Apuntes de Sistemas Distribuidos

por Ing. Cristina Abad Robalino
Facultad de Ingeniería en Electricidad y Computación (FIEC)
Escuela Superior Politécnica del Litoral (ESPOL)

Guayaquil, 20 de abril de 2007

Índice

Políticas del Curso

Profesor: Ing. Cristina Abad Robalino
Término: I 2007
E-mail: cabad@fiec.espol.edu.ec, cabadr@espol.edu.ec
Oficina: En el bloque de oficinas bajo secretaría de la FIEC (junto a oficina Ing. Vaca)
Página Web: http://www.visid.espol.edu.ec

Prerrequisitos:

- Fundamentos de Redes de Datos (o Introducción a las Redes de Computadoras)
- Comunicaciones de Datos
- Sistemas Operativos
- Programación Orientada a Objetos

Material de Referencia:

- George Coulouris, Jean Dollimore y Tim Kindberg, "Distributed Systems: Concepts and Design", 4ta edición. Addison-Wesley 2005.
- Markus Hofmann y Leland Beaumont, "Content Networking: Architecture, Protocols and Practice", Morgan Kaufmann 2005.
- Revista en línea "Distributed Systems Online", IEEE, edición mensual, http://dsonline.computer.org

Objetivo General del Curso:
Al finalizar el curso, el estudiante será capaz de implementar sistemas distribuidos medianos, así como evaluar sistemas existentes, en base a sus propiedades y diseño.

Contenido:

1. Introducción
    a. Características
    b. Complejidad
    c. Ejemplos
    d. Middleware
    e. Modelos (arquitectónico y fundamental)
2. Bases de redes para sistemas distribuidos
    a. Internet
    b. TCP y UDP
    c. HTTP
    d. Sistema DNS
3. Paradigmas de programación distribuida
    a. Comunicaciones entre procesos
    b. Objetos distribuidos
    c. Eventos distribuidos
4. Seguridades
    a. Pilares y ataques
    b. Encriptación y hash criptográficos
    c. Certificados digitales
    d. SSL
5. Teoría
    a. Tiempo lógico y tiempo físico
    b. Otros: elecciones y coordinación

6. Redes de contenidos
    a. Cacheo y replicación
    b. Facultando las redes de distribución de contenidos
    c. P2P
7. Tópicos especiales (opcionales)

Calificaciones:

- Parcial
    - Tareas de programación: 25%
    - Participación, lecciones sorpresa y otras tareas: 15%
    - Examen: 60%
- Final
    - Proyecto: 30%
    - Participación, lecciones sorpresa y otras tareas: 10%
    - Examen: 60%
- Mejoramiento
    - Proyecto: 30%
    - Examen: 70%

Asistencia:

Los estudiantes son responsables de asistir a clases regularmente y de ponerse al día con la materia en caso de faltar a clases. Tomaré lista todos los días, 15 minutos después de empezada la clase. Esta asistencia cuenta como parte de la nota de participación en clase. Los estudiantes atrasados generalmente no tendrán asistencia. Cualquier estudiante con 40% o menos de asistencia, deberá realizar un trabajo extra al finalizar la materia o caso contrario, reprobará la misma.

Tareas:

Cuando asigne una tarea, siempre indicaré en clase la fecha de entrega. Las tareas entregadas a tiempo se reciben sobre el 100% de la nota de la misma. Tareas entregadas hasta la clase siguiente en la cual la tarea fue recogida, serán calificadas sobre el 50% de la nota de la misma. Atrasos mayores a este no serán permitidos. A menos de que en clase indique lo contrario, las tareas siempre se recibirán impresas, durante el horario de la clase para la cual fue planificada. No se aceptarán tareas enviadas vía correo electrónico o Metis.

Participación en clase:

Esta nota comprende:

- Asistencias
- Participación activa en clase
- Actividades grupales desarrolladas durante la clase

Lecciones:

Podrá haber en cualquier momento una lección sorpresa del material cubierto desde la última lección sorpresa. Las lecciones se tomarán durante los 15 primeros minutos de la clase. Los estudiantes atrasados no podrán participar de la lección y perderán esos puntos.

Proyecto:

Los proyectos se desarrollarán en grupos de hasta 2 estudiantes. Los detalles del mismo serán proporcionados en otro documento.

Comunicaciones, información y material del curso:
La principal fuente de información será lo dado en clase. Si un estudiante falta a clase, deberá averiguar si se hizo algún anuncio durante la misma. El no haber asistido a clase no es excusa para no realizar una tarea u otra actividad asignada durante la misma. Gran parte de los anuncios de clase también serán publicados en Metis. Es responsabilidad del alumno revisar periódicamente la información publicada en Metis.

Cualquier duda con respecto al material del curso, la pueden consultar en mi oficina o publicar en la sección de foros de Metis. Yo contestaré estas dudas periódicamente. Adicionalmente, cualquier colaboración de calidad por parte de los alumnos en los foros tendrá una incidencia positiva en la nota de participación en clase.

Exámenes:
Los exámenes escritos cubrirán todo el material cubierto en clase y tareas. El mayor peso del examen será para preguntas de análisis y razonamiento, y no para preguntas de memorización de la materia. Por esta razón es muy importante que asistan a clases.

Exposiciones:
De vez en cuando, plantearé temas para ser expuestos en clase (individualmente por un estudiante). Los estudiantes pueden auto-nominarse para presentar determinado tema (ya sea planteado en clase o sugerido por los alumnos). Si no hay voluntarios, escogeré en clase a un estudiante para que presenten el tema. Si un estudiante que ha sido asignado para presentar un tema, no llega preparado o falta a clase, tendrá 5 puntos menos en la nota de participación. La presentación deberá de ser de máximo 10 minutos, con una ronda adicional de preguntas. Estas presentaciones tendrán un puntaje de HASTA 5 puntos (extra sobre la nota), dependiendo de la calidad de la presentación y del nivel de preparación de los estudiantes. La principal fuente de material para estas presentaciones es http://dsonline.computer.org. Sugiero que no esperen a que yo seleccione un tema a ser expuesto, sino que en cualquier momento se pueden acercar e indicarme que desean exponer un artículo cualquiera.

Plagio y copia:
A menos que indique lo contrario, las tareas son individuales. Es permitido trabajar en grupo, siempre y cuando cada cual desarrolle el trabajo individualmente. En caso de detectar tareas que sean, íntegra o parcialmente, copias de las tareas de otro(s) estudiante(s), recibirán una nota máxima del 10% del valor de la misma. Para los proyectos y tareas de programación, se incentiva que investiguen en Internet y hasta que adapten soluciones bajadas de Internet, siempre y cuando (1) hagan referencia al material citado, (2) indiquen qué parte del desarrollo es propio y qué parte es copia o adaptación, y (3) haya una contribución adicional significativa del grupo. Caso contrario, el proyecto o tarea de programación será calificado sobre el 10% de la nota del mismo.

Otros:
La materia y proyectos son muy interesantes, pero requieren preparación por parte ustedes. Se espera que participen activamente en clase y que hagan preguntas si un tema no queda claro. Sugerencias sobre cómo mejorar el curso serán bienvenidas.

¡Suerte!

1. Introducción

Objetivo

Introducir los conceptos básicos que nos permiten describir y entender a los sistemas distribuidos, así como proporcionar un marco de referencia sobre el cual analizaremos y clasificaremos las tecnologías estudiadas en este curso.

Palabras Clave

agente móvil
aplicación distribuida
calidad de servicio (QoS)
alta disponibilidad
canal (de comunicaciones)
cliente
cliente ligero
cliente/servidor
concurrencia
confidencialidad
detección de fallas
disponibilidad
efecto slashdot
enmascaramiento de fallas
escalabilidad
falla arbitraria
falla bizantina
falla de omisión
falla de tiempo
flash crowds
granja de servidores
integridad
middleware
paso de mensajes
peer
peer-to-peer (P2P)
red ad-hoc
reloj global
RFC
servidor
sistema distribuido
sistema distribuido abierto
sistema distribuido asíncrono
sistema distribuido síncrono
sistema multi-capa
tolerancia a fallas
transparencia de acceso
transparencia de localización

Enlaces de Interés

- IEEE Distributed Systems Online: http://dsonline.computer.org/
- Flash Crowds: http://en.wikipedia.org/wiki/Flash_crowd
- Efecto “slashdot”: http://en.wikipedia.org/wiki/Slashdotted
- “The Byzantine Generals Problem”: http://research.microsoft.com/users/lamport/pubs/byz.pdf
- “End-to-End Arguments in System Design”: (http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf
- “Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites”: http://nms.lcs.mit.edu/~jyjung/p342-jung/HTML/p342-jung.html
- “CNN.com: Facing A World Crisis”: http://www.tcsa.org/lisa2001/cnn.txt

Apuntes

Sistemas Distribuidos

Ing. Cristina Abad R., MSc

2007

Metodología

- Calificación
  - Proyecto, deberes (y participación), exámenes
- Información en línea:
  - Metis
- Consultas: cabad@fiec.espol.edu.ec
  - OJO: No enviar e-mails a cabad@espol.edu.ec, ya que pertenece a un estudiante!
- Más detalles en el documento de políticas

Sobre esta clase

- Los sistemas distribuidos están en todas partes
- Objetivo: diseñar e implementar sistemas distribuidos medianos, así como entender los principios y dificultades de los mismos
- Pre-requisitos recomendados
  - Sistemas Operativos
  - Buen nivel de programación (C++ y/o Java)
  - Fundamentos de Redes de Datos y Comunicaciones de Datos

Introducción

Caracterización

¿Qué van a aprender?

- Dificultades al desarrollar sistemas distribuidos
- Bases teóricas
- Objetos y eventos distribuidos
- Tecnología middleware
  - Sockets, RMI, etc.

Definición

- Un sistema distribuido es aquel en que los componentes (hw o sw) se encuentran en computadoras conectadas a alguna red, que se comunican y coordinan sus acciones por medio del paso de mensajes.

Características

- Concurrencia de componentes
- Falta de un reloj global
- Independencia de las fallas de los componentes

Fallas independientes

- Fallas no son predecibles
- Puede fallar
  - Computadoras individuales
  - Red

Concurrencia

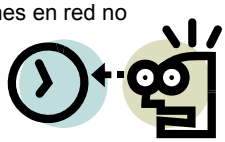
- Cada computadora está atendiendo requerimientos diferentes
- Usuarios se comportan de manera diferente
- Recursos cambian frecuentemente
- Miembros del sistema deben compartir recursos y coordinarse

¿Por qué construir un sistema distribuido?

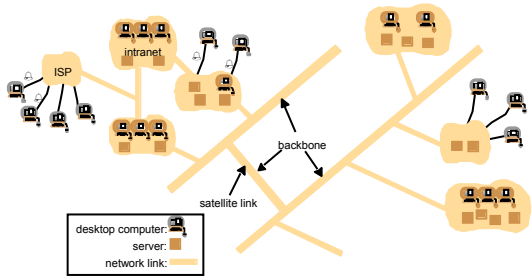
- Compartir recursos
- Ejemplos:
  - WWW
  - Cajeros automáticos
  - P2P (eDonkey, Kazaa, Gnutella)

No hay un reloj global

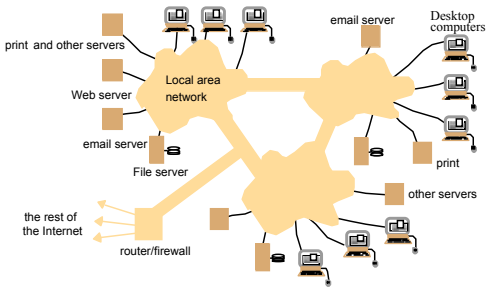
- Programas cooperan usando mensajes
- Cierta coordinación requiere saber en qué momento ocurrieron las cosas
  - Relojes tienen horas diferentes
  - Retraso en transmisiones en red no predecibles



Ejemplo 1: Internet



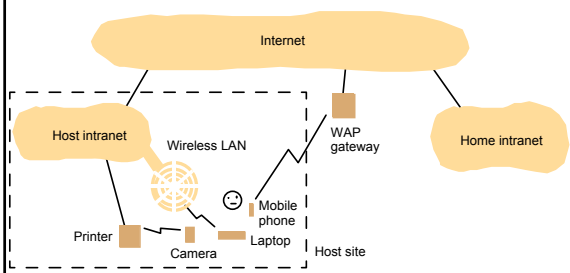
Ejemplo 2: Una Intranet



Complejidad

- Componentes heterogéneos
- Apertura (componentes pueden ser añadidos, reemplazados o eliminados)
- Seguridad
- Escalabilidad
- Manejo de fallas
- Concurrencia de componentes
- Transparencia

Ejemplo 3: Sistemas móviles



Componentes heterogéneos

- Redes
- Hardware
- Sistemas operativos
  - Representación de datos
- Lenguajes de programación
- Implementaciones

- Estándares
- Middleware
- Código portable
  - Java
- XML
- …

Aplicaciones distribuidas

- Conjunto de procesos distribuidos en una red de computadoras que trabajan juntas para resolver un problema en común
- Antes: principalmente cliente-servidor
  - Recursos administrados centralizadamente (servidor)
- Ahora: Peer-to-peer, se encaminan a ser “verdaderas” aplicaciones distribuidas

Definición: Middleware

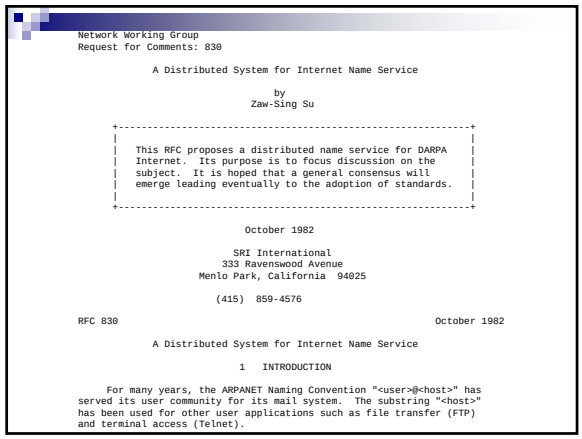
- Capa de software que proporciona una abstracción para programación y a la vez esconde la heterogeneidad de las redes, S.O, y lenguajes de programación
  - Ej.: CORBA, Java RMI
- Generalmente se implementa sobre los protocolos de Internet, que a la vez esconden la heterogeneidad de la red, representación de datos, etc.
- Modelos:
  - Remote method invocation
  - Remote event notification
  - Remote SQL access
  - Transacciones

Apertura

- Determina cuán fácilmente se pueden añadir nuevos recursos y nuevos tipos de recursos sin perturbar el sistema
- Interfaces deben ser publicadas (RFCs)
- Sistemas distribuidos abiertos

Escalabilidad

- Habilidad de trabajar bien aún cuando el número de usuarios aumente
  - Costo de recursos físicos
  - Disminución del rendimiento
  - Limitados recursos de software
  - Cuellos de botella
- Flash crowds (multitudes repentinas):
  - http://en.wikipedia.org/wiki/Flash_crowd
- Efecto “slashdot”:
  - http://en.wikipedia.org/wiki/Slashdotted



Network Working Group
Request for Comments: 830

A Distributed System for Internet Name Service

by
Zaw-Sing Su

This RFC proposes a distributed name service for DARPA Internet. Its purpose is to focus discussion on the subject. It is hoped that a general consensus will emerge leading eventually to the adoption of standards.

October 1982

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

(415) 859-4576

RFC 830 October 1982

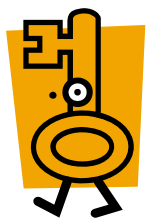
A Distributed System for Internet Name Service

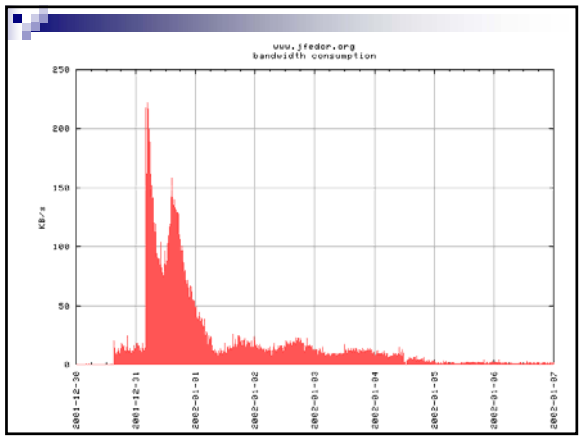
1 INTRODUCTION

For many years, the ARPANET Naming Convention "<user>@<host>" has served its user community for its mail system. The substring "<host>" has been used for other user applications such as file transfer (FTP) and terminal access (Telnet).

Seguridad

- Confidencialidad
  - Encriptación
- Integridad
  - Checksums
  - Autenticación
- Disponibilidad
  - Replicación
  - Autenticación
  - DoS





Manejo de fallas

- Detectar
- Enmascarar
- Tolerar
- Recuperarse de
- Redundancia

Modelos

Concurrencia

- Clientes tratan de tener acceso a recursos simultáneamente
  - Y si ambos cambian el valor simultáneamente?
  - Sistema debe ser “seguro”
- Sincronización para obtener “seguridad”
  - Estado del sistema debe ser consistente

Modelos

- Arquitectónico
  - ¿Dónde colocamos las partes?
  - ¿Cómo se relacionan las partes?
- Fundamental
  - Descripción formal de las propiedades comunes a todos los modelos arquitectónicos

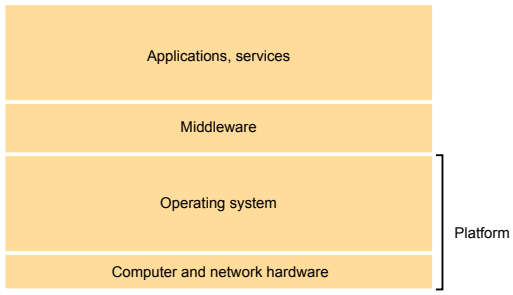
Transparencia

- Componentes del sistema deben estar ocultos y debe parecer uno solo
- Transparencias:
  - Acceso – usando operaciones idénticas
  - Localización – poder usar recursos sin saber dónde están
- Ejemplo: e-mail

Modelo arquitectónico

- Tipos de procesos
  - Servidores
  - Clientes
  - Compañeros (peer)

Capas de servicios de SW y HW en los sistemas distribuidos



Limitaciones del middleware

- Algunos problemas se pueden resolver de manera más eficiente en los “extremos” y no en las capas intermedias
  - Detección de errores
  - Encriptación
  - Etc.

Middleware

- Enmascara heterogeneidad
- Proporciona modelo de programación conveniente para programadores de aplicaciones
- Conjunto de procesos u objetos en un grupo de computadoras que interactúan entre si para implementar comunicación y soporte de recursos compartidos para sistemas distribuidos
- Bloques útiles para construir SW distribuido

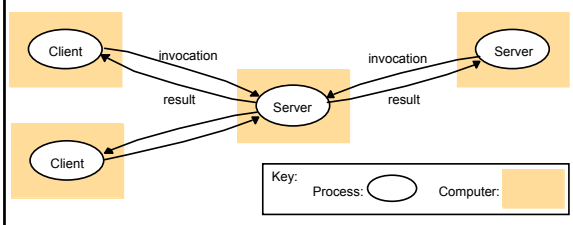
Principales modelos arquitectónicos

- Cliente-servidor
- Servidores múltiples
- Servidores proxy y caches
- Procesos compañeros (peer)

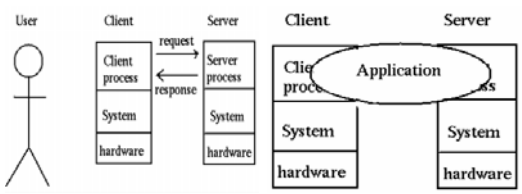
Middleware (cont…)

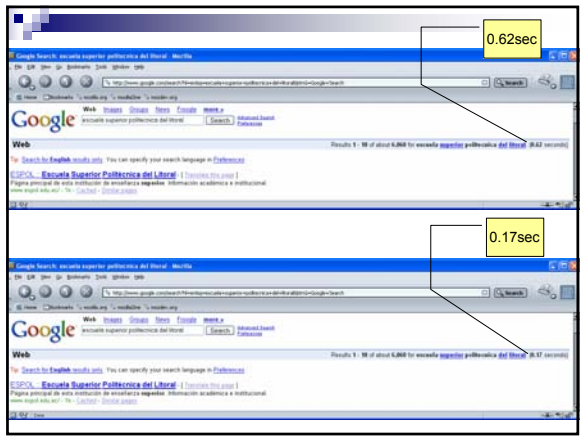
- Soporte de abstracciones
  - Invocación remota de métodos
  - Comunicación de grupos
  - Notificación de eventos
  - Replicación de datos
- Servicios
  - Naming
  - Seguridades
  - Transacciones
  - Almacenamiento persistente

Cliente/servidor

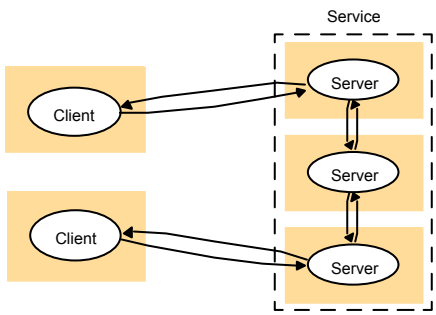


Cliente/servidor (cont…)

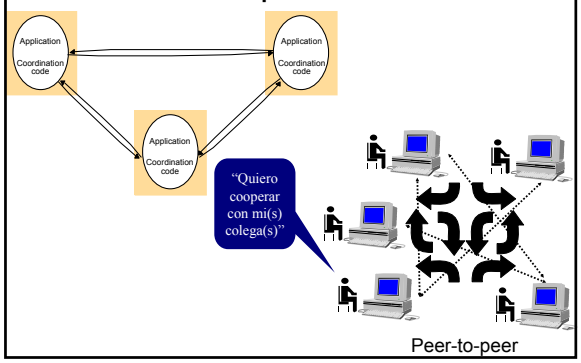




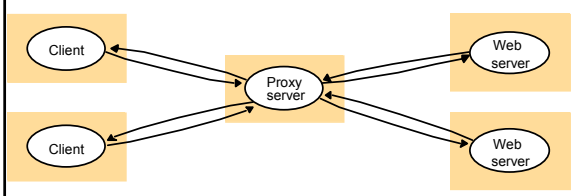
Múltiples servidores



Procesos compañeros



Servidores proxy y caches

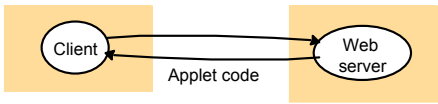


Variantes modelo cliente-servidor

- Código móvil
- Agentes móviles
- Computadoras en red (networked computers)
- Clientes ligeros (thin clients)
- Redes ad-hoc o espontáneas

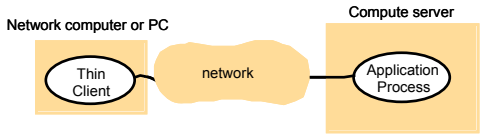
Código móvil: web applets

a) client request results in the downloading of applet code



b) client interacts with the applet

Clientes ligeros (thin clients)

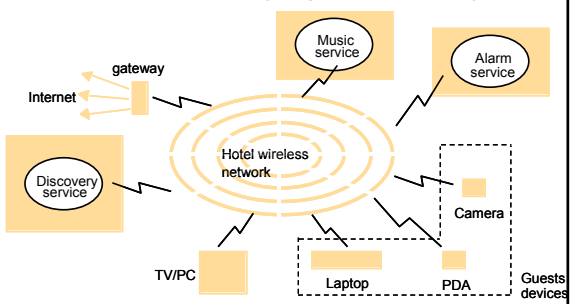


- Muy lento para ciertas aplicaciones
- Ej..: aplicaciones para X11 en Unix

Agentes móviles

- Código y datos
- Ejemplos:
  - Gusanos (e.g. Morris Worm)
  - Web crawlers
- Gran riesgo (seguridades)
- Existen alternativas más seguras
- Aplicabilidad limitada

Redes ad-hoc (espontáneas)

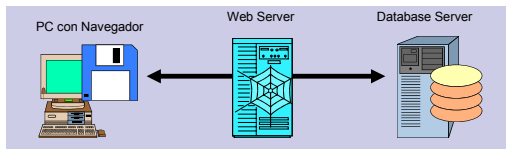


Computadoras en red

- Recursos en red
  - Sistema operativo
  - Software
  - Datos
- Computadora baja lo que necesita de la red y lo usa localmente
- Usuarios pueden usar recursos desde distintas computadoras

Terminología cliente/servidor

- Dos capas (two tier)
  - Cliente interactúa directamente con servidor
- Tres capas o multi-capa (three/multi tier)
  - Middleware



Requerimientos de diseño

- Rendimiento
  - Tiempo de respuesta, balanceo de cargas, throughput
- Calidad del servicio (QoS)
  - Confiabilidad, seguridad y rendimiento
- Cacheo y replicación
- Poder depender de (el sistema)
  - Correctness, seguridad y tolerancia a fallas
  - Alta disponibilidad

Sistemas síncronos

- Características
  - Tiempo en ejecutar cada paso tiene límite inferior y superior conocidos
  - Tiempo en transmitir mensajes tiene límites conocidos
  - Cada proceso tiene un reloj local con ratio de cambio conocido
- No existen en la realidad (excepto en sistemas de tiempo real)
- Utilidad teórica

Modelos fundamentales

- Modelo de interacción
- Modelo de fallas
- Modelo de seguridades

Sistemas asíncronos

- No se sabe los límites de:
  - Velocidad de ejecución de los procesos
  - Retrasos en la transmisión de mensajes
  - Ratio de variación de los relojes
- Soluciones válidas para sistemas asíncronos también sirven para sistemas síncronos
- Ej.: la web (navegadores)

Modelo de interacción

- Factores
  - Rendimiento de canales de comunicación
    - Latencia
    - Ancho de banda
    - Variación (jitter)
  - Relojes y toma de tiempo a eventos
    - Ratio de cambio del reloj (existe y difiere)
- Variantes (de sistemas distribuidos)
  - Síncronos
  - Asíncronos

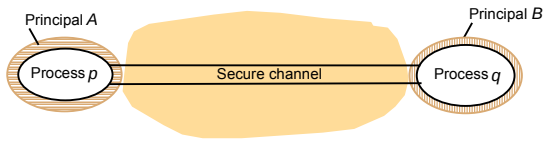
Modelo de fallas

- Fallas de omisión
  - Proceso o canal no hace lo que debe hacer
    - Ej.: Una PC que se bloquea
  - Dificultad: saber si ha fallado o solo está lento
  - Tres tipos: extremos (2) o canal
- Fallas arbitrarias o Bizantinas
  - Es el peor tipo de falla
  - No suelen ocurrir en el canal de comunicación
- Fallas de tiempo
  - En sistemas síncronos

Enmascaramiento de fallas

- Para obtener sistemas confiables usando partes no confiables
- Maneras de enmascarar fallas
  - Escondiendo la falla
    - Ej.: Rollback en bases de datos
  - Convirtiendo la falla en otro tipo más manejable
    - Ej.: checksums en transmisión de paquetes, convierten una falla bizantina en falla de omisión

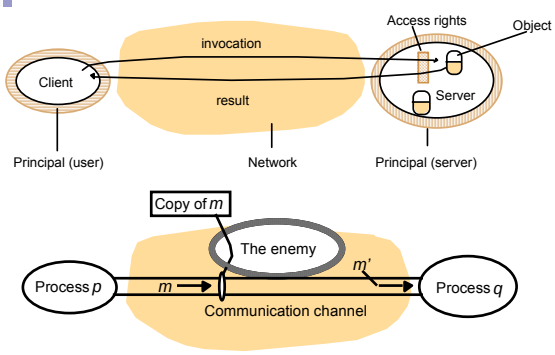
Canales seguros



NOTAS:

Modelo de seguridades

- Se debe proteger
  - Procesos
    - Memoria: control de acceso
    - Autenticación: firmas digitales
  - Canales ➔ canales seguros
    - Proteger los mensajes: encriptación
    - Integridad: checksums y firmas digitales
    - Disponibilidad: Ej.: DoS
  - Objetos
    - Derechos de acceso



Ejercicios

1. Identifique las ventajas y desventajas de los sistemas totalmente distribuidos o P2P (vs. los sistemas cliente/servidor).
2. Los sitios Web son frecuentemente organizados en un esquema cliente-servidor multi-capa (multi-tier), el cual consiste de un servidor Web, un servidor de aplicaciones y un servidor de bases de datos. ¿Cuáles son las ventajas de un esquema de este tipo?
3. ¿En qué consiste el enmascaramiento de fallas? Dé un ejemplo de enmascaramiento de una falla bizantina.
4. Dé un ejemplo que ilustre por qué no es una buena idea siempre tratar de lograr una transparencia de distribución completa.
5. Un sistema peer-to-peer híbrido (como Napster) depende de un servidor de directorios centralizado que mantiene una lista de archivos disponibles en los clientes conectados. Cada búsqueda cliente va inicialmente al servidor, el cual la procesa y responde con una lista de clientes disponibles que mantiene el archivo buscado. Discuta las limitaciones de este enfoque centralizado, en un ambiente a gran escala como Internet.
6. Las aplicaciones en Internet se pueden clasificar como cliente/servidor, peer-to-peer o híbridas (mezcla entre P2P y C/S). Proporcione un ejemplo de una aplicación de cada uno de estos tipos.
7. Los tres niveles lógicos en arquitecturas cliente-servidor son:
   - Interfaz de usuario
   - Aplicación o lógica del negocio
   - Base de datos o datos (archivos, etc.)

   Estos tres niveles pueden ser distribuidos entre el cliente y el servidor de cinco maneras diferentes mostradas a continuación. Proporcione ejemplos de aplicaciones reales que sigan estos cinco modelos (un ejemplo para cada caso).

Tareas

1. Investigue en Internet sobre las fallas arbitrarias o bizantinas y conteste las siguientes preguntas:
   a. ¿Por qué se llaman fallas Bizantinas?
   b. ¿Quién les dio ese nombre?
   c. Dar tres ejemplos ligados a sistemas distribuidos existentes.
   d. Análisis: ¿Por qué es el peor tipo de falla?
   e. Incluir fuente(s) bibliográficas (Nota: el artículo original está disponible en: http://research.microsoft.com/users/lamport/pubs/byz.pdf).
2. Lea el artículo "End-to-End Arguments in System Design" (http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf) e identifique en qué medida se aplica el principio descrito al concepto de middleware.

Caso de Estudio: CNN.com: Facing A World Crisis

Lea las notas detalladas a continuación, sobre una charla titulada “CNN.com: Facing A World Crisis”, que tuvo lugar en el “15th Large Installation System Administration Conference (LISA '01)”. Luego, conteste las siguientes preguntas:

- ¿Qué ocurrió con CNN.com el 9/11?
- Desde el punto de vista técnico, ¿por qué ocurrió el problema?
- ¿Qué hicieron los administradores de CNN.com para minimizar el impacto del problema?
- De las otras situaciones similares listadas en el documento, escoja una y explique con sus palabras la problemática de ese sitio Web en ese evento en particular.
- Extra: Investigue cómo Akamai permite evitar problemas como el analizado en este caso de estudio.

Referencia: http://www.tcsa.org/lisa2001/cnn.txt.

```
CNN.com: Facing A World Crisis
William LeFebvre, CNN Internet Technologies

Abstract:
In the span of 15 minutes the demand for our site increased by an
order of magnitude.  On September 11, with only 85% availability,
we still served over 132 million pages, nearly equaling our site's
all-time high.  On September 12, we shattered any previous site
records with 304 million page views.  This talk will tell the
story of CNN.com and the team that met the unbelievable user demand.


Tom Limoncelli opened the talk with:

  Are you okay?  Are you okay?  Are you okay?

This was a common question that I asked that day while I tried to
locate friends on September 11.  Later, the question became:

  Where were you on 9/11?

Bill Lefebvre is a longtime USENIX and SAGE member and tutorial
speaker.  He is a Technology Fellow at CNN Internet Technologies.

---

Who are we?  CNN.com (Turner Broadcasting)
- 50 web sites (CNN.com, cartoon network, etc)
- 200 servers

Network Bandwidth (on 9/11)
- 2 OC-12 1,244 Mbps
- 7 OC-3  1,085 Mbps
- Total   2,329 Mbps

Hardware
- Standard web server: Sun 420R 4x4 (4 CPU, 4GB RAM)
- CNN.com normally used a 15 server pool
- Load balancers front-end all web services

Typical Loads
            Peak     Total        Total
```

```
   Date    Hits/min  Hits/min   Page Views
- 9/11/00     220K     148M        11.8M    One year prior
- 11/8/00   1,217K     722M       139.4M    Day after Election Day
- 9/10/01     156K     104M        14.4M    Day before

  11/8/00 was the record page views to date

Managing Unexpected Loads
- swing servers (move servers from one web service to another)
- add additional servers
- reduce page complexity (remove advertisements, pictures, text)

Reducing Page Complexity
- Three page styles (standard, split, ultra light)
  - Standard
  - Split (half page with link to more info)
  - Untra light (minimal information, with links for more info)

On the morning of Sept 11, there were 10 servers providing CNN.com.

Loads on 9/11-12
            Peak     Total        Total
   Date    Hits/min  Hits/min   Page Views
- 9/10/01     156K     104M        14.4M    Day before
- 9/11/01   1,110K     411M       132.4M    Day of
- 9/12/01     948K     797M       304.8M    Day after

On 9/11, the peak demand was estimated at 1.8M hits per minute, or
20 times normal.

Timeline                          Hits/minute
-  8:46  AAL 11 hits Tower 1           84K
-  8:50                                87K
-  8:55                               129K    Page request rate doubled
-  8:56  First web page was published         every seven minutes
-  9:00                               229K
-  9:01  First 911 page was sent (to admin team)
-  9:03  UAL 175 hits Tower 2
-  9:03  Server swings begin
-  9:05  Server congestion collapse begins (system monitoring lost)
-  9:11  Second 911 page was sent (to admin team) [phone bridge]
-  9:15  Switch to split web page
-  9:17  NYC airports closed
-  9:27  Third 911 page was sent (to admin team) [new phone bridge]
-  9:40  US Airspace closed
-  9:42  18 servers providing CNN.com (at least)
-  9:43  AAL 77 hits the Pentagon
-  9:47  Switch to minimal web page
- 10:05  Tower 2 collapses
- 10:10  UAL 93 crashes in PA
- 10:28  Tower 1 collapses
- 10:30  24 servers providing CNN.com
- 11:00  44 servers providing CNN.com
- 11:15  Network ports shut down, effectively taking CNN.com down
- 11:30  HTML service restored, no images served
- 12:25  System Monitoring restored
- 13:00  52 servers providing CNN.com
- 13:30  Images restored
- 14:21  2 servers moved back to Cartoon network (kids sent home from
         school)
- 14:55  Published light page
- 16:15                            1,110K
```

Challenges
- rapid traffic growth (doubling every 7 minutes)
- senior staff travelling (3) (Bill was working from home that day)
- Conference call is required to reduce page
- no local phone bridges available
- external bridge was in NYC and became unavailable
- cascading failures - can't get out of without a site restart

Reduced Page
- request was made late
- 22K split page was not enough reduction
- light page was 1247 bytes of text, logo, one image
  - lowest ever amount of text

- Server Rate limiting might have helped server overload
- stripped configuration - turned off everything, did site restart

Surprises
- network congestion from broadcasts
- Hit max sessions on web servers, had to increase session limit

Aftermath
- volatility worse than ever before
- automate swing process (on to-do list for a year)
- faster page reduction
- network redesign
- increased WAN bandwidth (if the servers could have handled the load, the WAN link would have been saturated)
- Standing phone bridge reservation
- review crisis procedures

Why Do You Care?
- problem is rate of change in traffic
- Can't happen to you?
  - apoa.org, nbaa.org
    Pilot web sites swamped when US airspace closed
  - election.dos.state.fl.us
    Florida election results overwhelmed day after election
  - firestone.com
    Tire recall
  - jeffco.k12.co.us
    Columbine
  - aceshardware.com & slashdot effect
    New, improved web site announced on slashdot, site was down within 15 minutes because of traffic

Expect the unexpected

Contingency Plans
- Formalize
- write them
- rehearse them
- review and update them

US Airspace was shut down quickly because the plan to do so had been in place for 40 years and reviewed every year. Even though no one thought it would ever happen, they were prepared.

