Capítulo 9. Replicación, consistencia y tolerancia a fallas

Objetivo: Que el alumno analice y comprenda los diferentes esquemas relacionados a la replicación, consistencia y tolerancia a fallos, así como su impacto en el diseño de sistemas distribuidos.

9.1. Introducción

La replicación juega un importante rol en los sistemas distribuidos y, por lo general, se utiliza para incrementar la confiabilidad y mejorar el rendimiento de un sistema. Sin embargo, uno de los principales retos en los sistemas distribuidos es hacer que estas replicas se mantengan consistentes, lo que implica garantizar que todas las copias del sistema sean actualizadas.

Un reto importante en la consistencia es con respecto a los datos compartidos, que son accedidos por varios procesos simultáneamente, ya que implementar la consistencia crece en complejidad conforme el sistema distribuido es escalado. En este escenario, dos cuestiones relacionados a la consistencia deben ser considerados [Tanenbaum & Van Steen, 2008]. La primera está relacionada con la administración de la réplica, donde se consideran aspectos como la ubicación de los servidores de réplicas y distribución del contenido entre estos servidores. La segunda cuestión a considerar es sobre el mantenimiento de consistencia de las réplicas, lo cual implica que las actualizaciones de todas las réplicas se deben de realizar de una manera rápida (casi inmediata). Por otro lado, también los sistemas distribuidos son diseñados de manera que puedan recuperarse automáticamente de fallas parciales, sin que se afecte seriamente el desempeño total. Esto es una característica sobresaliente de los sistemas distribuidos que los distingue de los sistemas de una sola máquina o centralizados. Una falla se puede

presentar en un sistema distribuido y afectar el funcionamiento de algunos componentes, sin embargo, el sistema puede seguir operando. Esto no es posible en un sistema no distribuido.

Este capítulo revisa los temas relacionados a replicación, consistencia y tolerancia a fallas en los sistemas distribuidos.

9.2 REPLICACIÓN

La replicación significa mantener copias de una información en múltiples computadoras. Este recurso es ampliamente usado en los sistemas distribuidos, debido a que proporciona un mejor rendimiento, alta disponibilidad y tolerancia a fallas. Algunos ejemplos donde se usa la replicación son [Coulouris *et al.*, 2012]:

- Almacenamiento en caché en los servidores web y servidores proxy web.
- El servicio de nombres DNS.

9.2.1 BENEFICIOS DE USAR REPLICACIÓN EN LOS SISTEMAS DISTRIBUIDOS

9.2.1.1 Mejoras del rendimiento

Entre las características con respecto al rendimiento del sistema distribuido que deben de ser observadas al replicar datos se pueden mencionar las siguientes [Coulouris *et al.*, 2012]:

- La replicación se implementa principalmente a través de cachés en clientes o servidores.
- Es importante mantener copias de los resultados obtenidos en llamadas anteriores al servicio para evitar o reducir el coste de llamadas idénticas.
- Se evita el tiempo de latencia derivado del cálculo del resultado o de realizar consultas a otros servidores.
- La replicación de datos con actualizaciones muy frecuentes en la red genera un costo en el uso de protocolos de intercambio y actualización, además de limitar la efectividad de la réplica.

9.2.1.2 Alta disponibilidad

Entre las características con respecto a la disponibilidad del sistema distribuido que deben de ser observadas al replicar datos se pueden mencionar las siguientes [Coulouris *et al.*, 2012]:

- La proporción de tiempo que un servicio está accesible con tiempos de respuesta razonables, que debe de ser cercana al 100%.
- Existen pérdidas de disponibilidad debido a los siguientes factores:
  - Fallas en el servidor. Si se replica *n* veces el servidor, la disponibilidad será $1\text{-}p^n$. Por ejemplo, para $n = 2$ servidores, la disponibilidad es $1 - 0.05^2 = 1 - 0.0025 = 99.75\%$.
  - Particiones de red o desconexiones.
  - Operación desconectada (son aquellas desconexiones de comunicación que a menudo no se planifican y son un efecto secundario de la movilidad del usuario).

9.2.1.3 Tolerancia a fallas

Entre las características respecto a la tolerancia a falla en los sistemas distribuidos, se pueden mencionar las siguientes [Coulouris *et al.*, 2012]:

- Una alta disponibilidad no implica necesariamente corrección, esto se debe a que puede haber datos no actualizados, o inconsistentes, originados por procesos concurrentes.
- Se puede utilizar la replicación para alcanzar una mayor tolerancia a fallas.
  - Ante una caída o suspensión de servidores; por ejemplo, si se tienen *n* servidores, pueden fallar *n-1* sin que el servicio sufra alteración.
  - Ante fallas bizantinas, es decir, si *f* servidores presentan fallas bizantinas, un sistema con *2f+1* servidores proveería un servicio correcto.

9.2.2 Requisitos para realizar la replicación

Entre los requisitos más importantes que se deben considerar al implementar la replicación en un sistema distribuido, están:

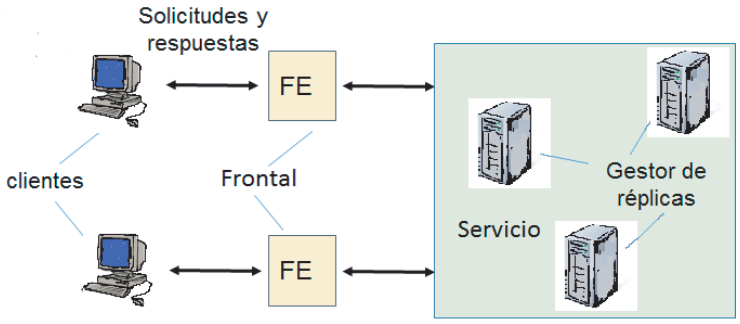
- Transparencia.
- Consistencia.

El requisito de transparencia implica que los clientes no son conscientes de que hay múltiples copias del recurso al que ellos están accediendo. Para los usuarios, solo existen recursos lógicos individuales. Por otro lado, la consistencia implica que las operaciones sobre un conjunto de objetos replicados deben de dar resultados que sigan la especificación de corrección definida para esos objetos. De acuerdo con el tipo de aplicación, la consistencia puede ser más o menos estricta.

9.2.3 Modelo general de gestión de réplica

Esta sección presenta un modelo de sistema general para la gestión de réplicas. Además, describe el papel de los sistemas de comunicación grupal en la consecución de la tolerancia a fallas a través de la replicación. El modelo general de gestión de réplica se muestra en la figura 9.1.

Figura 9.1. Modelo de arquitectura básica para la gestión de los datos replicados



Los componentes principales de esta arquitectura de gestión de datos replicados son:

- Gestor de réplicas.
- Frontal (front-end).

El gestor de réplicas son los componentes que almacenan las réplicas de un determinado objeto o servicio y operan sobre ellas. Frontal (FE) es el componente o interfaz que atiende las llamadas de los clientes y se comunica con los gestores de réplicas.

En general, cinco fases están involucradas en el desempeño de una única solicitud a los objetos replicados [Wiesmann, Pedone, Schiper, Kemme & Alonso, 2000]:

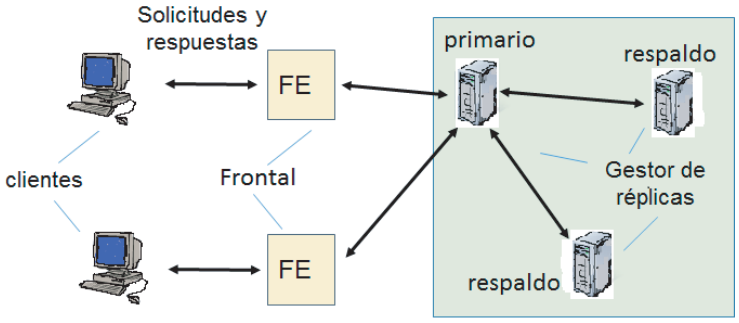
1. La *petición* es enviada por el frontal a uno o más gestores con las siguientes opciones:
   - Se envía la petición a un gestor y este la reenvía a otros.
   - Multidifundir la petición a varios gestores.
2. Los gestores se *coordinan* para ejecutar la petición de manera consistente. Los tipos de ordenación para que los gestores manipulen las réplicas son:
   - Ordenamiento FIFO.
   - Ordenamiento casual.
   - Ordenamiento total.
3. Se *ejecuta* la petición, la cual podría ser realizada de manera tentativa.
4. Se llega a un *acuerdo* o *consenso* antes de consumar la ejecución.
5. Uno o más gestores de réplicas entrega una *respuesta* al frontal.

9.2.4 Servicios de tolerancia a fallas basados en replicación

9.2.4.1 Replicación pasiva

En la replicación pasiva existe un gestor de réplicas primario y uno o más gestores secundarios también conocidos como "respaldos" o "esclavos". Los frontales se comunican con el gestor primario, quien ejecuta las operaciones y manda copias a los respaldos. Si el gestor de réplicas primario falla, entonces un gestor de réplicas de respaldo lo sustituye. La figura 9.2 muestra este escenario.

Figura 9.2. Modelo de replicación pasiva para tolerancia a fallas



Una replicación pasiva tiene las siguientes fases de ejecución [Coulouris *et al.*, 2012]:

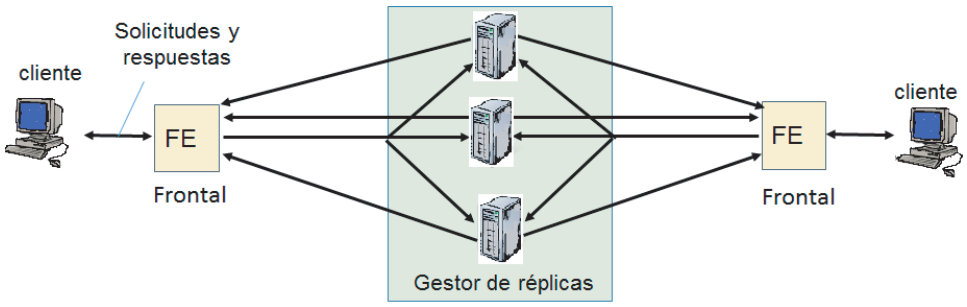
1. Petición.
2. Coordinación.
3. Ejecución.
4. Acuerdo.
5. Respuesta.

Durante la fase de petición, el frontal envía la petición al gestor primario, quien activa la fase de coordinación y ejecuta las peticiones siguiendo una ordenación FIFO. Se realiza la fase de ejecución de la petición y se almacena la respuesta. Si es una petición de actualización, entonces en la fase de acuerdo el gestor primario envía la actualización a todos los respaldos, quienes confirman la recepción. Finalmente, en la fase de respuesta, el gestor primario envía la respuesta correspondiente al frontal. La replicación pasiva tolera fallas de procesos pero no tolera fallas bizantinas. El frontal requiere poca funcionalidad, sin embargo, este esquema de replicación es propenso a problemas de cuellos de botella en el gestor primario.

