SISTEMAS DISTRIBUIDOS RESUMEN

Definiciones:

Algoritmo criptográfico simétrico → Un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y receptor. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usando la clave, lo envía al destinatario y este lo descifra con la misma clave. Es importante destacar que la clave es secreta y que E y D tienen la misma clave. Normalmente para atacar este tipo de sistemas se usa la fuerza bruta, para lo cual se hace la k suficientemente grande.

Algoritmo criptográfico asimétrico → Es un método que utiliza un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves solo se pueda generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves. Se caracteriza por tener una clave publica, por separar las claves de encriptación y desencriptación y basarse en el uso de funciones de puerta falsa. Tienen un coste computacional elevado y las claves son muy grandes.

Ataque de cumpleaños→ Tipo de ataque criptográfico que se basa en la matemática detrás de la paradoja de cumpleaños, haciendo uso de una situación de compromiso espacio-tiempo informática.

Corte inconsciente → Para cada evento que contiene, no contiene todos los sucesos que sucedieron antes que 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 del mensaje, y no se sabe los límites de deriva de cada reloj local donde se ejecuta cada proceso. Es complicado sincronizar todos los relojes. Imposible detectar fallos. Este sistema lo siguen sistemas distribuidos reales como internet. Una solución para este sistema también es valida para un sistema síncrono.

Sistema distribuido síncrono→ El tiempo de ejecución de cada etapa de un proceso tiene ciertos limites inferior y superior conocidos, cada mensaje transmitido sobre un canal se recibe en un tiempo límite conocido, cada uno de los procesos tiene un reloj local cuya tasa de deriva sobre el tiempo de referencia tiene un límite conocido. A nivel teórico, podemos establecer unos limites para tener una idea aproximada de cómo se comportará el sistema, pero a nivel práctico es imposible garantizar esos límites siempre.

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→ Son algoritmos que consiguen crear a partir de una entrada (ya sea un texto, una contraseña o un archivo, por ejemplo) una salida alfanumérica de longitud normalmente fija que representa un resumen de toda la información que se le ha dado (es decir,

a partir de los datos de la entrada crea una cadena que solo puede volverse a crear con esos mismos datos)

Reloj de Lamport→ Tipo de reloj lógico. No se sincronizan relojes, sino que se ordenan eventos según la relación de orden parcial "sucede antes". (Relojes lógicos) Es un contador software monótono creciente.

Sincronización interior de relojes→ Dos relojes se envían mensajes para su sincronización. Es el resultado de obtener en diferentes máquinas las mismas referencias de tiempo para un instante dado. Al alcanzarla se pueden medir eventos como el tiempo de transmisión de un mensaje.

Sincronización exterior de relojes→ Se produce cuando, por ejemplo, si se desea saber en una máquina concreta a que hora del día sucedió un evento, es necesario sincronizar la hora de esa máquina con algún reloj o fuente de hora autorizada.

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.

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 que él.

Corte inconsistente→ Un corte es inconsistente si para todo par de eventos (e, e') no se cumple que (e E C) ^ (e'->e)-> e' E C.

Reloj vectorial → Es un algoritmo para la generación de un ordenamiento parcial de eventos en un sistema distribuido y detecta violaciones causales.

Algoritmo de criptográfico híbrido→ Son usados en SSL, utiliza la criptografía asimétrica para transmitir la clave simétrica que se usa para encriptar la sesión.

Ataque de negación de servicio→ Consiste en el uso excesivo de los recursos hasta el punto de impedir su uso a los usuarios legítimos (Pj: ataque a Amazon y Yahoo! en febrero del 2000)

Caballo de Troya y otros virus → Los virus sólo pueden entrar en los ordenadores cuando el código del programa es importado.

Reloj defectuoso→ Es aquel que no cumple ninguna de las condiciones de corrección.

Fichero replicado→ Es aquel del que existen varias copias, cada una de las cuales están en un servidor diferente.

Needham-Schroeder→ Es un protocolo de autentificación y distribución de claves para redes locales, protege los servidores frente a ataques enmascarados y utiliza autentificación de clave secreta

Kerberos→ Hace lo mismo que el algoritmo Needham-Schroeder ya que está basado en este. Fue desarrollado en el MIT en el 80 y está estandarizado e incluido en muchos SO.

Firma digital→ Es un mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente identificar a la entidad originadora de dicho mensaje y confirmar que el mensaje no ha sido alterado desde que fue firmado por el originador.

