Sistemas Distribuidos

Dra. Cristina Abad R.

2017

Sobre esta clase

- Los sistemas distribuidos están en todas partes
- Objetivo:

  Aprender a diseñar e implementar sistemas distribuidos medianos y grandes, así como entender los principios y desafíos de los mismos
- Pre-requisitos
  - Sistemas operativos (recomendado)
  - Programación orientada a objetos (ej.: Java)
  - Fundamentos de redes de datos o equivalente

¿Qué van a aprender?

- Dificultades al desarrollar sistemas distribuidos
- Objetos y eventos distribuidos
- Tecnología middleware (sockets, RMI, pub/sub, etc.)
- Modelos de descomposición de tareas paralelas
- Localidad de datos
- Redes de distribución de contenidos
- Seguridades
- Bases teóricas

Metodología

- Calificaciones

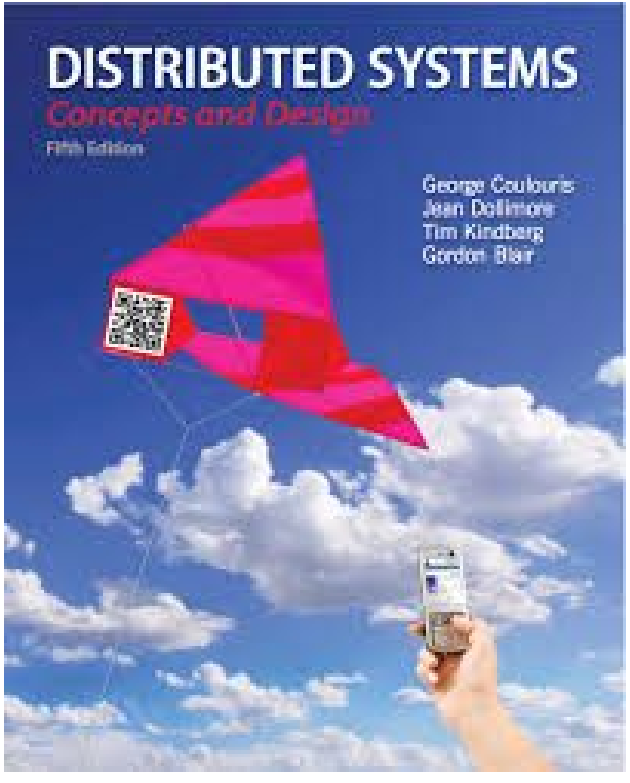
Proyecto	--	30%
Tareas, asistencias, actividades	10%	2%
Laboratorios	40%	18%
Examen	50%	50%

- Información en línea: SidWeb
- Consultas: cabad@fiec.espol.edu.ec o cabadr@espol.edu.ec
- Más detalles en el documento de políticas

Libro

Distributed Systems: Concepts and Design (5th Edition) 5th Edition

by George Coulouris (Author), Jean Dollimore (Author), Tim Kindberg (Author), Gordon Blair (Author)



Presentaciones

- Nombre
- Carrera
- Lugar de trabajo
- Recomiende una película
- Recomiende un libro

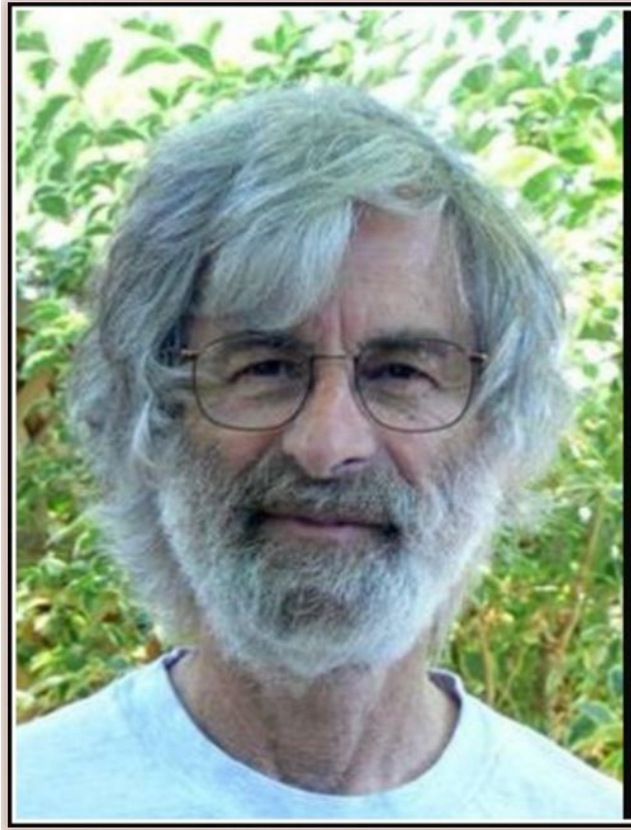
About me

- Nombre
  - Cristina Abad Robalino
- Carrera
  - Ing. en Sistemas Computacionales, MS in CS, PhD in CS (UIUC)
- Lugar de trabajo
  - ESPOL; antes: NCSA, UIUC, Yahoo
- Recomiende una película
  - The Martian, The Usual Suspects, Blade Runner, As good as it gets, Transpoitting
- Recomiende un libro
  - 1984, Un mundo feliz, The cuckoo’s egg, Do androids dream of electric sheep?

Introducción

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.



A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

— Leslie Lamport —

AZ QUOTES

¿Ejemplos?

- Hadoop
- Dropbox (**cloud computing**)
- Netflix
- BitTorrent
- Whatsapp
- Redis
- MongoDB
- Bitcoin
- Minecraft
- Internet of Things (IoT)

Características (vs. sistemas no distribuidos)

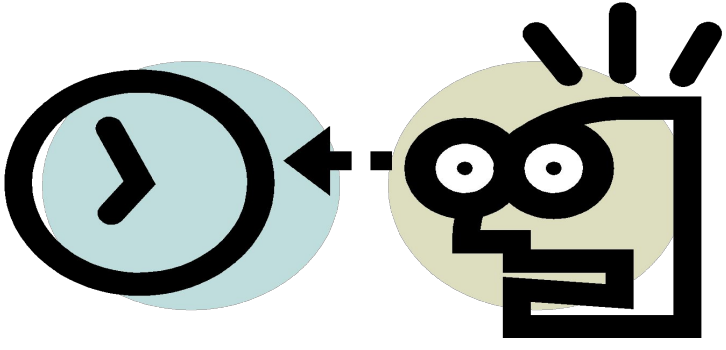
1. Concurrencia de componentes
2. No hay un reloj global
3. Independencia de las fallas de los componentes

1. Concurrencia

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

2. 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



3. Fallas independientes

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

¿Por qué construir un sistema distribuido?

- Compartir recursos
- Mayor rendimiento
- Menores tiempos de respuesta
- Mejor tolerancia a fallos
- Alta disponibilidad
- Anonimidad

Sesión 2

Anuncios

- Publiqué en SidWeb:
  - Nuevo Syllabus
  - Documento de “Apuntes de Sistemas Distribuidos”
    - Pero... no ha sido actualizado :-(
  - Pendiente: Políticas del curso

Definición: 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: altamente distribuidos
  - Peer-to-peer, multi-capas

Desafíos

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

También:

- Balanceo de cargas
- Coordinación y acuerdo
- Consistencia
- Rendimiento

Desafío: Componentes heterogéneos

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

¿Soluciones?

