01 - Caracterización de Sistemas Distribuidos

IIC2523 - Sistemas Distribuidos

Cristian Ruz – cruz@ing.puc.cl

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Semestre 2-2020

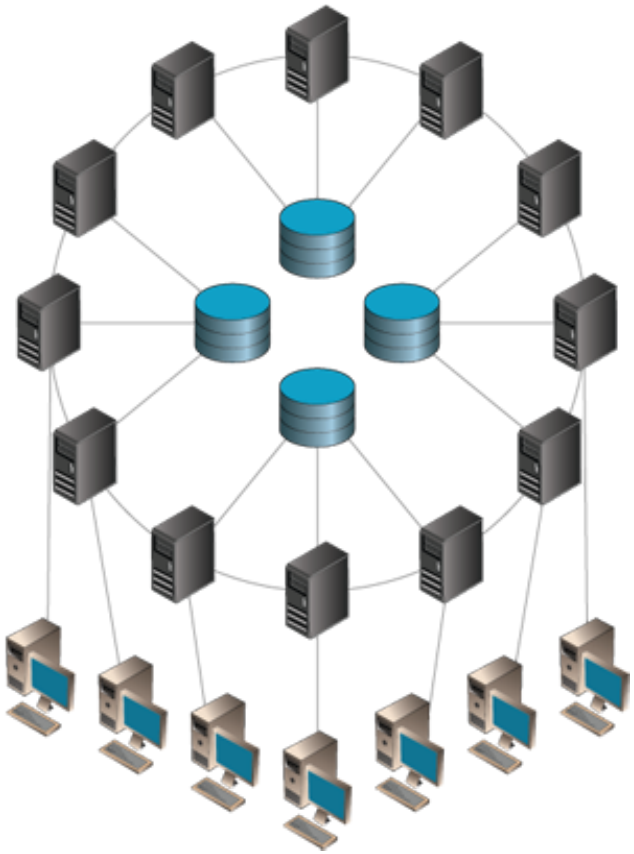
Contenidos

Contenidos

Actividad 1

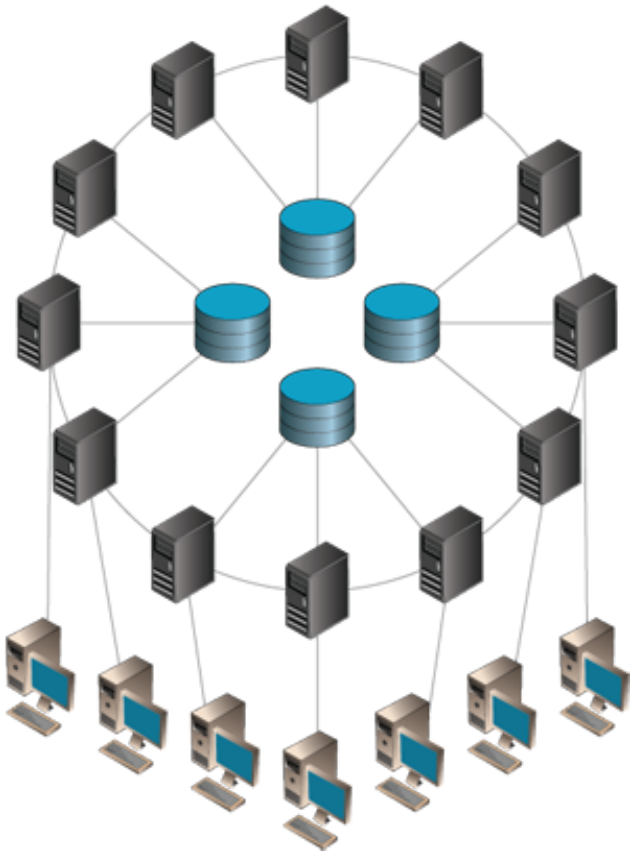
Actividad 1

1. Individualmente, elija 2 sistemas distribuidos que conozca o use



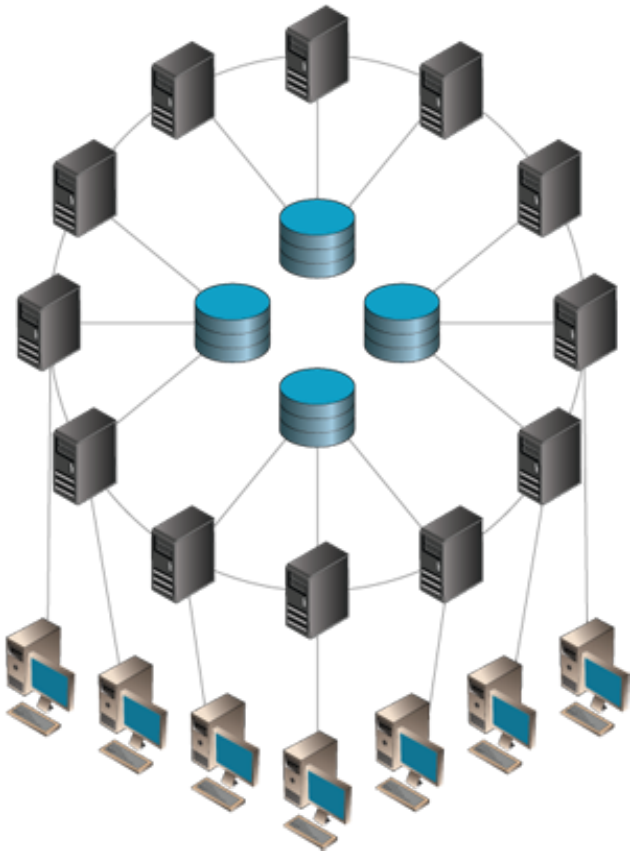
Actividad 1

1. Individualmente, elija 2 sistemas distribuidos que conozca o use
2. Individualmente, ¿Cuál es la mayor dificultad (técnica) que tendría si tuviera que implementar estos sistemas?



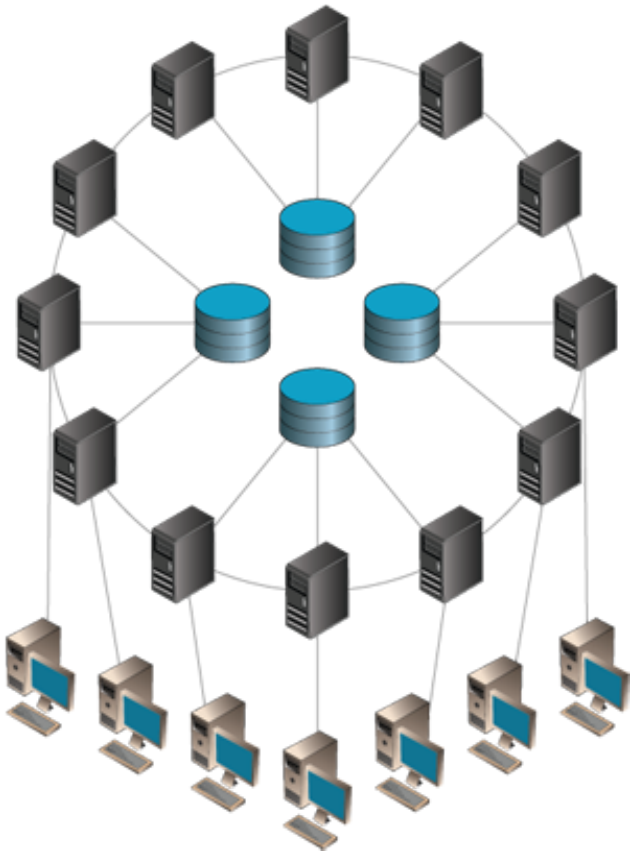
Actividad 1

1. Individualmente, elija 2 sistemas distribuidos que conozca o use
2. Individualmente, ¿Cuál es la mayor dificultad (técnica) que tendría si tuviera que implementar estos sistemas?
3. En grupo, escojan 2 y determinen la(s) mayor(es) dificultad(es) técnica(s) que tendrían que resolver si tuvieran que implementarlos.



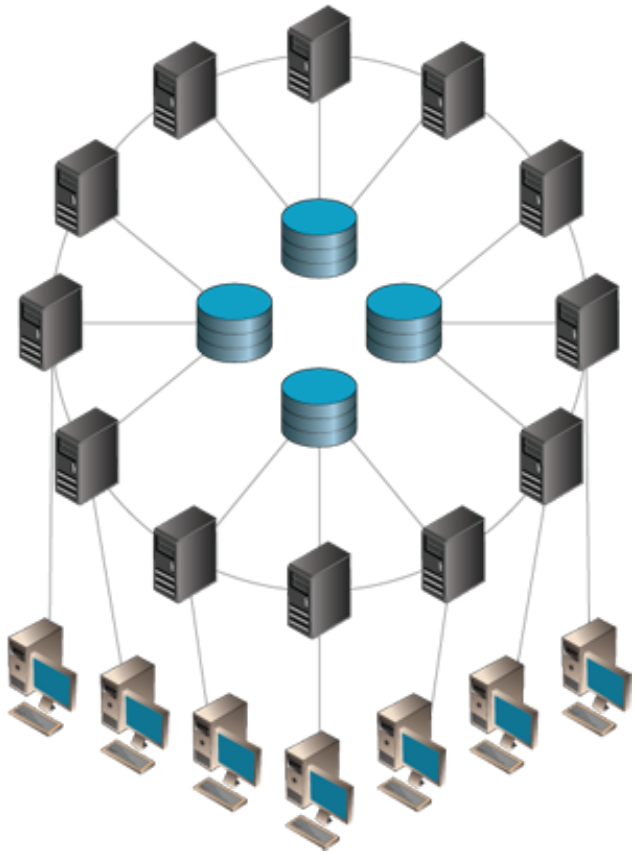
Actividad 1

1. Individualmente, elija 2 sistemas distribuidos que conozca o use
2. Individualmente, ¿Cuál es la mayor dificultad (técnica) que tendría si tuviera que implementar estos sistemas?
3. En grupo, escojan 2 y determinen la(s) mayor(es) dificultad(es) técnica(s) que tendrían que resolver si tuvieran que implementarlos.
4. En grupo, ¿qué características comparten los sistemas que escogieron?



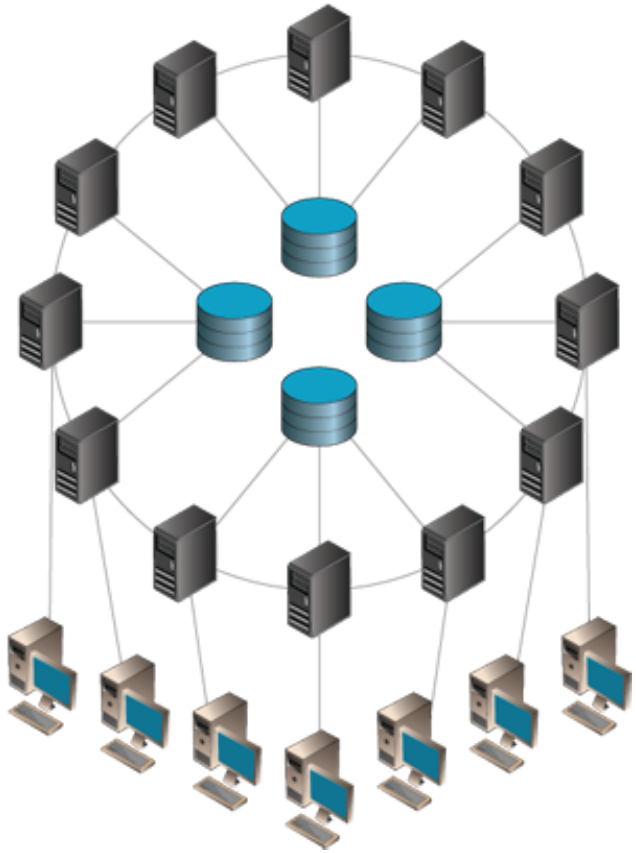
Actividad 1

1. Individualmente, elija 2 sistemas distribuidos que conozca o use
2. Individualmente, ¿Cuál es la mayor dificultad (técnica) que tendría si tuviera que implementar estos sistemas?
3. En grupo, escojan 2 y determinen la(s) mayor(es) dificultad(es) técnica(s) que tendrían que resolver si tuvieran que implementarlos.
4. En grupo, ¿qué características comparten los sistemas que escogieron?
5. Uno por grupo, comparte el resumen.



