Tema 4 Sistemas Distribuidos: tiempo y estados globales

La noción del tiempo es una problemática en un sistema distribuido, debido a que no existe un reloj global al sistema. Se pueden realizar algunas aproximaciones, con la sincronización de los relojes de los computadores, con relojes lógicos o con algoritmos para conocer los estados globales de un SD cuando ejecuta.

Cada computador de la red tiene su propio reloj interno. Dicho reloj es usado por los procesos locales, de modo que procesos en computadores distintos pueden tener marcas de tiempo distintas. Es importante saber que aunque todos los relojes del sistema distribuido se sincronicen, estos variarán significativamente con el tiempo.

Tasa de deriva de reloj: diferencia por unidad de tiempo en que el reloj del computador difiere del reloj perfecto.
Los relojes de cuarzo ordinarios derivan 1 seg. Cada 11-12 días. Los relojes de alta precisión derivan10^-7 segs.

Un sistema distribuido está distribuido como una colección de procesos. Cada uno de ellos tiene un estado, el cual está formado por todas sus variables u objetos y que puede cambiar en ejecución. Se comunican a través de la red mediante mensajes, pudiendo realizar las acciones de envío, recibimiento y cambio de estado.

Evento: ocurrencia de una acción que lleva a cabo un proceso al ejecutar. Los eventos en un proceso pueden ordenarse en relación a los procesos siguientes y a los que le preceden.

Historia del proceso P: es una serie de sus eventos ordenador por i.
Historia(Pi)=hi=<evento0i,evento1i…>.

RELOJES

Para establecer las marcas temporales se usa el reloj del computador. En un instante t el SO lee el valor del reloj hardware del computador. En general su cálculo no es completamente exacto, pero si funciona lo suficientemente bien puede ser usado como marcador de los eventos de P.

Resolución del reloj < intervalo de tiempo entre dos posibles eventos consecutivos. La resolución de un reloj es el periodo entre dos actualizaciones consecutivas del reloj.

Los relojes no siempre están en perfecto acuerdo. Por ello tenemos los siguientes términos:

Sesgo: diferencia de tiempo entre dos relojes en un instante determinado.
Tasa de deriva: explicada anteriormente.

¿Qué puede influir en la tasa de deriva del reloj?
El tiempo atmosférico debido a fenómenos naturales como tormentas o cambios de presión, así como cambios de magnetismo (por ejemplo un caso en el que la gravedad de la tierra se viera afectada).

TIEMPO UNIVERSAL COORDINADO (UTC)

Los relojes se pueden sincronizar con fuentes externas muy precisas: tiempo atómico internacional basado en un reloj con deriva de un segundo cada 300.000 años.
UTC es un estándar internacional de establecimiento y mantenimiento del tiempo transcurrido. Se basa en el tiempo atómico y ocasionalmente ajustado al tiempo astronómico. La señal se difunde mediante estaciones de radio por tierra y mediante satélites. Las computadoras pueden sincronizar sus relojes mediante receptores adecuados.

Sincronización externa: un reloj C se sincroniza con una fuente UTC exacta S.

$|S(t) - C_i(t)| < D$ para $i = 1, 2, \ldots N$ en un intervalo I de tiempo real

Los relojes C_i son precisos con el límite D.

Sincronización interna: cualquier par de computadores están sincronizados si sus relojes cumplen:

$|C_i(t) - C_j(t)| < D$ para $i,j = 1,2,\ldots N;\ i \neq j$ en un intervalo I de tiempo

Los relojes C_i y C_j concuerdan con el límite D.

Los relojes sincronizados internamente no necesariamente lo están externamente, puesto que pueden derivar juntos.

Corrección del reloj: se dice que un reloj hardware (H) es correcto si su límite de deriva es conocido (tiempo deriva > 0).
Por tanto el error en la medida de dos eventos está limitado para impedir que se produzcan saltos traumáticos en el valor leído. Se puede relajar la condición de error, por ejemplo en make de Unix, alcanzando la monotonicidad en un reloj hardware que funciona rápido, ajustando alfa y beta en C(t) = alfa*H(t) + beta.

Un reloj defectuoso: es aquel que no cumple ninguna de las condiciones de corrección.
Un fallo de ruptura: es cuando el reloj se para o no emite tics.

Un fallo arbitrario: cuando se produce cualquier otro fallo.

IMPORTANTE: un reloj no tiene por qué ser preciso para ser correcto.

Se dice que un SD es síncrono si están definidos los límites siguientes:

Tiempo máximo y mínimo para ejecutar cada paso de un proceso.
Tiempo máximo y mínimo de recepción de un mensaje.
Los límites de deriva de cada reloj local donde se ejecuta cada proceso son conocidos.

Primera aproximación(sincronización interna):

- Un proceso p envía su tiempo local t al proceso p2 en un mensaje m.
- P2 podría poner su reloj a t + Ttransmisión del mensaje.
- Ttransferencia es desconocido, pero min <= Ttrans <= max.
- La incertidumbre u = max-min. Si se establece el reloj a t + (max - min)/2 entonces el sesgo <= u/2.

