Tema 4.

1. **Analogias y diferencias entre los algoritmos de sincronización de relojes Cristian y NTP. Razona la respuesta, explica los aspectos clave de cada uno y contextualiza los escenarios de caso de uso. [1,5 puntos]**

Método de Cristian (sincronización externa): con conexión a servidor UTC S. Un proceso p solicita a S una sincronización mediante un mensaje m. S contesta y p pondrá su tiempo a t + Tround/2 (Tround es el tiempo de ida y vuelta). La precisión será de Tround/2 – min.

NTP (protocolo de tiempo de red). Estándar para el establecimiento del tiempo para internet. Se puede estudiar como un tipo árbol, donde los niveles primarios estarían conectados a fuentes UTC, los secundarios a los niveles primarios y la subred de sincronización sería el nivel más bajo, siendo los PC. Es tolerante a fallos. Si un primario pierde la conexión con UTC pasa a secundario, si un secundario pierde a su primario puede escoger otro. Entre pares de servidores NTP se utiliza la sincronización por mensajes en modo UDP.

Hay un modo de sincronización de NTP parecida a los relojes Cristian. Llamada a procedimientos, similar a la de Cristian, pero es más preciso.

Los algoritmos como Cristian o NTP sincronizan los relojes a pesar de sus derivas y el retardo de los mensajes.

2. **Define los siguientes conceptos: [0,5 puntos cada uno]**

Algoritmo criptográfico simétrico: un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor.

Ataque de cumpleaños: se basa en que es más probable encontrar un par idéntico entre un conjunto que la probabilidad de encontrar una pareja para un individuo dado.

Corte inconsistente: es un corte tal que, si para cada evento que contiene, no contiene todos los sucesos que sucedieron antes de él.

Sistema distribuido asíncrono: cuando no están definidos los límites de tiempo máximo y mínimo para ejecutar cada paso de un proceso y la recepción de un mensaje, y no se sabe los límites de deriva de cada reloj local donde se ejecuta cada proceso.

Tiempo UTC: tiempo universal coordinado, es un estándar para la comprobación y sincronización del tiempo, que se difunde por radio en tierra y mediante satélites. Se basa en el tiempo atómico y es calibrado eventualmente con el tiempo astronómico.

Función de resumen seguro: una función hash H es una función computable mediante un algoritmo tal que (h = H(M)) Dado M, debe ser fácil calcular h. Dado h, debe ser muy difícil calcular M. Dado M, deber ser difícil encontrar otro M', tal que H(M) = H(M'). También llamada función de dispersión de un solo sentido.

Reloj de Lamport: no se sincronizan relojes, sino que se ordenan eventos según la relación de orden parcial "suceder antes". (Relojes lógicos) Es un contador software monótono creciente.

Sincronización interior de relojes: dos relojes se envían mensajes para su sincronización.

Tasa de deriva: diferencia por unidad de tiempo que difiere un reloj de un computador del reloj perfecto.

Reloj correcto: se dice de aquel reloj H del cual conocemos su tasa de deriva.

Sistema distribuido asíncrono: un sistema distribuido en general suele ser asíncrono, debido a que es complicado sincronizar todos los relojes. Imposible detectar fallos.

Corte de la ejecución del sistema: transiciones entre estados globales.

Corte consistente es un corte tal que, si para cada evento que contiene, también contiene todos los sucesos que sucedieron antes de él.

3. Modos de Sincronización y algoritmos.

No existe un reloj universal, pero podemos hacer algunas aproximaciones con relojes lógicos, estados globales o relojes vectoriales.

Para la sincronización de relojes, se puede realizar mediante el tiempo universal coordinado (**UTC**). Es un estándar para la comprobación y sincronización del tiempo, que se difunde por radio en tierra y mediante satélites. Se basa en el tiempo atómico y es calibrado eventualmente con el tiempo astronómico. Dos tipos de sincronización:

- **Externa:** un reloj se conecta a una fuente UTC exacta.
- **Interna:** dos relojes se envían mensajes para su sincronización.