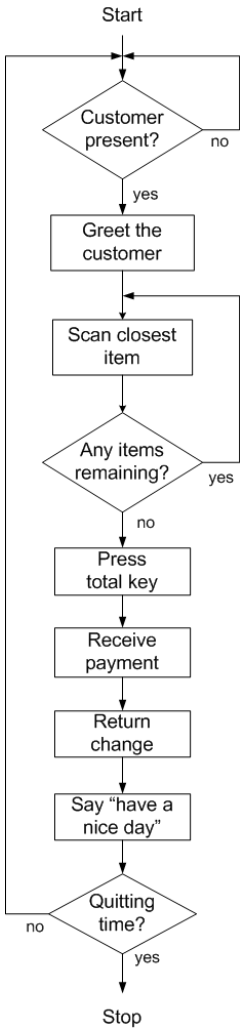
¿Dónde no hay sistemas distribuidos?

- Internet, WWW
- Streaming, YouTube, UStream,
- Juegos multiplayer
- Multicores, tarjetas gráficas
- Servicios remotos ..., Facebook, Twitter, Instagram
- Teléfonos móviles, vehículos,



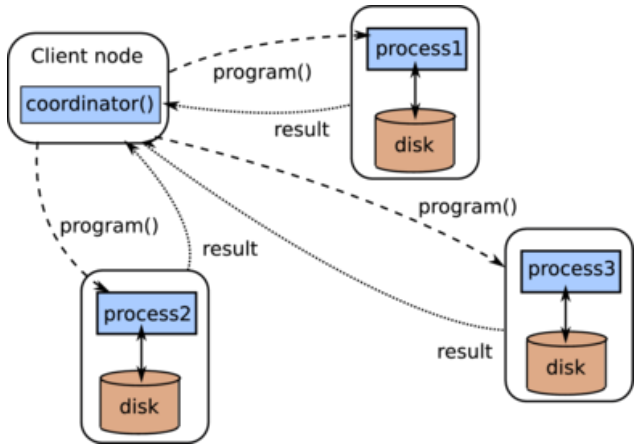
Desde lo secuencial ...

- Una CPU, un *thread*, un flujo de instrucciones
- También hay multitasking → múltiples *threads*



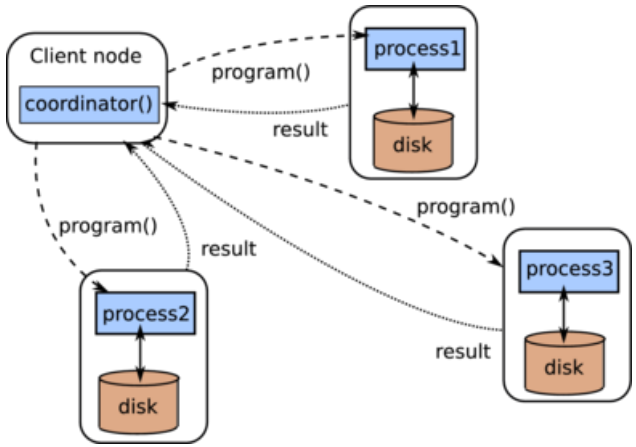
. . . hacia lo distribuido

- Muchas CPUs, muchos *threads*, muchos flujo de instrucciones



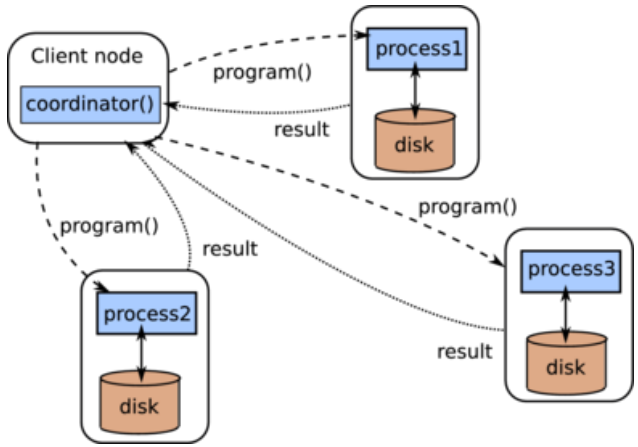
. . . hacia lo distribuido

- Muchas CPUs, muchos *threads*, muchos flujo de instrucciones
- Multitasking real
  - *Threads* ejecutan simultáneamente



. . . hacia lo distribuido

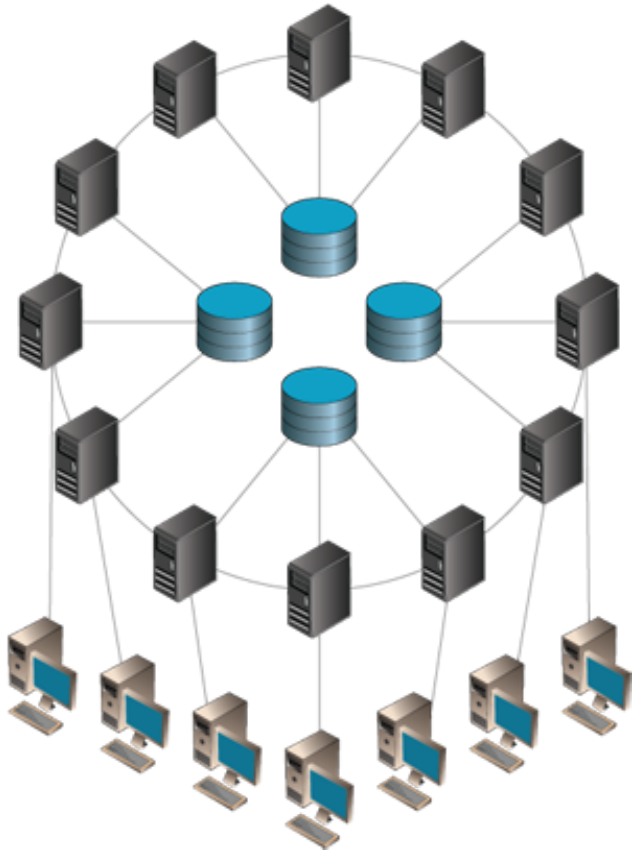
- Muchas CPUs, muchos *threads*, muchos flujo de instrucciones
- Multitasking real
  - *Threads* ejecutan simultáneamente
  - ¿Cómo se coordinan?
  - ¿Memoria compartida o paso de mensajes?



Actividad 2

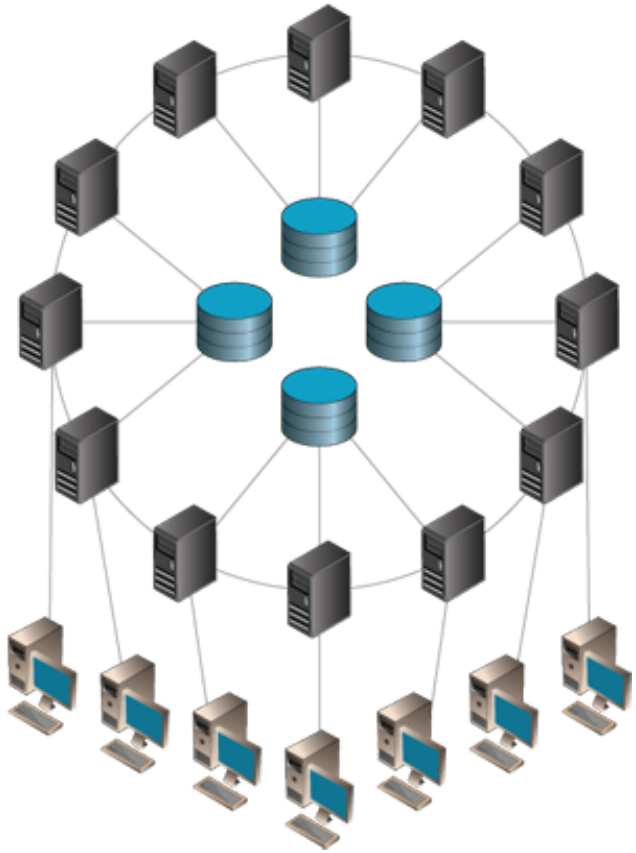
Actividad 2

1. Individualmente, elabore una definición de un sistema distribuido.



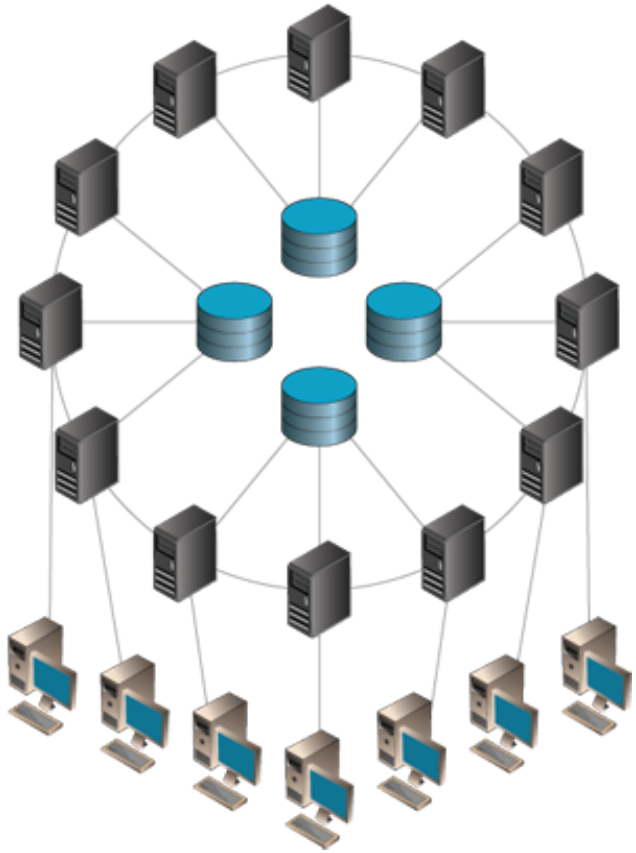
Actividad 2

1. Individualmente, elabore una definición de un sistema distribuido.
2. En grupo, compartir la definición y elaborar una grupal



Actividad 2

1. Individualmente, elabore una definición de un sistema distribuido.
2. En grupo, compartir la definición y elaborar una grupal
3. Uno por grupo, comparte la definición.



Book Your Concert!

