



Ingeniería de Software III

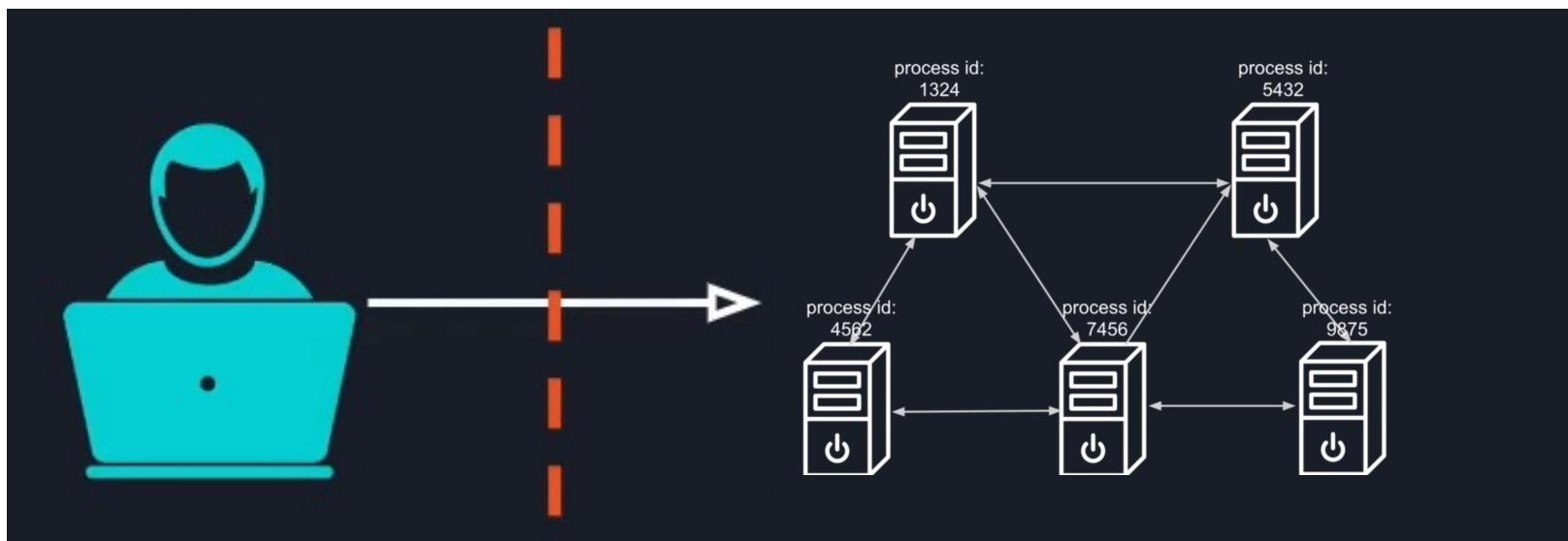
Diseño de Sistemas
Distribuidos

Parte 1

Dónde podemos encontrar sistemas distribuidos



- Los sistemas distribuidos están en todas partes
- Cada vez que:
 - Vemos una película on demand
 - Compramos online
 - Pedimos un taxi usando nuestro smartphone
 - Buscamos algo online



Sistemas distribuidos



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

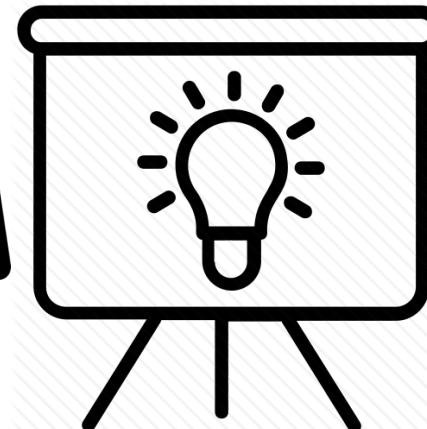
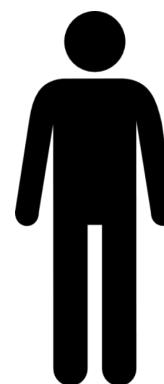
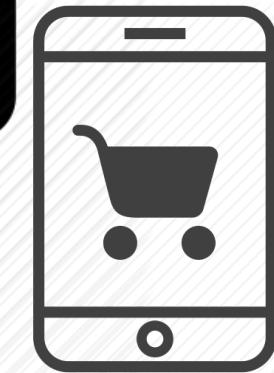
- Idealmente
 - Los usuarios no son conscientes de las complejidades del sistema que están utilizando
 - Aparecen ante el usuario como si el sistema estuviera siendo ejecutado en una sola máquina dedicada exclusivamente a él/ella

Transparencia: ¿El sistema aparece como único ante el usuario?