9.2.4.2. Replicación activa

En la replicación activa todos los gestores de réplicas tienen el mismo rol. Los frontales multidifunden las peticiones a todos los gestores y todos los frontales procesan la petición de manera idéntica pero independiente. En la figura 9.3 se muestra un escenario de replicación activa.

Figura 9.3. Modelo de replicación activa para tolerancia a fallas



Al igual que la replicación pasiva, una replicación activa tiene las siguientes fases de ejecución [Coulouris *et al.*, 2012]:

1. Petición.
2. Coordinación.
3. Ejecución.
4. Acuerdo.
5. Respuesta.

Sin embargo, las actividades que se realizan en cada fase difieren sustancialmente de las actividades que se hacen en la replicación pasiva. Durante la fase de petición, el frontal multidifunde la petición a los gestores usando multidifusión fiable y de ordenación total. No se envía otra petición hasta que se reciba la respuesta a la petición actual. El sistema de comunicación durante la fase de coordinación entrega la petición a todos los gestores según una ordenación total. Entonces, cada gestor realiza la fase para ejecutar la petición. La fase de acuerdo no es necesaria, debido a que se utiliza multidifusión. Finalmente, en la fase de respuesta, cada gestor manda su respuesta al frontal. El número de respuesta que recibe el frontal está en función de las asunciones de falla y del algoritmo de multidifusión.

En la replicación activa, la tolerancia a fallas está soportada principalmente por la multidifusión fiable y totalmente ordenada. La multidifusión es equivalente a un algoritmo de consenso, por lo que la replicación activa permite resolver fallas bizantinas. Esto se debe a que el frontal recoge *f+1* respuestas iguales antes de responder al cliente.

9.3 Consistencia

La consistencia en los sistemas de cómputo es una necesidad vital, ya que sin ella el sistema simplemente no funciona. El problema de la consistencia está presente tanto en los sistemas de cómputo centralizados como en los sistemas distribuidos. El acceso a los datos en un sistema centralizado puede caer en estado inconsistente ante accesos concurrentes, por lo que se requiere prever mecanismos de exclusión que eviten estos escenarios. Sin embargo, en los sistemas distribuidos el problema de la inconsistencia es de mayor dimensión, tanto por su importancia como por la gran cantidad de situaciones en que puede producirse. Debido a que un sistema distribuido es un único sistema, entonces debe tener un único estado global compartido por todas las computadoras que la componen. Este estado global comprende las tablas de mantenimiento del sistema, la hora actual o los datos que están siendo compartidos por distintas computadoras.

9.3.1 Tipos de inconsistencias

En un sistema distribuido, los diferentes tipos de consistencias más frecuentemente estudiadas son:

- Consistencia de actualización.
- Consistencia de replicación.
- Consistencia de caché.
- Consistencia de reloj.

9.3.1.1 Consistencia de actualización

La consistencia de actualización es importante observarla cuando varios procesos acceden concurrentemente a un dato para actualizarlo, ya que la actualización de todo el dato en su conjunto no se realiza como una única operación atómica y se puede caer en una inconsistencia de actualización. Este tipo de inconsistencia es un problema clásico en el acceso a bases de datos o datos compartidos. Sin embargo, en los sistemas distribuidos esto tiene una mayor relevancia debido a que estos sistemas generalmente tienen un gran número de usuarios. Este tipo de inconsistencia se evita usando *transacciones*.

Las transacciones son las primitivas equivalentes a las entradas y salidas de una región crítica usada para proteger la consistencia de un dato compartido. Asimismo, una transacción asegura que las operaciones incluidas en esta se ejecuten todas o no se ejecuta una sola.

9.3.1.2 Consistencia de replicación

Una situación de inconsistencia de replicación puede producirse cuando un conjunto de datos debe replicarse en diversas computadoras de la red, pudiendo ser modificado por cualquiera de estas. En este escenario, es muy probable que se puedan producir situaciones en que los datos no sean iguales en todas las computadoras al mismo tiempo. Los juegos interactivos multiusuarios pueden ser un ejemplo de inconsistencia de replicación, pues las acciones que introduzca un jugador deben de propagarse usando un protocolo de difusión de manera inmediata al resto de los jugadores en la red. En caso de que la red falle o tenga alta latencia, cada jugador tendrá una visión distinta del juego. La consistencia de replicación tiene gran importancia en

los sistemas distribuidos, más aun si estos son en ambientes colaborativos e interactivos.

9.3.1.3 Consistencia de caché

Cuando una aplicación cliente accede a archivo de datos, se pueden guardar estos datos en un espacio de la memoria local del lado del cliente para facilitar el acceso a este dato en futuras referencias. Esto se conoce como memoria caché y reduce la transferencia de datos por la red. La memoria caché tiene como objetivo mejorar los accesos locales mediante el modelo de jerarquías de memoria [Patterson & Hennessy, 1994]. Sin embargo, el problema de inconsistencia se puede presentar cuando el cliente también tiene que actualizar el mismo dato pero este reside en las memorias cachés de otros clientes. En este caso, existe el riesgo de que las copias del dato en las otras memorias cachés queden desactualizadas. Para evitar este escenario, se deben considerar técnicas que aseguren la consistencia de los cachés por parte de los sistemas operativos distribuidos o por las arquitecturas de sistemas multiprocesadores.

9.3.1.4 Consistencia de reloj

Diversas aplicaciones y programación en ambientes distribuidos basan su funcionalidad en algoritmos que muchas veces dependen de estampillas de tiempo. Estas estampillas de tiempo (ver capítulo 5. Sincronización) son usadas para indicar el momento en que un evento ha sucedido. Por ejemplo, algunos algoritmos de reemplazo de página en memoria virtual hacen uso de estampillas de tiempo. Una computadora en un sistema distribuido puede generar una estampilla de tiempo, la cual se puede enviar a cualquier otra computadora del sistema. Así, cada computadora puede comparar su estampilla de tiempo local con la estampilla de tiempo recibida. El problema de inconsistencia de reloj se presenta debido a que no es fácil mantener la misma hora física en todas las computadoras del sistema de manera simultánea.

9.3.2 Modelos de consistencia

Los modelos de consistencia para los datos compartidos son a menudo difíciles de implementar de manera eficiente en los sistemas distribuidos a gran escala [Tanenbaum & Van Steen, 2008]. Además, en muchos casos se pueden

utilizar modelos más simples, que también son a menudo más fáciles de implementar. Un modelo de consistencia es un contrato en los procesos y el almacenamiento de datos. Es decir, si los procesos acuerdan obedecer ciertas reglas, entonces el almacenamiento promete trabajar correctamente. Normalmente, una operación de lectura debe retornar la última actualización del dato. Los modelos de consistencia pueden ser:

- Centrados en los datos.
- Centrados en el cliente.

9.3.2.1 Modelo de consistencia centrado en los datos

Este modelo asume que un almacén de datos (base de datos distribuida o un sistema de archivos) puede estar físicamente distribuido en varias máquinas. Todo proceso que pueda acceder a datos del almacén tiene una copia local disponible de todo el almacén y todas las operaciones de escritura se propagan hacia las otras copias. Ante la ausencia de un reloj global, es difícil sincronizar y determinar cuál es la última operación de escritura. Lo anterior permite que una consistencia centrada en datos presente una gama de modelos de consistencia, entre los que se pueden mencionar los siguientes:

- Modelos de consistencia que no usan variables de sincronización:
  - Estricta.
  - Linealizada.
  - Secuencial.
  - Causal.
  - FIFO.

- Modelos de consistencia con operaciones de sincronización:
  - Débil.
  - Relajada.
  - Entry.

9.3.2.2 Modelo de consistencia centrado en el cliente

Este modelo de consistencia está orientado a un conjunto de datos que tienen un bajo número de actualizaciones simultáneas o, cuando estas ocurren, pueden resolverse fácilmente. Generalmente está orientado a operaciones

de lectura de datos, por lo que se puede catalogar como un modelo de consistencia bastante débil [Tanenbaum & Van Steen, 2008]. Los modelos de consistencias basados en el cliente permiten ver que es posible ocultar muchas inconsistencias de un sistema distribuido de una manera relativamente fácil.

Tanenbaum y Van Steen [2008] citan como ejemplos de modelos de consistencia centrados en el cliente los siguientes:

- Consistencia momentánea.
- Lecturas monotónicas.
- Escrituras monotónicas.
- Lea sus escrituras.
- Las escrituras siguen a las lecturas.

9.4. Tolerancia a fallas

Los sistemas distribuidos se distinguen principalmente de los sistemas centralizados de una sola computadora en su capacidad de falla parcial. La tolerancia a fallas ha sido un tema de investigación por muchos años en las ciencias de la computación. Un sistema tolerante a fallas es un sistema que posee la capacidad interna para asegurar la ejecución correcta y continuada a pesar de las presencia de fallas en hardware o software. El objetivo de este tipo de sistemas es ser altamente fiable.

9.4.1 Origen de una falla

El origen de las fallas en un sistema puede ser clasificado como:

- Fallas en hardware:
  - Son generalmente fallas permanentes o transitorias en los componentes del hardware.
  - Fallas permanentes o transitorias en los subsistemas de comunicación.

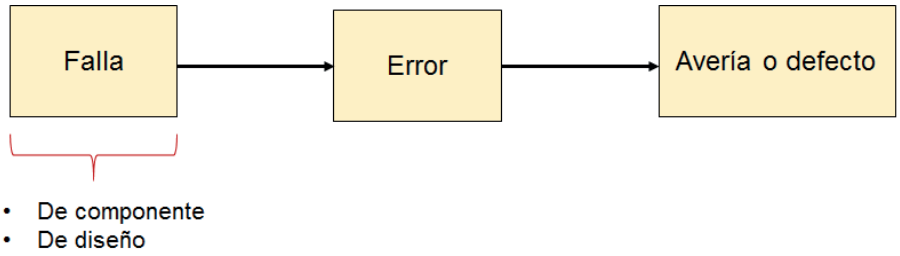
- Fallas en software:
  - Se originan por especificaciones inadecuadas.
  - Fallas introducidas por errores en el diseño y programación de componentes de software.

Se dice que un sistema distribuido es un sistema fiable cuando comprende varios requerimientos útiles, como los siguientes:

- Disponibilidad.
- Confiabilidad.
- Seguridad.
- Mantenimiento.