John decide proveer un servicio de venta de tickets para conciertos, eventos deportivos, etc.
Tres simples pasos: Usted nos llama, hacemos su reserva y cuando desee retira su entrada.

Adaptación del ejemplo publicado en:
http://ksat.me/a-plain-english-introduction-to-cap-theorem/

Book Your Concert!

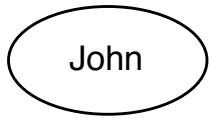
¡Implementémoslo!

Dos servicios:

- Reserva de puestos
- Entrega de entradas

Versión 1

- John recibe llamadas telefónicas
- John anota reservas en un cuaderno
- John entrega entradas



¡Sistema centralizado!, pero funciona

Book Your Concert!

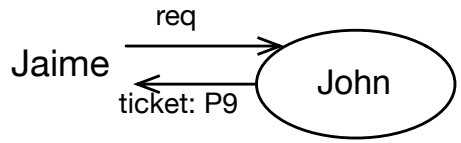
¡Implementémoslo!

Dos servicios:

- Reserva de puestos
- Entrega de entradas

Versión 1

- John recibe llamadas telefónicas
- John anota reservas en un cuaderno
- John entrega entradas



¡Sistema centralizado!, pero funciona

Book Your Concert!

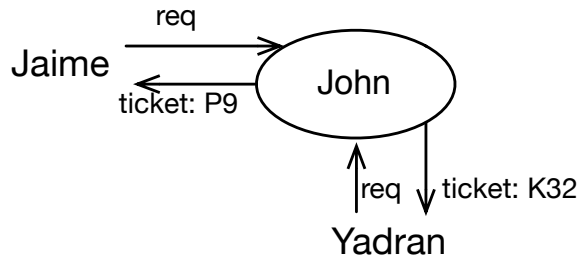
¡Implementémoslo!

Dos servicios:

- Reserva de puestos
- Entrega de entradas

Versión 1

- John recibe llamadas telefónicas
- John anota reservas en un cuaderno
- John entrega entradas



¡Sistema centralizado!, pero funciona

Book Your Concert!

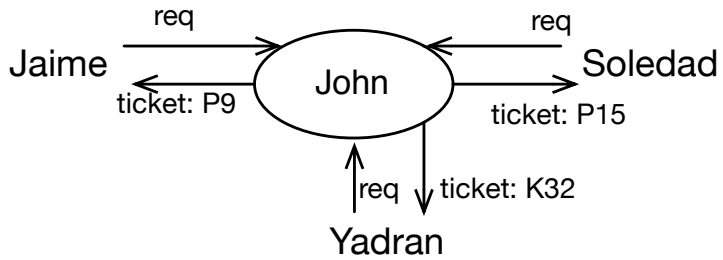
¡Implementémoslo!

Dos servicios:

- Reserva de puestos
- Entrega de entradas

Versión 1

- John recibe llamadas telefónicas
- John anota reservas en un cuaderno
- John entrega entradas



¡Sistema centralizado!, pero funciona

Book Your Concert!

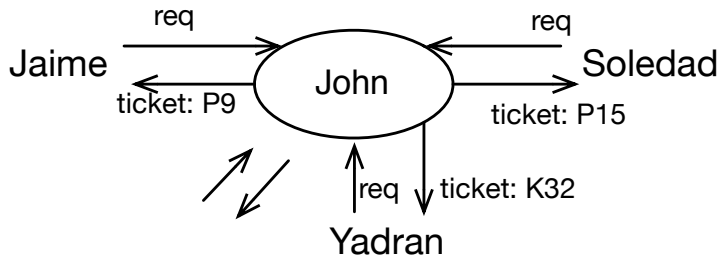
¡Implementémoslo!

Dos servicios:

- Reserva de puestos
- Entrega de entradas

Versión 1

- John recibe llamadas telefónicas
- John anota reservas en un cuaderno
- John entrega entradas



¡Sistema centralizado!, pero funciona

Book Your Concert!

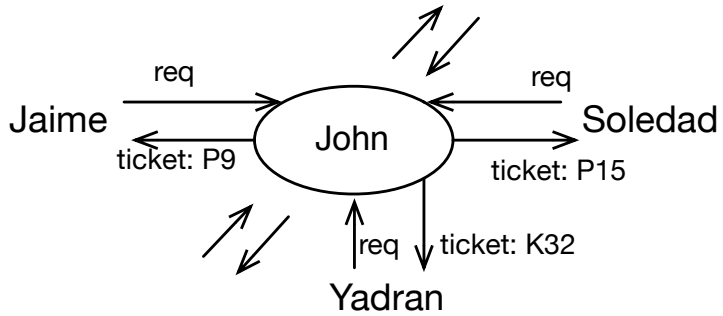
¡Implementémoslo!

Dos servicios:

- Reserva de puestos
- Entrega de entradas

Versión 1

- John recibe llamadas telefónicas
- John anota reservas en un cuaderno
- John entrega entradas



¡Sistema centralizado!, pero funciona

Book Your Concert!

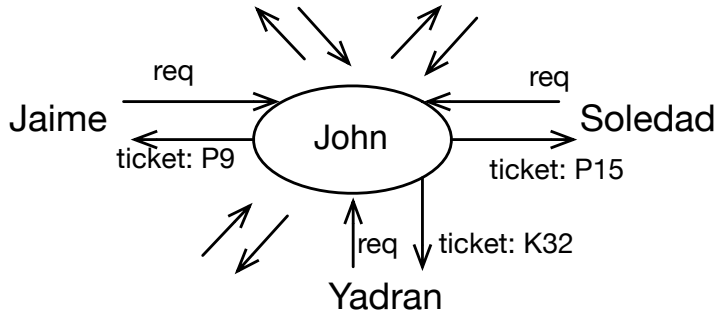
¡Implementémoslo!

Dos servicios:

- Reserva de puestos
- Entrega de entradas

Versión 1

- John recibe llamadas telefónicas
- John anota reservas en un cuaderno
- John entrega entradas

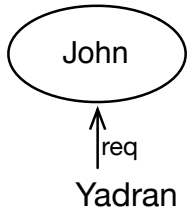


¡Sistema centralizado!, pero funciona

Book Your Concert!

¿Cómo atender la demanda? → Sistema distribuido

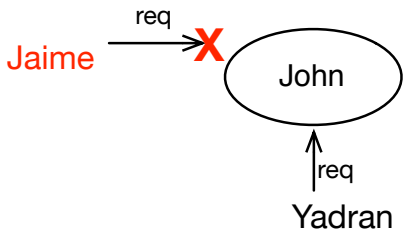
- Servicio muy popular



Book Your Concert!

¿Cómo atender la demanda? → Sistema distribuido

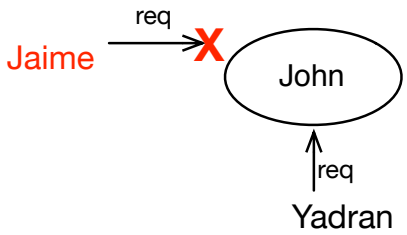
- Servicio muy popular
- Jaime llama, pero John está ocupado atendiendo a Yadran



Book Your Concert!

¿Cómo atender la demanda? → Sistema distribuido

- Servicio muy popular
- Jaime llama, pero John está ocupado atendiendo a Yadran
- Jaime se queja del mal servicio

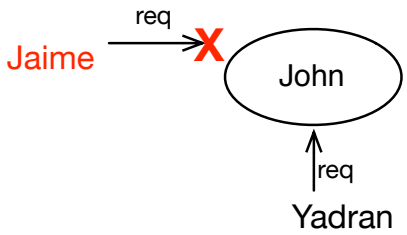


Book Your Concert!

¿Cómo atender la demanda? → Sistema distribuido

- Servicio muy popular
- Jaime llama, pero John está ocupado atendiendo a Yadran
- Jaime se queja del mal servicio

¡Baja disponibilidad (availability)!



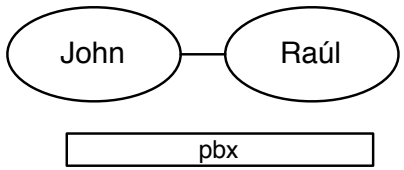
Book Your Concert!

¿Cómo atender la demanda? → Sistema distribuido

- Servicio muy popular
- Jaime llama, pero John está ocupado atendiendo a Yadran
- Jaime se queja del mal servicio

¡Baja disponibilidad (availability)!

Versión 2



Book Your Concert!

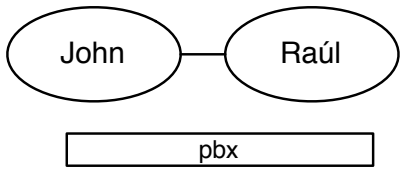
¿Cómo atender la demanda? → Sistema distribuido

- Servicio muy popular
- Jaime llama, pero John está ocupado atendiendo a Yadran
- Jaime se queja del mal servicio

¡Baja disponibilidad (availability)!

Versión 2

- John contrata a Raúl
- Ambos anotan reservas, entregan entradas y llevan sus registros
- Un pbx balancea las llamadas



Book Your Concert!

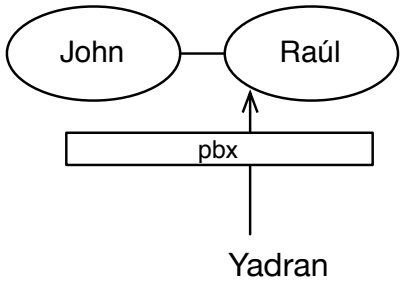
¿Cómo atender la demanda? → Sistema distribuido

- Servicio muy popular
- Jaime llama, pero John está ocupado atendiendo a Yadran
- Jaime se queja del mal servicio

¡Baja disponibilidad (availability)!

Versión 2

- John contrata a Raúl
- Ambos anotan reservas, entregan entradas y llevan sus registros
- Un pbx balancea las llamadas



Book Your Concert!

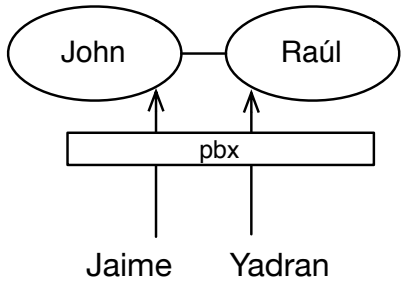
¿Cómo atender la demanda? → Sistema distribuido

- Servicio muy popular
- Jaime llama, pero John está ocupado atendiendo a Yadran
- Jaime se queja del mal servicio

¡Baja disponibilidad (availability)!

Versión 2

- John contrata a Raúl
- Ambos anotan reservas, entregan entradas y llevan sus registros
- Un pbx balancea las llamadas



Book Your Concert!

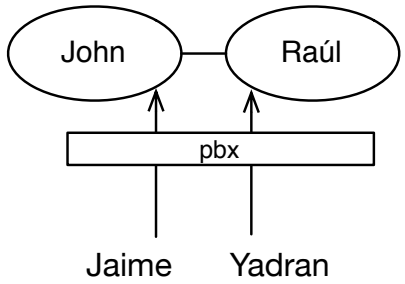
¿Cómo atender la demanda? → Sistema distribuido

- Servicio muy popular
- Jaime llama, pero John está ocupado atendiendo a Yadran
- Jaime se queja del mal servicio

¡Baja disponibilidad (availability)!

