. \*Verdadero
16. Si se cumple que a --> b podemos afirmar que el reloj lógico del evento "a" es menor al reloj lógico del evento "b", es decir C(a) < C(b). \*Verdadero
17. Si se cumple que a || b, sabiendo que son concurrentes, podemos afirmar que los relojes lógicos de estos eventos serán iguales, es decir, que C(a) == C(b). \*Falso

19. Sobre este sistema no se puede ejecutar el algoritmo de Chandy-Lamport. \*Verdadero

## TEMA 9: CHANDY-LAMPORT BATERÍA 12 preg => Selecc. 4

**Figura 9.2**



Respecto a la figura 9.2, donde las líneas discontinuas representan mensajes MARCA de la ejecución del algoritmo de Chandy-Lamport, las líneas continuas representan mensajes normales, mientras que "eX" representan los eventos de envío y/o recepción.

1. Al aplicar el algoritmo de Chandy-Lamport según se indica en la figura 9.2, los nodos almacenan su estado local en los eventos "e12", "e25" y "e31". \*Falso
2. El algoritmo de Chandy-Lamport devolverá un corte consistente que comprenderá el estado global registrado en los eventos "e12", e23", "e31". \*Verdadero
3. La figura 9.2 muestra que con este algoritmo se ha establecido un corte consistente en los eventos "e15", "e25" y "e37". \*Falso
4. En el canal (P1, P3) se habrán registrado 2 mensajes. \*Falso
5. El mensaje e11--> e34 forma parte de estado global calculado por el algoritmo de Chandy- Lamport. \*Verdadero
6. El mensaje e33--> e14 forma parte de estado global calculado por el algoritmo de Chandy- Lamport. \*Falso
7. En el canal (P2, P1) se habrá registrado 1 mensaje. \*Verdadero. (mensaje: e22-->e13)
8. En el canal (P2, P3) se habrá registrado 1 mensaje. \*Verdadero. (mensaje e21-->e32)
9. En el canal (P3, P1) se habrá registrado 1 mensaje. \*Falso. (no se registra ningún mensaje)
10. Para que el nodo iniciador (nodo P3) obtenga toda la información del estado global del sistema, se requiere que los nodos P1 y P2 le envíen todo lo que han registrado al aplicar el algoritmo de Chandy-Lamport, tras los eventos "e15" y "e25", respectivamente.

\*Verdadero

1. El nodo iniciador (nodo P3) obtendrá un estado global inconsistente, pues el nodo P2 ha recibido el mensaje MARCA del nodo P1 antes que el mensaje MARCA del nodo P3.

\*Falso

## TEMA 9: ELECCIÓN LÍDER BATERÍA 10 preg => Selecc. 4

Sea un sistema con 10 nodos (N0, N2, .., N9) donde en un momento determinado están activos solamente los nodos N0, N2, N4, N6 y N8 (y esta información aún no es conocida por los nodos). En el caso de la topología en anillo, asumimos que el sucesor de Ni es N( i+1 mod 10). Se asume que los nodos no activos permanecerán caídos durante todo el tiempo.

1. Si los nodos N2 y N8 inician el algoritmo Bully de elección de líder prácticamente a la vez, se podría llegar a seleccionar dos líderes diferentes (por ejemplo, el nodo N8 y el nodo N9).

\*Falso

1. Si los nodos N4 y N6 inician el algoritmo de elección de líder en anillo prácticamente a la vez, se podría llegar a seleccionar dos líderes diferentes (por ejemplo, el nodo N8 y el nodo N9). \*Falso
2. Si el nodo N2 inicia el algoritmo Bully de elección de líder, enviará un mensaje ELECCIÓN únicamente a los nodos N4, N6 y N8, pues son los nodos con mayor identificación que él que están activos. \*Falso
3. Si el nodo N0 inicia el algoritmo Bully de elección de líder, enviará un mensaje ELECCIÓN a todos los demás nodos del sistema. \*Verdadero
4. Si se inicia el algoritmo Bully, con independencia de quién lo inicie, el nodo N0 acabará siendo elegido líder. \*Falso
5. Si se inicia el algoritmo de elección de líder en anillo, con independencia de quién lo inicie, el nodo N8 acabará siendo elegido líder, asumiendo que se elige como líder al nodo activo con identificador más alto. \*Verdadero
6. Si se utiliza el algoritmo de elección de líder en anillo, si N4 es el iniciador, en algún momento recibirá un mensaje de tipo ELECCIÓN con el contenido (4, {4, 6, 8, 0, 2}).

\*Verdadero

1. Si se utiliza el algoritmo de elección de líder en anillo, si N2 es el iniciador, en algún momento recibirá un mensaje de tipo ELECCIÓN con el contenido (2, {0, 6, 8}). \*Falso
2. Si el nodo N2 inicia el algoritmo de elección de líder en anillo, enviará, entre otros mensajes, un mensaje ELECCIÓN al nodo N4, quien le contestará con un OK. \*Verdadero
3. Si se inicia el algoritmo de elección de líder en anillo, con independencia de quién lo inicie, el nodo N8 acabará enviando un mensaje COORDINADOR al resto de nodos. \*Falso
4. El algoritmo Bully de elección de líder permite obtener la información suficiente para que, con dicha información, los nodos puedan conocer la lista de nodos activos del sistema. \*Falso

## TEMA 9: EXCLUSIÓN MUTUA BATERÍA 10 preg => Selecc. 4

Con el objetivo de comparar los 3 algoritmos de exclusión mutua para sistemas distribuidos vistos en clase, hacemos un experimento con **4 nodos y un cerrojo**, ejecutando el mismo experimento mediante cada uno de los 3 algoritmos. El experimento consiste en lanzar los **4 nodos** al mismo tiempo; cada uno de los nodos trata de cerrar el cerrojo nada más comenzar su ejecución y lo abre al instante, tan pronto como logre cerrarlo. El experimento termina cuando todos los nodos han abierto el cerrojo.

Para el algoritmo centralizado, podemos suponer que el nodo líder es al mismo tiempo uno de los nodos participantes, y los mensajes que se envíe a sí mismo no consumen tiempo, ni se consideran mensajes enviados por la red.

Supongamos que cada mensaje tarda en transmitirse por la red exactamente **5 milisegundos**, y que el tiempo de cómputo o procesamiento es despreciable (tiempo de cómputo cero). Podemos afirmar:

1. En el algoritmo basado en un anillo, el experimento tardará como mínimo 15 milisegundos y no superará los 30 milisegundos. \*Verdadero
2. En el algoritmo basado en un anillo se utilizará un mínimo de 3 mensajes, correspondientes al paso del token. \*Verdadero
3. En el algoritmo centralizado, el experimento tardará exactamente 20 milisegundos.\*Falso
4. El algoritmo centralizado tardará siempre más de 50 milisegundos. \*Falso
5. El algoritmo centralizado utilizará entre 8 y 9 mensajes. \*Verdadero
6. El algoritmo distribuido tardará como mínimo 40 milisegundos y tardará menos de 100 milisegundos. \*Verdadero
7. El algoritmo distribuido utilizará 24 mensajes. \*Verdadero
8. El algoritmo distribuido utilizará como mínimo 6 mensajes y puede llegar a transmitir 60 mensajes. \*Falso

## TEMA 9: CONSENSO --> Se pueden añadir preguntas teóricas de otros algoritmos distribuidos

**BATERÍA 12 preg => Selecc. 5.** (Preguntas reformuladas respecto al 2º parcial --> no son iguales, pero similares)

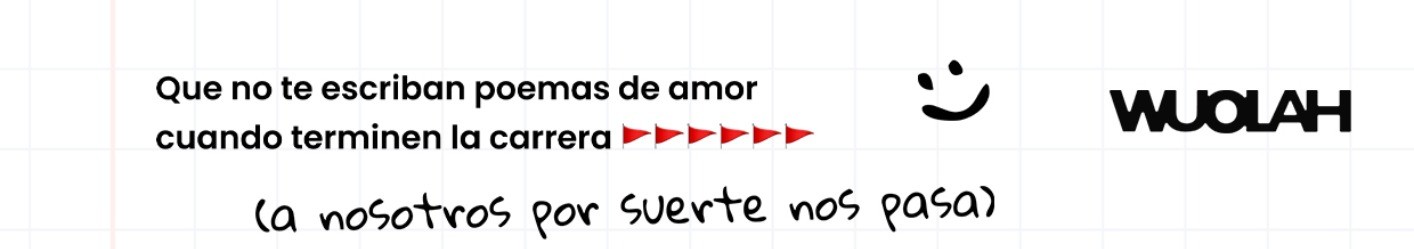
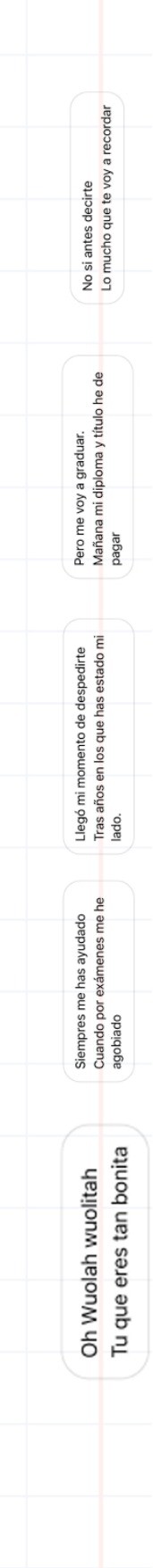
1. El algoritmo de consenso distribuido tolerante a fallos visto en clase no tolera fallos bizantinos, pues este tipo de fallos no puede ser detectado con este algoritmo.

\*Verdadero

1. En el algoritmo de consenso distribuido tolerante a fallos visto en clase necesitará un máximo de n rondas para completar su ejecución, donde 'n' es el número de nodos. \*Falso
2. Si consideramos una red formada por 11 nodos que ejecuten el algoritmo de consenso distribuido tolerante a fallos visto en clase, el algoritmo se bloquea si fallan 6 nodos o más.