Startup



- Sitio de comercio electrónico
 - Comprar
 - Dejar reviews
- Los usuarios pueden acceder via browser
 - Desde la pc
 - Desce el teléfono



Escalabilidad



- Con relativamente pocos usuarios el tiempo de respuesta es aceptable



Tiempo de respuesta (RT): tiempo que un sistema requiere para procesar un pedido visto desde fuera.

Escalabilidad: una medida que indica como se ve afectada la performance cuando se agregan nuevos recursos

Escalabilidad



- Cuando la base de usuarios crece el tiempo de respuesta se degrada considerablemente
- El servidor tiene que soportar una carga mucho mayor



Tiempo de respuesta (RT): tiempo que un sistema requiere para procesar un pedido visto desde fuera.

Carga: Es una medida de cuan estresado están un sistema. Usado generalmente como contexto de otra medida de performance. Ej: el RT = 0.5 s con 10 usuarios y 2 s con 20 usuarios

Escalabilidad Vertical



- Compramos el servidor más avanzado, con el máximo de memoria y de poder de procesamiento posibles
- El servidor más costoso



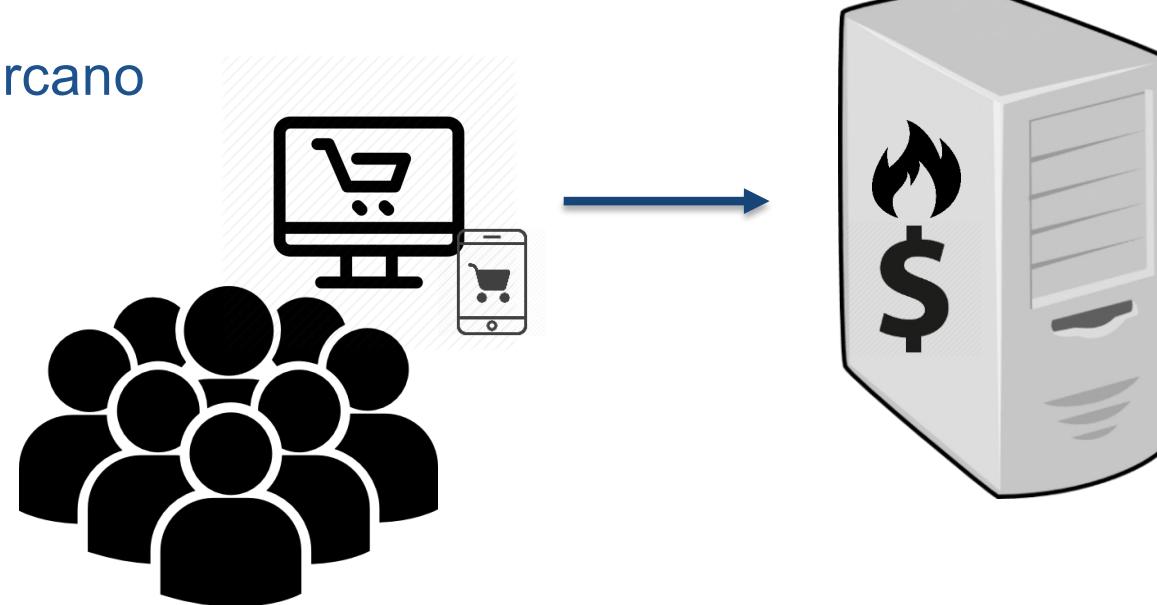
Escalabilidad Vertical

Scale Up: Agregar más o mejores recursos a un servidor

Escalabilidad Vertical



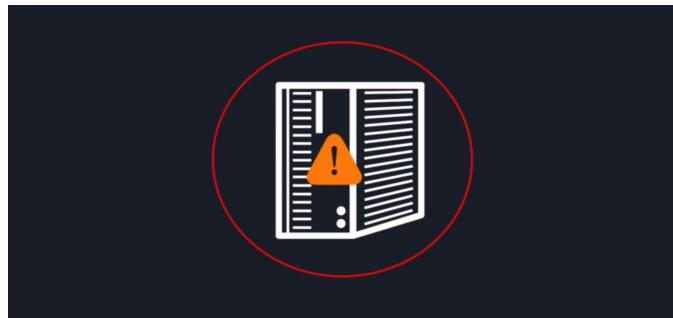
- Escalabilidad Vertical
 - Cambiar el hardware por uno más potente
 - Más memoria, procesador más rápido, más disco, etc
- Demora el problema
 - Si la base de usuarios sigue creciendo
- Tiene un límite cercano