Versión 2

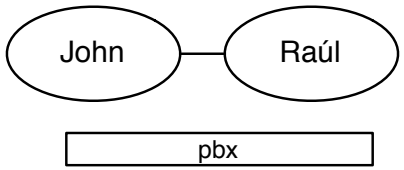
- John contrata a Raúl
- Ambos anotan reservas, entregan entradas y llevan sus registros
- Un pbx balancea las llamadas



¡Sistema ha escalado al doble de su capacidad!

Book Your Concert!

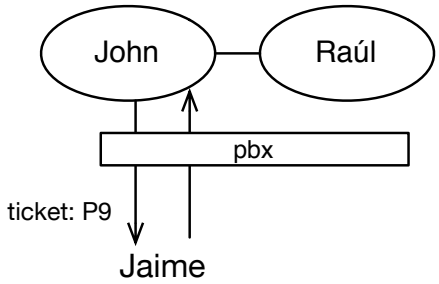
¡Clientes descontentos!



Book Your Concert!

¡Clientes descontentos!

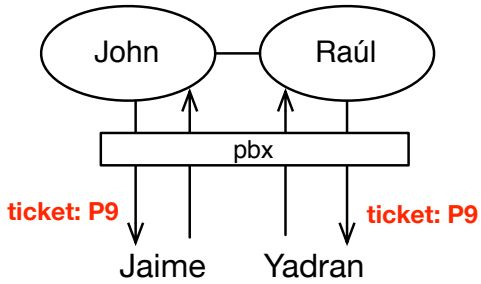
- John entrega a Jaime puesto P9



Book Your Concert!

¡Clientes descontentos!

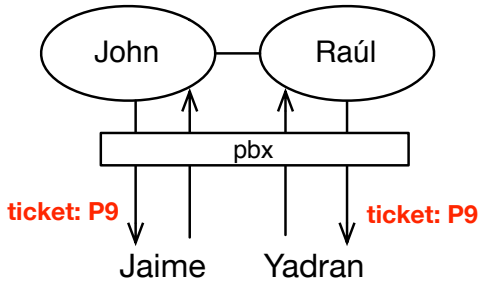
- John entrega a Jaime puesto P9
- Raúl entrega a Yadran puesto P9



Book Your Concert!

¡Clientes descontentos!

- John entrega a Jaime puesto P9
- Raúl entrega a Yadran puesto P9
- Jaime y Yadran se quejan del mal servicio

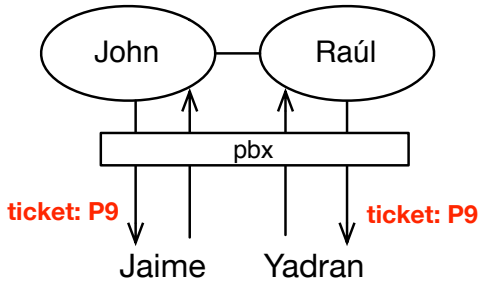


Book Your Concert!

¡Clientes descontentos!

- John entrega a Jaime puesto P9
- Raúl entrega a Yadran puesto P9
- Jaime y Yadran se quejan del mal servicio

¡No hay consistencia!



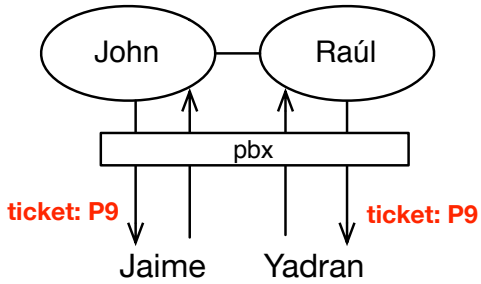
Book Your Concert!

¡Clientes descontentos!

- John entrega a Jaime puesto P9
- Raúl entrega a Yadran puesto P9
- Jaime y Yadran se quejan del mal servicio

¡No hay consistencia!

Versión 3



Book Your Concert!

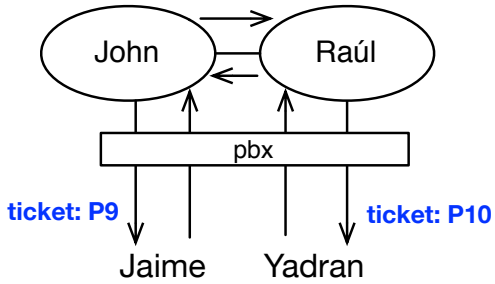
¡Clientes descontentos!

- John entrega a Jaime puesto P9
- Raúl entrega a Yadran puesto P9
- Jaime y Yadran se quejan del mal servicio

¡No hay consistencia!

Versión 3

- Cuando uno recibe una solicitud de reserva, primero debe avisar al otro.
- Después de eso la venta continúa.



Book Your Concert!

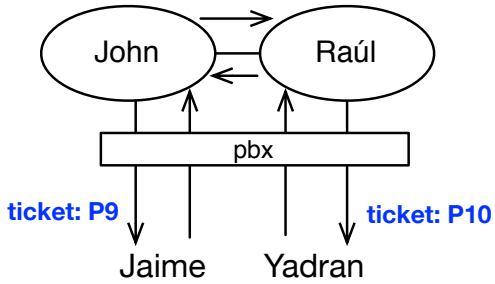
¡Clientes descontentos!

- John entrega a Jaime puesto P9
- Raúl entrega a Yadran puesto P9
- Jaime y Yadran se quejan del mal servicio

¡No hay consistencia!

Versión 3

- Cuando uno recibe una solicitud de reserva, primero debe avisar al otro.
- Después de eso la venta continúa.

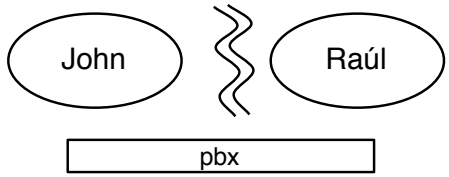


¡Consistencia solucionada!

Book Your Concert!

¿Y si uno no llega a trabajar?

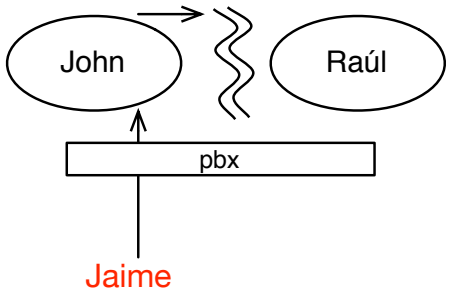
- Raúl usa el Transantiago.
- John no puede hacer reservas mientras Raúl no llegue.



Book Your Concert!

¿Y si uno no llega a trabajar?

- Raúl usa el Transantiago.
- John no puede hacer reservas mientras Raúl no llegue.
- Jaime y Yadran se quejan del mal servicio

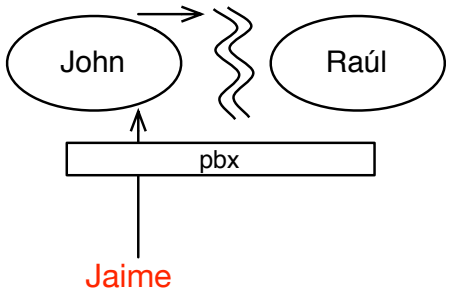


Book Your Concert!

¿Y si uno no llega a trabajar?

- Raúl usa el Transantiago.
- John no puede hacer reservas mientras Raúl no llegue.
- Jaime y Yadran se quejan del mal servicio

¡No hay disponibilidad!



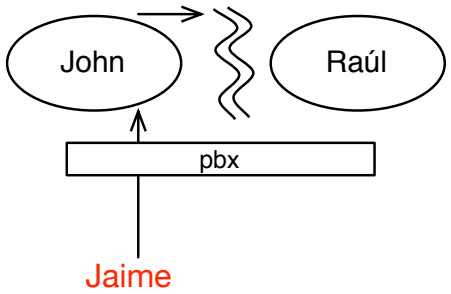
Book Your Concert!

¿Y si uno no llega a trabajar?

- Raúl usa el Transantiago.
- John no puede hacer reservas mientras Raúl no llegue.
- Jaime y Yadran se quejan del mal servicio

¡No hay disponibilidad!

Versión 4



Book Your Concert!

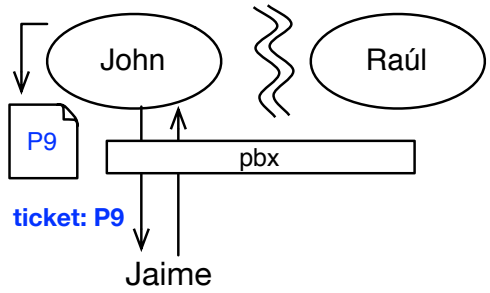
¿Y si uno no llega a trabajar?

- Raúl usa el Transantiago.
- John no puede hacer reservas mientras Raúl no llegue.
- Jaime y Yadran se quejan del mal servicio

¡No hay disponibilidad!

Versión 4

- John hace reservas y guarda un registro
- Cuando Raúl llega, le informa de los reservas hechas



Book Your Concert!

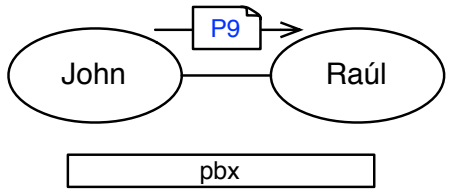
¿Y si uno no llega a trabajar?

- Raúl usa el Transantiago.
- John no puede hacer reservas mientras Raúl no llegue.
- Jaime y Yadran se quejan del mal servicio

¡No hay disponibilidad!

Versión 4

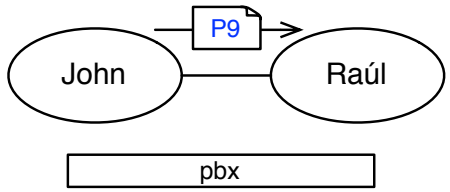
- John hace reservas y guarda un registro
- Cuando Raúl llega, le informa de los reservas hechas



Book Your Concert!

¿Y si uno no llega a trabajar?

- Raúl usa el Transantiago.
- John no puede hacer reservas mientras Raúl no llegue.
- Jaime y Yadran se quejan del mal servicio



¡No hay disponibilidad!

Versión 4

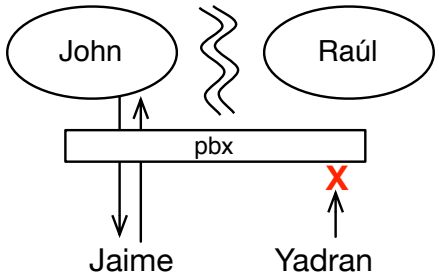
- John hace reservas y guarda un registro
- Cuando Raúl llega, le informa de los reservas hechas

¡Disponibilidad y consistencia!

Book Your Concert!

¡Oficina remota!

- Se pierde capacidad
- Misma disponiblidad que con sistema centralizado

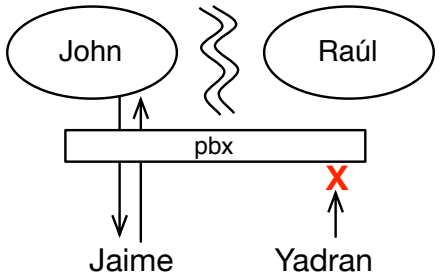


Book Your Concert!

¡Oficina remota!

- Se pierde capacidad
- Misma disponiblidad que con sistema centralizado

¡No hay tolerancia a particiones!