\*Verdadero

1. Si consideramos una red formada por 15 nodos que ejecuten el algoritmo de consenso distribuido tolerante a fallos visto en clase, el algoritmo se bloquea si fallan 7 nodos. \*Falso



1. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, el valor que deciden los nodos como decisión final siempre debe ser un valor de entre los valores propuestos por los distintos nodos. \*Verdadero
2. En el algoritmo de consenso tolerante a fallos visto en clase, si hiciéramos un corte preciso en un instante determinado, podríamos tener nodos en rondas distintas y, por tanto, comunicándose con coordinadores distintos. \*Verdadero
3. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, al principio de cada ronda los nodos envían al coordinador de la ronda un mensaje "propose" y, si son nodos ordinarios, se quedan esperando un mensaje "decision". \*Falso
4. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, si consideramos una red formada por 7 nodos que ejecuten este algoritmo, si el coordinador recibe 3 mensajes ACK y un NACK, genera una decisión. \*Falso
5. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, cuando un nodo recibe un "propose", contesta con un mensaje ACK si el valor que le llega en "propose" es igual al valor que tiene almacenado en su variable "lastEstimate" y el número de ronda es mayor o igual a su variable "lastR"; y con un mensaje NACK si dicho valor es distinto. \*Falso
6. Si tenemos una red formada por 5 nodos que ejecutan el algoritmo de consenso distribuido tolerante a fallos visto en clase, es posible que el algoritmo tarde un número de rondas superior a 5, aunque ningún nodo falle, pues sus detectores de fallos pueden no ser precisos. \*Verdadero
7. Un ejemplo de algoritmo de consenso consiste en que cada nodo difunde su propuesta, y todos eligen la propuesta del nodo con mayor identificador. \*Verdadero
8. Un ejemplo de algoritmo de consenso consiste en que cada nodo difunde su propuesta, y se elige la propuesta de menor valor. \*Verdadero

## [2º parcial 2020 --> examen online]

**TEMA 9: RELOJES FÍSICOS**

**BATERÍA 11 preg => Selecc. 3**

Supongamos que hemos de coordinar los relojes de 3 ordenadores, a los que llamamos nodos A, B y C. Sabemos que el reloj del nodo A es muy preciso, el reloj del nodo B retrasa 1 milisegundo cada hora, pero tiene una fecha correcta y el reloj del nodo C tiene como fecha 1970, cuando estamos en el año 2020.

1. En este sistema, es más adecuado ejecutar el algoritmo de Berkeley para sincronizar los relojes, que el algoritmo de Cristian. \*Falso
2. Si ejecutamos el algoritmo de Berkeley, siendo el nodo A el servidor, lograremos que los relojes se sincronicen mejor que si empleamos como servidor el nodo B.\*Falso
3. Si ejecutamos el algoritmo de Cristian empleando como servidor al nodo A, de forma periódica cada hora (tanto B como C), lograremos que todos los nodos sincronicen sus relojes aproximadamente al tiempo correcto. \*Verdadero
4. Si ejecutamos el algoritmo de Cristian empleando como servidor al nodo B, de forma periódica cada hora (tanto A como C), lograremos que todos los relojes tengan aproximadamente el mismo tiempo. \*Verdadero
5. Si ejecutamos el algoritmo de Berkeley utilizando como servidor al nodo C, los ordenadores sincronizarán sus relojes de forma que la fecha de todos y cada uno de ellos será posterior a 2000 y anterior a 2005, suponiendo que no se descarta ninguno de los valores de los relojes.

\*Verdadero

1. Para el algoritmo de Cristian podemos utilizar cualquiera de los nodos como servidor del algoritmo, aunque es preferible que sea el nodo A quien actúe como servidor, al ser el nodo con el reloj más preciso. \*Verdadero
2. No podemos aplicar el algoritmo de Berkeley incluyendo al nodo C entre los nodos del sistema, pues como su fecha es muy antigua, provocará que todos los relojes, incluyendo el del nodo servidor, deban retrasarse. \*Falso
3. Para poder aplicar el algoritmo de Cristian necesitamos conocer a priori el tiempo medio que tarda un mensaje en transmitirse desde un nodo concreto hasta el servidor. \*Falso
4. Para poder aplicar el algoritmo de Berkeley necesitamos conocer a priori el retardo medio de los mensajes que se intercambian entre el nodo servidor y el resto de nodos. \*Falso
5. Si ejecutamos el algoritmo de Cristian empleando como servidor al nodo A, para fijar la hora del nodo B se calcula el promedio entre el reloj del nodo A y el reloj del nodo B. \*Falso
6. Si ejecutamos el algoritmo de Berkeley empleando como servidor al nodo A, podríamos tener que el nodo B sincronizara su reloj respecto al nodo A, pero que el nodo C mantuviera su reloj en la fecha 1970. \*Falso

## Imagen que contiene avión, pasto, estacionado, aire Descripción generada automáticamenteTEMA 9: RELOJES LÓGICOS Y VECTORIALES. BATERÍA 18 preg => Selecc. 5

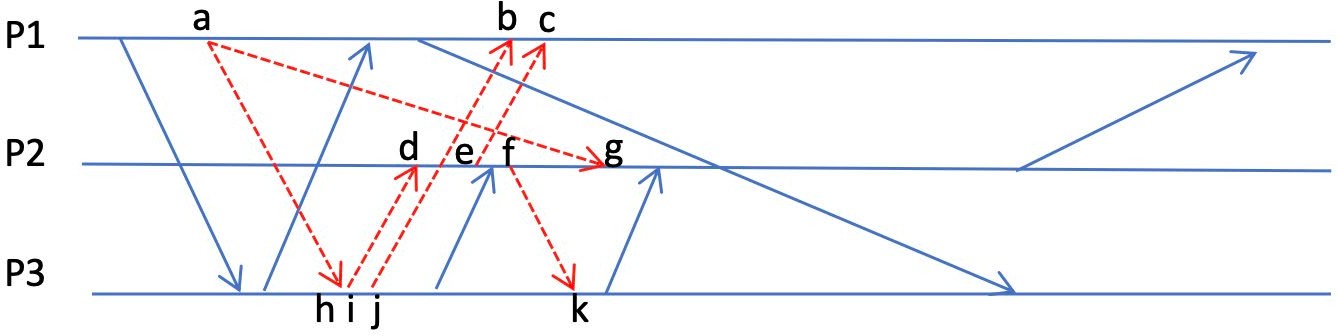
**Figura 9.1**

Observando la figura 9.1, podemos afirmar:

1. Los eventos e11 y e23 son concurrentes. \*Falso
2. Los eventos e12 y e24 son concurrentes. \*Verdadero
3. Los eventos e12 y e32 son concurrentes. \*Verdadero
4. Los eventos e32 y e14 son concurrentes. \*Verdadero
5. El reloj vectorial del evento e14 es [4, 0, 1].\*Verdadero
6. El reloj vectorial del evento e23 es [1, 3, 3].\*Verdadero
7. El reloj vectorial del evento e34 es [0, 0, 4].\* Falso
8. El reloj vectorial del evento e22 es [0, 2, 3].\*Falso
9. El reloj lógico del evento e23 es 5.\*Falso
10. El reloj lógico del evento e15 es 7.\*Verdadero
11. El reloj lógico del evento e33 es 4.\*Verdadero
12. El reloj lógico del evento e13 es igual al reloj lógico del evento e21.\*Verdadero
13. Se cumple que e11 --> e33, por lo que podemos afirmar que el reloj lógico de e11 es menor al reloj lógico de e33, es decir C(e11) < C(e33). \*Verdadero
14. Se cumple que e32 --> e15, por lo que podemos afirmar que el reloj lógico de e32 es menor al reloj lógico de e15, es decir C(e32) < C(e15). \*Verdadero
15. Se cumple que e12 || e24, por lo que podemos afirmar que los relojes lógicos de estos eventos son iguales, es decir, que C(e12) == C(e24). \*Falso
16. Como C(e22) == C(e14) podemos afirmar que se cumple que e22 || e14. \*Verdadero
17. Sobre este sistema no se puede ejecutar el algoritmo de Chandy-Lamport. \*Falso
18. Si se aplicara el algoritmo de Chandy-Lamport sobre este sistema, se podrían llegar a obtener instantáneas inconsistentes, pues no se cumplen todos los requisitos de dicho algoritmo. \*Falso.

## TEMA 9: CHANDY-LAMPORT. BATERÍA 12 preg => Selecc. 4

**Figura 9.2**



Respecto a la figura 9.2, donde las líneas discontinuas representan mensajes MARCA de la ejecución del algoritmo de Chandy-Lamport, las líneas continuas representan mensajes normales, mientras que las letras representan los eventos de envío o recepción de los mensajes MARCA.

1. Al aplicar el algoritmo de Chandy-Lamport según se indica en la figura 9.2, los nodos almacenan su estado local en los eventos "a", "h" y "g", al ser cuando llega el mensaje MARCA desde el iniciador al resto de nodos. \*Falso
2. El algoritmo de Chandy-Lamport devolverá un corte consistente que comprenderá el estado global registrado en los eventos "a", "h", "d". \*Verdadero
3. Con este algoritmo se establece un corte consistente en los eventos "c", "g" y "k", ya que estos representan los últimos mensajes MARCA que se reciben en cada nodo. \*Falso
4. En el canal (P1, P2) se habrá registrado 0 mensajes. \*Verdadero
5. En el canal (P1, P3) se habrá registrado 0 mensajes. \*Verdadero
6. En el canal (P2, P1) se habrá registrado 1 mensaje. \*Falso
7. En el canal (P2, P3) se habrá registrado 2 mensajes. \*Falso
8. En el canal (P3, P1) se habrá registrado 1 mensaje. \*Verdadero
9. En el canal (P3, P2) se habrá registrado 1 mensaje. \*Falso
10. Para que el nodo iniciador (nodo P1) obtenga el estado global, se requiere que tanto P2 como P3 le envíen la información del estado global que han registrado y de los mensajes registrados en sus canales de entrada al nodo P1, tras los eventos "g" y "k", respectivamente.

