

tema

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002.

Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

buidos

Arquitectura de los Sistemas Distribuidos

Plamamiento

la interacción entre entidades
coplamiento espacial y temporal
referencia)
e referencia directa a la otra entidad
nos frecuente)
cionan **no** tienen que coincidir
: memoria compartida
ndientes entre sí
rectos" proporcionan flexibilidad
subrutina):
e can be solved by another level of
m of too many layers of indirection."

Fernando Pérez Costoya

Índice

- Introducción
- Arquitecturas para computación distribuida
 - Arquitecturas de computación en Google
 - Modelo Map-Reduce y Pregel
- Arquitectura cliente-servidor
 - Variaciones del modelo
 - Aspectos de diseño del modelo cliente/servidor
- Arquitectura editor-subscriptor
- Arquitectura *Peer-to-peer*
 - Sistemas P2P desestructurados
 - Sistemas P2P estructurados
 - Protocolo Chord

Sistemas Distribuidos2

Fernando Pérez Costoya

Arquitecturas para CD

- Maestro-trabajador *M/T* (aka maestro-esclavo)
 - *M* va repartiendo trabajos entre nodos trabajadores *T*
 - Si n° trabajos $\gg n^{\circ}$ trabajadores $\rightarrow$ reparto automático de carga
 - Tolerancia a fallos:
 - Caída de *T*: *M* reasigna sus trabajos pendientes (¡efectos laterales!)
 - Caída de *M*: punto crítico de fallo
- Arquitecturas guiadas por "geometría" de los datos
 - Matrices multidimensionales, grafos, etc.
 - P.e. Matriz 2D
 - Cada nodo se encarga de sub-matriz
 - Comunicación más frecuente con nodos "vecinos cartesianos"
 - P.e. Grafo
 - Cada nodo se encarga de un sub-grafo
 - Comunicación siguiendo aristas

Sistemas Distribuidos5

Fernando Pérez Costoya

Arquitectura de los SD

- Organización lógica de componentes de aplicación distribuida
 - Cómo es su patrón de interacción
 - Qué roles ejercen los procesos involucrados
 - Y cuál es su correspondencia con nodos de SD físico
 - "Topología" de la aplicación distribuida
- En principio, tantas como aplicaciones
 - Pero hay patrones que se repiten de forma habitual
- Arquitecturas más frecuentes en SD de propósito general
 - Cliente/servidor
 - Editor/subscriptor
 - *Peer-to-peer* (Paritaria)
- Computación distribuida (CD) presenta sus propios patrones
 - Maestro/trabajador
 - Arquitecturas guiadas por la "geometría" de los datos

Sistemas Distribuidos3

Fernando Pérez Costoya

Arquit. de computación en Google

- *MapReduce*
 - Maestro-trabajador con dos fases: *Map* y *Reduce*
 - *Map*: *T* procesa su parte de datos de entrada y genera (*clave, valor*)
 - P.ej. Extrae de logs web $\rightarrow$ (página, usuario que la accede)
 - *Reduce*: *T* procesa valores asociados a una determinada clave
 - P.ej. Calcula n° accesos únicos a cada página $\rightarrow$ (página, n° accesos)
- *Pregel*
 - Modelo diseñado para procesar grafos de gran escala
 - Computación organizada en "*supersteps*" sincronicos:
 - Cada vértice recibe datos de otros vértices por aristas de entrada
 - Cambia su estado y genera datos por vértices de salida
 - Incluso puede cambiar topología del grafo
 - Inspirado en modelo "*Bulk Synchronous Parallel*" de Valiant
 - Implementado como arquitectura maestro/trabajador
 - *M* reparte grafo entre *T* y controla sincronización de "*supersteps*"

Sistemas Distribuidos6

Fernando Pérez Costoya

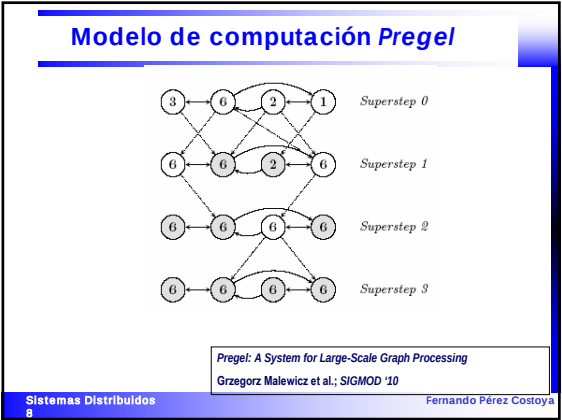
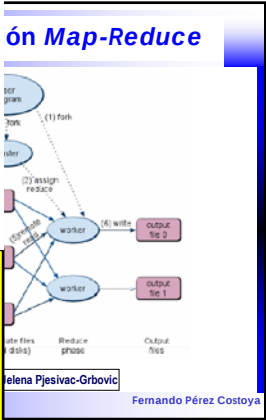
tema

Cartagena99

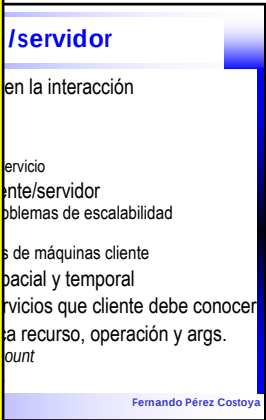
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.



- ### Arquitecturas en SD de propósito general
- Cliente-servidor
 - Extensión del modelo proveedor/consumidor de servicio a SD
 - Similar a biblioteca de servicio y programa que la usa pero en SD
 - Interacción 1-N
 - Editor/subcriptor
 - Extensión de esquema guiado por eventos a SD
 - Facilita el desacoplamiento espacial y, potencialmente, el temporal
 - Interacción M-N
 - Peer-to-peer
 - Procesos cooperantes con el mismo rol
 - Interacción N-N



- ### Reparto funcionalidad entre C y S
- ¿Qué parte del trabajo realiza el cliente y cuál el servidor?
 - "Grosor del cliente": Cantidad de trabajo que realiza
 - Pesados (*Thick/Fat/Rich Client*) vs. Ligeros (*Thin/Lean/Slim Client*)
 - Ventajas de clientes ligeros
 - Menor coste de operación y mantenimiento
 - Mejor seguridad
 - Ventajas de clientes pesados
 - Mayor autonomía
 - Mejor escalabilidad
 - Cliente gasta menos recursos de red y de servidor
 - Más ágil en respuesta al usuario
 - Ej. "inteligencia en cliente": Javascript valida letra NIF en form.

entre C y S

basada en 3 capas:
gráfica: GUI)

e asigna a máquina cliente?

ie:
e servidor gráfico (p.e. X11 o VNC)
do y recibe de app. info. a dibujar en:
ficas (X11)

io

de lógica de acceso

Fernando Pérez Costoya

Cliente/servidor con caché

- Mejora latencia, reduce consumo red y recursos servidor
- Aumenta escalabilidad
 - Mejor operación en SD → La que no usa la red
- Necesidad de coherencia: sobrecarga para mantenerla
 - ¿Tolera el servicio que cliente use datos obsoletos?
 - SFD normalmente no; pero servidor de nombres puede que sí (DNS)
- Puede posibilitar modo de operación desconectado
 - CODA
 - HTML5 *Offline Web Applications*
- *Pre-fetching*: puede mejorar latencia de operaciones pero
 - Si datos anticipados finalmente no requeridos: gasto innecesario
 - Para arreglar la falacia 2 hemos estropeado la 3

Sistemas Distribuidos 14 Fernando Pérez Costoya

Cliente/servidor con proxy

- Componentes intermediarios entre cliente y servidor
- Actúan como “tuberías”
 - Pueden procesar/filtrar información y/o realizar labor de caché
 - Simil con clases *FilterInputStream*/*FilterOutputStream* de Java
- Diversos tipos: *forward proxy*, *reverse proxy*, *gateways*, ...
- Interfaz de servicio de proxy debe ser igual que el del servidor:
 - *Proxy* se comporta como cliente y servidor convencional
 - Se pueden “enganchar” sucesivos *proxies* de forma transparente
 - Esta característica es una de las claves del éxito de la Web

Sistemas Distribuidos 15 Fernando Pérez Costoya

h proxy

Interfaz de Servicio
(args.)