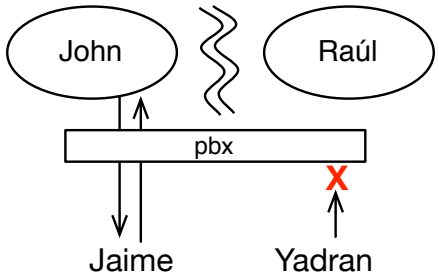
Book Your Concert!

¡Oficina remota!

- Se pierde capacidad
- Misma disponiblidad que con sistema centralizado

¡No hay tolerancia a particiones!

Versión 5



Book Your Concert!

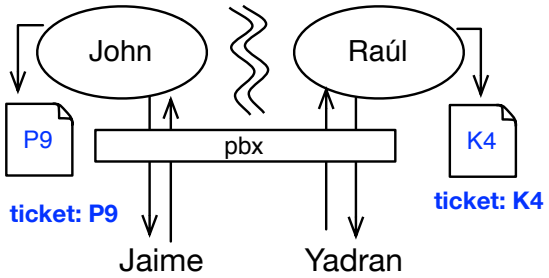
¡Oficina remota!

- Se pierde capacidad
- Misma disponiblidad que con sistema centralizado

¡No hay tolerancia a particiones!

Versión 5

- Raúl trabaja remotamente
- Cada uno anota sus reservas y guarda un registro



Book Your Concert!

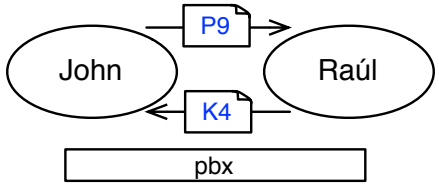
¡Oficina remota!

- Se pierde capacidad
- Misma disponiblidad que con sistema centralizado

¡No hay tolerancia a particiones!

Versión 5

- Raúl trabaja remotamente
- Cada uno anota sus reservas y guarda un registro
- Al final del día intercambian los registros



Book Your Concert!

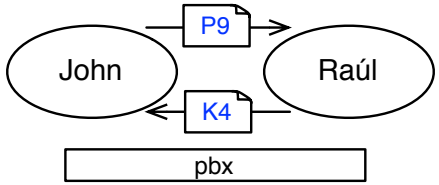
¡Oficina remota!

- Se pierde capacidad
- Misma disponiblidad que con sistema centralizado

¡No hay tolerancia a particiones!

Versión 5

- Raúl trabaja remotamente
- Cada uno anota sus reservas y guarda un registro
- Al final del día intercambian los registros

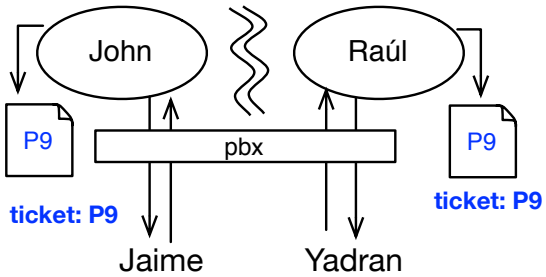


¡Disponibilidad y tolerancia a particiones!

Book Your Concert!

Pero perdimos consistencia . . .

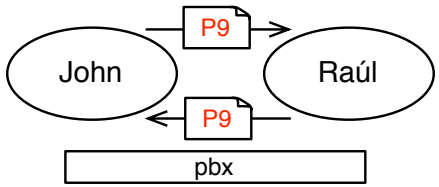
- Jaime reserva con John
- Yadran reserva con Raúl



Book Your Concert!

Pero perdimos consistencia . . .

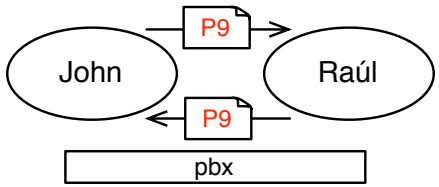
- Jaime reserva con John
- Yadran reserva con Raúl
- Reciben la misma entrada, pero no se sabe hasta que se intercambian registros.



Book Your Concert!

Pero perdimos consistencia . . .

- Jaime reserva con John
- Yadran reserva con Raúl
- Reciben la misma entrada, pero no se sabe hasta que se intercambian registros.
- Jaime y Yadran se quejan del mal servicio

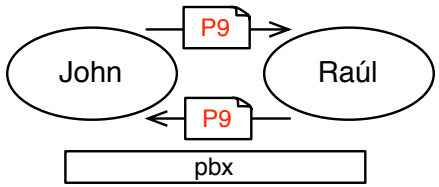


Book Your Concert!

Pero perdimos consistencia . . .

- Jaime reserva con John
- Yadran reserva con Raúl
- Reciben la misma entrada, pero no se sabe hasta que se intercambian registros.
- Jaime y Yadran se quejan del mal servicio

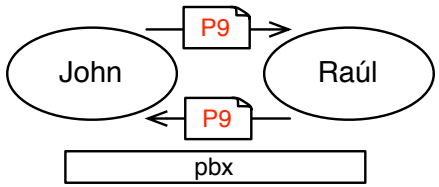
¡Perdimos la consistencia!



Book Your Concert!

Pero perdimos consistencia . . .

- Jaime reserva con John
- Yadran reserva con Raúl
- Reciben la misma entrada, pero no se sabe hasta que se intercambian registros.
- Jaime y Yadran se quejan del mal servicio



¡Perdimos la consistencia!

Versión 6

Book Your Concert!

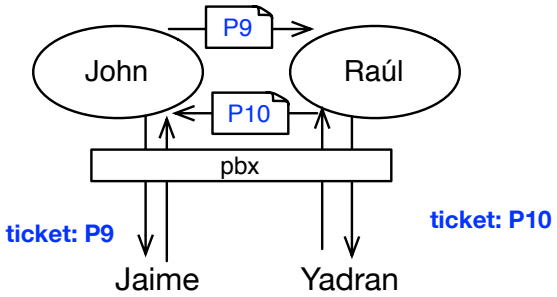
Pero perdimos consistencia . . .

- Jaime reserva con John
- Yadran reserva con Raúl
- Reciben la misma entrada, pero no se sabe hasta que se intercambian registros.
- Jaime y Yadran se quejan del mal servicio

¡Perdimos la consistencia!

Versión 6

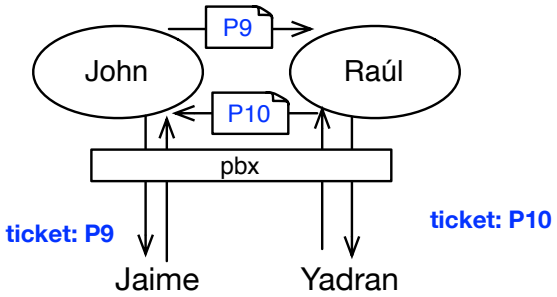
- Registros se envían inmediatamente: chat, whatsapp, email, teléfono



Book Your Concert!

Pero perdimos consistencia . . .

- Jaime reserva con John
- Yadran reserva con Raúl
- Reciben la misma entrada, pero no se sabe hasta que se intercambian registros.
- Jaime y Yadran se quejan del mal servicio



¡Perdimos la consistencia!

Versión 6

- Registros se envían inmediatamente: chat, whatsapp, email, teléfono

¡Consistencia y tolerancia a particiones!

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.
- Mensajería es asíncrona (problema de consistencia)

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.
- Mensajería es asíncrona (problema de consistencia)
- Se pueden perder mensajes (problema de consistencia)

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.
- Mensajería es asíncrona (problema de consistencia)
- Se pueden perder mensajes (problema de consistencia)
- Teléfono es síncrono, pero requiere esperar que el otro esté desocupado (problema de disponibilidad)

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.
- Mensajería es asíncrona (problema de consistencia)
- Se pueden perder mensajes (problema de consistencia)
- Teléfono es síncrono, pero requiere esperar que el otro esté desocupado (problema de disponibilidad)

¡Perdemos disponibilidad o consistencia!

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.
- Mensajería es asíncrona (problema de consistencia)
- Se pueden perder mensajes (problema de consistencia)
- Teléfono es síncrono, pero requiere esperar que el otro esté desocupado (problema de disponibilidad)

¡Perdemos disponibilidad o consistencia!

Varias soluciones ...

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.
- Mensajería es asíncrona (problema de consistencia)
- Se pueden perder mensajes (problema de consistencia)
- Teléfono es síncrono, pero requiere esperar que el otro esté desocupado (problema de disponibilidad)

¡Perdemos disponibilidad o consistencia!

Varias soluciones ...

- Volvemos a V4: Consistencia y disponibilidad

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.
- Mensajería es asíncrona (problema de consistencia)
- Se pueden perder mensajes (problema de consistencia)
- Teléfono es síncrono, pero requiere esperar que el otro esté desocupado (problema de disponibilidad)

¡Perdemos disponibilidad o consistencia!

Varias soluciones ...

- Volvemos a V4: Consistencia y disponibilidad
- Volvemos a V5: Disponibilidad y tolerancia a partición

Book Your Concert!

Pero ahora cada uno depende de la comunicación con el otro . . .

- Red puede fallar.
- Mensajería es asíncrona (problema de consistencia)
- Se pueden perder mensajes (problema de consistencia)
- Teléfono es síncrono, pero requiere esperar que el otro esté desocupado (problema de disponibilidad)

¡Perdemos disponibilidad o consistencia!

Varias soluciones ...

- Volvemos a V4: Consistencia y disponibilidad
- Volvemos a V5: Disponibilidad y tolerancia a partición
- Volvemos a V6: Consistencia y tolerancia a partición

Teorema CAP

- 2000: Eric Brewer introduce la noción de un *trade-off* inevitable entre **Consistencia**, **disponibilidad** (**A**vailability), y **tolerancia a Particiones**.
- 2002: Seth Gilbert y Nancy Lynch formalizan el Teorema CAP

Teorema CAP

In a network subject to communication failures, it is impossible for any web service to implement an atomic read/write shared memory that guarantees a response to every request.

En la práctica, un sistema distribuido solo puede proveer dos de estas características:

- Consistencia
- Disponibilidad (Availability)
- Tolerancia a Particiones

Teorema CAP

Referencias:

- Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News, 33(2):58-51, June 2002.
- Seth Gilbert and Nancy A. Lynch. Perspectives on the CAP Theorem. Computer, 45(2):30-35, 2012. IEEE.

Contenidos

Definición

Hay muchas pero . . .

Según Tanenbaum, Van Steen . . .

A distributed system is a collection of autonomous computing elementos that appears to its users as a single coherent system.

Definición

Hay muchas pero . . .

Según Tanenbaum, Van Steen . . .

A distributed system is a collection of autonomous computing elementos that appears to its users as a single coherent system.

Definición

Hay muchas pero . . .

Según Tanenbaum, Van Steen . . .

A distributed system is a collection of autonomous computing elementos that appears to its users as a single coherent system.

- *Computing elements*: nodos (*hardware*, *software*). Tenemos que **identificarlos** y **comunicarlos** entre sí.

Definición

Hay muchas pero . . .

Según Tanenbaum, Van Steen . . .

A distributed system is a collection of autonomous computing elementos that appears to its users as a single coherent system.

- *Computing elements*: nodos (*hardware*, *software*). Tenemos que **identificarlos** y **comunicarlos** entre sí.
- *autonomous*: se comportan de manera independiente. Pueden **fallar** de manera independiente. No poseen un reloj común. Hay que **sincronizarlos**.