\*Verdadero

1. El nodo iniciador (nodo P1) no podrá obtener el estado global, pues el nodo P2 ha recibido el mensaje MARCA del nodo P3 antes que el mensaje MARCA del nodo P1. \*Falso
2. Esta figura no representa de forma apropiada el algoritmo de Chandy-Lamport, pues según dicho algoritmo, los nodos recibirán siempre su primer mensaje MARCA a través del nodo iniciador, que en este caso es el nodo P1. \*Falso

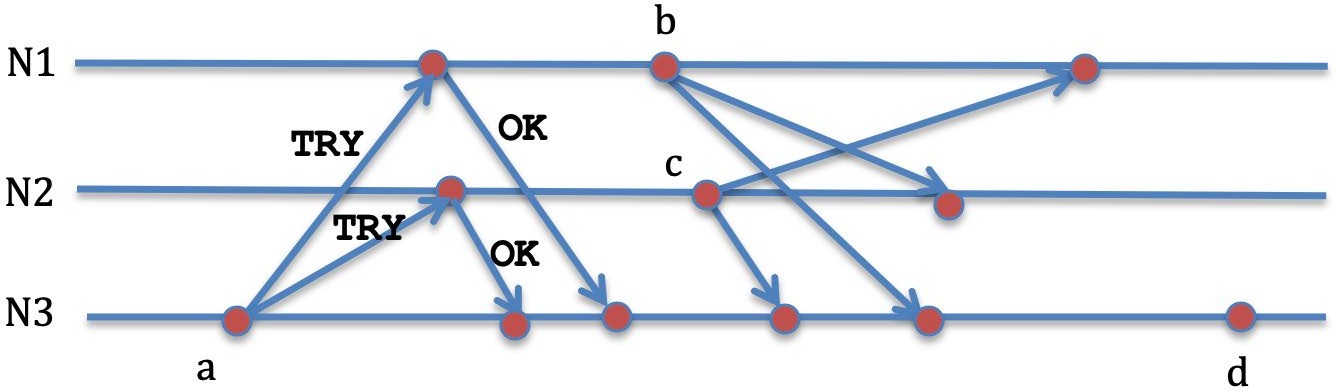
## TEMA 9: ELECCIÓN LÍDER BATERÍA 10 preg => Selecc. 3

Sea un sistema con 6 nodos (N0, N2, .., N5) donde en un momento determinado están activos solamente los nodos N0, N3 y N4 (y esta información aún no es conocida por los nodos). En el caso de la topología en anillo, asumimos que el sucesor de Ni es N( i+1 mod 6).

1. El algoritmo de elección de líder sobre anillos falla si los nodos N0 y N4 inician el algoritmo de forma prácticamente simultánea. \*Falso
2. El algoritmo Bully de elección de líder falla si los nodos N3 y N4 inician el algoritmo de forma simultánea. \*Falso
3. Si el nodo N0 inicia el algoritmo Bully de elección de líder, enviará un mensaje ELECCIÓN únicamente a los nodos N3 y N4, pues son los nodos que están activos. \*Falso
4. Si el nodo N3 inicia el algoritmo Bully de elección de líder, enviará un mensaje ELECCIÓN a todos los nodos del sistema. \*Falso
5. Si se inicia el algoritmo Bully, con independencia de quién lo inicie, el nodo N4 acabará siendo elegido líder. \*Verdadero
6. Si se utiliza el algoritmo de elección de líder en anillo, si N4 es el iniciador, en algún momento recibirá un mensaje de tipo ELECCIÓN con el contenido (4, {4, 0, 3}). \*Verdadero
7. Si el nodo N0 inicia el algoritmo de elección de líder en anillo, enviará un mensaje ELECCIÓN únicamente a los nodos N3 y N4, pues son los nodos que están activos. \*Falso
8. Si el nodo N3 inicia el algoritmo de elección de líder en anillo, enviará un mensaje ELECCIÓN al nodo N4, quien le contestará con un OK. \*Verdadero
9. Si se inicia el algoritmo de elección de líder en anillo, con independencia de quién lo inicie, el nodo N4 acabará enviando un mensaje COORDINADOR al resto de nodos. \*Falso
10. El algoritmo de elección de líder en anillo permite que los nodos conozcan la lista de nodos activos del sistema. \*Verdadero

## TEMA 9: EXCLUSIÓN MUTUA PARTE 1 BATERÍA 8 preg => Selecc. 3

**Figura 9.3**



La figura 9.3 representa la ejecución del algoritmo distribuido de exclusión mutua visto en clase. En esta figura se representan 3 nodos que comparten un único cerrojo. Inicialmente el nodo 3 cierra el cerrojo con éxito. Para ello ha enviado mensajes TRY en el evento a, y ha recibido los correspondientes mensajes OK. Posteriormente los nodos 1 y 2 tratan de cerrar el cerrojo (eventos b y c). Los identificadores de los nodos son N1 < N2 < N3. Y además no hay más eventos en el sistema que los aquí indicados. Además, cerrar el cerrojo implica ejecutar el protocolo de entrada en la sección crítica, mientras que abrir el cerrojo implica ejecutar el protocolo de salida de la sección crítica.

Podemos afirmar lo siguiente:

1. Podemos asegurar que el nodo 1 logrará el cerrojo antes que el nodo 2. \*Verdadero
2. Tanto el nodo 1, como el nodo 2, envían 2 mensajes TRY, pues ambos nodos necesitan la confirmación de todos los demás nodos. \*Verdadero
3. Suponiendo que el evento 'd' es el evento correspondiente a la apertura del cerrojo por parte del nodo 3. Podemos afirmar que alguno de los nodos N1 o N2 obtendrá el cerrojo en cierto instante posterior a 'd'. \*Verdadero
4. El nodo 1 o el nodo 2 lograrán el cerrojo en cierto momento a partir del instante 'd' aunque la red no garantice entrega FIFO. \*Verdadero
5. Suponiendo que el evento 'd' es el evento de apertura del cerrojo por parte de nodo 3, podemos afirmar que hasta ese instante 'd', tanto el nodo 1 como el nodo 2 habrán recibido exactamente un mensaje OK. \*Falso
6. Suponiendo que el evento 'd' es el evento de apertura del cerrojo por parte de nodo 3, la suma total de los mensajes OK entregados en el sistema desde el evento 'a' hasta el evento 'd' es como máximo 3 y como mínimo 2. \*Verdadero
7. Como los eventos 'b' y 'c' son concurrentes, cualquiera de los dos nodos podría obtener el cerrojo cuando lo abra el nodo 3. \*Falso
8. Suponiendo que el evento 'd' es el evento de apertura del cerrojo por parte de nodo 3, la suma total de los mensajes OK entregados en el sistema desde el evento 'a' hasta el evento 'd' es exactamente 3. \*Falso

**TEMA 9: EXCLUSIÓN MUTUA PARTE 2**

**BATERÍA 8 preg => Selecc. 3**

Con el objetivo de comparar los 3 algoritmos de exclusión mutua para sistemas distribuidos vistos en clase, hacemos un experimento con 3 nodos y un cerrojo, ejecutando el mismo experimento mediante cada uno de los 3 algoritmos. El experimento consiste en lanzar los 3 nodos al mismo tiempo; cada uno de los nodos trata de cerrar el cerrojo nada más comenzar su ejecución y lo abre al instante, tan pronto como logre cerrarlo. El experimento termina cuando todos los nodos han abierto el cerrojo.

Para el algoritmo centralizado, podemos suponer que el nodo líder es al mismo tiempo uno de los nodos participantes, y los mensajes que se envíe a sí mismo no consumen tiempo, ni se consideran mensajes enviados por la red.

Supongamos que cada mensaje tarda en transmitirse por la red exactamente 10 milisegundos, y que el tiempo de cómputo o procesamiento es despreciable (tiempo de cómputo cero). Podemos afirmar:

1. En el algoritmo basado en un anillo, el experimento tardará como mínimo 20 milisegundos y no superará los 30 milisegundos. \*Verdadero
2. En el algoritmo basado en un anillo, el experimento tardará como mínimo 0 milisegundos y puede llegar a tardar 60 milisegundos. \*Falso
3. En el algoritmo centralizado, el experimento tardará 30 milisegundos. \*Falso
4. El algoritmo centralizado tardará como mínimo 50 milisegundos y puede llegar a tardar 80 milisegundos. \*Falso
5. El algoritmo distribuido tardará como mínimo 40 milisegundos y tardará menos de 100 milisegundos. \*Verdadero
6. El algoritmo distribuido utilizará 12 mensajes. \*Verdadero
7. El algoritmo distribuido utilizará como mínimo 6 mensajes y puede llegar a transmitir 20 mensajes. \*Falso
8. El algoritmo centralizado transmitirá únicamente 4 mensajes por la red. \*Falso

**TEMA 9: CONSENSO**

**BATERÍA 12 preg => Selecc. 4**

Sobre el algoritmo de consenso tolerante a fallos visto en clase.

1. El algoritmo de consenso distribuido tolerante a fallos visto en clase tolera fallos de parada y fallos bizantinos. \*Falso
2. En el algoritmo de consenso distribuido tolerante a fallos visto en clase puede necesitar para completar su ejecución un número no acotado de rondas y de mensajes. \*Verdadero
3. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, en el mejor de los casos necesita orden (n) mensajes para alcanzar consenso, donde 'n' es el número de nodos.

\*Verdadero

1. Si consideramos una red formada por 7 nodos que ejecuten el algoritmo de consenso distribuido tolerante a fallos visto en clase, el algoritmo se bloquea si fallan 4 nodos o más.