Proxy

Interfaz de Servicio

Petición

Respuesta

Servidor

Fernando Pérez Costoya

Wikipedia: Proxy server

Sistemas Distribuidos 17 Fernando Pérez Costoya

Cliente/servidor jerárquico

- Servidor actúa como cliente de otro servicio
 - Igual que biblioteca usa servicio de otra biblioteca
- División vertical
 - Funcionalidad dividida en varios niveles (*multi-tier*)
 - P. ej. Aplicación típica con 3 capas:
 - Presentación
 - Aplicación: lógica de negocio
 - Acceso a datos
 - Cada nivel puede implementarse como un servidor
- División horizontal
 - Múltiples servidores idénticos cooperan en servicio
 - Traducir un nombre de fichero en SFD
 - Traducir de nombre de máquina a IP usando DNS

Sistemas Distribuidos 18 Fernando Pérez Costoya

tema

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

2-Arquit

tema

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

-servidor

a y ancho de banda
ilidad
idor (escalado vertical: *scale-up*)
abilidad ni tolerancia a fallos
eracción M-N)

tolerancia a fallos
carga
ados (p.e. DNS, Google FS)
ntener coherencia entre las réplicas
simultánea en todas las réplicas
n primario y propagar a secundarios
secundarios
a primario

Fernando Pérez Costoya

Scale-up vs Scale-out

Sistemas Distribuidos 20

Fernando Pérez Costoya

Código móvil

- Viaja el código en vez de los datos y/o resultados
- Requiere:
 - Arquitecturas homogéneas o
 - Interpretación de código o
 - Máquinas virtuales
- Modelos alternativos
 - Código por demanda (COD)
 - Servidor envía código a cliente
 - P.e. applets
 - Evaluación remota (REV)
 - Cliente dispone de código pero ejecución debe usar recursos servidor
 - P.ej. *Cyber-Foraging*
 - Agentes móviles
 - Componente autónomo y activo que viaja por SD

Sistemas Distribuidos 21

Fernando Pérez Costoya

demanda

Interfaz de Servicio

Servidor

Fernando Pérez Costoya

Evaluación remota

Sistemas Distribuidos 23

Fernando Pérez Costoya

Aspectos de diseño de cliente/servidor

Se van a considerar 5 aspectos específicos:

- Localización del servidor
- Esquemas de servicio a múltiples clientes
- Gestión de conexiones
- Servicio con estado o sin estado
- Comportamiento del servicio ante fallos

Sistemas Distribuidos 24

Fernando Pérez Costoya

tema

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

2-Arquit



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

el servidor

ión *DM* y usando puerto *PS*

inding

rmación → problema huevo-gallina

cias ID servicio → (*DM*, *PS*)

puerto del *Binder*

de servicio:

er jerárquico (como *pathname*)

(*DM1*, *PS1*) | (*DM2*, *PS2*)

es del servicio

ecanismo más general

estiona IDs de todos los recursos

Fernando Pérez Costoya

Alternativas en la ID del servicio

1. Uso directo de dirección *DM* y puerto *PS*
 - No proporciona transparencia
2. Nombre servicio + dir servidor (Java RMI *Registry*, *Sun RPC*)
 - Servidor (*binder*) en cada nodo: nombre de servicio → puerto
 - Impide migración del servidor
3. Nombre de servicio con ámbito global (DCE, CORBA, Mach)
 - Servidor con ubicación conocida en el sistema
 - Dos opciones:
 - a) Sólo *binder* global: nombre de servicio → [*DM*+*PS*]
 - b) Opción: *binder* global (BG) + *binder* local (BL) en puerto conocido
 - BG: ID → [*DM*] ; BL: ID → [*PS*]

- Uso de caché en clientes para evitar repetir traducción
- Mecanismo para detectar que traducción en caché ya no es válida

Sistemas Distribuidos

Fernando Pérez Costoya

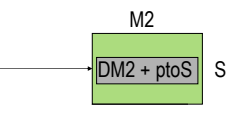
Símil basado en “hechos reales”

- Suponiendo n° tfno persona = (n° tfno empresa + extensión)
- Quiero contactar con persona *P* (TP = TE + EP)
- Alternativas:
 1. Conozco TP: lo uso
 2. Conozco TE y extensión de información de la empresa
 - Llamo a servicio información de la empresa y obtengo TP
 3. (a) Conozco teléfono general de información de personas
 - Llamo a servicio información general y obtengo TP
 3. (b) Conozco tño info. empresas y extensión info. de toda empresa
 - Llamo a servicio información general de empresas y obtengo TE
 - Llamo a servicio información de la empresa y obtengo TP

Sistemas Distribuidos

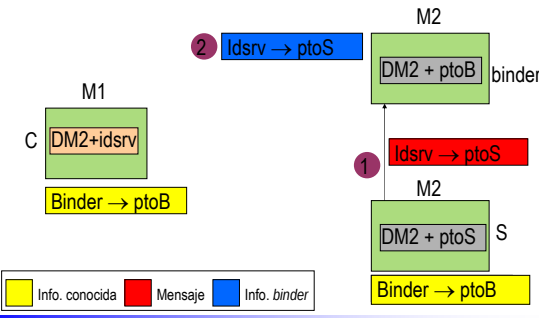
Fernando Pérez Costoya

[DM+pto]