Servidor con estado→

IFPS→

oAuth2→Framework de autorización, que permite a las aplicaciones obtener acceso limitado a las cuentas de usuario de servicios como Facebook, Twitter, Google …

Autenticación→ Proceso por el cual se verifica la identidad del usuario y se comprueba que es el que dice ser, evita accesos indeseados de usuarios ilegítimos al sistema.

Preguntas de desarrollo y algoritmos:

Exclusión mutua y algoritmos

Tipo de algoritmos que se utilizan en la programación concurrente para evitar el ingreso a sus secciones críticas por más de un proceso a la vez. La sección critica es el fragmento de código donde puede modificarse un recurso compartido. Es importante destacar que el acceso a dicha sección crítica se regula por medio de semáforos.

Entre sus requisitos encontramos seguridad (solo puede ejecutarse a la vez una sección crítica), pervivencia (a todo proceso que lo solicite se le concede la entrada/salida en la sección en algún momento, esto evita el abrazo mortal y la inanición) y ordenación (Si una petición de entrada ocurrió antes que otra entonces se garantiza la entrada a la sección crítica en ese orden).

Algoritmos:

Algoritmo con servidor central: Un servidor concede los permisos para entrar a la sección critica, dichos permisos se conceden a partir de un token que se envía por mensaje. En cuanto a la tolerancia a fallos destacamos que la caída del servidor es crítica y la caída de los clientes se puede solucionar con temporizadores.

En cuanto al cumplimiento de requisitos destacamos que cumple los requisitos de la exclusión mutua menos el de ordenación ya que no considera los tiempos locales en los que se enviaron los mensajes.

Algoritmo basado en anillo: Paso del token sin servidor central, se da a cada proceso la dirección de su vecino, el token está siempre circulando por el anillo. Cuando uno de los procesos recibe el token se diferencia si quiere entrar o no, en el caso que no quiera se envía a su vecina y en el caso contrario la sección crítica lo retiene. En cuanto la tolerancia a fallos destacamos que si se cae un proceso es necesario reconfigurar el anillo

En cuanto al cumplimiento de requisitos destacamos que cumple los requisitos de la exclusión mutua menos el de ordenación ya que cada proceso recibe cuando se lo pasa su vecina.

Problemas:

Carga de la red incluso si ningún proceso entra en la sección crítica.

Si un proceso se cae necesita reconfiguración

Para asegurarse de que el sistema está caído es necesario generar varios tokens

Una desconexión o ruptura de la red invalida el algoritmo

Algoritmo de Ricart Agrawala: Algoritmo basado en relojes lógicos y multidifusión el cual asegura seguridad, pervivencia y ordenación, es un algoritmo descentralizado que evita cuellos de botella.

Está basado en el algoritmo de Lamport ya que cada proceso conoce la dirección de los demás procesos y cada uno de estos tiene un reloj lógico.

Funcionamiento:

1.Cuando un proceso desea entrar en la sección crítica pregunta a los demás si puede entrar.

2.Cuando todos los demás procesos han contestado entra.

El acceso se realiza mediante un token en el que cada proceso guarda el estado en relación con la sección crítica: liberada, buscada o tomada y cada proceso tiene disponible la cola de solicitudes.

Rendimiento:

Entrada N-1 mensaje de petición y N-1 mensajes de concesión, se releva un mensaje y el ancho de banda es 2N-2.

Entre sus problemas destacamos que el fallo de cualquier proceso bloquea el sistema es decir se produce congestión.

En cuanto a este algoritmo cumple siempre las 3 exigencias de la exclusión mutua. Se cumple la exigencia de seguridad porque el algoritmo como máximo solo hay un proceso ejecutando la sección crítica, la de vitalidad porque todos los procesos se le permite entrada/salida a la sección crítica y ordenación porque todo proceso que lo solicite se le añade en una cola para permitirle una entrada a la sección critica en orden.

Algoritmo de Maekawa: Análogo del algoritmo de Ricart-Agrawala, pero reduciendo el número de mensajes. No se necesita que todos los procesos permitan el acceso, basta con obtener el permiso de un subconjunto K de procesos y siempre que los subconjuntos usados por un par de procesos se solapen con ciertos criterios.

Cumple con el requisito de seguridad, pero no con el de pervivencia pues puede producir interbloqueos a partir de 3 procesos. Solución mejorada por Saunders mediante el uso de la ocurrencia anterior. Se cumple también el requisito de ordenación.