Dos relojes que están sincronizados internamente no significan que lo estén externamente, puesto que pueden derivar juntos.

Podemos decir que un SD es síncrono cuando:

- se conoce tiempo máximo y mínimo de ejecución de cada paso en cada proceso.
- Se conoce tiempo máximo y mínimo para la recepción de mensajes.
- Se conocen los tiempos de deriva de los relojes implicados.

Los algoritmos de sincronización serían:

Primera aproximación (sincronización interna): un proceso p le envía un mensaje m a p2 con tiempo t. P2 actualizaría su reloj con tiempo t2 a t + Ttransmisión del mensaje.

Método de Cristian (sincronización externa): con conexión a servidor UTC S. Un proceso p solicita a S una sincronización mediante un mensaje m. S contesta y p pondrá su tiempo a t + Tround/2 (Tround es el tiempo de ida y vuelta). La precisión será de Tround/2 – min.

Algoritmo de Berkeley (sincronización interna): un computador actúa de maestro, y recoge los datos de reloj de todos los computadores del sistema (esclavos). Asumiendo los tiempos de ida y vuelta, realiza el promedio de los tiempos recibidos incluyéndose, enviando el tiempo calculado al resto de computadores. Si el maestro falla podremos elegir otro.

Otra forma de sincronización sería **NTP** (protocolo de tiempo de red). Estándar para el establecimiento del tiempo para internet. Se puede estudiar como un tipo árbol, donde los niveles primarios estarían conectados a fuentes UTC, los secundarios a los niveles primarios y la subred de sincronización sería el nivel más bajo, siendo los PC. Es tolerante a fallos. Si un primario pierde la conexión con UTC pasa a secundario, si un secundario pierde a su primario

puede escoger otro. Entre pares de servidores NTP se utiliza la sincronización por mensajes en modo UDP.

Los métodos de sincronización son:

- **Multicast.** En redes LAN de alta velocidad. Un servidor reparte el tiempo a los demás.
- **Por llamada a procedimiento.** Parecido al método de Cristian pero más preciso.
- **Simétrica.** Para niveles superiores, con alta precisión.

Otros métodos de sincronización serían Lamport (mediante sincronización por eventos y no por relojes) y los relojes lógicos; relojes vectoriales; estados globales, con el algoritmo de Chandy y Lamport.

4. Definir los pasos de Chandy- Lamport, cuando se acaba y finalidad.

El algoritmo de Chandy y Lamport se usa para determinar los estados globales de sistemas distribuidos. Registra un conjunto de estados de procesos de forma que el estado global sea consistente. Se registra el estado de cada proceso localmente.

Los pasos a seguir son:

- **Recepción de marcador para el proceso P.**
  - Si (p no ha registrado su estado). Lo registra. Registra el estado de C como el conjunto vacío. Activa el registro de mensajes.
  - Si (p ha registrado su estado). P registra el estado de C como el conjunto de mensajes recibidos desde que C guardó su estado.
- **Envío de marcador para el proceso P.**
  - Después de que P haya registrado su estado para cada canal de salida C, P envía un mensaje de marcador sobre C.

El algoritmo termina si se cumplen todas las restricciones de conectividad e inexistencia de fallo en la comunicación.

5. Justifica la existencia o no de un reloj universal de referencia. Razona también la respuesta en el contexto de los SD.

No existe un reloj universal de referencia. El tiempo es relativo. En relación con los SD, el tiempo es una de las problemáticas más usuales, ya que no existe dicho reloj. Podríamos realizar algunas aproximaciones que resultarían muy precisas, utilizando relojes lógicos, relojes vectoriales o algoritmos para determinar el estado global de SD en cada momento.

Cada computador que conforma el SD tendría un reloj local, que sería el utilizado por cada uno de los procesos que se ejecutasen en dicho computador. De manera que, aunque nosotros sincronizáramos todos los relojes al mismo tiempo, dicho reloj global variaría significativamente con el tiempo.

Es cierto que hay aproximaciones muy válidas para marcar los eventos en cada computador, como por ejemplo utilizando estándares de sincronización como NTP (protocolo de tiempo de red) o ajustándonos al UTC (coordinación de tiempo universal), relojes con gran precisión y baja tasa de deriva.