\*Verdadero

1. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, el valor que deciden los nodos como decisión final es el promedio de los valores propuestos por los distintos nodos. \*Falso
2. En el algoritmo de consenso tolerante a fallos visto en clase, el nodo coordinador de cada ronda debe ser elegido previamente empleando un algoritmo de elección de líder. \*Falso
3. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, al principio de cada ronda los nodos envían al coordinador de la ronda un mensaje "estimate" y, si son nodos ordinarios, se quedan esperando un mensaje "propose". \*Verdadero
4. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, si consideramos una red formada por 7 nodos que ejecuten este algoritmo, si el coordinador recibe 4 mensajes ACK (incluyendo el suyo), genera una decisión. \*Verdadero
5. En el algoritmo de consenso distribuido tolerante a fallos visto en clase, cuando un nodo recibe un "propose", contesta con un mensaje ACK si el valor que le llega en "propose" es igual a su valor estimate; y con un mensaje NACK si dicho valor es distinto. \*Falso
6. Si tenemos una red formada por 3 nodos que ejecutan el algoritmo de consenso distribuido tolerante a fallos visto en clase, es posible que el algoritmo tarde un número de rondas superior a 3, aunque ningún nodo falle. \*Verdadero
7. En el algoritmo de consenso distribuido tolerante a fallos visto en clase garantiza que antes o después todos los nodos correctos deciden el mismo valor garantizando además que este valor habrá sido propuesto por algún nodo correcto, suponiendo una mayoría de nodos correctos. \*Verdadero
8. Un ejemplo de algoritmo de consenso consiste en que cada nodo difunde su propuesta, y todos eligen la propuesta del nodo con mayor identificador. \*Verdadero

# [FINAL 2019]

Sobre los algoritmos de sincronización de relojes físicos:

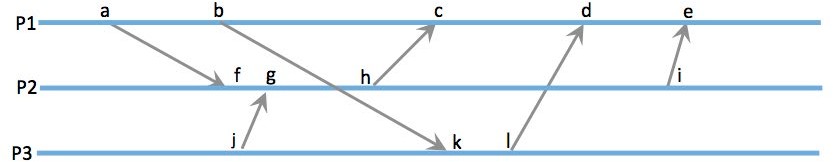
|  |  |
| --- | --- |
| 1. El algoritmo de Cristian para la sincronización de relojes se basa en el uso de un servidor con un reloj más exacto en el cual todos confían. | V |
| 2. El algoritmo de Cristian sólo puede aplicarse si el tiempo que necesita el servidor para procesar el mensaje es 10 milisegundos. | F |
| 3. En el algoritmo de Berkeley, no es necesario conocer a priori el retardo  medio de los mensajes que intercambian los distintos nodos. | V |
| 4. El algoritmo de Berkeley permite ajustar todos los relojes a una misma hora,  pero no garantiza que se trate de la hora exacta (real). | V |

Sobre los relojes lógicos de Lamport y los relojes vectoriales:

|  |  |
| --- | --- |
| 5. 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 |
| 6. 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 | V |
| 7. 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 |

Sobre el algoritmo de Chandy-Lamport:

|  |  |
| --- | --- |
| 8. Por sus características, el algoritmo de Chandy-Lamport puede considerarse descentralizado. | V |
| 9. 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 sistema distribuido, cada uno en un nodo distinto:

|  |  |
| --- | --- |
| 10. Los relojes de Lamport de los eventos "f" y "b" son iguales, por lo que estos dos eventos son concurrentes. | V |
| 11. El reloj de Lamport del evento "i" es 5. | V |
| 12. El reloj vectorial en "h" es V(h)=[1,3,1] | V |
| 13. El reloj vectorial de "k" es V(k)=[2,0,2]. | V |

Sobre los algoritmos de exclusión mutua y consenso:

|  |  |
| --- | --- |
| 14. En el algoritmo distribuido de exclusión mutua, todo nodo necesita mantener una lista de respuestas OK pendientes. | V |
| 15. 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. | F |
| 16. 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. | F |
| 17. 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:

|  |  |
| --- | --- |
| 18. 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 |
| 19. El algoritmo Bully requiere que cada nodo disponga de un identificador único, que además debe ser conocido por los otros nodos. | V |
| 20. 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 | F |

|  |  |
| --- | --- |
| otro iniciador). |  |

# [2º parcial 2019]

Sobre los algoritmos de consenso:

|  |  |
| --- | --- |
| 1. El algoritmo de consenso tolerante a fallos no soporta fallos bizantinos. | V |
| 2. 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 |
| 3. El algoritmo de consenso tolerante a fallos con N nodos requiere un total de N rondas. | F |
| 4. 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:

|  |  |
| --- | --- |
| 1. 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 |
| 2. En el algoritmo de Berkeley cada cliente puede sincronizar su reloj con independencia del resto de clientes. | F |
| 3. El reloj de un nodo nunca debe adelantarse. | F |

Sobre los relojes lógicos de Lamport y los relojes vectoriales:

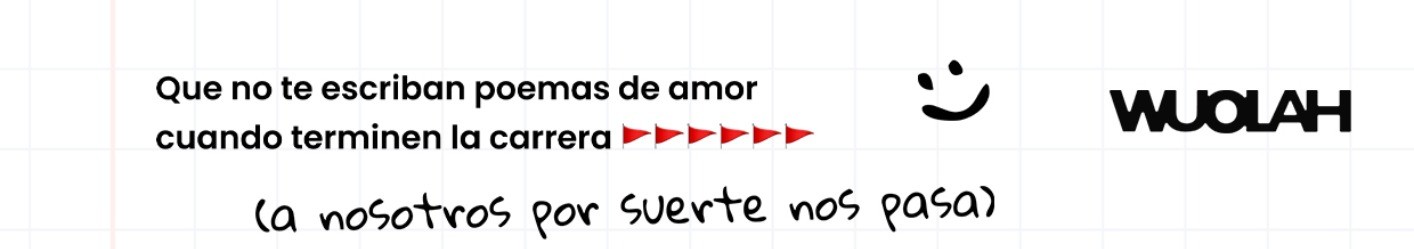
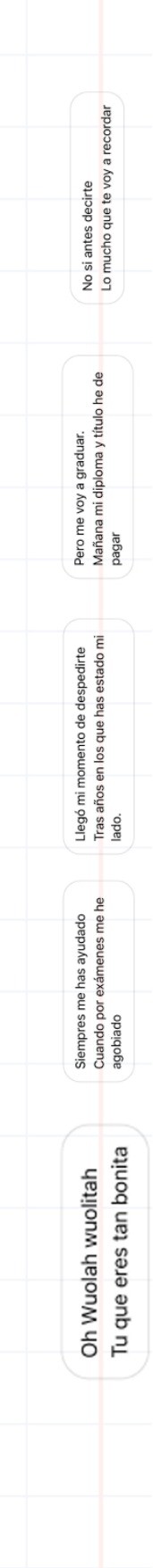
|  |  |
| --- | --- |
| 4. Asumiendo relojes lógicos de Lamport, si a->b implica que C(a)<C(b), donde  C(x) representa el valor de contador asociado con el evento x. | V |
| 5. Con relojes lógicos vectoriales, si V(a)=[0,0,1] y V(b)=[2,2,0], entonces a || b | V |
| 6. Con relojes lógicos vectoriales, V(a)<V(b) implica que a->b | V |
| 7. 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 |
| 8. 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:

|  |  |
| --- | --- |
| 9. En el algoritmo de elección de líder con topología en anillo, el líder es aquel nodo que posee el token. | F |
| 10. En el algoritmo de Bully, todo nodo que recibe OK sabe que no será elegido como líder. | V |

Sobre el algoritmo de Chandy-Lamport:

|  |  |
| --- | --- |
| 11. Chandy-Lamport sólo funciona si hay conectividad total y todos los canales son Fiables y FIFO. | V |
| 12. El algoritmo de Chandy-Lamport garantiza que la instantánea que se obtiene es | V |



|  |  |
| --- | --- |
| consistente. |  |
| 13. 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:

|  |  |
| --- | --- |
| 14. 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 |
| 15. En el algoritmo centralizado, en caso de que falle el coordinador, podemos aplicar el algoritmo de Bully para elegir a otro. | V |
| 16. 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 |
| 17. En el algoritmo de exclusión mutua distribuido, el protocolo de salida consiste en difundir ok a todos los nodos. | F |
| 18. En el algoritmo de exclusión mutua en anillo, sólo puede acceder a la SC el nodo que tiene el token. | V |
| 19. 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 |

## [FINAL 2018]

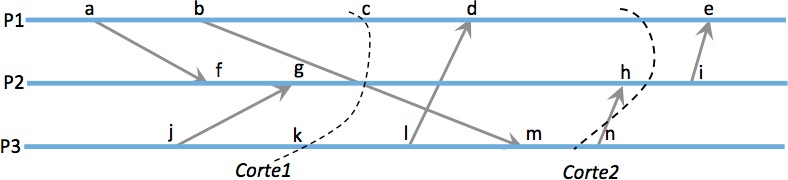
Sobre la sincronización de relojes físicos y el uso de relojes lógicos:

|  |  |
| --- | --- |
| 1. En el algoritmo de Berkeley, se requiere de la existencia de un reloj muy preciso con el que poder sincronizar los relojes de los nodos del sistema. | F |
| 2. Dado que los relojes lógicos de Lamport, junto con los identificadores de los nodos, permiten  establecer un orden total de los eventos, se pueden utilizar éstos para desempatar entre dos mensajes TRY del algoritmo de exclusión mutua distribuido. | V |
| 3. Con relojes lógicos podemos conocer la hora precisa en la que ocurren los diferentes eventos en un  sistema distribuido | F |
| 4. En el algoritmo de Cristian, si el servidor contesta a un cliente con un valor igual a 1500, el nuevo  valor del reloj local del cliente debe hacerse igual a 1500 más un valor de ajuste que también será facilitado por el servidor. | F |
| 5. Supongamos que utilizamos la notación "reloj lógico de Lamport . identificador del nodo" y sabemos que el evento “a” tiene asociado el valor 7.1 y el evento “b” tiene asociado el valor 9.2, entonces  podemos afirmar que el evento “a” ha ocurrido antes que el evento “b”. | F |
| 6. Siendo V(a) = [9, 5, 8, 7, 3] y V(b) = [3, 4, 1, 9, 10] dos relojes vectoriales, podemos afirmar que los eventos “a” y “b” son concurrentes entre sí. | V |

Sobre el algoritmo de consenso en presencia de fallos visto en clase:

|  |  |
| --- | --- |
| 7. Garantiza que los nodos correctos deciden el mismo valor, aun cuando fallen la mitad de los nodos. | F |
| 8. Tolera fallos bizantinos y fallos de parada. | F |
| 9. En caso de fallos múltiples puede requerir un número de rondas elevado | V |

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





Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

|  |  |
| --- | --- |
| 10. Los relojes de Lamport de los eventos "c " y "k" son cero, puesto que no corresponden con ningún evento de envío o recepción de mensajes. | F |
| 11. Se observa que h-->n | F |
| 12. El reloj de Lamport de “h” es C(h)=6 | V |
| 13. El reloj vectorial de V(g)=[1, 1, 1 ] | F |
| 14. En esta figura todos los eventos que tienen el mismo valor de reloj lógico de Lamport son concurrentes. | V |
| 15. El algoritmo de Chandy-Lamport podría habernos facilitado el corte indicado por Corte1, pero no así el Corte2. | V |

Sobre los algoritmos de elección de líder y de exclusión mutua en sistemas distribuidos vistos en clase:

|  |  |
| --- | --- |
| 16. En el algoritmo de elección de líder para anillos, el nodo iniciador finaliza su trabajo enviando por el  anillo un mensaje de tipo COORDINADOR en el que se indica cual es el nuevo coordinador a partir de ahora. | V |
| 17. 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. | F |
| 18. En el algoritmo de exclusión mutua para anillos, cuando un nodo sale de la sección crítica devuelve el  token al nodo coordinador, para que este se lo dé eventualmente al siguiente nodo interesado en entrar en la sección crítica. | F |