La *fiabilidad* de un sistema es una medida de su conformidad con una especificación autorizada de su comportamiento [Tanenbaum & Van Steen, 2008]. En contraste, una *avería* o *defecto* es una desviación del comportamiento de un sistema respecto a esta especificación. Las averías se manifiestan en el comportamiento externo del sistema pero son el resultado de *errores* internos. Las fallas pueden ser consecuencia de averías en los componentes del sistema. Las fallas pueden ser pequeñas pero los defectos muy grandes. En la figura 9.4 se muestran las relaciones causa-efecto de una falla.

Figura 9.4. Relación causa-efecto de una falla



9.4.2. Clasificación de fallas

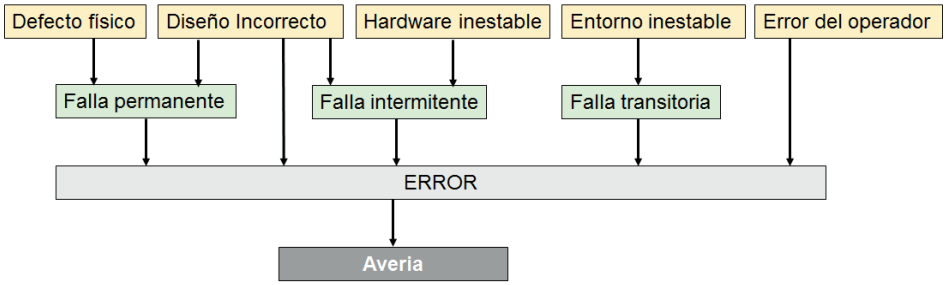
El objetivo de la tolerancia a fallas en un sistema distribuido es evitar que las fallas produzcan averías. Las fallas se clasifican generalmente como (ver figura 9.5):

- Transitorias.
- Intermitentes.
- Permanentes.

Es una falla transitoria, si esta desaparece sola al cabo de cierto tiempo. Ejemplos de este tipo de falla son las interferencias en las señales de comunicación o las fallas transitorias en los enlaces de comunicación. Por otro

lado, las fallas permanentes son aquellas fallas que permanecen hasta que el componente se repare o sustituya, por ejemplo, los errores de software o la rotura de una tarjeta de video. Finalmente, las fallas intermitentes se refieren a aquellas fallas transitorias que ocurren de vez en cuando, por ejemplo, el calentamiento de algún componente de la computadora.

Figura 9.5. Clasificación de fallas



9.4.3 Fallas en los procesos distribuidos

En un sistema distribuido, tanto los procesos como los canales de comunicación pueden fallar, es decir que pueden apartarse de lo que se considera el comportamiento correcto o deseable. En la figura 9.6 se ilustra un modelado de procesos y canales. El modelo de fallas define las formas en las que puede ocurrir una falla, con el fin de proporcionar una comprensión de los efectos de los fallas. Bajo este enfoque, Hadzilacos y Toueg [1994] proporcionaron una taxonomía que distingue entre las fallas de los procesos y canales de comunicación y los presenta como fallas por omisión, fallas arbitrarias y fallas temporales. La tabla 9.1 resume las principales características de este tipos de fallas.

Figura 9.6. Procesos y canales

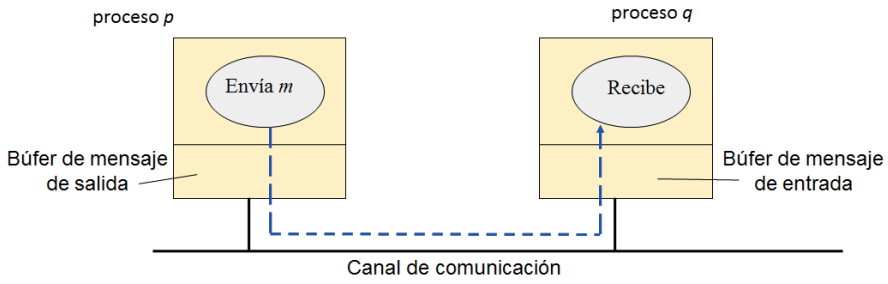


Tabla 9.1. Fallas por omisión y arbitrarias [Coulouris et al, 2012]

Tipo de falla	Efecto	Descripción
Falla de congelación	Proceso	El proceso para y permanece así. Otros procesos pueden detectar este estado
Crash (caída)	Proceso	El proceso para y permanece así. Otros procesos pueden no ser capaces de detectar este estado
Omisión	Canal	Un mensaje insertado en un buffer de mensajes de salida no llega al siguiente buffer de llegada de mensajes
Omisión de envío	Proceso	Un proceso completa un envío pero el mensaje no es puesto en su buffer de mensajes de salida
Omisión de recepción	Proceso	Un mensaje es puesto en el buffer de mensajes de llegada de un proceso pero este no lo recibe
Arbitraria (Bizantino)	Proceso o canal	El proceso/canal muestra un comportamiento arbitrario: podría enviar/transmitir arbitrariamente mensajes en tiempos arbitrarios, comete omisiones; un proceso puede detenerse o tomar un paso incorrecto

Las fallas más serias son las arbitrarias o bizantinas. Cuando estas fallas ocurren, los clientes deben de estar preparados para lo peor. Este problema suele conocerse en la literatura como *el problema de los Generales Bizantinos*, que fue planteado originalmente por Lamport *et al*. [1982].

9.4.4 Redundancia

Todas las técnicas de tolerancia a fallas se basan en el uso de la *redundancia*, de la cual pueden ser identificados los siguientes tipos:

- *Redundancia estática*: Los componentes redundantes se utilizan dentro del sistema para enmascarar los efectos de los componentes con defectos.
- *Redundancia dinámica*: La redundancia se utiliza solo para la detección de errores. La recuperación debe de realizarla otro componente.
- *Redundancia en el diseño*: Partes de un diseño pueden ser redundantes.
- *Redundancia temporal*: Mediante el uso repetido de un componente en presencia de fallas. Adecuado para fallas transitorias.

Ejercicios

1. Define el concepto de tolerancia a fallas en los sistemas distribuidos.

2. ¿Cuál es la importancia del problema de los generales bizantinos para la tolerancia a fallas?

3. Investiga sobre los esquemas de redundancia más usuales para soportar la tolerancia a fallas en los sistemas distribuidos.

4. Cita tres ejemplos de fallas transitorias, intermitentes y permanentes.

5. Explica las ventajas y desventajas de usar una replicación en los sistemas interactivos multiusuarios desplegados globalmente.

6. Explica qué es un modelo de consistencia.

7. Explica la diferencia entre la replicación pasiva y activa.

8. ¿Cuáles son los retos a que se enfrenta la consistencia de caché en los sistemas distribuidos?

9. Explica la relación entre transparencia, consistencia y replicación en los sistemas distribuidos.

10. ¿Qué requerimientos debe de cumplir un sistema distribuido para considerarlo fiable?

Actividad integradora

1. De la aplicación de los contenidos, así como de las habilidades y actitudes desarrolladas.
   Se recomienda exposición general del tema por parte del profesor, así como un análisis de los esquemas de replicación y tolerancia a fallas más usados en los sistemas distribuidos, además de su impacto en tipos de consistencia específicos. Se recomienda la discusión en grupo con alta participación de los alumnos para identificar las posibles soluciones que ofrece cada esquema de replicación y tolerancia a fallas, así como los

retos para implementarlas en la práctica. Este tema se puede cubrir en una semana de tres sesiones.

El alumno debe de mostrar un gran interés por la observación, análisis, abstracción e interpretación de esquemas de flujo de datos, necesidades de replicación y costos de comunicación.

2. Del logro de los objetivos establecidos en el capítulo o apartado del material.

   Se espera alcanzar los objetivos de este capítulo con el apoyo en la solución de los ejercicios, la exposición del profesor sobre el tema, así como un taller de análisis de diagramas sobre soluciones de replicación y tolerancia a fallas.

Capítulo 10. Seguridad

Objetivo: Que el alumno tenga una visión general sobre la importancia de la seguridad en los sistemas distribuidos desde una perspectiva del encriptado y la autenticación de acceso.

10.1 Introducción

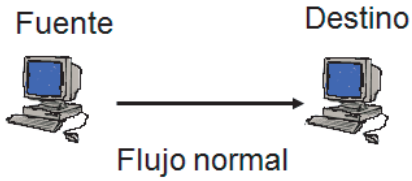
Seguridad se refiere en general a un conjunto de medidas de procedimiento, tanto lógicas como físicas, orientadas a la prevención, detección y corrección de casos de mal uso. Desde un enfoque informático, la seguridad se refiere a las características que debe de tener un sistema de cómputo para resistir ataques y mal uso, ya sea este intencional o no. En los sistemas distribuidos, la seguridad de la información juega un rol muy importante, ya que se debe de garantizar que los recursos de cómputo y la información estén protegidos. Para un enfoque de seguridad informática se recomienda observar los siguientes aspectos [Menezes, Van Oorschot & Vanstone, 1996]:

- *Integridad*: Solo los usuarios autorizados podrán modificar la información.
- *Confidencialidad*: Solo los usuarios autorizados tendrán acceso a los recursos y a la información que utilicen.
- *Disponibilidad*: La información debe de estar disponible cuando se necesite.
- *Vinculación (no repudio)*: El usuario no puede refutar o negar una operación realizada.

10.2 ATAQUES A LA SEGURIDAD

Los ataques a la seguridad en una red de comunicaciones se pueden caracterizar modelando el sistema como un flujo de información desde una fuente (usuario o archivo) a un destino (otro archivo o usuario), tal como se muestra en la figura 10.1 [Stalling, 2004].

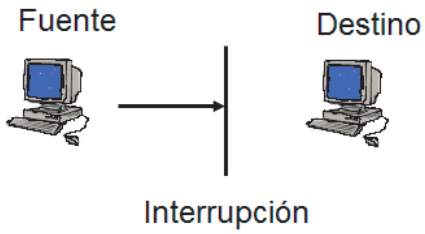
Figura 10.1. Flujo de información normal entre una fuente y un destino



Los cuatro tipos genéricos de ataques a la seguridad son los siguientes [Stalling, 2004]:

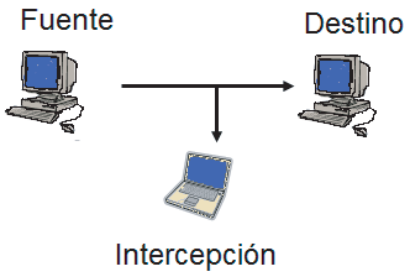
- *Interrupción*: Una parte del sistema resulta destruida o no disponible en un momento dado. Ejemplos de este tipo de ataque pueden ser la destrucción de una parte del hardware o el corte de una línea de comunicación.