Auto-Evaluación

1. ¿Cuál de los siguientes tipos de sistemas distribuidos NO es una variante del esquema cliente/servidor?
   a. Código móvil
   b. Clientes ligeros
   c. Agentes móviles
   d. Sistemas peer-to-peer (P2P)
2. ¿Cuál de lo los siguientes puntos es una ventaja de los sistemas peer-to-peer (P2P)?
   a. Facilidad de administración
   b. Facilidad para añadir seguridades al sistema
   c. Tiempos de respuesta menores
   d. Mayor escalabilidad
3. ¿Cuál de los siguientes NO se utiliza para solucionar/mitigar los problemas que surgen debido a la heterogeneidad de componentes un los sistemas distribuidos?
   a. Middleware
   b. Cachés
   c. Estándares
   d. XML
4. ¿Cuál de lo los siguientes puntos es una desventaja de los sistemas P2P?
   a. Susceptible a cuellos de botella en la red
   b. Hay un punto único de fallo
   c. Baja escalabilidad
   d. Dificultad de administración
5. En sistemas distribuidos, ____ quiere decir que los componentes del sistema deben estar ocultos y deben parecer uno solo.
   a. apertura
   b. heterogeneidad
   c. concurrencia
   d. transparencia
6. ¿Cuál de los siguientes NO se utiliza para mejorar la escalabilidad de sistemas distribuidos?
   a. middleware
   b. cachés
   c. réplicas
   d. múltiples servidores
7. La interfaz de un sistema distribuido puede ser publicada en un ____.
   a. XML
   b. SHA
   c. RFC
   d. RSA

Asocie los siguientes términos con su respectivo concepto.

a. sistema distribuido
b. aplicación distribuida
c. middleware
d. escalabilidad
e. servidor

____ 1. conjunto de procesos distribuidos en una red de computadoras que trabajan juntas para resolver un problema en común

____ 2. sus componentes se encuentran en computadoras conectadas a alguna red, y se comunican y coordinan sus acciones por medio del paso de mensajes

____ 3. proporciona una abstracción para programación y a la vez esconde la heterogeneidad de los componentes del sistema

____ 4. habilidad de trabajar bien aún cuando el número de usuarios aumente

____ 5. subsistema que proporciona un tipo de servicio particular a clientes previamente desconocidos

2. Bases de Redes para Sistemas Distribuidos

Objetivo

Reforzar conceptos de redes de datos cuyo conocimiento es imprescindible para poder diseñar sistemas distribuidos eficaces y eficientes.

Palabras Clave

ACK
ARP
cabecera
calidad de servicio
capa de aplicación
capa de enlace de datos
capa de red
capa de transporte
colapso por congestión
control de congestión
control de flujo
datagrama
DNS
dominio
entrega ordenada
gateway
HTTP
incremento aditivo, decremento multiplicativo
IP
latencia
mejor esfuerzo
mensaje
multiplexación
orientado a conexión
paquete
puerto
router/ruteador
RTP
saludo de tres vías
servidor de nombres
TCP
TCP/IP
Ethernet
TFRC
trama
transmisión confiable
UDP
ventana deslizante

Enlaces de Interés

- Ethereal: A Network Protocol Analyzer: http://www.ethereal.com
- How Domain Name Servers Work: http://www.howstuffworks.com/dns.htm
- How DNS Works: http://www.faqs.org/docs/linux_network/x-087-2-resolv.howdnsworks.html
- Domain name system: http://en.wikipedia.org/wiki/Domain_Name_System
- Real-time Transport Protocol: http://en.wikipedia.org/wiki/Real-time_Transport_Protocol
- RTP, Real-time Transport Protocol: http://www.networksorcery.com/enp/protocol/rtp.htm
- Introduction to RTP A Made Easy Tutorial: http://geocities.com/intro_to_multimedia/RTP/
- TCP Friendly Rate Control (TFRC): http://www.ietf.org/internet-drafts/draft-ietf-dccp-rfc3448bis-01.txt
- Comparative analysis - TCP – UDP: http://www.laynetworks.com/Comparative%20analysis_TCP%20Vs%20UDP.htm

Apuntes

Bases de Redes para el Desarrolladores de Sistemas Distribuidos

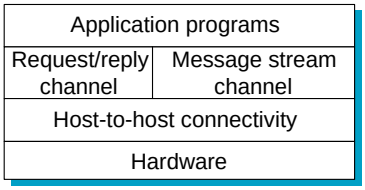
Arquitectura

Capas

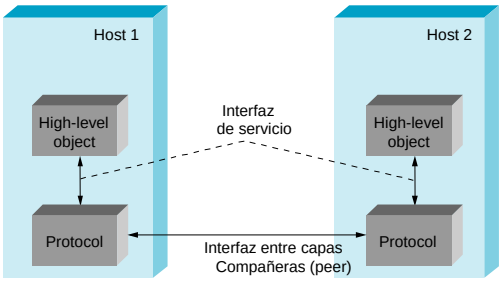
- Sistemas se vuelven cada vez más complejos
- Abstracciones son necesarias
  - Abstracción: define modelo que captura algunos de aspectos más importantes de un sistema, encapsula este modelo en un objeto que proporciona una interfaz que puede manipularse por otros componentes, y esconde los detalles de implementación
- Abstracciones llevan naturalmente a diseño por capas

Capas (cont…)

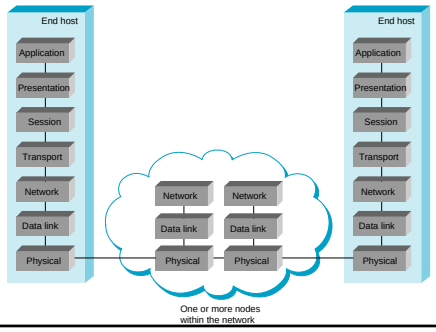
- Cada capa proporciona un nivel más alto (más abstracto) de servicio
- Servicios proporcionados en capas altas se implementan en términos de servicios de capas bajas



Interfaces de un protocolo

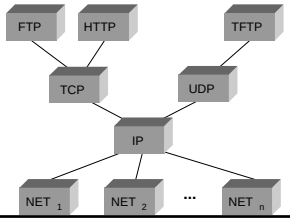


Modelo Referencial OSI (Open Systems Interconnection)



Arquitectura de Internet

- Definida por Internet Engineering Task Force (IETF)
- Diseño tipo “reloj de arena”
- Aplicación ≠ protocolo de aplicación



Internetworking

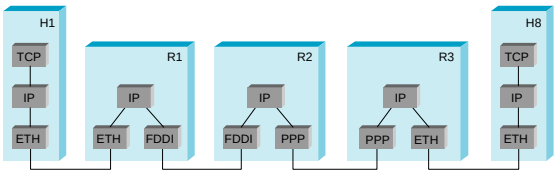
¿Qué es una Inter-red?

- También llamada internet (“i” minúscula)
- Colección de redes interconectadas para proporcionar servicio de entrega host-a-host
- Red lógica construida de redes físicas
- Ejemplos:
  - internets coorportivas que conectan varias redes departamentales y de sucursales
  - Internet

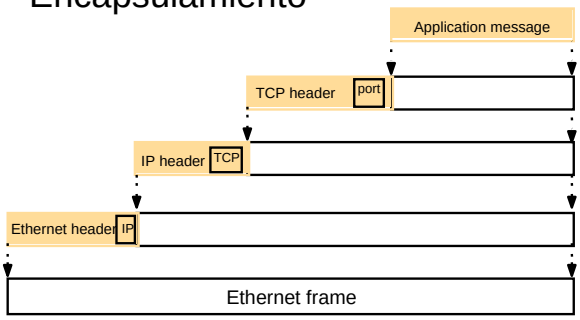
Routers

- A veces también llamados gateways
- Conectan redes para formar una inter-red
- Funcionan en capa 3 del modelo OSI
- Originalmente: implementados en SW
- Actualmente: implementados en HW
- Conecta al menos dos redes entre sí

Ejemplo

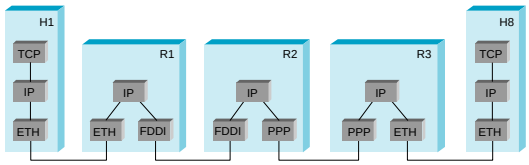


Encapsulamiento



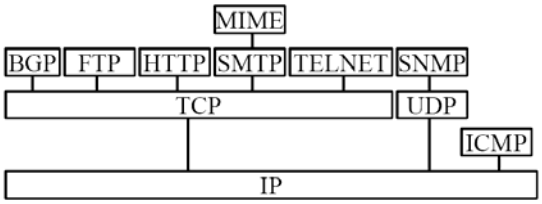
IP (Internet Protocol)

- Diseñado por Kahn y Cerf a principios de los 70s
- Protocolo de internetworking más escalable y exitoso



IP

- Base del conjunto de protocolos conocido como TCP/IP



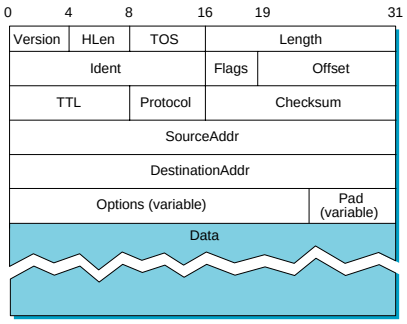
Modelo del Servicio

- Dos partes:
  - Esquema de direccionamiento
  - Modelo de datagrama para entrega de datos
- Servicio de *mejor esfuerzo* (best effort)
  - Sin garantías
    - Entrega puede demorarse
    - Entrega puede no realizarse

Modelo de Servicio: Implicaciones

- Datagrama
  - Cada paquete debe llevar suficiente información para ser ruteado a lo largo del camino origen-destino
- Mejor esfuerzo
  - Si un paquete se pierde/descarta, la red no hace nada por recuperarlo
    - Servicio no confiable
- Simpleza
  - Secreto del éxito de TCP/IP

Formato del Paquete



Protocolos Extremo-a-Extremo

Protocolos Extremo-a-Extremo

- Capa de red
  - Protocolo IP proporciona conectividad entre dos hosts
  - Necesario conectividad entre dos aplicaciones
- Protocolos extremo-a-extremo
  - Convierten mecanismos de entrega de host-a-host en canales de comunicación de proceso-a-proceso
  - Capa de transporte

Servicio Disponible

- Protocolo IP: mejor esfuerzo
  - Mensajes se descartan
  - Mensajes se re-ordenan
  - Se entregan duplicados de mensajes
  - Mensajes tienen un tamaño máximo
  - Mensajes pueden ser entregados luego de una demora arbitrariamente larga

Servicio Deseado

- Garantías de entrega de mensajes
- Entrega de mensajes en el mismo orden en que se envían
- Entrega de a lo mucho una copia de c/ mensaje
- Mensajes de tamaño arbitrariamente largo
- Sincronización entre origen y destino
- Permitir a receptor aplicar control de flujo a origen
- Soportar múltiples procesos/apl. en cada host

Servicio Deseado

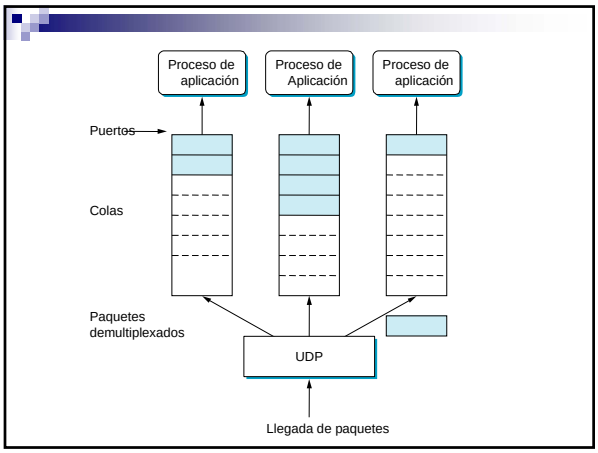
- Características adicionales que pueden ser proporcionadas por la capa de transporte
  - Seguridad
  - Entrega causal y con ordenamiento total

Identificación de Procesos

- Usar id del proceso
- Usar un localizador abstracto: puerto
  - Proceso origen envía mensajes a un puerto
  - Proceso destino procesa mensajes recibidos por un puerto
- ¿Cómo sabe una aplicación el número del puerto del proceso con el cual se quiere comunicar?
  - Comunicación cliente/servidor
  - Servidor corre en puertos bien conocidos (DNS: 53)
  - Generalmente puerto inicial se cambia y se continua la comunicación en otro puerto, liberando el puerto conocido para otros clientes

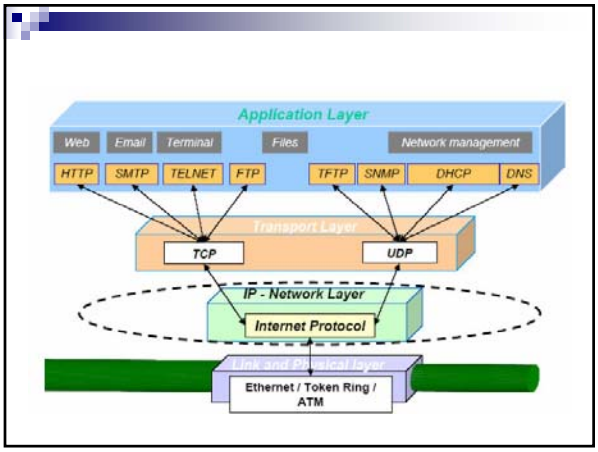
Puertos

- Abstracción
- En la práctica, hosts reciben todos los mensajes a una misma IP por una misma interfaz
- Multiplexación/demultiplexación



Tipos de Protocolos de Transporte

- Servicio de de-multiplexación asíncrono (UDP)
- Servicio de corriente de bytes confiable (TCP)



UDP

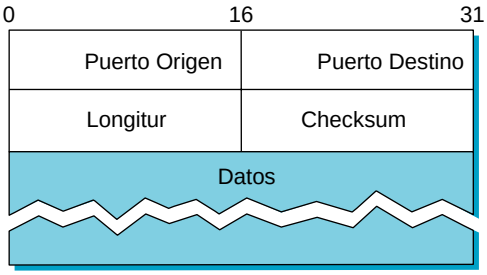
Protocolo Universal de Datagramas

- Hay varios procesos corriendo en cada host
- Añadir otro nivel de multiplexación/de multiplexación
  - Permite que procesos de varias aplicaciones en un host compartan la red
- No se añade ninguna otra funcionalidad

UDP

- Servicio de datagrama
  - No confiable
  - No ordenado
  - Sin control de flujo
  - Añade multiplexación para comunicación de proceso-a-proceso
  - Puertos identifican extremos
    - Servidores usan puertos bien conocidos

Formato Mensaje UDP



TCP

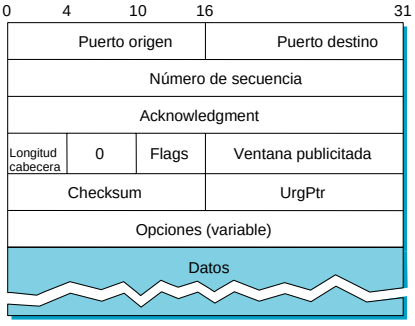
Protocolo de Control de Transmisión

- Orientado a conexión
- Confiable
- Corriente de bytes
- Full duplex
- Proporciona otro nivel de demultiplexación (puertos)
- Control de flujo
- Control de congestión

Control de Flujo y Congestión

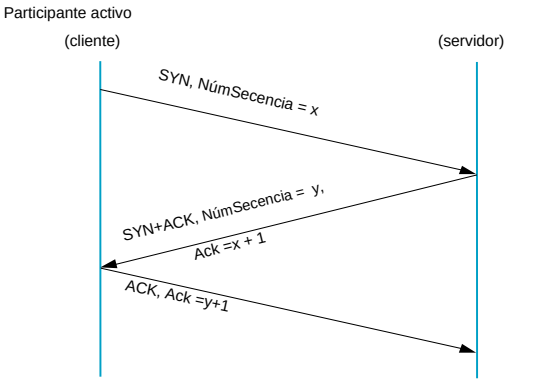
- Control de flujo
  - Evita que origen sobrecarge al destino
  - Mecanismo de ventana deslizante
- Control de congestión
  - Evita que origen sobrecarge la red
  - Mecanismos
    - Incremento aditivo/decremento multiplicativo
    - Inicio lento

Formato de la Cabecera



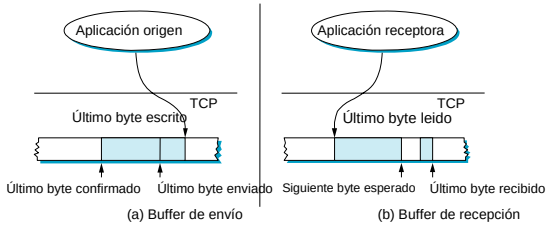
Conexión

1. Cliente pide abrir conexión con servidor
   - Cliente: inicio activo de conexión
   - Servidor: inicio pasivo de conexión
2. Intercambio de datos
3. Finalización – cada lado puede terminar la conexión



Ventana Deslizante

- Cada byte tiene un número de secuencia
- ACKs son acumulativos



Control de Flujo

- Mecanismo que permite que un proceso lento (en recibir) pueda regular velocidad de envío de el proceso que envía
  - Buffer receptor se llena
  - Ventana publicitada = 0
    - Se envía en segmentos ACK
  - Origen no puede enviar más datos
  - Bloqueo de aplicación origen cuando trata de enviar más datos

Control de Congestión

Congestión

- Informalmente: “demasiadas fuentes enviando muchos datos muy rápido para que la *red* lo pueda manejar”
- Diferente de control de flujo
- Manifestaciones
  - Paquetes perdidos (descarte de paquetes en routers)
  - Demoras largas (encolamiento en buffers de routers)

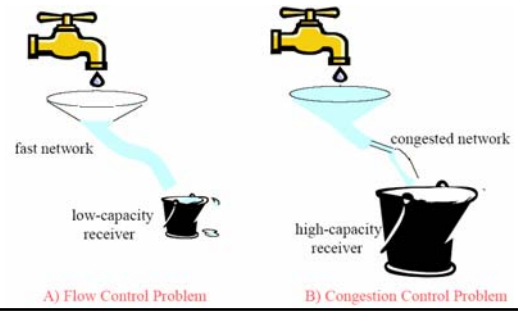
Causas

- Demoras de encolamiento en router a medida que ratio de llegada de los paquetes se acerca a la capacidad del canal
  - Costos: desperdicio de ancho de banda para re-enviar copias no necesitadas
    - Disminuye el *goodput*

Costos

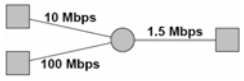
- Costos de retransmisión:
  - Routers tienen buffers finitos
    - Paquetes se descartan (drops)
  - Paquetes necesitan retransmitirse
  - Datos retransmitidos se comen el ancho de banda
  - Cuando un paquete se descarta en el camino, los routers anteriores de la ruta desperdiciaron parte de su capacidad en enviar el paquete

Control de Congestión vs. Control de Flujo



Colapso por Congestión

- Si ambas fuentes envían ventanas completas, podemos llegar a colapso por congestión
- Colapso por congestión: orígenes pierden datos debido a congestión y los re-envían, causando más congestión



Caso de Estudio: Control de Congestión de TCP

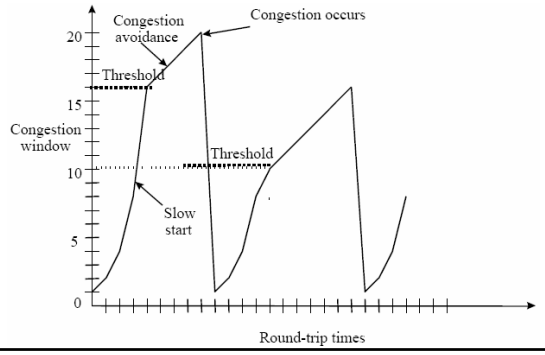
Importancia

- Desde 1988, la Internet ha permanecido funcional, a pesar de su crecimiento exponencial, de que hay routers fallosos o mal configurados, cambios rápidos en los patrones de uso de las aplicaciones y flash crowds (multitudes repentinas: miles a millones de usuarios tratando de tener acceso a un mismo sitio repentinamente)
- Esto es así principalmente porque la mayoría de las aplicaciones usan TCP y TCP implementa control de congestión de extremo-a-extremo

Control de Congestión de TCP

- Extremo-a-extremo
- Basado en ventanas
- Dos Fases
  - Inicio lento
    - Crecimiento exponencial del tamaño de la ventana (por RTT)
  - Evitar congestión
    - AIMD: Incremento Aditivo, Decremento Multiplicativo
      - Incrementar ventana en 1 por RTT
      - Si hay congestión reducir la ventana a la mitad y empezar nuevamente con el inicio lento

AIMD



Control de Congestión y UDP

TFRC

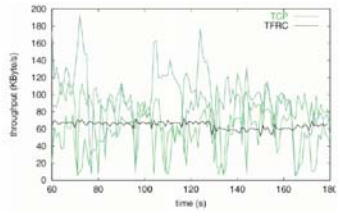
- TCP Friendly Rate Control
- Usado por RTP (Real Time Protocol)
- Así se denomina al control de congestión realizado por aplicaciones que no funcionan sobre tcp, pero que a la vez son *amigables* con TCP
  - Es decir, no se abusan del buen comportamiento de TCP

Control de Congestión de Aplicaciones

- Aplicaciones que funcionan sobre UDP deberían implementar su propio control de congestión
- Pero, generalmente no lo hacen
  - Si no adaptan sus flujos, TCP lo hará y dará paso a flujos UDP (esto es considerado un abuso)
  - Desventajas
    - Complejidad adicional
    - Rendimiento reducido
    - Calidad variable puede molestar a los usuarios

Control de Congestión de Aplicaciones

- Alternativa: diseñar mecanismos TFRC que no varíen tanto los ratios de los flujos como lo hace TCP, pero que a la vez sean amigables a TCP



RTP

- Real-time Transport Protocol
- Para aplicaciones multimedia y tipo streaming
- RTCP: RTP sobre TCP (para información de control)

Application
RTP
UDP
IP
Subnet

Caso de Estudio: DNS

Motivación

- Hasta el momento, hemos visto como dos hosts se pueden comunicar entre sí utilizando sus direcciones IP
- Pero, no siempre sabemos la IP del otro host
  - Mejor usamos nombres de dominio
    - Ej.: www.fiec.espol.edu.ec

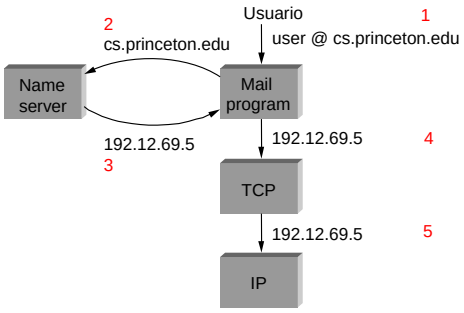
¿Por qué nombres?

- Computadoras usan direcciones (IP)
- Humanos no pueden recordar direcciones
  - Necesitan nombres
- Solución simple: cada host tiene un nombre único y una tabla de resoluciones
- Problema: no escalable
- Solución: DNS (adoptado en 1983)
- Nombres jerárquicos: nogal.fiec.espol.edu.ec

DNS

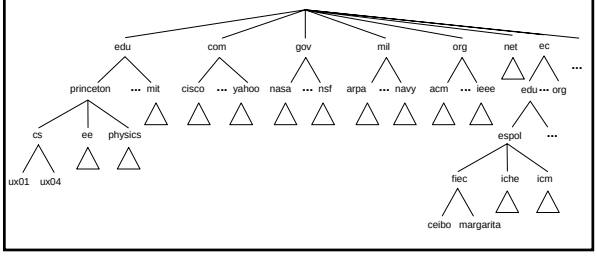
- (Domain Name System)
- Sistema de Nombres de Dominio
- No siempre se ha usado DNS en Internet
- Originalmente, eran pocos hosts, y la autoridad NIC mantenía una tabla plana con todos los pares nombre-dirección (hosts.txt)

Name Servers



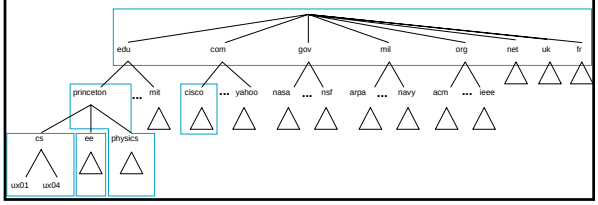
Jerarquía de Dominios

- DNS implementa un espacio de nombres jerárquico para objetos en Internet



Jerarquía Particionada en Zonas

- Cada zona puede ser vista como correspondiente a una autoridad administrativa responsable de esa parte de la jerarquía

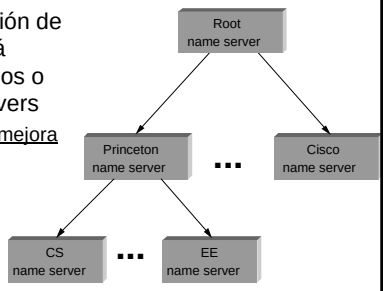


Jerarquía de Name Servers

- Name server: unidad de implementación en el DNS
- La información de cada zona está contenida en dos o más name servers
- Cada name server es un programa que puede ser accesado desde Internet
- Clientes envían consultas a name servers
- Name servers responden con IP o con puntero a otro name server

Jerarquía de Name Servers

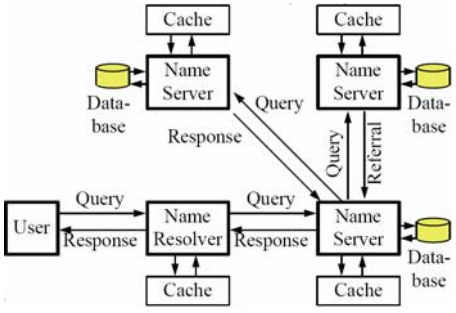
- Nota: información de cada zona está contenida en dos o más name servers
  - Redundancia mejora disponibilidad



Resolución de Nombres

- Dos elementos
  - Resolvers: en cada host
  - Servidores de nombres: servidores DNS

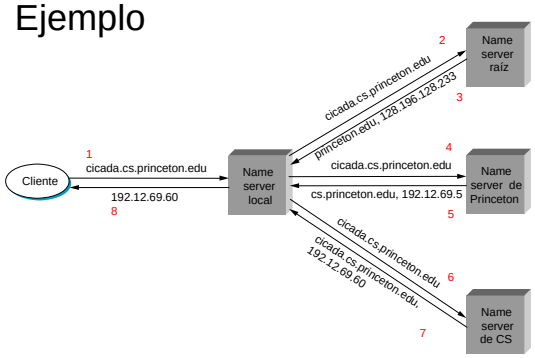
Consultas DNS



Resolución de Nombres

- Cada host tiene un resolver
  - Rutina gethostbyname en UNIX
- Cada resolver sabe quién es el servidor DNS local
- Resolver envía consulta a servidor DNS
- Servidor DNS envía consulta a servidor raíz, quien le envía una referencia al servidor DNS encargado

Ejemplo



Optimizaciones DNS

- Localidad espacial
  - Hosts locales se accesan más frecuentemente que remotos
- Localidad temporal
  - Mismo conjunto de dominios referenciado frecuentemente ➔ cacheo
- Replicación: múltiples servidores raíz; usar servidor geográficamente más cerca

Caso de Estudio: HTTP

HTTP (www)

- Versión 1.0 tenía dos graves problemas:
  - Usaba una conexión TCP para c/ objeto Web
    - Ocasionaba un servicio lento debido a la sobrecarga de establecer la conexión TCP para cada objeto, y además, TCP siempre se quedaba en su fase de "inicio lento"
  - Servidor debía conocer tamaño de objeto Web antes de poder enviarlo al cliente que lo solicitó
    - Problemas de lentitud en páginas Web dinámicas
  - Ocasionó que por un tiempo a la Web se la conozca como la World Wide Wait

NOTAS:

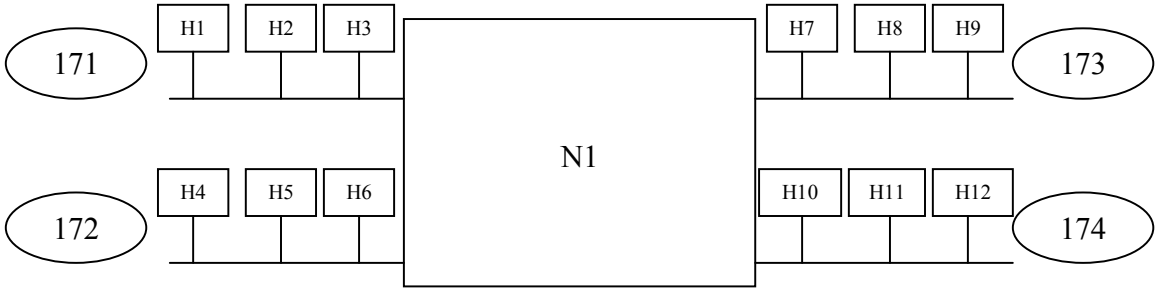
Versiones de HTTP

- Versión 1.1 solucionó problemas anteriores:
  - Conexiones persistentes
    - Cliente y servidor pueden intercambiar múltiples mensajes pedido/respuesta sobre la misma conexión TCP
    - Elimina sobrecarga de establecimiento de conexiones
    - Mecanismo de ventana de congestión de TCP funciona de manera más eficiente
  - Objetos pueden empezar a ser enviados antes de conocer su tamaño (simplemente se envía un indicador al final del archivo)

Ejercicios

1. Para cada una de las siguientes preguntas, determinar si N1 es un switch/bridge, un hub o un router (asuma que no se usan VLANs). En cada caso, el paquete puede ser visto por otros hosts a más de los especificados. Deberán indicar todas las respuestas posibles y explicar la razón por la cual llegaron a esa conclusión. Respuestas sin explicación no recibirán el puntaje completo.

   H2 IP = 192.134.171.2
   H2 MAC= ab:89:09:67:45:12
   H8 IP= 192.134.173.8
   H8 MAC= ab:89:09:67:45:ad



   a. El host H2 envía 10 paquetes a la dirección ethernet ab:89:09:67:45:ad y todos los paquetes los ven los hosts H11, H5 y H12. ¿Qué tipo de elemento de red es N1?
   b. El host H2 envía un paquete a la dirección ethernet ff:ff:ff:ff:ff:ff y lo ve el host H11. El elemento de red N1 no es un _____.
   c. El host H2 envía un paquete a la dirección Ethernet ab:89:09:67:45:ad y no lo ven los hosts H11 ni H5. El elemento de red N1 no es un _____.
   d. El host H2 quiere enviar un paquete a la dirección IP 192.134.173.8, así que envía un requerimiento ARP a la dirección IP 192.134.171.1. Usando la información en la respuesta, el host H2 usa la dirección ethernet 12:42:65:ef:89:ad como dirección ethernet destino, para enviar un paquete a 192.134.173.8. El elemento de red N1 es un _____.
   e. El host H2 envía un paquete a la dirección ethernet ab:89:09:67:45:ad y este llega a H8. El elemento de red N1 es un _____.
2. Para cada una de las siguientes aplicaciones, explique si es mejor construirlas sobre TCP o UDP. Justifique su respuesta.
   a. Voz sobre IP.
   b. Transferencia de archivos MP3 (ej.: Napster).
   c. Juego de tablero distribuido (ej.: damas chinas).
   d. Juego de acción 3D distribuido (ej.: Doom, James Bond).
   e. Media streaming (ej.: radio en Internet).
3. Control de congestión y control de flujo. Considere una red en la que para cada enlace, la capacidad del enlace es mayor que la suma de los ratios de entrada de todos los sistemas finales (end systems o hosts) en la red.
   a. ¿Se necesita control de congestión? Explique
   b. ¿Se necesita control de flujo? Explique.

Tareas

1. Yo estoy usando la computadora azalea.fiec.espol.edu.ec, y quisiera tener acceso al URL http://www.dcs.cs.dg.ac.tw. Asuma que mi servidor de nombres local es el encargado del dominio de la fiec y que espol, edu.ec son otras zonas administrativas. También asuma que tw, ac, dg y cs son zonas administrativas para el URL que estoy tratando de contactar. Además, se sabe que existe un (en realidad, más de uno) servidor DNS raíz para toda la Internet. Muestre, por medio de un diagrama, cómo puedo resolver el nombre del host al que quiero contactar. Indique en cada paso qué parte del nombre se resuelve.
2. Capas 2 y 3
    a. Considere un paquete IP transmitido en una red Ethernet. ¿Es posible que la dirección MAC destino en la trama Ethernet sea diferente de la dirección MAC del host destino del paquete IP? Explique.
    b. Un router recibe un paquete IP en una de sus interfaces Ethernet. ¿Cómo puede saber el router quién es el nodo conectado a la Ethernet que acaba de enviar el paquete?
    c. ¿Cuántas direcciones IP tiene un router?
    d. ¿Un *bridge* necesita una dirección IP? Explique.
3. ¿Qué papel juega el control de congestión en el poder proporcionar garantías de calidad de servicio (QoS)?
4. Suponga que hemos decidido re-diseñar DNS contratando los servicios de Google para que establezcan un servidor DNS centralizado bien grande y bien rápido. Todos los clientes en Internet contactarían a este servidor para mapear los nombres de hosts a sus direcciones IP. Describa las desventajas de este enfoque.

NOTAS:

Caso de Estudio: Ataque de Hombre en el Medio

Alice se encuentra utilizando el host lrcpc3 y se conecta a la máquina 'ezinfo.ethz.ch' utilizando el protocolo Telnet. Un usuario malicioso (Mallory) lee todos los mensajes que se envían en la red. El siguiente es el primer paquete capturado por Mallory:

```
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 1 arrived at 19:03:32.39
ETHER: Packet size = 60 bytes
ETHER: Destination = ff:ff:ff:ff:ff:ff
ETHER: Source = 0:0:c0:b8:c2:8d
ETHER: Ethertype = 0806
ETHER:
ARP: ----- ARP/RARP Frame -----
ARP:
ARP: Hardware type = 1
ARP: Protocol type = 0800 (IP)
ARP: Length of hardware address = 6 bytes
ARP: Length of protocol address = 4 bytes
ARP: Opcode 1 (ARP Request)
ARP: Sender's hardware address = 0:0:c0:b8:c2:8d
ARP: Sender's protocol address = 128.178.156.7, lrcpc3.epfl.ch
ARP: Target hardware address = ?
ARP: Target protocol address = 128.178.156.1, in-inr-e4.epfl.ch
```

1. ¿Cuál es el propósito de esta trama?
2. ¿Qué estaciones de trabajo reciben esta trama? ¿Qué estaciones de trabajo contestan a esta trama?
3. ¿Cómo se pudo determinar que esta trama es una trama ARP?

Posteriormente, Mallory captura las siguientes dos tramas:

```
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 2 arrived at 19:03:32.39
ETHER: Packet size = 74 bytes
ETHER: Destination = 0:0:c:2:78:36
ETHER: Source = 0:0:c0:b8:c2:8d
ETHER: Ethertype = 0800
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes
IP: Type of service = 0x00
IP: xxx. .... = 0 (precedence)
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 60 bytes
IP: Identification = 2947
IP: Flags = 0x0
IP: .0.. .... = may fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 64 seconds/hops
IP: Protocol = 17
IP: Header checksum = c2ba
IP: Source address = 128.178.156.7
IP: Destination address = 128.178.15.8, IP: No options
IP:
UDP: ----- UDP Header -----
```

```
UDP:
UDP: Source port = 1267
UDP: Destination port = 53 (DNS)
UDP: Length = 40
UDP: Checksum = B672
UDP:
DNS: ----- DNS: -----
DNS:
DNS: ""
DNS:

ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 3 arrived at 19:03:32.40
ETHER: Packet size = 202 bytes
ETHER: Destination = 0:0:c0:b8:c2:8d, Western Digital
ETHER: Source = 0:0:c:2:78:36, Cisco
ETHER: Ethertype = 0800
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes
IP: Type of service = 0x00
IP: xxx. .... = 0 (precedence)
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 188 bytes
IP: Identification = 38579
IP: Flags = 0x0
IP: .0.. .... = may fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 58 seconds/hops
IP: Protocol = 17
IP: Header checksum = 3d0a
IP: Source address = 128.178.15.8,
IP: Destination address = 128.178.156.7,
IP: No options
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 53
UDP: Destination port = 1267
UDP: Length = 168
UDP: Checksum = 0000
UDP:
DNS: ----- DNS: -----
DNS:
DNS: ""
DNS:
```

1. ¿Qué acaba de ocurrir?
2. ¿Cuál es la dirección IP de lrcpc3? ¿Y la de ezinfo.ethz.ch? ¿Cuál es la dirección IP de origen del paquete 3? ¿Cuál es la dirección MAC de origen del paquete 3?
3. ¿Para qué propósito está reservado el puerto UDP 53? ¿Y el 1267? ¿Cómo puedo saber si un paquete contiene un segmento UDP?

Referencia: Introduction to Computer Networking (SSC 4), Exercise Series, Prof. Jean-Yves Le Boudec, Ecole Polytechnique Federale de Lausanne (EPFL).

Auto-Evaluación

1. Indique al menos un protocolo que funcione en cada una de las capas del modelo OSI (excepto las capas de sesión y presentación).

Capa	Protocolo

2. Mencione dos mecanismos que son usados por DNS para proporcionar un servicio de directorios *escalable* y *eficiente*?

   a. _____

   b. _____

3. Mencione un problema de rendimiento de HTTP 1.0 e indique cómo HTTP 1.1 los soluciona.

   Problema: _____

   Solución: _____

4. Conteste verdadero (V) o falso (F):
   a. _____ Las conexiones persistentes de HTTP 1.1 aumentan el número de fases de inicio lento de TCP que se llevan a cabo en la red.
   b. _____ El protocolo IP proporciona confiabilidad salto-a-salto a través del cambo *checksum* de su cabecera.
   c. _____ La cabecera UDP no tiene un campo de número de secuencia.
   d. _____ Las respuestas de consultas DNS se cachean, lo cual, en ocasiones, puede llevar a que se direcciones incorrectamente a ciertos paquetes IP.
5. ¿Cuál de los siguientes enunciados es correcto con respecto a TCP y UDP?
   a. Utilizan un mecanismo de ventana deslizante
   b. Son protocolos confiables
   c. Son protocolos de capa de red
   d. Tienen en su cabecera un campo para el puerto de origen y uno para el puerto destino

3. Paradigmas de Programación Distribuida

Objetivo

Presentar una visión de los diferentes paradigmas de programación aplicables al desarrollo de sistemas distribuidos, profundizando en el caso particular de programación de eventos distribuidos con Java RMI.