- Estándares
- Middlewares
- Código portable (Java)
- Ejemplos concretos:
  - XML, JSON
    - Usados en Web Services, REST
    - Texto plano
    - Ineficientes
  - Protocol Buffers, Apache Thrift
    - En combinación con Lightweight RPCs

Definición: Middleware

- Capa de SW que proporciona abstracción de programación y esconde heterogeneidad de redes, S.O, y lenguajes de prog.
  - Ej.: CORBA, Java RMI, ODBC, JDBC, etc.
- Generalmente se implementa sobre protocolos de Internet, que esconden la heterogeneidad de red, representación de datos, etc.
- Modelos:
  - Remote method invocation
  - Remote event notification
  - Remote SQL access
  - Transacciones
  - Paso de mensajes
  - Publish-subscribe

"Software glue"

Desafío: 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

Lectura recomendada: http://www.infoq.com/articles/API-Design-Joshua-Bloch

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

1. INTRODUCTION

Desafío: Seguridad

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

Desafío: 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
- Es difícil planificar para un futuro incierto
  - Flash crowds (multitudes repentinas)
  - Efecto “slashdot”:
    - https://en.wikipedia.org/wiki/Slashdot_effect
- Posible solución: Sistemas elásticos

http://www.cnn.com/ - Mi

File Edit View Favorites Tools Help

Back Search Favorites History

Address http://www.cnn.com/ Go Links

CNN.com

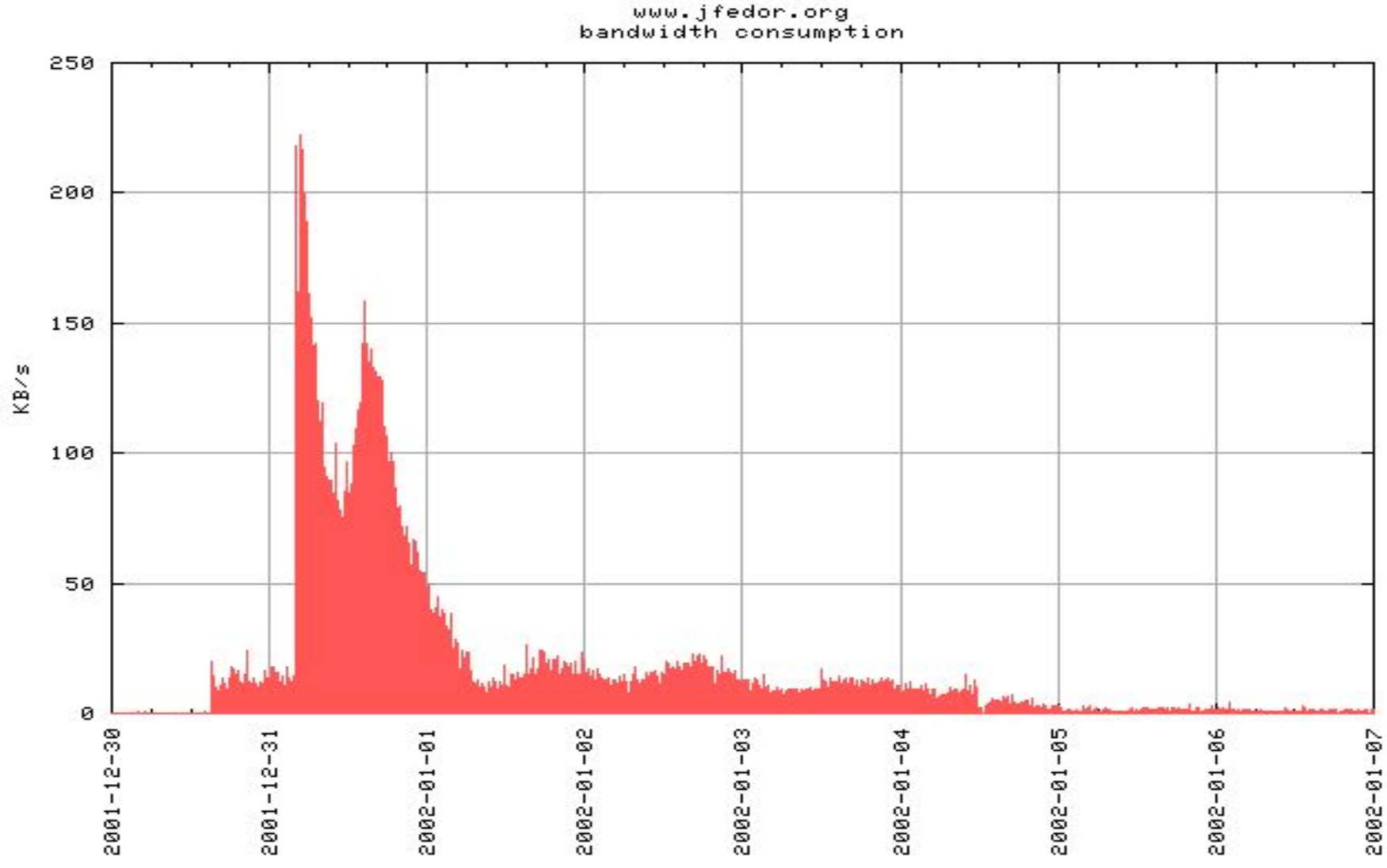
BREAKING NEWS

AMERICA UNDER ATTACK

- Both World Trade Center towers collapse after plane crashes
- Plane crashes into Pentagon, part of the Pentagon collapses
- Bush calls trade center crashes terrorist act
- Major federal buildings, United Nations evacuated
- FAA grounds all domestic flights, sends trans-Atlantic flights to Canada
- Israel evacuates embassies

DEVELOPING STORY »

Internet


www.jfedor.org
bandwidth consumption
KB/s
250
200
150
100
50
0
2001-12-30
2001-12-31
2002-01-01
2002-01-02
2002-01-03
2002-01-04
2002-01-05
2002-01-06
2002-01-07

Slashdot

News for Nerds. Stuff that matters.



Login
Why Login?
Why Subscribe?