Figura 10.2. Ataque del tipo interrupción entre una fuente y un destino



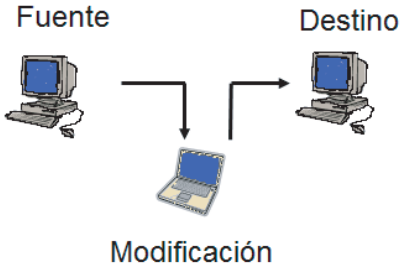
- *Intercepción*: Una entidad no autorizada accede a una parte de la información. Esta entidad no autorizada puede ser una persona, una computadora o un programa. Ejemplos de este tipo de ataques son la escucha del canal, la intercepción vía radio de comunicaciones móviles o la copia ilícita de archivos o programas transmitidos a través de la red.

Figura 10.3. Ataque del tipo intercepción entre una fuente y un destino



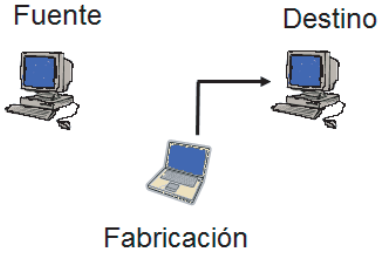
- *Modificación*: Una entidad no autorizada puede acceder a una parte de la información y, además, es capaz de modificar su contenido. Ejemplos de estas modificaciones son la alteración de archivos de datos, alteración de programas y la modificación de mensajes mientras estos son transmitidos por la red.

Figura 10.4. Ataque del tipo modificación entre una fuente y un destino



- *Fabricación*: Una entidad no autorizada envía mensajes y se hace pasar por un usuario legítimo.

Figura 10.5. Ataque del tipo fabricación entre una fuente y un destino



Otra clasificación usada es la división de los ataques en términos de *pasivos* y *activos*. En los *ataques pasivos*, el atacante no altera la comunicación o información, solo escucha o monitorea la red para obtener la información que se está transmitiendo. Entre los objetivos de un ataque pasivo se encuentra la intercepción de datos y el análisis de tráfico, que puede ser usada para obtener información de la comunicación. Los ataques pasivos son muy difíciles de detectar, ya que no provocan alteración en los datos. La información se puede proteger de este tipo de ataque usando encriptado y otros mecanismos de protección.

En los *ataques activos*, el atacante altera las comunicaciones. Este tipo de ataque puede subdividirse en las siguientes categorías:

- Suplantación de identidad.
- Reactuación.
- Modificación de mensajes.
- Degradación fraudulenta del servicio.

Durante la suplantación de identidad, el intruso simula ser una entidad diferente, mientras que en la reactuación uno o varios mensajes legítimos son capturados y repetidos para producir un efecto no autorizado. En el ataque de modificación de mensajes, el intruso varía los datos transmitidos. Finalmente, en un ataque de degradación fraudulenta del servicio, el intruso intenta impedir que las entidades dialogantes puedan funcionar de manera correcta.

10.3 SERVICIOS DE SEGURIDAD

Para afrontar las amenazas a la seguridad del sistema de una organización, se requiere definir una serie de servicios. Estos servicios hacen uso de uno o más mecanismos de seguridad. Una clasificación de los servicios de seguridad es la siguiente [Stalling, 2004]:

- *Confidencialidad*: Garantiza que la información es accesible únicamente por las entidades o personas autorizadas.
- *Autenticación*: Ofrece un mecanismo que permite una identificación correcta del origen del mensaje, asegurando que la entidad no es falsa.
- *Integridad*: Garantiza la corrección y completitud de la información. Es decir que la información solo pueda ser modificada por las entidades

autorizadas. Una modificación puede consistir en escritura, cambio, borrado, creación y reactuación de los mensajes transmitidos.
- *Vinculación*: Vincula un documento a una transacción, a una persona o a un sistema de gestión de información, de tal manera que ni el emisor ni el receptor puedan negar la transmisión.
- *Control de acceso*: Requiere que el acceso a la información sea controlado por el sistema destino.

10.4 MECANISMOS DE SEGURIDAD

No existe un único mecanismo que provea todos los servicios de seguridad antes descritos. Sin embargo, la mayoría de los mecanismos de seguridad usan técnicas criptográficas basadas en el cifrado de los datos. Las técnicas criptográficas más importantes son [Stalling, 2004]:

- *Intercambio de autenticación*: Se encarga de comprobar que la entidad, ya sea origen o destino de la información, sea la indicada.
- *Encriptado*: Altera la representación de los mensajes de manera que garantiza que el mensaje es ininteligible a receptores no autorizados.
- *Integridad de datos*: Garantiza la corrección y completitud de la información. Cifra una cadena comprimida de datos a transmitir y envía este mensaje al receptor junto con los datos ordinarios. El receptor repite la compresión y el cifrado posterior de los datos y compara el resultado obtenido con el recibido para verificar que los datos no han sido modificados.
- *Firma digital*: Cifra una cadena comprimida de datos que se va a transferir usando la clave secreta del emisor. La firma digital se envía al receptor junto con los datos ordinarios. Este mensaje se procesa en el receptor para verificar su integridad.
- *Control de acceso*: Permite que solo aquellos usuarios autorizados accedan a los recursos del sistema o a la red.
- *Tráfico de relleno*: Es el envío de tráfico falso junto con los datos válidos para que el enemigo no sepa si se está enviando información ni la cantidad de datos útiles que se están transfiriendo.
- *Control del enrutamiento*: Permite enviar cierta información por determinadas rutas clasificadas. Asimismo, posibilita la solicitud de otras rutas en caso de que se detecten persistentes violaciones de integridad en una ruta determinada.

10.5 SISTEMAS CRIPTOGRÁFICOS

La criptografía tiene una larga y fascinante historia, desde su uso inicial y limitada por los egipcios hace unos 4 mil años, hasta el siglo XX, cuando jugó un rol crucial en las dos guerras mundiales. La criptografía a través de la historia ha sido usada como una herramienta para proteger los secretos y estrategias nacionales [Menezes *et al.*, 1997]. A través de los siglos, se ha creado y elaborado una gran diversidad de protocolos y mecanismos para hacer frente a los problemas de seguridad de la información. Sin embargo, la proliferación de computadoras y sistemas de comunicación en la década de 1960 trajo consigo una demanda por parte del sector privado de los medios necesarios para proteger la información digital. Lo anterior ha incrementado la demanda por servicios de criptografía. La *criptografía* es el estudio de técnicas matemáticas relacionadas con los aspectos de seguridad de la información, tales como la confidencialidad, integridad de datos, autenticación de la entidad y autenticación del origen de datos [Menezes *et al.*, 1997]. Los sistemas criptográficos son muy útiles para la seguridad en los sistemas distribuidos, ya que permiten mitigar muchas de las posibles vulnerabilidades que estos presentan. Los sistemas de encriptado más usados son los siguientes:

- Encriptado simétrico.
- Encriptado asimétrico.

En el método de encriptado simétrico, tanto el emisor como el receptor usan una misma clave secreta para cifrar como descifrar. Por otro lado, en el encriptado asimétrico cada usuario tiene una pareja de claves, una pública y otra privada, con la característica de que aquello que se cifra con una de las claves solo se puede descifrar con la otra clave. La figura 10.6 muestra el modelo de un encriptado simétrico, mientras que la figura 10.7 muestra el modelo para un encriptado asimétrico o de llave pública.

Figura 10.6. Modelo simplificado de encriptado simétrico [Stalling, 2004]

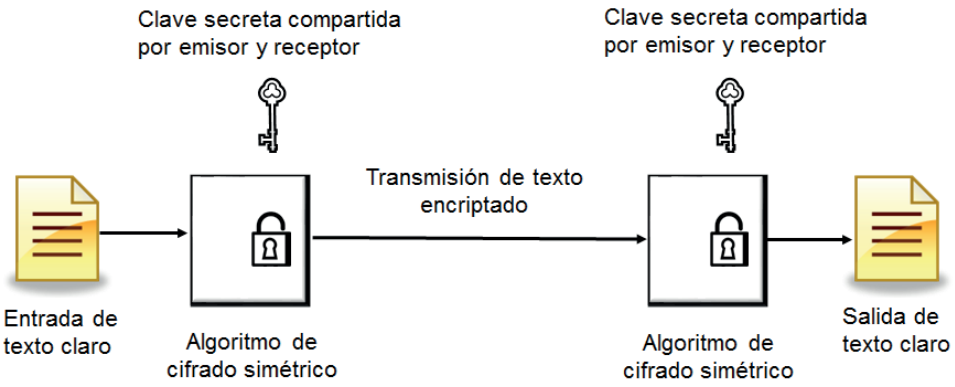
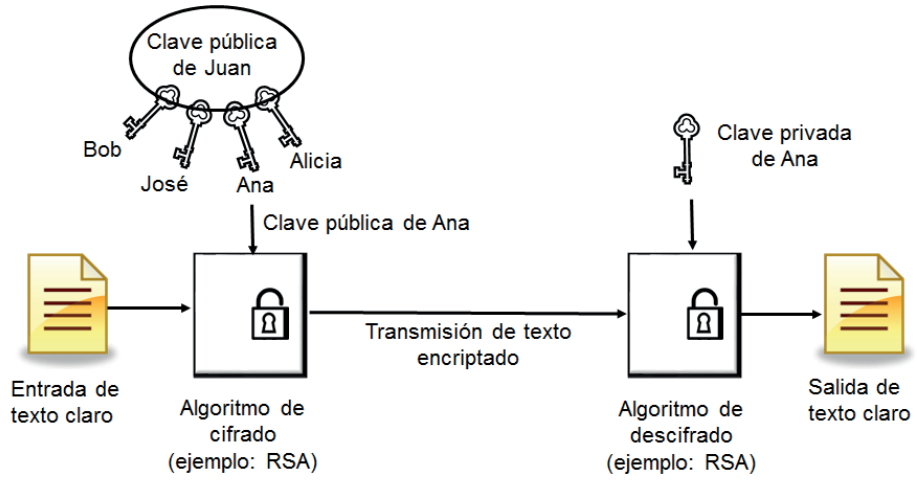


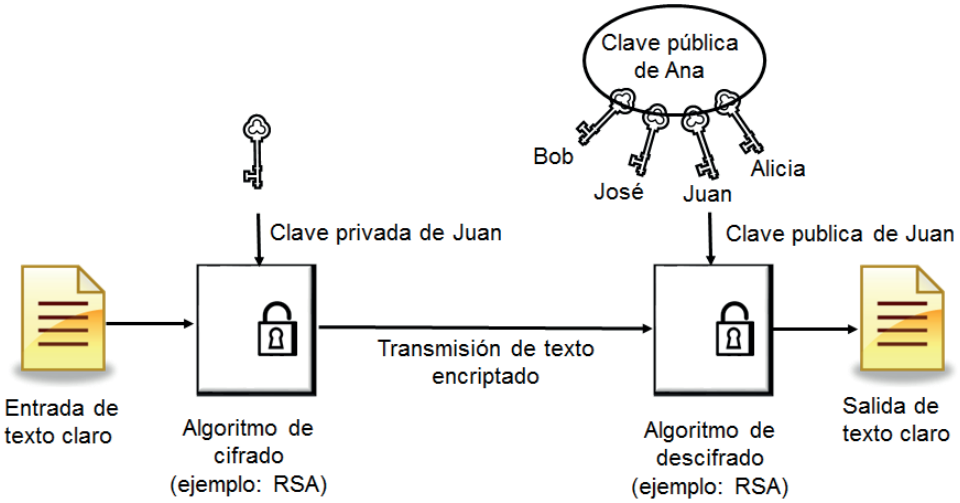
Figura 10.7. Modelo simplificado de encriptado asimétrico o público [Stalling, 2004]



En la figura 10.7, Juan se quiere comunicar secretamente con Ana, para lo cual usa la clave pública de Ana para cifrar su mensaje. Ana podrá abrir el mensaje cifrado usando su clave privada.