Palabras Clave

a lo mucho una vez, semántica
al menos una vez, semántica
almacenamiento persistente de objetos
canal confiable
comunicación entre grupos
CORBA
corriente de bytes
despachador
enlazador (binder)
esqueleto
evento distribuido
exactamente una vez, semántica
fábrica de objetos, patrón de diseño tipo
falla arbitraria
falla crash
falla de omisión
hilo (thread)
interfaz remota
invocación local
invocación remota
jini
jndi
marshalling
memoria compartida
método tipo fábrica
multicast
multicast en capa de aplicación
notificación de evento
objeto distribuido
paso de mensajes
pedido-respuesta
préstamos (esquema tolerante a fallos)
proxy o stub
publicar-suscribir
puerto
recolector de basuras distribuido
referencia remota
RMI
rmiregistry
RPC
semántica de invocación
serialización
servicio de nombres
tal vez, semántica

Enlaces de Interés

- All About Sockets (Tutorial): http://msdn2.microsoft.com/en-us/webservices/aa740645.aspx
- Java RMI: http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp
- JNDI: http://java.sun.com/javase/technologies/core/jndi/index.jsp
- Jini: http://www.sun.com/software/jini/
- .NET Remoting: http://www.developer.com/net/cplus/article.php/1479761
- .NET Remoting versus Web Services: http://www.developer.com/net/net/article.php/2201701
- .NET Remoting: http://msdn2.microsoft.com/en-us/webservices/aa740645.aspx
- Java RMI and .Net Remoting Performance Comparison: http://www.cs.ru.nl/~marko/onderwijs/oss/RMI_and_Remoting.pdf

Apuntes

Paradigmas de Programación Distribuida

Secciones

- Comunicación entre procesos
- Objetos y eventos distribuidos
- Caso de estudio: Java RMI

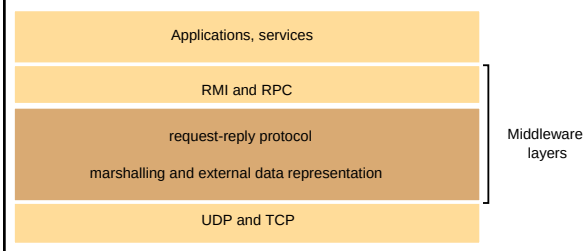
Comunicación entre Procesos

Comunicación entre procesos

- Procesos necesitan comunicarse
- Comunicación: base de sistemas distribuidos
- Alternativa 1: memoria compartida
  - No existe comúnmente
  - No sirve para sincronización
- Alternativa 2: uso de mensajes
  - Forma más común
  - Facilita abstracciones útiles para programadores

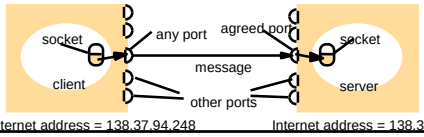
Tipos

- Abstracciones básicas
  - Paso de mensajes: Ej. UDP
  - Streams (corrientes): Ej. TCP
- Otras abstracciones
  - Comunicación cliente/servidor
    - Protocolo request-reply (pedido-respuesta)
  - Comunicación en grupos
    - Multicast



Características: Destino (mensajes)

- Direcciones: dirección IP + puerto local
- Direcciones IP no dan transparencia
- Soluciones
  - Usar nombres para identificar los servicios y/o servidores. Ej.: DNS
  - Soporte del sistema operativo (listas internas que asocian direcciones con nombres)



Características: Confiabilidad

- Validez
  - Se garantiza entrega de mensaje a pesar de que un número "razonable" de paquetes puede perderse o descartarse
- Integridad
  - Mensajes no deben ser alterados, corrompidos ni duplicados (a nivel de aplicación, no de red)

Características: Orden

- Ciertas aplicaciones requieren que los mensajes sean entregados en "orden de envío"

UDP

- Envíos sin bloqueo, recepciones con bloqueo (se pueden usar timeouts)
- Recepción no ligada a un origen en particular (receive from any)
- Modelo de fallas
  - Integridad: checksums
  - Fallas de omisión: paquetes pueden perderse o descartarse (omisión en el canal)
  - Ordenamiento: mensajes pueden ser entregados fuera de "orden de envío"

UDP (cont…)

- UDP no es confiable, por lo que es más ligero que TCP
  - No necesita almacenar información en origen ni destino
  - No incurre en transmisión extra de mensajes
  - El origen no debe esperar (los acknowledgements)
- Ejemplo de uso: DNS

TCP

- Abstracción: Corriente (stream) de bytes
- Origen-destino establecen conexión
- Comunicación en conexión no necesita indicar direcciones ni puertos
- Establecimiento de conexión es innecesario y excesivo para simples pedido-respuesta de cliente/servidor
- Procesos acuerdan tipo de información a transmitir
- Recepciones con bloqueo (si cola está vacía)
- Envíos con bloqueo (si cola de destino está llena)

TCP (cont…)

- No es realmente confiable: no garantiza entrega de mensajes (en caso de congestión extrema, se rompe la conexión)
- Modelo de fallas
  - Integridad: checksums y paquetes con no. de secuencia (rechaza paquetes fuera de orden)
  - Validez: timeouts y retransmisión para detectar y recuperarse de paquetes perdidos
- Ejemplos: HTTP, FTP, Telnet, SMTP

Datos y marshalling

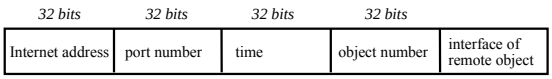
- Representación de datos varía en diferentes plataformas. ¿Cómo enviarlos?
  - Usando un estándar
  - Usar formato original + descripción del formato
- Marshalling: juntar ítems (datos) a transmitir y re-organizarlos para su transmisión
  - Usar estándares. Ej.:
    - CORBA: Common data representation
    - Java: Serialización de objetos

¿Quién se encarga del marshalling?

- ¡Middleware!
  - Evitar re-inventar la rueda
  - Evitar introducir errores
  - Mayor eficiencia
- Ejemplos:
  - CORBA y JAVA → formato binario
  - HTTP → ASCII (texto); ocupa más espacio
- Otros usos (a más de RMI y RPC):
  - transmisión de mensajes y almacenamiento en archivos

Invocación de objetos remotos

- Se necesita una referencia al objeto
  - Debe ser única, aún con el paso del tiempo
  - No deben ser re-utilizadas
  - Ejemplo:



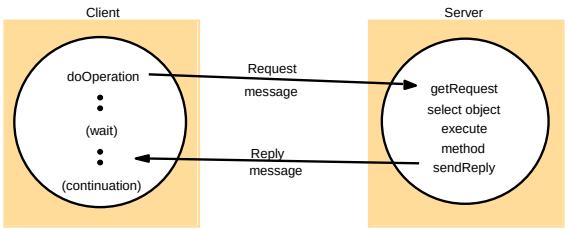
Comunicación cliente/servidor

- Protocolo pedido-respuesta (request-reply)
  - Síncrono (cliente espera respuesta del servidor)
  - Confiable (respuesta cliente ayuda al servidor)
- Implementada típicamente sobre UDP
  - Evita sobrecarga (overhead) de TCP
    - ACKs son redundantes (respuestas son suficientes)
    - Establecer conexiones no es necesario
    - Control de flujo es usualmente no necesario (mayoría de invocaciones pasan pocos datos)

Protocolo pedido-respuesta

- Primitivas de comunicación
  - public byte[] doOperation(ReferenciaAObjetoRemoto o, int IDmetodo, byte[] argumentos)
    - Envocar operación remota
    - Retorna una respuesta (reply) – cliente espera bloquado
    - Uso de timeouts por si el proceso servidor falla, pedido se pierde o respuesta se pierde
  - public byte[] getRequest()
    - Servidor obtiene pedido (request) del cliente
  - public void sendReply(byte[] respuesta, DireccionIP cliente, int puertoCliente)
    - Servidor envía respuesta al cliente

Protocolo pedido-respuesta (cont…)



Protocolo pedido-respuesta (cont…)

- Modelo de fallas (para impl. sobre UDP)
  - Fallas de omisión
  - Ordenamiento no es garantizado
  - Fallos en los procesos
    - Asume modelo "crash": proceso falla y permanece en ese estado; no se consideran fallas bizantinas
- Luego de un timeout se puede
  - Retornar con error al cliente
  - Tratar nuevamente
    - Servidor descarta pedido repetido y re-envía respuesta

Alternativas para protocolos RPC

- Request (R)
- Request-reply (RR)
- Request-reply-acknowledge reply (RRA)

Name	*Messages sent by*		
	Client	*Server*	*Client*
R	*Request*		
RR	*Request*	*Reply*	
RRA	*Request*	*Reply*	*Acknowledge reply*

Pedido-respuesta sobre TCP

- Diferencias con implementación sobre UDP, dadas por las características de TCP
  - Confiable
  - Mensajes no son duplicados (servidor no necesita mantener historial)
  - Cliente no necesita preocuparse de re-transmitir pedidos
- Reduce complejidad en protocolo p-r
- Ejemplo: Java RMI

Comunicación de grupos

- Intercambio entre dos puntos no es ideal para comunicación de grupos
- Alternativa: multicast
  - Un mensaje es dirigido a todo un grupo
  - Manejo de membresía debe ser transparente para quien envía
- Puede ser confiable, ordenado, ambos o ninguno

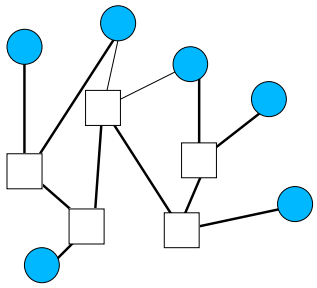
Ejemplos de uso de multicast

- Tolerancia a fallos basado en servicios replicados
- Encontrar servidores de descubrimiento en redes espontáneas (ad hoc)
- Mejorar el rendimiento usando datos replicados
- Propagación de notificaciones de eventos
- Distribución multimedia (ej.: videoconferencias)

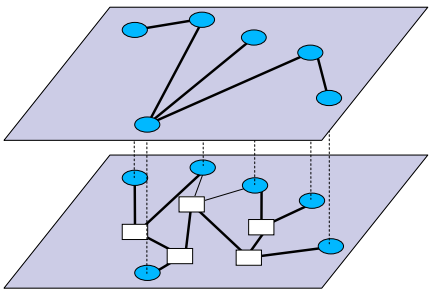
Implementaciones de comunicación de grupos

- IP multicast
- Multicast en capa de aplicaciones
  - Narada
  - ALMI
  - NICE
- Comunicación de grupos confiable
  - RMTP
  - Totem, Horus, Transis

IP Multicast



Multicast en capa de aplicación



Objetos y Eventos Distribuidos

Introducción

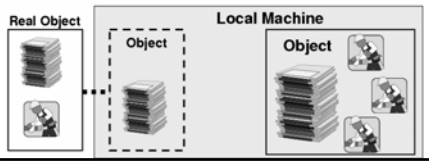
- Este capítulo estudia modelos de programación para aplicaciones distribuidas
- Modelos tradicionales han sido adaptados:
  - Llamadas a procedimientos → RPC
  - Programación Orientada a Objetos → RMI
  - Programación basada en eventos → Eventos distribuidos

Elemento Clave: Middleware

- Transparencia de
  - Ubicación: Local o remoto
  - Protocolos de comunicación: TCP o UDP
  - Hardware (Representación de datos)
  - Sistema operativo
  - Lenguaje de programación: Ej. CORBA
    - IDL: Lenguaje de Definición de Interfaces

¿Qué son objetos (distribuidos)?

- Colección bien definida de variables (datos) y la colección de funciones que trabajan sobre ellos
- Objetos distribuidos "parecen" objetos regulares, pero pueden encontrarse en otras computadoras



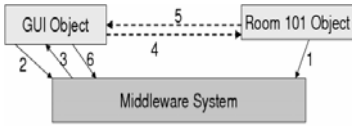
¿Por qué distribuir objetos?

- Modelo abstracto
- Tolerancia a fallos
- Escalabilidad
- Disponibilidad
- Rendimiento

Problemas con Objetos Distribuidos

- Modelo abstracto
- Rendimiento
- Latencia
- Falla parcial
- Sincronización
- Complejidad

Ejemplo: Sistema de Reservación de Habitaciones



1. Regitrar control de habitación 101
2. ¿Quién controla la habitación 101?
3. El objeto 5a62b9D
4. Objeto 5a62b9D, reservar habitación 101 de 3pm-5pm para miércoles 15 de nov de 2000
5. Si valor de retorno es TRUE, reservación culmidada con éxito
6. "Soltar" objeto 5a62b9D

Servicios Necesarios

- Nombramiento (naming)
  - ¿Dónde puedo encontrar un objeto llamado X?
- Transacciones
  - Realizar todo (X, Y, Z) o ninguno
- Detección de fallas
  - ¿Qué objeto o servicio ha *muerto*?
- Comercialización (trading)
  - ¿Dónde puedo encontrar un objeto que haga Y?

Interfaces

- Diseños modulares requieren interfaces
- Interfaces definen posibles interacciones entre módulos
- Encapsulamiento
- Uso de interfaces permite cambiar implementaciones sin cambiar interacciones

Interfaces en Sistemas Distribuidos

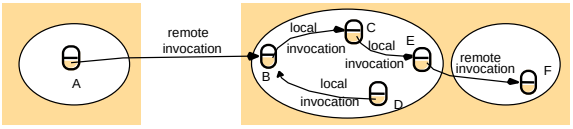
- Acceso a variables no debe ser directo
  - Ej.: Generación automática de métodos de obtención y modificación de variables
- Paso de parámetros: paso por valor o referencia no es apropiado
  - Parámetros deben ser tipo *input* o *output*
- Paso de punteros no es posible
  - Paso de referencias si lo es

Implementaciones de RMI

- CORBA – Independiente del lenguaje
- DCOM – Microsoft
- .Net Remoting - .Net
- Java RMI – Java
  - Permite paso de "comportamiento" (código de la clase puede ser pasado por valor si cliente no lo tiene)
- SOAP (Simple Object Access Protocol)
  - Usa HTTP (ya es pedido-respuesta)
  - Representación de datos: XML

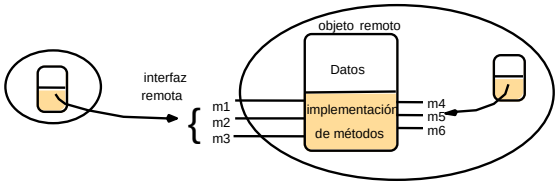
Comunicación entre objetos distribuidos

Modelo de Objetos Distribuidos



- Cada proceso contiene objetos, algunos de los cuales pueden recibir invocaciones remotas
- Objetos que pueden recibir invocaciones remotas se denominan *objetos remotos*
- Objetos necesitan saber la *referencia de objeto remoto* a un objeto en otro proceso para poder invocar sus métodos
- *Interfaz remota* especifica qué métodos pueden ser invocados remotamente

Objeto y su Interfaz Remota



Aspectos de Diseño de RMI

- Semánticas de invocación
  - Invocaciones locales se ejecutan "exactamente una vez"
  - Semánticas de invocación en sistemas distribuidos difieren según implementación de protocolo pedido-respuesta
- Transparencia
  - ¿Cuánta transparencia debe haber con respecto a invocaciones remotas?
    - Concenso: sintaxis debe ser igual a objetos locales, pero interfaces deben marcar diferencias. Ej.: métodos deben generar excepciones remotas

Semánticas de Invocación

- Determinadas por implementación de *doOperation*
  - Tratar de retransmitir mensaje: hasta que llegue respuesta o se asume servidor ha fallado
  - Filtrado de duplicados: si se usan retrans-misiones, filtrar (o no) pedidos duplicados
  - Retransmisión de resultados: se puede mantener historial de resultados para retransmitir resultados sin re-ejecutar operaciones

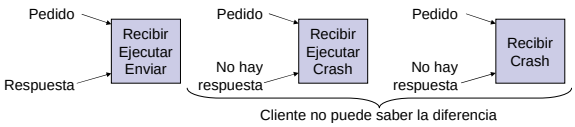
Semánticas de Invocación Remota

Medidas de Tolerancia a Fallos			*Semánticas de invocación*
Retransmitir mensaje de pedido	*Filtrado de duplicados*	*Re-ejecutar procedimiento o re-transmitir respuesta*	
No	N/A	N/A	*Tal-vez*
Si	No	Re-ejecutar proc.	*Al-menos-una-vez*
Si	Si	Re-transmitir respuesta	*A-lo-mucho-una-vez*

Semánticas de Invocación Remota

- Fallas crash (ej.: el servidor remoto falla) afectan a todas
- *Tal-vez* – si no hay respuesta, cliente no sabe si método fue ejecutado o no
  - Fallas de omisión si el pedido o respuesta se pierden
  - Usar en sistemas en que fallas en invocaciones son aceptables
- *Al-menos-una-vez* – el cliente recibe la respuesta (y el método se ejecutó al menos una vez), o una excepción (sin resultado)
  - Fallas arbitrarias. Si el pedido es retransmitido, objeto remoto puede ejecutar método más de una vez, pudiendo causar resultados incorrectos
  - Si se usan operaciones idempotentes, fallas arbitrarias no ocurren
- *A-lo-mucho-una-vez* – el cliente recibe la respuesta (y el método fue ejecutado exactamente una vez), o una excepción (en lugar de la respuesta, en cuyo caso el método se ejecutó una vez o ninguna)

Si el servidor se cuelga (crash)…

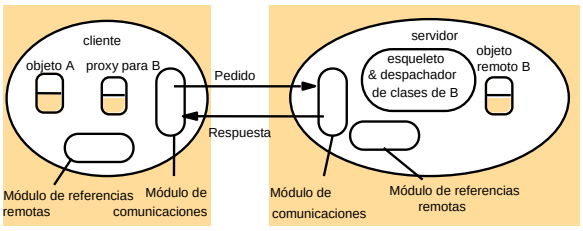


- Al-menos-una-vez: se sigue tratando hasta que se recibe respuesta del servidor
- A-lo-mucho-una-vez: no se vuelve a tratar
- Exactamente-una-vez: imposible en sist. distribuidos

Semánticas de Invocación Remota

- Diferentes implementaciones → diferentes semánticas de invocación
- Java RMI → *a-lo-mucho-una-vez*
- CORBA → *a-lo-mucho-una-vez*
  - Se puede usar *tal-vez*, bajo pedido, para métodos que no retornan resultados
- Sun RPC → *al-menos-una-vez*

Implementación de RMI



Módulo de Comunicaciones

- Los dos módulos de comunicaciones cooperan entre sí y realizan el protocolo pedido-respuesta (entre cliente y servidor)
- Proporciona la semántica de invocación

Módulo de Referencias Remotas

- Realiza conversión entre referencias locales y remotas
- Usa una tabla de objetos remotos
  - Mantiene correspondencia entre referencias a objetos locales y referencias a objetos remotos
- Llamado cuando se recibe una llamada remota, para saber a qué referencia local la referencia remota se refiere
- CORBA: Object adaptor

Software RMI

- Capa entre objetos de aplicación y módulos de comunicación y de referencias remotas
- Proxy: hace RMI transparente al cliente. Clase implementa interfaz remota. Marshalling y unmarshalling (de pedidos y resultados). Recibe pedidos y los reenvía a proceso remoto.
- Despachador: recibe pedidos de módulo de comunicaciones e invoca método en esqueleto
- Esqueleto: implementa métodos en interfaz remota. Unmarshalling y marshalling (de pedidos y resultados). Invoca método en objeto remoto.

Implementación: Otros Detalles

- Clases proxies, despachadoras y esqueletos: generadas por compilador de interfaces
- Programa servidor
  - Despachadores, esqueletos e implementaciones de clases de objetos remotos
  - Sección de inicialización: crea e inicializa al menos un objeto remoto (generalmente usado para accesar otros objetos remotos)
  - Registra uno o más obj. remotos con *enlazador*

Implementación: Otros Detalles

- Programa cliente
  - Clases proxies de objetos remotos a invocar
  - Usa enlazador para buscar referencias a objetos remotos
  - Puede crear objetos remotos usando métodos tipo *fábrica* (no hay constructores remotos)
- Enlazador (binder): servicio que mantiene una tabla con mapeos entre nombres textuales y referencias a objetos remotos
  - CORBA: Naming service
  - Java: RMI registry

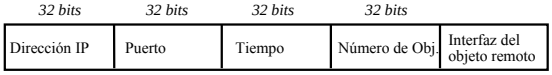
Implementación: Otros Detalles

- Hilos del servidor (threads)
  - Nuevo hilo para c/ invocación remota u objeto
  - Evitan retrasos en invocaciones concurrentes
  - Tener en cuenta efectos de concurrencia
- Para Δ eficiencia, procesos y obj. remotos pueden activarse cuando van a ser usados
  - Obj. pasivo: no en memoria. Se activa con implementación de métodos y estado (marshalled)
  - CORBA: *activador* → *repositorio de implementaciones*

Implementación: Otros Detalles

- Almacenamiento persistente de objetos
  - Objeto persistente: garantizado que existe entre activaciones de procesos
  - Bodega de objetos persistentes: almacena los objetos en disco
    - CORBA: servicio de objetos persistentes
    - Java: Java persistente
  - Dependiendo de implementación, pueden activarse en cliente (usando caché y protocolo de consistencia)
  - Optimización: almacenar únicamente objetos que han cambiado su estado

Ubicación de Objetos



- Referencia a objetos remotos debe ser única en un SD. No debe ser reutilizada si objeto ha sido eliminado.
- IP + Puerto permiten localizar objeto a menos que haya migrado o sido re-activado en un nuevo proceso
- Cuarto campo identifica a objeto dentro del proceso
- Interfaz: indica a receptor qué métodos tiene
- Pero, hay objetos que pueden cambiar de ubicación
  - Solución: usar un servicio de localización
    - Base de datos con posible ubicación de objetos

Recolección de Basura

- Objetos deben existir si referencias locales o **remotas** existen
- Objetos deben ser recolectados y memoria liberada al momento que no hay referencias a ellos
- Solución: Usar conteo de referencias
- Proxies se crean para usar referencias remotas
- Servidor debe ser informado cuando un nuevo proxy es creado y cuando uno termina

Recolector de Basura Distribuido

- Coopera con recolector de basura local
- Java
  - Tolera fallas de procesos clientes usando *préstamos*
  - Servidores *prestan* objetos (referencias) a clientes por un tiempo determinado. Clientes renuevan préstamos antes de que el tiempo expire.

Caso de Estudio: Java RMI

Transparencia

- Sintaxis igual a invocación local
  - referenciaRemota.metodo()
- Chequeo de formato de datos (igual a invocaciones locales)
  - refRem.procesaString(variableEntera) // error!

No Todo es Transparente

- Objetos son consientes de que ciertas invocaciones son remotas
  - Asegurado por el uso de excepciones remotas
- Objeto local sabe que un objeto es remoto y debe manejar *RemoteException*
- Quien implementa un objeto que va a ser usado remotamente debe extender la interfaz *Remote*

Pizarrón distribuido

- Usuarios pueden compartir una vista común de un área de dibujo donde pueden añadir objetos gráficos
- Servidor mantiene el estado actual del dibujo
- Clientes informan a servidor sobre cada figura añadida al dibujo
- Clientes puede preguntar a servidor qué otras figuras han sido añadidas
- Cada figura tiene un número de versión

Interfaces Remotas en Java RMI

- Interfaces remotas se definen al extender interfaz *Remote*, del paquete *java.rmi*
- Métodos deben arrojar *RemoteException*
- Otras excepciones particulares de la aplicación también pueden utilizarse
- *GraphicalObject*: clase que almacena el estado de un objeto gráfico
  - Debe implementar *Serializable*

Interfaces Remotas

```
public interface Shape extends Remote {
  int getVersion() throws RemoteException;
  GraphicalObject getAllState()
               throws RemoteException;
}

public interface ShapeList extends Remote {
  Shape newShape(GraphicalObject g)
       throws RemoteException;
  Vector allShapes() throws RemoteException;
  int getVersion() throws RemoteException;
}
```

Paso de Parámetros y Resultados

- Argumentos de métodos son parámetros de entrada
- Retorno de método: parámetro de salida
- Cualquier objeto *Serializable* puede ser mandado como parámetro o retornado remotamente
  - Todos los tipos primitivos de datos y objetos remotos son *Serializable*

Paso de Parámetros y Resultados

- Clases de argumentos y valores de retorno son bajadas (download) al destino por el sistema RMI, si es necesario
- Pasando objetos remotos:
  - Si tipo de un parámetro o retorno es interfaz remota, entonces se pasa la referencia remota
- Pasando objetos no remotos:
  - Todo objeto *Serializable* no remoto es copiado y pasado por valor

Paso de Parámetros y Resultados

- Serializando objetos:
  1. Objetos *Remote*: se reemplaza por su referencia remota, la cual contiene el nombre de su clase
  2. Otros objetos: Java los serializa e incluye una dirección para ubicar su clase (un URL), para que pueda ser bajada por el receptor de ser necesario

Bajando Clases

- Si el destino no contiene la clase del objeto que recibe, se baja el código de la clase automáticamente
- De igual manera, si quien recibe una referencia remota no tiene la clase proxy, se baja el código automáticamente

Ejemplo

- Supongamos que *GraphicalObject* no soporta texto
- Cliente puede crear una subclase que soporte texto y mandarla como parámetro de *newShape*
- Cuando otros clientes se enteren de la existencia de esa nueva figura, se bajarán el código de la clase automáticamente

RMIregistry

- Enlazador de Java RMI
- Una instancia de RMIRegistry debe correr en cada servidor que tenga objetos remotos
- Mantiene tabla de mapeo entre nombres (con formato parecido a un URL) a referencias a objetos remotos presentes en el servidor
  - *//nombreComputadora:puerto/nombreObjeto*

RMIregistry

- Contactado por métodos de clase *Naming*
- Servicio proporcionado por *RMIregistry* y *Naming* no es un enlazador para todo un sistema
  - Consultas de clientes deben ser dirigidas al host particular que contiene el objeto remoto
- Default: puerto 1099
- Cliente, servidor y registry pueden estar en el mismo host (*localhost*)

`void rebind (String name, Remote obj)`
Usado por un servidor para registrar el identificador a un objeto remoto en base a su nombre.

`void bind (String name, Remote obj)`
Alternativamente, utilizado por servidor para registrar un objeto remoto, pero si su nombre ya se encuentra asociado a una referencia remota, arroja una excepción.

`void unbind (String name, Remote obj)`
Elimina un enlace nombre-referencia.

`Remote lookup(String name)`
Usado por clientes para buscar un objeto remoto basado en su nombre. Retorna una referencia a un objeto remoto.

`String [] list()`
Retorna un arreglo de Strings con todos los nombres de objetos enlazados en el registro..

Programa Servidor

```
import java.rmi.*;

public class ShapeListServer
{
   public static void main(String args[])
   {
      System.setSecurityManager(new RMISecurityManager());
       try{
         ShapeList aShapeList = new ShapeListServant();
          Naming.rebind("Shape List", aShapeList );
         System.out.println("ShapeList server ready");
       }catch(Exception e) {
         System.out.println("ShapeList server main "
                            + e.getMessage());
       }
   }
}
```

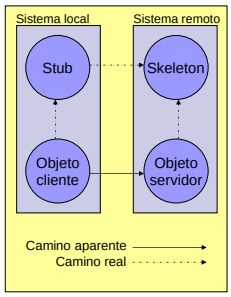
```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.util.Vector;

public class ShapeListServant
       extends UnicastRemoteObject
       implements ShapeList
{
    private Vector theList;
    private int version;
    public ShapeListServant()throws RemoteException{...}
    public Shape newShape(GraphicalObject g)
           throws RemoteException
   {
      version++;
      Shape s = new ShapeServant(g, version);
      theList.addElement(s);
      return s;
   }
   public Vector allShapes() throws RemoteException{...}
   public int getVersion() throws RemoteException { ... }
}
```

- *UnicastRemoteObject*: para objetos remotos que viven mientras vive su proceso padre
- *newShape*: método fábrica, crea objetos remotos
- *RMISecurityManager*: manejador de serguridades por defecto
- Si no se proporciona un manejador de \seguridades, proxies y otras clases solo pueden ser cargados del *CLASSPATH* local (no bajadas de otros procesos)

Programa Cliente

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.Vector;
public class ShapeListClient{
    public static void main(String args[])
   {
     System.setSecurityManager(new RMISecurityManager());
     ShapeList aShapeList = null;
     try{
        aShapeList  =
          (ShapeList) Naming.lookup("//bruno/ShapeList");
        Vector sList = aShapeList.allShapes();
     } catch(RemoteException e) {
        System.out.println(e.getMessage());
     }catch(Exception e) {
        System.out.println("Client: " + e.getMessage());}
     }
}
```

Flujo de una Llamada Remota



Más Sobre Java RMI

- Métodos estáticos (*static*) no pueden ser parte de la interfaz remota
- Campos no pueden ser miembros de interfaz remota
- Compilador de interfaces:
  - *rmic*
  - *NombreClase_Stub.class*
  - *NombreClase_Skel.class*

Cambios en Java 5.0

- Generación dinámica de clases *stub*
  - Se generan automáticamente al correr programa
  - No es necesario utilizar *rmic*
  - Si se desea interactuar con versiones previas, se debe usar *rmic* para generar *stubs*
- Aumento de sockets seguros estándar (SSL)
  - *javax.rmi.ssl.SslRMIClientSocketFactory*

Mejoras en Versiones Anteriores

- *skeleton* no son necesarios desde hace algunas versiones (1.2)
- HTTP fallback – si puertos usados por *rmi* están bloqueados, se puede utilizar HTTP tunneling
- Otras mejoras de rendimiento y seguridades

Seguridades

- RMISecurity.policy

```
grant{
  permission java.net.SocketPermission "*:1024-65535", "connect,accept";
}
```

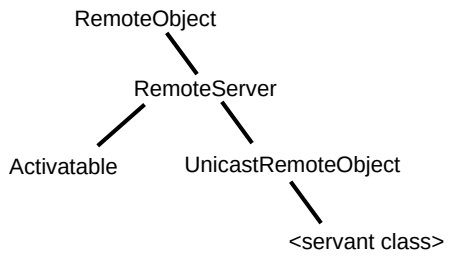
- Cargador de clases usado por RMI no se baja ninguna clase de localidades remotas a menos que haya un *SecurityManager* instalado

```
System.setProperty("java.security.policy", "RMISecurity.policy");
```

JNDI

- RMIRegistry no es escalable
  - Debe haber uno por host con objetos remotos
- JNDI: Java Naming and Directory Interface
- Es una interfaz unificada a múltiples servicios de naming y directorios
- *javax.naming*

Clases que dan Soporte a RMI



Eventos y Notificaciones

Eventos y Notificaciones Distribuidas

- Notificaciones de eventos son asíncronas
- Paradigma *publicar-suscribir* (*publish-subscribe*)
  - Objetos generan eventos
  - Objetos *publican* tipos de eventos disponibles
  - Objetos se *suscriben* a los tipos de eventos que les interesan (recibir notificaciones)

Terminología

- Notificaciones: Obj. que representan eventos
  - Pueden ser almacenadas, enviadas en mensajes, consultadas y aplicados
- Suscribirse a un tipo de evento también es llamado *registrar interés* en el evento

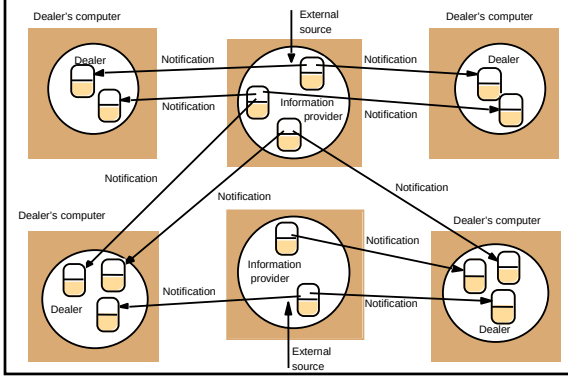
Ejemplos de Usos de Eventos

- Comunicar que:
  - Una forma se ha añadido a un dibujo
  - Un documento ha sido modificado
  - Una persona ha entrado o salido de un cuarto
    - Computación ubicua
  - Un equipo ha sido re-ubicado
  - Un libro (con un dispositivo especial) ha cambiado de ubicación

Características de Sistemas Basados en Eventos

- Heterogeneidad
  - Uso de eventos lleva a interoperabilidad de componentes que no fueron diseñados para cooperar entre sí
- Asincronía
  - Notificaciones se envían a suscriptores de manera asíncrona ➔ se evita sincronización entre publicadores y suscriptores

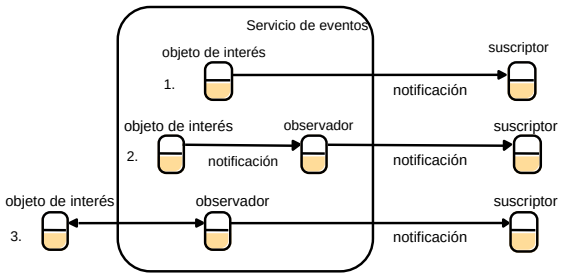
Ej.: Sistema de Mercado de Valores



Tipos de Eventos

- Eventos tienen atributos que especifican información sobre ese evento. Ej.:
  - Nombre
  - ID de objeto que lo generó
  - Operación
  - Parámetros
  - Estampa de tiempo
- En ciertos sistemas es posible suscribirse a tipos de eventos, pero siempre y cuando cumplan cierto criterio (especificado en atributos)

Participantes



Servicio de Eventos

- Componente principal
- Mantiene una base de datos de:
  - Eventos publicados
  - Intereses de suscriptores
- Eventos de un objeto de interés son publicados en el servicio de eventos
- Suscriptores le informan de los tipos de eventos en que están interesados

Roles de Objetos Participantes

- Objeto de interés: sus cambios de estado pueden ser de interés para otros objetos. Puede ser parte del servicio de eventos, si transmite notificaciones
- Evento: resultado de ejecución de un método
- Notificación: Objeto con info. sobre evento
- Suscriptor: suscrito a eventos, recibe notificaciones
- Objetos observadores: separa obj. de interés de suscriptores. Simplifica trabajo de obj. de interés
- Publicador: genera notificaciones; puede ser objeto de interés u observador

Semánticas de Entrega

- Garantías de entrega dependen de implementación y de requerimientos de aplicación
- Implementaciones con IP multicast son de *mejor esfuerzo* y no garantizan entrega
  - No apropiado para ciertas aplicaciones
- Requerimientos pueden ser de *tiempo real*
- Alternativa: multicast en capa de aplicación

Roles de Observadores

- Re-envío de notificaciones
- Filtrado de notificaciones
  - De acuerdo a atributos de eventos e interés de suscriptores
- Patrones de eventos
  - Involucra asociaciones entre eventos
- Casillas de notificaciones
  - Para entrega tardía de notificaciones

NOTAS:

Ejercicios

1. Los sistemas RMI pueden usar una implementación basada o en TCP o en UDP. ¿Cuáles son las ventajas y desventajas de usar una implementación basada en UDP?
2. Describa una manera simple y eficiente de realizar recolección de basura automática en un sistema que proporciona soporte para objetos distribuidos. ¿Es su esquema tolerante a fallos? Si la respuesta es no, indique qué se puede hacer para que sea tolerante a fallos.
3. Un cliente realiza llamadas a procedimientos remotos de un servidor. Al cliente le toma 5 milisegundos calcular los argumentos, mientras que al servidor le toma 10 milisegundos procesar cada pedido. El tiempo de procesamiento del sistema operativo local para cada operación de enviar o recibir es de 0.5 milisegundos, mientras que la demora en la red para cada mensaje transmitido es de 3 milisegundos. Las operaciones de marshalling y unmarshalling se demoran 0.5 milisegundos por mensaje. Calcule el tiempo que le toma al cliente generar y recibir el resultado de dos pedidos, asumiendo que los procesos no son multi-hilos. ¿Cambiaría su respuesta si los procesos son multi-hilos?
4. ¿Cómo puede un servidor sin estado mantener información sobre sus clientes?

Tareas

1. Describa el protocolo pedido-respuesta de HTTP e identifique su modelo de fallas.
2. Realice un cuadro comparativo de comunicaciones distribuidas usando RMI versus usando servicios Web.
3. Realice un cuadro comparativo de Java RMI versus .NET Remoting.
4. Liste los servicios que proporcionan Jini y JNDI que pueden ser utilizados para complementar los servicios proporcionados por Java RMI.
5. El siguiente programa escucha en un puerto, acepta pedidos HTTP, y luego muestra en consola la línea inicial y la cabecera del pedido.