Fernando Pérez Costoya

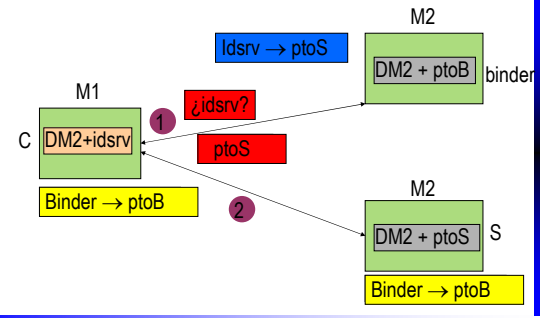
(2) ID servicio = [DM+idsrv]: Alta



Sistemas Distribuidos

Fernando Pérez Costoya

(2) ID servicio = [DM+idsrv]: Consulta



Sistemas Distribuidos

Fernando Pérez Costoya

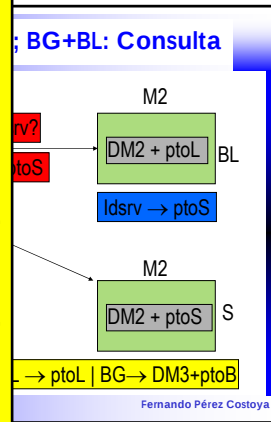
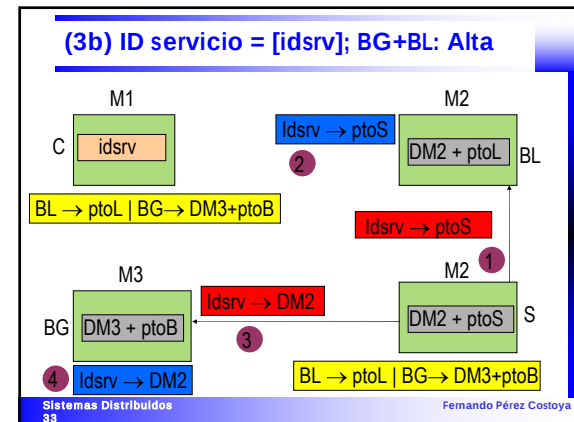
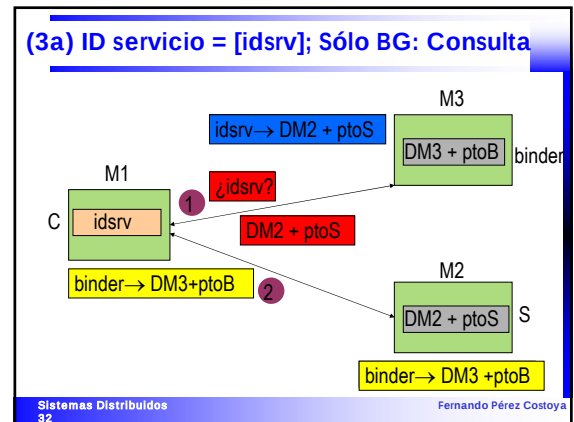
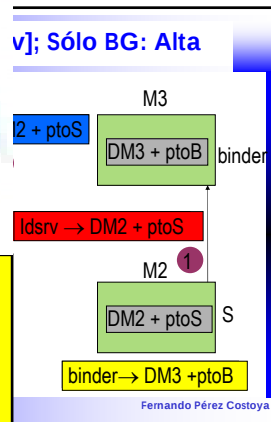
tema

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.



Recapitulación del Binding

- Caso con BG y BL + versiones:
 - Servidor:
 - Elige puerto local
 - Informa a *binder* local del alta:
 - (id. servicio + versión) = puerto
 - Informa a *binder* global del alta:
 - (id. servicio + versión) = dir máquina
 - Al terminar, notifica la baja a ambos *binder* :
 - Ambos eliminan (id. servicio + versión)
 - Cliente:
 - Consulta a *binder* global
 - (id. servicio + versión) -> dir. máquina
 - Consulta a *binder* en dir. máquina
 - (id. servicio + versión) -> puerto

Sistemas Distribuidos 35 Fernando Pérez Costoya

Servicio a múltiples clientes

- Servidor concurrente (p.e. servidor web Apache)
 - Un flujo de ejecución atiende sólo una petición en cada momento
 - Se bloquea esperando datos de ese cliente y envía respuesta
- Servidor basado en eventos (p.e. servidor web Nginx)
 - Un flujo de ejecución atiende múltiples peticiones simultáneamente
 - Espera (y trata) evento asociado a cualquiera de las peticiones
 - Evento: actividad asociada con 1 petición (p.e. llegada datos de cliente)
 - Implementación en UNIX de espera simultánea; alternativas:
 - *select/poll/epoll*; uso de señales de t.real; operaciones asíncronas (*aio*)
 - Para aprovechar paralelismo HW: un flujo de ejecución/procesador
- Servidor concurrente vs. basado en eventos:
 - Peor escalabilidad (*The C10K problem*: <http://kegel.com/c10k.html>)
 - Sobrecarga creación/destrucción/planificación de procesos/*threads*, más cambios de contexto, más gasto de memoria (p.e. pilas de *threads*),...
 - Programación menos "oscura"

Sistemas Distribuidos 36 Fernando Pérez Costoya

e: alternativas

- os y comparten más recursos zación
- n llegan peticiones xión
- rancados: más peticiones

y máximo M

→ se crea
 m → se destruye

Sistemas Distribuidos
36

Fernando Pérez Costoya

Gestión de conexiones

- En caso de que se use un esquema con conexión
- 1 conexión por cada petición
 - 1 operación cliente-servidor
 - conexión, envío de petición, recepción de respuesta, cierre de conexión
 - Más sencillo pero mayor sobrecarga (¡9 mensajes con TCP!)
 - Protocolos de transporte orientados a C/S (T/TCP, TCP Fast Open)
- Conexiones persistentes: N peticiones cliente misma conexión
 - Más complejo pero menor sobrecarga
 - Esquema usado en HTTP/1.1 (además, *pipeline* de peticiones)
 - Dado que servidor admite nº limitado de conexiones
 - Dificulta reparto de servicio entre clientes
 - Implica que servidor mantiene un estado
 - Dificulta reparto de carga en esquema con escalado horizontal
 - Facilita *server push*

Sistemas Distribuidos
38

Fernando Pérez Costoya

Evolución de HTTP

- Cliente necesita pedir varios objetos al mismo servidor
- HTTP/1.0
 - Connect | GET | Resp | Close | Connect | GET | Resp | Close | ...
 - Sobrecarga conexiones + latencia de conexiones y peticiones
- HTTP/1.0 + conexiones persistentes
 - Connect | GET | Resp | GET | Resp | ... | Close
 - Latencia de peticiones
- HTTP/1.1 (conexiones persistentes + *pipeline* de peticiones)
 - Connect | GET | GET | ... | Resp | Resp | ... | Close
 - No latencia acumulada
 - Servicio paralelo de peticiones
 - aunque respuestas deben llegar en orden

Sistemas Distribuidos
39

Fernando Pérez Costoya

simultáneas

conexiones simultáneas

entes
sp | ... | Close

sp | ... | Close

tes + *pipeline* de peticiones)
Resp | ... | Close

Resp | ... | Close

Sistemas Distribuidos
40

Fernando Pérez Costoya

Client Pull vs Server Push

- C/S: modo *pull* → cliente "extrae" datos del servidor
- Escenario: servidor dispone de información actualizada
 - P.e. retransmisión web en modo texto de acontecimiento deportivo
 - P.e. servicio de chat basado en servidor centralizado
- ¿Cómo recibe cliente actualizaciones? Alternativas:
 - Cliente *polling* periódico al servidor (web: HTTP *refresh*; Ajax *polling*)
 - Servidor responde inmediatamente, con nuevos datos si los hay
 - Long Polling*: Servidor no responde hasta que tenga datos
 - Server Push*: servidor "empuja" datos hacia el cliente
 - Cliente mantiene conexión persistente y servidor envía actualizaciones
 - Web: HTTP *Server Push*, *Server-Sent Events*, *Web Sockets*
 - Usar editor/subscriptor en vez de cliente/servidor