El método de clave asimétrica o pública también se puede usar para autenticar mensajes o firma digital. Este escenario es mostrado en la figura 10.8. En este caso, Juan quiere autenticarse ante Ana, para tal propósito necesita enviar su firma digital cifrada usando su clave privada. Ana autentica la firma usando la clave pública de Juan. Si Juan no fue quien envió la firma, entonces esta no podrá ser descifrada con su llave pública.

Figura 10.8. Modelo simplificado del uso de clave pública para autenticación [Stalling, 2004]



10.5.1 Requisitos para la criptografía de llave pública

Los requisitos que deben de ser observados para generar las llaves en el modelo de llave pública son [Stalling, 2004]:

1. Debe de ser computacionalmente fácil para generar un par de llaves (llave pública *KUb* y llave privada *KRb*).
2. Debe de ser fácil para el remitente generar texto cifrado (C):
$$C = E_{KUb}(M)$$
3. Debe de ser fácil para el receptor descifrar texto cifrado usando la llave privada:
$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$
4. Debe de ser computacionalmente imposible determinar la llave privada (KRb) conociendo la llave pública (KUb).
5. Debe de ser computacionalmente imposible recuperar el mensaje M, sabiendo KUb y el texto cifrado C.
6. Cualquiera de las dos llaves se puede usar para cifrar, mientras que la otra llave se usa para descifrar el mensaje (M):
$$M = D_{KRb}[E_{KUb}(M)] = D_{Kun}[E_{KRb}(M)]$$

10.6 AUTENTICACIÓN

La autenticación es el acto que sirve para establecer o confirmar algo (o a alguien) como auténtico. La autenticación de un objeto puede significar la confirmación de su procedencia, mientras que la autenticación de una persona a menudo consiste en verificar su identidad. La autenticación está ligada a las preocupaciones de seguridad en los sistemas distribuidos.

Entre las preocupaciones de seguridad se pueden citar las siguientes:

- Confidencialidad y puntualidad.
- Para proporcionar confidencialidad, se debe de cifrar la identificación y la información de la clave de sesión.
- ¿Qué requiere el uso de llaves públicas o privadas previamente compartidas?
- La puntualidad es requerida para prevenir los ataques de repetición.
- La puntualidad puede alcanzarse al usar números de secuencia o estampillas de tiempo o reto/respuesta.

10.6.1 Kerberos

El sistema Kerberos es un proyecto de autenticación desarrollado en el Instituto Tecnológico de Massachusetts [Kerberos, 2014]. El nombre de este proyecto está inspirado en la mitología griega, donde un perro de tres cabezas es el guardián de la entrada al infierno.

Cuando los usuarios desean acceder a los servicios en los servidores, existen las siguientes posibles amenazas:

- El usuario puede pretender ser otro usuario.
- El usuario puede alterar la dirección de la red de una estación de trabajo.
- El usuario puede espiar los intercambios y usar un ataque de repetición.

Kerberos proporciona un servidor de autenticación centralizado para autenticar los usuarios a los servidores y los servidores a los usuarios. El protocolo se basa en el cifrado convencional, sin hacer uso del encriptado de clave pública. Existen dos versiones, la 4 y 5. La versión 4 hace uso de DES.

Un diálogo hipotético simple de autenticación es mostrado en la figura 10.9, donde los términos usados son indicados en la figura 10.10.

Figura 10.9. Diálogo hipotético simple de autenticación

(1)	C → AS:	IDc || P_c || IDv
(2)	AS → C:	Ticket
(3)	C → V:	IDc || Ticket

Ticket = E_{Kv}[IDc || P_c || IDv]

Figura 10.10. Términos empleados en diálogo hipotético simple de autenticación

C = Cliente
AS = Servidor de autenticación
V = Servidor
Idc = Identificador de usario en C
IDv = Identificador de V
P_c = Contraseña de usuario en C
ADc = Dirección de red de C
Kv = Clave de cifrado secreta compartida por AS en V
TS = Estampilla de tiempo
|| = Concatenacion

Riesgos a considerar durante el diseño:

- El tiempo de vida de un mensaje es asociado con el ticket de concesión.
- Si es demasiado corta, entones es solicitado repetidas veces por la contraseña.
- Si es demasiado larga, entonces existe una mayor oportunidad para un ataque de repetición.

La amenaza es que un oponente se robe el ticket y lo use antes de que el tiempo expire.

La figura 10.11 muestra el protocolo real de Kerberos. En este se considera el problema del ticket TGT (ticket que concede un ticket) capturado y la necesidad de determinar que quien presenta el ticket es el mismo que el cliente para quien se emitió dicho ticket. Aquí existe la amenaza de que un oponente pueda robar el ticket y usarlo antes de que expire. Kerberos soluciona este problema usando una clave se sesión (mensaje 2 de la figura 10.11).

Con el ticket y la clave de sesión, C está preparado para dirigirse al TGS. Para esto, C envía al TGS un mensaje que incluye el ticket y el ID del servicio solicitado (mensaje 3 de la figura 10.11). Adicionalmente, C transmite un autentificador que incluye el ID y la dirección del usuario de C y un sello de tiempo.

La respuesta desde el TGS (mensaje 4) sigue la forma usada en mensaje 2 [Stalling, 2004]. Es decir, el mensaje se cifra con la clave de sesión compartida por el TGS y C, e incluye una clave de sesión que comparten C y V, así como el ID de V y el sello de tiempo del ticket.

El usuario C envía este ticket de servicio reutilizable al servidor V (mensaje 5) junto con un autentificador. El servidor V descifra el ticket, recupera la clave de sesión y descifra el autentificador. En caso de que se requiera autentificación mutua, el servidor responde como se indica en el mensaje 6 de la figura 10.11. El esquema general de Kerberos se muestra en la figura 10.12.

Figura 10.11. Diálogo de autenticación de Kerberos [Stalling, 2004]

Intercambio de Servicio de Autenticacion: Para obtener un TGT (otorgamiento de entrada)

(1) C → AS: IDc || IDtgs ||TS1

(2) AS → C: $E_{Kc}[K_{c,tgs}|| IDtgs || TS_2 || Lifetime_2 || Ticket_{tgs}]$

Intercambio de TGS: Para obtener ticket de concesión de servicio

(3) C → TGS: IDv ||$Ticket_{tgs}$ ||Authenticatorc

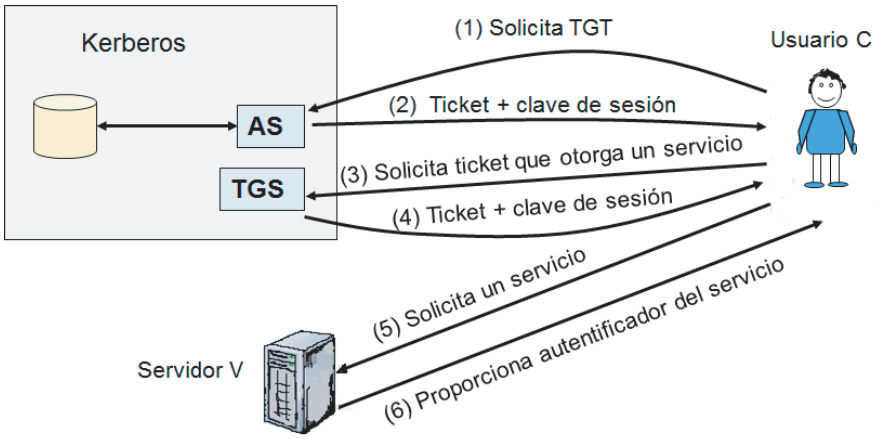
(4) TGS → C: $E_{Kc}[K_{c,v}|| IDv || TS_4 || Ticket_v]$

Intercambio de autenticación Cliente/Servidor: Para obtener un servicio

(5) C → V: $Ticket_v$ || $Authenticator_c$

(6) V → C: EKc,v[TS5 +1]

Figura 10.12. Esquema general de Kerberos



Existen casos en que las redes de clientes y servidores pertenecen a diferentes organizaciones constituyendo diferentes dominios. Sin embargo, los usuarios de un dominio pueden necesitar acceso a servidores de otro dominio y algunos servidores pueden estar dispuestos a ofrecer el servicio a esos usuarios, previa autenticación. En este entorno se propone un esquema conocido como Kerberos múltiple [Stalling, 2004]. Este esquema necesita lo siguiente:

1. El ID de usuario y la contraseña de todos los usuarios están registrados en la base de datos del servidor Kerberos.
2. Todos los servidores están registrados con el servidor Kerberos, quien comparte una clave secreta con cada uno de ellos.
3. Los servidores Kerberos de cada dominio se registran entre sí y comparten una clave secreta entre ellos.

La interoperabilidad de un servicio Kerberos múltiple para dos dominios es mostrado en la figura 10.13.