```
import java.net.*;
import java.io.*;

public class httpSimpleServer {
    public static void main(String args[]) {
        int port = 1500;
        ServerSocket server_socket;
        try {
            _____________________________________________// linea 1
            Socket clientSocket = null;
            while(true) { // lazo infinito del servidor
                ____________________________________     // linea 2
                System.out.println("Conexión establecida.");
                BufferedReader input = new BufferedReader(new
                    InputStreamReader(clientSocket.getInputStream()));
                PrintWriter output = new
                    PrintWriter(clientSocket.getOutputStream(),true);
                String headerLine;
                while(true) {
                    ____________________________________________// linea3
                    System.out.println(headerLine);
                    if ____________________________________________// linea4
                            break;
                }
                try {                                                    //A
                    output.close();                                 //A
                    input.close();                                  //A
                    clientSocket.close();                           //A
                } catch(Exception e) {}                  //A
            } // end server loop
        } catch(Exception e) {
            System.out.println(e);
        }
    } // end main
} // end class
```

Complete el programa para que funcione correctamente.

Caso de Estudio: Sistema Distribuido de Procesamiento de Transacciones

Considere un sistema distribuido de procesamiento de transacciones entre un cliente y un servidor remoto. El cliente recibe pedidos de transacciones de usuarios locales. Estos pedidos deben ser enviados al servidor, el cual ejecutará el pedido de la transacción y retornará el resultado de la misma (por ejemplo, se puede pedir el balance de una cuenta bancaria al servidor de base de datos, y la respuesta contendrá el balance solicitado). El cliente y el servidor se comunican por un canal de comunicaciones no confiable, en el cual los mensajes se pueden perder o retrasar; el tiempo de demora máxima en el canal de comunicaciones no es conocido. Sin embargo, el canal nos asegura que los mensajes no se corrompen (no llegan con errores) ni fuera de orden.

El cliente puede recibir pedidos de usuarios locales (vía el evento *callbyuser(request)*) y retornar resultados a los usuarios (vía el evento *returndatatouser(data)*) en el orden en que los pedidos fueron generados. El servidor recibe mensajes del cliente vía el evento *messagefromclient(clientmsg)*, ejecuta las transacciones vía la llamada *result = execute(clientmsg)*, y envía mensajes al cliente vía el evento *messagetoclient(servermsg)*, donde *clientmsg* y *servermsg* son mensajes (que usted define) enviados del cliente y del servidor, respectivamente. El cliente recibe mensajes del servidor vía el evento *messagefromserver(servermsg)*.

Proporcione un diagrama de estados (FSM: máquina de estados finita) del cliente y del servidor. Describa el formato de los mensajes enviados del cliente al servidor y del servidor al cliente. Su protocolo debe ser lo más sencillo posible, en el sentido de que no deberá contener ninguna funcionalidad que no sea estrictamente necesaria para satisfacer los requerimientos descritos anteriormente.

Referencia: CMPSCI 591: Computer Networks, Fall 2000, Prof. Jim Kurose, University of Massachusetts Amherst.

NOTAS:

Auto-Evaluación

1. En Java, una clase que implemente la interfaz ____ puede ejecutarse en un hilo independiente.
   a. Runnable
   b. Scheduler
   c. Thread
   d. Serializable
2. ¿Por qué se utiliza un ServerSocket cuando programamos un servidor en Java?
   a. Para permitir que la comunicación sea bi-direccional
   b. Para que se pueda dar servicio a múltiples clientes simultáneamente
   c. Para que el servidor escuche en un puerto por defecto
   d. Para que se encargue de anunciar la presencia del servicio
3. ____ no proporciona un servicio de invocación a métodos remotos.
   a. .Net Remoting
   b. Java RMI
   c. CORBA
   d. DCE
4. ¿Cómo se llama el servicio que proporciona el RMIREGISTRY de Java?
   a. enlazador
   b. búsqueda
   c. registro
   d. publicar-suscribir
5. ¿Qué semántica de invocación remota (a objetos) no es posible obtener?
   a. exactamente-una-vez
   b. a-lo-mucho-una-vez
   c. tal-vez
   d. al-menos-una-vez
6. En Java, la palabra clave ____ se puede usar para evitar problemas de inconsistencia en estructuras de datos compartidas en sistemas multi-hilos.
   a. Serializable
   b. Synchronized
   c. Runnable
   d. Thread
7. Considere un sistema que proporciona soporte para objetos remotos. ¿Qué información se necesita almacenar en un repositorio de referencias remotas (para todo el sistema) de tal manera que se puedan realizar invocaciones remotas sobre dichas referencias?

_____

_____

_____

_____

4. Seguridades

Objetivo

Recalcar la importancia de las seguridades en los sistemas distribuidos, a través de una descripción de los diferentes problemas que se pueden presentar, análisis de vulnerabilidades de sistemas y estudio de mecanismos para la creación de canales seguros.

Palabras Clave

algoritmo criptográfico
ataque de hombre-en-el-medio
ataque de repetición
autenticación
autoridad de certificación
buffer overflow
canal seguro
certificado digital
checksum criptográfico
clave compartida
clave privada
clave pública
clave secreta
código de autenticación de mensajes (MAC)
confidencialidad
control de acceso
credencial
criptografía simétrica
CRL
DDoS
DES
Diffie-Hellman
disponibilidad
distribución de claves
encriptar
firma digital
firmar
hash criptográfico
integridad
Kerberos
MD5
negación del servicio (DoS)
no repudio
resumen de mensajes
revocación (de certificados digitales)
RSA
servidor de autenticación
SHA-1
SSL
texto cifrado
texto libre
ticket
TLS
X.509

Enlaces de Interés

- Buffer Overflow: http://en.wikipedia.org/wiki/Buffer_overflow
- Transport Layer Security: http://en.wikipedia.org/wiki/Transport_Layer_Security
- X.509: http://en.wikipedia.org/wiki/X.509
- Diffie-Hellman Key Exchange: http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange
- MD5: http://en.wikipedia.org/wiki/MD5
- SHA-1: http://en.wikipedia.org/wiki/SHA-1
- Using Java RMI with SSL: http://java.sun.com/j2se/1.5.0/docs/guide/rmi/socketfactory/SSLInfo.html
- "An Analysis of the Schemes for Detecting and Preventing ARP Cache Poisoning Attacks": http://visid.espol.edu.ec/templates/a6greyflash/publicaciones/icdcs2007.pdf

Apuntes

Seguridades

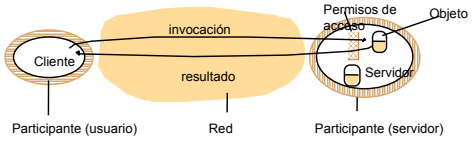
Temas a Tratar

- Modelo de seguridades
  - Tipos de amenazas
- Técnicas básicas
  - Técnicas criptográficas
    - Secretismo
    - Autenticación
    - Certificados y credenciales
    - Control de acceso
  - Rastros de auditoría
- Algoritmos de encriptación simétrica y asimétrica
- Firmas digitales
- Enfoques para un diseño de sistemas seguros
- Caso de estudio: SSL

Pilares

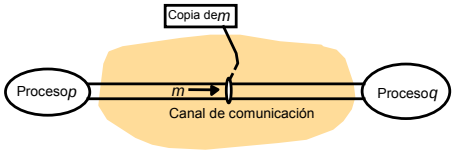
- Confidencialidad
  - Privacidad: canales seguros
  - Anonimidad: anonimizadores
- Integridad
  - Autenticación del origen: firmas digitales
  - No alteración de datos: checksums criptográficos
- Disponibilidad
  - Que el servicio/recurso esté siempre disponible: replicación y enfoques peer-to-peer

Objetos y Participantes



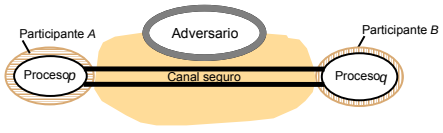
- Objeto (o recurso)
  - Mailbox, archivo del sistema o parte de un website comercial
- Participante
  - Usuario o proceso que tiene la autoridad (permisos) para realizar acciones
  - La identidad del participante es importante

Adversario



- Ataques
  - A aplicaciones que administran transacciones financieras u otro tipo de información cuya confidencialidad o integridad es importante
- Adversario
  - Puede tratar de "escuchar" la comunicación, hacerse pasar por uno de los participantes, alterar los mensajes, inundar puertos con los mensajes
- Amenazas
  - A los procesos, canales de comunicación, negación del servicio

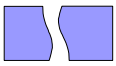
Canales Seguros



- Propiedades
  - Cada proceso está seguro de la identidad del otro
  - Los datos son confidenciales (no pueden ser leídos por terceros)
  - Los datos no pueden ser alterados por terceros
  - Existe protección contra ataques de repetición y re-ordenamiento de los datos
- Solución: usar criptografía
  - Confidencialidad
  - Autenticación

Uso de Criptografía

- Para confidencialidad
  - Confusión
  - Difusión
- Basado en secretos
  - Compartidos: claves secretas de encriptación
  - No compartidos: claves privada/pública



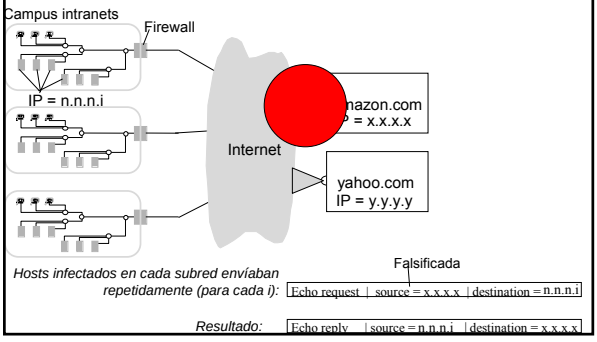
Amenazas y Formas de Ataques

- Escuchar
  - Para obtener información privada o secreta
- Enmascararse
  - Se asume la identidad de otro usuario/participante
- Alteración de mensajes
  - Alteración del contendido de los mensajes en tránsito
    - Ataque de hombre-en-el-medio
- Repeticiones
  - Almacenamiento de mensajes seguros y re-envío de los mismos posteriormente
- Negación del servicio
  - Inundación de un canal o de otro recurso, para negar el acceso a otros

Amenazas no Combatidas por los Canales Seguros

- Ataques de negación del servicio
  - Uso de recursos excesivos de tal manera que se le afecta el acceso a los mismos para los usuarios legítimos
- Caballos de Troya y otros virus
  - Defensas:
    - Autenticación del código (firmado)
    - Validación del código (pruebas formales)
    - Cajas de arena

Ej.: Ataque DDoS en Feb. 2000



Nomenclatura

K_A	Clave secreta de A
K_B	Clave secreta de B
K_{AB}	Clave secreta compartida entre A y B
K_{Apriv}	Clave privada de A
K_{Apub}	Clave pública de A
$\{M\}_K$	Mensaje *M* encriptado con la clave *K*
$[M]_K$	Mensaje *M* firmado con la clave *K*

Convención

Alice	Primer participante
Bob	Segundo participante
Carol	Participante en protocolos de tres y cuatro miembros
Dave	Participante en protocolos de cuatro miembros
Eve	Eavesdropper (que está escuchando en el canal)
Mallory	Ataque malicioso
Sara	Un servidor

Comunicación Secreta con una Clave Compartida

- Alice y Bob comparten una clave secreta K_{AB}
  1. Alice utiliza K_{AB} y una función de encriptación acordada $E(K_{AB}, M)$ para encriptar y enviar cualquier número de mensajes $\{M\}_{K_{AB}}$ a Bob
  2. Bob desencripta el mensaje con la función correspondiente $D(K_{AB}, \{M\}_{K_{AB}})$

Comunicación Secreta con una Clave Compartida

- Problemas
  - Distribución de claves
    - ¿Cómo puede Alice enviarle a Bob la clave secreta K_{AB} de manera segura?
  - Frescura de los mensajes
    - ¿Cómo puede Bob saber que cualquier $\{M_i\}$ no es una copia de un mensaje anterior de alice que fue capturado por Mallory y re-enviado posteriormente?

Comunicación Autenticada con un Servidor

Bob es un servidor de archivos, Sara es el servicio de autenticación. Sara comparte una clave secreta con Alice (K_A) y una con Bob (K_B)

1. Alice envía un mensaje (no encritpado) a Sara, indicando su identidad y pidiéndole que le otorgue un *ticket* para poder tener acceso a Bob
2. Sara envía una respuesta a Alice $\{\{Ticket\}_{K_B}, K_{AB}\}_{K_A}$. Está encriptado con K_A
3. Alice usa K_A para desencriptar la respuesta
4. Alice envía Bob un pedido P para accesar un archivo: $\{Ticket\}_{K_B}$, Alice, R
5. El ticket en realidad es $\{K_{AB}, Alice\}_{K_B}$; Bob usa K_B para desencriptarlo y verifica que el nombre de Alice concuerde y luego utiliza K_{AB} para encriptar las respuestas a Alice

Ejemplo: Kerberos
Problema: no es escalable ➔ no es apropiado para e-commerce

Comunicación Autenticada con Claves Públicas

Bob tiene un par de claves $<K_{Bpub}, K_{Bpriv}>$

1. Alice obtiene la clave pública de Bob, K^{Bpub}
2. Alice crea una nueva clave secreta compartida K_{AB}, y la encripta usando K_{Bpub}, con un algoritmo de clave pública
3. Bob usa su correspondiente clave privada K_{Bpriv} para desencriptar el mensaje y obtener K_{AB}

Si quieren estar seguros de que el mensaje no ha sido alterado, Alice puede añadir un valor acordado al mismo, para que Bob pueda verificarlo

Problema: distribución segura de la clave pública
Solución: uso de certificados digitales firmados por una autoridad de certificación

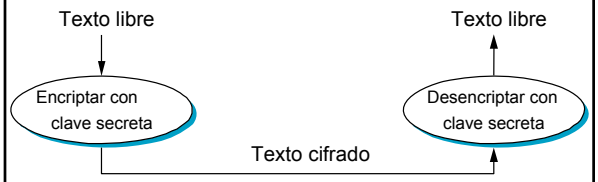
Algoritmos Criptográficos

Encriptación

- Proceso mediante el cual se convierten los datos (texto libre) en algo no interpretable por quien no tiene la clave para hacerlo (texto cifrado)
- Dos tipos
  - Simétrica
  - Asimétrica

Encriptación Simétrica

- También llamada de *clave secreta*
- A veces mal llamada: de *clave privada*

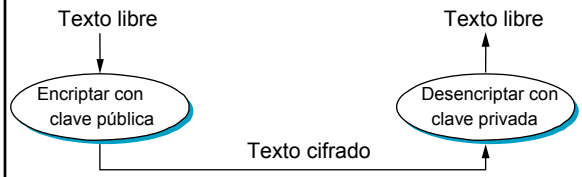


Encriptación Simétrica

- Algoritmos
  - DES: Data Encryption Standard
  - Triple DES
  - IDEA: International Data Encryption Standard

Encriptación Asimétrica

- También llamada de *clave privada/clave pública*
- Más lenta que la encriptación simétrica



Encriptación Asimétrica

- Algoritmos
  - Diffie-Hellman: para establecer secretos compartidos
  - RSA: Rivest, Shamir y Adleman
- Ventaja sobre encriptación simétrica
  - No hay problemas de distribución de claves

Resumen de Mensajes

- También llamado
  - Hash
    - Mapean un mensaje potencialmente grande en un número pequeño de tamaño fijo
  - Checksum criptográfico
    - Checksums protegen contra alteraciones en mensajes
    - Checksums criptográficos protegen contra alteraciones *maliciosas* de mensajes

Resumen de Mensajes

- Resumen (*digest*) producido por una función de *una-vía*
  - Función de una-vía: Dado un checksum criptográfico, es virtualmente imposible hallar un mensaje que produzca ese checksum
- Algoritmos
  - MD5: Message Digest versión 5
    - Muy eficiente: muy rápido de calcular
  - SHA: Secure Hash Algorithm

Firmas Digitales usando Resúmenes de Mensajes

Alice quiere publicar un documento M de tal manera que cualquiera pueda verificar que proviene de ella

1. Alice computa un resumen de un mensaje Digest(M)
2. Alice encripta el resumen con su clave privada y lo adjunta a M; el resultado es el documento firmado (M, $\{Digest(M)\}K_{Apriv}$), el cual es publicado para que lo puedan accesar otros usuarios
3. Bob obtiene el documento firmado, extrae M y calcula Digest(M)
4. Bob usa la clave pública de A para desencriptar $\{Digest(M)\}K_{Apriv}$ y lo compara con el resumen calculado. Si son iguales, se ha verificado la firma de A

Resumen de Algoritmos Criptográficos

- Simétricos (clave secreta)
  $E(K, M) = \{M\}_K$ $D(K, E(K, M)) = M$
  Ataque: fuerza bruta (tratar todas las claves posibles)
  Solución: hacer K grande
- Asimétricos (clave pública)
  Claves separadas para encriptación y desencriptación: K_e y K_d
  $D(K_d. E(K_e, M)) = M$
  "E" tiene un alto costo de computación
- Híbridos
  Utilizan criptografía asimétrica para transmitir la clave simétrica que posteriormente se utiliza para encriptar la sesión
  Ej.: SSL (TSL)

Certificados Digitales

- Certificado: un enunciado firmado por una autoridad pertinente
- Un certificado requiere:
  - Un formato pre-establecido
  - Un acuerdo de cómo formar cadenas de confianza
  - Fechas de expiración, para que puedan ser revocados

Certificados Digitales

Certificado digital del banco de Bob

1. *Typo*	:	Clave pública
2. *Nombre*		Banco de Bob
3. *Clave pública*		K_{Bpub}
4. *Autoridad de cert.*	:	Fred – La Federación de Bancos
5. *Firma*	:	$\{Digest(campo\ 2 + campo\ 3)\}_{K_{Fpriv}}$

Formato X.509

Sujeto	Nombre (o dominio) y clave pública
Emisor	Nombre y firma
Periodo de validez	No antes de y no después de
Información administrativa	Versión y número de serie
Información extendida	

Certificados como Credenciales

- Certificados sirven de credenciales
  - Evidencia para probar identidad de un participante
- Tener un certificado digital no prueba nada (son públicos)
- Tener la clave privada de la correspontiente K_{pub} del certificado, prueba la identidad de la entidad
  - Desafío respuesta

Revocación de Certificados

- Si se sospecha que la clave privada ha sido comprometida, certificado debe ser revocado
- CRL: lista de revocación de certificados
  - Generada y distribuida por la CA
  - Actualizada periódicamente
  - Disponible públicamente
  - Cuando A recibe un certificado de B, debe revisar que no haya sido revocado

Control de Acceso

- Protección de dominios
  - Conjunto de pares <recurso, permisos>
- Dos principales enfoques
  - ACLs: listas de control de acceso asociadas con cada objeto
    - Ej.: permisos de archivos en Unix
  - *Capacidades* asociadas con participantes
    - Como una llave
    - Formato: <id del recurso, operaciones permitidas, código de autenticación>
    - No deben poder ser falsificadas

Control de Acceso