Sections
Main
Apache
Apple
1 more
Askslashdot
7 more
Books
BSD
Developers
7 more
Games
10 more

Google Traffic Takes Down Web Site

Posted by simoniker on Wednesday February 04, @09:11PM

from the comparisons-inevitable dept.

bazonkers writes "*Searchenginelowdown.com reports that it appears that the Google logo yesterday (honoring Gaston Julia) linked to the Google image search results for the words 'julia fractal'. The resulting traffic generated from clicking on that 'featured logo' incapacitated the servers of the top-listed images, hosted at an Australian university. This more than inconvenienced the owners of that site, who had to move pages and ended up displaying this page instead.*"



Desafío: Manejo de fallas ← Lo veremos en la Unidad 2

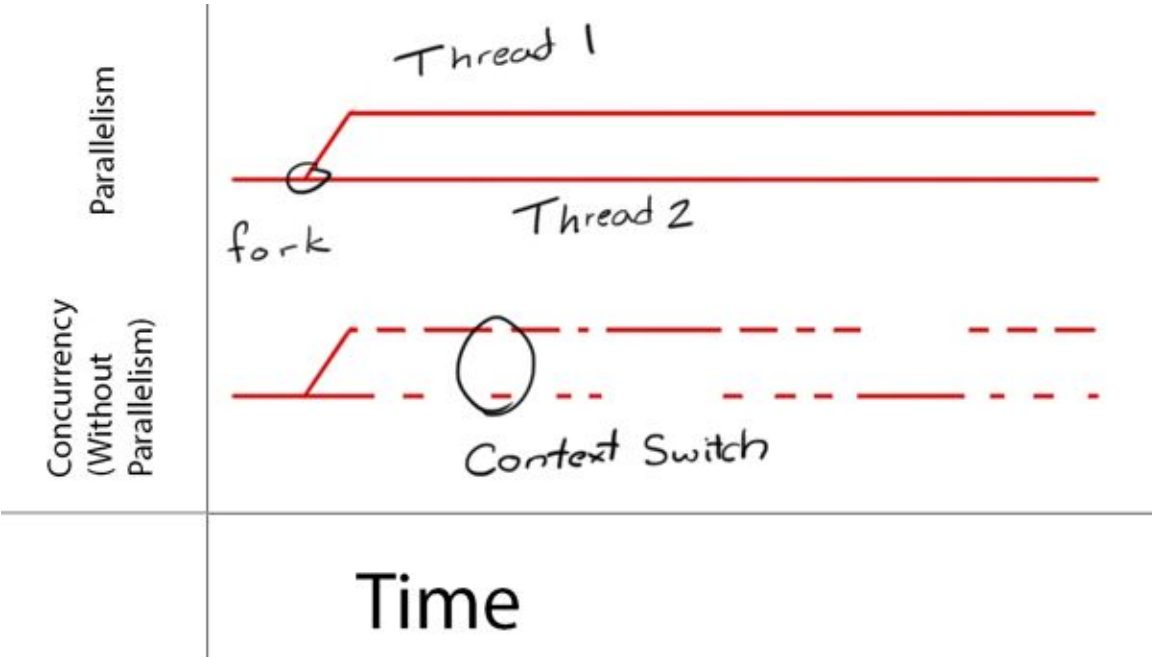
- Detectar
- Enmascarar
- Tolerar
- Recuperarse de
- Redundancia

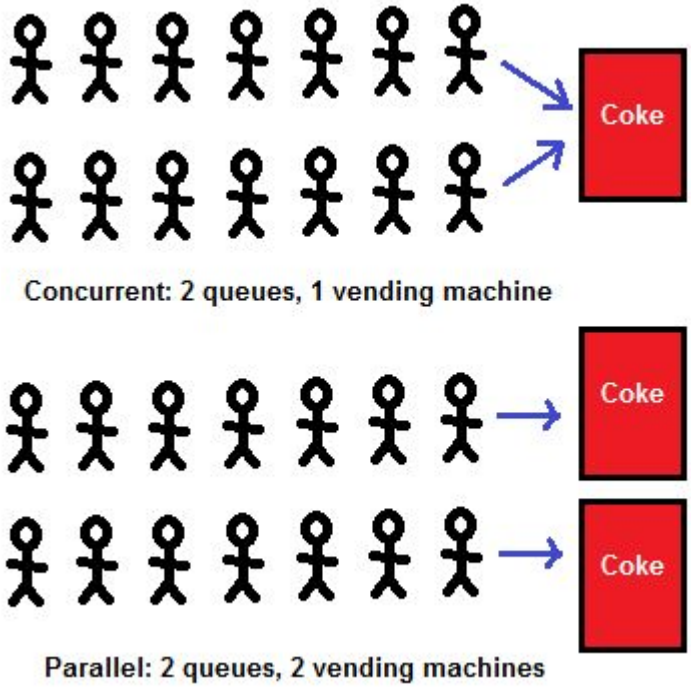
Desafío: Concurrencia

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

- Soluciones:
  - Sincronización
  - Transacciones con rollback & recovery
  - Eliminar concurrencia en recursos compartidos (ej.: MapReduce)
  - Relajar consistencia (weak consistency)

Paralelismo vs. Concurrencia





Desafío: 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

Modelos

(Sesión 3)

Entidades y paradigmas de comunicación

Communicating entities (what is communicating)		*Communication paradigms (how they communicate)*		
System-oriented entities	*Problem-oriented entities*	*Interprocess communication*	*Remote invocation*	*Indirect communication*
Nodes	Objects	Message passing	Request-reply	Group communication
Processes	Components	Sockets	RPC	Publish-subscribe
	Web services	Multicast	RMI	Message queues
				Tuple spaces
				DSM

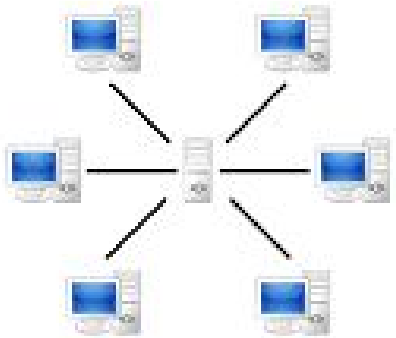
Modelos

- Arquitectónico
  - ¿Dónde colocamos las partes?
  - ¿Cómo se relacionan las partes?

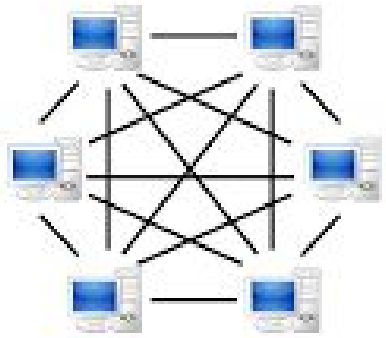
- Fundamental
  - Descripción formal de las propiedades comunes a todos los modelos arquitectónicos