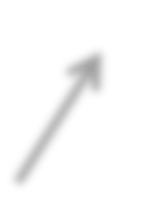
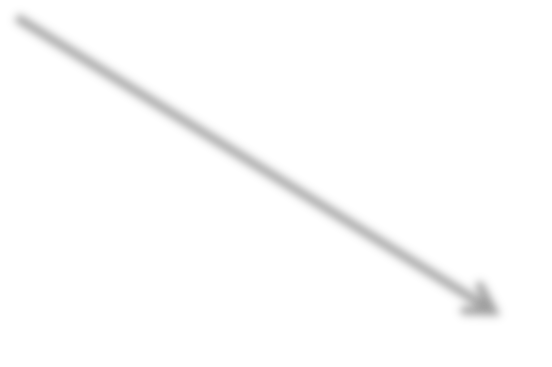
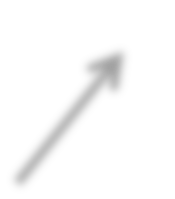
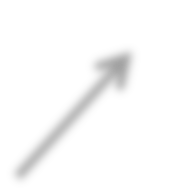
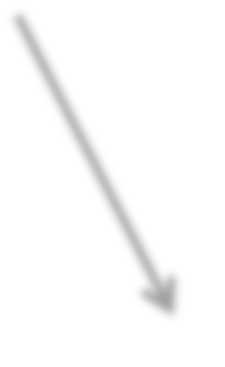
# [2º PARCIAL 2018]

Sobre la sincronización de relojes físicos y el uso de relojes lógicos:

|  |  |
| --- | --- |
| 1. Dados los relojes lógicos C(a)=5 y C(b)=12, podemos afirmar que a-->b | F |
| 2. Dados los relojes vectoriales V(x)=[2,3,6] y V(z)=[1,1,3] podemos afirmar que z--> x | V |
| 3. Si los eventos "e" y "f" establecen el envío y recepción de un mismo mensaje "m", produciéndose "e" en el nodo N1 y "f" en el nodo N2, y si V(e)=[1,1,2], entonces V(f)=[1,2,2], con independencia del valor de reloj vectorial que tuviera el nodo N2 antes de recibir el  mensaje "m". | F |
| 4. Mediante algoritmos de sincronización de relojes, podemos sincronizar el reloj físico de dos ordenadores de forma que tengan un valor bastante similar, pero no exactamente idéntico,  pues siempre existirá cierto error en la sincronización. | V |
| 5. Un sistema que utilice el algoritmo de Berkeley para sincronizar los relojes, deberá contar con algún algoritmo de elección de líder. | V |

Dado el siguiente diagrama temporal de eventos, podemos afirmar:

e1 e1 e1 e1 e1



e2

e2

e2

e2

e3

e3

e3

e3

|  |  |
| --- | --- |
| 6. e11 || e22. | V |
| 7. V(e14) = [4,4,3] | V |
| 8. e12 --> e24 | F |
| 9. C(e34) = 5 | V |
| 10. Si iniciamos la captura del estado global en el primer nodo, justo después de ejecutar el  evento e13 y antes de ejecutar el evento e14, proporcionará un estado global que siempre incluirá como mensaje en tránsito al mensaje enviado desde e13 a e34. | F |

Sobre los algoritmos exclusión mutua, elección de líder y consenso:

|  |  |
| --- | --- |
| 11. El algoritmo de elección de líder para anillos es un ejemplo de algoritmo de consenso en el  que los nodos participantes acuerdan que el líder es el nodo de la lista "nodos participantes" con identificador más alto. | V |
| 12. En el algoritmo Bully, el iniciador envía un mensaje ELECCIÓN a todos los nodos del grupo, para así detectar qué nodos están activos y cuáles han caído. | F |
| 13. En el algoritmo de exclusión mutua distribuido, cuando un nodo recibe un mensaje TRY de otro nodo, y no quiere entrar en la sección crítica, simplemente responde con un mensaje OK. | V |
| 14. En el algoritmo de consenso distribuido considerando fallos, el nodo coordinador ha sido  previamente elegido empleando un algoritmo de elección de líder. | F |
| 15. En el algoritmo de consenso distribuido considerando fallos, al principio de cada ronda los nodos envían al coordinador de la ronda un mensaje "estimate" y, si son nodos ordinarios, se  quedan esperando un mensaje "propose". | V |
| 16. En un sistema con 7 nodos, el nodo coordinador del algoritmo de consenso distribuido considerando fallos se quedaría bloqueado si fallan 3 nodos. | F |
| 17. En caso de que no ocurran fallos, en el algoritmo de consenso distribuido considerando  fallos, el número de mensajes necesarios para alcanzar consenso entre "n" nodos, es O(n2). | F |

# [FINAL 2017]

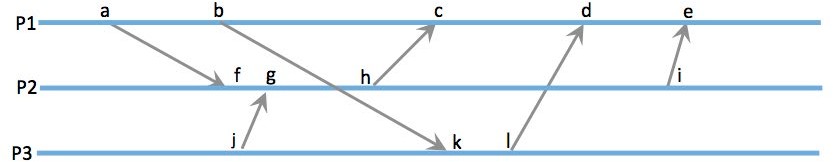
En un sistema distribuido con 4 nodos se emplea el algoritmo de Berkeley para sincronizar sus relojes. Uno de los nodos (nodo A) actúa como servidor del algoritmo, mientras que los otros nodos (nodos B, C y D) actúan como clientes. Supongamos que cada nodo tiene un reloj que indica el número de ticks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

|  |  |
| --- | --- |
| 19. En el primer paso del algoritmo, el nodo A enviará un mensaje al resto de nodos, cuyo contenido será 2000. | V |
| 20. 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*. | V |
| 21. La diferencia promedio calculada por A será de 4 unidades de tiempo, por lo que el servidor ajustará su reloj incrementándolo en 4. | F |
| 22. El nodo D recibe como valor de ajuste -1. | V |

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

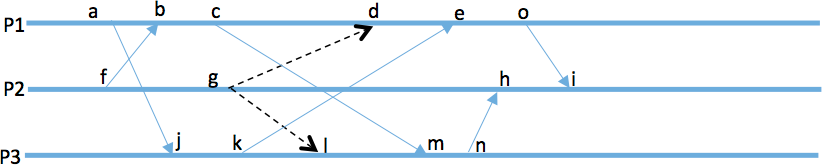
|  |  |
| --- | --- |
| 23. En el algoritmo de exclusión mutua centralizado, para que un nodo entre en la sección crítica, primero el resto de nodos deben notificar al coordinador que están  de acuerdo. | F |
| 24. En el algoritmo de exclusión mutua para anillos, cuando un nodo sale de la sección crítica devuelve el token al nodo coordinador, para que éste se lo dé eventualmente al siguiente nodo interesado en entrar en la sección crítica. | F |
| 25. 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 | V |

|  |  |
| --- | --- |
| se les ha contestado). |  |
| 26. En el algoritmo distribuido, en caso de empate (varios nodos solicitan acceso al mismo tiempo), se desempata utilizando un algoritmo de elección de líder. | F |

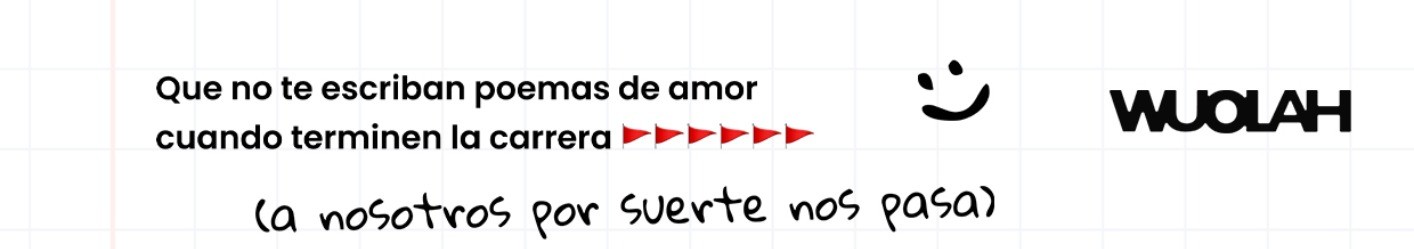
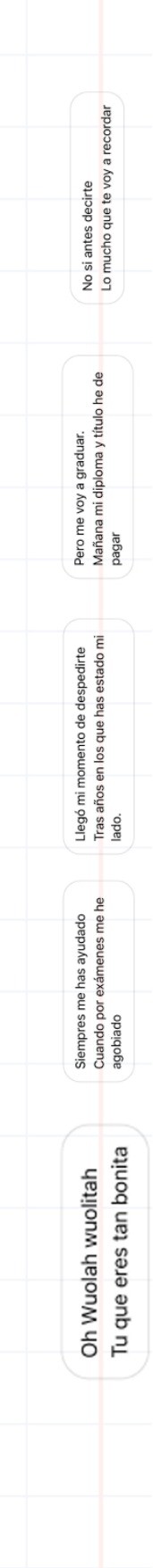
Dado el siguiente cronograma que muestra la ejecución de tres procesos en un sistema distribuido, cada uno en un nodo distinto:

|  |  |
| --- | --- |
| 27. Los relojes de Lamport de los eventos "c ", "i" son iguales. | V |
| 28. Teniendo en cuenta solamente los valores de los relojes de Lamport podemos afirmar que el evento “g” ocurre antes que el evento “l” | F |
| 29. El reloj vectorial en "h" es V(h)=[1, 2, 1 ] | F |
| 30. En esta figura todos los eventos que tienen el mismo valor de reloj lógico de Lamport son concurrentes. | V |
| 31. El reloj vectorial de "e" es V(e)=[5, 4, 3 ]. | V |
| 32. Con relojes lógicos de Lamport se garantiza que dos eventos correspondientes a nodos distintos no pueden tener asociado el mismo valor lógico. | F |

Dada la siguiente figura, donde se tienen 3 nodos en un sistema que cumple con las condiciones 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.