```
drwxr-xr-x  gfc22  staff       264 Oct 30 16:57 Acrobat User Data
-rw-r--r--  gfc22  unknown       0 Nov  1 09:34 Eudora Folder
-rw-r--r--  gfc22  staff    163945 Oct 24 00:16 Preview of xx.pdf
drwxr-xr-x  gfc22  staff       264 Oct 31 13:09 iTunes
-rw-r--r--  gfc22  staff       325 Oct 22 22:59 list of broken
apps.rtf
```

Firmas Digitales

- Requerimientos
  - Autenticar documentos almacenados o mensajes enviados
  - Proteger contra falsificaciones
  - Prevenir que el origen niegue su participación (y niegue así su responsabilidad)

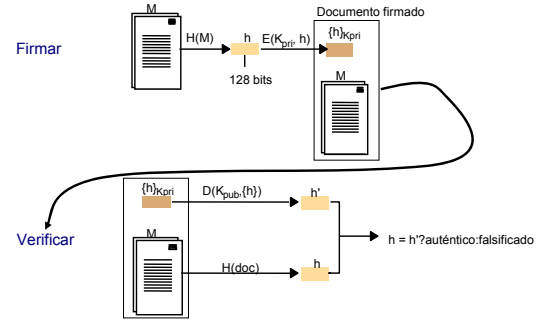
Firmas Digitales

- Encriptar un documento con una clave privada constituye una firma digital
  - Imposible que otros lo realicen sin poseer la clave
  - Autenticación del documento
  - Protección contra falsificaciones
  - Negación de participación: origen puede aludir que la clave estuvo comprometida

Firmas Digitales

- Esquema descrito en diapositiva anterior es muy caro:
  - Costo de procesamiento
  - Firma es muy grande
- Alternativa: resumen seguro
  - $E(K_{priv}, Digest(M))$
- Otra alternativa: MACs

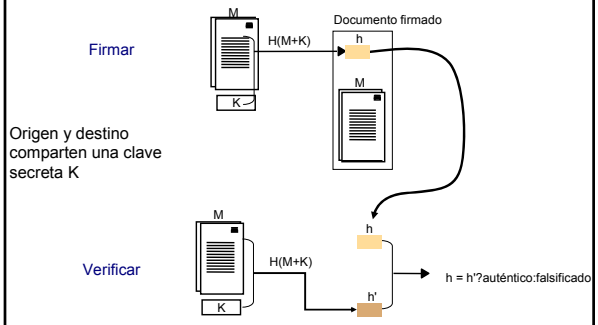
Ejemplo



MACs

- Códigos de autenticación de mensajes
- Firmas de muy bajo costo (de procesamiento y almacenamiento)
- Posible usarlas cuando se tiene un secreto compartido

MACs



SSL

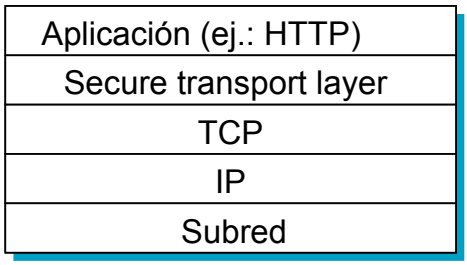
SSL: Secure Socket Layer

- Actualmente: TLS
  - Seguridad de capa de transporte
- Para transacciones Web (u otras) seguras
- Canales seguros para comercio electrónico
- Protocolo híbrido

Requerimientos de Diseño

- Comunicación segura sin negociación previa o ayuda directa de 3eros
- Opción de escoger algoritmos criptográficos
- Comunicación en ambas direcciones puede ser autenticada, encriptada o ambas

Seguridad de Capa de Transporte



TLS

- Tiene dos partes
  - Protocolo de saludo
    - Usado para negociar parámetros de la comunicación
  - Protocolo de registro
    - Usado para la transferencia (encriptada) de datos

NOTAS:

TLS

- No especifica una estructura de claves en particular
- Actualmente, una CA (VeriSign) firma los certificados
  - Clave pública de VeriSign incluida con todos los browsers

Ejercicios

1. Protocolos de seguridades:
    a. Explique cómo Bob puede utilizar criptografía simétrica para hacer que Alice demuestre que ella tiene la misma clave compartida que él. Asuma que no existe un participante externo de confianza.
    b. Explique cómo Bob puede usar criptografía asimétrica para hacer que Alice demuestre que ella está en posesión de su clave privada (es decir, que ella es Alice). Asuma que existe un repositorio (de confianza) de claves públicas (por ejemplo, Bob puede obtener la clave pública de Alice y ella puede obtener la clave pública de Bob), de manera segura.
    c. Explique cómo Bob puede usar criptografía de clave pública para dar a Alice una clave de sesión de manera que los dos puedan comunicarse de manera segura. Asuma que no hay un repositorio de claves públicas ni una autoridad externa de confianza. ¿Es su solución segura?
2. Está usted navegando en Internet y se conecta a un sitio Web que desea establecer un canal seguro. En ese momento, el navegador le muestra una ventana de alerta en pantalla indicando que ha habido algún problema con el certificado digital presentado por el sitio Web. ¿Por qué pudo haber ocurrido este problema? Nota: Existen (al menos) cinco razones por las cuales esto puede haber ocurrido.
3. ¿Qué es una clave de sesión y qué ventajas ofrece el uso de este tipo de claves?
4. Kerberos, SSH y SSL evitan los ataques de repetición. (i) ¿Qué es un ataque de repetición?, (ii) ¿Qué técnica básica es utilizada por este tipo de sistemas para prevenir ataques de repetición?

Tareas

1. Usted ha diseñado un middleware para la comunicación de grupos en el cual cualquier miembro del grupo puede enviar un mensaje y la capa de middleware se encarga de que dicho mensaje llegue a todos los demás miembros del grupo. El middleware funciona bien y está diseñado para grupos de hasta 50 participantes, pero ahora usted quiere añadir confidencialidad a los mensajes enviados al grupo. Para esto, planea diseñar un mecanismo basado en criptografía simétrica. La idea es que todos los miembros del grupo conozcan una clave secreta compartida. Cuando se desea enviar un mensaje confidencial al grupo, la capa de middleware se encargará de encriptar el mensaje utilizando la clave secreta. Cuando un participante recibe un mensaje, su contraparte de la capa de middleware se encargará de desencriptar el mensaje utilizando la clave secreta. Conteste las siguientes preguntas:
    a. ¿Proporcionará confidencialidad de los mensajes dicho esquema? ¿Proporcionará integridad de los mensajes dicho esquema? Justifique sus respuestas.
    b. ¿Qué limitaciones tiene la solución propuesta? No se olvide de tomar en cuenta que hay aplicaciones que usan grupos estáticos mientras hay otras que usan grupos dinámicos.
    c. ¿Qué sugeriría usted para sobreponerse a las limitaciones descritas en la respuesta anterior?
2. Lea el artículo “Attacks on Cryptographic Hashes in Internet Protocols” (http://www.ipa.go.jp/security/rfc/RFC4270EN.html) y haga un resumen de la problemática actual alrededor de las vulnerabilidades encontradas en MD5 y SHA-1.

Caso de Estudio: Vulnerabilidad de MD5

En agosto de 2004 unos matemáticos chinos publicaron un paper donde indican que como hallar colisiones para la función MD5; es decir, no es realmente una función de una vía. A raíz de eso, se publicaron varios papers en el 2005 donde se indica paso a paso cómo se puede crear dos certificados digitales X.509 que generen el mismo resumen MD5.

NOTA: cabe recalcar que aún no se sabe cómo, dado un certificado digital *V*, generar otro certificado *W* que tenga el mismo resumen MD5. Lo que se sabe es cómo crear dos certificados digitales que generen el mismo resumen MD5.

1. ¿Qué implicaciones tiene esto para una autoridad de certificación (CA)?
2. ¿Qué habría que hacer para montar un ataque que explote esta vulnerabilidad de MD5?
3. ¿Qué puede hacer una CA para evitar problemas con los certificados digitales que firma? (hay al menos dos maneras de evitar tener problemas).

Referencias:

- MD5, Wikipedia: http://en.wikipedia.org/wiki/MD5
- Hash Collission Q&A: http://www.cryptography.com/cnews/hash.html
- Attacks on Cryptographic Hashes in Internet Protocols: http://www.ipa.go.jp/security/rfc/RFC4270EN.html
- Colliding X.509 Certificates: http://eprint.iacr.org/2005/067
- Finding MD5 Collisions – a Toy For a Notebook: http://eprint.iacr.org/2005/075
- Tunnels in Hash Functions: MD5 Collisions Within a Minute: http://eprint.iacr.org/2006/105
- Target Collisions for MD5 and Colliding X.509 Certificates for Different Identities: http://eprint.iacr.org/2006/360

NOTAS:

Auto-Evaluación

1. Los mecanismos de autenticación de mensajes buscan asegurar la ____ de los datos.
   a. confidencialidad
   b. integridad
   c. disponibilidad
   d. anonimidad
2. El "no-repudio" en seguridad informática, pertenece a la categoría de ____ de los datos.
   a. confidencialidad
   b. integridad
   c. disponibilidad
   d. anonimidad
3. La anonimidad en seguridad informática, pertenece a la categoría de ____ de los datos.
   a. confidencialidad
   b. integridad
   c. disponibilidad
   d. autenticación
4. ____ es un formato estándar para certificados digitales.
   a. XML
   b. SHA-1
   c. MD5
   d. X.509
5. En seguridades se utiliza ____ para mejorar la disponibilidad de un servicio.
   a. encriptación
   b. replicación
   c. firmas digitales
   d. resúmenes de mensajes
6. ____ es un algoritmo de encriptación simétrica.
   a. RSA
   b. DES
   c. MD5
   d. Diffie-Hellman
7. Un ataque tipo DoS atenta contra la ____ de un sistema.
   a. disponibilidad
   b. privacidad
   c. integridad
   d. confidencialidad
8. ¿Cuál de los siguientes enunciados NO es una propiedad de un canal seguro?
   a. Los datos no pueden ser alterados por terceros
   b. Existe protección contra ataques de repetición
   c. Son susceptibles a ataques de hombre-en-el-medio
   d. Los datos son confidenciales
9. ____ es un algoritmo de resumen de mensajes.
   a. IDEA
   b. SHA-1
   c. RSA
   d. DES
10. En seguridades se utiliza ____ para conseguir confidencialidad.
    a. firmas digitales
    b. autenticación
    c. resúmenes de mensajes
    d. encriptación
11. ____ es un ejemplo de una firma digital.
    a. $E(M, K_{priv})$
    b. $E(M, K_{pub})$
    c. $E(Hash(M), K_{pub})$
    d. $E(M, K_{AB})$
12. En seguridades se utiliza ____ para autenticar el origen de los datos.
    a. firmas digitales
    b. encriptación
    c. resúmenes de mensajes
    d. replicación
13. ____ es un algoritmo de encriptación asimétrica.
    a. IDEA
    b. SHA-1
    c. RSA
    d. DES

5. Teoría de Sistemas Distribuidos

Objetivo

Describir varios problemas que se pueden presentar en algoritmos distribuidos, recalcando su importancia y las diferentes maneras de atacarlos. Más específicamente, (1) discutir la problemática alrededor de la falta de un reloj global en los sistemas distribuidos, y explicar las diferentes maneras para lidiar con este problema; y, (2) describir y analizar varios algoritmos de coordinación y acuerdo para sistemas distribuidos.

Palabras Clave

algoritmo abusivo o Bully (elecciones)
algoritmo basado en anillo (elecciones)
algoritmo basado en anillo (EMD)
algoritmo de Berkeley
algoritmo de Cristian
algoritmo de Ricart y Agrawala (EMD)
coordinador
detector de fallos
esclavo
estado
evento
exclusión muta distribuida (EMD)
falla bizantina
falla crash
GPS
inanición (starvation)
interbloqueo (deadlock)
maestro
mensaje
modelo de fallas
multicast
NTP
ocurrió-antes
orden causal
partición en la red
proceso
proceso correcto
ratio de desfase
recurso compartido
reloj global
reloj lógico
reloj lógico de Lamport
reloj lógico totalmente ordenado
reloj lógico vectorial
RTT
sección crítica
seguridad (propiedad de algoritmo)
servidor de tiempo
servidor de tokens (EMD)
sincronización externa
sincronización interna
sistema asíncrono
sistema síncrono
tiempo atómico
tiempo físico
tiempo lógico
UTC
viveza (propiedad de algoritmo)

Enlaces de Interés

- “The Byzantine Generals Problem”: http://research.microsoft.com/users/lamport/pubs/byz.pdf
- “My Writings” by Leslie Lamport: http://research.microsoft.com/users/lamport/pubs/pubs.html
- “Leslie Lamport's Logical Clocks: a tutorial” by Rob R. Hoogerwoord: http://alexandria.tue.nl/extra1/wskrap/publichtml/200210804.pdf

Apuntes

Teoría de Sistemas Distribuidos

Contenido

- Tiempo físico y tiempo lógico
  - NTP
  - Relojes lógicos
- Coordinación y acuerdo
  - Detectores de fallos
  - Exclusión mutua distribuida
  - Elecciones

Tiempo Físico y Tiempo Lógico

Planteamiento del Problema

- En el esquema de manejo de la caché del cliente NFS,
  - ¿Qué propiedades requerimos de los relojes?

Planteamiento del Problema

- Se debe poder medir el tiempo de manera precisa
  - Para saber cuándo ocurrió un evento en una máquina
  - Necesario sincronizar reloj con reloj externo confiable
- Algoritmos de sincronización de relojes, útiles para
  - Control de concurrencia basado en ordenamiento de timestamps
  - Autenticidad de pedidos, por ejemplo, en Kerberos
- No hay un reloj global en un sistema distribuido
  - Esta sección: precisión de relojes y sincronización
- Tiempo lógico es una alternativa
  - Permite ordenar eventos
  - También útil para consistencia de datos replicados

Relojes de Computadoras

- Cada computadora en un SD tiene un reloj interno
  - Usado por procesos locales para obtener hora actual
  - Procesos en diferentes computadoras pueden obtener estampas de tiempo para sus eventos
  - Relojes en diferentes computadoras pueden diferir
  - Relojes se desfasan del tiempo exacto; ratios de desfase varían Ratio de desfase del reloj: Cantidad relativa que un reloj difiere de un reloj perfecto
- Aún si los relojes en todas las computadoras en un SD se setean con el mismo tiempo, sus relojes eventualmente variaran significativamente
  - Necesario aplicar correcciones apropiadas

Relojes, Eventos y Estados

- Un SD se define como una colección P de N procesos p_i, i = 1, 2, … N
- Cada proceso p_i tiene un estado s_i que consiste de sus variables (que son transformadas conforme se ejecuta)
- Procesos se comunican únicamente via mensajes (en la red)
- Acciones de procesos:
  - Enviar, recibir y cambiar su propio estado

Relojes, Eventos y Estados

- Evento: ocurrencia de una acción que un proceso realiza mientras se ejecuta. Ej.: enviar, recibir y cambiar estado
- Eventos en un proceso p_i pueden colocarse en un orden total denotado por la relación $\rightarrow_i$ entre los eventos
  - e $\rightarrow_i$ e' si y solo si e ocurre antes de e' en p_i
- Un historial de procesos p_i es una serie de eventos ordenados por $\rightarrow_i$
  - historial(p_i) = h_i = <e_i^0, e_i^1, e_i^2, …>
- Concepto clave: *OCURRIÓ-ANTES*

Relojes

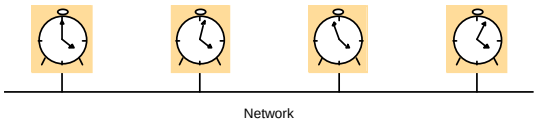
- Eventos se pueden ordenar (ocurrió-antes) en un proceso
- Para colocar estampas de tiempo en eventos, usar reloj interno
- En tiempo real t, el SO lee la hora del reloj interno $H_i(t)$
- Luego, calcula la hora en su reloj de software

  $C_i(t) = \alpha H_i(t) + \beta$
  - Ej.: un número de 64 bits que denota el número de nanosegundos desde una hora base
  - En general, el reloj no es 100% exacto
  - Pero si C_i se comporta apropiadamente, se puede utilizar para colocar estampas de tiempo en los eventos en p_i

Relojes

- Resolución del reloj: periodo entre actualizaciones del valor del reloj
- Eventos sucesivos corresponden a diferentes estampas de tiempo solo si la resolución del reloj es menor que el intervalo entre los eventos sucesivos

Variaciones de los Relojes



- Relojes generalmente marcan horas diferentes
- Desviación: Diferencia entre tiempos en 2 relojes (en cualquier instante)
- Desfase del reloj: Relojes miden el paso del tiempo en ratios diferentes
- Ratio de desfase del reloj: Diferencia por unidad de tiempo de algún reloj referencial ideal
- Relojes de quartz ordinario se desfasan en aprox. 1 segundo cada 11-12 días (10^{-6} segundos/segundo)
- Relojes de quartz de alta precisión se desfasan 10^{-7} or 10^{-8} segundos/segundo

Tiempo Universal Coordinado (UTC)

- T. atómico basado en relojes físicos de alta precisión
- UTC: estándar internacional para llevar el tiempo
- Se basa en tiempo atómico; se ajusta ocasional-mente para igualarse con el tiempo astrológico
- Se anuncia en estaciones de radio y satélites (GPS)
- Computadoras con receptores pueden sincronizar sus relojes con estas señales
- Señales de radio son precisas en 0.1-10 milisegundos
- Señales de GPS son precisas en 1 microsegundo
- O, llamando a un número de teléfono (precisión de unos cuantos milisegundos)

Sincronizando Relojes Físicos

Sincronizando Relojes Físicos

- Dos tipos:
  - Externa e Interna
- Sincronización externa
  - El reloj C_i de una máquina se sincroniza con una fuente externa confiable S, de tal manera que
  - $|S(t) - C_i(t)| < D$ para $i = 1, 2, \ldots, N$ sobre un intervalo I de tiempo real
  - Los relojes C_i son precisos dentro de límite D

Sincronizando Relojes Físicos

- Sincronización interna
  - Relojes de un par de máquinas se sincronizan entre sí
  - $|C_i(t) - C_j(t)| < D$ para $i = 1, 2, \ldots N$ sobre un intervalo I de tiempo real
  - Relojes C_i y C_j concuerdan entre sí con un límite D
- Relojes sincronizados internamente no necesariamente están sincronizados externamente (pueden desfasarse de manera colectiva)
- Si un conjunto de procesos P está sincronizado externamente con un límite D, también están sincronizados internamente con un límite $2D$

Relojes Correctos

- Se dice que un reloj de HW H es correcto si su ratio de desfase está dentro de un límite $\rho > 0$ (ej.: 10^{-6} segs/seg)
- $\Rightarrow$ error al medir el intervalo entre tiempos reales t y t' tiene un límite:
  - $(1 - \rho)(t' - t) \leq H(t') - H(t) \leq (1 + \rho)(t' - t)$ donde $t' > t$
  - Lo cual no permite saltos en las lecturas del tiempo de relojes de hardware

Relojes Correctos

- Se puede requerir que reloj de SW cumpla medida anterior
- Generalmente, basta con cumplir condición de monotonía
  - $t' > t \Rightarrow C(t') > C(t)$
  - Ejemplo, *make* en Unix requiere esta condición
  - Se puede lograr monotonía con un reloj de hw que se adelanta, al ajustar los valores α y β de $C_i(t) = \alpha H_i(t) + \beta$

Relojes Correctos

- Híbridos: monotonía dentro de puntos de sincronización
- Un reloj con fallos es uno que no es correcto (según alguna definición de correcto previamente acordada)
  - Fallas crash: reloj deja de avanzar (tics)
  - Fallas arbitrarias: culquier otro tipo de fallas; ej.: saltos en el tiempo

Ejemplo

- Un reloj con problema del año 2000
- ¿Qué tipo de falla es?
- Arbitraria
  - Quiebra condición de monotonía
  - Registra la fecha de “1 enero de 1900” después de del “31 de diciembre de 1999”
- Otro ejemplo: un reloj cuyo ratio de desfase cambia drásticamente cuando se le está acabando la pila

Sincronización en Sistemas Síncronos

- Sistemas síncronos
  - Tiempo en ejecutar cada paso tiene límite inferior y superior conocidos
  - Tiempo en transmitir mensajes tiene límites conocidos
  - Cada proceso tiene un reloj local con ratio de cambio conocido
- ¿Es la Internet síncrona?

Sincronización en Sistemas Síncronos

- Sincronización interna
  - Un proceso p_1 envía su tiempo local t a un proceso p_2, en un mensaje m
  - p_2 setea su reloj a $t + T_{trans}$ donde T_{trans} es el tiempo que toma transmitir m
  - T_{trans} es desconocido, pero $min \leq T_{trans} \leq max$
  - Incertidumbre u = max – min. Setear reloj a t + (max – min)/2, entonces desviación ≤ u/2

Sincronización en Sist. Asíncronos

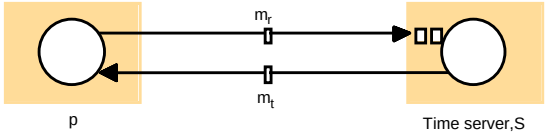
- Mayoría de los sistemas son asíncronos
  - No podemos definir un tiempo máximo que se puede tomar en transmitir un mensaje
- Ejemplo: Internet
- En un sistema asíncrono, $T_{trans} = min + x$, donde $x \geq 0$
  - x es desconocido, pero en ciertos casos puede seguir una distribución conocida

Algoritmo de Cristian (1989)

- Para sistemas asíncronos
- Algoritmo probabilístico
- *max* es desconocido, pero generalmente RTT es bastante pequeño
- Se puede sincronizar relojes con uno confiable (UTC) siempre y cuando el RTT observado es lo suficientemente pequeño, comparado con la precisión requerida

Algoritmo de Cristian (1989)

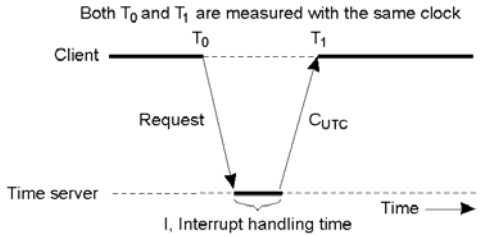
- Servidor de tiempo *S* envía señales de una fuente UTC
  - Proceso p pide tiempo en m_r y recibe t en m_t
  - p setea su reloj a $t + T_{rtt}/2$
  - Precisión ± ($T_{rtt}/2$ – min):
    - Si *S* coloca t en m_t lo más pronto, será *min* después de que p recibe m_r
    - Lo más tarde es *min* antes de que m_t llegue a p
    - Momento según S cuando m_t llega, está en rango $[t + min, t + T_{rtt} – min]$



Algoritmo de Cristian (1989)

- Discusión:
  - Asume que RTT se reparte simétricamente en enlaces de ida y de venida
    - ¿Cable modem?
  - Apropiado para redes locales o intranets
  - Un solo servidor ➔ sensible a fallos
  - Cristian con múltiples servidores
    - ¿Qué pasa si relojes fallan?
      - Alternativa: algoritmo de Berkeley
    - ¿Qué pasa si hay procesos maliciosos?
      - Usar autenticación

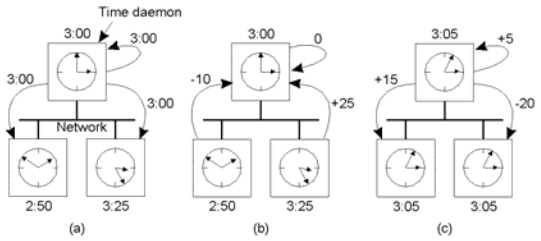
Algoritmo de Cristian (1989)



Algoritmo de Berkeley (1989)

- Algoritmo para sincronización interna de grupos
- *Maestro* pide valores de relojes del resto (*esclavos*)
- Maestro usa RTTs para estimar relojes de esclavos
- Obtiene un promedio de los relojes
  - Se eliminan valores mayores a un RTT máximo
- Tolerancia a fallos:
  - Solo se consideran relojes que no difieren más de *x* entre sí
- Maestro envía *ajuste* de tiempo a esclavos
  - No envía nuevo tiempo *t*, sino valor a sumar (o restar)
- Si maestro falla, se puede *elegir* un nuevo maestro

Algoritmo de Berkeley (1989)



NTP

- Algoritmos de Cristian y Berkeley diseñados para intranets
- NTP: Protocolo de Tiempo de Red
  - Servicio de sincronización de relojes para Internet
    - Arquitectura para el servicio
    - Protocolo de distribución
  - Sincroniza clientes a UTC

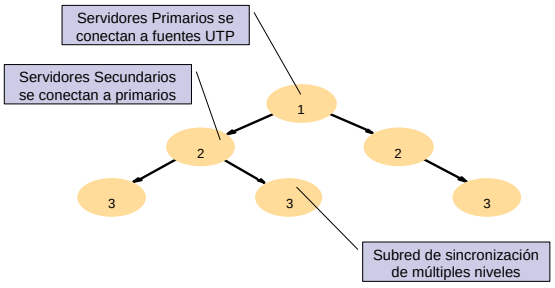
NTP: Cronología

- Varios predecesores, desde 1979
- Versión 0 (1985) a 4 (1994), por David Mills
- Desde Versión 2, basado en algoritmo de Marzullo (tesis de PhD, Stanford, 1984)
  - Mejoras fueron necesarias para eliminar jitter (variación en delay)

NTP: Metas del Diseño

- Permitir sincronizarse a UTC, en Internet
- Confiabilidad; sobreviva largas desconexiones
  - Varios servidores y varias rutas a c/u; continúan si uno falla
- Permitir a re-sincronizaciones frecuentes
  - Escalabilidad
- Proporcionar protección contra interferencias maliciosas o accidentales
  - Autenticación

NTP: Organización



NTP

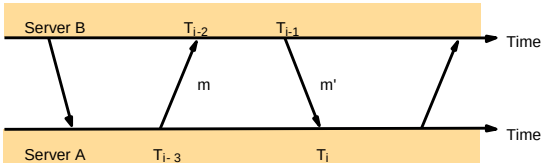
- Subred de sincronización se puede reconfigurar si ocurren fallos, como por ejemplo:
  - Un primario pierde su fuente UTC y puede convertirse en secundario
  - Si secundario pierde primario, usa otro primario
- Ejemplo de primario: ntp.nasa.gov
- Ejemplo de secundario: ntp0.cornell.edu
- Puede mantener sincronización de 10 ms en Internet (1 ms en LANs)

NTP: Modelos de Sincronización

- Multicast
  - Un servidor con una red LAN de alta velocidad envía tiempo actual a otros usando multicast, los cuales ajustan sus relojes asumiendo una demora (delay) – poca precisión
- Llamadas a procedimientos
  - Un servidor acepta pedidos de clientes – mayor precisión
  - Útil si multicast no es disponible
- Simétrico
  - Pares de servidores intercambian mensajes
  - Alta precisión (ej.: para niveles altos de la jerarquía)

NTP: Intercambio de Mensajes

- Comunicación UDP
- Cada mensaje contiene estampas de tiempo de eventos recientes
  - Tiempo local de *send* y *receive* del mensaje anterior
  - Tiempos locales del *send* del mensaje actual



Tiempo Lógico y Relojes Lógicos

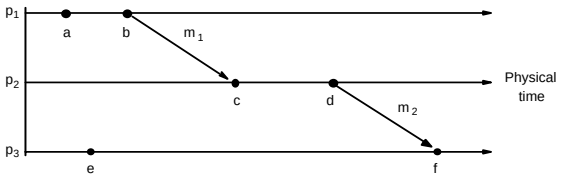
Relojes Lógicos y Orden Causal

- Eventos en un proceso pueden ordenarse por tiempo de reloj físico local
- Lamport 1978
  - Relojes no pueden sincronizarse perfectamente en un sistema distribuido
  - No se debe depender de tiempo físico para ordenar un par arbitrario de eventos del SD
  - Usar relojes lógicos
    - Relación *ocurrió-antes*

Orden Causal

- Basado en dos puntos simples:
  - Si dos eventos ocurren en el mismo proceso p_i (i = 1, 2, …, N), entonces ocurrieron en el orden en que se observaron en p_i ($\rightarrow_i$)
  - Cuando un mensaje m se envía entre dos procesos, *send*(m) ocurre antes de *receive*(m)
- Lamport: el orden *parcial* que se obtiene al combinar puntos anteriores lleva a relación *ocurrió-antes* ($\rightarrow$)
  - También llamada de *orden (potencialmente) causal*
- Además, la relación ocurrió-antes es transitiva
  - Si e $\rightarrow$ e' y e' $\rightarrow$ e", entonces e $\rightarrow$ e"

Ejemplo



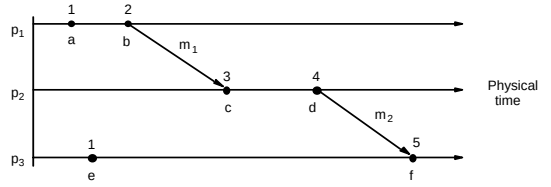
- a $\rightarrow$ b (en p1)
- c $\rightarrow$ d (en p2)
- b $\rightarrow$ c
- d $\rightarrow$ f

- Conclusión:
  - a $\rightarrow$ b $\rightarrow$ c $\rightarrow$ d $\rightarrow$ f
  - e $\rightarrow$ f
- No se puede concluir relación causal entre a y e
  - a || e

Relojes Lógicos de Lamport

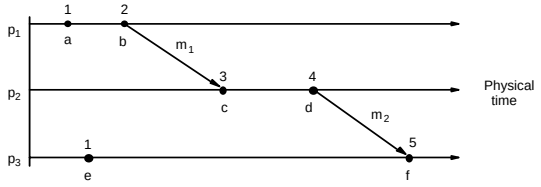
- Reloj lógico: contador de software que crece monotónicamente
  - No necesita tener relación con reloj físico
- C/ proceso p_i tiene reloj lógico L_i que se puede usar para aplicar estampas de tiempo lógicas a eventos:
  - LC1: L_i se incrementa en 1 antes de cada evento en p_i
  - LC2:
    - Cuando p_i envía mensaje m, adjunta $t = L_i$
    - Cuando p_j recibe (m,t), setea $L_i = max(L_i,t)$ y aplica LC1 antes de colocar una estampa de tiempo al evento *receive*(m)

Ejemplo



1. L_1=0, L_2=0, L_3=0
2. m_1 lleva "2" adjunto
3. Estampa de tiempo de c = *max*(0, 2) + 1 = 3
4. Estampa de tiempo de f = *max*(4, 1) + 1 = 5

Ejemplo (cont …)



- e $\rightarrow$ e' $\Rightarrow$ $L(e) < L(e')$
- Pero, lo contrario no es cierto:
  - $L(e) < L(e')$ no implica e $\rightarrow$ e'
  - Ejemplo: $L(b) > L(e)$ pero b || e

Relojes Lógicos Totalmente Ordenados

- En ocasiones necesitamos aplicar un orden global y único a todos los eventos
- Relojes Lógicos *Totalmente* Ordenados:
  - Estampa de tiempo global de un evento en un proceso p_i con estampa lógica local T_i es (T_i, i)
  - $(T_i, i) < (T_j, j)$ si y solo si $(T_i < T_j)$ o $(T_i = T_j$ e $i < j)$
- Ejemplo de uso: ordenamiento de procesos que desean entrar a *sección crítica*

Relojes Lógicos Vectoriales

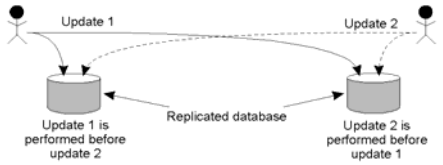
- Falencias de relojes lógicos de Lamport:
  - $L(e) < L(e')$ no implica e → e'
- Alternativa: relojes lógicos vectoriales (1989)
  - Estampas de tiempo local: V_i, un vector de N enteros (si sistema tiene N procesos)
  - Estampas de tiempo vectoriales se adjuntan a los mensajes que se envían los procesos

Relojes Lógicos Vectoriales

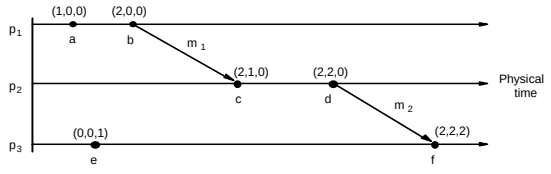
- Reglas de actualización de los relojes:
  - VC1: Inicialmente, $V_i[j] = 0$, para $j = 1, 2, \ldots, N$
  - VC2: Justo antes de que p_i estampe un evento, setea $V_i[i] = V_i[i] + 1$
  - VC3: p_i incluye valor $t = V_i$ en cada mensaje
  - VC4: Cuando p_i recibe una estampa t en un mensaje, setea $V_i[j] = max(V_i[j], t[j])$, para $j = 1, 2, \ldots, N$
    - A esta operación se la conoce como merge

Relojes Lógicos Vectoriales

- Se los usa en esquemas de replicación de datos (ej.: Coda), y en multicast causal

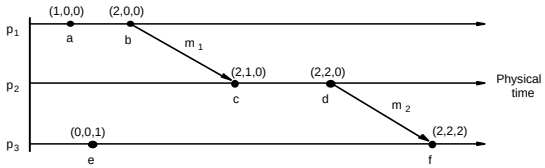


Ejemplo



- Las estampas de tiempo vectoriales se comparan así:
  - $V = V'$ si y solo si $V[j] = V'[j]$ para $j = 1, 2, \ldots, N$
  - $V \leq V'$ si y solo si $V[j] \leq V'[j]$ para $j = 1, 2, \ldots, N$
  - $V < V'$ si y solo si $V \leq V'$ y $V \neq V'$

Ejemplo



- ¿Qué eventos son concurrentes?
  - e || c, porque ni V(c) ≤ V(e), ni V(e) ≤ V(c)
  - ¿Algún otro?

Coordinación y Acuerdo

Coordinación y Acuerdo

- Exclusión mutua distribuida
- Elecciones
- Comunicación multicast
- Consenso y problemas relacionados

Coordinación Distribuida

- ¿Pueden los procesos coordinar sus acciones?
- ¿Pueden los procesos llegar a un acuerdo con respecto a un grupo compartido de valores?
- ¿Qué tipos de algoritmos podemos utilizar para resolver estos problemas?
- ¿Cómo afectan las fallas a estos algoritmos?

Importancia

- Esquemas maestro/esclavo son muy rígidos
  - Tolerancia a fallos
  - Escalabilidad
- Alternativa: grupos de procesos que coordinan sus actividades

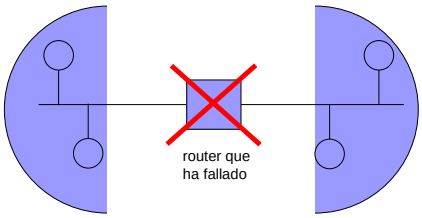
Modelo de Fallas

- Canales confiables
  - Retransmisión y redundancia
- Fallo de un proceso no afecta comunicación de otros procesos
- Pueden haber retrasos en comunicaciones
- Retrasos en comunicaciones entre procesos no son necesariamente iguales
- Fallas de proceso son tipo *crash*
  - Proceso correcto: no falla durante ejecución

Fallas en los Canales

- Pueden particionar la red
- Conectividad puede ser asimétrica
- Conectividad puede ser intransitiva
  - P – Q y Q – R, pero no P – R
- Puede que no todos los procesos puedan comunicarse al mismo tiempo
- Eventualmente cualquier enlace o router que haya fallado será reparado o podrá ser circunvalado

Una Partición en la Red



Detección de Fallos

- Detector de fallos
  - Servicio que procesa consultas sobre si un proceso en particular ha fallado
  - Usualmente implementado por un objeto local a cada proceso que corre un algoritmo de *detección de fallos* en conjunto con su contraparte en los otros procesos
    - Parte local: detector de fallos local

Detectores de Fallos

- No son necesariamente precisos
  - Mayoría son *detectores de fallos no confiables*
    - Dos valores
      - *No se sospecha*: hay razones para creer que proceso está en funcionamiento
      - *Se sospecha*: hay razones para creer que proceso ha fallado
        - Ejemplo: no se ha escuchado nada de un proceso X en determinado tiempo, en un sistema asíncrono
- *Detectores de fallos confiables*
  - Dos valores: *No se sospecha* o *ha fallado*

Detectores de Fallos

- No confiables
  - Tipo más común
- Confiables
  - Solo en sistemas síncronos
- Detectores de fallos locales pueden dar respuestas diferentes a sus procesos
  - Pueden no tener información completa

Detectores No Confiables

- Algoritmo:
  - Cada proceso *p* envía un mensaje "*p* está vivo" periódicamente a los otros procesos
  - Detector de fallos usa un estimado del tiempo máximo de transmisión de un mensaje, *D*
  - Si detector local en *q* no ha recibido "*p* está vivo" en *T* + *D* segundos, reporta a *q* que *se sospecha* de *p*
  - Si posteriormente se recibe mensaje de *p*, se reporta que *no se sospecha* de *p*

Detectores No Confiables

- Tiempo límite
  - Si es muy pequeño, se sospechará frecuentemente de los procesos
  - Si es muy grande, hay que esperar mucho para poder sospechar de un proceso
  - Valor debe reflejar las condiciones de demora observadas en la red
    - Ej.: si detector de fallos recibe un "*p* está vivo" en 20 segundos, y no en 10 como lo esperaba, puede hacer *T* = 20 segs.

Detectores Confiables

- Algoritmo anterior puede ser usado para obtener un detector confiable en un sistema síncrono
- *D* no es estimado, sino real (conocido)

Uso

- Detectores confiables
  - Pocos sistemas son síncronos
- Detectores *no confiables*
  - No son precisos
- Entonces, ¿Qué tan prácticos son los detectores de fallos?
  - Detectores no confiables, bajo ciertas propiedades bien definidas, ayudan a proporcionar soluciones prácticas a problema de coordinación de procesos en presencia de fallos

Exclusión Mutua Distribuida

Exclusión Mutua Distribuida

- Muchos procesos quieren tener acceso a un recurso compartido
- ¿Cuál es el resultado si múltiples actualizaciones se ejecutan en el recurso compartido?
  - Inconsistencia
    - Ej.: actualizaciones en bases de datos
- Comúnmente servidores administran recursos para lograr exclusión mutua

Exclusión Mutua

- Recurso
  - Debe poder ser usado exclusivamente
  - Necesidad de exclusión mutua para asegurar validez
- Sección crítica
  - Parte del procesamiento donde se usan los recursos compartidos

Problema de Sección Crítica

- Se necesita ejecutar las secciones críticas bajo exclusión mutua
- Asegura validez, ya que a lo mucho un proceso a la vez puede modificar recurso
- Problema común en sistemas operativos
  - Soluciones usan variables compartidas y/o dependencia de un *kernel* local

Problema de Sección Crítica

- En sistemas distribuidos hay 2 alternativas:
  - Un servidor administra el recurso
    - Servidor puede proporcionar mecanismos locales (ej.: usando el S.O.) para exclusión mutua
  - No hay un servidor que administre el recurso
    - Ejemplo: servidores sin estado (ej.: bloqueo de archivos en NFS), acceso a un medio compartido (ej.: ethernet, token ring)
    - Debe resolverse únicamente por paso de mensajes

Algoritmos

- Conjunto *N* de procesos p_i
  - No comparten variables
  - Accesan recursos comunes, pero solo en la sección crítica
- Asumimos que
  - Sistema es asíncrono
  - Procesos no fallan
  - Entrega de mensajes es confiable
    - Entrega eventual, exactamente una vez e íntegra

Algoritmos

- Sección crítica se protege de la siguiente manera:

```
Entrar() // se bloquea entrada
Sección crítica
Salir() // otros procesos pueden entrar ahora
```

- Requerimientos de exclusión mutua:
  - Seguridad
  - Viveza

Requerimientos

- ME1: Seguridad
  - A lo mucho un proceso puede estar en la sección crítica en cualquier momento
- ME2: Viveza
  - Requerimientos de entrar y salir de la sección crítica deben tener éxito eventualmente
- ME3: Requerimiento adicional: orden causal
  - Entradas a la sección crítica deben respetar el orden causal de los pedidos

Propiedades

- Exclusión mutua (ME1)
- No ocurren interbloqueo (deadlocks) (ME2)
- No hay inanición (starvation) (ME2)
- Equidad (ME3)

Evaluación del Rendimiento

- ¿Cómo comparar algoritmos de exclusión mutua?
  - Ancho de banda
    - Número de mensajes enviados en cada operación de *entrar* y *salir*
  - Demora para el cliente
    - Tiempo de espera en cada *entrada* y *salida*
  - Demora de sincronización
    - Tiempo entre *salida* y *entrada* del siguiente proceso

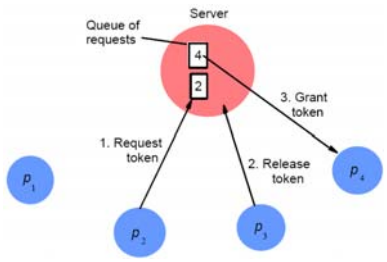
Enfoque Centralizado

- Uno de los procesos del sistema coordina las entradas a la sección crítica
- Un proceso que quiere entrar a la sección crítica envía un pedido al coordinador
- El coordinador decide qué proceso puede entrar a la sección crítica, y le envía una *respuesta* a ese proceso
- Cuando un proceso recibe un mensaje de respuesta del coordinador, entra en la sección crítica
- Al salir de la sección crítica, proceso envía un mensaje de *soltar* al coordinador

Comportamiento del Servidor

- Sin pedidos pendientes
  - Esperar por pedidos
- Con pedidos pendientes
  - Encolados en orden FIFO
  - Si el token está prestado
    - Esperar hasta recibir el token
  - Si el token no está prestado
    - Remover la cabeza de la cola y entregarle el token

Algoritmo de Servidor de Tokens



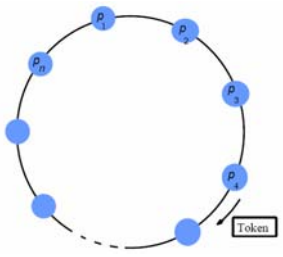
Caracterización del Algorimto de Servidor de Tokens

- ME1
  - # de procesos en SC limitado por número de tokens (1)
- ME2
  - Cola FIFO
  - No hay fallas
  - Procesos que entran a la cola serán atendidos eventualmente
- ME3
  - Servidor ignora relación →
  - Procesos atendidos en orden en que mensajes son recibidos
  - Se puede violar condición ME3
- Características
  - Entrada
  - Salida
  - Demora del cliente
  - Demora de sincronización

Algoritmo Basado en Anillo

- Procesos compañeros organizados en un anillo
- Un token circula en el anillo
  - Para entrar a sección crítica
    - Entrar cuando se recibe el token
  - Para salir de sección crítica
    - Pasar el token

Token Ring para Exclusión Mutua



Caracterización del Algoritmo de Token Ring

- ME1
  - Número de procesos en sección crítica limitado por número de tokens (=1)
- ME2
  - Topología de anillo con token circulando
  - No hay fallas
  - Procesos que quieren entrar recibirán el anillo eventualmente
- ME3
  - Ubicación del token independiente de relación →
  - Procesos atendidos en orden en que se recibe el token
  - Pueden ocurrir violaciones de ME3
- Características
  - Entrada
  - Salida
  - Demora del cliente
  - Demora de sincronización

Algoritmo de Ricart y Agrawala

- *N* procesos compañeros usando multicast y relojes lógicos
- Para entrar a sección crítica
  - Multicast el pedido
  - Entrar únicamente cuando se ha recibido respuesta de todos los procesos
  - Condiciones de respuesta diseñadas para asegurar que ME1, ME2 y ME3 se cumplan

Algoritmo

- Puntos básicos
  - Cada proceso tiene un identificador único (a ser usado en caso de empates)
  - Cada proceso mantiene un reloj lógico de Lamport
  - Mensaje de pedido tiene la forma $<T,p_i>$
  - $<S,p> < <T,q>$
    - Si $S < T$ o ($S = T$) y $p < q$

Algoritmo

- Lo que se asume
  - Los procesos no fallan
  - No hay fallos en los mensajes
- Estados de los procesos
  - LIBERADO
    - Fuera de la sección crítica
  - DESEADO
    - Desea entrar a sección crítica
  - SOSTENIENDO
    - En la sección crítica

Inicialización

```
estado := LIBERADO;
```

Para entrar a SC

```
estado := DESEADO;
Multicast pedido a todos los procesos;
T := estampa de tiempo del pedido;
Esperar hasta (# de respuestas recibido = (N – 1));
estado := SOSTENIDO;
```

Al recibir pedido $<T_i, p_i>$ en p_j ($i \neq j$)

```
if (estado = SOSTENIDO or (estado = DESEADO and
(T, p_j) < (T_i, p_i))) then
    encolar pedido de p_i sin contestar;
else
    contestar inmediatamente a p_i;
end if
```

Para salir de sección crítica