Sistemas Distribuidos
41

Fernando Pérez Costoya

Servicio con/sin estado

- ¿Servidor mantiene información de clientes?
- Ventajas de servicio con estado (*aka* con sesión remota):
 - Mensajes de petición más cortos
 - Mejor rendimiento (se mantiene información en memoria)
 - Favorece optimización de servicio: estrategias predictivas
- Ventajas de servicio sin estado:
 - Más tolerantes a fallos (ante rearranque del servidor)
 - Peticiones autocontenidas.
 - Reduce nº de mensajes: no comienzos/finales de sesión.
 - Más económicos para servidor (no consume memoria)
 - Mejor reparto carga y fiabilidad en esquema con escalado horizontal
- Servicio sin estado base de la propuesta REST
- Estado sobre servicios sin estado
 - Cliente almacena estado y lo envía al servidor (p.e. HTTP+cookies)

Sistemas Distribuidos
42

Fernando Pérez Costoya



tema

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

2-Arquit

tema

www.cartagenaa99.com no se hace responsable de la información contenida en el presente documento en virtud al

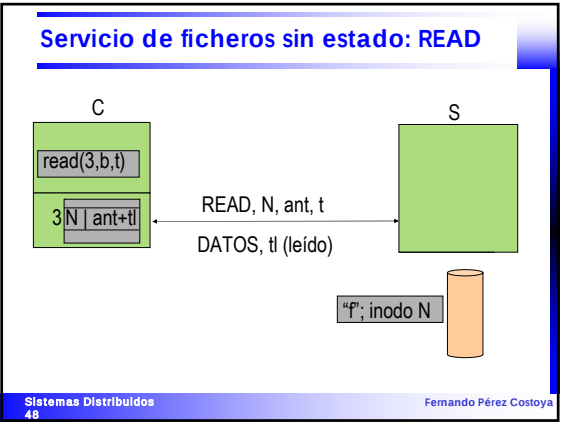
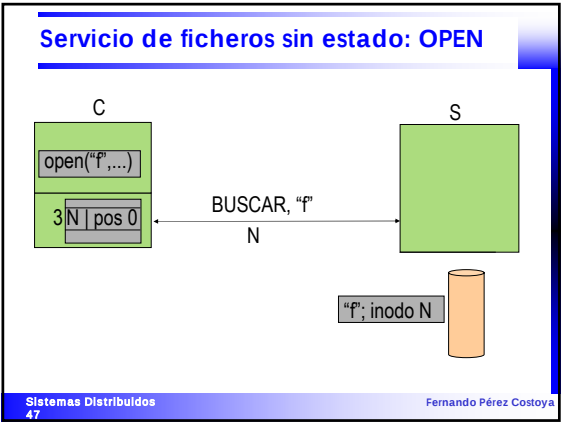
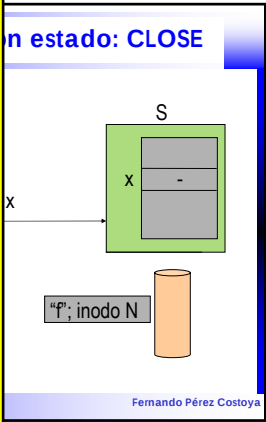
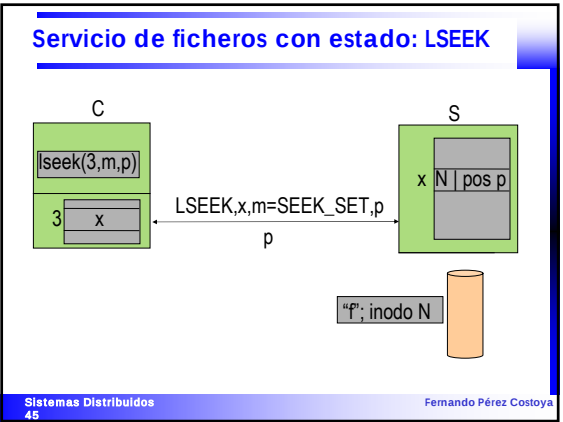
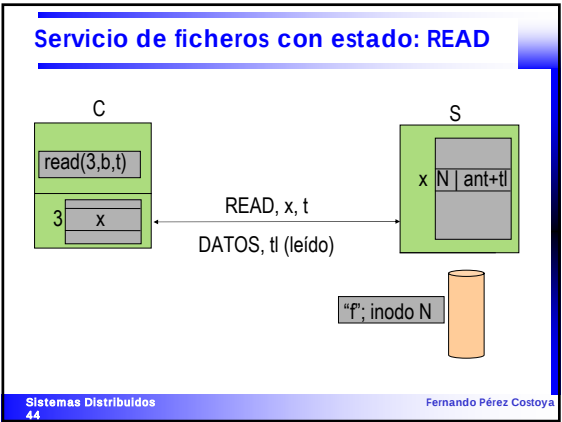
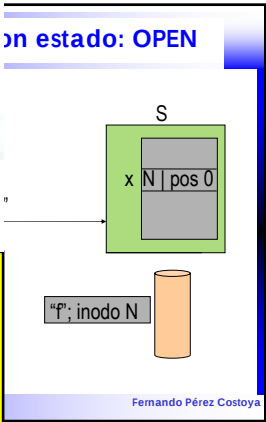
Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002.

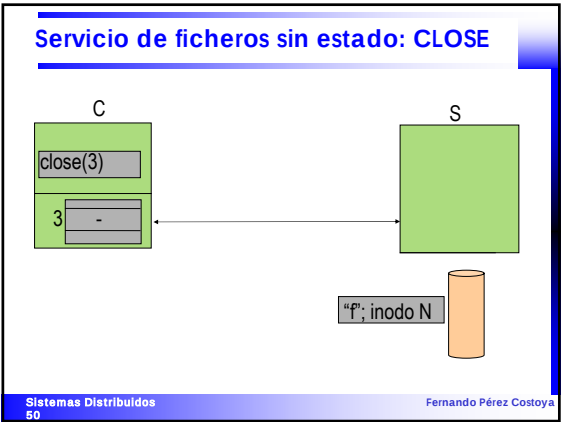
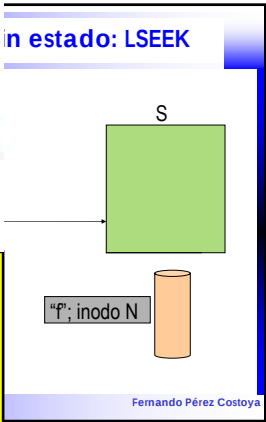
Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

2-Arquit

Cartagena99

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
LLAMA O ENVÍA WHATSAPP: 689 45 44 70
CALL OR WHATSAPP:689 45 44 70





Leases

- Mecanismo para mejorar tolerancia a fallos en SD
 - Detección y tratamiento de caídas de nodos
- Aplicación típica (genérica) de *leases*:
 - Proceso A gestiona algún tipo de recurso vinculado con proceso B
 - Proceso B no tiene por qué contactar de nuevo con A
 - Si B cae, A no lo detecta y el recurso queda "abandonado"
- Modo de operación
 - A otorga a B un *lease* que dura un plazo de tiempo
 - B debe enviar a A mensaje de renovación *lease* antes de fin de plazo
 - Si B cae y no renueva *lease*, se considera recurso "abandonado"
 - Si A cae, en reinicio obtiene renovaciones
 - Con suficiente información, puede "reconstruir" los recursos
- No confundir con un simple temporizador
 - Proceso envía petición a otro y arranca temporizador
 - Si se cumple antes de ACK, vuelve a enviar petición ($\neq$ *lease*)