Método de Cristian (para sincronizar relojes externamente):
Un servidor de tiempo S recibe señales UTC.

- El proceso p solicita el tiempo en un mensaje m y recibe t en Mt de S.
- P establece su tiempo a t + Tround/2 (Tround es el tiempo de ida y vuelta).
- Precisión: más o menos será (Tround/2 - min). El min será el mínimo estimado de transmisión.
    - El momento más temprano en que S pone t en Mt es min después de que P enviara Mr.
    - El momento más tardío es min antes de que Mt llegue a p.
    - El tiempo de S cuando Mt llega está en el rango (t + min, t + Tround - min).

Algoritmo de Berkeley (para sincronizar relojes internamente).

- Un maestro consulta y recoge valores de reloj del resto de computadores, esclavos.
- El maestro utiliza los tiempos de ida y vuelta de los mensajes para estimar el valor de los relojes esclavos.
- Promedia los resultados incluyéndose y eliminando cualquier valor que no sea consistente.
- Envía la magnitud de ajuste de cada reloj, que puede ser positivo o negativo.

Si el maestro falla, se puede elegir a un nuevo maestro.

Protocolo de tiempo de red (NTP)

Se trata de un servicio de tiempo para internet. Sincroniza a los clientes con UTC. Servicio fiable, redundante, reconfigurables si alguno cae, escalables, con autenticación de las fuentes de tiempo.(se puede pensar como una forma de árbol).

- Los servidores primarios están conectados a fuentes UTC.
- Los servidores secundarios sincronizados a los primarios.
- [Subred de sincronización] y en el nivel más bajo de servidores están los PC.

Sincronización de servidores en NTP.

La subred de sincronización se puede reconfigurar si se produce un fallo:

- un primario que pierde su conexión con UTC puede pasar a secundario.
- Un secundario que pierde a su primario puede seleccionar a otro primario.

Los modos de sincronización son:

- Multidifusión (multicast)
  - En LAN de alta velocidad. Un servidor reparte el tiempo al resto que establecen su tiempo asumiendo un retraso de transmisión.
- Llamada a procedimiento.
  - Similar a de Cristian. El servidor acepta peticiones. Precisión más alta.
- Simétrica.
  - Pares de servidores se intercambian mensajes conteniendo información de tiempo.
  - Usado en los casos en que se necesita muy alta precisión (primeros niveles).

Intercambio de mensajes entre pares de servidores NTP.

Todos los modos usan UDP. Cada mensaje lleva marcas de tiempo de los eventos recientes (del evento anterior y del tiempo local de envío del mensaje actual).
El receptor anota el tiempo local cuando recibe T.
El protocolo UDP carece de seguridad, y puede haber retraso entre la llegada de un mensaje y el envío del siguiente, o pérdida de mensajes.

Precisión de NTP.

Para cada par de mensajes entre servidores, NTP estima una compensación entre los dos relojes (deriva) y un retardo d(medida de precisión. Tiempo total de transmisión para los dos mensajes t y t').
Los servidores NTP mantienen pares del tipo <O , d> estimando la fiabilidad de las variaciones y permitiendo cambiar el propio par.

Lamport (1978)

No se sincronizan relojes, sino que se ordenan eventos según la relación de orden parcial "suceder antes".

1. Si los eventos ocurren en Pi (i=1,2,…N) entonces ocurren en el orden observador por Pi, es decir i.
2. Cuando m es enviado entre dos procesos, el envío (m) ocurre antes que recepción(m).
3. La relación es transitiva.

Lamport. Relojes lógicos.

Un reloj lógico es un contador software monótono creciente. No se debe confundir con un reloj físico.
Cada proceso tiene su reloj lógico que se utiliza para fijar las marcas temporales a los eventos.

Relojes vectoriales.

Mattern y Fidge [1989 - 1991] los desarrollan para superar la deficiencia de los relojes lógicos de Lamport: L(e) < L(e') no implica e -> e'.
Un reloj vectorial V en el proceso P es un array de N enteros, que cada proceso utiliza para establecer marcas de sus eventos locales.

ESTADOS GLOBALES

Se examina si una propiedad particular de un SD es cierta cuando éste se ejecuta.
Ejemplos:

- Compactación automática de memoria.
- Detección distribuida de bloqueos indefinidos.
- Detección de terminación distribuida.
- Depuración distribuida.

Ilustran la necesidad de observar el estado del SD globalmente.

Compactación de memoria: antes de eliminar información se deben analizar las referencias existentes y los canales de comunicación.

Bloqueo indefinido: el abrazo mortal clásico, pero entre procesos no ubicados en la misma máquina.

Terminación: dos procesos pueden en un instante ser pasivos, pero no ser susceptibles de ser eliminados. Por ejemplo, P1 a pesar de ser pasivo puede pasar a activo por un mensaje en tránsito.

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

Estado global consistente: un estado que corresponde con un corte consistente.

Caracterización de la ejecución del SD: transiciones entre estados globales.

Linealización o ejecución consistente: ordenación de los sucesos en una historia global que es consistente.