|  |  |
| --- | --- |
| 33. Cuando el nodo P1 envíe el mensaje de MARCA al nodo P2, éste deberá llegar antes  del evento i. | V |
| 34. Cuando el nodo P3 envíe el mensaje de MARCA al nodo P2, éste deberá llegar después  del evento h y antes del evento i. | F |
| 35. Cuando finalice el algoritmo, en total se habrán registrado 2 mensajes en tránsito. | V |
| 36. El nodo P3 registrará su estado local en el evento l. | V |



## [2º PARCIAL 2017]

Sobre la sincronización de relojes:

|  |  |
| --- | --- |
| 37. En el algoritmo de Cristian cada cliente puede sincronizar su reloj con independencia del resto de clientes. | V |
| 38. 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 |
| 39. Con relojes lógicos establecemos un orden causal (causa-efecto), de forma que  decimos que el evento "a" precede al evento "b" (a->b) si todos los nodos observan exactamente dicho orden. | V |
| 40. En los relojes vectoriales, para cualquier par de vectores distintos V(a) V(b), se cumple que V(a)<V(b) o V(b)<V(a). | F |
| 41. Si en el algoritmo de Berkeley el reloj del servidor es igual a 10.005 y el de un cliente 10.000, el servidor comunicará a dicho cliente que debe adelantar su reloj en 5 unidades de tiempo, independientemente de los valores de los relojes del resto de  clientes. | F |
| 42. Si el reloj lógico de Lamport asociado al evento “a” es igual a 3 y el reloj lógico de Lamport asociado al evento “b” es igual a 5, podemos afirmar que el evento “a” ha  ocurrido antes que el evento “b”. | F |
| 43. Suponga que el reloj lógico de Lamport de un nodo es igual a 5. Si dicho nodo recibe un mensaje etiquetado con un reloj lógico de Lamport igual a 3, el nuevo valor de su reloj lógico de Lamport será igual a 4, es decir, uno más que el reloj lógico de  Lamport del mensaje. | F |
| 44. Siendo V(a) = [5, 9, 7, 3, 8] y V(b) = [4, 3, 9, 1, 10] dos relojes vectoriales, podemos  afirmar que los eventos “a” y “b” son concurrentes entre sí. | V |
| 45. Cuando llega un mensaje a un nodo, el reloj vectorial de dicho nodo se actualiza sumando al valor actual el valor del reloj vectorial que se transmite en el mensaje. | F |

Sobre el estado global (Algoritmo Chandy-Lamport) y la elección de líder:

|  |  |
| --- | --- |
| 46. Chandy-Lamport sería válido para una topología en anillo, pero sólo si se garantiza que los canales son fiables y FIFO. | F |
| 47. En el algoritmo de elección de líder con topología en anillo, el líder es aquel nodo  que posee el token. | F |
| 48. Cualquier algoritmo de elección de líder requiere que cada nodo disponga de un  identificador único, que además debe ser conocido por los otros nodos. | V |
| 49. Una instantánea consistente se caracteriza porque todos los nodos registran su estado en el mismo instante de tiempo real. | F |
| 50. En el algoritmo de Chandy-Lamport, cuando un nodo “p” recibe el mensaje MARCA por el canal “c”, si ya había registrado su estado deja de registrar la actividad en dicho canal “c”. | V |
| 51. En el algoritmo Bully, se escoge como líder al primer nodo que envía el mensaje  ELECCIÓN al nodo coordinador. | F |

Sobre los algoritmos de exclusión mutua en sistemas distribuidos:

|  |  |
| --- | --- |
| 52. En el algoritmo de exclusión mutua distribuido, un nodo entra en la sección crítica cuando el resto de nodos le responden con un mensaje OK. | V |
| 53. En el modelo centralizado, el coordinador debe mantener una lista de respuestas pendientes (nodos que han solicitado acceso al recurso, pero a los que todavía no se | V |



Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

|  |  |
| --- | --- |
| les ha contestado). |  |
| 54. 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. | F |

# [FINAL 2016]

Respecto a los algoritmos de sincronización de relojes:

|  |  |  |
| --- | --- | --- |
| 1. | 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 |
| 2. | En el algoritmo de Cristian, si el cliente determina que su reloj está adelantado,  detiene su avance el tiempo necesario para alcanzar la hora correcta | V |
| 3. | En el algoritmo de Berkeley, cada cliente puede sincronizar su reloj en cualquier momento, con independencia de cuándo sincronizan su reloj otros clientes | F |

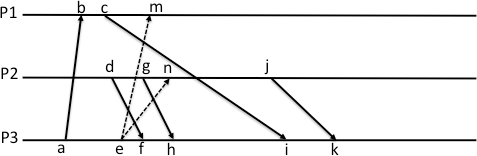
Respecto a los relojes lógicos de Lamport y la relación causal "ocurre-antes":

|  |  |  |
| --- | --- | --- |
| 4. | Decimos que el evento "a" precede al evento "b" (a->b) si todos los nodos están de  acuerdo en que primero ven el evento "a" y después ven el evento "b". | V |
| 5. | Los relojes lógicos de Lamport garantizan que dentro de un mismo nodo los valores L(x) son siempre estrictamente crecientes. | V |
| 6. | Dos eventos "a" y "b" de distintos nodos no pueden tener valores lógicos asociados L(a) y L(b) tales que L(a)=L(b). | F |

Respecto a los relojes lógicos vectoriales:

|  |  |  |
| --- | --- | --- |
| 7. | Asocia un valor vectorial a cada evento (ej. para evento "x" hablamos de V(x)), de forma que si para dos eventos "a" y "b" cualesquiera se cumple V(a)<V(b),  entonces a->b. | V |
| 8. | Para cualquier par de vectores distintos V(a) V(b), se cumple que V(a)<V(b) o V(b)<V(a). | F |
| 9. | En cada nodo necesitamos un vector de valores enteros, con un elemento por evento posible. | F |

Dada la siguiente figura, donde se tienen 3 procesos (P1, P2, P3), cada uno en un nodo distinto. Suponga que P3 inicia el algoritmo de Chandy-Lamport en el evento "e", siendo las líneas discontinuas los mensajes que envía dicho nodo como consecuencia de la ejecución de este algoritmo.



|  |  |  |
| --- | --- | --- |
| 10. | Cuando finalice el algoritmo, en el canal (P2,P3) se habrán registrado 2 mensajes. | V |
| 11. | El nodo P1 solamente necesita enviar un mensaje de MARCA al nodo P2, pues con el nodo P3 no ha establecido ningún tipo de comunicación. | F |
| 12. | El nodo P3 registrará su estado local tras recibir el mensaje de MARCA del nodo P1. | F |

Respecto a los algoritmos de elección de líder y de exclusión mutua en sistemas distribuidos:

|  |  |  |
| --- | --- | --- |
| 13. | El algoritmo Bully garantiza que todos los participantes llegan al mismo resultado  (nodos distintos no pueden llegar a conclusiones distintas sobre la identidad del nuevo líder). | V |
| 14. | En el algoritmo de elección de líder en anillo, cuando un nodo recibe el mensaje y  determina que no puede ser el nuevo líder, se limita a descartar dicho mensaje. | F |
| 15. | En el algoritmo de elección de líder en anillo, el líder es aquel que consigue el token. | F |
| 16. | En el algoritmo de exclusión mutua en anillo, el nodo que dispone del token mantiene una lista de respuestas pendientes (nodos que han solicitado acceso al  recurso, pero a los que todavía no se les ha contestado). | F |
| 17. | En el algoritmo distribuido de exclusión mutua, en caso de empate (varios nodos  solicitan acceso al mismo tiempo), se desempata utilizando un algoritmo de elección de líder. | F |

# [2º PARCIAL 2016]

Respecto a los algoritmos de sincronización de relojes físicos:

|  |  |  |
| --- | --- | --- |
| 18. | El algoritmo de Cristian requiere conocer a priori el retardo medio de los mensajes que intercambian los distintos nodos | F |
| 19. | En el algoritmo de Cristian, si un cliente C pregunta al servidor su hora en el instante 10.000 y recibe como respuesta del servidor el valor 10.006 en el instante  10.008, entonces deberá detener su reloj un total de 2 unidades de tiempo. | F |
| 20. | Suponga que seis nodos utilizan el algoritmo de Berkeley para sincronizar sus relojes. El nodo N1 inicia el algoritmo cuando los valores de los relojes de los seis nodos son respectivamente N1=2002, N2=2008, N3=2005, N4=2010, N5=2000 y N6=2005 y recibe las respuestas de los otros nodos al mismo tiempo. Dados estos  supuestos, el nodo N1 determina que debe adelantar su propio reloj en 3 unidades de tiempo. | V |
| 21. | En el algoritmo de Berkeley, si asumimos un total de N nodos, se requiere el envío de 3\*(N-1) mensajes en total. | V |

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | a | b | c | d | e | f | g |
| P2 | h |  | i | j |  |  |  |
| P3 |  |  |  |  |  |  |  |
| k | l | m |  | n o |  |  | p |

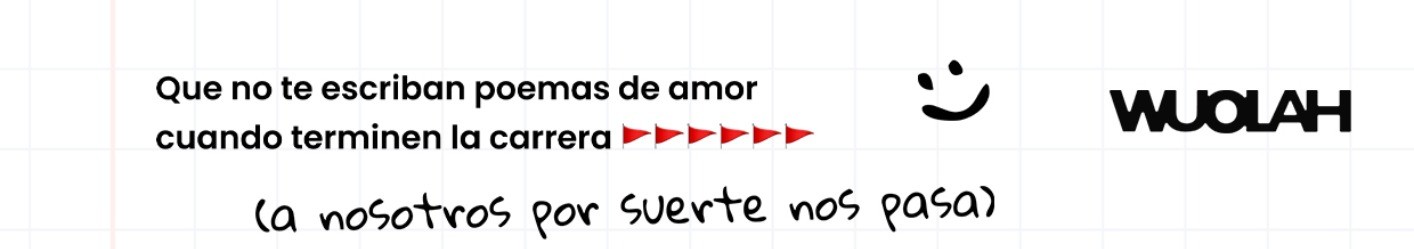
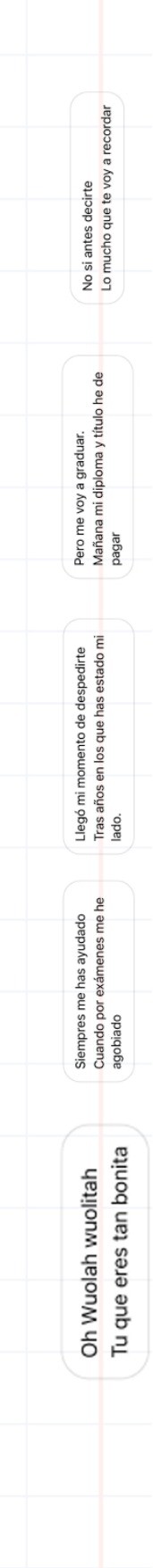
|  |  |  |
| --- | --- | --- |
| 22. | Se determina que los eventos "d " y "j" son concurrentes. | V |
| 23. | h || n | V |
| 24. | El reloj de Lamport de “g” es C(g)=9 y su reloj vectorial es V(g)=[7, 3, 5 ] | V |
| 25. | El reloj vectorial de "m" es V(m)=[2,1,3], el reloj vectorial de "e" es V(e)=[5,3,4] y podemos afirmar que me | F |
| 26. | Se requiere utilizar relojes vectoriales para ordenar de forma total los eventos de este sistema. | F |