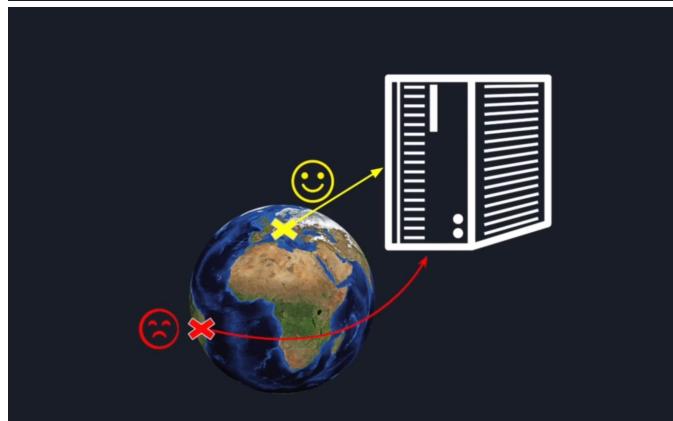
Problema de los sistemas centralizados



- Escalabilidad
 - Vertical
 - Limitada



- Un único punto de fallo
 - Error
 - Mantenimiento



- Localización
 - Aumenta la latencia con la distancia
 - Experiencia de usuario degradada para usuarios remotos

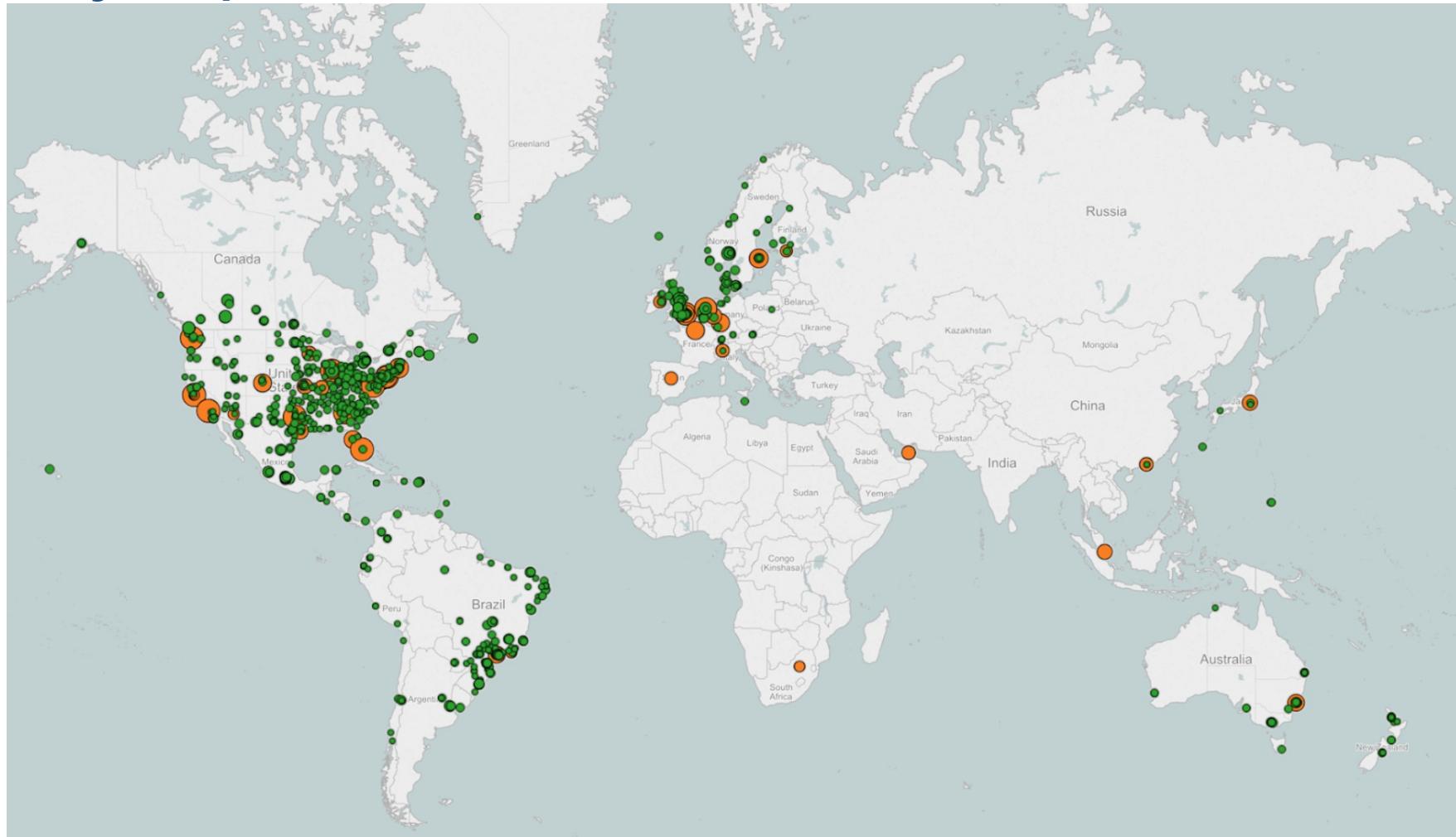
Latencia: El tiempo mínimo requerido para obtener cualquier tipo de respuesta