ALGORITMO DE CHANDY Y LAMPORT

Es un algoritmo de instantánea para determinar estados globales de sistemas distribuidos. Registra un conjunto de estados de procesos y canales de los procesos del sistema de forma que el estado global sea consistente.
Se registra el estado de cada proceso localmente, no hay proceso recolector.

Regla de recepción del marcador para el proceso P.

Cuando P recibe un mensaje marcador sobre el canal c:
Si (p no ha registrado todavía su estado):

- registra su estado de proceso ahora.
- Registra el estado de c como el conjunto vacío.
- Activa el registro de los mensajes que llegan sobre otros canales entrantes.

Sino

- p registra el estado de c como el conjnto de mensajes que ha recibido sobre c desde que guardó su estado.

Regla de envío del marcador para el proceso P.

Después de que P haya registrado su estado para cada canal de salida c:

- p envía un mensaje marcador sobre c (antes que envíe otro mensaje sobre c).

En el algoritmo de Chandy y Lamport el estado registrado puede diferir de todos los estados globales por los que ha pasado realmente el sistema.
El algoritmo selecciona un corte de la historia de la ejecución y cuyo estado registrado es consistente.
El algoritmo termina si se cumplen las restricciones de conectividad total e inexistencia de fallo en la comunicación.

Conclusiones.

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

La sincronización de relojes no siempre es suficiente para satisfacer los requisitos de ordenación de dos eventos arbitrarios que sucedan en dos computadores.

La relación “suceder antes” es un orden parcial sobre los eventos que reflejan un flujo de información entre ellos.

Los relojes de Lamport son contadores que se actualizan de acuerdo con la relación “suceder antes” entre eventos.

Los relojes vectoriales mejoran los de Lamport, ya que determinan si dos eventos están ordenados por la relación “suceder antes” o son concurrentes, comparando los vectores de marcas.

Tema 5 Sistemas Distribuidos. Seguridad

Los objetos y principales en la seguridad de un SD son:

- **Objeto o recurso:** buzón de corre, sistema de archivo, parte de una web…
- **Principal:** usuario o proceso que tiene permiso para realizar acciones. La identidad del principal es importante.

El enemigo que ataca a la seguridad de nuestro sistema distribuido realizará:

- **Ataques:** en aplicaciones que manejan transacciones comerciales u otra información cuyo secreto o integridad es crucial.
- **Amenazas:** sobretodo a procesos, a los canales de comunicación, denegación de servicio.

Los canales seguros en nuestro sistema distribuido tendrá:

- Propiedades
    - Cada proceso está seguro de la identidad del otro.
    - Los datos son privados y protegidos contra la manipulación.
    - Protección contra repeticiones y reordenación de datos.
- Utiliza criptografía.
    - El secreto se preserva mediante ocultamiento criptográfico.
    - La autenticación basada en la prueba de posesión de secretos.
    - **Ocultamiento criptográfico:** basado en confusión y difusión.
    - **Posesión de secretos:** claves convencionales compartidas. Pares de claves públicas/privadas.

Amenazas y formas de ataque a nuestra seguridad.

Escuchar a escondidas: obteniendo información privada o secreta.
Enmascararse: asumiendo la identidad de otro usuario/principal.
Manipular mensajes: alterando el contenido de mensajes en tránsito.
Reenviar: almacenando mensajes seguros y enviándolos más tarde.
Negación de servicio: inundando un canal u otro recurso, negando acceso para los otros.

Algunas de las amenazas que superan los canales seguros son:

Ataques de negación de servicio. El uso excesivo de recursos hasta el grado de impedir su uso a usuarios legítimos. Amazon y Yahoo en Febrero de 2000. PSN en Diciembre de 2014.

Los caballos de Troya y otros virus. Los virus sólo pueden entrar en computadoras cuando el código de programa es importado.
Pero los usuarios a menudo requieren programas nuevos.

Las defensas que se presentan a estas amenazas son entre otras: la autenticación de código mediante firmas, la validación de código con comprobación de tipo, seguridad JVM…ANÁLISIS, DISEÑO Y PRUDENCIA.

NOMENCLATURA EN TÉCNICAS DE SEGURIDAD.

Alice	Primer participante
Bob	Segundo participante
Carol	Otro participante en los protocolos a tres o cuatro bandas
Dave	Participante en los protocolos a cuatro bandas
Eve	Fisgón
Mallory	Atacante malevolente
Sara	Un servidor

K_A	Clave secreta de Alice
K_B	Clave secreta de Bob
K_{AB}	Clave secreta compartida por Alice y Bob
K_{Apriv}	Clave privada de Alice (sólo conocida por Alice)
K_{Apub}	Clave pública de Alice (publicada por Alice para la lectura de cualquiera)
$\{M\}_K$	Mensaje M encriptado con la clave K
$[M]_K$	Mensaje M firmado con la clave K

Escenario 1: secreto con clave compartida.

Alice y Bob comparte una clave secreta Kab

1. Alice usa Kab y acuerda una función de encriptación E(Kab, M) para codificar y enviar una serie de mensajes {Mi}Kab.
2. Bob lee los mensajes encriptados usando la correspondiente función D(Kab, M).