Sobre el algoritmo de Chandy-Lamport visto en clase:

|  |  |  |
| --- | --- | --- |
| 27. | El algoritmo 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 |
| 28. | En el algoritmo de Chandy-Lamport los procesos avanzan tomando sus decisiones  en base a información local. | V |
| 29. | Todos los procesos deben esperar a que llegue el mensaje MARCA por todos sus  canales de entrada y posteriormente remitir el estado resultante al iniciador | V |

Sea un sistema con 8 nodos (N1, N2, ... N8) donde en un momento determinado están activos los nodos impares (y esta información aún no es conocida por los nodos).

|  |  |  |
| --- | --- | --- |
| 30. | Para la elección del líder, el algoritmo Bully implica que el nodo N1 envía un mensaje ELECCIÓN únicamente a los siguientes nodos: N3, N5, N7. | F |
| 31. | Si el nodo N5 inicia el algoritmo de elección de líder en anillo, en algún momento recibirá un mensaje de tipo ELECCIÓN con el siguiente contenido:  (5, {5,6,7,8,1,2,3,4}) | F |
| 32. | En el algoritmo de elección de líder en anillo, si un nodo cualquiera recibe un mensaje de tipo ELECCIÓN con el contenido (5, {5, 3}), este mensaje anuncia que el  nuevo líder es el nodo N5. | F |
| 33. | Si se utiliza el algoritmo Bully de elección de líder, el nodo N7 no enviará ningún mensaje de tipo ELECCIÓN. | F |



## [FINAL 2015]

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

|  |  |  |
| --- | --- | --- |
| 34. | 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.* | F |
| 35. | 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.* | F |
| 36. | En el algoritmo de exclusión mutua para anillos no se admite que haya simultáneamente más de un token para una misma sección crítica. | V |
| 37. | 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.* | V |
| 38. | 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.* | F |
| 39. | 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.* | F |

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

|  |  |  |
| --- | --- | --- |
| 40. | El algoritmo de Cristian requiere un nodo con un reloj más exacto que el resto. | V |
| 41. | 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. | F |
| 42. | 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. | V |

Respecto a los relojes lógicos:

|  |  |  |
| --- | --- | --- |
| 43. | 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 | V |
| 44. | 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 | F |
| 45. | 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) | V |



Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

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]:

|  |  |  |
| --- | --- | --- |
| 46. | Podemos afirmar que a || c  *JUSTIFICACIÓN: No se cumple que Va<Vc ni tampoco que Vc<Va, por lo que ambos eventos*  *son concurrentes.* | V |
| 47. | 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* | V |
| 48. | A diferencia de los relojes lógicos de Lamport, no se cumple la transitividad  (si m->n y n->p, no implica m->p). | F |

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

|  |  |  |
| --- | --- | --- |
| 49. | Un corte se considera inconsistente si incluye mensajes enviados y todavía no recibidos.  *JUSTIFICACIÓN: La explicación dada se corresponde con un corte consistente.* | F |
| 50. | Se requiere conectividad total: entre cada par de nodos a,b deben existir canales unidireccionales en sentido a-b y b-a. | V |
| 51. | 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.* | F |
| 52. | Cuando un proceso recibe un mensaje MARCA, debe guardar su estado local sólo si es la primera vez que recibe dicho mensaje. | V |

# [2º PARCIAL 2015]

Respecto a la sincronización en sistemas distribuidos:

|  |  |  |
| --- | --- | --- |
| 53. | En un sistema 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 utilizar un algoritmo de selección de líder para escoger al nodo que puede hacer uso del recurso.  *JUSTIFICACIÓN: Para el acceso en exclusión mutua a un recurso compartido debemos utilizar un algoritmo de exclusión 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 |
| 54. | El algoritmo de Chandy-Lamport permite establecer un orden total entre los eventos de un sistema distribuido.  *JUSTIFICACIÓN: El algoritmo Chandy-Lamport permite obtener el estado global del sistema distribuido. Para establecer un orden total, utilizaremos los relojes lógicos de Lamport y los*  *identificadores de los nodos.* | F |
| 55. | Si los nodos de un sistema distribuido están dispuestos en un anillo lógico, resultaría sencillo obtener el estado global del sistema, pues pueden registrar su estado cuando reciben el token que circula por el anillo.  *JUSTIFICACIÓN: El estado global incluye no sólo 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 para sistemas distribuidos:

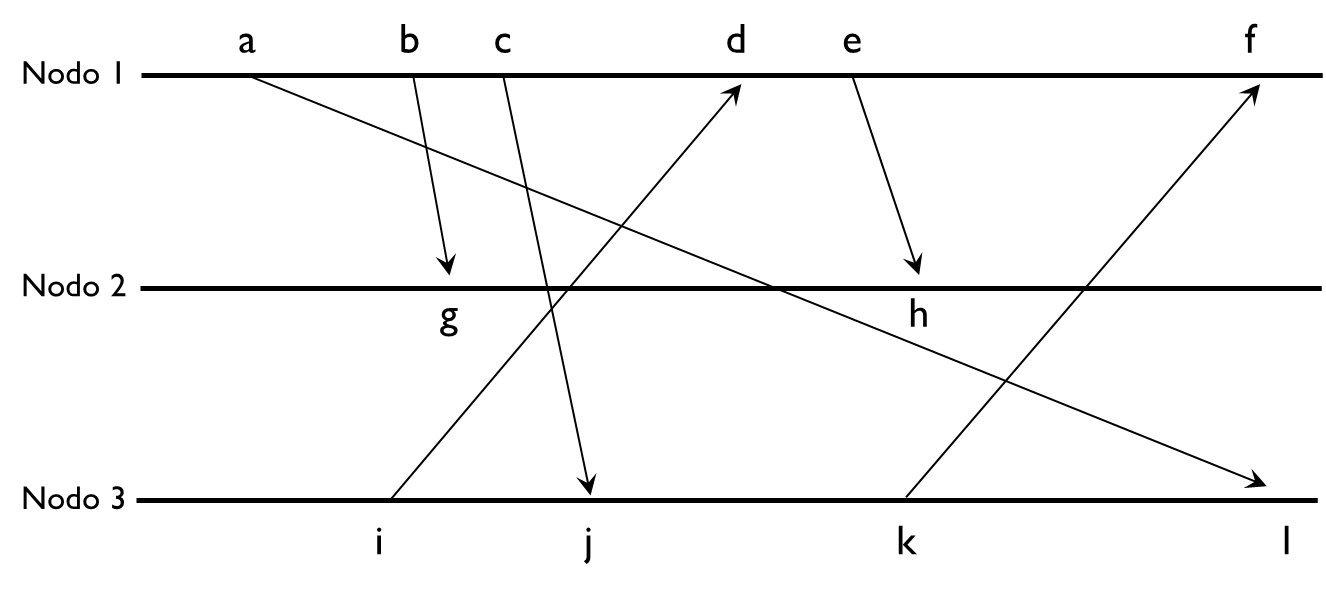
|  |  |  |
| --- | --- | --- |
| 56. | El algoritmo distribuido de exclusión mutua requiere que se utilicen relojes lógicos de Lamport e identificadores para decidir la precedencia de las solicitudes.  *JUSTIFICACIÓN: Se requiere establecer un orden total de los eventos para así resolver los empates que se puedan producir cuando dos o más nodos desean acceder “a la vez” a la sección crítica.* | V |
| 57. | 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 activo con mayor identificador. Por tanto, él es el nuevo líder.* | V |
| 58. | 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 |
| 59. | En el algoritmo centralizado de exclusión 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 último, 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 |
| 60. | En el algoritmo distribuido de exclusión 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.

|  |  |  |
| --- | --- | --- |
| 61. | 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” respecto al cliente.* | F |
| 62. | 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 participan en el algoritmo. | V |
| 63. | El algoritmo de Berkeley asume que el envío 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:

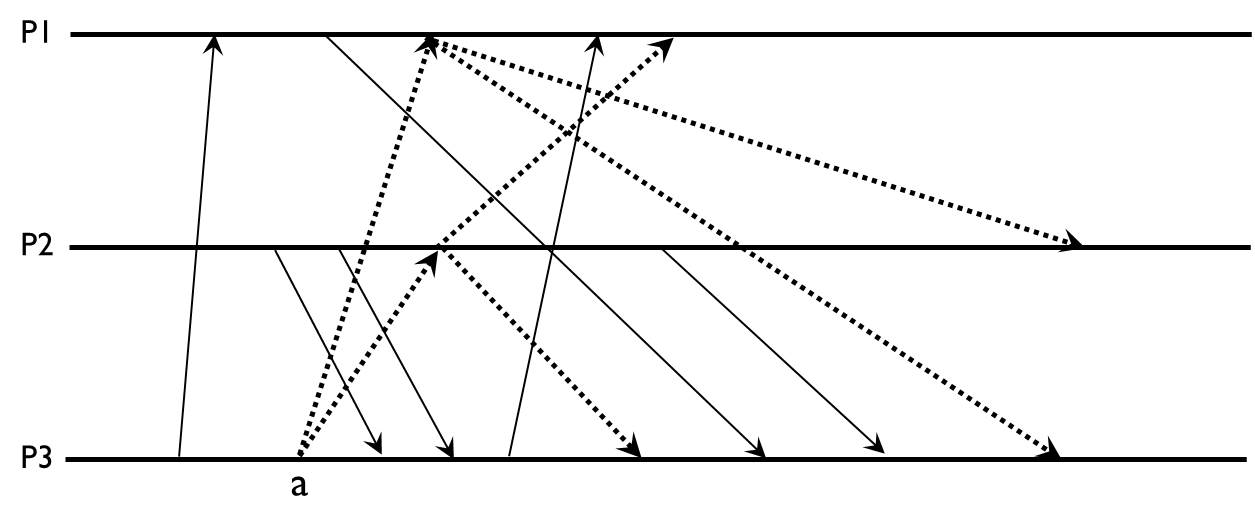
|  |  |  |
| --- | --- | --- |
| 64. | Si el valor en un evento **x** del reloj lógico de Lamport es igual a C(**x**)=1 y el valor en **y** del reloj lógico de Lamport es igual a C(**y**)=6, entonces podemos afirmar que **x** **y**.  *JUSTIFICACIÓN: Dados los valores de los relojes lógicos de Lamport, no se puede afirmar si dos eventos*  *son concurrentes o si uno ocurre antes que el otro.* | F |
| 65. | 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 podemos afirmar que **x || y** | V |