Rendimiento:

Entrada de sqrt(N) mensajes de petición y sqrt(N) mensajes de concesión

Relevo de sqrt(N)

Ancho de banda de 3*sqrt(N)

Sincronización de relojes y algoritmos:

Para poder ejecutar tareas en entornos distribuidos es importante que el orden en las secuencias de las operaciones de procesos sea estricto y universal. Mantener la base del tiempo impone varios problemas tecnológicos importantes como pueden ser la propagación, el cual es el principal problema en sistemas que necesitan grandes cantidades de datos de alta velocidad

La sincronización se define como la forma de forzar un orden parcial o total en cualquier conjunto de eventos y se utiliza para definir 3 problemas distintos relacionados entre sí:

1- La sincronización entre emisor y receptor.
2- La especificación y el control de la actividad común entre procesos cooperativos
3- La serialización de accesos concurrentes a objetos compartidos por múltiples procesos

La sincronización de relojes en un sistema distribuido consiste en garantizar que los procesos se ejecuten de forma cronológica y a la vez respetando el orden de los eventos del sistema. Este problema deriva de la necesidad de comunicar distintas máquinas que trabajan en una misma tarea de forma conjunta, para la cual cada máquina posee un reloj independiente cuya arquitectura por muy precisa que sea hace imposible que se obtengan tiempos iguales.

Un reloj defectuoso es aquel que no cumple ninguna de las condiciones de corrección.

Un fallo de ruptura de reloj es cuando el reloj se para y no emite tics.

Un fallo arbitrario se considera cualquier fallo.

Existen dos tipos de relojes:

-Físicos→ Relacionados con el tiempo real. Existen dos tipos de sincronización, interna (los relojes no se sincronizan con una fuente de tiempo externa) y externa (los relojes se sincronizan con una fuente de tiempo externa fiable).

Algoritmo de Cristian (sincronización interna y relojes físicos):

Consiste en un servidor conectado a una fuente de UTC y unos clientes que se sincronizan con él.

Algoritmo de Berkeley (sincronización interna y relojes físicos): Utilizado para entornos en los que no es posible disponer de fuentes de tiempos UTC, gracias a este algoritmo los relojes del entorno pueden mantenerse sincronizados. Es un algoritmo centralizado en el que encontramos un maestro (solicita su hora periódicamente para obtener la diferencia promedio y que los esclavos actualicen su hora) y varios esclavos.

Network Time Protocolo o NTP (sincronización externa): Protocolo de internet para sincronizar relojes de diferentes sistemas mediante enrutamiento de paquetes en redes con latencia variable. Utiliza UDP como capa de transporte usando el puerto 123 por defecto. Es utilizado para internet a diferencia de Cristian y Berkeley que se usan para intranets, es un servicio confiable que puede sobrevivir a las largas pérdidas de conectividad. Proporciona protección contra interferencias con el servicio de horario.

-Lógicos→ Relacionados con el orden de los eventos y no con el tiempo en el que ocurren.

La idea principal de un reloj lógico consiste en crear un sistema de convergencia del tiempo mediante la medición de derivas. Se pueden usar para ordenar eventos en ausencia de un reloj común estos únicamente muestran una relación de orden parcial.

Lamport: Para la sincronización de relojes lógicos Lamport definió una relación llamada ocurrencia anterior, a -----> b es decir a ocurre antes que b, si a y b son eventos del mismo proceso y a ocurre antes que b entonces se dice a ----> b es verdadero. Si a es el evento que envía el mensaje y b es el proceso que lo recibe esta expresión también es verdadera.

Conceptos importantes:

1- La ocurrencia anterior es una relación transitiva, por lo que si a---->b y b--->c entonces a----->c
2- Los eventos son concurrentes.

El funcionamiento consiste en dos momentos:

1- Envío de evento: El proceso incrementa su reloj y envía su tiempo.
2- Recepción de evento: El proceso debe comparar entre su tiempo actual y el que recibe.

Relojes Vectoriales: Aparecen para solucionar el problema que genera Lamport ya que no siempre se cumple que a ---> b ya que los relojes no captan la causalidad. La solución consiste en que cada proceso tiene un contador o reloj de tiempo vectorial de cada proceso en el sistema.