Alice y Bob pueden funcionar con Kab mientras estén seguros que Kab no es conocida.

Algunos problemas que nos encontramos son:

- Distribución de clave. ¿Cómo envía Alice una clave compartida a Bob de forma segura?.
- Caducidad de la comunicación. ¿Cómo sabe Bob que el mensaje no es una copia capturada por Mallory y reenviada más tarde?.

Escenario 2: autenticación con servidor.

Un ticket es un mensaje encriptado conteniendo la identidad del principal solicitante y una clave compartida para la sesión.

Bob es un servidor de ficheros; Sara es un servidor de autenticación. Sara comparte Ka con Alice y Kb con Bob.

1. Alice envía un mensaje no encriptado a Sara identificándose y solicitando un ticket para acceder a Bob.
2. Sara responde a Alice con {{Ticket}Kb, Kab}ka. Consistente en un mensaje codificado según Ka con un ticket (para comunicar con Bob para cada fichero) encriptado según Kb y una nueva clave Kab.

3. Alice usa K_a para desencriptar la respuesta.
4. Alice envía a Bob el ticket, su identidad y una respuesta R para acceder al fichero: {Ticket}K_b, Alice, R.
5. El ticket es realmente {K_{ab}, Alice}K_b. Bob usa K_b para desencriptarlo, chequea la identidad y usa K_{ab} para encriptar las respuestas a Alice.

Este proceso es una simplificación del protocolo Needham and Schroeder (y Kerberos). No es válido para comercio electrónico.

Escenario 3: autenticación con clave pública.

Bob genera un par de claves pública/privada <K_{Bpub}, K_{Bpriv}>.

1. Alice obtiene un certificado firmado por una autoridad de confianza que posee la clave pública de Bob, K_{Bpub}.
2. Alice crea una clave compartida K_{AB}, la encripta según K_{Bpub} un algoritmo de clave pública y envía el resultado a Bob.
3. Bob usa K_{Bpriv} para desencriptar K_{AB}.

(Si desean asegurar que el mensaje no ha sido manipulado, Alice puede incluir algún dato aceptado por ambos y Bob chequearlo).

Mallory puede interceptar la solicitud de certificado de clave pública y enviarle su propia clave pública, pudiendo desencriptar el resto de mensajes. La firma digital lo impide.

Escenario 4: firma digital con resumen seguro.

Alice quiere publicar un documento M de forma que cualquiera pueda verificar su procedencia.

1. Alice calcula un resumen de longitud fija del documento Resumen(M).
2. Alice encripta el resumen con su clave privada, lo adjunta a M y hace el resultado (M, {Resumen(M)}K_{Apriv}) público.
3. Bob obtiene el documento firmado, extrae M y computa Resumen(M).
4. Bob usa la clave pública de Alice para desencriptar {Resumen(M)} K_{Apriv} y lo compara con el resumen calculado por él. Si coincide, entonces la firma es válida.

La función de resumen debe ser segura frente al “ataque del cumpleaños”.

Función de resumen seguro h = H(M):

1. Dado M, debe ser fácil calcular h.
2. Dado h, debe ser muy difícil calcular M.
3. Dado M, deber ser difícil encontrar otro M’, tal que H(M) = H(M’).

También llamada función de dispersión de un solo sentido.

Paradoja del cumpleaños.

La probabilidad de encontrar un par idéntico en un conjunto es mucho mayor que la de encontrar la pareja para un individuo dado. Estos son los elementos:

- A = {al menos dos personas celebran su cumpleaños a la vez}.
- A^c = {no hay dos personas que celebren su cumpleaños a la vez}.
- P(A) = 1 – P(A^c).
- P = Casos favorables / Casos posibles.

El ataque del cumpleaños se produce cuando:

1. Alice prepara dos versiones M y M' de un contrato para Bob. M favorable y M' desfavorable.
2. Alice fabrica varias versiones de M y M' sutilmente diferentes (espacios al final de línea,…). Ella compara los valores de dispersión de todos los M con todos los M' buscando un par igual.
3. Alice envía el contrato favorable M a Bob, éste lo firma digitalmente usando su clave privada.
4. Cuando lo devuelve, Alice sustituye M por M', pero manteniendo la firma de Bob sobre M.

Por ejemplo, que para generar colisiones en una función aleatoria perfecta (en funciones hash) de n bits, con una probabilidad del 50% aproximadamente, se requieren solo $2^{n/2}$ intentos.

Certificado: sentencia firmada por un principal que sirve de credencial y/o autenticación. Un certificado necesita:

- Un formato estándar acordado.
- Acuerdo sobre la forma en que se construyen las cadenas de certificados.
- Fechas de expiración, de forma que pueda ser revocado.

ALGORITMOS CRIPTOGRÁFICOS

Mensaje: M, clave: K, funciones criptográficas E,D.