Por la diferencia existente entre los relojes de un SD tenemos dos términos:

- **Sesgo**: diferencia de tiempo entre dos relojes en un instante determinado.
- **Tasa de deriva**: diferencia por unidad de tiempo que difiere el reloj de cada computador del reloj perfecto.

6. **Desarrolla el concepto de reloj vectorial y compáralo con el reloj lógico de Lamport, indicando las características principales, reglas de fijación de marcas temporales y álgebras de comparación del orden de sucesión de los eventos en cada caso.**

Los relojes lógicos de Lamport son contadores software monocrecientes. No debemos confundirlos nunca con los relojes físicos. Todos los procesos de un sistema tienen un reloj lógico.

Mattern y Fidge crean los relojes vectoriales para arreglar las deficiencias de los relojes lógicos de Lamport. Un reloj vectorial es un array de N enteros que utilizan los procesos para llevar a cabo su ejecución. Cada uno de los procesos utiliza este reloj para establecer marcas de sus eventos locales.

Tema 5.

1. **Explicar por qué el escenario 2 “el del servidor” no es apto para comercio electrónico.**

En el caso del escenario 2, el servidor Sara es quien recoge todos los datos de credenciales de todos los implicados en la comunicación. Esto es bueno cuando hay un número de participantes concretos, pero no para un tipo de comercio en creciente evolución, como es el comercio electrónico. Debido a las constantes altas y posibles bajas de usuarios en el servidor, sería fácil una posible caída del mismo, o incluso una sobrecarga en las peticiones al mismo, por lo que este modelo sería no apto para este tipo de comercio.

2. **¿El algoritmo TEA se podría llevar a otra arquitectura hardware?**

La operación de desplazamiento depende de la arquitectura, no es lo mismo en un equipo de 32 que 64 bits, existen problemas a la hora de trasladar el algoritmo a otra arquitectura hardware.

3. **Definir entre los algoritmos simétricos y asimétricos criptográficos cuáles son mejor o peor.**

En los algoritmos criptográficos podemos encontrar:

- **Algoritmos simétricos** (con clave secreta). La forma usual de ataque es la fuerza bruta.
- **Algoritmos asimétricos** (con clave pública). Se basa en el uso de funciones de puerta falsa.

Simétricos:

- **TEA:** triple de veloz que el DES. Muy efectivo a pesar de su simplicidad.
- **DES:** su modelo original no era muy veloz. Debido a su carga de código, se implementó en VLSI.
- **Triple-DES:** ejecuta DES tres veces con 2 claves distintas.
- **IDEA:** muy parecido al TEA, pero no tan veloz.
- **AES.**

Asimétricos:

- **RSA:** es el más común y el más utilizado. Para encriptar con RSA el texto es dividido en bloques.
- **Curvas elípticas:** nuevo modelo. Mucho más eficiente. Claves más cortas y más veloz.

Los algoritmos asimétricos son 1000 veces más lentos y no son prácticos para encriptaciones masivas. Sin embargo, sus propiedades los hacen idóneos para distribución de claves y para autenticación.

4. Desarrolla el ataque del cumpleaños, en qué contexto se puede dar y qué aspectos son determinantes para protegernos de él.

El ataque del cumpleaños puede darse en el escenario 4: firma digital con resumen seguro. Se puede dar en el caso de que la función resumen no sea segura.

El ataque del cumpleaños se basa en que es más probable encontrar un par idéntico entre un conjunto que la probabilidad de encontrar una pareja para un individuo dado.

El ataque del cumpleaños se produce cuando.

1. Alice prepara 2 versiones M y M' de un contrato para Bob.
2. Alice fabrica varias versiones sutilmente diferentes de M y M'.
3. Alice envía M a Bob, quien lo firma digitalmente usando su clave privada.
4. Cuando lo devuelve, Alice sustituye M por M', pero manteniendo la firma de Bob sobre M.