Sistemas Distribuidos 51 Fernando Pérez Costoya

Leases

opción en Ed/Su, caché de SFD, etc.

ente con estado

errojos distribuido

entras en posesión de cerrojo

deben renovar su *lease*

primero peticiones de renovación

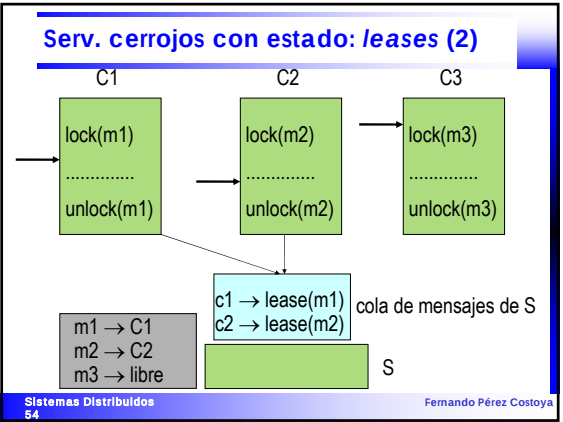
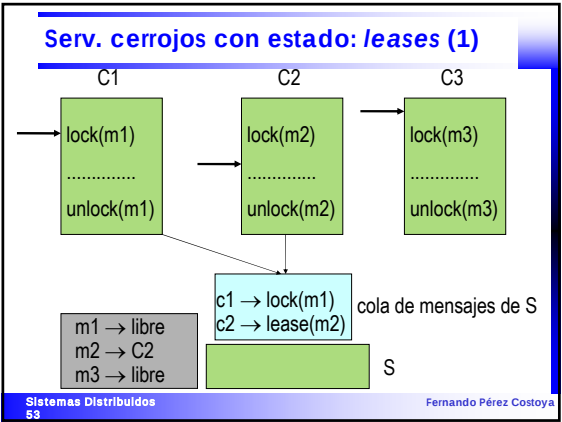
tiempo de renovación

tado después de re-arranque de S

ción

os de ese cliente

Fernando Pérez Costoya



tema

Cartagena99

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

tema

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al

Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002.

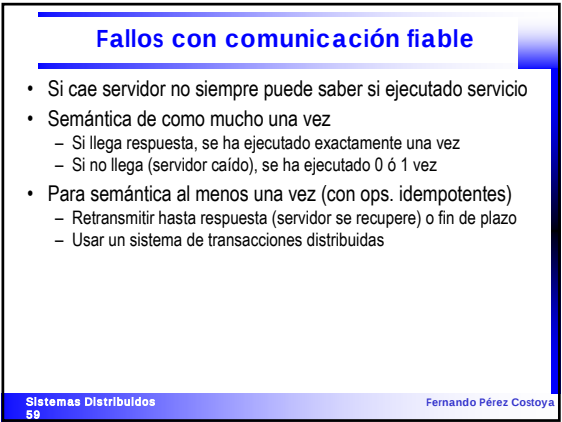
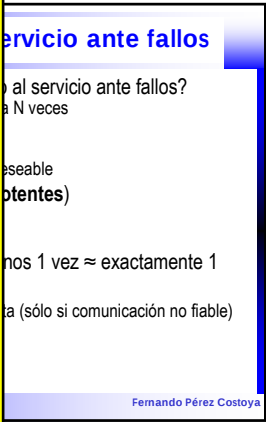
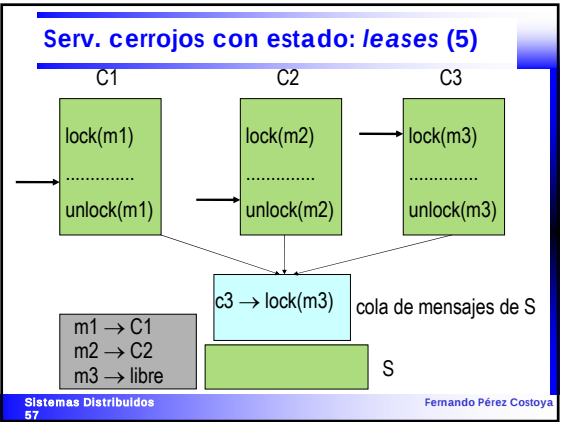
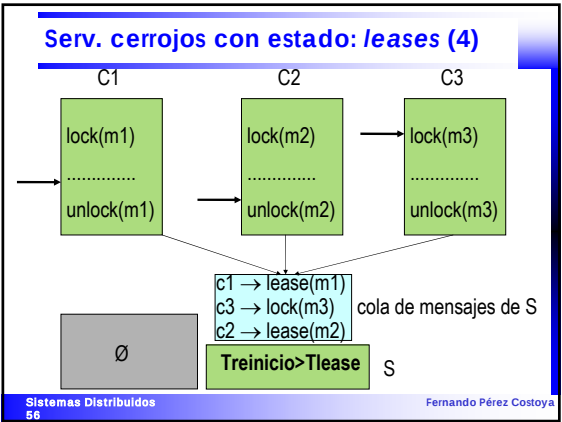
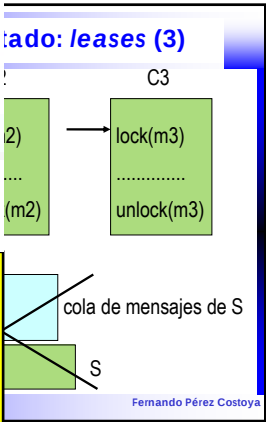
Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

2-Arquit

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



cliente

le computación huérfana
vidor

diadas un n° de época
smite (++n° de época) a servidores
on n° de época menor

ura el servicio
orta el servicio

o del servidor (p.e. cerrojos)
abortar

Fernando Pérez Costoya

Modelo editor/subscriptor

- Sistema de eventos distribuidos (Jini, s. eventos/notifi. CORBA)
- Suscriptor *S* (*subscriber*): interés por ciertos eventos (**filtro**)
- Editor *E* (*publisher*) genera un evento
 - Se envía a subscriptores interesados en el mismo
- Paradigma asíncrono y desacoplado en espacio
 - Editores y subscriptores no se conocen entre sí (≠ cliente/servidor)
- Normalmente, *push*: suscriptor recibe evento
 - Alternativa, *pull*: suscriptor pregunta si hay eventos de interés
 - Híbrido: suscriptor recibe notificación pero tiene que solicitar evento
 - Pull* e híbrido requieren que se almacenen eventos (+ complejo)
 - Posibilitan mecanismo desacoplado en el tiempo
- Facilita uso en sistemas heterogéneos
- Diversos aspectos relacionados con la calidad de servicio
 - orden de entrega, fiabilidad, persistencia, prioridad, transacciones,...