- Simétricos (clave secreta).
  - $E(K, M) = \{M\}_k$ $\quad$ $D(K, E(K, M)) = M$
    La misma clave para E y D.
    M debe ser difícil de computar si se desconoce K.
    La forma usual de ataque es la fuerza bruta. Resistente haciendo K suficientemente grande ~128 bits.
- Asimétricos (clave pública)
  - Claves de encriptación y desencriptación separadas: K_e , K_d.
    $D(K_d * E(K_e, M)) = M$.
    Se basa en el uso de funciones de puerta false, E tiene un alto coste computacional. Las claves son muy grandes > 512 bits.
- Protocolos híbridos (usados en SSL actualmente llamado TLS).
  - Usa criptografía asimétrica para transmitir la clave simétrica que es usada para encriptar la sesión.

CIFRADORES DE BLOQUE, DE CADENA Y DE FLUJO.

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

ALGORITMOS DE ENCRIPTACIÓN SIMÉTRICA

Todos estos algoritmos realizan operaciones de confusión y de difusión sobre bloques de datos binarios.

- **TEA**: desarrollado en la Universidad de Cambridge en 1994. Muy simple pero efectivo. Triple de veloz que el DES.
- **DES**: US Data Encryption Standard (1977). El original no era demasiado fuerte. Debido al coste computacional se implementó VLSI. En 1997 fue derrotado por fuerza bruta por un consorcio de usuarios de internet. En 1998 una máquina de EFF podía resolver claves DES en 3 días.
- **Triple-DES**: aplica DES tres veces con dos claves distintas.
- **IDEA**: International Data Encryption Algorithm (1990). Parecido al TEA.
- **AES**: US Advanced Encryption Standard (1997).

ALGORITMOS DE ENCRIPTACIÓN ASIMÉTRICA

Todos ellos dependen del uso de funciones de puerta falsa, que son funciones de un solo sentido con una salida secreta.

- **RSA**: el primer algoritmo práctico (Rivest, Shamir y Adelman 1978) y el más frecuentemente usado. El tamaño de la clave puede variar. Para encriptar según RSA, el texto se divide en bloques de k bits donde $2k < N$ (el valor numérico de un bloque es siempre menos que N; k entre 512 y 1024).
- **Curvas elípticas:** método reciente, 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.

ALGORITMOS DE RESUMEN SEGURO

Cualquier algoritmo simétrico se puede usar en CBC (cifrador de cadena), donde el último bloque es el resumen H(M).

- MD5: desarrollado por Rivest (1992). Calcula un resumen de 128 bits. Cuatro vueltas con una de cuatro funciones no lineales sobre cada 32 bits de un bloque de 512 bits de texto.
- SHA: (1995) basado en MD4 de Rivest, pero más seguro, produce un resumen de 160 bits.

PROTOCOLO NEEDHAM-SCHROEDER

En los primeros sistemas distribuidos (1974-84) era difícil proteger los servidores. No había mecanismos de autenticación del origen de las peticiones y la criptografía pública no estaba disponible.

Needham y Schroeder desarrollaron un protocolo de autenticación y distribución de claves para uso en red local:

- Supuso un primer ejemplo del cuidado en el diseño de protocolos de seguridad.
- Introdujeron varias ideas de diseño.

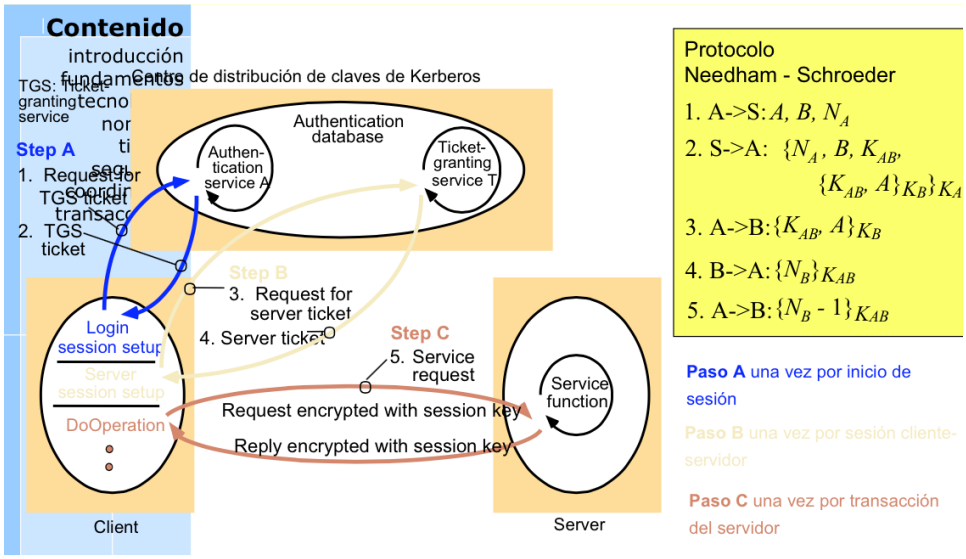
La autenticación de clave secreta sigue el siguiente proceso:

1. A solicita una clave a S para comunicarse con B.
2. S devuelve un mensaje encriptado en la clave secreta de A, con una clave nueva K_{AB} y un "ticket" encriptado en la clave secreta de B. La ocasión N_A demuestra que el mensaje fue enviado en respuesta al anterior. A confía en que S envió el mensaje porque sólo S conoce la clave secreta de A.
3. A envía el ticket a B.
4. B desencripta el ticket y utiliza la nueva clave K_{AB} para encriptar otra ocasión N_B.
5. A demuestra a B que fue el emisor del mensaje anterior devolviendo una transformación acordada sobre N_B.