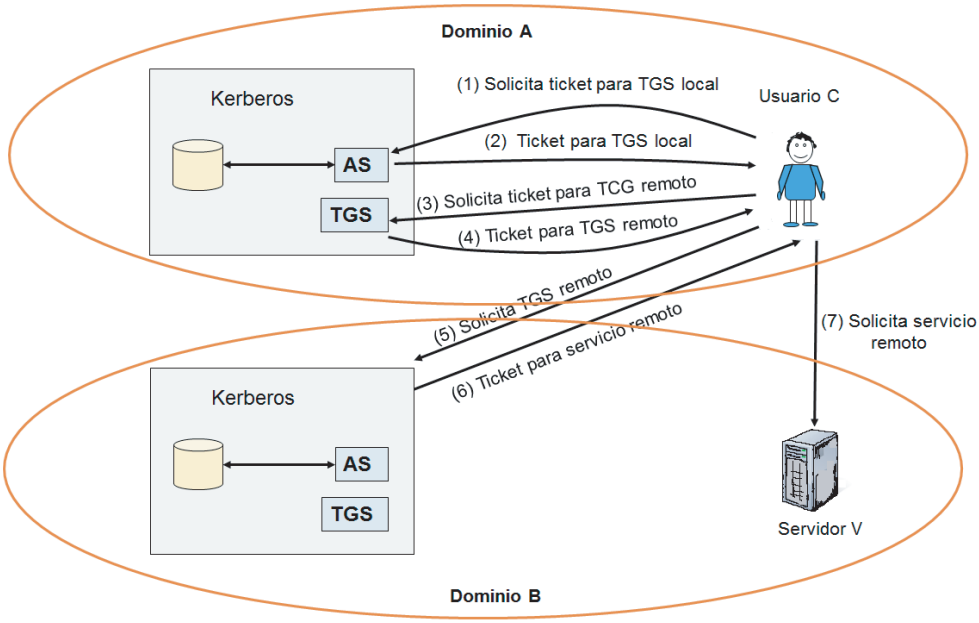
La versión 4 (Kerberos v4) está restringida a un solo dominio y la versión 5 (Kerberos v5) permite autenticación entre varios dominios. Kerberos v5 es un estándar de Internet especificado en RFC1510 y es usado por muchas aplicaciones.

Entre las principales diferencias entre la versión 4 y 5 se encuentran las siguientes:

- Dependencia del sistema de cifrado (V.4 DES).
- Dependencia del protocolo de Internet.

- Ordenamiento de bits de mensaje.
- Tiempo de vida del ticket.
- Reenvío de autenticación.
- Autenticación de intercampos.

Figura 10.13. Kerberos para solicitud de servicio en otro dominio



EJERCICIOS

1. Cita al menos tres requisitos que debe de cumplir la criptografía de llave pública.

2. Indica los componentes que constituyen la criptografía de llave pública y describe el escenario (usando un esquema) para un caso de encriptamiento y un caso de autenticación.

3. ¿Cuál es el proposito de la estampilla de tiempo en el sistema Kerberos?

4. Explica en qué consiste un ataque de interrupción.

5. Explica qué es un ataque a la confidencialidad.

6. ¿Qué tipo de ataque sería un ataque a la autenticidad?

7. Cita algunas recomendaciones relacionadas con las políticas en la seguridad.

8. Investiga el uso del algoritmo de llave pública (asimétrico) en la firma digital.

9. Cita dos requisitos para la criptografía de llave pública.

10. ¿Cuál es la función del tráfico de relleno en un mecanismo de seguridad?

11. ¿Cuáles son los aspectos que se deben de cuidar en un enfoque de seguridad en la información?

ACTIVIDAD INTEGRADORA

1. De la aplicación de los contenidos, así como de las habilidades y actitudes desarrolladas.
   Se recomienda exposición general del tema por parte del profesor, resaltando la importancia de los sistemas de seguridad en los sistemas distribuidos desde una perspectiva de los sistemas criptográficos y la autenticación de acceso a los sistemas. Debido a que los alumnos de la Licenciatura en Tecnologías y Sistemas de Información cursan una UEA llamada Seminario de seguridad, se recomienda no ahondar mucho en el tema, sino comprender los riesgos que implica tener la información en entornos distribuidos y las posibles soluciones de seguridad que se pueden ofrecer. El tema puede cubrirse en dos sesiones.
   El alumno debe de mostrar un gran interés por la observación, análisis, abstracción e interpretación de los riesgos de seguridad en los sistemas distribuidos, así como de algunos esquemas de seguridad para mitigar este tipo de riesgos.
2. Del logro de los objetivos establecidos en el capítulo o apartado del material.
   Se espera alcanzar los objetivos de este capítulo con el apoyo en la solución de los ejercicios, la exposición del profesor sobre el tema, así como un taller de análisis de diagramas sobre riesgos de seguridad y sus probables soluciones para mitigar estos riesgos.

Capítulo 11. Multimedia distribuida

Objetivo: Que el alumno comprenda aspectos del flujo de video respecto a sus estándares e infraestructuras de distribución, así como su importancia en los sistemas distribuidos actuales.

11.1. Introducción

Multimedia quiere decir "múltiples medios". Los medios pueden ser desde texto e imágenes hasta animación, sonido o video. Específicamente, la multimedia es una combinación de texto, sonidos, imágenes o gráficos ya sea estáticos o en movimiento. El video es un tipo de contenido que combina los tipos de medios anteriores, por lo que se constituye en un caso ideal de multimedia.

Los últimos avances de las tecnologías de Internet y la computación han abierto nuevas oportunidades para aplicaciones multimedia. En este nuevo escenario, la transmisión de video a través de Internet tiene una gran popularidad. Este hecho ha generado una revolución tecnológica y social, y diferentes sistemas de distribución de video han emergido. La transmisión de video se realiza desde un proveedor de servicio centralizado o desde múltiples proveedores de servicios a una gran multitud de personas, quienes utilizan diversas técnicas, tales como video bajo demanda o video IP para obtenerlos. La reproducción de video, en general, requiere de una alta velocidad de datos, baja latencia o de alto rendimiento, con el fin de ofrecer una buena calidad de transmisión. Sin embargo, la calidad de servicio (QoS) para la transmisión de video a través de Internet sigue siendo insuficiente e inconsistente. Esto se traduce en tiempos de transferencia muy largos y la imposibilidad de ver un video en tiempo real. Este capítulo revisa los con-

ceptos básicos relacionados con la difusión del video, como infraestructura y técnicas de codificación de video. El capítulo también analiza brevemente algunos sistemas de flujo de video.

Durante el flujo de video, el receptor puede ver el video mientras que una parte del archivo aún se sigue recibiendo y decodificando. De esta manera, la transmisión de video reduce el tiempo entre el inicio de la entrega y el inicio de la reproducción en el visor. Este retardo normalmente es de 5 a 15 segundos [Apostolopoulos, Tan & Wee, 2002]. Por otro lado, ya que solo una pequeña parte del video se almacena por el espectador durante la transmisión de video, los requisitos de almacenamiento son bajos. Sin embargo, la transmisión de video es sensible a la demora, debido a que los paquetes de video deben de llegar al receptor antes de sus plazos de emisión [Wang, Ostermann & Zhang, 2002]. Por lo tanto, este tipo de transmisión, como ya se señaló, requiere de alta velocidad de datos, baja latencia o de un alto rendimiento, lo cual es difícil de conseguir, ya que actualmente la Internet está basada en el protocolo IP [Wu *et al.*, 2001], que no garantiza una buena calidad de servicio (QoS).

11.2 Estándares de codificación de video

La compresión es un tema importante a considerar para la distribución de video digital, pues ayuda a reducir la alta tasa de bits resultante de la conversión digital a un nivel que puede ser soportado por las redes de comunicación. La compresión elimina las redundancias inherentes en señales de audio y de video digitalizadas con el fin de reducir la cantidad de datos que necesita ser almacenada y transmitida. Para alcanzar esta meta, se ha realizado un gran esfuerzo durante los últimos 30 años en el desarrollo de técnicas y estándares de compresión de video. Algunos de estos estándares de compresión son:

- H.261 [ITU-T, 1993].
- MPEG-1 [ISO / IEC, 1993].
- H.262 (MPEG-2) [ITU-T e ISO / IEC JTC, 1994].
- H.263 [ITU-T, 1995].
- MPEG-4 [ISO / IEC, 1999].
- H.264 / AVC [2009].
- H.265.

La norma H.265 es la más reciente especificación de video, sucesora del estándar H.264/AVC. En julio de 2014 se completó la segunda parte del estándar y se programó su publicación para finales de 2014. Este nuevo estándar incluye diferentes extensiones de mejora de video, entre las que se encuentran video en 3D. En este capítulo nos referimos al estándar H.264/AVC y una de sus extensiones.

11.2.1 Estándar H.264/AVC

H.264/Video Advanced Coding –comúnmente conocido como H.264 / AVC [2009]– es el estándar de codificación de video desarrollado por la ITU-T Video Coding Grupo de Expertos y la ISO/IEC Moving Picture Experts Group [Wiegand, Sullivan, Bjontegaard & Luhtra, 2003]. La estandarización de H.264 / AVC, también llamada MPEG-4/Video Advanced Coding, se completó en mayo de 2003 [ITU-T, 2003]. H.264/AVC ofrece una amplia gama de velocidades de bits, velocidades de cuadro y resoluciones espaciales, por lo que es muy versátil y garantiza su funcionalidad para aplicaciones diversas [Marpe, Wiegand & Sullivan, 2006], [Richardson, 2007], como difusión sobre los distintos tipos de redes, almacenamiento interactivo o de serie, el video bajo demanda o flujos de video.

Conceptualmente, H.264/AVC incluye [Wiegand *et al.*, 2003]:

- *La capa de codificación de video (VCL)*: Crea una representación codificada de los datos de video de la fuente con el fin de proporcionar flexibilidad y personalización para aplicaciones y redes heterogéneas.
- *La capa de abstracción de red (NAL)*: Formatea la VCL para hacerlo compatible con diferentes capas de transporte y dispositivos de almacenamiento, además de proporcionar la información de cabecera necesaria en un formato adecuado.

H.264/AVC organiza los datos de video codificados en unidades NAL, paquetes que contienen, cada uno, un número entero de bytes. Una unidad NAL comienza con una cabecera de un byte, que indica el tipo de los datos contenidos. Los bytes restantes representan datos de carga útil. Un conjunto de unidades NAL consecutivos, con propiedades específicas, se conoce como una unidad de acceso. La decodificación de una unidad de acceso se traduce en exactamente una imagen decodificada. Un conjunto de unidades de acceso consecutivos con ciertas propiedades se refiere como una secuencia de video codificada. Una secuencia de video codificada

representa una parte independientemente decodificable de una corriente de bits unidad NAL. Por otro lado, el diseño básico de VCL en H.264/AVC es muy similar al de las normas de codificación de video anterior, tales como H.261, MPEG-1 video, H.262 MPEG-2 de video, H.263, o MPEG-4 Visual. Sin embargo, H264/AVC incluye nuevas características que permiten una mejora significativa en la eficiencia de compresión con respecto a cualquier norma de codificación de video anterior. La principal diferencia con respecto a las normas anteriores es el gran aumento en la flexibilidad y la adaptabilidad.