Sistemas Distribuidos 62 Fernando Pérez Costoya

Operaciones modelo editor/subscriptor

- Estructura típica del evento: [*atrib1=val1; atrib2=val2; ...*]
 - Un atributo puede ser el **tema** del evento
- suscribe(tipo)* [*S*→]: interés por cierto tipo de eventos
 - Posible uso de *leases* en suscripción
- baja(tipo)* [*S*→]: cese del interés
- publica(evento)* [*E*→]: generación de evento
- notifica(evento)* [*→S*]: envío de evento (*push*)
- Extensión de modelo: creación dinámica de tipos de eventos
 - anuncia(tipo)* [*E*→]: se crea un nuevo tipo de evento
 - baja_tipo(tipo)* [*E*→]: se elimina tipo de evento
 - notifica_tipo(tipo)* [*→S*]: aviso de nuevo tipo de eventos
- Ejemplos de aplicación
 - Mercado bursátil, subastas, *chat*, aplicación domótica, etc.

Sistemas Distribuidos 63 Fernando Pérez Costoya

subscriptor

nuevo tipo de evento (E→S)

Fernando Pérez Costoya

Filtro de eventos por tema

- S* se suscribe a tema y recibe notificaciones sobre el mismo
- Temas disponibles:
 - Carácter estático: implícitamente conocidos
 - Carácter dinámico: uso de operación de anuncio
 - Ej. Creación de un nuevo valor en el mercado
- Organización del espacio de temas:
 - Plano
 - Jerárquico: (Ej. *bolsas_europeas/españa/madrid*)
 - Uso de comodines en la suscripción
- Filtrados adicionales deben hacerse en aplicación
 - En ocasiones filtro por tema resulta un grano demasiado grueso
 - Ej. Interesado en todos los valores de un ámbito de negocio

Sistemas Distribuidos 65 Fernando Pérez Costoya

Filtro de eventos por contenido

- Debe cumplirse condición sobre atributos del evento
 - Extensión del esquema previo: tema es un atributo del evento
- Uso de lenguaje para expresión de la condición (≈ SQL)
- Filtrado de grano más fino y dinámico
 - Ej. Interés en valores de mercado en alza
- Menor consumo de ancho de banda
 - Llegan menos datos a nodos subscriptor
- Simplifica *app.* subscriptora pero complica esquema Ed/Su
 - Puede involucrar varios tipos de eventos de forma compleja
 - Ejemplo (Tanenbaum):
 - "*Avisame cuando la habitación H420 esté desocupada más de 10 segundos estando la puerta abierta*"

Sistemas Distribuidos 66 Fernando Pérez Costoya

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

...s por tipo

...uidos basados en objetos
...terminada clase
...enece a una jerarquía de herencia
...OS

...de temas
...ción de filtrado
...do el sistema de tipado
...un esquema de objetos homogéneo

Fernando Pérez Costoya

Cliente/servidor vs. Editor/suscriptor

¿En cuál es más fácil añadir nuevo consumidor de datos?

Sistemas Distribuidos 68 Fernando Pérez Costoya

Implementaciones editor/suscriptor

- Comunicación directa
 - No proporciona desacoplamiento espacial
- Uso de intermediario (*broker*)
 - Desacoplamiento espacial pero cuello de botella y único punto de fallo
- Uso de red de intermediarios
 - Distribución de eventos y aplicación de filtros “inteligente”
- Posible uso de comunicación de grupo en cualquiera de ellas
 - Ej. Comunicación directa + comunicación de grupo
 - Ed/Su basado en temas: tema = dirección de grupo

Sistemas Distribuidos 69 Fernando Pérez Costoya

...sin intermediario

...susc.
...acial

Fernando Pérez Costoya

Implementación ed/su con intermediario

Proceso intermediario
↑ Desacoplamiento espacial
↓ Cuello botella y punto fallo

Sistemas Distribuidos 71 Fernando Pérez Costoya

Implementación ed/su con red interm.

Red de intermediarios
↑ Desacoplamiento espacial
↑ Escalabilidad y fiabilidad

Sistemas Distribuidos 72 Fernando Pérez Costoya

tema

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al
Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002.
Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

2-Arquit

Peer (P2P)

ol y funcionalidad (*servents*)
os críticos de fallo
i las máquinas
por:
del SD; variabilidad en conectividad
a autoridad global centralizada
y):
e red
amiento y localización de recursos
P
mayores problemas de seguridad

Fernando Pérez Costoya

Esquema Peer-to-Peer (P2P)

Sistemas Distribuidos
74

Fernando Pérez Costoya

Aplicaciones de P2P

- **Compartir contenidos (P2P content-sharing)**
 - Diversos aspectos no tratados: *copyright*, *anónimo*, *free-riding*, ...
- Distribución de contenidos (1 editor; múltiples descargadores)
- Almacenamiento distribuido
 - Sistemas de ficheros P2P
 - Cachés P2P
- Comunicación
 - Mensajería instantánea
 - VoIP
 - Videoconferencia
- *Streaming* (P2PTV)
- Computación distribuida ...

Sistemas Distribuidos
75

Fernando Pérez Costoya

content-sharing

P2P

¿Cómo se encuentra un recurso?
- Una convención (p.ej. *artista + canción*)
- Detectar recursos repetidos
- Metadatos (p.e. estilo musical))

... uso de comodines)

Fernando Pérez Costoya

Tipos de sistemas P2P

- **Desestructurados:**
 - Topología de conexión lógica arbitraria
 - Ubicación de recursos impredecible e independiente de la topología
 - Cada nodo posee un conjunto de recursos
 - Corresponden a sistemas +volátiles con nodos más autónomos
 - Tipo de sistemas P2P desestructurados:
 - Centralizados, Descentralizados y Parcialmente centralizados
- **Estructurados:**
 - Topología de conexión prefijada (p.e. anillo en protocolo *Chord*)
 - Ubicación de recursos predecible y dependiente de la topología
 - *lookup*(ID recurso) → máquina que posee recurso
 - Corresponden a sistemas -volátiles con nodos más cooperativos
 - Posesión de recursos cambia según sistema evoluciona

Sistemas Distribuidos
77

Fernando Pérez Costoya

Sistemas P2P centralizados

- P2P + Cliente/servidor
 - Uso de servidor centralizado que conoce nodos y recursos
 - Simple y fácil de gestionar
 - Cuello de botella y punto único de fallo (puede estar replicado)
- Uso para compartir contenidos (p.e. Napster: 1999-2001)
 - **Alta:** cliente C contacta con servidor S
 - En Napster: (*artista + canción*) que se busca en nombre de fichero
 - **Pubicar:** en alta C informa a S de su lista de ficheros publicados
 - **Buscar:** C envía petición a S y recibe lista de clientes con ese fichero
 - En Napster: (*artista + canción*) que se busca en nombre de fichero
 - **Descargar:** C elige cliente con mejor t. respuesta y pide descarga
 - C informa a S de que ya posee copia de fichero
 - Se lo podrán pedir otros clientes a partir de ahora
 - No hay descarga en paralelo de múltiples clientes

Sistemas Distribuidos
78

Fernando Pérez Costoya

tema

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Si la información contenida en el presente documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

2-Arquit

tema

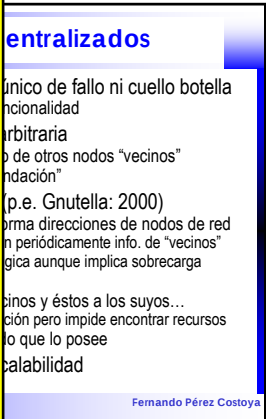
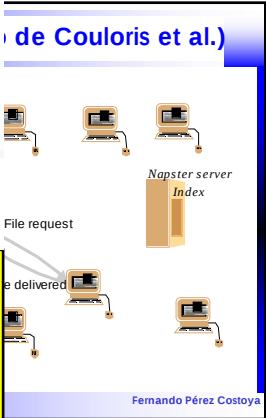
www.cartagenaa99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