N_A es una ocasión: entero que se añaden a los mensajes para demostrar la frescura de la transacción. Son generador por el proceso emisor cuando se necesita.

KERBEROS

Comunicación segura con servidores en una red local. Desarrollado para ofrecer seguridad en la red del campus > 5000 usuarios, basado en Needham-Shroeder. Estandarizado e incluido en muchos sistemas operativos. El servidor Kerberos crea una clave secreta compartida para cada servidor solicitado y la envía encriptada al computador del usuario. El password del usuario es el secreto compartido inicial en Kerberos.



GUÍAS DE DISEÑO. ASUNCIÓN DEL PEOR CASO.

Las interfaces están desprotegidas, y por tanto un atacante puede enviar un mensaje a cualquier interfaz.
Las redes son inseguras. Para ello hemos de limitar el tiempo y el alcance del secreto, que se usará una sola vez y tendrá caducidad.
Los algoritmos y códigos están disponibles. Lo mejor es publicar los algoritmos criptográficos empleados.
Los atacantes tienen acceso a suficientes recursos. Hay que presuponer que a lo largo de la vida del sistema se desarrollarán computadores mucho más potentes.
Minimizar la base confiable.

Tema 6. Interacción y cooperación

Los procesos distribuidos necesitan a menudo coordinar sus actividades. Debemos tener en cuenta la tolerancia a fallos o la exclusión mutua entre otros. Para solucionar este último problema, no se pueden utilizar:

- Variables compartidas.
- Facilidades dadas por un único núcleo central.

La solución se basa en el paso de mensajes.

COORDINACIÓN DISTRIBUIDA

- Algunos servidores implementan sus propios métodos para sincronizar los accesos a sus recursos.
- Otros no incluyen sincronización, por ejemplo Sun NFS. Estos servidores requieren de:
  - Un servicio de exclusión mutua.
  - Exclusión mutua distribuida: dar a un único proceso el derecho de acceder temporalmente a los recursos compartidos.
- En otros casos se necesita elegir un proceso de un conjunto para que desarrolle un papel privilegiado (necesario un algoritmo de elección). Por ejemplo en redes Ethernet o inalámbricas.

Requisitos en la exclusión mutua

EM1: Seguridad. En todo momento como máximo en la sección crítica, un proceso ejecutando.

EM2: Vitalidad. A todo proceso que lo solicita, se le da permiso para entrar en sección crítica en algún momento. Se evita el abrazo mortal (deadlock) e inanición (starvation).

EM3: Ordenación. La entrada en la región crítica debe concederse según la relación sucedió – antes.

Algoritmo basado en servidor central

El servidor central concede permisos en forma de testigo que concede acceso a la sección crítica. Al salir de dicha sección, el proceso devuelve el testigo.
Suponiendo que no hay caídas ni pérdidas de mensajes:

- Se cumplen EM1 y EM2.
- EM3 está asegurada en el orden de llegada de los mensajes al servidor.

Se necesitarían 2 mensajes para entrar a la sección crítica y 1 mensaje para salir de ella.

Este algoritmo puede producir cuello de botella en el servidor, caídas en el mismo, lo cual nos llevaría a elegir un nuevo servidor y no aseguramos la EM3 o una caída o fallo del proceso en la sección crítica.

Algoritmo basado en anillo

La exclusión se logra por la obtención de un testigo.
Anillo lógico: se creada dando a cada proceso la dirección de su vecino. El testigo siempre circula por el anillo. Cuando un proceso lo recibe:

- Si quiere entrar a SC lo retiene.
- Si no quiere entrar a SC lo envía a su vecino.
- Al salir de SC lo envía a su vecino.

Con este algoritmo se verifican EM1 y EM2 pero no aseguramos EM3. La obtención del recurso requiere entre 1 y (n-1) mensajes.

Se presentan algunos problemas:

- Carga de la red aunque ningún proceso quiera entrar a SC.
- Si un proceso cae necesita reconfiguración y si además tenía el testigo se debe regenerar el mismo.
- Para asegurarnos de que un proceso ha caído, es necesario varios testigos.
- Desconexión o ruptura de la red.

Algoritmos basados en relojes lógicos

En cada proceso se conocerá la dirección de los demás. Cada proceso posee un reloj lógico.

RICART Y AGRAWALA

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.

El acceso se realiza mediante testigo, de manera que cada proceso guarda el estado en relación a la SC: liberada, buscada o tomada. Los mensajes son tuplas: Tupla<Ti,Pi,SCi>.