Funcionamiento:

1- Envío de evento: El proceso debe incrementar su propio reloj y enviar el reloj vectorial.
2- Recepción del evento: El proceso debe comparar los tiempos actuales con los tiempo que recibió y al valor máximo entre ellos dejarlo en su reloj vectorial.

Elección y algoritmos:

Elegir un proceso único para que tome un determinado rol o para reducir una determinada acción. Entre sus aplicaciones destacamos: Elegir un nuevo servidor si se cae el actual, Elegir un nuevo proceso para entrar en una sección crítica, elegir un nuevo proceso menos activo y elegir un proceso con la copia más reciente.

Un proceso convoca elecciones cuando lleva a cabo una acción que indica el algoritmo de elección puede haber N elecciones concurrentes. Un proceso siempre toma uno de estos roles: Participante o No participante.

El proceso de elección debe ser único incluso en elecciones concurrentes todos los procesos tienen un identificador único para el conjunto y totalmente ordenados, el proceso elegido es aquel de mayor identificador.

Algoritmo de anillo: Los procesos se consideran dispuestos en un anillo lógico.

Un proceso convoca elecciones y pasa su identificador al siguiente proceso en el anillo. Si el identificador es mayor que el proceso lo cambia.

Requisitos:

Seguridad: Si, tras la primera vuelta se obtiene el proceso con el identificador más alto

Pervivencia: Si, la última vuelta asegura que todos los procesos activos conocen el resultado de la elección.

Rendimiento:

El ancho de banda es 3N-1 mensajes.

Algoritmo del matón o Bully: Todos los procesos deben conocer las identidades y direcciones del resto de procesos. Se presupone una comunicación fiable. El algoritmo selecciona al proceso superviviente con el identificador mayor. Utiliza timeouts para detectar fallos en los procesos.

3 tipos de mensajes:

1- Elección
2- Respuesta
3- Coordinador

Requisitos:

Pervivencia: Todos fijan ei mediante la multidifusión final.

Seguridad: Tienen problemas de seguridad como la recepción de dos mensajes coordinador con identificadores distintos o que los valores del timeout no sean precisos.

Rendimiento:

Mejor caso: El proceso con el segundo id más alto detecta el fallo de coordinador N-2

Peor caso: El proceso con id más bajo detecta el fallo del coordinador O(N^2)

Algoritmo de invitación: Intenta resolver los problemas de los algoritmos de bully y de anillo, si un proceso detecta la pérdida de líder, entonces se declara líder y forma su propio grupo. Periódicamente el líder de cada grupo busca otros lideres de otros grupos, dos grupos de se unen por medio de mensajes de aceptación.

Consenso distribuido:

En un entorno distribuido los procesos tienen dificultades para ponerse de acuerdo en un valor cuando dos o más procesos han propuesto valores distintos. Entre los protocolos de acuerdo encontramos: Exclusión mutua, Elección y multidifusión.

CRIPTOGRAFÍA:

Se considera que un algoritmo de cifrado es resistente si cumple con la difusión y la confusión.

Difusión: Nos indica que, si 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 permutaciones.
Confusión: 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 dicho objetivo.

Criptografía simétrica: Solo utiliza clave pública parar cifrar y descifrar el mensaje, dicha clave la tienen que conocer tanto emisor como receptor (esto se considera un punto débil del sistema). La intercepción de la clave es más sencilla que un ataque por fuerza bruta. Los algoritmos de clave simétrica son seguros y altamente eficientes si se usan adecuadamente. Son útiles para cifrar grandes cantidades de datos sin tener un efecto negativo en el rendimiento. Un buen cifrado pone toda la seguridad en la clave y no en el algoritmo. Tenemos:

1- Cifrados de flujo: Cifran el mensaje con correspondencia bit a bit sobre el flujo o stream.

   RC4: Este algoritmo cifra un byte a la vez. Una entrada clave es un generador de bits pseudoaleatorio que produce un número de flujo de 8 bits. La salida del generador se llama flujo de claves, se combina un byte a la vez con el cifrado de flujo de texto plano usando la operación XOR.

2- Cifrados de bloque: Cifran el mensaje dividiendo el flujo en bloques de K bits.