Modelo arquitectónico

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



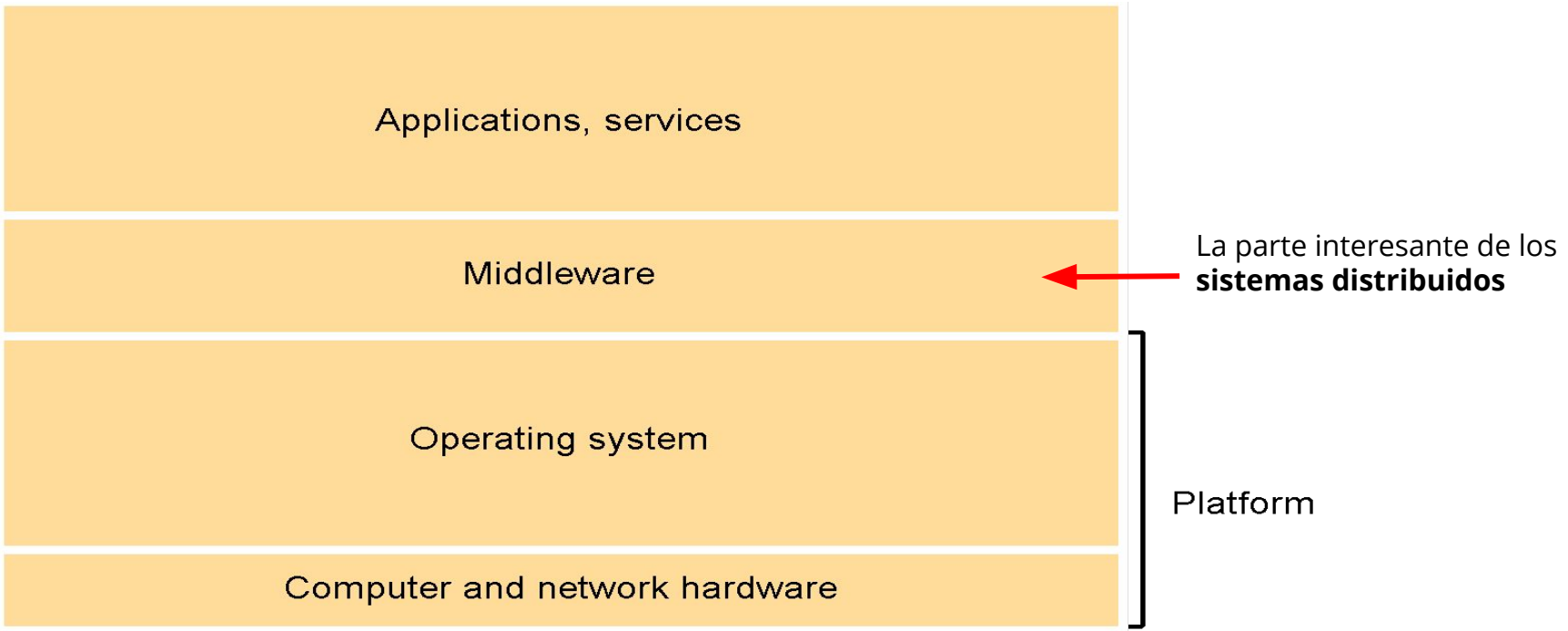
Server-based



P2P-network

Modelo arquitectónico

- Sistemas se organizan en **capas de software**:



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 sí para implementar comunicación y soporte de recursos compartidos para sistemas distribuidos
- Bloques útiles para construir SW distribuido

Middleware (cont...)

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

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.

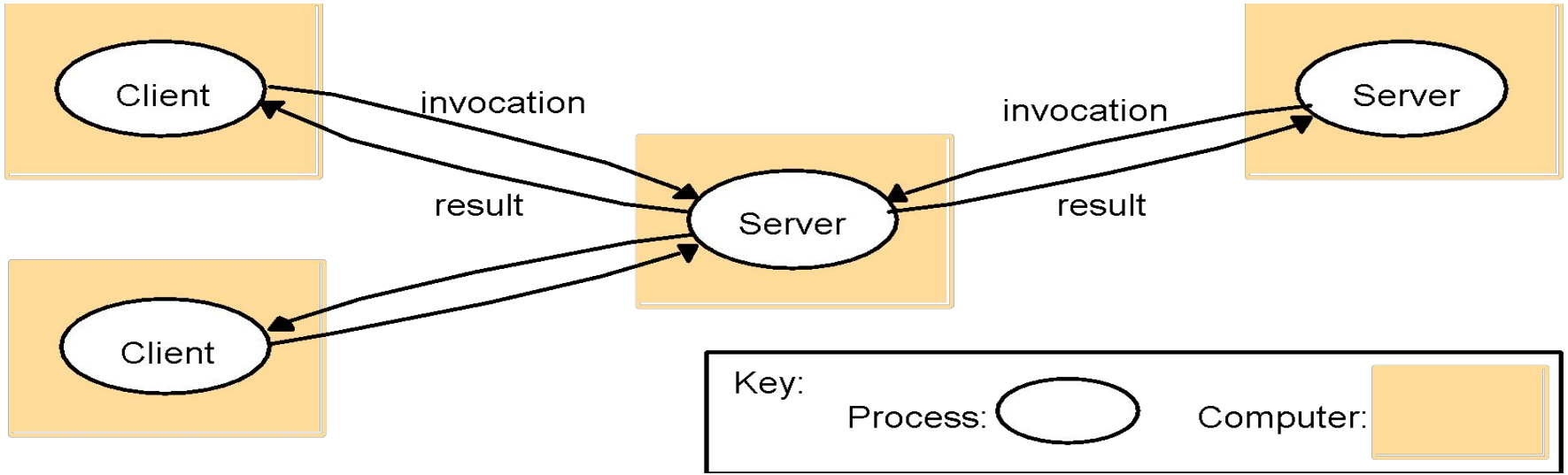
Lectura recomendada: The end-to-end argument in systems design (http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf)

Principales modelos arquitectónicos

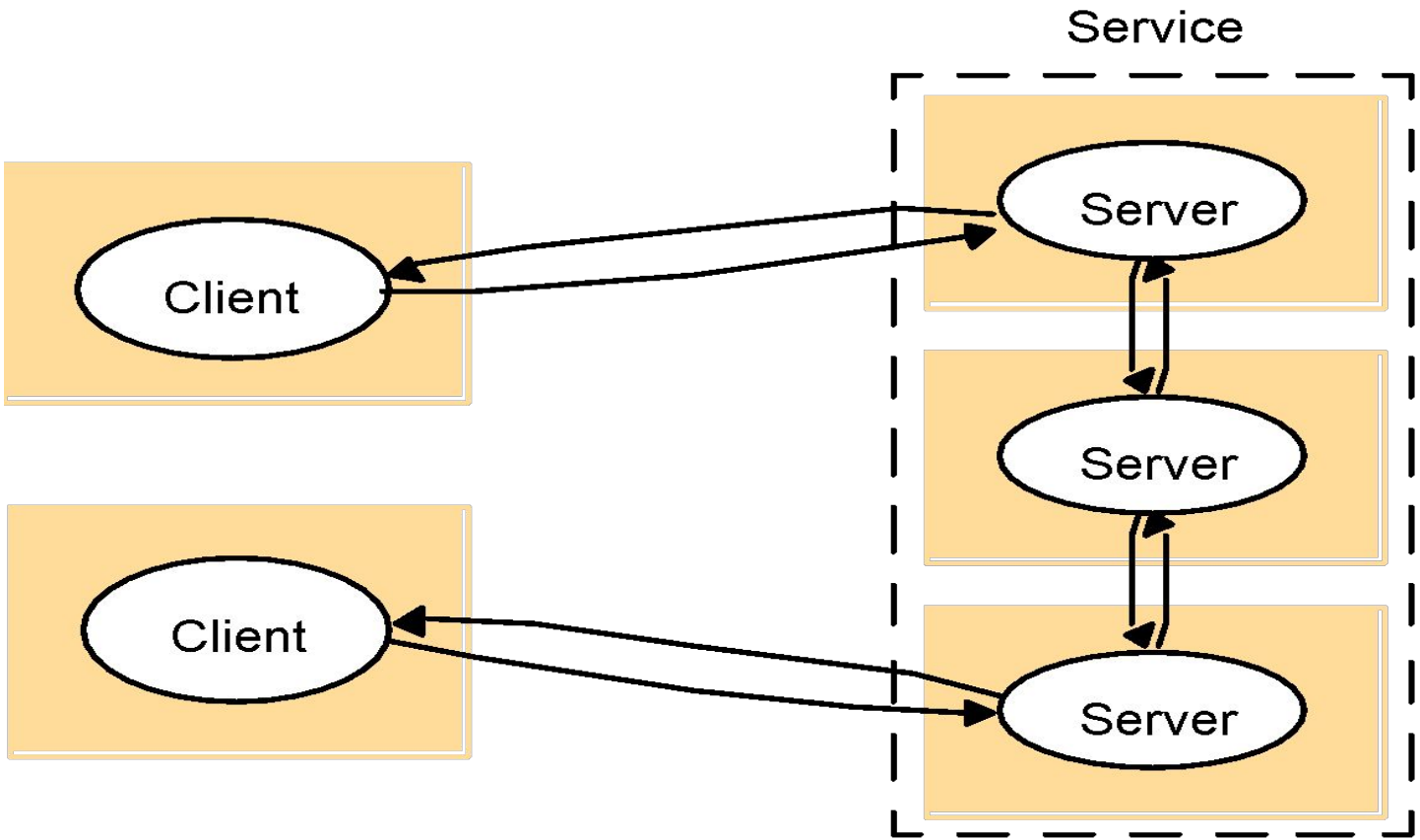
- Cliente-servidor
- Servidores múltiples
- Servidores proxy y cachés
- Procesos compañeros (peer)