```
estado := LIBERADO;
responder a pedidos encolados;
```

Caracterización del Algoritmo

- Mensajes $<S,p>$ están totalmente ordenados
  - Considerar $<S,p>$ y $<T,q>$
  - $S = T$
    - $p < q$ entonces $<S,p> < <T,q>$
    - $p = q$ entonces $<S,p> = <T,q>$
    - $p > q$ entonces $<S,p> > <T,q>$
  - S > T entonces $<S,p> > <T,q>$

Condición ME1 se Satisface

- Prueba por contradicción
  - Supongamos que dos procesos *p* y *q* entran a la SC al mismo tiempo
    - *p* envió <S,*p*> a *q*
    - *q* envió <T,*q*> a *p*
    - *p* entró a SC
      - *q* respondió a *p*, entonces <*S*,*p*> < <*T*,*q*>
    - *q* entró a SC
      - *p* respondió a *q*, entonces <*T*,*q*> < <*S*,*p*>
    - Una contradicción *Q.E.D.*

Condición ME2 se Satisface

- Si *p* envía pedido <*S*,*p*> a *q*, todos los eventos subsequentes en *q* (después de haber recibido el mensaje) tendrán una estampa de tiempo mayor que *S*
  - Dadas las reglas de los relojes lógicos
- Si *p* envía pedido <*S*,*p*>, entonces los otros procesos pueden entrar la SC antes que *p* a lo mucho una vez antes de que le toque el turno a *p* en el ordenamiento total
- Pedidos de entrada a SC
  - Supongamos que *p* se pospone indefinidamente
  - Algún proceso *q* debe existir que no responda a *p*
  - Si *q* recibió el pedido de *p*
    - Contesta inmediatamente
    - Contesta al salir de SC
  - *q* no debe haber recibido el mensaje
    - Contradice lo que asumimos: no hay fallas

Caracterización del Algoritmo de Ricart y Agrawala

- Características
  - Entrada
  - Salida
  - Demora en el cliente
  - Demora de sincronización
- ME3 también se cumple
  - Orden →

Algoritmo de Votación de Maekawa

- Elecciones
  - ¿Cuántos votantes se necesitan para garantizar resultado?
    - Más de la mitad
  - ¿Cuántos mensajes se intercambias?
    - Número lineal
  - ¿Problema potencial?
    - Deadlock (ej.: tres candidatos y cada uno recibe 1/3 de los votos)

Reduciendo Número de Mensajes Enviados

- Ej.: 18 votantes
  - Dividir en dos grupos de 10 personas, con 2 personas en ambos grupos
  - Los 2 votantes en ambos grupos deciden quien gana
  - Si todos en tu grupo votan por ti, ganas
    - Número reducido de mensajes
    - ¿Deadlock?
    - ¿ Equidad?

Tolerancia a Fallos

- Fallos
  - Procesos
  - Canal
- Fallos afectan a cada algoritmo de manera diferente
- Para reducir impacto debemos tener información adicional
  - Ej.: detector de fallos

Elecciones

Elección

- Cómo *elegir* a un componente para que realice un rol particular
- Ejemplo: falla el reloj externo
  - ¿Qué otro componente en el sistema puede realizar la función?

Algoritmo

- P_i llama a una elección, e inicia el algoritmo
  - Un proceso solo llama una elección al mismo tiempo, pero *N* procesos pueden llamar *N* elecciones (la misma)
  - Un proceso participa o no en una elección
  - Proceso elegido necesita ser único aún si muchos procesos llaman a elección
    - Ej.: coordinador falla y dos procesos llaman a elección
- Criterios de selección del proceso
  - Orden total tiene que estar definido en el criterio de selección
    - Proceso con el mayor identificación
    - Proceso con la carga más baja

Algoritmo

- Cada P_i tiene una variable $elegido_i$ que contiene un ID del proceso elegido
  - Al inicio P_i participa en una elección, y $elegido_i = \perp$
  - Al final de la elección, $elegido_i = \text{ID}(P_{elegido})$
  - Si un proceso aún no está participando en la elección, $elegido_j = \text{ID}(P_{elegido\_anterior})$

Uso de IDs

- Decisión de qué proceso debe ganar elección debe ser única
- Alternativas
  - Prioridad
    - Basado en carga < 1/carga, *i* >
    - Basado en ancho de banda
    - Basado en alguna función específica
  - ID asignado mayor
  - Dirección IP mayor

Coordinadores

- Coordinador: proceso que realiza alguna función crítica en el sistema
  - Detección de interbloqueos (deadlocks)
  - Mantenía el token en un anillo
  - Encargado de asegurar exclusión mutua
  - Servidor de tiempo
  - …

Requerimientos

- E1: Seguridad
  - Un proceso participante P_i tiene $elegido_i = \perp$ o $elegido_i = P$, donde P es un proceso elegido que no ha fallado (crash), con el identificador más grande
- E2: Viveza
  - Todos los procesos P_i participan y eventualmente asignan $elegido_i \neq \perp$ o han fallado (crash)

Eficiencia

- Se mide por
  - Consumo total de ancho de banda de la red
    - Proporcional al número de mensajes enviados
  - Tiempo de terminación
    - Tiempos de la transmisión de mensajes serializados entre el inicio y fin de una sola ronda
      - Medido en número de mensajes

Algoritmo Basado en Anillo

- Organización
  - Colección de P_i organizados lógicamente en un anillo
  - Sistema es asíncrono
  - Meta: elegir un único proceso con el ID más grande como el coordinador
  - No tolera fallas

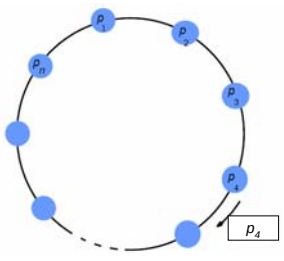
Algoritmo Basado en Anillo

- Inicialmente: cada proceso no está participando en la elección
- Cualquier proceso P_i puede iniciar la elección
  - Se marca como *participante*
  - Coloca su ID en el mensaje de elección
  - Envía mensaje a vecino (sentido del reloj)

Algoritmo Basado en Anillo

- Si proceso recibe mensaje de elección, compara longitud de ID en mensaje con propio
  - Si ID recibido > propio, pasa el mensaje y se convierte en *participante*
  - Si ID recibido < propio y no es *participante*, lo reemplaza con el propio y pasa el mensaje
  - Si ID recibido = propio, ahora es el coordinador, deja de participar y envía un mensaje de *elegido*
  - Cualquier proceso P_j que recibe un mensaje *elegido*, deja de participar y setea $elegido_j = \text{ID}(P_i)$

Algoritmo Basado en Anillo



Requerimientos

- E1: Seguridad
- E2: Viveza
- Tiempo de terminación
  - Máximo: $3N - 1$

Algoritmo Abusivo

- Nombre en inglés: Algoritmo del “Bully”
- Tolera fallas (crash) de los procesos
- Asume comunicación confiable
- Asume que c/ proceso conoce a los otros, sus IDs y se puede comunicar con ellos
- Sistemas síncronos (usa tiempos de expiración)
  - Usa detector de fallos confiable ($T = 2T_{trans} + T_{procesamiento}$)

Mensajes

- Tres tipos
  - Elección
    - Para anunciar una elección
  - Respuesta
    - En respuesta a un mensaje de elección
  - Coordinador
    - Para anunciar la identidad del coordinador elegido

Algoritmo Abusivo

- Si un proceso P_i envía un pedido que no es contestado por el coordinador dentro de un intervalo T, asume que coordinador falló y trata de elegirse a sí mismo como el nuevo coordinador
- P_i envía mensaje de *elección* a todos los otros procesos con ID mayor
- P_i espera que estos procesos *respondan* en un tiempo T

Algoritmo Abusivo

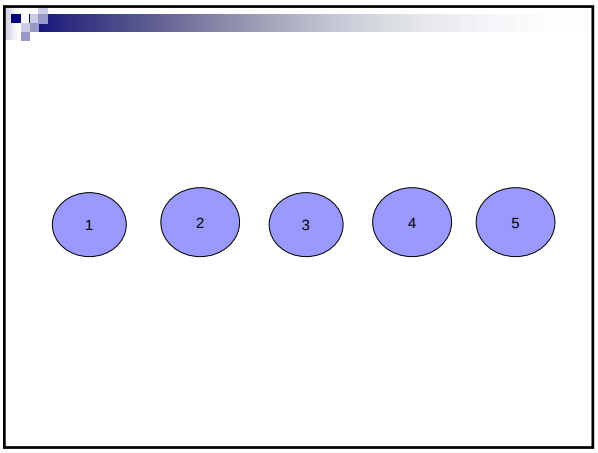
- Si no hay respuesta en dentro de T, asume que procesos con ID mayores han fallado; P_i se elige a sí mismo como el nuevo coordinador
- Si se recibe una respuesta, P_i inicia intervalo T', y espera recibir mensaje que que proceso con ID mayor ha sido elegido
- Si no llegan mensaje dentro de T', asume que el proceso con ID mayor ha fallado e re-inicia algoritmo

Algoritmo Abusivo

- Si P_i no es el coordinador, en cualquier momento durante la ejecución, P_i recibirá uno de los dos siguientes mensajes de P_j:
  - P_j es el nuevo coordinador ($j > i$); entonces, P_j registra esta información
  - P_j ha iniciado una elección ($i > j$); entonces, P_i envía un mensaje a P_j e inicia su propio algoritmo de elección
- Después de que un proceso que ha fallado se recupera, inmediatamente inicia algoritmo
- Si no hay procesos activos con ID mayor, proceso recuperado forza todos que le permitan ser el nuevo coordinador, aún cuando ya haya un nuevo coordinador

Algoritmo Abusivo

- Si un proceso ya sabe que tiene el ID mayor, se salta la elección y se anuncia directamente como *coordinador*



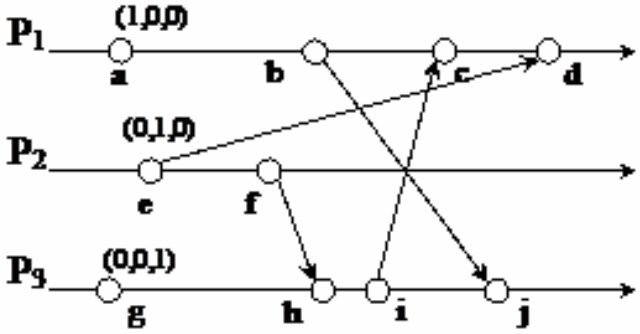
Requerimientos

- E1: Seguridad
  - Solo puede haber un coordinador a la vez (ID mayor)
  - No se cumple si *T* está mal estimado
    - Detector de fallos no confiable
- E2: Viveza
  - Si, porque canales son confiables

NOTAS:

Ejercicios

1. Este diagrama muestra una serie de eventos que ocurren en varios procesos. El mensaje de P_2 a P_1 se demora más de lo normal en llegar a su destino, de tal manera que el mensaje de P_3 llega primero.
    a. Asigne estampas vectoriales a los siete eventos que faltan de asignar en la tabla.



a	(1,0,0)		f	
b			g	(0,0,1)
c			h	
d			i	
e	(0,1,0)		j	

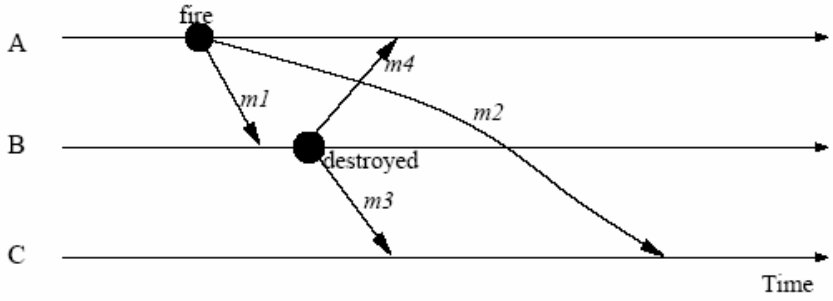
    b. Basado en las estampas de tiempo vectoriales, ¿qué eventos son concurrentes con el evento i?
2. Usted ofrece un servicio de red a hosts arbitrarios en Internet, pero el servicio solamente puede ser accesado por un proceso a la vez. ¿Por qué razón el algoritmo de exclusión mutua basado en anillo no funcionará para esta aplicación?
3. Bajo qué circunstancias el algoritmo de exclusión mutua centralizado (servidor de tokens) resulta en más mensajes que el de Ricart y Agrawala? Asuma que un solo proceso desea entrar a la sección crítica y los canales son confiables.
4. En el algoritmo abusivo (bully) de elección de un coordinador, explique qué sucede si dos procesos independientemente descubren que el coordinador no está respondiendo. Justifique su respuesta.
5. Suponga que hay N procesos en un grupo, numerados 1..N. El proceso *i* detecta que el coordinador del grupo no está respondiendo e inicia una elección. Asuma que los canales son confiables y cuenten también los mensajes que se envían a un proceso muerto.
    c. ¿Cuántos mensajes se envían si se usa el algoritmo abusivo?
    d. ¿Cuántos mensajes se envían si se usa el algoritmo del anillo? Por facilidad, para el caso del anillo, asuma que los procesos están usando un detector de fallos y el anillo ha sido reconfigurado para que se "salte" al proceso que ha fallado. Una vez reconfigurado el anillo, empieza la elección.
    e. Suponga que N = 10. ¿Bajo qué circunstancias el número de mensajes enviados usando el algoritmo abusivo será menor que el número enviado usando el algoritmo del anillo?
6. Asuma un sistema asíncrono en una intranet. Un cliente se va a sincronizar con un servidor de tiempo. El cliente registra tiempos de ida y vuelta (RTT) y las estampas de tiempo retornadas por el servidor en la siguiente tabla.

RTT (en milisegundos)	Hora (hh:mm:ss.ms)
22	10:54:23.674
25	10:54:25.450
20	10:54:28.342

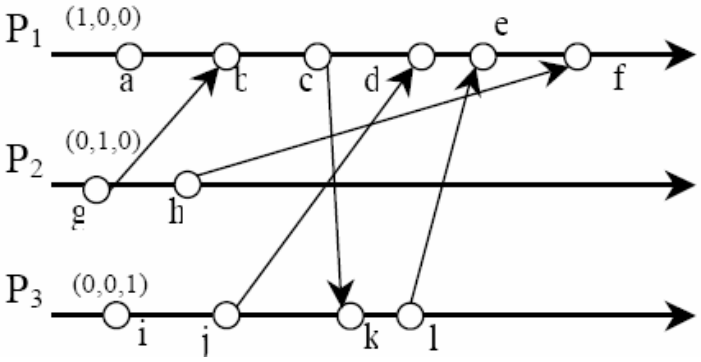
    f. ¿Cuál de estos tiempos debe usar el cliente para sincronizar su reloj? ¿A qué hora pondrá (seteará) el reloj el cliente? Estimar la precisión del seteo con respecto al reloj del servidor.
    g. Si se sabe que el tiempo entre que se envía y se recibe un mensaje en el sistema es al menos 8 milisegundos, ¿cambia eso su respuesta? Explique.

Tareas

1. Imagine una implementación distribuida de un juego de guerra en el que tanques se disparan los unos a los otros. Cada participante usa una computadora separada que se comunica a través de la red. Los eventos locales se envían a todos los miembros (multicast). La implementación es tal que cuando un jugador A dispara a B, A hace multicast del evento *fire* (fuego! o disparo!). El jugador B al recibir el evento *FIRE* evalúa el daño y de ser necesario, hace multicast del evento *destroyed* (destruido!). Esta implementación puede resultar en una anomalía temporal debido a demoras en la red. Como se ilustra en la figura, el jugador *C* puede observar el daño en B antes de saber que A a disparado a B. Diseñe una solución para evitar esta anomalía temporal basado en el uso de estampas de tiempo vectoriales. La implementación debe observar las estampas de tiempo en los mensajes y procesar el evento (para mostrar en la pantalla el disparo o la destrucción de un tanque) en orden causal. Especifique cómo se incrementan las estampas de tiempo en cada evento y cómo un jugador decide cuándo procesar un evento (es decir mostrarlo en pantalla).
   PISTAS: (1) recuerde que en el algoritmo de estampas de tiempo, se incrementa el valor la estampa cuando se envían mensajes (indicados con círculos en la figura) y también cuando se reciben los mensajes (indicados con el final de la flecha en la figura). (2) su solución puede usar una cola o buffer en cada PC para almacenar mensajes recibidos pero que no deben ser procesados aún.



2. Sugiera cómo adaptar el algoritmo abusivo (Bully) para que pueda lidiar con particiones en la red.
3. Asigne estampas de tiempo vectoriales a los siguientes eventos:



a	(1,0,0)	g	(0,1,0)
b		h	
c		i	(0,0,1)
d		j	
e		k	
f		l	

   - Analice las estampas vectoriales que usted colocó en la tabla. ¿Qué eventos son concurrentes con el evento h?
   - ¿Por qué razón las estampas vectoriales representan una mejora sobre las estampas de Lamport?

Caso de Estudio: Leslie Lamport

Leslie Lamport, uno de los padres de la teoría de sistemas distribuidos, actualmente trabaja para el centro de investigación Microsoft Research. En su página personal, Lamport detalla la historia de todas sus publicaciones. Lea los siguientes extractos de "My Writings" de Leslie Lamport (http://research.microsoft.com/users/lamport/pubs/pubs.html) y conteste las preguntas listadas a continuación:

This document contains descriptions of almost all my technical papers and electronic versions of many of them for downloading. Each description attempts to explain the genesis of the work.

11. A New Solution of Dijkstra's Concurrent Programming Problem
Communications of the ACM 17, 8 (August 1974), 453-455.

This paper describes the bakery algorithm for implementing mutual exclusion. I have invented many concurrent algorithms. I feel that I did not invent the bakery algorithm, I discovered it. Like all shared-memory synchronization algorithms, the bakery algorithm requires that one process be able to read a word of memory while another process is writing it. (Each memory location is written by only one process, so concurrent writing never occurs.) Unlike any previous algorithm, and almost all subsequent algorithms, the bakery algorithm works regardless of what value is obtained by a read that overlaps a write. If the write changes the value from 0 to 1, a concurrent read could obtain the value 7456 (assuming that 7456 is a value that could be in the memory location). The algorithm still works. I didn't try to devise an algorithm with this property. I discovered that the bakery algorithm had this property after writing a proof of its correctness and noticing that the proof did not depend on what value is returned by a read that overlaps a write.

I don't know how many people realize how remarkable this algorithm is. Perhaps the person who realized it better than anyone is Anatol Holt, a former colleague at Massachusetts Computer Associates. When I showed him the algorithm and its proof and pointed out its amazing property, he was shocked. He refused to believe it could be true. He could find nothing wrong with my proof, but he was certain there must be a flaw. He left that night determined to find it. I don't know when he finally reconciled himself to the algorithm's correctness.

Several books have included emasculated versions of the algorithm in which reading and writing are atomic operations, and called those versions "the bakery algorithm". I find that deplorable. There's nothing wrong with publishing a simplified version, as long as it's called a simplified version.

What is significant about the bakery algorithm is that it implements mutual exclusion without relying on any lower-level mutual exclusion. Assuming that reads and writes of a memory location are atomic actions, as previous mutual exclusion algorithms had done, is tantamount to assuming mutually exclusive access to the location. So a mutual exclusion algorithm that assumes atomics reads and writes is assuming lower-level mutual exclusion. Such an algorithm cannot really be said to solve the mutual exclusion problem. Before the bakery algorithm, people believed that the mutual exclusion problem was unsolvable--that you could implement mutual exclusion only by using lower-level mutual exclusion. Brinch Hansen said exactly this in a 1972 paper. Many people apparently still believe it. (See [90].)

The paper itself does not state that it is a "true" mutual exclusion algorithm. This suggests that I didn't realize the full significance of the algorithm until later. But I don't remember.

For a couple of years after my discovery of the bakery algorithm, everything I learned about concurrency came from studying it. Papers like [24], [32], and [69] were direct results of that study. The bakery algorithm was also where I introduced the idea of variables belonging to a process--that is, variables that could be read by multiple processes, but written by only a single process. I was aware from the beginning that such algorithms had simple distributed implementations, where the variable resides at the owning process, and other processes read it by sending messages to the owner. Thus, the bakery algorithm marked the beginning of my study of distributed algorithms.

The paper contains one small but significant error. In a footnote, it claims that we can consider reads and writes of a single bit to be atomic. It argues that a read overlapping a write must get one of the two possible values; if it gets the old value, we can consider the read to have preceded the write, otherwise to have followed it. It was only later, with the work eventually described in [69], that I realized the fallacy in this reasoning.

24. On Concurrent Reading and Writing
Communications of the ACM 20, 11 (November 1977), 806-811.

This paper came out of my study of the bakery algorithm of [11]. The problem with that algorithm is that it requires unbounded state. To allow the state to be bounded in practice, I needed an algorithm for reading and writing multidigit numbers, one digit at a time, so that a read does not obtain too large a value if it overlaps a write. This paper shows that this can be done by simply writing the digits in one direction and reading them in the other. It also has some other nice algorithms. The paper assumes that reading and writing a single digit are atomic operations. The original version introduced the notion of a regular register and proved the results under the weaker assumption that the individual digits were regular. However, the editor found the idea of nonatomic reads and writes to individual digits too heretical for CACM readers, and he insisted that I make the stronger assumption of atomicity. So, the world had to wait another decade, until the publication of [69], to learn about regular registers.

26. Time, Clocks and the Ordering of Events in a Distributed System
Communications of the ACM 21, 7 (July 1978), 558-565. Reprinted in several collections, including Distributed Computing: Concepts and Implementations, McEntire et al., ed. IEEE Press, 1984.

Jim Gray once told me that he had heard two different opinions of this paper: that it's trivial and that it's brilliant. I can't argue with the former, and I am disinclined to argue with the latter.

The origin of this paper was a note titled The Maintenance of Duplicate Databases by Paul Johnson and Bob Thomas. I believe their note introduced the idea of using message timestamps in a distributed algorithm. I happen to have a solid, visceral understanding of special relativity (see [4]). This enabled me to grasp immediately the essence of what they were trying to do. Special relativity teaches us that there is no invariant total ordering of events in space-time; different observers can disagree about which of two events happened first. There is only a partial order in which an event e1 precedes an event e2 iff e1 can causally affect e2. I realized that the essence of Johnson and Thomas's algorithm was the use of timestamps to provide a total ordering of events that was consistent with the causal order. This realization may have been brilliant. Having realized it, everything else was trivial. Because Thomas and Johnson didn't understand exactly what they were doing, they didn't get the algorithm quite right; their algorithm permitted anomalous behavior that essentially violated causality. I quickly wrote a short note pointing this out and correcting the algorithm.

It didn't take me long to realize that an algorithm for totally ordering events could be used to implement any distributed system. A distributed system can be described as a particular sequential state machine that is implemented with a network of processors. The ability to totally order the input requests leads immediately to an algorithm to implement an arbitrary state machine by a network of processors, and hence to implement any distributed system. So, I wrote this paper, which is about how to implement an arbitrary distributed state machine. As an illustration, I used the simplest example of a distributed system I could think of--a distributed mutual exclusion algorithm.

This is my most often cited paper. Many computer scientists claim to have read it. But I have rarely encountered anyone who was aware that the paper said anything about state machines. People seem to think that it is about either the causality relation on events in a distributed system, or the distributed mutual exclusion problem. People have insisted that there is nothing about state machines in the paper. I've even had to go back and reread it to convince myself that I really did remember what I had written.

The paper describes the synchronization of logical clocks. As something of an afterthought, I decided to see what kind of synchronization it provided for real-time clocks. So, I included a theorem about real-time synchronization. I was rather surprised by how difficult the proof turned out to be. This was an indication of what lay ahead in [61].

This paper won the 2000 PODC Influential Paper Award (later renamed the Edsger W. Dijkstra Prize in Distributed Computing).

28. The Implementation of Reliable Distributed Multiprocess Systems
Computer Networks 2 (1978), 95-114.

In [26], I introduced the idea of implementing any distributed system by using an algorithm to implement an arbitrary state machine in a distributed system. However, the algorithm in [26] assumed that processors never fail and all messages are delivered. This paper gives a fault-tolerant algorithm. It's a real-time algorithm, assuming

upper bounds on message delays in the absence of faults, and that nonfaulty processes had clocks synchronized to within a known bound.

To my knowledge, this is the first published paper to discuss arbitrary failures (later called Byzantine failures). It actually considered malicious behavior, not using such behavior simply as a metaphor for completely unpredictable failures. Its algorithm was the inspiration for the digital signature algorithm of [40]. With its use of real-time, this paper presaged the ideas in [54].

29. SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control (with John Wensley et al.)
Proceedings of the IEEE 66, 10 (October 1978), 1240-1255.

When it became clear that computers were going to be flying commercial aircraft, NASA began funding research to figure out how to make them reliable enough for the task. Part of that effort was the SIFT project at SRI. This project was perhaps most notable for producing the Byzantine generals problem and its solutions, first reported in [40].

This paper gives an overview of the complete SIFT project, which included designing the hardware and software and formally verifying the system's correctness. It announces the results that appear in [40]. It also is a very early example of the basic specification and verification method I still advocate: writing a specification as a state-transition system and showing that each step of the lower-level specification either implements a step of the higher-level one or is a "stuttering" step that leaves the higher-level state unchanged. The paper doesn't mention the use of an invariant, but that was probably omitted to save space.

This paper was a group effort that I choreographed in a final frenzy of activity to get the paper out in time for the issue's deadline. I don't remember who wrote what, but the section on verification seems to be my writing.

45. The Byzantine Generals Problem (with Marshall Pease and Robert Shostak)
ACM Transactions on Programming Languages and Systems 4, 3 (July 1982), 382-401.

I have long felt that, because it was posed as a cute problem about philosophers seated around a table, Dijkstra's dining philosopher's problem received much more attention than it deserves. (For example, it has probably received more attention in the theory community than the readers/writers problem, which illustrates the same principles and has much more practical importance.) I believed that the problem introduced in [40] was very important and deserved the attention of computer scientists. The popularity of the dining philosophers problem taught me that the best way to attract attention to a problem is to present it in terms of a story.

There is a problem in distributed computing that is sometimes called the Chinese Generals Problem, in which two generals have to come to a common agreement on whether to attack or retreat, but can communicate only by sending messengers who might never arrive. I stole the idea of the generals and posed the problem in terms of a group of generals, some of whom may be traitors, who have to reach a common decision. I wanted to assign the generals a nationality that would not offend any readers. At the time, Albania was a completely closed society, and I felt it unlikely that there would be any Albanians around to object, so the original title of this paper was The Albanian Generals Problem. Jack Goldberg was smart enough to realize that there were Albanians in the world outside Albania, and Albania might not always be a black hole, so he suggested that I find another name. The obviously more appropriate Byzantine generals then occurred to me.

The main reason for writing this paper was to assign the new name to the problem. But a new paper needed new results as well. I came up with a simpler way to describe the general 3n+1-processor algorithm. (Shostak's 4-processor algorithm was subtle but easy to understand; Pease's generalization was a remarkable tour de force.) We also added a generalization to networks that were not completely connected. (I don't remember whose work that was.) I also added some discussion of practical implementation details.

68. LaTeX: A Document Preparation System
Addison-Wesley, Reading, Mass. (1986).

In the early 80s, I was planning to write the Great American Concurrency Book. I was a TeX user, so I would need a set of macros. I thought that, with a little extra effort, I could make my macros usable by others. Don Knuth had begun issuing early releases of the current version of TeX, and I figured I could write what would become its standard macro package. That was the beginning of LaTeX. I was planning to write a user manual, but it never occurred to me that anyone would actually pay money for it. In 1983, Peter Gordon, an Addison-Wesley editor, and his colleagues visited me at SRI. Here is his account of what happened.

Our primary mission was to gather information for Addison-Wesley "to publish a computer-based document processing system specifically designed for scientists and engineers, in both academic and professional environments." This system was to be part of a series of related products (software, manuals, books) and services (database, production). (La)TeX was a candidate to be at the core of that system. (I am quoting from the original business plan.) Fortunately, I did not listen to your doubt that anyone would buy the LaTeX manual, because more than a few hundred thousand people actually did. The exact number, of course, cannot accurately be determined, inasmuch as many people (not all friends and relatives) bought the book more than once, so heavily was it used.

Meanwhile, I still haven't written the Great American Concurrency Book.

148. On Hair Color in France (with Ellen Gilkerson)
Annals of Improbable Results, Jan/Feb 2004, 18-19.

While traveling in France, Gilkerson and I observed many blonde women, but almost no blonde men. Suspecting that we had stumbled upon a remarkable scientific discovery, we endured several weeks of hardship visiting the auberges and restaurants of France to gather data. After several years of analysis and rigorous procrastination, we wrote this paper. Much of our magnificent prose was ruthlessly eliminated by the editor to leave space for less important research.

149. Formal Specification of a Web Services Protocol (with James E. Johnson, David E. Langworthy, and Friedrich H. Vogt)
Proceedings of the First International Workshop on Web Services and Formal Methods (WS-FM 2004), held February 23-24, 2004 in Pisa, Italy.

Fritz Vogt spent part of a sabbatical at our lab during the summer and fall of 2003. I was interested in getting TLA+ used in the product groups at Microsoft, and Fritz was looking for an interesting project involving distributed protocols. Through his contacts, we got together with Jim Johnson and Dave Langworthy, who work on Web protocols at Microsoft in Redmond. Jim and Dave were interested in the idea of formally specifying protocols, and Jim suggested that we look at the Web Services Atomic Transaction protocol as a simple example. Fritz and I spent part of our time for a couple of months writing it, with a lot of help from Jim and Dave in understanding the protocol. This paper describes the specification and our experience writing it. The specification itself can be found by clicking here. This was a routine exercise for me, as it would have been for anyone with a moderate amount of experience specifying concurrent systems. Using TLA+ for the first time was a learning experience for Fritz. It was a brand new world for Jim and Dave, who had never been exposed to formal methods before. They were happy with the results. Dave began writing specifications by himself, and has become something of a TLA+ guru for the Microsoft networking group. We submitted this paper to WS-FM 2004 as a way of introducing the Web services community to formal methods and TLA+.

Conteste las siguientes preguntas:

1. ¿En qué año se publicó el primer trabajo que hace referencia a las fallas arbitrarias (Bizantinas)? ¿En qué año se las empieza a llamar fallas Bizantinas?
2. Mencione una contribución de Lamport al área de los sistemas distribuidos.
3. Mencione una contribución de Lamport al área de los sistemas operativos.
4. ¿Por qué razón Lamport tituló un paper “The Byzantine Generals”?
5. ¿Qué tipo de investigaciones hace Lamport ahora?
6. ¿Qué es LaTeX?

Auto-Evaluación

1. Si dos eventos e y e' tienen las estampas vectoriales [0, 0, 1] y [2, 1, 0], respectivamente, ¿Qué podemos decir con respecto al orden (causal) en que ocurrieron estos eventos si comparamos los vectores entre sí?

   _____

2. ¿Cuál es la relación entre un reloj físico y un reloj lógico? (respuesta no necesita más de una oración).

   _____

3. De los tres algoritmos de exclusión mutua estudiados en clase, todos tienen sus desventajas. ¿Cuál es la principal desventaja para cada uno de los siguientes algoritmos? (a) Servidor de tokens, (b) Algoritmo basado en anillo, y (c) Algoritmo distribuido de Ricart y Agrawala.

   (a) _____

   (b) _____

   (c) _____

4. Cuando se usa el algoritmo basado en anillo para solucionar el problema de exclusión mutua distribuida, ____.
   a. la única falla que se puede presentar es que se pierda el token
   b. la única falla que se puede presentar es que falle un proceso
   c. un proceso que sostiene el token puede entrar a su sección crítica
   d. un proceso que sostiene el token debe esperar para entrar a su sección crítica

5. Al implementar el algoritmo de relojes lógicos de Lamport, cuando un proceso A envía un mensaje a un proceso B, ____.
   a. el reloj lógico de A puede retroceder
   b. el reloj lógico de A puede avanzar
   c. el reloj lógico de B puede retroceder
   d. el reloj lógico de B puede avanzar

6. En el algoritmo abusivo para elecciones, ____.
   a. el primer nodo en declararse coordinador gana
   b. el último nodo en declararse coordinador gana
   c. el nodo de mayor prioridad en declararse coordinador gana
   d. el nodo de mayor rendimiento en declararse coordinador gana

6. Redes de Distribución de Contenidos

Objetivo

Proporcionar una visión actualizada de las redes de distribución de contenidos, incluyendo sus ventajas, importancia actual y tecnologías utilizadas para implementarlas.

Palabras Clave

acierto en caché (hit)
caché
caja NAT
CDN
DHT
directorio centralizado
DNS round robin
enrutamiento de pedidos (DHT)
escalabilidad
falla en caché (miss)
forward proxy
free rider
gateway
Gnutella
GSLB
heurística de frescura (en cachés)
ICP
inundación de pedidos (gossip)
IP virtual (VIP)
modelo epidemiológico
proxy de intercepción
re-escritura HTML
reglas de reemplazo (en cachés)
replicación
reverse proxy (acelerador de servidor)
round robin
ruteo de pedidos basado en DNS
ruteo global de pedidos
servidor de nombres
sesión TCP
sistema p2p híbrido
sistema p2p puro
switch de capa 4
switch de capa 7
switch Web
TTL
VRRP

Enlaces de Interés

- Fenómeno pequeño mundo:
  - http://smallworld.columbia.edu/
  - http://en.wikipedia.org/wiki/Small_world_phenomenon
  - http://oracleofbacon.org/
- Distributed Hash Table: http://en.wikipedia.org/wiki/Distributed_hash_table
- Gnutella: http://en.wikipedia.org/wiki/Gnutella
- Akamai:
  - Wikipedia: http://en.wikipedia.org/wiki/Akamai_Technologies
  - Clientes: http://www.akamai.com/html/customers/customer_list.html
  - Video sobre Akamai: http://mitworld.mit.edu/video/199/
  - How Akamai Works: http://www.cs.washington.edu/homes/ratul/akamai.html
  - Akamai y CDNs: https://ecc.equinix.com/peering/downloads/CDN-peering.GPF9.ppt
  - Artículo en IEEE Internet Computing: http://www.akamai.com/dl/technical_publications/GloballyDistributedContentDelivery.pdf
- Content Delivery Network: http://en.wikipedia.org/wiki/Content_Delivery_Network

Apuntes

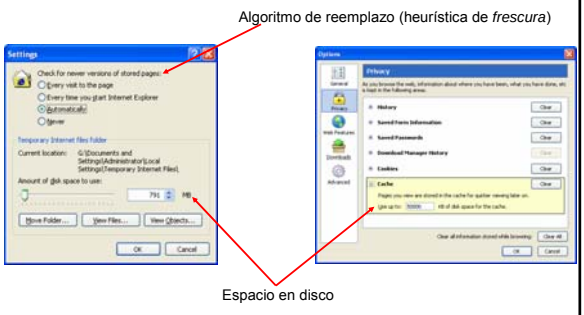
Redes de Distribución de Contenidos

Cacheo y Replicación

Introducción

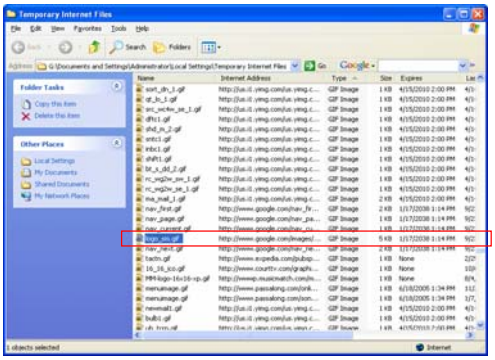
- Caché: sitio de escondite usado especialmente para almacenar provisiones
  - Ejemplo: ardillas y nueces
- Las cachés pueden estar colocadas en múltiples lugares en una red
  - Dentro del grupo de trabajo, gateway de red, empresa, ISP, backbone de la red, como parte de una granja de servidores

Cacheo Local



Cacheo Local

- Ahorra tiempo y reduce la carga en la red al almacenar contenido frecuentemente utilizado cerca del usuario
- Diseño asume que: si un usuario ha visitado una página una vez, es probable que la vuelvan a visitar
  - Ej.: logo de Yahoo o Google



Motivación y Metas del Cacheo Web

- Requerimiento más importante de una caché Web: que nunca otorgue silenciosamente contenido alterado o viejo
- Otros:
  - Confiabilidad: si falla no debe afectar las comunicaciones
  - No cachear objetos marcados como "no cacheables"

Cacheo Web

- Propósito: acelerar el acceso a contenido Web, reducir la carga en la red y reducir la carga en los servidores Web
- OJO: en caso de un "cache miss", los pasos extra involucrados en una caché Web pobremente diseñada pueden hacer más lento el acceso Web e incluso incrementar la carga en la red

Cacheo Web

- Se deben considerar los aciertos (hit) y faltas (miss) en la caché, para todos los usuarios, para determinar si en promedio acelera el acceso Web o no

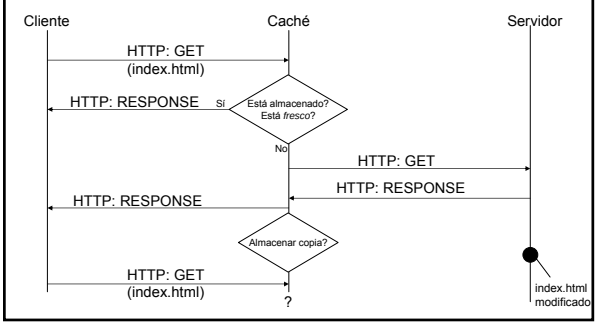
Cacheo Web: Parámetros que afectan el rendimiento

- Tamaño máximo
- Reglas de cacheabilidad
- Velocidad de la conexión de la red de la caché
- Tasa de transacciones máxima de la caché
- Localización en la red
- Características del contenido pedido por los usuarios

Cacheo Web: Factores que ayudan al administrador a su mejor configuración

- Instalación sencilla
- Reportes de tráfico y rendimiento
- Logs del tráfico de la caché
- Análisis de cacheabilidad de los objetos Web
- Identificación de pedidos que no pasarán por la caché (bypass)

Operaciones Básicas de una Caché Web Compartida



Reglas de Reemplazo

- Esquema general usado para decidir qué objetos se eliminan de la caché cuando está llena y nuevos objetos necesitan almacenarse
- Reglas Dinámicas de Objetos: usadas para examinar las características de un objeto y estimar su valor futuro para determinar si vale la pena añadirlo a la caché

Frescura

- Para asegurarse que objetos frescos se entreguen al cliente, la caché debe decidir sobre la frescura de un objeto en cada pedido del cliente

Reglas de Reemplazo

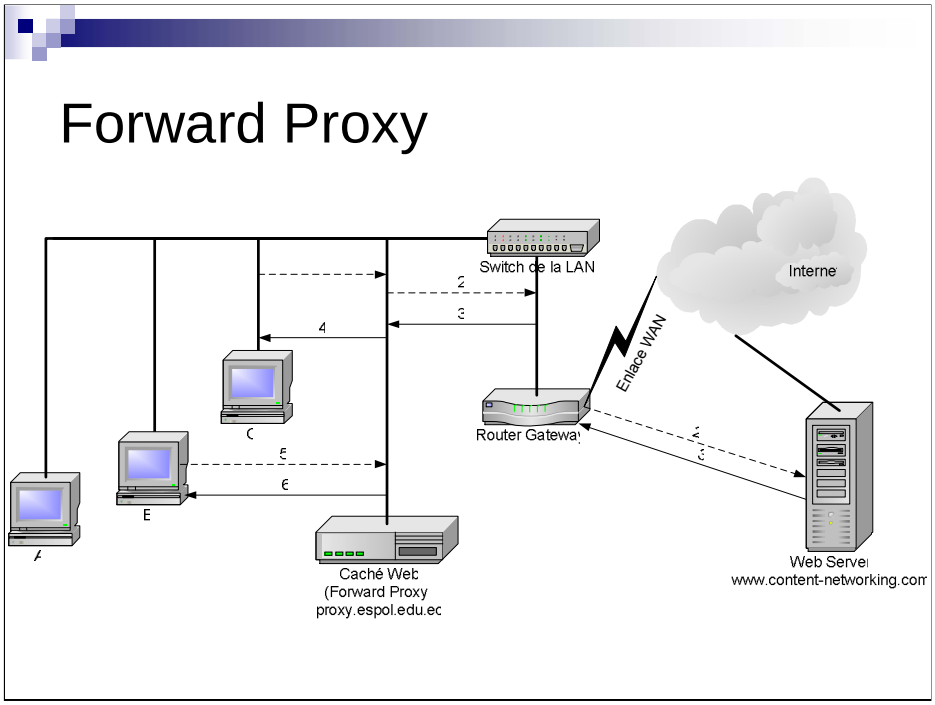
- LRU (least recently used)
- FIFO (first in first out)
- LFU (least frequently used)
- NTE (next to expire)
- LFF (largest file first)

Generalmente se combinan y complementan con heurísticas para proporcionar soluciones prácticas

Ubicación de una Caché en la Red

- Forward proxy
  - Actúa en representación de un grupo de consumidores de contenido
  - En cliente se configura en Internet Options → Connections → LAN Settings → Proxy
- Reverse proxy o acelerador de servidor
  - Actúa en representación del servidor de origen y ayuda a un grupo específico de servidores
  - Ej.: Google, Yahoo! (transparente para el cliente)
- Proxy de intercepción
  - Sirve el tráfico de red que se le direcciona
  - Ej.: caché de un ISP (transparente para el cliente)

NOTAS:



En este ejemplo de una caché tipo **Forward Proxy**, la ESPOL tiene instalado un proxy con servicio de cacheo tipo forwarding, para mejorar el rendimiento de la red (proxy.espol.edu.ec).

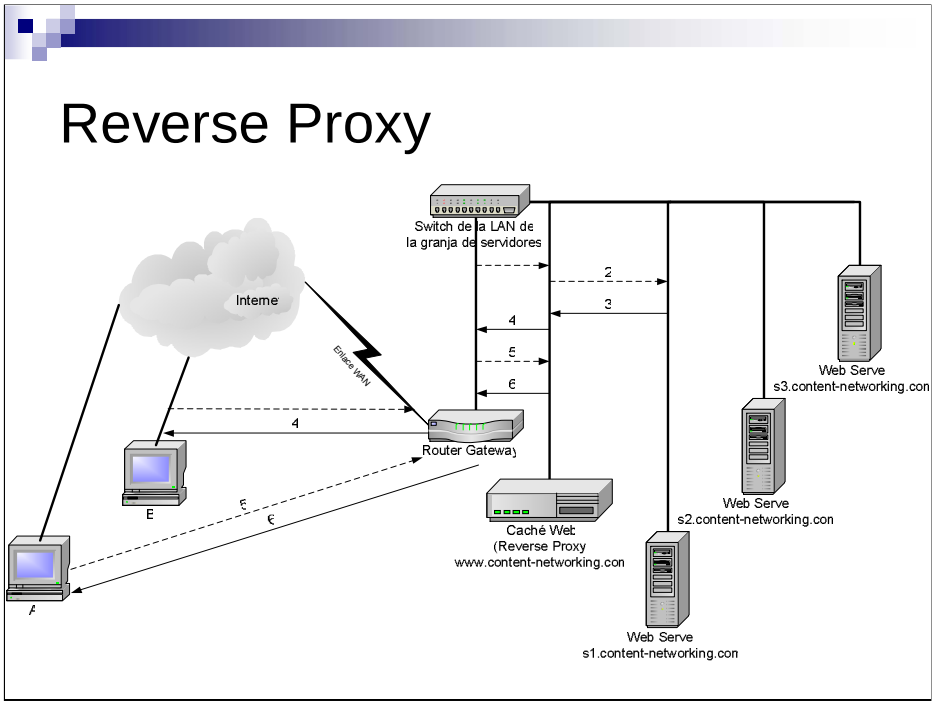