Localización



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Ejemplo Netflix CDN



Capacidad de los sistemas distribuidos



- Un sistema distribuido bien diseñado
 - Escalabilidad horizontalmente
 - Para manejar el aumento de carga
 - Para manejar el aumento de datos
 - Es altamente escalable
 - Idelamente de manera lineal
 - Aumentar y disminuir la capacidad según la demanda
 - Es altamente disponible
 - No tiene un solo punto de falla
 - Experiencia de usuario
 - Latencia independiente de dónde se encuentra el usuario

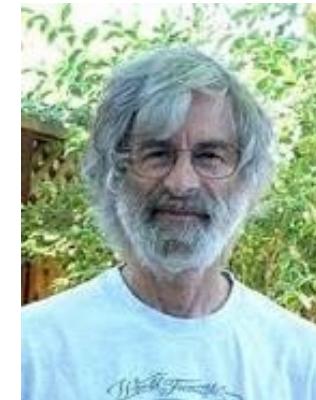
Scale Out: Agregar más servidores

¿Qué es un Sistema Distribuido?



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

“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](#)

- Un sistema que consiste de varias computadoras trabajando de manera conjunta

Definición de sistema distribuido



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

“ Un conjunto de computadoras independientes que aparecen ante el usuario como una única computadora.”

Concurrencia

Todas las computadoras operan al mismo tiempo

Independencia

Las computadoras funcionan y fallan de manera independiente

No hay reloj global

Las computadoras no comparten un reloj global

¿Qué es un Sistema Distribuido?



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

“Un conjunto de nodos de computación, conectados a través de una red, el cual aparece ante sus usuarios como un solo sistema coherente”