Definición

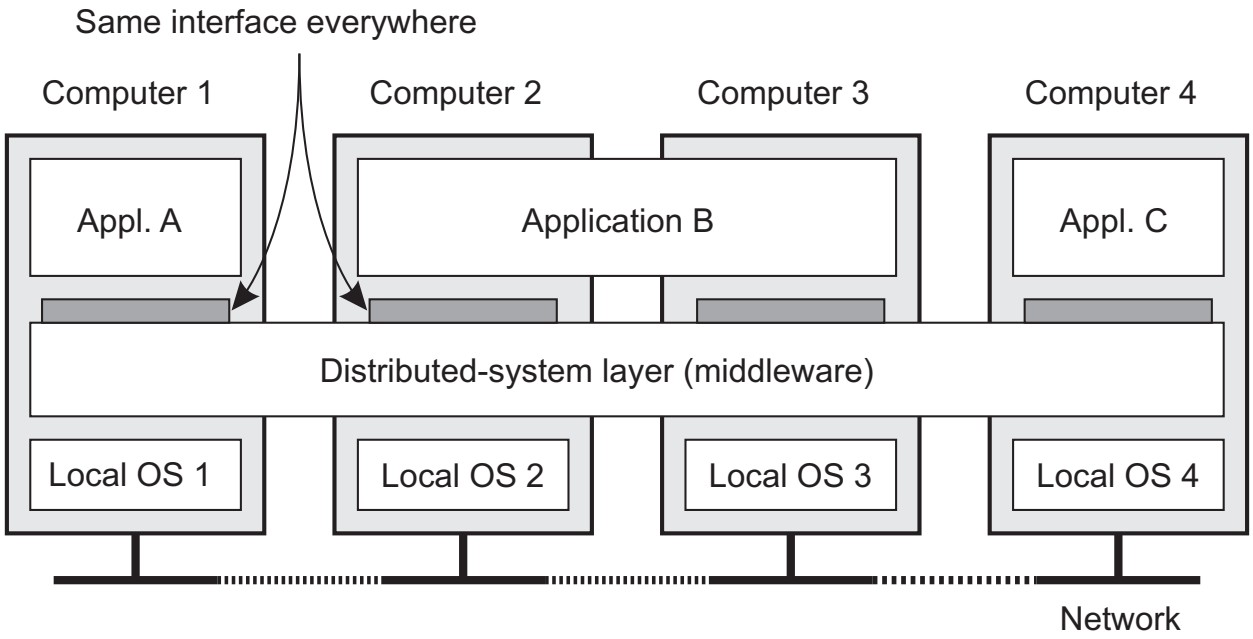
Hay muchas pero . . .

Según Tanenbaum, Van Steen . . .

A distributed system is a collection of autonomous computing elementos that appears to its users as a single coherent system.

- *Computing elements*: nodos (*hardware*, *software*). Tenemos que **identificarlos** y **comunicarlos** entre sí.
- *autonomous*: se comportan de manera independiente. Pueden **fallar** de manera independiente. No poseen un reloj común. Hay que **sincronizarlos**.
- *appears as a single coherent system*: tiene que **coordinarse** y mantener algún nivel de **consistencia**. Transparencia de distribución.

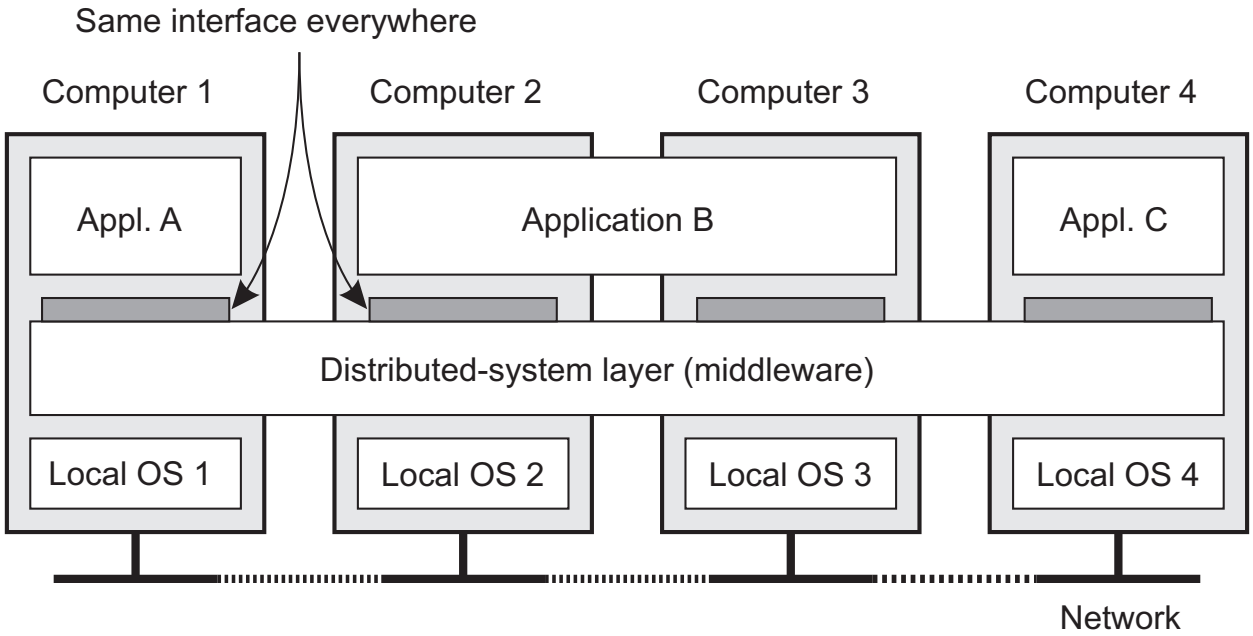
Middlewares



Esconde (hasta cierto punto) diferencias entre *hardware* y sistemas operativos

- Comunicación entre aplicaciones
- Seguridad y *accountability*
- Enmascaramiento y recuperación de errores

Middlewares



Esconde (hasta cierto punto) diferencias entre *hardware* y sistemas operativos

- Comunicación: invocaciones remotas, RPC
- Transacciones distribuidas, propiedades ACID
- Composición de servicios: *web-services*, *mashups*
- Reliability: mensajes son recibidos por todos o por ninguno

Contenidos

Falacias de la programación distribuida

Peter Deutsch (Sun Microsystems) planteó[1] (1991):

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

[1] https://blogs.oracle.com/jag/resource/Fallacies.html

Diseño de sistemas distribuidos

Solo porque se pueda construir un sistema distribuido, no significa que sea una buena idea.

- Sistema debe soportar **resource sharing**
- La distribución debe ser **transparente**
- El sistema debe ser **abierto**
- El sistema deber ser **escalable**

Resource sharing

Software para trabajo colaborativo: **groupware**

Algunos ejemplos

- Almacenamiento en la nube
- Servicios de *streaming peer-to-peer* (P2P)
- Sistemas de correo compartido (*outsource*)
- Sistemas web compartidos (*Content Distribution Networks*)

Cita

"The network is the computer"

John Gage, Sun Microsystems

Distribución debe ser transparente

Transparency	Description
Access	Hide differences in data representation and how an object is accessed
Location	Hide where an object is located
Relocation	Hide that an object may be moved to another location while in use
Migration	Hide that an object may move to another location
Replication	Hide that an object is replicated
Concurrency	Hide that an object may be shared by several independent users
Failure	Hide the failure and recovery of an object

Distribución debe ser transparente

Transparency	Description
Access	Hide differences in data representation and how an object is accessed
Location	Hide where an object is located
Relocation	Hide that an object may be moved to another location while in use
Migration	Hide that an object may move to another location
Replication	Hide that an object is replicated
Concurrency	Hide that an object may be shared by several independent users
Failure	Hide the failure and recovery of an object

¿Cuánta transparencia es "buena"?

Distribución debe ser transparente

El grado de transparencia es importante.

¿Quiero transparencia de distribución?

- Si usamos servicios que aprovechan geolocalización
- Si tenemos componentes/usuarios en distintas zonas horarias
- Para entender la razón de un efecto o falla

Distribución debe ser transparente

El grado de transparencia es importante.

¿Quiero transparencia de distribución?

- Si usamos servicios que aprovechan geolocalización
- Si tenemos componentes/usuarios en distintas zonas horarias
- Para entender la razón de un efecto o falla

¿Siempre transparente? (nunca intrasparente)

El grado de transparencia necesario depende la aplicación.

- No podemos ocultar toda la latencia
- No podemos ocultar todas las fallas
- Balance entre grado de transparencia y rendimiento

Sistema debe ser **abierto**

Sistema debe ser capaz de interactuar con otros a pesar de los detalles internos.

Sistemas abiertos

- Sistema debe proveer y cumplir con **interfaces definidas**
- Sistema debe ser capaz de **interoperar** con otros
- Sistema debe soportar **portabilidad** de aplicaciones
- Sistema debe ser **extensible**

Sistema debe ser **escalable**

¿En qué sentido debe escalar?

Componentes de escalamiento

- Número de usuario y procesos: **escalabilidad de tamaño**
- Máximo distancia entre componentes: **escalabilidad geográfica**
- Número de dominios administrativos: **escalabilidad administrativa**

¿Y en qué topamos?

Límites a la escalabilidad

- CPUs limitan la escalabilidade computacional
- Transferencia CPU/disco limita la escalabilidad del almacenamiento
- La red entre el usuario y un servicio centralizado

Sistema debe ser **escalable**

¿Y en qué topamos?

Límites a la escalabilidad

- CPUs limitan la escalabilidade computacional
- Transferencia CPU/disco limita la escalabilidad del almacenamiento
- La red entre el usuario y un servicio centralizado

¿Cómo lo resolvemos?

Sistema debe ser **escalable**

¿Y en qué topamos?

Límites a la escalabilidad

- CPUs limitan la escalabilidade computacional
- Transferencia CPU/disco limita la escalabilidad del almacenamiento
- La red entre el usuario y un servicio centralizado

¿Cómo lo resolvemos?