El cliente C pide la página index.html del servidor www.content-networking.com. Como el cliente tiene configurado el uso de un proxy, el navegador Web crea el pedido HTTP utilizando un URI absoluto (GET http://www.content-networking.com/index.html HTTP/1.1) y lo envía a través de la red LAN al proxy (proxy.espol.edu.ec), *no* al Web server. El proxy examina el pedido y busca el objeto Web en su almacenamiento local. En este ejemplo, la caché Web determina que no tiene una copia local del objeto.

La caché Web (proxy.espol.edu.ec) re-envía el pedido de la página index.html al Web server. El pedido sale por el router gateway y llega finalmente al servidor www.content-networking.com.

El Web server responde a la caché Web (proxy), enviándole la página requerida. La caché almacena una copia localmente, y

La caché Web envía la página Web solicitada, al cliente C. El navegador del cliente C la muestra en pantalla.

Algún tiempo más tarde, el cliente B solicita la misma página, y, tal como ocurrió en el paso 1, el pedido se lo envía al proxy, el cual determina que sí tiene una copia local actualizada y obtiene dicha copia de su almacenamiento local.



El proveedor de contenidos content-networking.com ha instalado un **reverse proxy** para mejorar el rendimiento de su granja de servidores. Al reverse proxy se lo ha designado www.content-networking.com, de tal manera que el tráfico de Internet destinado a su sitio Web irá primero al proxy y no a los servidores Web. Cada servidor Web tiene un identificador único (por ejemplo, s1.content-networking.com), de tal manera que el proxy los puede contactar directamente.

El cliente B pide la página index.html del servidor www.content-networking.com (GET index.html HTTP/1.1). El pedido es enrutado en Internet y llega al reverse proxy (www.content-networking.com). Cuando la caché Web recibe el pedido, busca el objeto (index.html) en su almacenamiento local y (en este caso) determina que no tiene una copia del objeto.

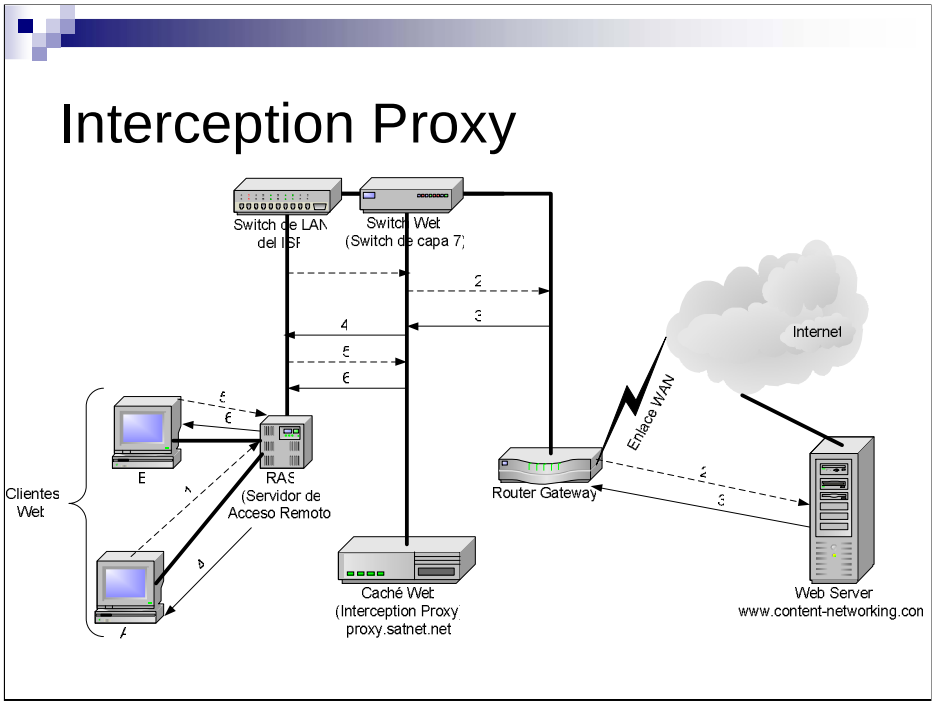
La caché Web (www.content-networking.com) re-envía el pedido a uno de los servidores Web de la granja. La caché puede utilizar diferentes algoritmos para decidir a qué servidor enviar el pedido. Estos algoritmos están diseñados para balancear el tráfico en la red o el tráfico que reciben los servidores, para mejorar los tiempos de respuesta de los pedidos, o para separar contenidos (por ejemplo, s1 atiende contenido estático, s2 atiende contenido dinámico y s3 atiende contenido multimedia). En este ejemplo, el pedido es enviado a través de la LAN al servidor s1.

El Web server responde a la caché Web (proxy), enviándole la página requerida. La caché almacena una copia localmente, y

La caché Web responde a través de Internet al cliente B, enviándole la página solicitada. El navegador Web al recibir la página la muestra en pantalla.

Algún tiempo más tarde, el cliente A solicita la misma página, y, tal como ocurrió en el paso 1, el pedido viaja a través de Internet hasta llegar al proxy, el cual ésta vez determina que sí tiene una copia local actualizada y obtiene dicha copia de su almacenamiento local.

La caché Web (proxy) le contesta al cliente A enviándole la página solicitada. Noten que en este caso, ningún tráfico ha sido enviado a la granja de servidores para satisfacer el



El proveedor de servicios de Internet (ISP) ha instalado un **proxy de intercepción** para mejorar el servicio que proporciona a sus clientes y a la vez reducir el tráfico que viaja por su enlace WAN hacia Internet. En este caso, es indispensable el uso de un Web switch, aunque hay routers que pueden configurarse para proporcionar un servicio similar. El propósito del switch es redireccionar el tráfico HTTP a la caché Web (proxy de intercepción). Los clientes A y B están conectados al ISP via modem.

El cliente A pide la página index.html del servidor www.content-networking.com (GET index.html HTTP/1.1). En la red del ISP, el Web switch intercepta este pedido y lo redirecciona a través de la LAN a la caché Web (proxy.satnet.net). Cuando la caché (proxy) recibe el pedido, busca el objeto en su almacenamiento local, y, (en este caso) determina que no tiene una copia local.

La caché Web re-envía el pedido al servidor Web (www.content-networking.com), el cual sale por el router gateway del ISP y viaja a través de Internet hasta llegar al Web server.

El Web server responde a la caché Web (proxy), enviándole la página requerida. La caché almacena una copia localmente, y

La caché Web responde al cliente A, enviándole la página solicitada. El navegador Web al recibir la página la muestra en pantalla.

Algún tiempo más tarde, el cliente B solicita la misma página, y, tal como ocurrió en el paso 1, el pedido llega al Web switch el cual lo redirecciona a la caché Web (proxy), el cual ésta vez determina que sí tiene una copia local actualizada y obtiene dicha copia de su almacenamiento local.

Evolución de Sistemas de Cacheo: Redes de Cachés

- En cada nivel de la jerarquía de las redes de Internet hay una o más cachés, las cuales se pueden comunicar entre sí
  - ICP: Internet Cache Protocol

Desafíos y Mitos de las Caché

- Desafío: determinar correctamente qué contenido cachear y que no, a pesar de que muchos servidores Web no indican correctamente si se debe cachear ciertos documentos o no
- Mitos ("técnicas" que no funcionan bien):
  - Pre-fetching (predecir qué va a pedir el usuario)
  - Push (pre-fetching iniciado por el servidor)

Facultando a las Redes de Distribución de Contenidos

Temas

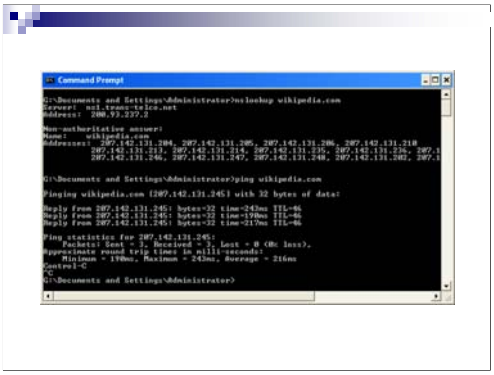
- DNS para compartir cargas
- Switching en capas 4-7
- Ruteo global de pedidos

DNS para Compartir Cargas

- Se puede configurar varios registros de dirección para cada nombre
- El name server puede rotar el orden de estos registros al contestar cada respuesta (round robin)
- El TTL del cliente influye en cuánto tiempo el cliente contacta a una IP
  - TTL: tiempo de vida de los registros DNS cacheados en el cliente

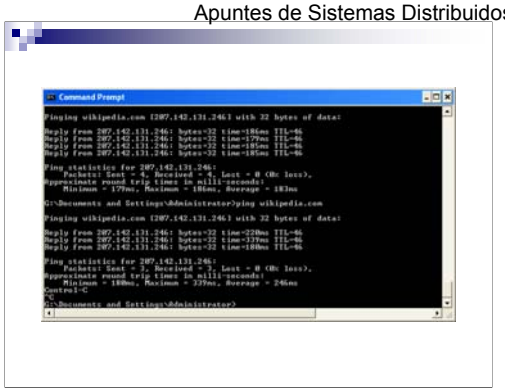
DNS para Compartir Cargas: Desventajas

- TTL de clientes limita la rotación
- Sirve para compartir carga pero no para balancear carga (ciertos servidores pueden estar más saturados)
- El cacheo de servidores DNS en varios niveles proporciona resultados no predecibles
- Al administrador del DNS puede no interesarle el balanceo (y reduce rendimiento de name server)
- No sabe el contexto bajo el cual se hizo el pedido; esta info. podría servir para dar un mejor servicio

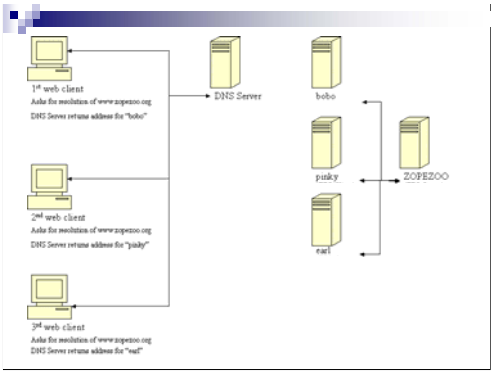


Para que comprendan mejor cómo funciona este mecanismo pueden hacer un experimento:

1. En el command prompt escriban “nslookup wikipedia.com”. De esta manera pueden ver que el servidor de wikipedia no es uno solo sino muchos (ver todas las IPs asignadas; es una IP por cada servidor de la granja o cluster).
2. Hagan un ping a wikipedia.com. Miren como el ping se hace a una de las IPs antes listada anteriormente. Esto es porque al hacer “ping wikipedia.com” el programa ping primero hace una consulta DNS y obtiene una IP asignada.
3. Escriban “nslookup” y den enter luego, “set debug” y den enter. Finalmente, escriban “wikipedia.com” y den enter. Vean todas las respuestas (“ANSWERS:”). Para cada una hay un TTL asociado. En el caso de mi computador es una hora (bueno, 59 minutos). Esto significa que mientras no pase una hora, el programa ping (o cualquier otro programa en su PC) seguirá utilizando la misma IP. Pueden probar haciendo ping a wikipedia nuevamente. Deben ver la misma IP que obtuvieron anteriormente.



4. Si dejan pasar el tiempo del TTL (es decir, en mi caso, una hora) y vuelven a hacer ping a wikipedia.com, lo más probable es que en esta ocasión hagan ping a una IP diferente. Esto se debe a que la entrada en la caché local ya expiró, y entonces el programa ping volvió a hacer una consulta DNS y el DNS de wikipedia nos retornó una IP diferente para asegurarse de distribuir la carga en los diferentes servidores de la granja.



NOTAS:

Switching en Capas 4-7

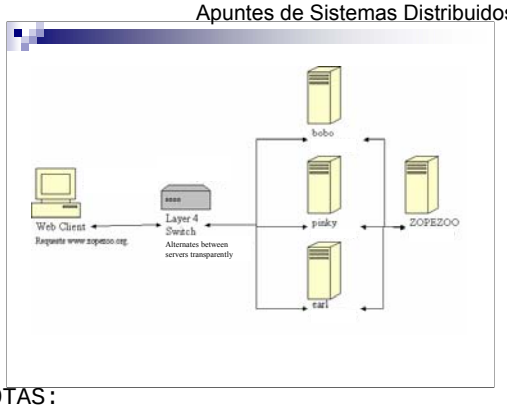
- A medida que los requerimientos sobre un servidor Web aumentan, el aumentar la capacidad del servidor para dar un mejor servicio deja de ser la mejor opción
- Mejor: usar múltiples servidores (granjas de servidores o clusters)
- Alternativa 1: balanceo con DNS, pero no balancea cargas
- Alternativa 2: Switching en capas 4-7 (a veces llamados “Web switches”), sí balancea cargas
- Pueden haber múltiples switches para evitar punto único de falla (protocolo VRRP)

Como ya vimos, cuando usamos granjas de servidores Web, podemos usar DNS round robin para distribuir las cargas, pero el resultado no es del todo ideal. Alternativamente, podemos utilizar switching en capas 4-7. A estos dispositivos a veces se los llama Web switches. Con estos dispositivos podemos balancear cargas e incluso utilizar políticas para decidir cómo distribuir los pedidos de los clientes.

Un problema del uso de un Web switch es que crea un punto único de fallo. Pero podemos usar múltiples switches Web, los cuales se comunican entre sí utilizando un protocolo como VRRP.

Switching en Capa 4

- IP virtual (VIP)
- Conmutación (switching) basado en puertos
  - Puede haber múltiples servidores para un puerto (ej.: 80)
- Permite que servidores tengan IP no pública (ej.: NAT)
  - 192.168.*.* y 10.*.*.*



NOTAS:

Los switches en capa 4 utilizan información de la capa de transporte para decidir a qué servidor envían los pedidos.

Este esquema funciona así: nuestro sitio tiene una sola IP asignada (no múltiples como en el caso de DNS round robin). Esta es la IP del Web switch. A esta IP se la llama IP virtual o VIP. Entonces, si queremos visitar la página de www.misitio.com, el pedido GET va a llegar al switch Web y este va a ver que el pedido va al puerto 80, por lo que re-direcciona el pedido al servidor Web de la empresa (o a uno de los servidores Web de la empresa, si se usa una granja de servidores). En cambio, si yo uso Outlook para a revisar mi correo de misitio.com, el switch Web verá que estoy utilizando el puerto POP y re-direcciona el pedio al mail server.

Otra ventaja es que los servidores de mi red no necesitan tener IP pública, ya que el switch Web puede realizar el papel de caja NAT.

Balanceo de Cargas de Servidores

- Switcheo en capa 4 permite una gran variedad de políticas de balanceo
  - Mejor disponible
    - Para pedidos nuevos
  - Persistencia
    - Conectar al mismo cliente con el mismo servidor
  - Servicios diferenciados
    - Diferentes clases de usuarios pueden recibir diferente servicio

Si en mi empresa yo tengo una granja de servidores, el Web switch puede alternar entre diferentes servidores de manera transparente para el cliente, y asegurarse de balancear las cargas.

El Web switch puede utilizar una combinación de diferentes políticas al momento de decidir a cuál servidor de la granja direccionar el pedido.

Políticas de Mejor Disponible

- Elección aleatoria del servidor
- Round Robin

  } Distribución uniforme

- Distribución ponderada (estática)
  - Administrador puede especificar % de tráfico que debe ir a cada servidor
- Distribución ponderada (dinámica)
  - Se dirige más tráfico a los servidores con mejor tiempo de respuesta
- Menos conexiones
- Menos paquetes (recibidos recientemente)
- Servidor menos ocupado
  - Agente en servidor monitorea su capacidad e informa al switch

Por ejemplo, puede escoger el “mejor disponible” de acuerdo a alguna heurística, como por ejemplo:

-Escoger aleatoriamente un servidor. A la larga, las cargas deben estar bastante bien balanceadas.

-O, escoger un servidor en orden round robin. A la larga, las cargas deben estar bastante bien balanceadas.

-O el administrador puede decir: el 25% de los pedidos debe ir al servidor X, el 25% al servidor Y, el 12,5% al servidor Z, etc. La idea es que el administrador puede que sepa que el servidor Z tiene una capacidad menor a la de X y Y, por lo tanto puede querer que reciba una carga menor.

-O, el Web switch puede llevar un control de cuál es el servidor que está teniendo mejores tiempos de respuesta, y direccionar los pedidos a ese servidor.

-O, puede direccionar el pedido al servidor que actualmente tenga el menor número de conexiones.

-O al servidor que haya recibido menos paquetes recientemente

-O al servidor que actualmente esté menos ocupado (un agente en cada servidor monitorea el status del servidor y envía reportes periódicos al switch).

Políticas de Persistencia

- Para mantener sesiones TCP o SSL, o en sesión activa (carrito de compras)
- Tiene precedencia a las de "mejor disponible"
- Se puede asociar una IP origen a un servidor en particular
  - Con timer para expirar sesiones no activas
- Problema: clientes detrás de caja NAT o servidor proxy parecen ser uno solo (ya que tienen la misma IP) y se los direccionaría a un mismo servidor
  - Solución: monitoreo de sesiones; se puede usar info. de capa 7 para diferenciar a un cliente de otro

Adicionalmente, hay que tener en cuenta que no podemos destruir sesiones TCP o sesiones SSL. Es decir, no podemos enviar el primer pedido de un cliente a un servidor y el segundo a otro servidor, ya que probablemente ambos pedidos pertenezcan a la misma sesión TCP y deben ser manejados por el mismo servidor. Para lograr esto, el Web switch puede asociar una IP origen a un servidor en particular.

Políticas Diferenciales

- Meta: dar el mejor uso a los recursos limitados
- Identifica a las transacciones o usuarios más importantes y a esos les da mejor servicio
  - Ej.: sesiones SSL
- Funciona bien cuando capacidad de servidor es limitada pero no está saturada
  - Con pocos clientes, no mejora en nada
  - Con servidor saturado, espera puede ser muy larga y aún a clientes "estrella" no se les podrá dar un buen servicio

En ocasiones, puede que queramos dar un servicio mejor a ciertos clientes. Por ejemplo, Amazon puede querer darles un mejor servicio a los clientes en sesiones SSL porque esos son los que actualmente están en medio de una transacción de compra, mientras que los otros clientes simplemente están navegando por el sitio.

NAT (del lado del servidor)

- NAT: Network Address Translation
- Controversial por:
  - Crean un punto único de fallo en la red
  - Se dificulta si se tiene multi-homing (múltiples accesos a Internet)
  - No funciona con IPs encriptadas (IPSec)
  - Complican o impiden el uso de mecanismos de autenticación basados en IP
  - Dificultad para iniciar sesión detrás de NAT

Los Web switches funcionan también como cajas NAT, ya que permiten que los servidores detrás del switch no tengan IP pública sino privada.

Switching en Capa 7

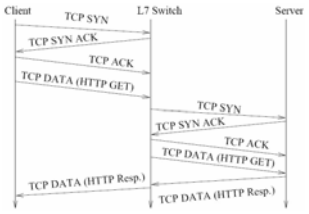
- Switches en capa 7 usan información como URL, cabecera HTTP, cookies, identificadores de sesión SSL
- Uso:
  - Dedicar servidores para usos específicos: ej.: servidor de imágenes, o servidor de páginas dinámicas (servidores pueden estar *tuned* apropiadamente)
  - Usar información de cacheabilidad (cabecera HTTP) para determinar si re-enviar el pedido al proxy de intercepción o no
  - Aplicación de políticas en base a cookies

El switching en capa 7 permite tomar decisiones más complejas que en capa 4, ya que el switch puede utilizar información de la capa de aplicación para tomar decisiones como:

Se puede dedicar servidores para usos específicos. Por ejemplo, si el pedido es de un GIF, se lo puede direccionar al servidor Web de imágenes, si es de un WAV se lo puede direccionar al servidor Web de archivos de audio. La ventaja de esto, es que servidores dedicados a usos específicos pueden ser *tuned* (configurados) de la manera más eficiente para ese uso en particular.

-Además, el Web switch puede ver los pedidos HTTP y analizar si el contenido del pedido es típicamente cacheable (ej.: HTML, JPG) o no (ej.: PHP, ASP). Si sí lo es, re-direcciona el pedido a la caché, y si no lo es, lo manda a uno de los servidores Web directamente.

Switching en Capa 7



Algo que cabe mencionar es una diferencia de los switches de capa 4 con los de capa 7. En un switch de capa 4, cada mensaje es automáticamente re-direccionado a uno de los servidores de la granja. En un switch de capa 7 esto no puede ser así, ya que los primeros mensajes de un pedido TCP corresponden al saludo de tres vías de TCP. El web switch no puede re-direccionar estos mensajes automáticamente, ya que durante el saludo inicial el Web switch todavía no ha tenido acceso a ninguna información de capa 7 que le permita tomar una decisión con respecto a qué servidor enviar el pedido. El switch debe esperar a que termine el saludo y reciba el primer mensaje con el pedido en sí (en este ejemplo, un GET HTTP) y ahí recién puede contactar a uno de los servidores. Pero no puede simplemente re-enviar los mensajes del cliente, ya que TCP maneja unos números de secuencia que no cuadrarían. Entonces, mientras que en un switch de capa 4 el switch simplemente está en medio de la sesión TCP, sin alterarla, en un switch de capa 7 el servidor debe mantener una sesión TCP con el cliente y otra con el servidor seleccionado, y mandar la información recibida por una sesión a la otra.

Proxies de Intercepción

- Un switch Web es clave para el funciona-miento correcto de una configuración con proxies de intercepción
  - Para que no todo pedido vaya al proxy
    - Puede usar reglas de capa 4 (ej.: puerto 80)
    - O, de capa 7 (ej.: información cacheable)
  - Para mejor rendimiento se puede tener cluster de proxies y el Web switch balancea la carga entre ellos

NOTAS:

Ruteo Global de Pedidos

- Para mejorar rendimiento yo puedo querer re-direccionar pedidos a servidores (réplicas) en otros lugares geográficos
- Desafíos
  - Cómo re-direccionar a clientes
  - Cómo estimar cercanía

Pero si mi sitio Web sigue creciendo en popularidad, aumentar más servidores a mi granja no es de gran ayuda, ya que probablemente tenga un cuello de botella en mi conexión a Internet. Entonces, puedo colocar múltiples granjas de servidores distribuidas geográficamente (en distintos puntos del país, en diferentes países o hasta en diferentes continentes). Pero esto debe ser transparente para el usuario. La idea es que el usuario apunte su navegador a www.google.com y automáticamente sea direccionado a un servidor en California o a uno en la India, según sea lo más conveniente. De hecho, podemos direccionar el pedido al servidor más “cercano”.

Re-direccionando Pedidos de Clientes

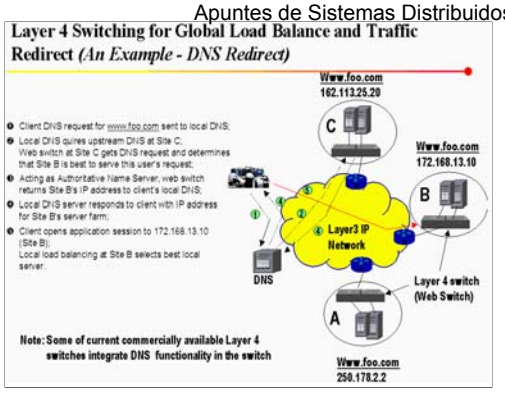
- Balanceo global de cargas de servidores (GSLB)
- Ruteo de pedidos basado en DNS
- Re-escribiendo HTML

Para re-direccionar los pedidos, tenemos los tres esquemas listados arriba.

Balanceo Global de Cargas de Servidores

- Proporcionado por al menos dos fabricantes de switches Web: Nortel y Alteon
- Extensión a switches de capas 4-7 en el cual los switches se comunican con otros distribuidos globalmente para estar concientes de la situación de los servidores locales y los remotos; además, combinan esto con comportamiento de DNS
  - Por ejemplo, el Web switch de Nortel también combina info. de cercanía; el switch de manera pasiva recopila info. de las redes, para saber cuál está más cerca de cada granja de servidores

En este esquema, en cada una de las granjas de servidores distribuidas globalmente tengo un switch Web. Estos switches Web corren un algoritmo a través del cual se comunican con los otros switches Web mi empresa (esto lo debe configurar el administrador). De esta manera, los switches están al tanto de todos los servidores Web de esta empresa, aún cuando muchos de estos servidores están en sitios geográficos diferentes. Estos switches Web se comunican entre sí para intercambiar información con respecto al status de los diferentes servidores Web. Además, estos switches también son servidores DNS. Entonces, cuando un cliente hace un pedido, primero debe hacer una consulta DNS, la cual va a ser direccionada a uno de los Web switches de la empresa, el cual va a tomar la decisión de a cual de los servidores de la empresa direccionar al cliente. La decisión la puede tomar en base al "mejor disponible" o incluso puede usar alguna medida de "cercanía", en el cual a un cliente de EEUU lo direccionaría a un servidor en ese país, mientras que a un cliente en Europa lo direccionaría a un servidor en ese continente.



NOTAS:

¿Cómo se selecciona al mejor servidor?

- En base a chequeos periódicos de *salud*
- En base a chequeos *flashback* en tiempo real (pings)
- En base a info. geográfica (ya que IANA asignó ciertas IP por continente)
- Condiciones de carga del servidor
- Heurísticas configuradas por el administrador
- Estimados de proximidad de los switches Web a los clientes (pasivo, en base a RTT)

NOTAS:

Ruteo de Pedidos Basado en DNS

- Lucent produjo su producto WebDNS por un tiempo limitado
- Medidas de proximidad basadas en agentes que corren en cachés Web de proxy reversos
- Agentes cooperan con WebDNS; recopilan info. de latencia a clientes y la reportan al WebDNS
- WebDNS usa info. recopilada para decidir cómo direccionar pedidos clientes

Este esquema es como el anterior. La diferencia es que además de los Web switches, este esquema necesita que unos agentes corran en los proxy de cada ubicación geográfica de los servidores. Estos agentes están encargados de pasivamente monitorear a los diferentes clientes, para determinar cuál está más "cerca" de qué grupo de servidores

Re-escribiendo HTML

- No se re-direcciona el primer pedido sino pedidos subsiguientes
- Re-escritura puede ser
  - *a priori*: cuando yo hago pedido el servidor decide cuál versión del HTML me envía
  - *Bajo demanda:* HTML es re-escrito para cada cliente/pedido
- Ej.: Akamai

Este esquema es diferente a los anteriores. En este esquema los pedidos van todos al sitio central de la empresa. No son re-direccionados a otras ubicaciones geográficas. Una vez que el pedido llega al servidor Web, este re-escribe el HTML de la página Web y cambia los URL locales por URLs en diferentes ubicaciones geográficas (dependiendo de cercanía o de qué servidor está más disponible, o alguna otra heurística).

La empresa Akamai (que significa “pilas” o “inteligente” en el idioma de los nativos de Hawaii) funciona bajo este esquema. Akamai tiene una red de distribución de contenidos formada por varios clusters de servidores Web distribuidos geográficamente en varias localidades en todo el mundo. Entonces, una empresa que contrata sus servicios recibe los pedidos de los clientes, pero cuando le retorna la página HTML a los clientes, re-escribe el HTML cambiando los URLs de las imágenes (u otro tipo de archivos) por un URL en la red de Akamai.

Como ejemplo de esto pueden realizar el sgt. experimento. Vayan a un sitio que contrate el servicio de Akamai, por ejemplo, www.1800flowers.net

Luego analicen el contenido HTML de la página. Noten la cantidad de URLs que apuntan a servidores de Akamai. El funcionamiento de Akamai es más complejo y combina una serie de técnicas. Por ejemplo, esos URL puede que sean direccionados a un servidor de Akamai en una ubicación geográfica o a otra, usando GSLB.

Estimando la Proximidad

- Sondeo reactivo
  - Introduce latencia adicional
- Sondeo proactivo
  - Monitoreo puede molestar a DNSs
- Monitoreo de conexiones
  - Totalmente pasivo (SYN/ACK – ACK)

Sondeo reactivo: cuando el Web switch recibe un pedido, en ese momento les pide a los otros Web switches que hagan ping al DNS del cliente para ver cuál está más cerca (es decir, cuál tiene el menor RTT).

Sondeo proactivo: constantemente los Web switches están haciendo pings a servidores DNS en todo el mundo para estimar la cercanía a ellos.

Monitoreo de conexiones: Cuando un cliente inicia una conexión con uno de los Web switches, el Web switch utiliza el saludo inicial de TCP para estimar la cercanía (RTT) y esta información la almacena en una tabla, de tal manera que en pedidos futuros pueda reportar cuán cerca está de ese cliente.

Una Comentario Adicional

- Los Web switches, cajas NAT y proxies de intercepción son muy útiles y utilizados actualmente, pero a muchos no les agradan porque van en contra del principio extremo-a-extremo
- Como alguien alguna vez dijo:
  - Quiero que mis autos sean rápidos
  - Que mi café esté caliente
  - Y que mis routers sean… TONTOS

NOTAS:

Redes de Contenidos Peer-to-Peer

NOTAS:

Temas

- ¿Qué son las redes peer-to-peer?
- Desafíos tecnológicos en las redes P2P
- Aspectos de negocios

Peer = Compañero = igual a los otros

Redes P2P se forman con relaciones simétricas de uno-a-uno

A diferencia de cliente/servidor donde hay relaciones asimétricas, de uno-a-muchos

Lo que se puede estudiar sobre las redes P2P es tanto que tomaría al menos un curso completo cubrirlo todo. Esta unidad simplemente presenta una visión general, concentrándose primeramente en los aspectos que tienen interés en el ámbito de las redes de contenidos.

OJO: Las redes P2P pertenecen a la categoría de redes superpuestas (Overlay Networks http://en.wikipedia.org/wiki/Overlay_network), ya que los enlaces entre los nodos son lógicos y no físicos.

NOTA: muchas de las diapositivas se pueden entender sin explicación adicional. En esos casos, no he incluido una sección de “Notas”.

http://en.wikipedia.org/wiki/Peer-to-peer

¿Qué son las Redes Peer-to-Peer?

- Son sistemas distribuidos en donde el software que corre en cada nodo proporciona funciones equivalentes
- Otra definición es: “conjunto de tecnologías que habilitan el intercambio directo de servicios o datos entre computadores”

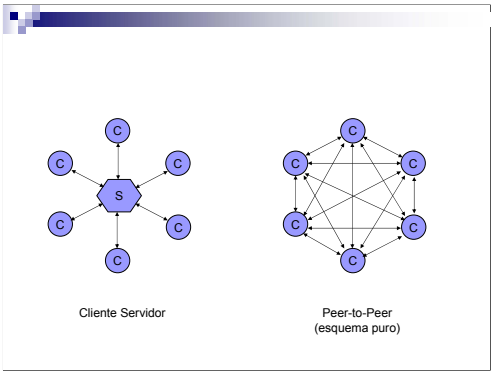
OJO: en un sistema P2P todos los peers o nodos están al mismo nivel (o casi al mismo nivel), lo cual no quiere decir que sean iguales. Recordemos que unos PCs pueden tener muchas mejores características que otros, etc.

A los nodos en una red P2P también se los suele llamar servents, ya que son clientes (clients) y servidores (servers) a la vez.

Tipos

- Puro: corre sin ningún control centralizado u organización jerárquica
- Híbrido: usa algunos recursos centralizados o jerárquicos

NOTAS:



NOTAS:

Sistemas P2P Tradicionales

- La red para distribución de mensajes Usenet era tradicionalmente P2P, incluso cuando cambió a trabajar sobre NNTP siguió siendo P2P
  - http://en.wikipedia.org/wiki/Usenet
  - http://en.wikipedia.org/wiki/Network_News_Transfer_Protocol
  - Actualmente pueden leer los mensajes de Usenet en http://groups.google.com
- DNS, aunque es híbrido por su jerarquía

El punto de esta diapositiva es que si bien la moda de los sistemas P2P es reciente, el concepto existe desde hace muchos años, como son los ejemplos de Usenet y el sistema DNS.

Una pregunta: Si la idea de sistemas P2P existía hace tantos años, ¿por qué es la moda tan reciente?

Respuesta: Solo a partir de 1995 empieza a ser común el tener conexiones a Intenet desde las casas, y en sus inicios, esas conexiones eran muy lentas como para usarlas para compartir archivos. A medida que las conexiones a ISPs fueron mejorando y bajando de precio, empezó a haber más usuarios en capacidad de formar parte de los actuales sistemas P2P.

Requerimientos del Sistema

- Características deseables para sistemas de administración de contenidos:
  - Disponibilidad
  - Durabilidad (persistencia de la información)
  - Control de acceso
  - Autenticidad
  - Que sea robusto
  - Escalabilidad masiva

Las siguientes son características ideales para sistemas confiables de administración de contenidos; son de particular interés ya que muchos de estos requerimientos son fácilmente alcanzables en sistemas P2P:

•**Disponibilidad: se desea poder tener acceso al sistema las 24 horas del día, 7 días a la semana, 365 días al año.**

•**Durabilidad (persistencia de la información): se desea que la información almacenada en el sistema esté disponible indefinidamente.**

•Control de acceso: la información en la red debe poder ser protegida de acceso no autorizado o alteración.

•Autenticidad: los documentos deben poder ser protegidos contra falsificaciones y alteraciones no deseadas.

•**Que sea robusto: resistente a ataques maliciosos, como por ejemplo, ataques DoS**

•**Escalabilidad masiva: Sistema debe funcionar bien con miles, millones o hasta billones de peers.**

Están en negritas los requerimientos que son más fáciles de conseguir en un sistema P2P que en uno centralizado.

Otros Requerimientos del Sistema

- En ciertos casos se desea:
  - Anonimidad
  - Expresión libre
  - Negación de responsabilidad
- Estas características de libertad de expresión y privacidad son más fáciles de obtener en un sistema P2P que en uno centralizado

Anonimidad: que el autor de un documento sea difícil o imposible de rastrear.

Expresión libre: que el contenido no pueda ser censurado, alterado o eliminado una vez que ha sido publicado en el sistema.

Negación de responsabilidad: si los usuarios no saben qué es lo que está almacenado en sus nodos, entonces pueden negar responsabilidad sobre su propiedad (es decir, pueden alegar que no es de ellos)

Características (de Sistemas P2P)

- Descentralización
- Escalabilidad masiva
- Auto-organización
- Conexiones ad hoc
- Rendimiento
- Seguridad
- Tolerancia a fallos
- Interoperabilidad

•Descentralización: sistemas descentralizados no tienen cuellos de botella ni puntos únicos de fallo. Los recursos son compartidos entre los miembros. Pero, al no haber nadie a cargo, puede ser difícil administrar la red, o encontrar un recurso en la ella.

•Escalabilidad masiva: (mientras los peers quieran seguir participando). El problema es que mientras más grande es la red P2P, más complejos deben ser los algoritmos que rigen su funcionamiento, ya que si no es así, perderíamos eficiencia y tendríamos un rendimiento no aceptable.

•Auto-organización: al no existir una entidad central que rija el comportamiento del sistema, los nodos deben auto-organizarse para su administración y manejo del contenido. Los algoritmos deben preocuparse de lograr escalabilidad, operación confiable, rendimiento satisfactorio y tolerancia a fallos, a partir de una colección de nodos no confiables con conexiones intermitentes.

•Conexiones ad hoc: los nodos se unen y salen de la red P2P en cualquier momento. Esto dificulta el poder proporcionar garantías de servicio.

•Rendimiento: la escala masiva potencial de estos sistemas puede tanto mejorar como empeorar el rendimiento (todo depende de los algoritmos que rigen a la red P2P). Un ejemplo de esto es la red Gnutella original, la cual tuvo serios problemas de escalabilidad.

•Seguridad: No hay manera de confiar en un 100% en los otros participantes.

•Tolerancia a fallos: ya que no hay un punto único de fallo. Sin embargo, las desconexiones de los nodos, congestiones en la red, o nodos fallosos pueden afectar la confiabilidad del sistema. La solución es usar una gran redundancia (de archivos, rutas en la red y otros recursos esenciales).

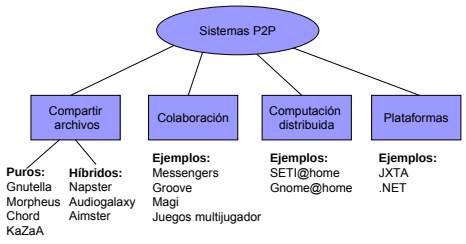
•Interoperabilidad:para que dos nodos se comuniquen, deben hablar el mismo lenguaje. Un nodo de Napster no puede comunicarse con uno de Gnutella.

El Efecto de la Red

- Ley de Metcalfe: El valor de una red crece con el cuadrado del número de usuarios
- La red P2P más exitosa es la que atrae a la mayor cantidad de usuarios (o al menos a los usuarios más valiosos)
  - Usuarios valiosos son aquellos que contribuyen bastante al contenido de la red, consumen poco, etc.

NOTAS:

Tipos de Redes P2P



NOTA: De los tipos anteriores, solamente vamos a estudiar en detalle el primero, ya que este tipo de sistemas son útiles bajo el contexto de las redes de contenidos.

También existen otros usos, como por ejemplo, las cachés Web P2P (ej.: Dijjer http://en.wikipedia.org/wiki/Dijjer).

Compartir Archivos

- Este es el tipo de aplicación P2P importante en el contexto de la distribución de contenidos
- Pueden ser puros o híbridos

NOTAS:

Compartir Archivos: Servicios

- Intercambio de archivos (directo entre peers)
  - Ej.: Freenet, Gnutella, KaZaa, eMule, Fasttrack, etc.
- Almacenamiento seguro y persistente
  - Ej.: OceanStore
- Anonimidad para autores
  - Ej.: Publius, Freenet

Los puntos anteriores son servicios proporcionados por ciertos sistemas P2P para compartir archivos (el primero es, lógicamente, un servicio básico proporcionado por todo sistema de este tipo).

Sistemas de Colaboración

- Proporcionan interacción en tiempo real que permite a usuarios cooperar por un fin común o competir en un juego
- Ejs.: MSN Messenger (híbrido), MyJXTA, Groove (www.groove.net), Magi (www.endeavors.com), DOOM, Quake

NOTAS:

Computación Distribuida

- Crea una gran supercomputadora virtual al agregar la capacidad de procesamiento de un gran número de PCs individuales conectadas a Internet
- Ejs.: SETI@home, Genome@home (genomeathome.stanford.edu), Entropia (www.entropia.com), United Devices (www.ud.com)

OJO: Entropia quebró y actualmente ya no presta sus servicios.

Plataformas

- Proporcionan soporte para servicios de naming, descubrimiento, comunicaciones, seguridad, y agregación de recursos para sistemas P2P
- Ejemplos:
  - JXTA y .NET
  - BOINC

NOTAS:

Desafíos Tecnológicos

- Localizar contenido
  - Modelo de directorio centralizado
  - Inundación de pedidos
  - Enrutamiento de documentos
- Confianza, responsabilidad, y reputación

Una vez que ya he formado mi red P2P, ¿cómo localizo lo que yo busco en la misma? Para esto existen tres alternativas estudiadas a continuación.

El segunto punto tiene que ver con el hecho de que cada nodo en la red P2P está administrado por alguien diferente, y esto puede dificultar la colaboración en la red.

Modelo de Directorio Centralizado

- Enfoque utilizado por el Napster original
- Peers se conectan a un directorio centralizado, al cual le indican qué archivos tienen disponibles
- Cuando se quiere buscar un archivo, se emite un pedido al directorio, el cual contesta con una lista de los peers en el directorio que contienen ese archivo
- El peer que busca el archivo luego contacta directamente a uno de los que tiene el archivo

NOTAS:

Ej: Napster (el original)

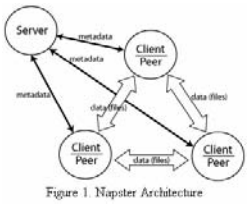


Figure 1. Napster Architecture

NOTAS:

Modelo de Directorio Centralizado

- Ventaja: sencillo y eficiente
- Desventajas:
  - Punto único de fallo
    - Solución: granja de servidores
      - El Napster original llegó a tener hasta 150 servidores que contenían el directorio centralizado
  - Punto único de censura
    - Este problema sí afectó al Napster original, ya que la RIAA los demandó por incentivar el intercambio ilegal de canciones, por lo que una corte decidió que debían apagar sus servidores. Sin directorio centralizado no habría como "apagar" el sistema.

http://en.wikipedia.org/wiki/Napster

Inundación de Pedidos

- También llamado modelo de “chismes” o epidemiológico
- Al conectarse a la red, el peer que busca un archivo envía una consulta a todos sus vecinos en la red P2P, estos a su vez propagan el pedido a sus vecinos y así sucesivamente
  - Esto termina cuando se encuentra una respuesta o el mensaje excede un límite de propagaciones (TTL)

Cabe aclarar que el modelo epidemiológico no es exactamente igual al de inundación. En el epidemiológico no se envía el pedido a todos nuestros vecinos, sino a un número aleatorio de ellos (después de todo, si a mi me da gripe, yo no contagio a todos mis conocidos, sino solo a algunos de ellos). Jugando con las matemáticas de la epidemiología, se puede alcanzar una “infección” de casi el 100% de los nodos del sistema, con un eficiencia mejor que la de los algoritmos de inundación.

Referencias:
http://www.research.ibm.com/nips03workshop/Presentations/Kempe.pdf (pero solo hasta la diapositiva 20)

http://en.wikipedia.org/wiki/Mathematical_modelling_in_epidemiology

Ej.: Gnutella (la original)

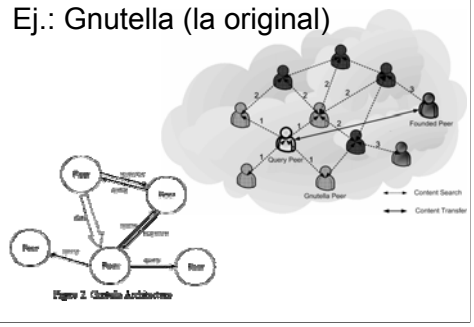


Figure 2. Gnutella Architecture

Las figuras muestran cómo se propaga (inunda) la consulta (query) en la red. Luego el archivo se transfiere directamente uno de los nodos que lo tiene. Los números en la figura superior son los pasos de la inundación.

El problema de este esquema es que consume mucho ancho de banda. De hecho, la versión original de Gnutella tiene problemas de escalabilidad debido a esto. Estos problemas son resueltos en el protocolo Gnutella2.

NOTA: a pesar de sus problemas de escalabilidad, la red Gnutella todavía está en uso y no ha colapsado. Lo ha logrado a través de la asignación de un TTL (tiempo de vida, medido en “saltos”) pequeño a cada pedido, de tal manera que los pedidos no saturen a la red.

http://en.wikipedia.org/wiki/Gnutella

http://en.wikipedia.org/wiki/Gnutella2

Inundación de Pedidos

- Ventajas:
  - Sencillo de implementar
  - Muy efectivo
  - No hay punto único de fallo ni de censura
- Desventajas:
  - Genera mucho tráfico innecesario y por lo tanto consume mucho ancho de banda
    - Esto limita la escalabilidad

Es importante aclarar que los problemas de escalabilidad pueden ser superados con algoritmos más complejos que combinen la inundación con otras técnicas para limitar la duplicación de mensajes.

Inundación de Pedidos: Versión Modificada

- En una versión modificada, la red tiene dos tipos de nodos: los normales y los súper nodos (súper peers o hubs)
- Un cliente inunda a los súper nodos que conoce hasta que recibe una respuesta al pedido
- Los súper nodos tienen conexiones a muchos otros nodos e indexan los archivos de estos
- Es decir, este esquema combina la inundación con un directorio centralizado, logrando lo mejor de ambas soluciones

Ejemplos: Gnutella2 y FastTrack (usada por KaZaA, Grokster, iMesh y (antes) Morpheus)

http://en.wikipedia.org/wiki/Gnutella2

http://en.wikipedia.org/wiki/FastTrack

http://en.wikipedia.org/wiki/Kazaa

Pregunta: ¿quienes son los super nodos?

Respuesta: son nodos normales cuyo programa cliente se percata de que tienen “buenas” características (es decir, pasa una gran cantidad de horas conectado, tiene un buen acceso a Internet, etc.), y por lo tanto, la red los promueve a super nodos.

Noten que este tipo de red no puede ser censurada como Napster, ya que aún cuando yo logre “apagar” a varios súper nodos de la red, siempre hay otros nodos que pueden hacer el papel de súper nodos.