Para protegernos de esta paradoja, es necesario tener funciones de resumen seguras y funcionar con firmas digitales conocidas, de forma que sean todavía más seguras. Si encontramos firmas desconocidas en el proceso, es posible que nuestro resumen o mensaje haya sido interceptado.

5. Describe los conceptos de difusión y confusión en criptografía y pon un ejemplo razonado de cada aspecto.

La **difusión** nos dice que, si se cambia un bit en el texto sin cifrar, deberían cambiarse la mayor cantidad posible de bits en el texto cifrado. Para conseguir este efecto se realizan las permutaciones. En cambio, la **confusión** quiere decir que la relación entre el texto cifrado y la clave sea lo más compleja posible. Para este caso las sustituciones cumplen con dicho objetivo.

Los diferentes tipos de algoritmos criptográficos (simétricos, asimétricos o protocolos híbridos).

6. **Explica las tres razones principales por las que proteger una red WIFI y las medidas que debemos aplicar para evitar su ataque o reducir el peligro.**

Las tres razones son: El tráfico de la red puede ser capturado y examinado, los recursos de la red están expuestos a usuarios desconocidos directamente por la vulnerabilidad del canal de transmisión y uso de la red con fines maliciosos.

Medidas para evitar su ataque o reducir el peligro: Cambiar las opciones por defecto del router y webs de configuración, actualizar el firmware y hardware, apagar el AP cuando no se usa, filtrado de MAC y número de clientes simultáneos, bajar al mínimo útil la potencia de transmisión de AP, encriptación WPA o WPA2 con claves largas, encriptar los volúmenes/particiones y ficheros del sistema o incorporar siempre antivirus, firewalls de dos direcciones y software anti-intrusión.

7. **Explica en qué consisten los cifradores de bloques, tipos y para qué se utilizan. Pon un ejemplo de cada uno, funcionamiento y debilidades.**

La mayoría de los cifradores de bloques se basan en bloques de 64 bits. La debilidad de un cifrador simple es que los patrones repetidos pueden ser detectados. La conexión debe ser fiable, no se pueden perder bloques.

Los tipos y sus ejemplos son:

- **Simétricos.** El TEA es el más rápido a pesar de ser muy simple.
- **Asimétricos.** El RSA es el más utilizado actualmente. Para la encriptación necesita dividir el texto en varios bloques.
- **De resumen seguro.** SHA. Basado en MD4 pero más seguro.

Los algoritmos asimétricos con 1000 veces más lentos que los simétricos y no son adecuados para trabajar con gran carga. A pesar de ello, son útiles para la distribución de claves y autenticación.

En los algoritmos de resumen seguro se pueden producir ataques de cumpleaños, por lo que los algoritmos de resumen deben trabajar sobre firmas digitales conocidas y resúmenes seguros.

Tema 6.

1. **Desarrolla el algoritmo de acceso a sección crítica llamado "RicartAgrawala" ¿se cumplen en todos los casos las tres exigencias de exclusión mutua? Razona la respuesta. Además, indica y justifica el número de mensajes que se necesitan en su funcionamiento.**

Dicho algoritmo es un algoritmo basado en relojes lógicos. Se trata de que cuando un proceso quiera entrar en la sección crítica compartida, preguntará a todos los demás si puede hacerlo. Cuando obtenga respuesta de TODOS, podrá entrar. La comunicación se realiza mediante mensajes en forma de tuplas.

El número de mensajes necesario para su funcionamiento depende de las infraestructuras:

- Si no se permite envío multicast se necesitará 2(n-1) mensajes.
- Si se permite envío multicast serán necesarios n mensajes.
- El algoritmo fue refinado para usar n mensajes en envío no multicast.

Se cumplirían con este algoritmo las exigencias EM1 y EM2 (seguridad y vitalidad), pero no aseguraríamos EM3 (ordenación), debido a que este método puede hacer que el servidor sufra caídas. Además, dicho método tendría igual o peor congestión en el servidor que el método de servidor central, y sería más costoso.

2. Describe las 3 exigencias de los algoritmos de exclusión mutua distribuida. Además, razona si se cumplen en anillo y Ricart-Agrawala.