- **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;

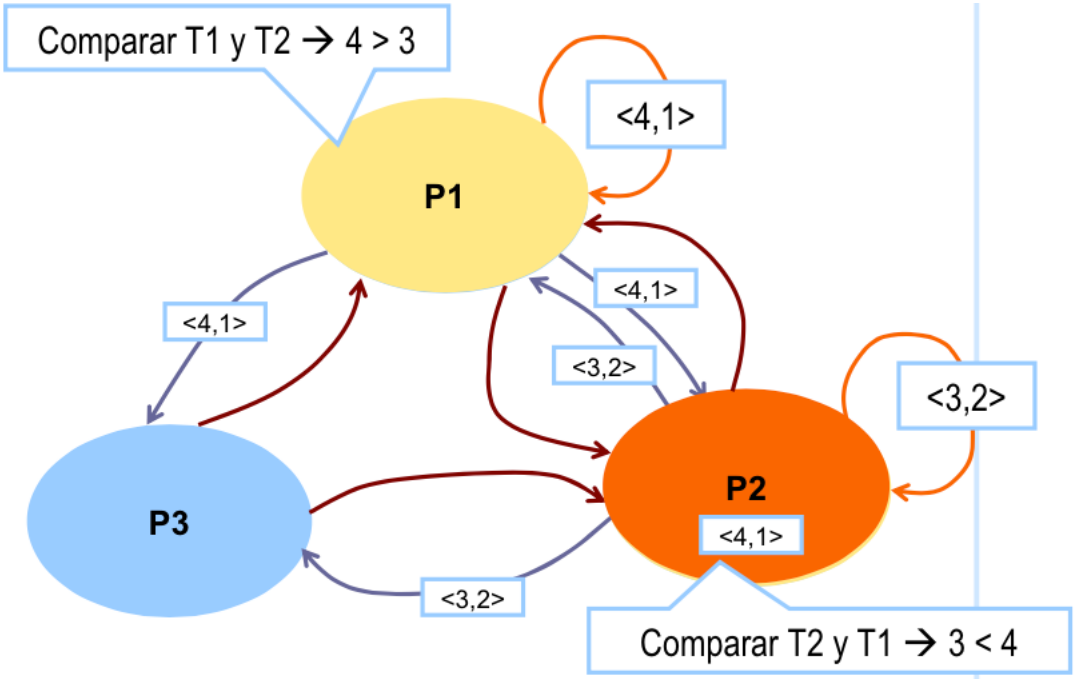
  ⇨ Se aplaza el procesamiento de peticiones

- **Al recibir una petición $<T_i, p_i>$ en el proceso p_j ($i \neq j$)**

  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;



Algoritmo de multidifusión y relojes lógicos: Ricart y Agrawala

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.

ALGORITMOS DE ELECCIÓN

Procedimiento para elegir a un proceso dentro de un grupo, por ejemplo en el caso de que caiga un coordinador o maestro y haya que seleccionar otro.
La principal exigencia es la elección única. Hay dos algoritmos principales:

- Basado en anillo: Chang y Roberts.
- Algoritmo del matón (bully): Silberschatz.

Anillo

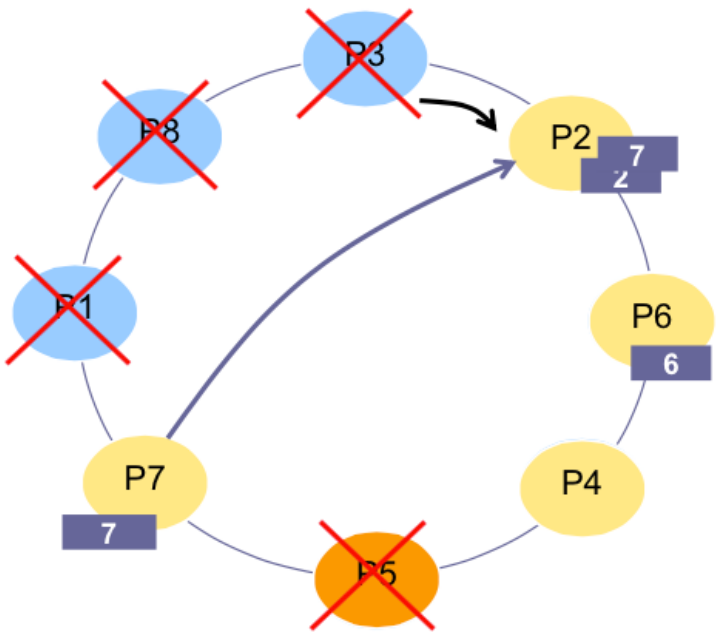
Inicialmente todos los procesos son no-candidatos. Cualquiera de ellos podría empezar una elección, de modo que se marcaría como candidato y se enviaría un mensaje de elección con su identificador.

Cuando un proceso recibe un mensaje de elección:

- Si identificador del mensaje es mayor que el suyo: envía mensaje a sus vecinos.
- Si es menor: si no es candidato se sustituye el identificador y envía mensaje al vecino, marcando como candidato.
- Si es el suyo: se marca como no candidato y se envía mensaje de elegido a su vecino añadiendo su identidad.