Los Rendezvous Points/Peers (RP) de JXTA son algo así como súper nodos en las redes JXTA.

Ej.: FastTrack

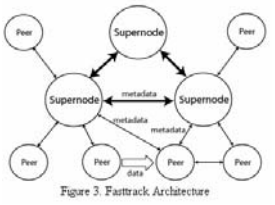


Figure 3. Fasttrack Architecture

NOTAS:

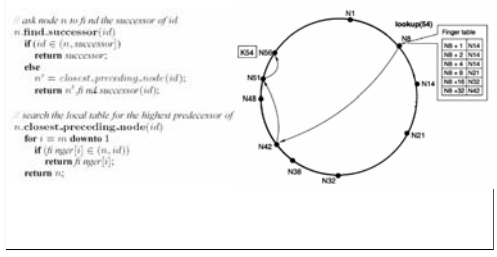
Enrutamiento de Pedidos

- Cada peer tiene información útil y parcial con respecto a la ubicación de los archivos en la red P2P
  - Cada nodo sabe algo que nos acerca a la respuesta, hasta así llegar a encontrar el archivo deseado
- Ej.: Chord, Pastry, CAN, Kademilia
  - También llamadas DHT (hash tables distribuidas)

A estas a veces se las llama redes P2P de tercera generación. Para una buena explicación de este tema, ver:
http://en.wikipedia.org/wiki/Distributed_hash_table

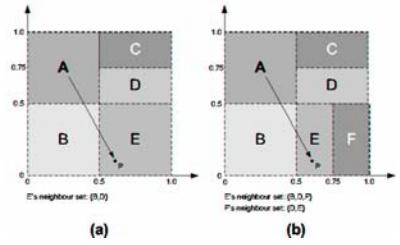
Ejemplos de sistemas P2P que usan el concepto de DHTs como base para su desarrollo son: BitTorrent (http://en.wikipedia.org/wiki/Bittorrent), Dijjer y Freenet (http://en.wikipedia.org/wiki/Freenet). Kademilia (http://en.wikipedia.org/wiki/Kademlia) es usada en Overnet, la cual es usada por eDonkey y MLDonkey.

Ej.: Chord



(esta diapositiva es presentada únicamente como ilustración al concepto de DHT. No es necesario que entiendan cómo funciona Chord).

Ej.: CAN



(esta diapositiva es presentada únicamente como ilustración al concepto de DHT. No es necesario que entiendan cómo funciona CAN).

Ej.: Pastry

Figure 1. Routing table of a Pastry node with nodeId 65a1x, $b = 4$. Digits are in base 16, x represents an arbitrary suffix.

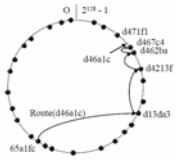


Figure 2. Routing a message from node 65a1fc with key d46a1c. The dots depict live nodes in Pastry's circular namespace.

(esta diapositiva es presentada únicamente como ilustración al concepto de DHT. No es necesario que entiendan cómo funciona Pastry).

Enrutamiento de Pedidos

- Ventajas:
  - Las búsquedas se pueden completar en un número máximo de pasos, por lo que es más escalable y reduce el consumo de ancho de banda
  - Genera sistemas con buen rendimiento
- Desventajas:
  - Complejidad de implementación puede presentar problemas al realizar búsquedas complejas y cuando los nodos se conectan y desconectan frecuentemente

NOTAS:

Confianza

- Los sistemas P2P son susceptibles al mal comportamiento de los miembros
  - Integridad del contenido
  - La "tragedia de lo comunitario"
    - *Free riders* (nodos que viajan de gratis)
  - Integridad del protocolo

•Integridad del contenido: ¿qué pasa si un miembro dice que tiene un archivo X y lo pone a disposición de la red, pero el archivo en realidad no es X sino otra cosa? Se reduce la utilidad de la red.

•Un ejemplo de esto ocurrió en el 2003 cuando Madonna (o alguien que la representaba) colocó en varias redes P2P archivos que decían ser canciones de ella, pero al tocar los MP3 se oía a la cantante decir "What the fuck do you think you are doing?" (http://news.zdnet.co.uk/hardware/emergingtech/0,39020357,2133749,00.htm)

•La tragedia de lo comunitario: La "tragedia de lo comunitario" es una frase que se usa para referir a una clase de fenómenos que tiene que ver con el conflicto de recursos entre los intereses del individuo y los del bien común (http://en.wikipedia.org/wiki/Tragedy_of_the_commons). Por ejemplo, en una red para compartir archivos hay quienes no comparten archivos, solo los bajan de otros (*free riders*). Esto ocurre porque como individuo, yo saco el máximo de la red si nadie se baja nada de mi, y más bien yo bajo mucho de otros usuarios. Pero si todos siguen este enfoque, la red no tendría utilidad (bien común).

•Integridad del protocolo: Si un miembro no sigue el protocolo de la red, puede hacer que las cosas empiecen a funcionar mal; si muchos no siguen el protocolo, pueden colapsar la red.

Responsabilidad y Reputación

- Responsabilidad
  - Para asegurarme de que no hayan free riders puedo utilizar algún tipo de incentivo u otro mecanismo que obligue a que los nodos compartan también sus archivos
- Reputación
  - En ciertas redes puede utilizarse un sistema de reputación (al estilo eBay) para asegurar el buen comportamiento de los nodos en la red

Responsabilidad: Por ejemplo, en Mojo Nation (red P2P que ya no existe) había que tener "créditos" o "Mojo" para poder bajar archivos. Se ganaba Mojo al compartir archivos. De esta manera se obligaba a compartir a la vez de bajar. Otras redes P2P usan otros mecanismos para asegurar que los clientes compartan archivos (ej.: BitTorrent: tit-for-tat)

Aspectos de Negocios

- ¿Cómo han hecho dinero los desarrolladores de sistemas P2P?
  - Patentes (Napster original)
  - Venta de publicidad (ej.: Kazaa)
  - Venta de licencias a usuarios (ej.: Kazaa Lite)
  - Venta de suscripciones (ej.: nuevo Napster)
  - Donaciones y charlas (ej.: BitTorrent)
  - Consultoría y entrenamiento (ej.: Jabber)
  - Venta de licencias a compañías (ej.: Jabber)

Dato interesante: Roxio pagó $5 millones por el portafolio de patentes de Napster

Enlace de interés:

http://en.wikipedia.org/wiki/Jabber

Aspectos Legales

- Copiar archivos entre PCs es legal
- Infringir leyes de copyright es ilegal
- Varios creadores de sistemas P2P han sido demandados por las disqueras por infringir en leyes de copyright al incentivar el intercambio ilegal de canciones
  - De hecho, también se ha demandado a universidades (por las acciones de sus estudiantes) y usuarios comunes
    - Por ahora, no se puede demandar a los ISPs

Los ISPs al utilizar cachés, generalmente cachean archivos MP3 ilegales. No se los puede demandar* ya que se ha determinado que es la caché y no el ISP quien decide qué cachear o no, y por esto no se puede hacer responsable al ISP por el contenido que cachea (según regulación de 1998 de la DMCA de EE.UU.).

* En muchos países no hay leyes al respecto, así que en ellos no queda claro si es legal o ilegal.

En realidad este tema es muy complejo, y hasta hay libros que detallan todos los aspectos involucrados (http://www.amazon.com/gp/product/0471377481/102-8707986-2854555?v=glance&n=283155).

Aspectos Legales

- Quienes son partidarios de legalizar la distribución de archivos en redes P2P (algo que probablemente no ocurra) alegan que de hecho beneficia a los artistas ya que más gente tiene acceso a su música y luego es más probable que pagen por ir a sus conciertos, etc.
  - De hecho, hay varios casos de "éxito" de grupos cuyos ingresos han mejorado gracias a las redes P2P. Para un ejemplo, ver: http://archives.cnn.com/2000/fyi/real.life/10/26/post.napster/

NOTAS:

Aspectos Legales

RIAA: Recording Industry Association of America
http://en.wikipedia.org/wiki/RIAA

NOTAS:

Aspectos Legales

(continúa)

NOTAS:

NOTAS:

Una Última Nota: Otra Clasificación de los Sistemas P2P

	Centralization		
	Hybrid	Partial	None
Unstructured	Napster, Publius	Kazaa, Morpheus, Gnutella, Edutella	Gnutella, FreeHaven
Structured Infrastructures			Chord, CAN, Tapestry, Pastry
Structured Systems			OceanStore, Mnemosyne, Scan, PAST, Kademlia, Tarzan

No existe un consenso general sobre cómo clasificar a los sistemas P2P. Por ejemplo, en la tabla superior híbridos son los sistemas con un recurso centralizado, parciales los que no hay servicios centralizados pero sí tienen ciertos nodos que son más importantes que otros (los súper nodos) y puros si no hay jerarquía alguna.

Además, son no estructurados si no usan un esquema de DHT y estructurados si sí lo usan.

Las infraestructuras estructuradas son las DHTs en sí, mientras que los sistemas estructurados son aquellos construidos sobre DHTs.

Ejercicios

1. Explique por qué razón el *hit rate* (tasa de pedidos que pueden ser satisfechos directamente de la caché) es probablemente más alto en un *reverse proxy* (instalado en el sitio de un servidor Web) que en un proxy de intercepción instalado en un ISP.
2. En Gnutella es más fácil encontrar archivos comunes que archivos no comunes.
    a. ¿Por qué?
    b. Describa brevemente cómo podría modificarse Gnutella para que haga que los archivos no comunes sean más fáciles de encontrar en la red. La solución no debería afectar la escalabilidad del protocolo.
3. Para mejorar la escalabilidad de sitios Web, muchas veces se utilizan clusters de en lugar de un solo servidor Web. En esos casos, un enfoque muy común para atender los pedidos de los clientes se denomina *distribución de pedidos en base a contenidos*. En este esquema, un pedido o requerimiento de un cliente es re-direccionado a uno de los computadores del cluster en base al contenido del pedido. ¿Cuáles serían las ventajas y desventajas de este esquema? Para contestar esta pregunta puede ser útil contrastar este esquema con uno en que los pedidos se envían de manera round-robin a cada uno de los computadores del cluster. Contestar esta pregunta pensando en el servicio específico (un servidor Web), y en base al diseño de sistemas distribuidos en general.
4. Una institución tiene una conexión T1 (1.544 Mbps) a Internet, la cual da servicio a pedidos Web de aproximadamente unos 500 usuarios en la red de la institución. La institución está considerando poner un servidor proxy para que sirva de caché Web para todos los usuarios de la institución. Cuando un usuario hace un pedido Web, el pedido primero se envía al servidor proxy para ver si puede responder el pedido con la información disponible en la caché. Si no es así, el proxy re-envía el pedido al servidor adecuado.
    a. Explique brevemente los beneficios de instalar un servidor proxy como el descrito para que haga el papel de caché Web.

Tareas

1. Recopile una lista de usos “legales” de los sistemas P2P, incluyendo ejemplos específicos.
2. Investigue la diferencia entre un cluster de alto rendimiento (High Performance) y un cluster de alta disponibilidad (High Availability). De los sistemas estudiados en este capítulo, ¿cuál utiliza/utilizó un cluster de alto rendimiento?
3. Investigue sobre los switches CSS (Content Services Switch) de Cisco e indique qué mecanismos de los estudiados en clase utilizan este tipo de switches.

Caso de Estudio: Akamai

Lea los siguientes documentos con casos de estudio de Akamai y descripciones de algunas de los servicios que ofrece esta compañía. Luego conteste las siguientes preguntas:

1. ¿Qué problemas se les presentan a las compañías que deciden de contratar servicios de CDNs como Akamai?
2. ¿Qué mecanismos utilizan las CDNs como Akamai para proporcionar servicios a sus clientes?
3. ¿Qué efectos tiene el uso de CDNs en compañías como las descritas en los casos de estudio?

A Gold Medal Performance for NBCOlympics.com

The Situation

With exclusive U.S. rights to broadcast television coverage of the 2004 Olympics, NBC decided to take advantage of increases in broadband adoption to deliver real-time, interactive, and media-rich coverage of the Olympics on the www.nbcolympics.com site. NBC wanted to build a premier Olympic Web site with multimedia integration and exclusive content that complemented television broadcasts, including those of its affiliates around the country. The goal was to make the site an interactive and definitive resource of the Athens Games for an anticipated audience of 20 million site visitors during August.

The Challenge

A lot was riding on the site—with more planned television coverage than ever before (1,200 hours across seven networks), the NBC Olympics site had the potential to become the most visited site for a one-time event. And if all went as planned, the site would generate significant revenues from advertisers. The success of both these factors hinged on the site's performance and scalability. NBC solicited bids from a variety of solution providers for help developing the necessary online applications and a solid infrastructure. Internet Broadcasting (IB) was one of those asked to bid.

"NBC had a number of key requirements. Its broadcasting rights limited the distribution of its competition videos to a U.S. audience, so they had to be delivered securely, allowing only the intended recipient to view them. The site needed to serve near-real-time event results to millions of viewers per day without fail. NBC needed applications that helped them run polls and deliver information in a multimedia fashion. Content had to be localized for each NBC affiliate around the country for the O-Zone portion of the site and NBC wanted insight into the traffic across its site as well as at the affiliate level. And not surprisingly, NBC wanted all this done cost-effectively," explains Dave Abbott, CTO of IB.

The Goal

To ensure successful results from their Olympics programming NBC needed to:

- **Deliver Real-time, Interactive, and Media-rich Coverage of The Olympics**
- **Use The Web to Complement Television Broadcasts** including those of its affiliates around the country
- **Enhance Viewing Experience** for anticipated audience of 20 million site visitors
- **Implement Resource-efficient Solution for Limited-time Event**
- **Meet Requirements to Limit Access to Video Highlights**

Why Akamai

Proven, Resource-Efficient Solution

IB immediately honed in on the economic factor—for a two-week event, it made no sense to deploy hundreds of servers and contract for additional bandwidth from multiple vendors. Installing the 500 servers needed to support the site's goals would require a huge footprint in NBC's datacenter. Beyond that, NBC needed to figure out how to ensure video highlights were not distributed outside the U.S. And managing the procurement, installment, and operation of all the necessary equipment and technology would require a fair number of dedicated personnel.

(continued on back)

"The whole idea was to enable the NBC Olympics' site to be a perfect online complement to broadcast coverage, and to do so cost-effectively by minimizing the origin data center infrastructure and offloading all traffic to the Akamai network. By meeting these requirements, the Akamai solution has provided a stellar value for both NBC and IB."

—Dave Abbott, CTO of Internet Broadcasting (IB)

COMPANY

NBC
New York, NY
www.nbc.com

INDUSTRY

Media & Entertainment

SOLUTION

Akamai EdgeSuite®
Akamai Streaming
Akamai EdgeComputing℠
Akamai On Demand Registration
Akamai Access Control
Akamai Content Targeting

KEY IMPACTS

- Supported fastest and most heavily trafficked Olympics' site in history—276 million page views over 17-day period
- Delivered first 100% available Olympics' site
- Processed 6 separate interactive Web applications, including polling and email-a-friend applications
- Enabled significant infrastructure savings by obviating need for over 500 servers
- Delivered average page download time of under 3 seconds throughout the Olympics
- Ensured only authorized users could access restricted video clips

Why Akamai *(continued)*

"It was not practical to install 500 servers. Besides, that plan didn't align with NBC's goals for a cost-effective solution. For IB, the solution was a no-brainer. We offered to package up our proprietary content management system and other software tools and install them in NBC's New Jersey datacenter. Then we recommended that NBC extend its existing infrastructure across the Akamai EdgePlatform and take advantage of the bottomless well that Akamai's on demand network provides," continues Abbot.

Agile Applications

IB has long relied on Akamai's on demand computing platform to effectively increase the performance and flexibility of its own infrastructure and online applications while reducing the resources required to support its Web operations.

"We felt confident in our ability to develop the needed online Flash applications, such as games that provided moment-by-moment analysis of track and field races and explained how different events are scored. And we knew that Akamai could quickly and reliably deliver these applications and the heavy site pages, which ended up averaging more than 300 KBs. We already had proof that our interactive surveys and polls would scale successfully across the Akamai Platform," continued Abbott.

Targeting The Right Content to The Right Users

On top of that, Akamai's geographic and bandwidth verification technologies would help NBC identify whether or not visitors were based in the U.S. and were on a dial-up or broadband connection. Knowing this, NBC could prevent large video files from being served to visitors outside of the U.S. or on slow connections. "To further ensure only authorized users could access the videos, Akamai offered the Windows Media 9 digital rights management wrapper. And let's face it—there's no other company that can ensure the security and uninterrupted operation of a site the way Akamai can," states Abbott.

Real-Time, Easy-to-Use Reporting

Equally important, IB would have access to Akamai's comprehensive reporting tools for insight into site trends and performance. IB could view site traffic in real-time and see daily summaries of a variety of visitor activities. The reporting tools supported IB's requirements for a customized dashboard and enabled IB to easily aggregate all data related to NBC affiliates around the country who were participating in the O-Zone.

Record-Breaking Results

Akamai supported all content delivery for NBC's Olympics coverage—including streaming and dynamic applications such as polling, live event scoring, and Web email that site visitors sent to friends or commentators. Akamai also streamed all pre-Game highlights, which included exclusive footage of the U.S. Olympic Team Trials, great moments from the 1996 Atlanta and 2000 Sydney Games, and interviews with more than 40 top U.S. Olympians.

As predicted, the www.nbcolympic.com site beat all previous records, registering an unprecedented number of visitors and a traffic peak of 1.6 gigabits per second. Using third parties to measure site performance, IB and NBC were assured that the site was available 100% of the time, a first for an Olympics' site. The average page download time during the two-week Olympics was under 3 seconds. While NBC is thrilled that Akamai and IB ensured a unique and satisfying online experience for millions of site visitors, it's equally pleased that it sold nearly twice as many ads as during the 2002 Olympics.

"To know that Akamai can deliver any content —even if it's interactive and personalized— quickly and reliably to a large audience made them the obvious choice. But on top of that, they provided functionality such as geographic and bandwidth verification technology that enabled NBC to distribute its content in creative ways," concludes Abbott.

About IB

Founded in 1995 as a pioneer in TV/Web convergence, Internet Broadcasting is the first and largest network of local news Web channels—Web sites that extend the local news, information, and advertising reach of major TV stations to the Web. The IB network of local Web channels covers 50 cities, including 20 of the top 25, a total of 64% of US households, and represents the leading American television broadcasting stations, such as NBC, Hearst-Argyle Television, McGraw-Hill Broadcasting Group, and Cox Television, to name a few. For more information, visit **www.ibsys.com**.

About NBC

NBC began its online coverage of the Olympic Games in 1996 during the Atlanta Olympics. That Internet presence was dramatically increased in 2000 during NBC's Sydney Games coverage and again in 2002 at Salt Lake where the NBC Web site became one of the Web's most trafficked and critically acclaimed sports Internet sites. NBCOlympics.com averaged more than one million visitors a day during the Salt Lake Winter Games and set a single-day record with 2.4 million visitors the day of the ladies figure skating final. NBC, America's Olympic Network, holds the exclusive U.S. media rights to the Olympic Games through 2012, which include Athens in 2004, Torino, Italy in 2006, Beijing in 2008, Vancouver 2010 and the Summer Games of 2012. For more information, visit **www.nbc.com**.

About Akamai

Akamai® is the leading global service provider for accelerating content and business processes online. Thousands of organizations have formed trusted relationships with Akamai, improving their revenue and reducing costs by maximizing the performance of their online businesses. Leveraging the Akamai EdgePlatform, these organizations gain business advantage today, and have the foundation for the emerging Web solutions of tomorrow. Akamai is "The Trusted Choice for Online Business." For more information, visit **www.akamai.com**.

Akamai
The Trusted Choice For Online Business™

Akamai Technologies, Inc.
U.S. Headquarters
8 Cambridge Center, Cambridge, MA 02142
Tel 617.444.3000
Fax 617.444.3001
U.S. toll-free 877.4AKAMAI (877.425.2624)

Akamai Technologies GmbH
Park Village, Betastrasse 10 b
D-85774 Unterföhring, Germany
Tel +49 89 94006-0

www.akamai.com

AKAMCS-NBC-1106

Sony Ericsson and Akamai Launch A New Age In Mobile Communications

The Challenge

Provide Customers Around The World With Exceptional Online Service

In 2001, Sony Corporation and Ericsson formed Sony Ericsson Mobile Communications to deliver accessible mobile multimedia communication solutions to customers worldwide. In efforts to build the Sony Ericsson brand and become the premiere provider of mobile multimedia communication devices, the company wanted to provide an exceptional online experience to its global customer base. The company chose the Akamai EdgeSuite solution to ensure the consistent, reliable, and secure delivery of its Web site content, and to help Sony Ericsson meet its joint venture launch date without additional Web infrastructure build-out.

Realizing the importance of differentiating itself from the competition, Sony Ericsson next developed innovative and practical online applications such as a dealer locator, phone configuration tools, and screen image downloads, to help its customers gain the most from their phones. Providing this functionality and ease of use became the driving force behind Sony Ericsson's Web sites. These value-added Web capabilities were also key to establishing brand loyalty and ultimately driving global, online visitors into brick and mortar Sony Ericsson dealers. However, providing critical Java-based Web applications to most international markets without unacceptably slow download times proved difficult.

Sony Ericsson considered building out data centers in several regions to support its worldwide customer base, but this option proved costly and added to the complexity of sourcing and managing dispersed data centers. Sony Ericsson wanted to both extend and simplify its Web infrastructure that supported more than 60 sites and millions of visitors per month.

The Goal

Sony Ericsson needed to provide:

- **Web Application Deployment**

Why Akamai

Process Applications On The Internet's Edge, Close to Users

As a satisfied EdgeSuite customer, Sony Ericsson turned to Akamai for help in solving its online application issues. Akamai's EdgeComputing solution supports the execution of Java Server Pages, Servlets, and JavaBeans on the edge of the Internet. This application server edge processing allows Sony Ericsson to offload infrequently changing Web data and page elements to Akamai's globally distributed computing platform, reducing expensive and inefficient roundtrips to the origin site. Components such as product catalogs, dealer information, and shopping carts can be cached and processed locally to deliver fast and reliable applications that provide users around the world with the information they want and yield measurable business results.

EdgeSuite and EdgeComputing enabled Sony Ericsson to dramatically improve the performance of its Web site and mission-critical dealer locator application-guaranteeing the successful launch of the joint venture and ensuring online customer satisfaction.In addition, the EdgeComputing solution paved the way for many other current and future Sony Ericsson applications to be delivered with high quality and reach—something Sony Ericsson could not achieve without Akamai. The Akamai EdgeComputing solution supports Sony Ericsson's goal of driving site-to-store traffic through a high-quality online customer experience—all while enabling the company to simplify its e-business infrastructure and realize millions in infrastructure savings.

(continued on back)

" It's of critical importance that Sony Ericsson leverage the Internet's reach and economies of scale to enhance existing business processes and uncover new opportunities. EdgeComputing for Java will allow us to deploy applications that meet our requirements for 24/7 global business operations." —Mats Frisk, Core Application Architect, Sony Ericsson Mobile Communications

COMPANY

Sony Ericsson Mobile Communications
London, England
www.sonyericsson.com

INDUSTRY

Technology

SOLUTION

Akamai EdgeSuite®
Akamai EdgeComputing℠

KEY IMPACTS

- Improved performance worldwide by reducing dealer locator application response time by over 400%
- Increased online application availability from 92% to 100%
- Realized substantial infrastructure savings from reduced bandwidth, hardware, networking, and maintenance costs
- Offloaded nearly 100% of Java application server processing to the Akamai network
- Reduced cost per user session by 33%
- Simplified e-business infrastructure (reduced servers by 65%)
- Streamlined application maintenance and content management to support corporate goals
- Enabled faster time-to-market and lower development costs through reuse of object-oriented development components
- Helping Sony Ericsson build brand equity through premium services

How EdgeComputing Works

EdgeComputing for Java is a new deployment model for enterprise Java Web applications in which the customer selects components of the applications to run on the Akamai network, taking advantage of the global reach, on demand scalability, and improved performance it offers.

To use EdgeComputing for Java, an enterprise deploys components of its applications to Akamai's distributed network. Typically the components best-suited and supported for deployment to the edge are JSPs, servlets, and beans that contain presentation logic. Components that are tightly coupled with backend applications or a database typically remain at the origin. This two-layer model supports customer control over content and applications at the origin, while achieving superior performance and scalability at the edge.

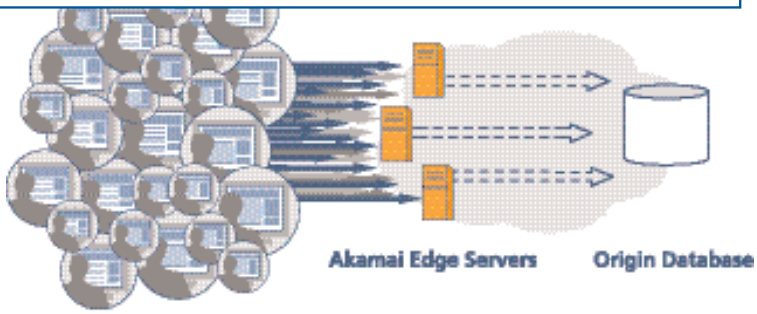
Developers assemble Java code for these components into Web archive (WAR) files, which are then deployed to the Akamai Platform. The Java environment at the edge provides a Java environment, in addition to multiple forms of back-end communication. RMI, JDBC, HTTP-based, and raw socket connections are supported, and are transparently forwarded to their destination—the origin infrastructure or elsewhere—over an HTTP tunnel. This forwarded connection takes advantage of numerous EdgeSuite optimizations for performance and reliability, as well—including persistence, Akamai's SureRoute technology, compression, automatic failover functionality, and more.

Akamai also makes available accurate anonymous geographic, network, and corporate information about end users' IP addresses for use within the application. As the Akamai network is a shared environment, careful attention has been paid to secure sandboxing of memory, CPU, and other resources to optimize the performance, security, and reliability of each application equally.

Popular Web applications that fit the two-layer deployment model include dealer or store locators, promotional contests, search functionality, online voting or polls, and registration. These are front-end applications that drive users toward purchase and other key areas of a site. They are typically one of the first interactions a customer has with a site, and often need to support large amounts of traffic produced by advertising and other promotional campaigns. Therefore, it is important to ensure their availability, scalability, and good performance.

Sony Ericsson's dealer locator exemplifies characteristics of applications that benefited enormously from EdgeComputing. A store locator running on EdgeComputing for Java can service virtually every user request completely at the edge, thus achieving maximum performance and scalability. The JSP presentation templates that dictate the look and feel of the pages have no data dependencies, and so do not need to communicate with the origin server at all. The servlets that read, validate, and process the user queries do need to contact the origin server to retrieve information from the dealer location database, but because this information changes infrequently, it can be cached at the edge in XML format and reused by subsequent requests. With very few requests requiring expensive roundtrips to the origin server, applications reap the full benefits of EdgeComputing for Java.

EdgeComputing enables end users to interact with your application running on the Akamai network, providing high performance results for users while minimizing requests to the Origin infrastructure



About Sony Ericsson Mobile Communications

Sony Ericsson Mobile Communications serves the global communications market with innovative and feature-rich mobile phones, accessories and PC-cards. Established as a joint venture by Sony and Ericsson in 2001, with global corporate functions located in London, the company employs approximately 6,000 people worldwide, including R&D sites in Europe, Japan, China and America. Sony Ericsson celebrated the 5th anniversary of the start of the joint venture on 1st October, 2006. For more information, visit **www.sonyericsson.com**.

About Akamai

Akamai® is the leading global service provider for accelerating content and business processes online. Thousands of organizations have formed trusted relationships with Akamai, improving their revenue and reducing costs by maximizing the performance of their online businesses. Leveraging the Akamai EdgePlatform, these organizations gain business advantage today, and have the foundation for the emerging Web solutions of tomorrow. Akamai is "The Trusted Choice for Online Business." For more information, visit **www.akamai.com**.

Akamai
The Trusted Choice For Online Business™

Akamai Technologies, Inc.
U.S. Headquarters
8 Cambridge Center, Cambridge, MA 02142
Tel 617.444.3000
Fax 617.444.3001
U.S. toll-free 877.4AKAMAI
(877.425.2624)

Akamai Technologies GmbH
Park Village, Betastrasse 10 b
D-85774 Unterföhring, Germany
Tel +49 89 94006-0

www.akamai.com

AKAMCS-SE1204

Friendster Increases Performance, Page Views, and Revenues Using Akamai Solution

The Situation

Founded in 2002, Friendster—which ranks as one of the top 40 sites on the Internet*—is the leading online community for making new friends, communicating with existing friends, sharing information, organizing activities, finding dates, and re-connecting with lost friends. With more than 31+ million members and growing, Friendster is benefiting from the meteoric rise of social networking on the Web. Committed to fully exploiting the potential of this new way of connecting, Friendster is constantly developing and adding technologies to help its 16 million unique global users find new ways to interact and express themselves.

The Challenge

With the growing popularity of social networking sites, Friendster realized the potential for spikes in site traffic. At the same time, it was well aware of the negative impacts of poor site performance—it could discourage site visitors, which would in turn cut into advertising revenues. The company knew it would need to add infrastructure on an on-going basis to support its anticipated growth and yet it did not want to incur the associated costs and management overhead.

The Goal

Friendster needed to meet three key requirements to support its brand and objectives:

- **Reduced Infrastructure Needs** Friendster wanted to support any number of members without incurring additional co-location and web server costs.
- **Support Peak Traffic Load** The company's site needed to scale to support peak traffic that could potentially overload its existing infrastructure.
- **Serve Media-rich Content** Friendster needed to support members' use and sharing of more media-rich content.

Why Akamai

Increasing Membership without Adding Infrastructure

Due to the large and varied nature of its membership, Friendster must store and deliver a considerable number of small files which are not requested very frequently. Fortunately, Akamai's network provides the intelligence and processing flexibility to handle content with such traffic patterns. Since implementing Akamai's solution, Friendster has offloaded a significant percentage of the total traffic from its origin servers. In fact, it realized about a 50% decrease in load on the servers that host photos and avoided the need to purchase additional, costly infrastructure. "Leveraging the market leader in content delivery allows us to ensure that as demand for our site grows, we can handle it and continue to expand our services," said Chander Sarna, VP Engineering and Operations.

(continued on back)

** As measured by Alexa.com on August 1, 2006*

"Akamai's services were a primary factor in our ability to steadily increase overall daily page views in March. Our sales team seized on the opportunity to monetize this additional traffic, and the result was increased revenue in a short period of time."

—Aaron Barnes, VP Sales, Friendster

COMPANY
Friendster.com
San Francisco, CA
www.friendster.com

INDUSTRY
Media & Entertainment

SOLUTION
Akamai Content Delivery and Streaming

KEY IMPACTS

- Significant revenue increase in a short period of time
- 33% improvement in page response time
- Tripled page views in a single month
- Decreased server load by 50%
- Reduced bandwidth usage by 50%
- Gained ability to deliver richer content

Why Akamai *(continued)*

More Page Views Means More Revenues

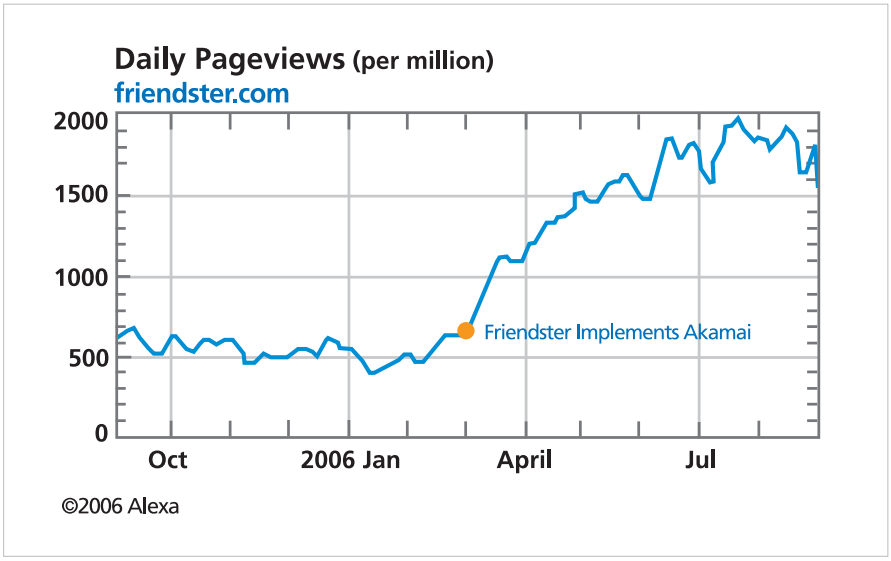
After implementing Akamai global content delivery and managed storage services, the images on Friendster's site load faster, which in turn means that the entire page renders more quickly. According to Sarna, "We had a seamless integration onto the Akamai platform—literally plug and play within minutes—and immediately saw a 33% percent improvement in the overall page loading times."

As an advertising-supported site, increased performance enables Friendster to serve more page views and generate more revenue. In fact, Friendster tripled page views in about one month, to more than 27.5 million page views.

Satisfying Member Demands

As a growing number of social network members become content creators and publishers, Friendster faces the increasingly difficult task of planning, provisioning, maintaining, securing, and scaling its systems to meet member demands. This problem is compounded by the increased utilization and sharing of more media-rich content.

Friendster plans to use Akamai's streaming solution to offer users the ability to add streaming audio and video to their online profiles. Using Akamai's turnkey solutions, Friendster can focus on core functionality and applications, while dramatically decreasing time-to-market.



Friendster's page views climbed significantly in part due to implementing Akamai in March 2006

About Friendster.com

With more than 31 million members, Friendster is a leading social network. Friendster is focused on helping people stay in touch with friends and discover new people and things that are important to them. Friendster's trusted network, which emphasizes genuine, real-world friendships, is popular among people of all ages, but is particularly used by young urban adults. Headquartered in San Francisco, California, Friendster is backed by Kleiner Perkins Caufield & Byers, Benchmark Capital, and individual investors. For more information, visit **www.friendster.com**.

About Akamai

Akamai® is the leading global service provider for accelerating content and business processes online. Thousands of organizations have formed trusted relationships with Akamai, improving their revenue and reducing costs by maximizing the performance of their online businesses. Leveraging the Akamai EdgePlatform, these organizations gain business advantage today, and have the foundation for the emerging Web solutions of tomorrow. Akamai is "The Trusted Choice for Online Business." For more information, visit **www.akamai.com**.

Akamai
The Trusted Choice For Online Business™

Akamai Technologies, Inc.
U.S. Headquarters
8 Cambridge Center, Cambridge, MA 02142
Tel 617.444.3000
Fax 617.444.3001
U.S. toll-free 877.4AKAMAI (877.425.2624)

European Headquarters
Park Village, Betastrasse 10 b
D-85774 Unterföhring, Germany
Tel +49 89 94006-0

www.akamai.com

AKAMCS-FRIENDSTER-0906

Auto-Evaluación

1. ¿Cuál de los siguientes puntos no es una ventaja de las CDNs?
   a. Mejor utilización de la red
   b. Mejores tiempos de respuesta
   c. Mejor utilización de disco

2. ¿Cuál de los siguientes puntos no es una ventaja los switches de capa 5 (o también llamados, switches basados en URL)?
   a. Mejor utilización de disco
   b. Mejor rendimiento caché
   c. Mejor balanceo de carga

3. ¿Cuál de los siguientes enunciados no es cierto con respecto a los switches de capa 7?
   a. Implementan la máquina de estados TCP
   b. Pueden ser usados en flujos dentro de túneles IPsec
   c. Pueden interpretar protocolos de aplicación como HTTP
   d. Modifican las cabeceras de los paquetes que van en ambas direcciones

4. ¿Cuál de lo los siguientes puntos es una desventaja de los sistemas P2P?
   a. Baja escalabilidad
   b. Hay un punto único de fallo
   c. Dificultad de administración
   d. Susceptible a cuellos de botella en la red

5. La versión original de Gnutella era ____.
   a. una red P2P pura
   b. una red P2P híbrida
   c. una DHT
   d. una red P2P para colaboración