2-Arquit



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

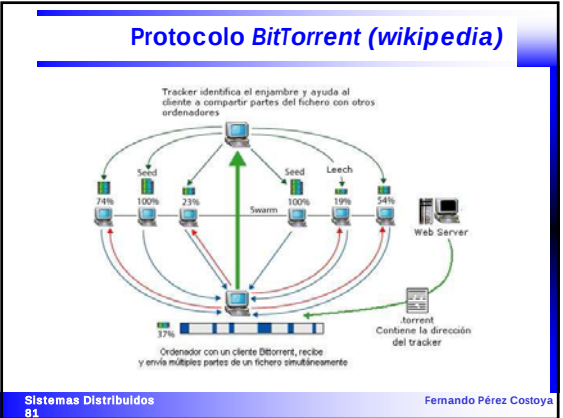
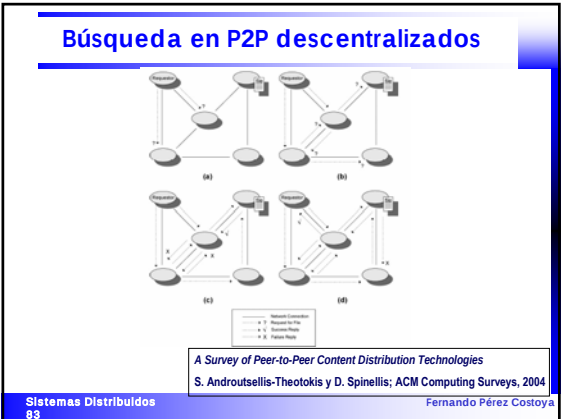
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70



Protocolo BitTorrent (2001)

- Distribución de contenidos mediante P2P centralizado
 - Se centra en descarga; búsqueda externa al protocolo (p.e. Google)
 - Descarga paralela de trozos del fichero de clientes que los poseen
 - Servidor (*tracker*) conoce qué clientes tienen trozos del fichero
 - Fichero *.torrent*: información del fichero y del *tracker*
- Operaciones:
 - **Publicar**: Cliente inicial (*initial seeder*) crea *.torrent*
 - Puede especificar un *tracker* propio o público
 - **Buscar**: Cliente C encuentra de alguna forma *.torrent* del recurso
 - **Alta**: C contacta con *tracker* → lista de algunos clientes con trozos
 - C contacta con esos clientes que le dicen qué trozos tienen
 - **Descargar**: C descarga en paralelo trozos de clientes
 - Primero los "más raros"
 - Cuando completa descarga de un trozo, lo notifica a otros clientes
 - C envía trozos a clientes que se los solicitan

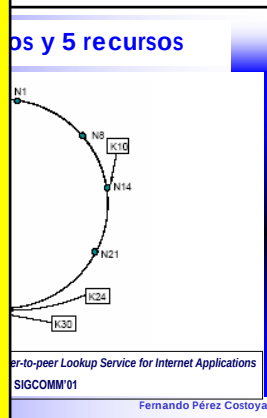
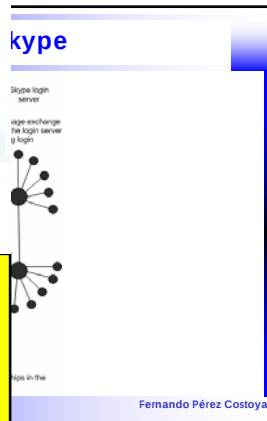
Sistemas Distribuidos 80 Fernando Pérez Costoya



Sistemas P2P parcialmente centralizados

- Sistema jerárquico: nodos ordinarios ON y supernodos SN
 - Nodo ordinario debe estar asociado a un supernodo
 - Supernodo tiene información de grupo de nodos asociados
 - Supernodos dinámicamente elegidos por potencia, conectividad,...
- Mezcla de descentralizado y centralizado
 - Red descentralizada de supernodos (como Gnutella)
 - Cada SN servidor de ONs asociados al mismo (como Napster)
- Uso para compartir contenidos (p.e. Kazaa/FastTrack: 2001)
 - **Alta**: nodo N obtiene dirección de algún supernodo SN
 - En algún momento N puede llegar a ser SN
 - **Publicar**: en alta N informa a SN de su lista de ficheros publicados
 - **Buscar**: N petición a su SN y éste la propaga sólo por SNs (con TTL)
 - N recibe lista de nodos con ese fichero
 - **Descargar**: N pide recurso a nodo que lo posee
 - Extensión: descarga simultánea desde múltiples nodos

Sistemas Distribuidos 84 Fernando Pérez Costoya



Sistemas P2P estructurados

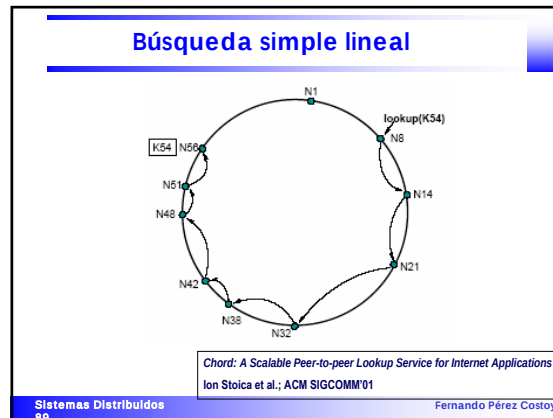
- Sistema descentralizado con búsquedas $O(\log n)$ en vez $O(n)$
 - Protocolos *Chord* (Stoica et al. MIT 2001), *CAN*, *Tapestry* y *Pastry*
 - Red superpuesta de topología prefijada
- ¿En qué nodo se almacena un recurso? → función *hash*
 - "Tabla *hash* distribuida" (DHT: *Distributed Hash Table*)
 - lookup*(ID recurso) → ID nodo que posee recurso
 - Esquema estable y escalable ante n° enorme y dinámico de nodos
- Uso para compartir contenidos
 - Alta:** nodo *N* obtiene dirección de algún nodo y se incorpora a red
 - Publicar:** *N* aplica la función de *lookup* para ubicarlo
 - Buscar:** *N* aplica la función de *lookup* para encontrarlo
 - Descargar:** No definida por el protocolo
 - Nodo "poseedor" puede tener el recurso o una referencia al mismo

Sistemas Distribuidos 86 Fernando Pérez Costoya

Protocolo *Chord*

- Hashing* "consistente" asigna ID a clave recurso y a IP de nodo
 - ID con m bits tal que n° recursos (K) + n° nodos (N) $\ll 2^m$
 - IDs organizados en anillo módulo 2^m
 - Proporciona distribución uniforme
- Asignación de recurso K a nodo N
 - 1^{er} nodo $ID(N) \geq ID(K) \rightarrow$ sucesor(K) (NOTA: $\geq$ siguiendo módulo)
- Localización de recurso: N busca K ; algoritmo simple
 - Cada nodo sólo necesita almacenar quién es su sucesor directo
 - NOTA: nodo almacena también predecesor
 - Búsqueda lineal de sucesor a sucesor
 - Hasta encontrar par de nodos a, b tal que $ID \in (a, b]$
 - Ineficiente y no escalable $O(N)$