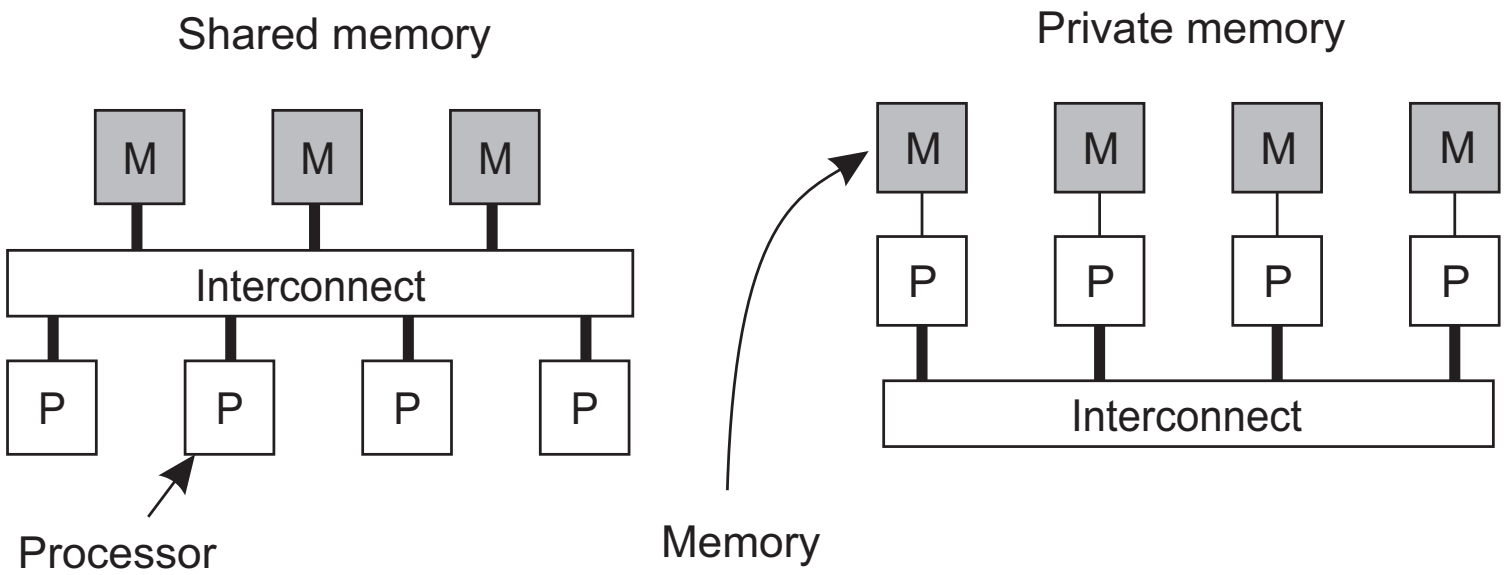
Escondiendo la latencia

- Comunicación **asíncrona**
- Uso de *threads/handlers* para manejar respuestas
- Particionar datos y dominios
- Sistemas decentralizados de ubicación e información
- Ejecución local
- Replicación

Contenidos

Sistemas distribuidos para cómputo de alto rendimiento

El cómputo de alto rendimiento dsitrbiuido empezó con el cómputo paralelo
Multiprocesadores y *multicore*, versus multicomputadores

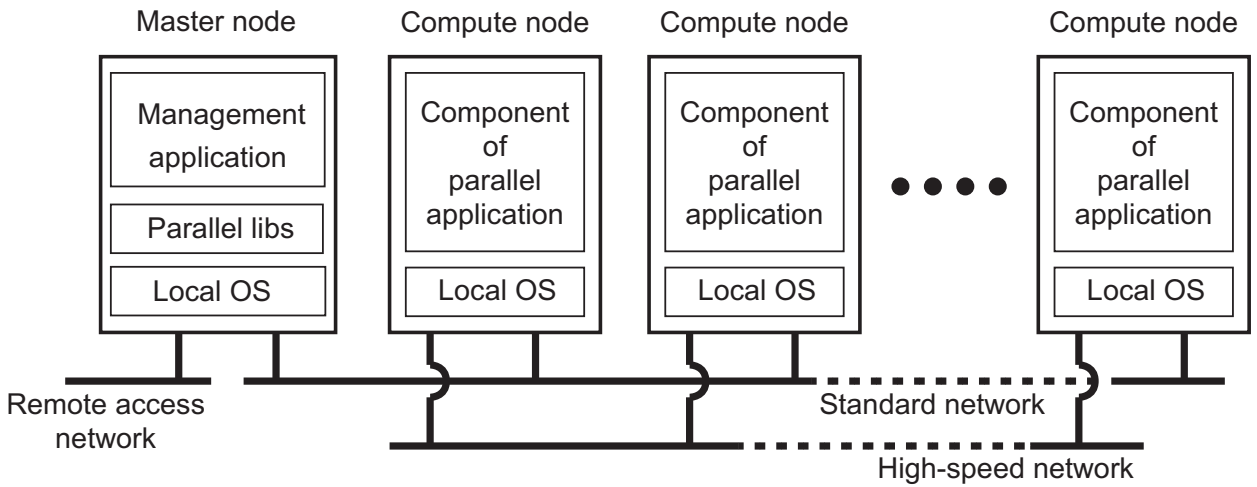


¿Cómo programar en multiprocesadores?

Cómputo en *Cluster*

Computadores conectados a una LAN

- Homogéneos: mismo S.O., y *hardware*
- Nodo de administración (*master*)



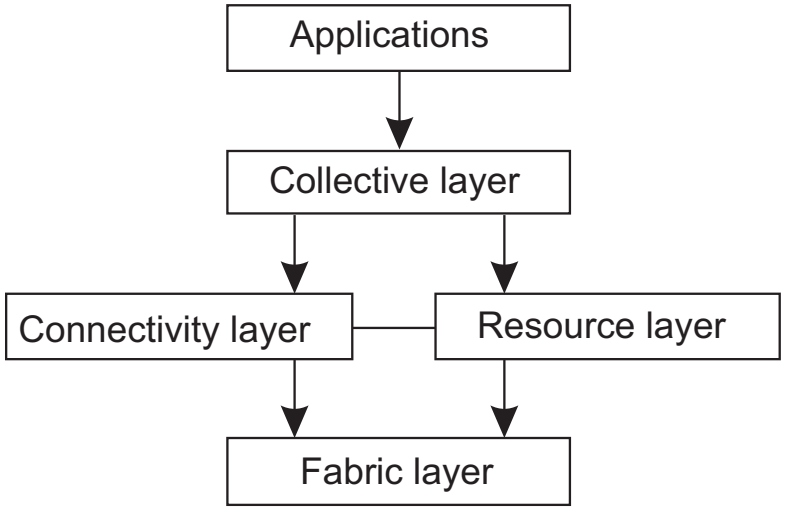
- Ej: Linux-based Beowulf clusters
- *Master* con un *resource manager* (*middleware*)
- Ej: MOSIX. *Single-system image*. Migración de trabajos. Transparencia.

Cómputo en *Grid*

Clusters federados

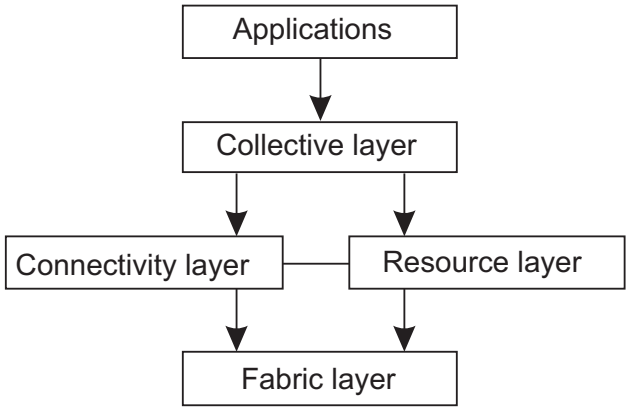
- Heterogéneos
- Dispersos a través de múltiples organizaciones
- Accesible mediante WANs

La administración se lleva a cabo mediante **virtual organization** (VO), que controlan el acceso a los recursos.



Foster et al, 2001

Cómputo en *Grid*



- Fabric. Interfaz de acceso a recursos locales (estado, *locking*, . . .) de un sitio.
- Connectivity. Autenticación y transferencia de datos entre sitios.
- Resource. Administración de cada recurso
- Collective. Administración de conjuntos de recursos: *discovery, allocation, scheduling, . . .*)
- Aplications. Aplicaciones ejecutando sobre una VO.

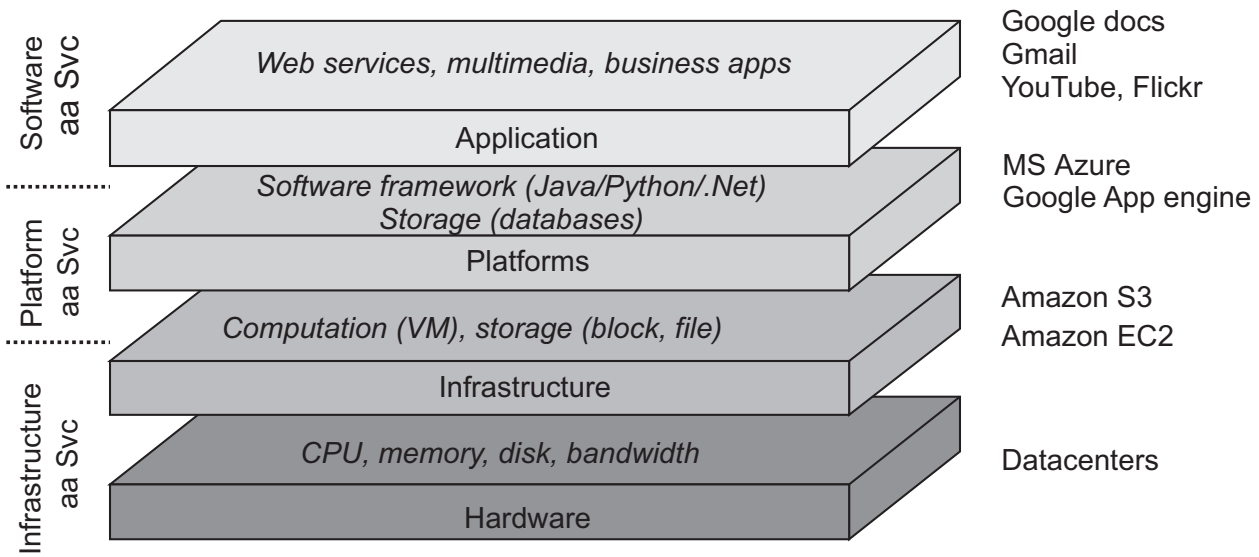
Connectivity + Resource + Collective = **Grid Middleware**

- OGSA: Open Grid Services Architecture. Foster et al. 2006.

Cómputo en *Cloud*

Utility Computing

- Conjunto de recursos **virtualizados** a distintos niveles
- Configurables dinámicamente, y escalabilidad
- *Pay-per-use* y *Service Level Agreements*: SLAs.