11.2.2 Codificación de video escalable (SVC)

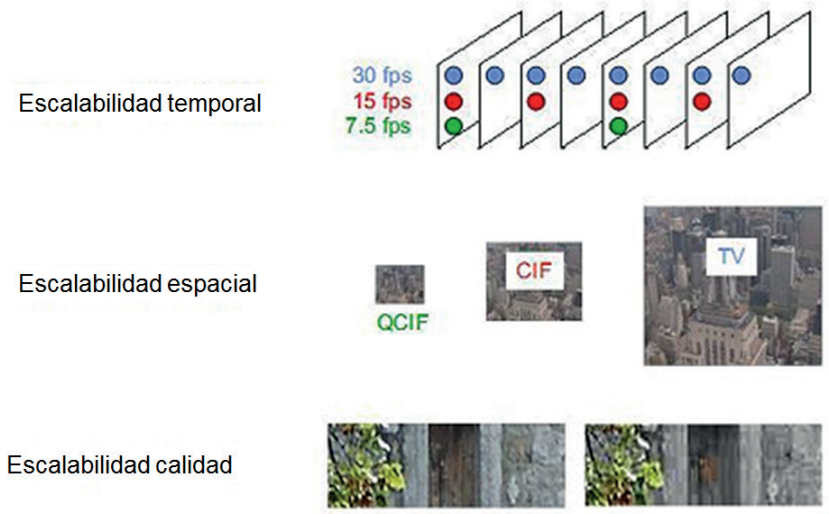
La capacidad de recuperar información importante de una imagen o video mediante la decodificación de solo partes de un flujo de bits comprimido se conoce como escalabilidad [Wang *et al.*, 2002]. Usando codificación de video escalable, partes de una secuencia de video se pueden quitar y la subclasificación resultante constituye otra fuente de video válido para algún decodificador objetivo. Este flujo de video parcial representa el contenido de código con una calidad de reconstrucción, y que es menor que la calidad del flujo de video completa del video original, pero alta si se considera la menor cantidad de datos restantes [Schwarz, Marpe & Wiegand, 2007]. Actualmente las redes heterogéneas están de moda y parecen ser la tendencia hacia el futuro. En estas condiciones, SVC emerge como una solución valiosa por adaptar fácilmente la velocidad de codificación a las distintas tasas de transmisión y diferentes dispositivos físicos.

SVC representa el video en una capa base (BL) y una o más en las capas de mejora (EL), donde cada EL actualiza de manera acumulativa la calidad de video aportado por la BL. En esencia, SVC codifica el video una vez, en un flujo de bits comprimido. Cada uno de los diferentes usuarios puede ahora extraer la cantidad justa de los datos de esta secuencia de video común, según las condiciones de la red o el tipo de dispositivo que esté utilizando. La principal limitación es que para decodificar una capa de mejora particular, se requiere haber obtenido antes la capa base (BL), así como todas las capas de mejora inferiores. Las formas habituales de escalabilidad son (ver figura 11.1):

- Escalabilidad de calidad o escalabilidad SNR (relación señal-ruido).
- Escalabilidad espacial.
- Escalabilidad temporal.

En la escalabilidad SNR, el video es representado por diferentes capas que varían el nivel de calidad de percepción. La decodificación de la capa base proporciona una baja calidad del video reconstruido. Sin embargo, la calidad resultante del video reconstruido se incrementa mediante la decodificación de las capas de mejora. En escalabilidad espacial, la BL y el EL suelen utilizar diferentes resoluciones de imagen espacial (la resolución de la EL es más alta que la BL), por lo que es útil para dispositivos con tamaño limitado de pantalla, como los teléfonos celulares (*smartphones*). Por último, la escalabilidad temporal permite diferentes resoluciones temporales o tasas de cuadros para representar el mismo video. Esta solución permite a los usuarios con conexiones más lentas de datos acceder al mismo contenido de video, pero con velocidades más lentas.

Figura 11.1. Tipos básicos de codificación de video escalable



Cortesía de Fraunhofer Heinrich Hertz-Institute [HHI, 2010].

11.3 INFRAESTRUCTURAS PARA FLUJOS DE VIDEO

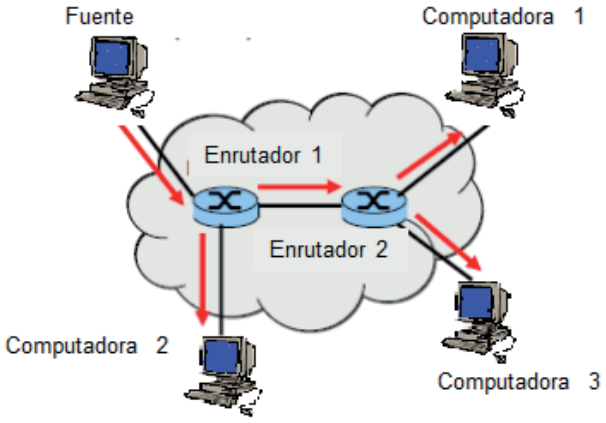
Esta sección describe las diferentes alternativas para distribución de video. Estas infraestructuras son:

- Multicast IP (multidifusión IP).
- Red de distribución de contenidos.
- Multicast de capa de aplicación.

11.3.1 Multicast IP

IP Multicast fue propuesto por Deering [1988] como una solución para realizar difusión de contenidos de uno-a-muchos. Multicast presenta una mejor eficiencia que unicast debido a su reducida sobrecarga de transmisión en el emisor y la red, lo que reduce el tiempo de entrega para la distribución de contenido. Una alternativa de distribución basada en unicast requiere que la fuente envíe un flujo individual a cada usuario final, lo cual es crítico para aplicaciones de alto ancho de banda, como el video, el cual requiere una gran porción de ancho de banda para un solo flujo. IP Multicast reduce el tráfico mediante la distribución simultánea de una sola copia a miles de potenciales usuarios finales, mientras que los paquetes de multidifusión se replican en la red por los enrutadores. Diferentes aplicaciones, como videoconferencias, educación a distancia y noticieros, se basan en la tecnología de multidifusión. La figura 11.2 muestra cómo se distribuyen los datos de una fuente a varios usuarios finales usando multicast IP.

Figura 11.2. Multicast IP



Multicast IP ha demostrado ser una tecnología de rendimiento eficiente para la entrega de datos desde una fuente a un gran número de receptores. Desafortunadamente, IP multicast no se ha desplegado plenamente en la Internet debido principalmente a:

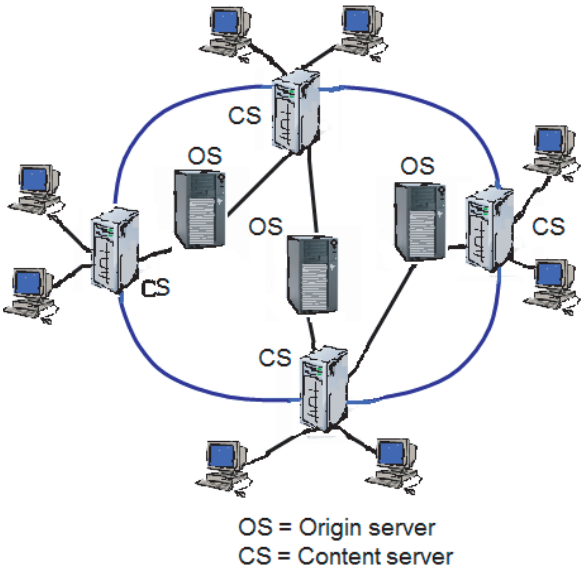
- Que su implementación requiere apoyo de enrutamiento de todos los ISP (proveedores de servicios de Internet), lo cual implica modificaciones sustanciales de la infraestructura de Internet.

- Cuestiones relacionadas con el control y gestión de la red [Pendarakis, Verma & Waldvogel, 2001], como la fiabilidad de extremo a extremo, y el flujo y control de congestión.

11.3.2 Red de distribución de contenidos

En la Internet actual, las aplicaciones de flujos de video de uno-a-muchos se basan en el modelo cliente-servidor tradicional de redes de distribución de contenido (CDN). Soluciones comerciales como Akami [Kontothanassisy *et al.*, 2004], [Akami, 2014], y otras, se ofrecen a través de un sistema de CDN. Un CDN está formado por servidores de contenidos conectados en red a través de Internet, que cooperan entre sí para distribuir contenidos de modo transparente para los usuarios finales. Un ejemplo de una CDN se muestra en la figura 11.3.

Figura 11.3. Red de distribución de contenidos



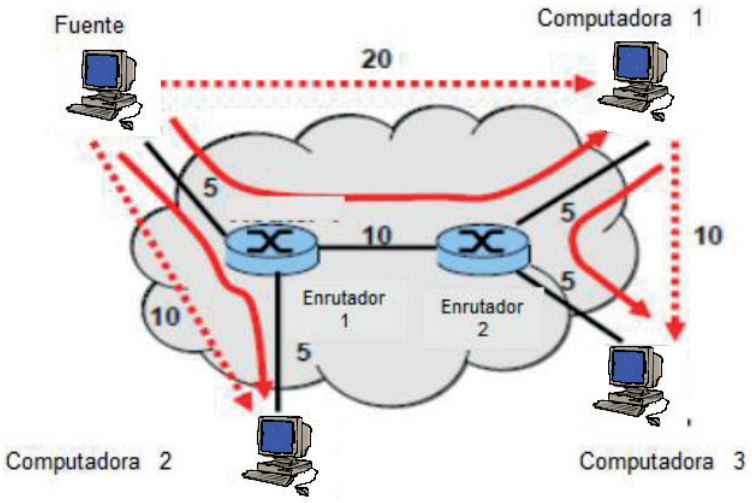
Típicamente, los servidores de contenido (CS) están situados cerca de los usuarios para servir el contenido solicitado rápidamente. Los servidores de contenido de la CDN están conectados a los proveedores de contenido (OS) a través de una red interna, que se utiliza para transferir el contenido de los proveedores a los servidores de contenido (CS).

Sin embargo, el enfoque de CDN se enfrenta a una serie de problemas tales como que el punto único de falla y el acceso es costoso para las redes de alta tasa. Además, incluso los grandes servidores de flujos de video no son capaces de alimentar a más de algunos cientos de sesiones de flujos de video de manera simultánea y la selección del mejor servidor de video en un CDN durante una sesión es difícil. Estas restricciones limitan el desempeño de la CDN.

11.3.3 Multicast de capa de aplicación

Para afrontar las limitantes de los modelos ya explicados, en los últimos años diversos investigadores optaron por utilizar soluciones en el nivel de aplicación del modelo OSI como una alternativa para implementar multidifusión [Pendarakis *et al.*, 2001], [Banerjee, Bhattacharjee & Kommareddy, 2002], [Chu, Rao, Seshan & Zhang, 2002]. En el multicast de capa de aplicación (ALM), todas las tareas de multidifusión se aplican en las computadoras anfitrionas finales exclusivamente, mientras que la infraestructura de la red se mantiene. Un ejemplo de este tipo de sistemas son las redes P2P. La figura 11.4 representa un ejemplo ALM, donde los números indican los retrasos de enlace.

Figura 11.4. Multicast de capa de aplicación



Un inconveniente en el sistema de ALM es la penalización de rendimiento asociada con entornos dinámicos y heterogéneos de Internet, debido a

que el rendimiento se afecta por la ubicación y la estabilidad de los usuarios finales.