   TEA: Un simple pero efectivo algoritmo desarrollado en la universidad de Cambridge. Clave de 128 bits. Bloques de 64 bits.
   DES: No demasiado fuerte en su formato original. Clave de 56 bits. Bloques de 32 bits.
   Triple DES: Aplica DES tres veces con dos claves distintas. Clave de 168 bits. Bloques de 64 bits.
   IDEA: Parecido a TEA. Clave de 128 bits. Bloque de 64 bits.
   AES: Hasta la fecha no se ha encontrado ninguna vulnerabilidad. Clave de 128,160,192,224 o 256 bits. Bloques de 128 bits. El único “ataque” que puede considerarse “efectivo” es el ataque biclicuo, pero no es una amenaza real, este ataque coge bloques de 4 bits y esto reduce la seguridad a 124 bits.

Criptografía asimétrica: Se utilizan dos claves la pública y la clave privada. La clave pública se puede difundir sin problema a quienes quieren mandarte un mensaje cifrado. La clave privada no debe ser revelada nunca.

RSA: El primer algoritmo práctico y el que se utiliza con mayor frecuencia. Su seguridad radica en el problema de factorización de numero enteros y se basa en el producto de dos números primos grandes. Clave entre 512 y 2049 bits.

ECC: Se basa en las matemáticas de las curvas elípticas. Es más rápida y utiliza claves más cortas (256 bits) que otros métodos como el RSA. Clave de 256 bits ofrece la misma seguridad que una clave de 3072 bits.

DSA: Estándar en el gobierno federal de los EEUU para firmas digitales. Sirve para firmar, pero no para autenticar. La desventaja frente a RSA es que requiere mucho más tiempo de cómputo.

Criptografía híbrida: La criptografía simétrica es menos segura que la asimétrica en cambio la asimétrica es más lenta que la simétrica, la criptografía hibrida surge para arreglar los problemas de ambos tipos de criptografía.

El receptor genera una clave pública y otra privada, el emisor cifra el archivo de forma síncrona y el receptor envía su clave pública. Se envía el archivo cifrado síncronamente y la clave del archivo cifrado asíncronamente.

CUANDO UTILIZAR CADA TIPO DE CRIPTOGRAFÍA:

Simétrica: Banca y almacenamiento de datos.

Asimétrica: Firmas digitales, Blockchain e infraestructuras de clave pública.

Híbrida: SSL/TLS y sistemas de chat móvil.

HASH DEFINICIÓN TIPOS Y USOS:

Un hash es una conversión irreversible de los datos u otra fuente de entrada. Es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con longitud fija, independientemente de su longitud la salida tendrá siempre la misma longitud.

Se suele utilizar para asegurar la integridad de los mensajes, por ejemplo, las contraseñas o para la detección de algún tipo de malware.

MD5: Fue desarrollado en 1991

SHA: SHA-3 es el algoritmo mas reciente de la familia SHA. Se basa en la construcción de esponjas, que consiste en una permutación aleatoria de los datos.

Propiedades:

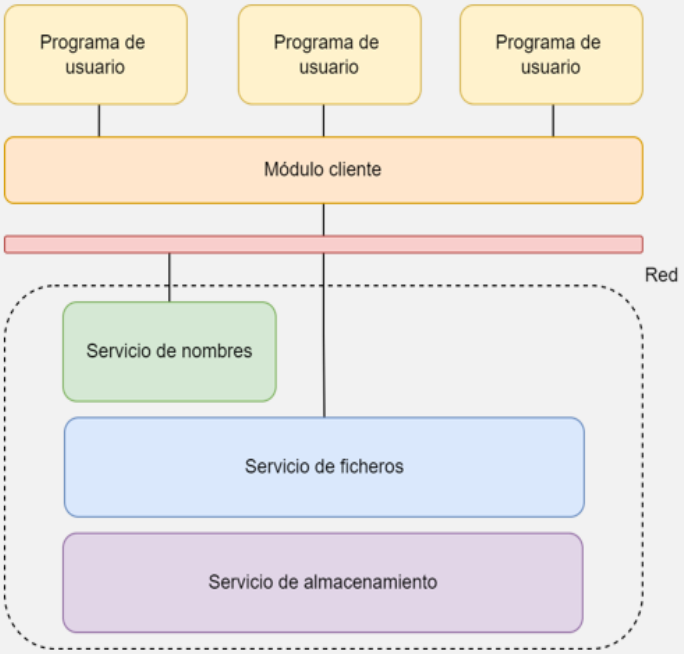
1- Libre de colisiones: Esto hace posible la propiedad determinista. Es muy difícil, prácticamente imposible que dos entradas resulten en la misma salida
2- Ocultamiento: Propiedad que hace que la función hash sea no invertible. Dado un has entrada es imposible conocer un hash salida.
3- Computacionalmente eficiente: El algoritmo ha de ser computacionalmente eficiente, ha de obtener el valor del código hash rápidamente.
4- Pequeños cambios en la entrada cambian completamente el código hash: Al cambiar por ejemplo un solo bit los cambios en el resto del código son enormes. Esto quiere decir que los algoritmos han de ser sensibles a cualquier mínimo cambio.