Cliente/servidor

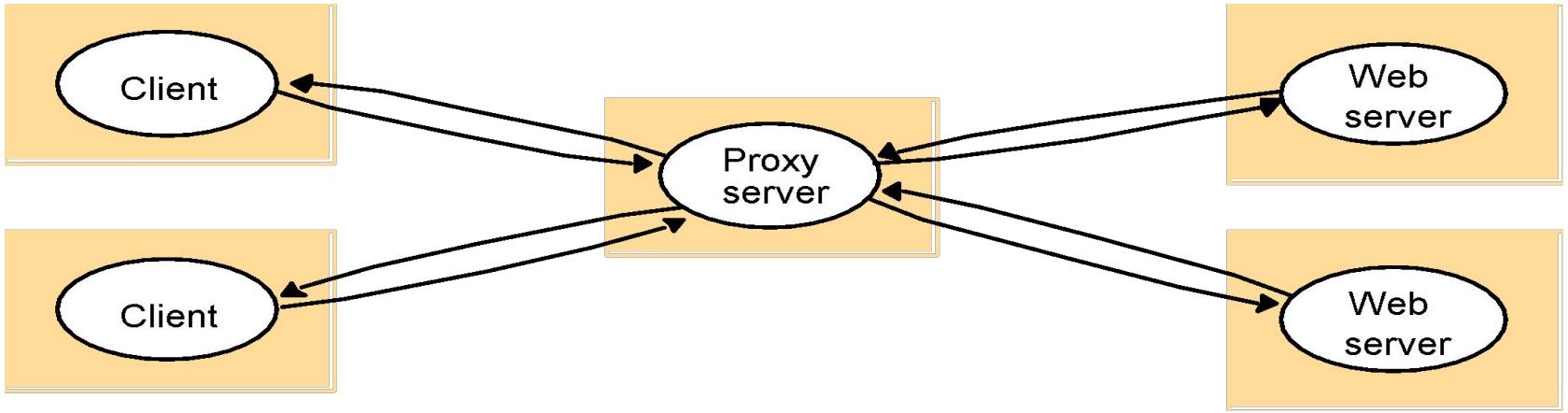
Middleware



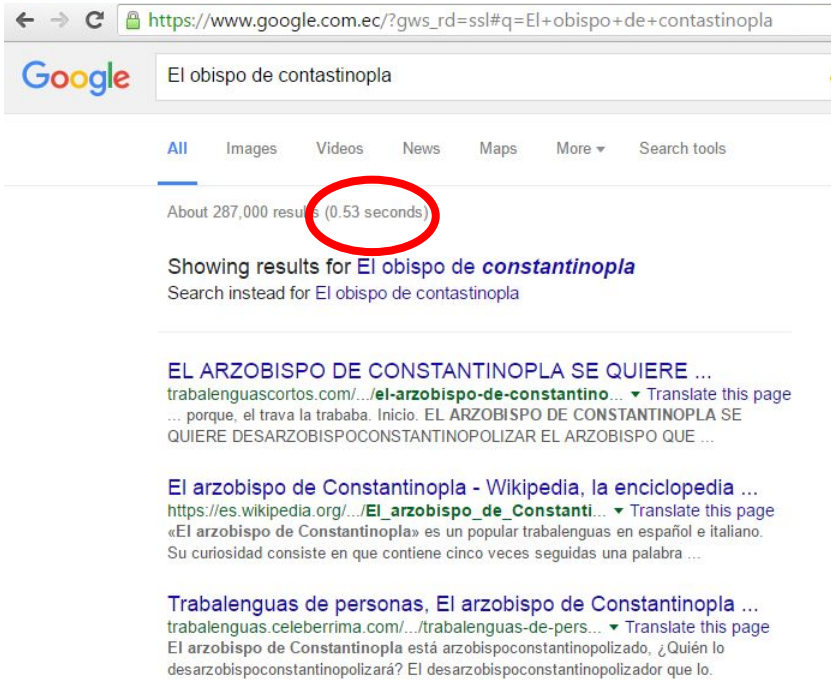
Múltiples servidores

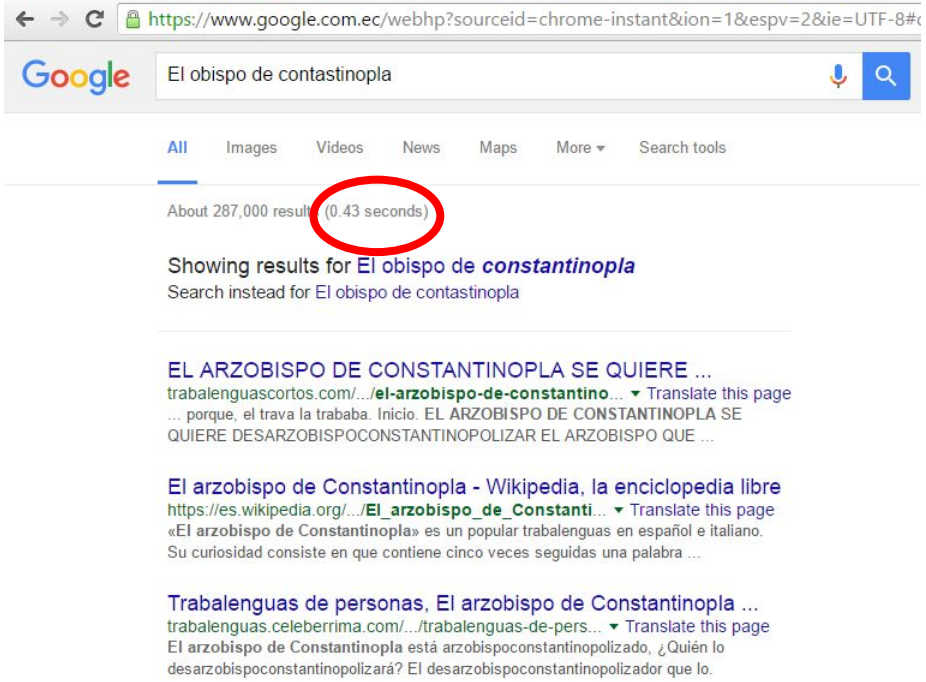


Servidores proxy y cachés

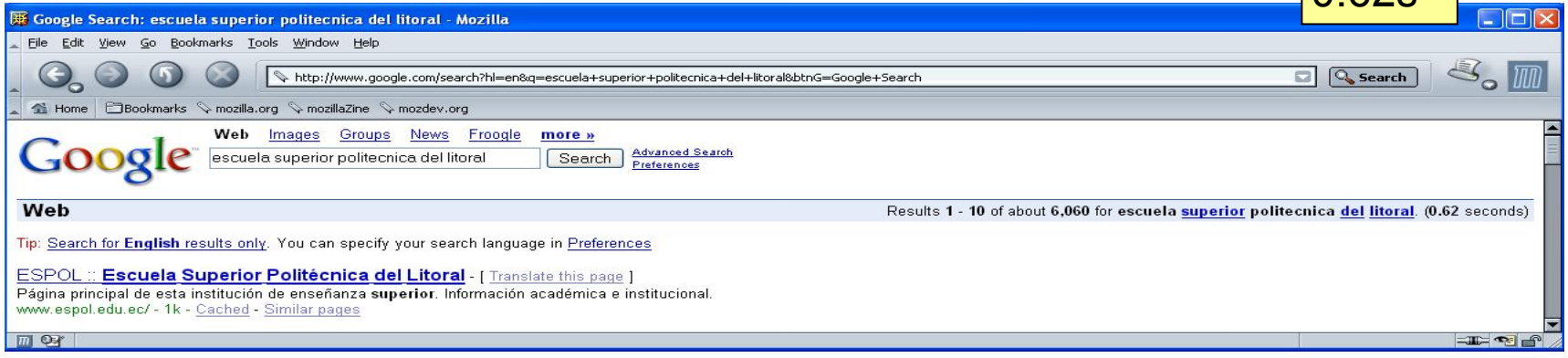


Ejemplo: Cachés

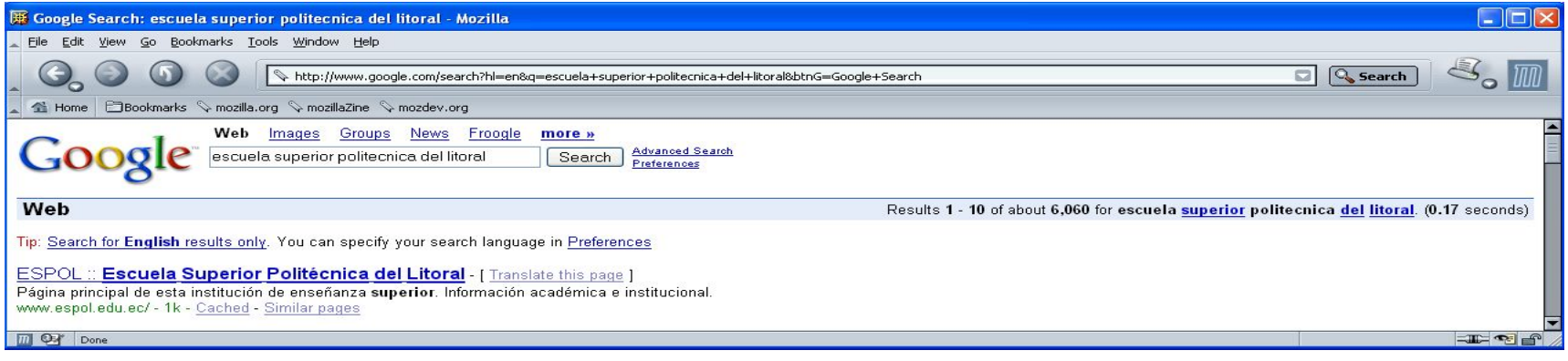




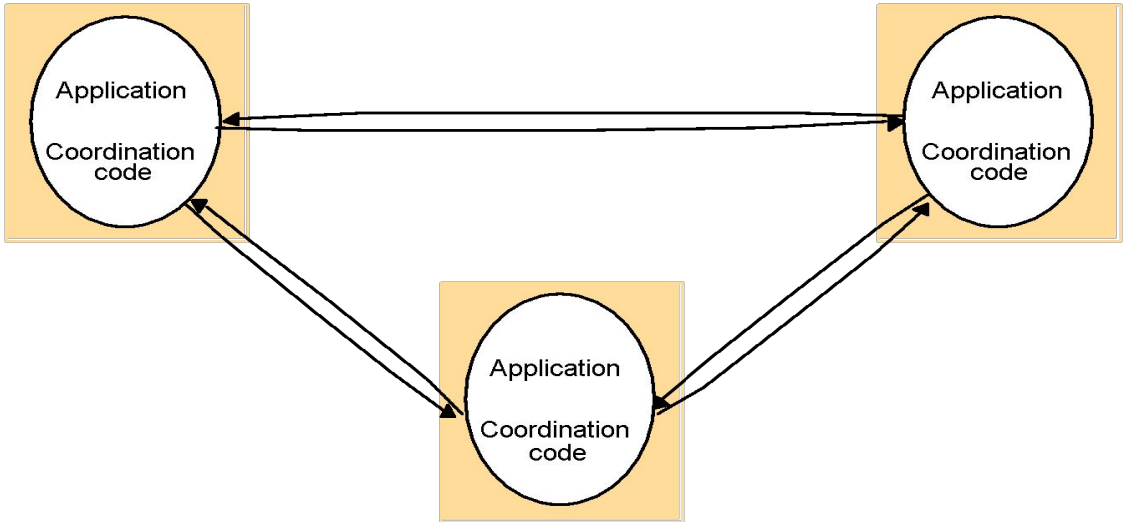
0.62s


Google Search: escuela superior politecnica del litoral - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.google.com/search?hl=en&q=escuela+superior+politecnica+del+litoral&btnG=Google+Search
Search
Home Bookmarks mozilla.org mozillaZine mozdev.org
Google
Web Images Groups News Froogle more »
escuela superior politecnica del litoral
Search
Advanced Search
Preferences
Web
Results 1 - 10 of about 6,060 for escuela superior politecnica del litoral. (0.62 seconds)
Tip: Search for English results only. You can specify your search language in Preferences
ESPOL :: Escuela Superior Politécnica del Litoral - [Translate this page]
Página principal de esta institución de enseñanza superior. Información académica e institucional.
www.espol.edu.ec/ - 1k - Cached - Similar pages

0.17s


Google Search: escuela superior politecnica del litoral - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.google.com/search?hl=en&q=escuela+superior+politecnica+del+litoral&btnG=Google+Search
Search
Home Bookmarks mozilla.org mozillaZine mozdev.org
Google
Web Images Groups News Froogle more »
escuela superior politecnica del litoral
Search
Advanced Search
Preferences
Web
Results 1 - 10 of about 6,060 for escuela superior politecnica del litoral. (0.17 seconds)
Tip: Search for English results only. You can specify your search language in Preferences
ESPOL :: Escuela Superior Politécnica del Litoral - [Translate this page]
Página principal de esta institución de enseñanza superior. Información académica e institucional.
www.espol.edu.ec/ - 1k - Cached - Similar pages
Done

Procesos compañeros (peer-to-peer, P2P)



Para mejorar el rendimiento, se suele tener sistemas P2P híbridos en el que ciertos nodos asumen el rol de *super nodos*.