11.4 SISTEMAS DE FLUJOS DE VIDEO BASADOS EN REDES P2P

Un sistema P2P de distribución de video implica dos componentes importantes [Magharei & Rejaie, 2006a]:

- Una red superpuesta.
- Un mecanismo de entrega de contenido.

Una red superpuesta se construye sobre la red subyacente física IP [Gao & Huo, 2007] utilizando un mecanismo que determina cómo se conectan los pares. El mecanismo de entrega de contenido es responsable de la transmisión de los contenidos de cada peer a través de la red de superposición. Una multidifusión P2P de superposición tiene las siguientes ventajas sobre multicast IP [Zheng *et al.*, 2005]:

- No se requiere soporte de enrutadores.
- Mayor flexibilidad y adaptabilidad a las diversas necesidades de las aplicaciones.

Zheng *et al.* [2005] señalan que para construir y mantener una red P2P de superposición eficiente, principalmente deben de considerarse tres problemas :

1. La arquitectura de la red P2P, la cual define la topología que se utiliza para construir la red superpuesta.
2. La gestión de la red y cómo se gestionan los peers en el grupo multicast, especialmente cuando los peers presentan capacidades y comportamientos heterogéneos.
3. La adaptabilidad de la red de superposición para encaminar y planificar el video en un ambiente de Internet donde los enlaces tienen un comportamiento impredecible.

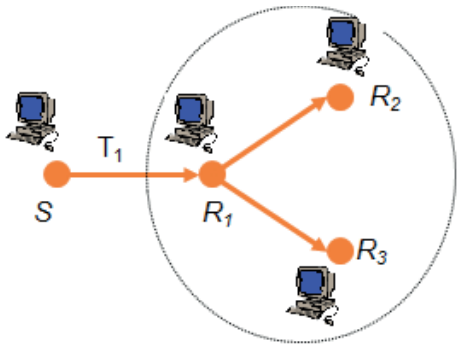
Se han propuesto varias soluciones con el fin de abordar estas cuestiones. Los principales tipos de topologías P2P para ofrecer multidifusión P2P son [Magharei & Rejaie, 2006], [Gao & Huo, 2007], [Venot & Yang, 2007]:

- Árbol.
- Bosque.
- Malla.

11.4.1 Topología de flujo de video P2P basado en árbol

En el enfoque basado en árbol, un mecanismo de construcción de superposición organiza a los peers participantes en un solo árbol cuya raíz se encuentra en el nodo fuente. Los peers que participan están organizados como nodos interiores o nodos de hoja en un solo árbol. En la figura 11.5 se muestra una topología basada en árbol. La fuente S envía los datos al peer R1, que reenvía los datos a los peers R2 y R3. Un flujo de video en esta configuración es básicamente empujado desde un enrutador padre a sus enrutadores hijos a lo largo de una ruta bien definida.

Figura 11.5. Topología de flujo de video P2P basado en árbol

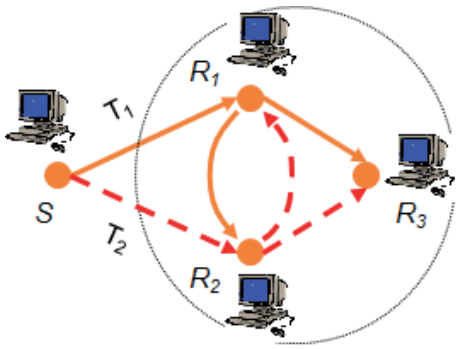


En la figura 11.5 se observa que la capacidad del peer R1 es utilizado por el árbol de multidifusión para la distribución de contenidos, mientras que la capacidad de los peer hoja R2 y R3 no se utiliza. Aunque un enfoque de árbol probablemente representa la estructura de distribución más eficaz en términos de ancho de banda y optimización de retardo [Wang *et al.*, 2007], esta configuración tiene un inconveniente debido a que toda la carga generada por el reenvío de mensajes multicast la realizan un número pequeño de nodos interiores.

11.4.2 Topología de flujo de video P2P basado en bosque

Un enfoque basado en bosque organiza a los peers participantes en una difusión de video en múltiples árboles [Gao & Huo, 2007] y distribuye la carga de reenvío entre estos de una manera eficiente. En una distribución basada en bosque, cada peer determina un número adecuado de árboles para unirse sobre la base de su capacidad de carga. La figura 11.6 muestra un ejemplo de una superposición basado en bosque. En este escenario, los peers participantes se organizan en varios árboles. Para este fin, cada peer se coloca como un nodo interno en al menos un árbol y como un nodo de hoja en otros árboles. La fuente S divide su contenido en dos pedazos y distribuye cada pedazo usando dos árboles de multidifusión separados T1 y T2. Cada nodo interno en cada árbol de distribución reenvía la parte de contenido recibida a todos sus nodos secundarios. Por lo tanto, el algoritmo de construcción del árbol representa un componente importante del enfoque basado en bosque.

Figura 11.6. Topología de flujo de video P2P basado en bosque



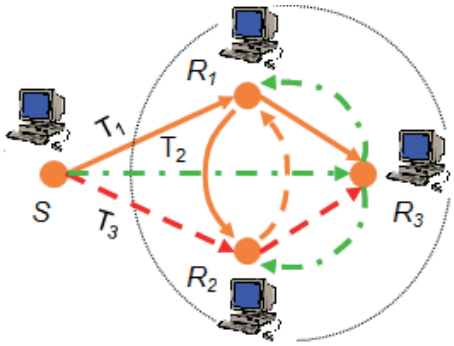
Una estrategia basada en múltiples árboles reduce al mínimo el efecto de rotación [Magharei *et al.*, 2007] y optimiza el uso de los recursos disponibles en el sistema. Sin embargo, la determinación del número de árboles necesarios para maximizar el rendimiento global es un problema abierto.

11.4.3 Topología de flujo de video P2P basado en malla

Las redes superpuestas basadas en malla es un enfoque inspirado en BitTorrent [Cohen, 2003] o Bullet [Kostic, Rodriguez, Albrecht & Vahdat, 2003]. La

figura 11.7 presenta un ejemplo de superposición basado en malla. Dicho esquema está formado por peers conectados al azar, donde cada peer (excepto la fuente) intenta mantener cierto número de peers padres y también sirve a un número determinado de peers hijos utilizando un mecanismo de enjambre para la entrega de contenido [Magharei & Rejaie, 2006b]. En una superposición basada en malla, un peer puede concurrentemente recibir datos desde diferentes emisores, contribuyendo cada uno con una parte de su capacidad de carga. Además, los peers que solicitan también pueden enviar y recibir datos entre sí.

Figura 11.7. Topología de flujo de video P2P basado en malla



Una topología basada en malla es robusta, sin embargo, puede presentar un alto retardo y sobrecarga. También, debido al comportamiento dinámico e impredecible de los peers, es difícil seleccionar los proveedores de video adecuados, así como la manera adecuada de cooperar y programar los datos de peers solicitantes. La tabla 11.1 resume las principales características de estas topologías.

Tabla 11.1. Principales características de las topologías P2P para flujos de video

Basada en árbol		Basado en bosque		Basado en malla	
Ventajas	*Desventajas*	*Ventajas*	*Desventajas*	*Ventajas*	*Desventajas*
Estructura simple	No usa capacidad de los peers hojas	Usa capacidad de peers hojas	La construcción es compleja	Usa capacidad peers hojas	Largas demoras en el arranque
Baja sobrecarga	Vulnerabilidad	Calidad escalable	Vulnerabilidad	Robustez	Sobrecarga
Arranque rápido				Alta disponibilidad	Reproducción de video se congela frecuentemente
Ejemplos de sistemas: ZigZag End-System Multicast		Ejemplos de sistemas: Splitstream CoopNet		Ejemplos de sistemas: BitTorrent Mutualcast	

Los contenidos multimedia se pueden distribuir en Internet tanto como transmisiones en vivo o bajo demanda. Algunos ejemplos de sistemas de difusión de video comercial y experimental son PPLive [2014] y P2P-Next [2014].

P2P-next es un proyecto Paneuropeo lanzado en enero de 2008, el cual es patrocinado por 21 socios industriales, proveedores de contenidos multimedia e instituciones de investigación. P2P-Next es un proyecto integrado a gran escala, cuyo objetivo es construir una plataforma de entrega de contenido de próxima generación basado en el paradigma peer-to-peer (P2P). Los principales objetivos de este proyecto son [Wilson & Miles, 2010]:

- Una plataforma que soporte portales de video de banda ancha y entregue contenidos a través de redes P2P a pantallas de televisión y a computadoras personales.
- Integrar funciones de Gestión de Derechos Digitales (DRM) y funciones para ayudar a los productores de contenidos a convertir contenido de video lineal en contenidos interactivos.
- Crear modelos de negocios sostenibles para el sistema P2P-Next.

EJERCICIOS

1. Actualmente Internet no ofrece reservación de recursos o calidad de servicio (QoS). ¿Cómo logran alcanzar las aplicaciones de flujo de video actual algunos niveles de calidad?

2. Explica la diferencia entre flujo de video y descarga de video

3. ¿Qué ventajas con respecto al almacenamiento y la reproducción introduce el flujo de video?

4. Investiga las principales diferencias entre el estándar H.265 y H.264.

5. ¿Qué ventaja introduce la distribución de video basado en bosque con respecto a la distribución de video basado en árbol?

6. ¿Por qué existe la posibilidad de que la reproducción de un video se congele frecuentemente en una distribución basada en malla?

7. ¿Cuáles son las principales limitantes para desplegar sistemas de video usando IP Multicast?

8. ¿Cuáles son los principales beneficios de usar codificación de video escalable?

9. Cita algunos casos en que sea aplicable y útil usar codificación de video escalable.

10. ¿Por qué es más robusta una red P2P basada en malla que una red P2P basada en único árbol?

ACTIVIDAD INTEGRADORA

1. De la aplicación de los contenidos, así como de las habilidades y actitudes desarrolladas.
   Se recomienda una exposición general por parte del profesor. Además, se pueden dejar temas específicos a investigar al alumno como actividad

extraclase para desarrollar y presentar una exposición la última sesión del curso. Se recomienda realizar algún prototipo de flujo de video que puede servir como proyecto final.

El alumno de debe mostrar un gran interés por el diseño y programación de infraestructuras de redes sobrepuestas basadas en protocolos TCP/IP para flujos de video.

2. Del logro de los objetivos establecidos en el capítulo o apartado del material.

   Se espera alcanzar los objetivos de este capítulo con el apoyo en la solución de los ejercicios, la exposición del profesor, el desarrollo de un prototipo básico como proyecto final y la exposición de los alumnos en la última sesión de clases.