Zhang et al, 2010

Sistemas distribuidos para información

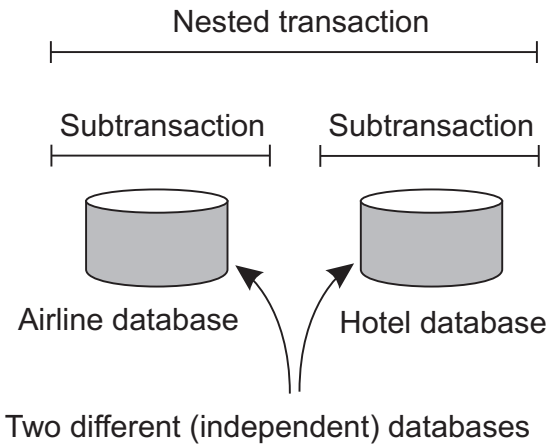
Sistemas construidos para **integración** de múltiples **servicios preexistente en red**.
Atacan el problema de la *interoperabilidad*

Sistemas de integración distribuida

- Conjunto de clientes y un servidor
- Integración (inicial) simple: servidor administra solicitud de múltiples clientes, y ofrece visión coherente al usario
- EAI: *Enterprise Application Integration*

Enterprise Application Integration: Transacciones anidadas

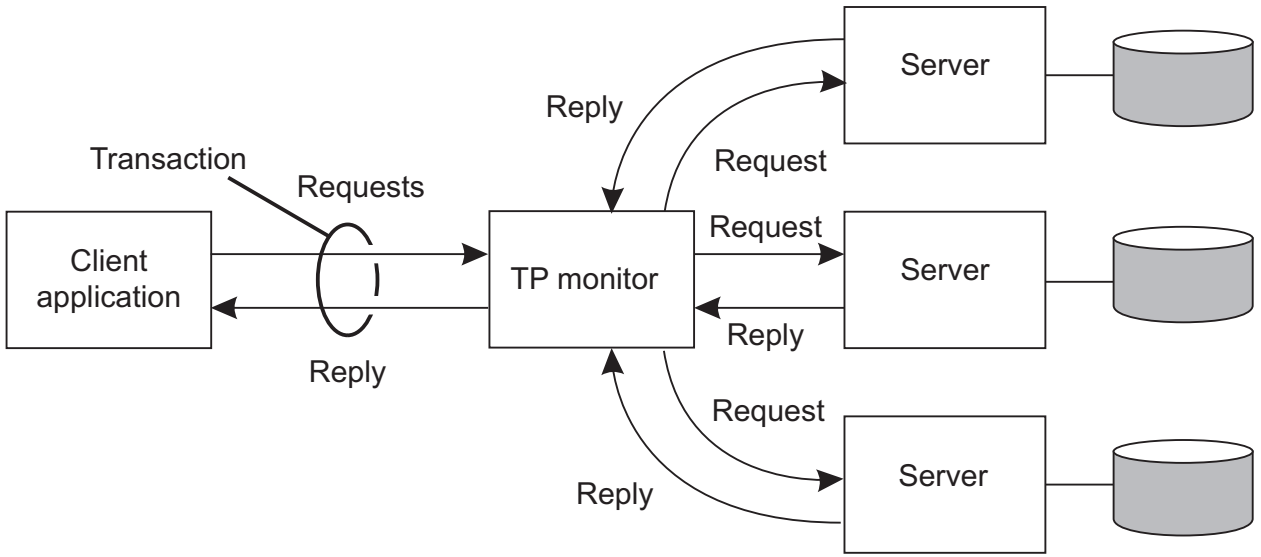
Primitive	Description
BEGIN_TRANSACTION	Mark the start of a transaction
END_TRANSACTION	Terminate the transaction and try to commit
ABORT_TRANSACTION	Kill the transaction and restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise



Transacciones distribuidas: todo o nada

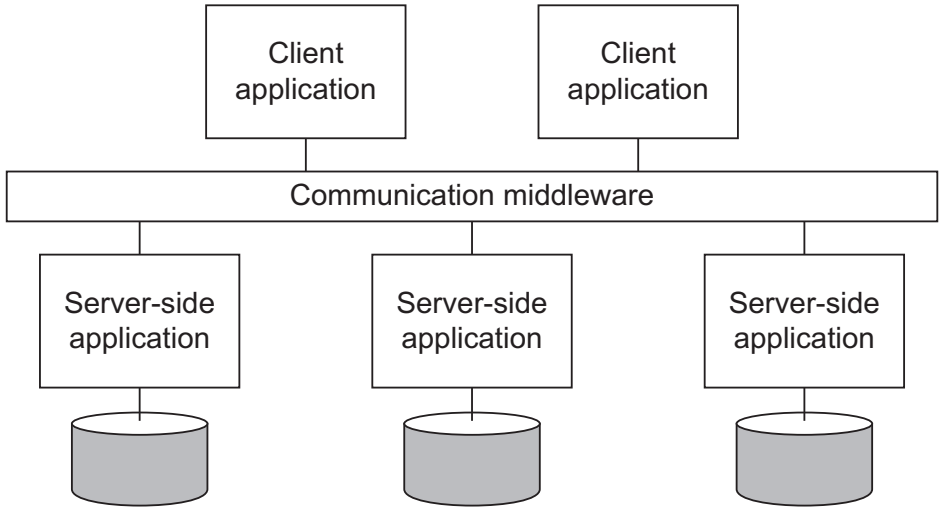
- **Atomic**: ocurren de manera indivisible
- **Consistente**: no viola invariantes del sistema
- **Isolated**: no interfiere con otras transaciones
- **Durable**: *commit*s son permanentes

TPM: Transaction Processing Monitor



TP Monitor funciona como coordinador.

Middleware y EAI



Middlware provee medios de integración

- RPC. *Remote Procedure Call*. Modelo *request/reply*
- MOM. *Message Oriente Middleare*. Modelo *publish/subscribe*

Sistemas distribuidos "pervasivos"

Nodos pequeños y móviles dentro sistemas grandes

- Sistema naturalmente integrado con el entorno
- **Cómputo ubicuo**. "Pervasive" y **contínuamente presentes**
- **Cómputo móvil**. "Pervasive", pero inherentemente móviles
- **Redes de sensores**. "Pervasive", con enfoque en interacción con el ambiente a través de **sensores** y **actuadores**

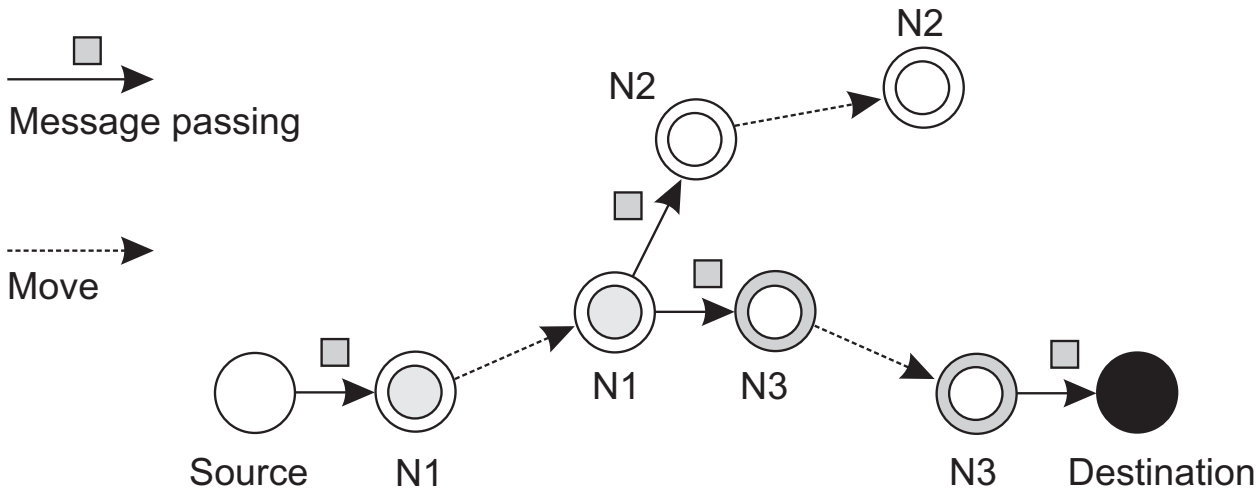
Sistemas distribuidos ubicuos

Elementos principales

- Están **distribuidos**, con transparencia de acceso
- **Interactúan** naturalmente de forma liviana
- **Context Awareness**. Conocen el contexto del usuario.
- **Autónomos**. No necesitan intervención humana.
- **Inteligentes**. Amplia gama de acciones.

Sistemas distribuidos móviles

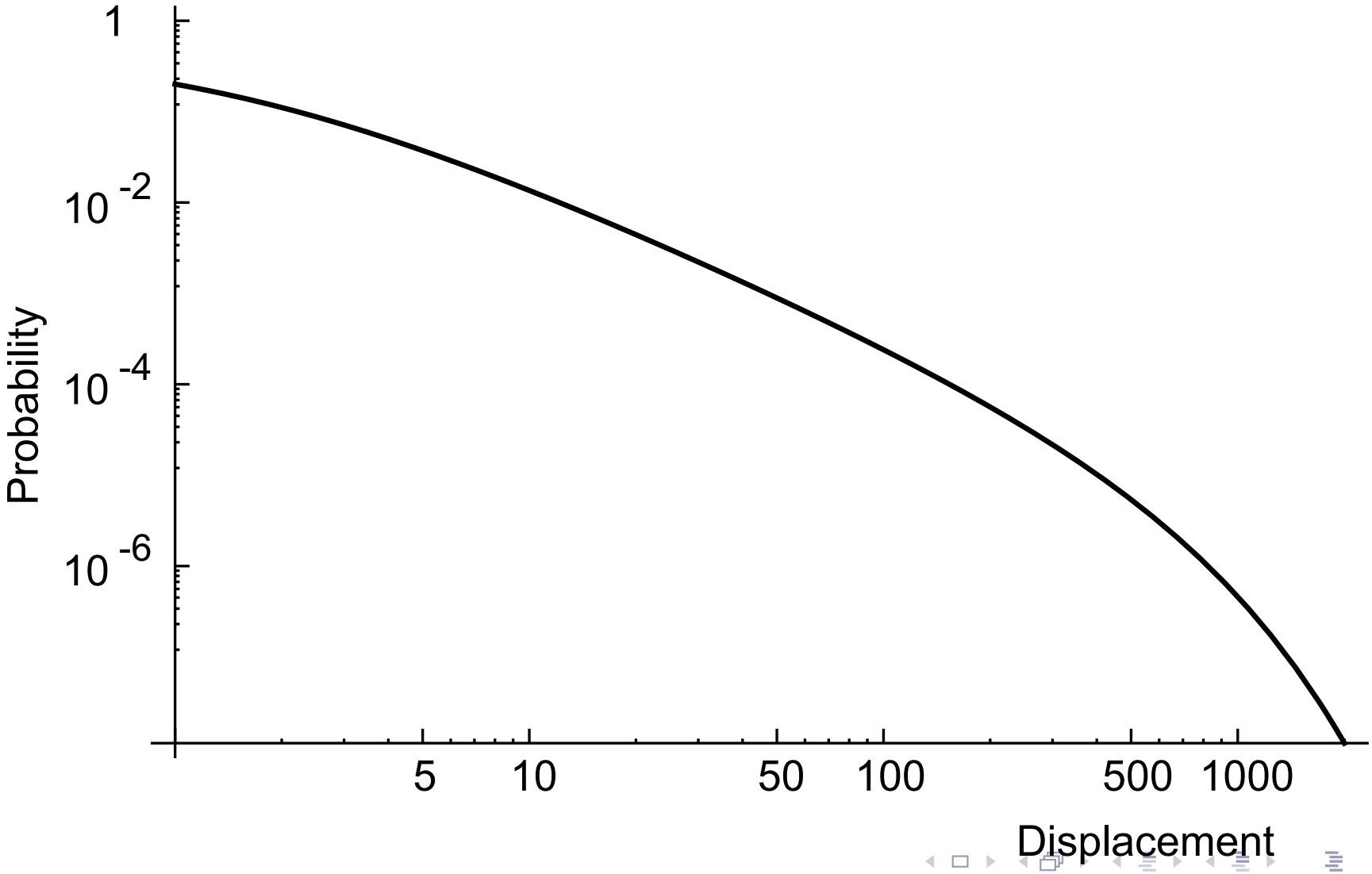
Se espera que la ubicación del usuario pueda cambiar.



Algoritmos especial de *flooding* para encontrar ubicación dinámicamente.

Sistemas distribuidos móviles

¿Cuán móviles somos?



Redes de sensores

Elementos principales

- Muchos nodos (10's-1000's)
- Pequeños y simples (memoria, procesador, red)
- Dependiente de baterías

Redes de sensores