Tanenbaum and Van Steen



Definición de sistema distribuido



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

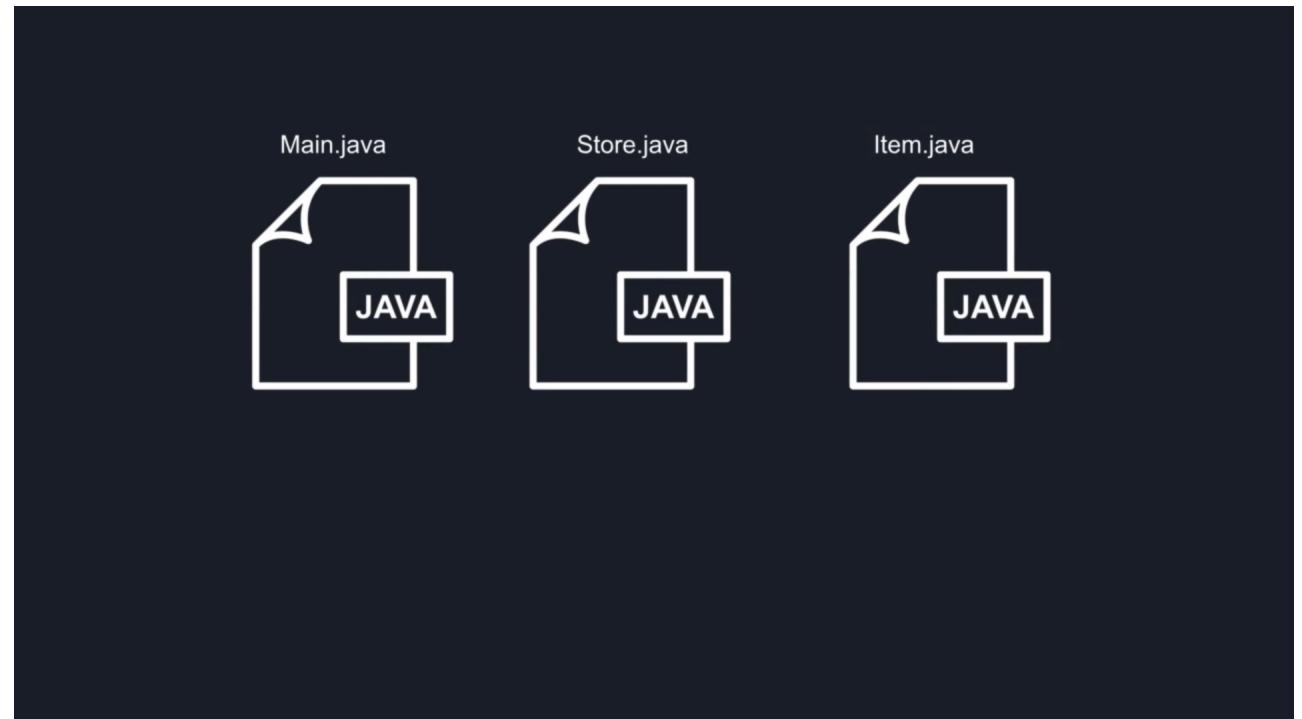
“ Un sistema distribuido es un sistema que consiste de varios procesos ejecutándose en computadoras diferentes que se comunican entre sí a través de una red y comparten un estado o trabajan en conjunto para lograr un objetivo común. ”

Proceso



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

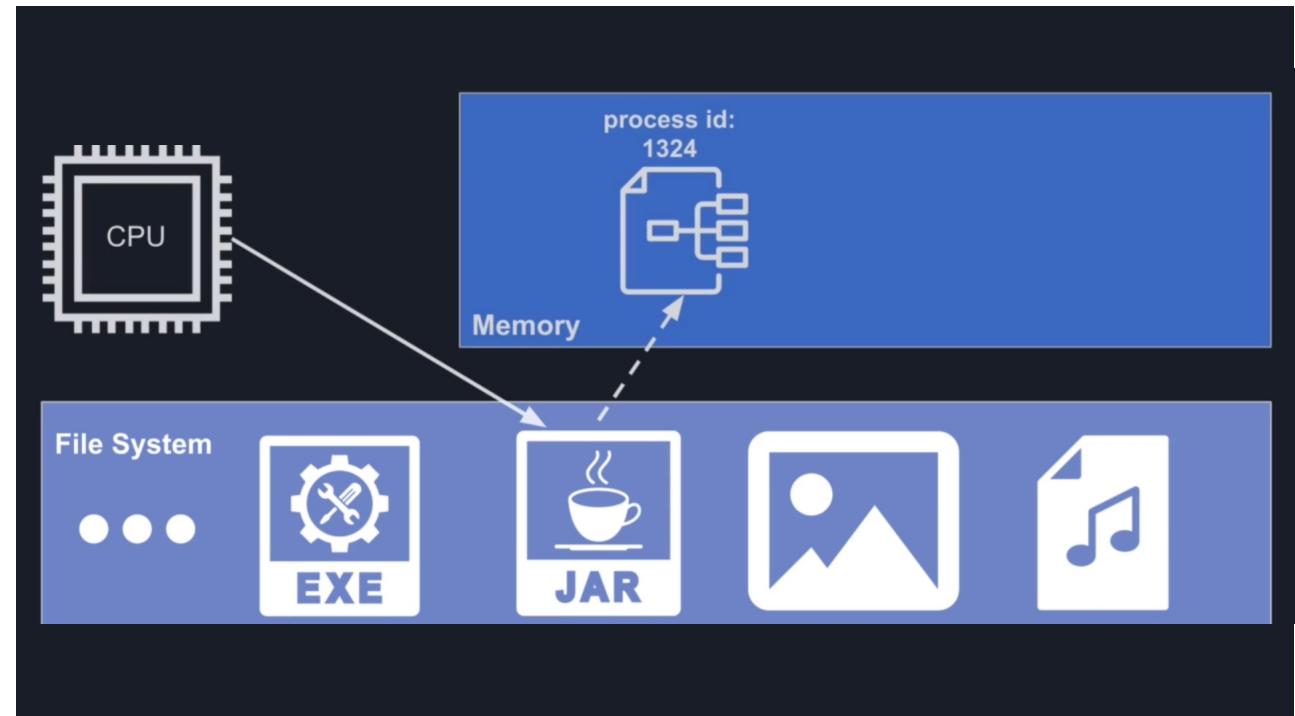
- Proceso
 - Instancia de una aplicación