EM1. Seguridad. Sólo un proceso en ejecución en la zona de sección crítica compartida del sistema distribuido.

EM2. Vitalidad. Si un proceso solicita entrar a la sección crítica, se le debe conceder el permiso en algún momento de su ejecución.

EM3. Ordenación. Los procesos deberán entrar en la sección crítica según el orden parcial de "suceder antes".

En el algoritmo de anillo, el testigo pasa de computador en computador, conociendo cada uno de éstos solamente la dirección de su vecino. Se cumplirán la EM1 y EM2, ya que si un computador tiene el testigo será el que entre en SC si lo requiere y en algún momento se otorgará permiso para que un proceso que lo ha solicitado entre en la SC. La EM3 no siempre se cumplirá, debido a que al conocer solamente la dirección del vecino no podemos asegurar que las peticiones se atiendan en el orden correcto.

En el algoritmo de Ricart-Agrawala cumpliremos las mismas reglas, ya que al preguntar a todos los demás nos aseguramos de que no hay nadie ocupando la SC y además siempre entraremos en la misma si lo hemos pedido. El único problema es que no podemos asegurar la EM3, debido a que si se produce fallo en el servidor o cuello de botella podemos perder el orden.

3. Indica y justifica el número de mensajes necesarios para la elección de coordinador mediante el algoritmo "Bully".

El número de mensajes para elegir coordinador:

- En **el caso mejor**: se da cuenta el segundo más alto (n-2) mensajes.
- En **el caso peor**: se da cuenta el más bajo (n^2)

4. Sobre el algoritmo Ricard-Agrawala: (a) Escribe el algoritmo, (b) Indica y explica la complejidad en número de mensajes necesarios; (c) Prueba y razona si siempre se cumplen las tres exigencias de la exclusión mutua en coordinación distribuida. [1,5 puntos]

La idea básica es que cuando un proceso quiere entrar en la SC les pregunta a los demás si puede entrar. Cuando todos los demás le contesten entra.

- **En la inicialización**
  estado := LIBERADA;

- **Para entrar en la sección crítica**
  estado := BUSCADA;
  Multitransmite *petición* a todos los procesos;
  T := marca temporal de la petición;
  Espera hasta que (número de respuestas recibidas = (*N* – 1));
  estado := TOMADA;

  Se aplaza el procesamiento de peticiones

- **Al recibir una petición <T_i, p_i> en el proceso p_j (i ≠ j)**
  si (*estado* = TOMADA o (*estado* = BUSCADA *y* (*T*, p_j) < (T_i, p_i)))
  entonces
  pone en la cola la *petición* por parte de p_i sin responder;
  sino
  responde inmediatamente a p_i;
  fin si

- **Para salir de la sección crítica**
  estado := LIBERADA;
  responde a cualquiera de las peticiones en la cola;

El número de mensajes necesarios para obtener el recurso varía según las características del servidor:

- Sin soporte multicast: 2(n-1).
- Con soporte multicast: n.
- El algoritmo fue refinado hasta n mensajes sin soporte multicast por Raynal en 1988.

Los problemas que presenta el algoritmo son:

- Más costoso que el del servidor central.
- El fallo de cualquier proceso bloquea el sistema.
- Los procesos implicados reciben y procesan cada solicitud: igual o peor congestión que el servidor central.

En resumen, ninguno de los algoritmos vistos puede tratar el problema de caídas. En el algoritmo de servidor es el que tiene menor número de mensajes, pero supone cuello de botella.

En conclusión, es preferible que el servidor que gestiona el recurso implemente también la exclusión mutua.

Otras preguntas:

1. **Dentro de las prácticas no guiadas realizadas este curso, de sockets, RMI y servicios WEB, explica esquemáticamente (con un gráfico y breve explicación del funcionamiento) la comunicación e interacción del controlador con los servicios web y componentes RMI [1 punto] ¿Qué ventajas e inconvenientes relacionados con los conceptos de SD has encontrado en el uso de cada una de las dos propuestas? [0,5 puntos]**