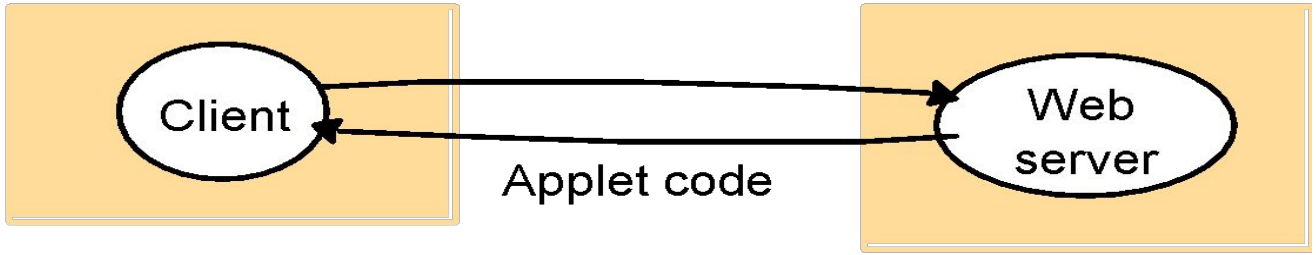
Sesión 4

Variantes modelo cliente-servidor

- Código móvil
- ~~Computadoras en red (networked computers)~~
- Clientes ligeros (thin clients)
- Redes ad-hoc o espontáneas
- Plataformas de computación distribuida
- Variantes de computación en las nubes

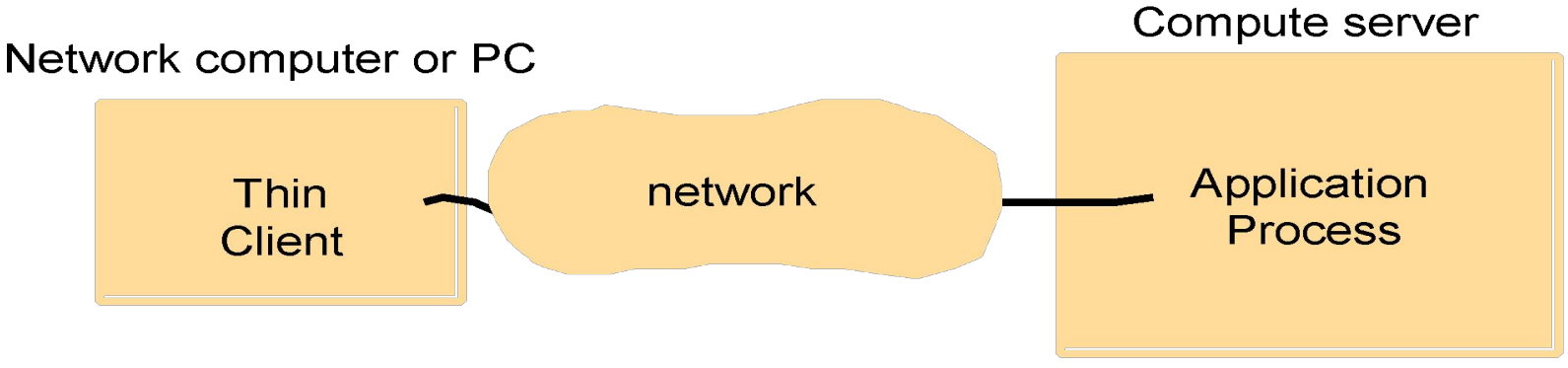
Código móvil

a) client request results in the downloading of applet code



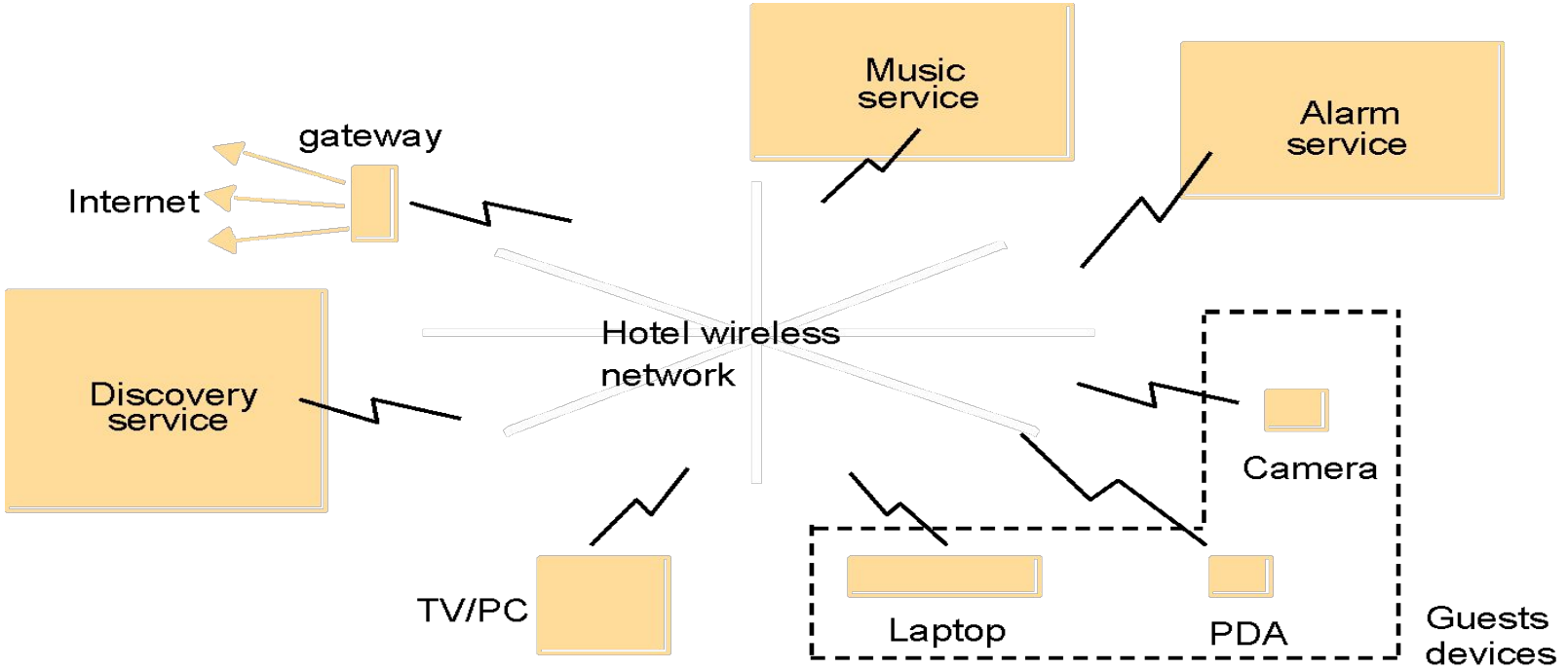
b) client interacts with the applet

Clientes ligeros (thin clients)