DFS:

Tiene las mismas funciones que el sistema de ficheros de un SO convencional, pero es más complejo. Los usuarios y los dispositivos de almacenamiento se encuentran dispersos por la red. Permite compartir archivos de una forma controlada y autorizada.

Componente de estructuración de un SFD:

- Servicio de almacenamiento
- Servicio de ficheros
- Servicio de nombres o de directorio
- Módulo cliente: biblioteca de clases o API



Modelo de acceso a ficheros remotos:

Modelo de servicio remoto: Las operaciones se realizan en los servidores, Tienen problemas de eficiencia.

(Servidores sin estado) No almacena información entre solicitudes de un mismo cliente, tiene mayor tolerancia a fallos, no se desprecia memoria en tablas y no se producen problemas si cae en cliente.

(Servidores con estado) Almacena información entre solicitudes de un mismo cliente (mejor rendimiento y permite el bloqueo de ficheros

Modelo de caché de datos: Se accede a los ficheros de forma remota, Aumenta el rendimiento, pero tienen problemas de consistencia.). Su objetivo es retener en memoria principal aquellos daros que han sido usados recientemente.

Existen 4 lugares en los que almacenar la caché: Disco del servidor, En la memoria del servidor, En el disco del cliente y en la memoria del cliente.

Combinación de ambos modelos

Ventajas de la replicación de ficheros:

Aumenta la disponibilidad.

Aumenta la fiabilidad

Mejora el tiempo de respuesta

Reduce el tráfico de la red

Mejora el rendimiento

Beneficio de escalabilidad

Permite trabajar en modo operación desconectada

APLICACIONES:

NFS: Es una arquitectura cliente-servidor que permite al usuario de una máquina ver, almacenar y actualizar archivos de forma remota. El protocolo NFS es uno de los varios estándares de sistemas de archivos distribuidos para el almacenamiento conectado en red (NAS).

SMB: O bloque de mensajes del servidor, es un protocolo para compartir un archivo inventado por IBM. El protocolo SMB se creó para permitir que una máquina pueda realizar operaciones de lectura y escritura en archivos de un host remoto a través de una red de área local (LAN). Se puede acceder a los directorios en el host remoto a través de SMB y se denominan recursos compartidos.

Diferencias entre NFS y SMB

- SMB no distingue entre mayúsculas y minúsculas (*case sensitive*) mientras que NFS sí.
- NFS es generalmente más rápido cuando se lee o escriben archivos pequeños.
- NFS también es más rápido en la navegación entre directorios.
- NFS utiliza un sistema de autenticación basado en host y SMB basado en usuario.
- NFS es más sencillo de configurar que SMB.
- NFS es adecuado para usuarios de Linux mientras que SMB lo es para usuarios de Windows.
- En redes domésticas se recomienda NFS sin cifrar en Linux para máximo rendimiento.

SEGURIDAD:

BUENAS PRÁCTICAS EN SEGURIDAD INFORMÁTICA:

1- **Educación a los usuarios**: Ninguna herramienta puede proteger de los errores cometidos por los propios usuarios, estos deben sensibilizarse sobre lo que les puede ocurrir. Los programas de sensibilización deben tratar: Política de seguridad corporativa, Establecimiento de contraseñas y renovación, Uso y abuso del email, ingeniería social …
2- **Defensa Elástica:** En vez de parar el ataque se busca ralentizarlo al máximo con elementos como: Firewall, VPN, Biometría, Acceso temporizado…
3- **Robustecimiento del sistema:** Eliminar utilidades y programas no esenciales, evitar arranques de SO desde elementos externos. TCP/IP único protocolo instalado. Evitar compartición de ficheros…
4- **Actualizaciones automáticas:** parches y mejoras.
5- **Virtualización:** Se mejoran consumos y mantenimiento, recuperación ante desastres y procedimientos de seguridad.
6- **Uso de herramientas de control (unix**).
7- **Registros externalizados:** Guardando temporalmente copias de seguridad de estos.