Sistemas Distribuidos 87 Fernando Pérez Costoya



Búsqueda basada en *fingers*

- Cada nodo con ID n incluye tabla *fingers* TF con m entradas:
 - Entrada i : primer nodo a una distancia $\geq 2^{i-1}$
 - sucesor*($n + 2^{i-1}$) tal que $1 \leq i \leq m$
 - Entrada 0: sucesor directo
- Búsqueda de K desde nodo n :
 - Si $ID \in (n, s: \text{sucesor directo de } n] \rightarrow K$ asignado a s
 - Sino: buscar en TF empezando por Entrada m :
 - Nodo más inmediatamente precedente
 - Pedirle que continúe la búsqueda
- Tiempo localización e info. requerida: $O(\log N)$

Sistemas Distribuidos 90 Fernando Pérez Costoya

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

tema

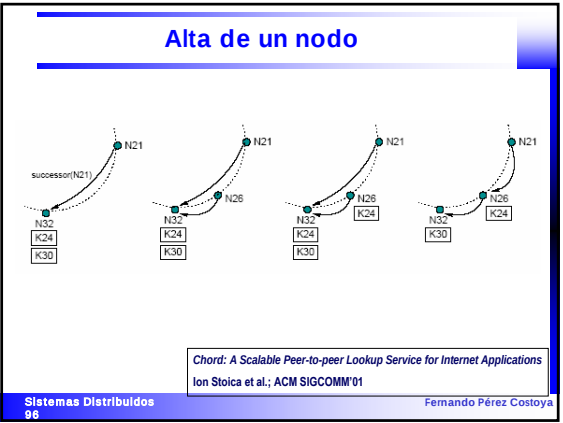
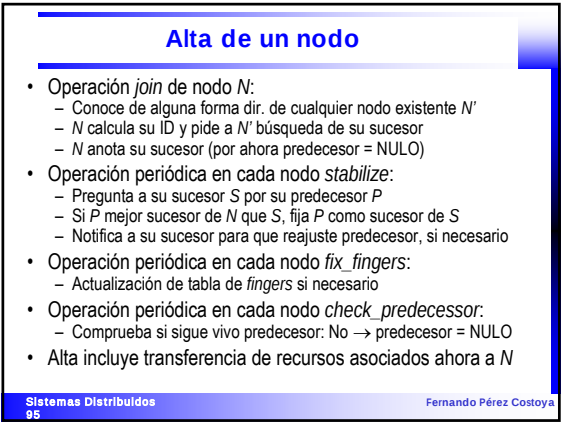
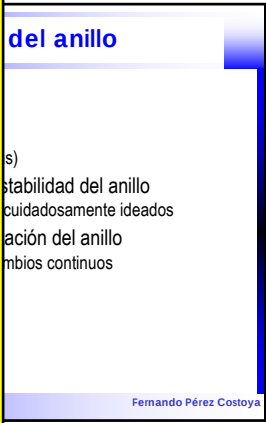
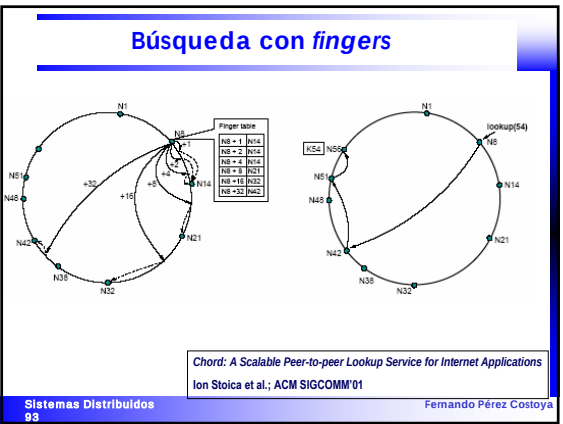
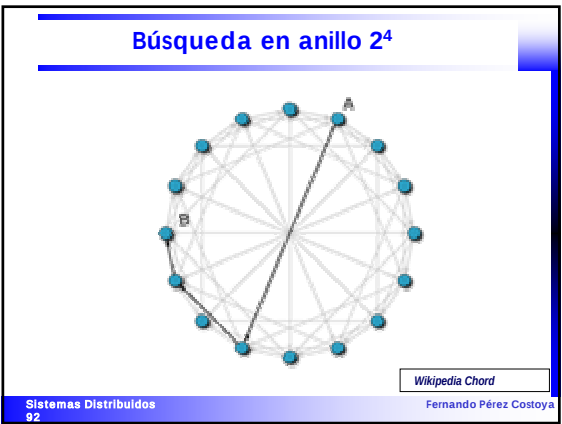
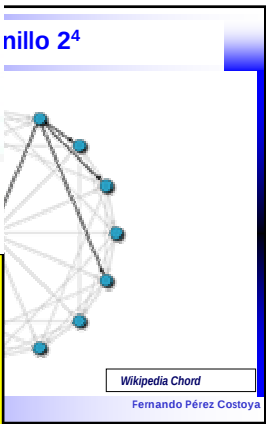
www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

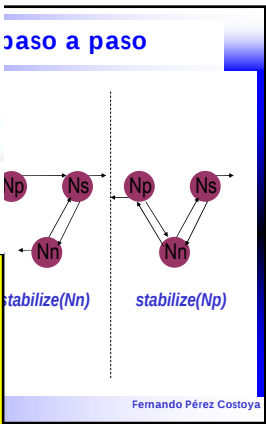
2-Arquit



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

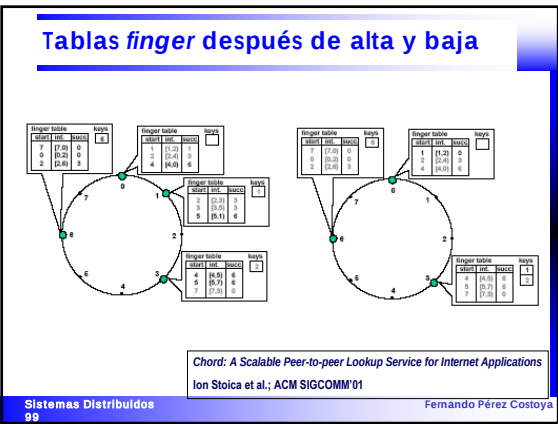




Baja de un nodo

- Baja voluntaria de nodo implica acciones complementarias
 - Devolver recursos a nuevo sucesor
 - Informar a predecesor y sucesor para que reajusten estado
- Caída de nodo (baja involuntaria) más problemática
 - Operaciones periódicas de estabilización van reajustando el anillo
 - Pero puede haber problemas en búsqueda hasta reajuste
 - Nodo sucesor caído hasta que se actualiza nuevo sucesor
 - Solución: Cada nodo guarda lista de sus m sucesores
 - ¿Qué pasa con los recursos del nodo caído?
 - Protocolo no especifica política de replicación de recursos
- Algoritmo estable ante altas y bajas simultáneas
 - Es capaz de trabajar con info. no totalmente actualizada

Sistemas Distribuidos 98 Fernando Pérez Costoya



tema

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

2-Arquit