- Ej: X11, NX, VNC, TeamViewer

Redes ad-hoc (espontáneas)

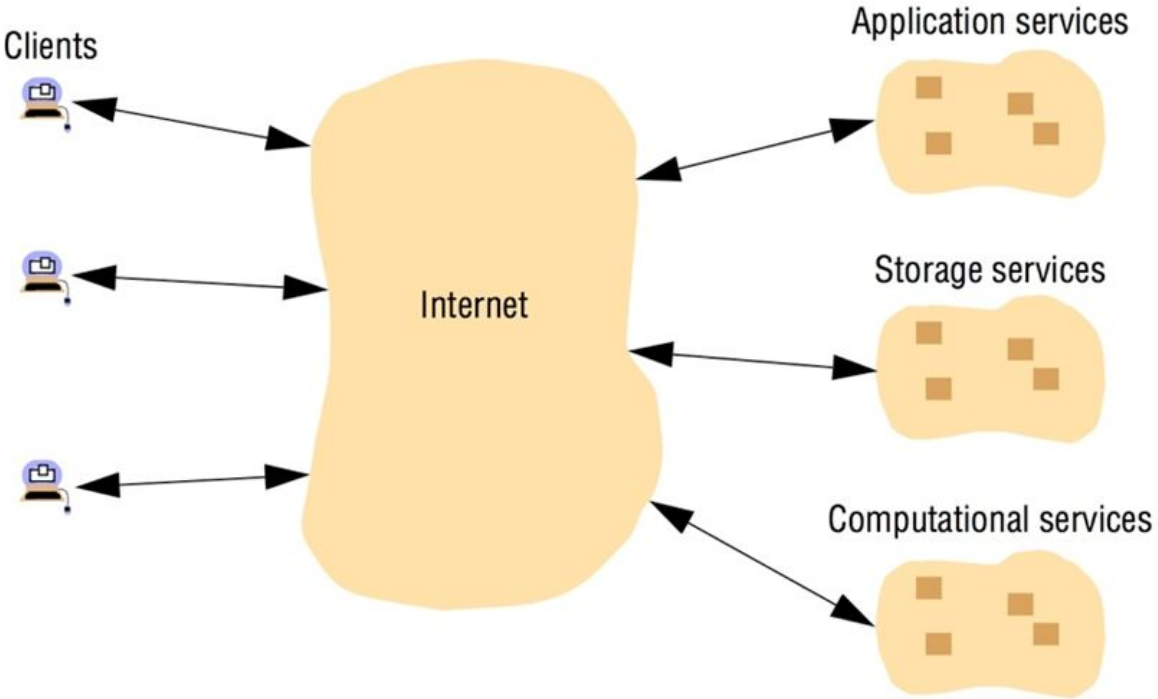


- Ubiquituous computing (computación ubícua), Pervasive computing, IoT

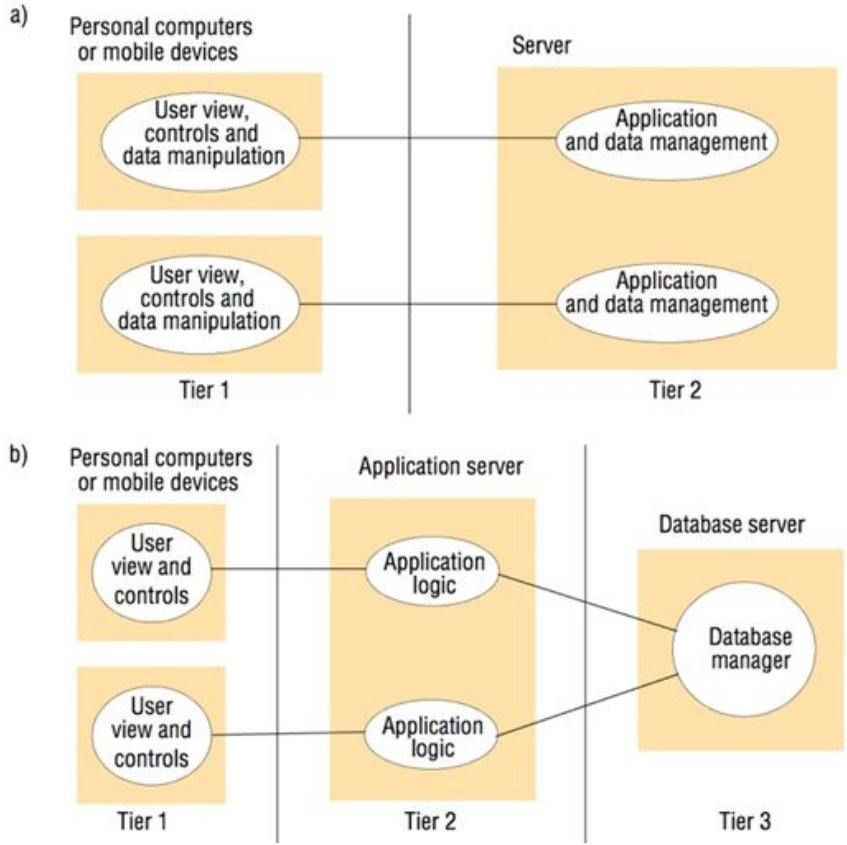
Plataformas de computación distribuida

- Se distribuye código (y, de ser necesario, datos)
- Ejemplos:
  - Volunteer computing
    - BOINC
  - Cluster computing (ubicados en data centers)
    - Apache Hadoop
    - Apache Storm
    - Apache Spark
    - Google's MapReduce
- ¿Problemas de **seguridad**?

Computación en las nubes



Arquitecturas por capas



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

Modelos fundamentales

- Modelo de interacción
- Modelo de fallas ← Lo veremos en la Unidad 2
- Modelo de seguridades ← Lo veremos en la Unidad 2

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

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

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

¿Preguntas?

Discusión

- Un sistema P2P híbrido tiene elementos centralizados
- Por ejemplo, Napster dependía de un servidor de directorios centralizado que mantenía una lista de archivos disponibles en los clientes conectados
  - Cada búsqueda de un cliente iba inicialmente al servidor, el cual la procesaba y respondía con una lista de los clientes disponibles que mantienen el archivo buscado
- Discuta las limitaciones de este enfoque centralizado, en un ambiente a gran escala como Internet
- Otro ejemplo: Skype