Cuando un proceso recibe un mensaje de elegido:

- Se marca como no-candidato.
- Lo envía a su vecino.



Anillo de procesos que transfieren un testigo de exclusión mutua

Anillo lógico: cada proceso sólo sabe comunicarse con su vecino. Se elige al proceso con identificador más alto. Tanenbaum propone un modelo donde los procesos pueden caer.

El número de mensajes para elegir coordinador puede variar:

- en el peor caso: lanza elección sólo el siguiente al futuro coordinador. 3n-1 mensajes.
- En el mejor caso: lanza elección el futuro coordinador. 2n mensajes.

No detecta fallos.

Bully

Para este algoritmo de elección, todos los miembros del grupo deben conocer las identidades y direcciones de los demás miembros. Se supone una comunicación fiable.
El algoritmo selecciona al miembro superviviente con mayor identificador. Los procesos pueden caer durante la elección. Hay 3 tipos de mensajes:

- mensaje de elección: para anunciar una elección.
- Mensaje de respuesta a un mensaje de elección.
- Mensaje de coordinador: anuncia identidad de nuevo coordinador.

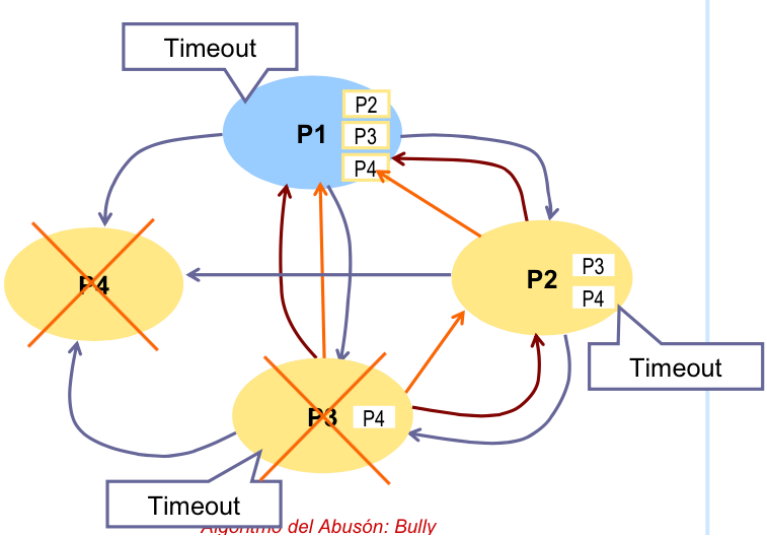
El número de mensajes para elegir coordinador:

- En el caso mejor: se da cuenta el segundo más alto. (n-2) mensajes.
- En el caso peor: se da cuenta el más bajo. $0(n^2)$.

Un proceso inicia una elección al darse cuenta de que el coordinador ha caído:

- Envía un mensaje de elección a los procesos con identificador mayor que el suyo.
- Espera algún mensaje de respuesta. Si el temporizador vence, el proceso actual se erige como coordinador y envía mensaje de coordinador a todos los procesos con identificadores más bajos.
- Si recibe respuesta, espera mensaje de coordinador. Si vence el temporizador, lanza una nueva elección.

Si un proceso recibe un mensaje de coordinador, guarda el identificador y trata a ese proceso como nuevo coordinador.
Si un proceso recibe un mensaje de elección, contesta con un mensaje de respuesta y lanza una elección.
Cuando un proceso se reinicia, lanza una elección a menos que sea el de identificador más alto, en cuyo caso se convertiría en el nuevo coordinador.



Algoritmo del Abusón: Bully

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.

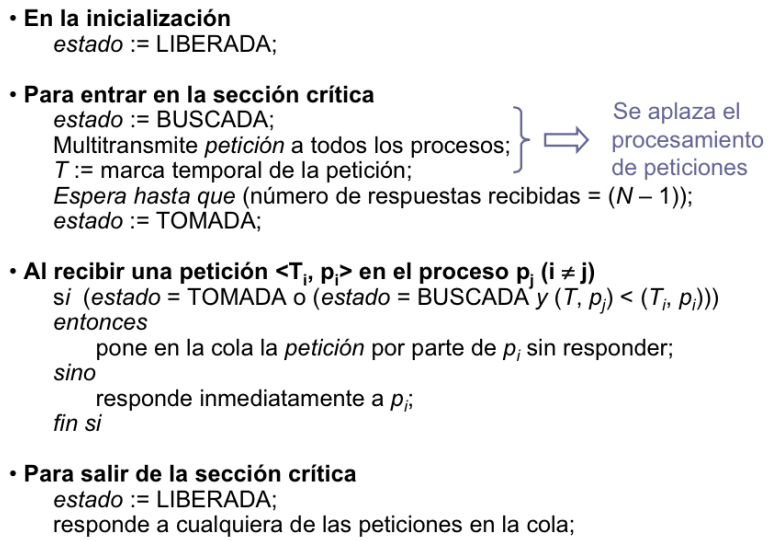
3. **Indica y justifica el número de mensajes necesarios para la elección de coordinador mediante el algoritmo "Bully".**

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.



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]**