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

Respecto a la figura 1:

|  |  |  |
| --- | --- | --- |
| 66. | Se cumple que **c**  **l** y que **e**|| **l** | V |
| 67. | El valor en **l** del reloj lógico de Lamport es igual a 6 | V |
| 68. | 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 |



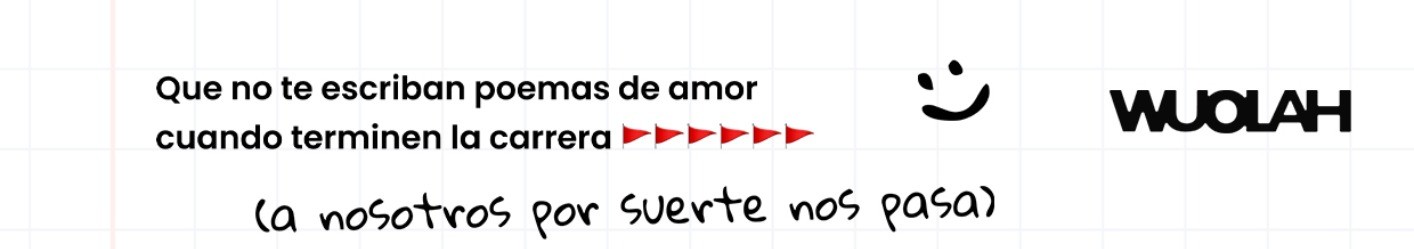
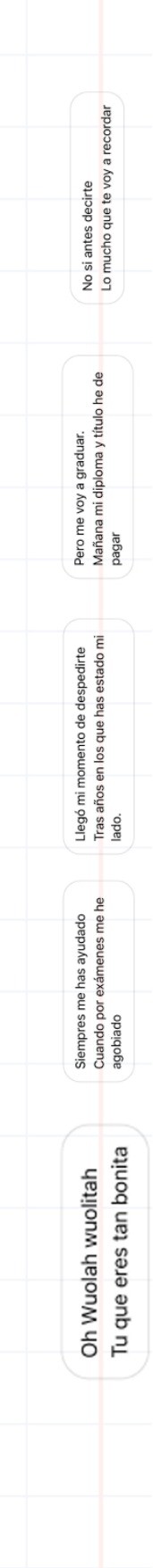
# Figura 2

Respecto a la arquitectura de los sistemas distribuidos:

|  |  |  |
| --- | --- | --- |
| 69. | Las arquitecturas de capas, las arquitecturas basadas en objetos y las arquitecturas basadas en eventos son ejemplos de estilos arquitectónicos de arquitecturas software. | V |
| 70. | 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 |
| 71. | 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,

|  |  |  |
| --- | --- | --- |
| 72. | 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* | V |



|  |  |  |
| --- | --- | --- |
|  | *(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).* |  |
| 73. | en el canal (P2,P3) se habrán registrado 3 mensajes | F |
| 74. | en el canal (P1,P3) se habrá registrado 1 mensaje. | V |

## [FINAL 2014]

Sobre los relojes lógicos de Lamport:

|  |  |
| --- | --- |
| 55. Si dos eventos tienen el mismo valor de reloj, entonces son concurrentes. | V |
| 56. Si dos eventos son concurrentes, entonces tienen el mismo valor de reloj. | F |
| 57. Si un evento ocurre antes que otro, el primero tendrá un valor de reloj menor. | V |
| 58. 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):



P1

a

b

c

P2

e

d

f

g

h

P3

|  |  |
| --- | --- |
| 59. El valor del reloj lógico de Lamport del evento “g” es 5. | V |
| 60. El valor del reloj vectorial del evento “h” es [2,2,3] | V |
| 61. Podemos observar que los eventos “b” y “d” son concurrentes. | V |
| 62. Podemos deducir que el valor del reloj lógico de “c” en ocasiones será 7 y en ocasiones tendrá otro valor. | F |
| 63. Se cumple que f -> c y que e||b | V |

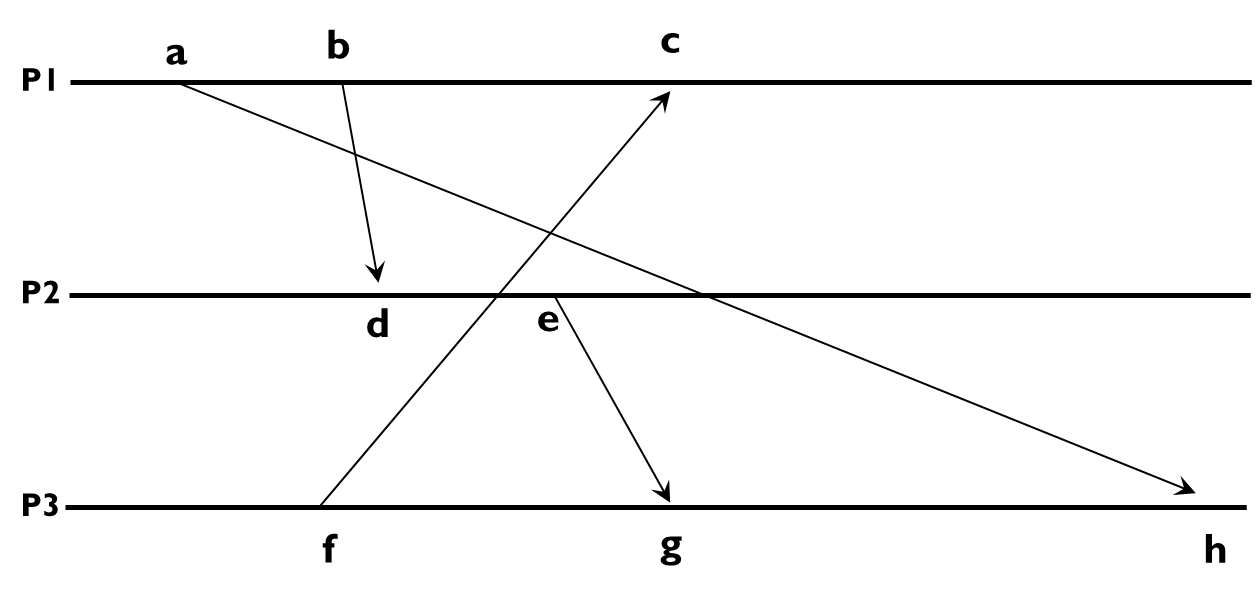


Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Respecto a los algoritmos de sincronización de relojes:

|  |  |
| --- | --- |
| 64. En el algoritmo de Berkeley no se asume que exista un ordenador con un reloj preciso. | V |
| 65. 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 |
| 66. Permiten asignar valores numéricos a los eventos de envío y recepción de mensajes a  partir de la relación “ocurre-antes”. | F |
| 67. En general es problemático hacer que el reloj de un ordenador retroceda. | V |

# [2º PARCIAL 2014]

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

|  |  |
| --- | --- |
| V | A.- Se cumple que f  h |
| V | B.- Se cumple que a  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. |

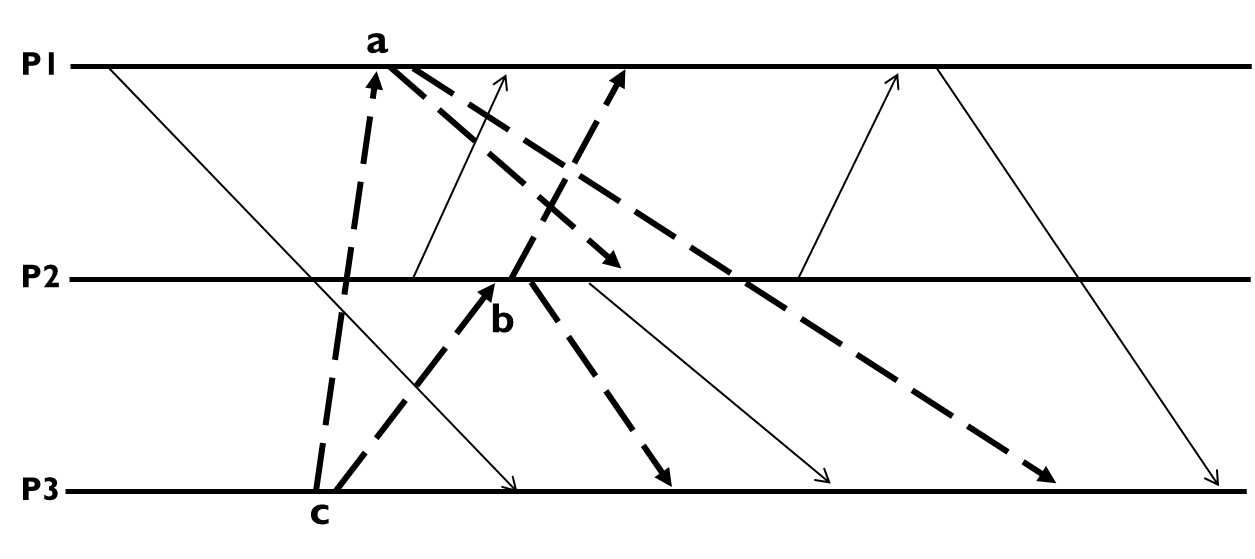
1. Respecto a la figura anterior:

|  |  |
| --- | --- |
| 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] |

1. 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. |

1. 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 |