Proceso



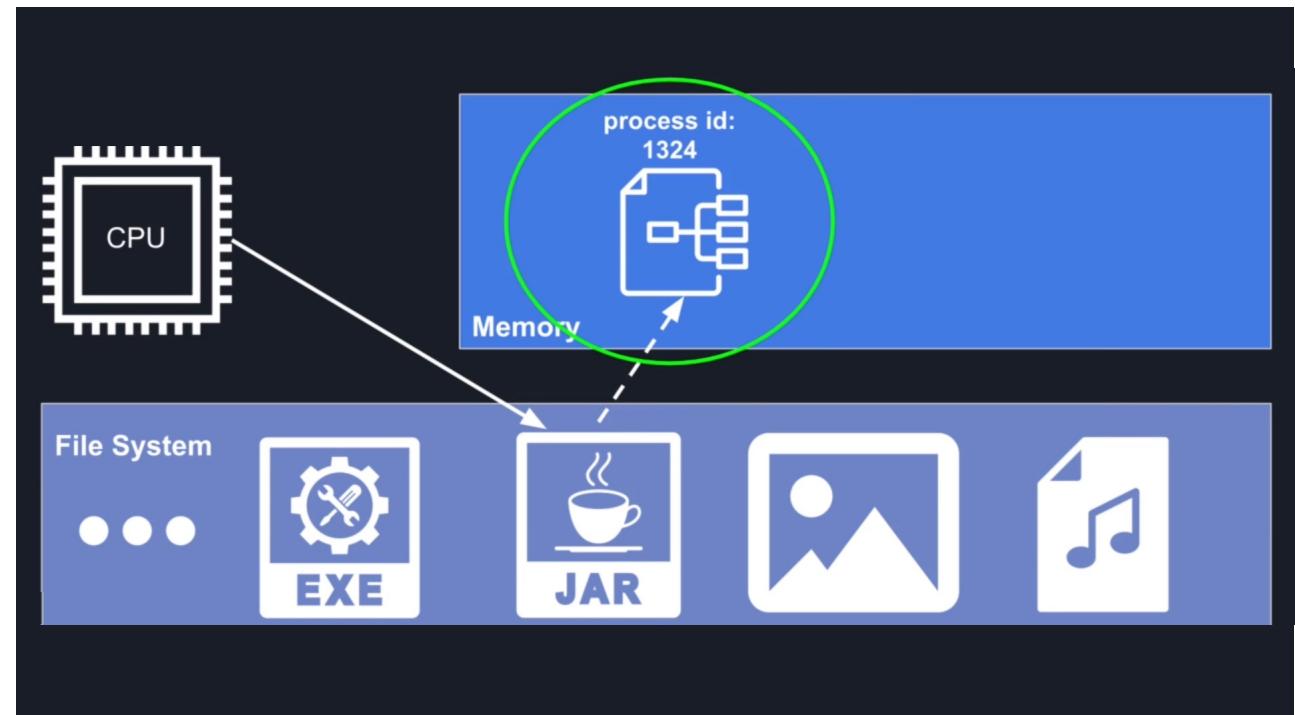
- Proceso
 - Instancia de una aplicación
 - En memoria



Proceso



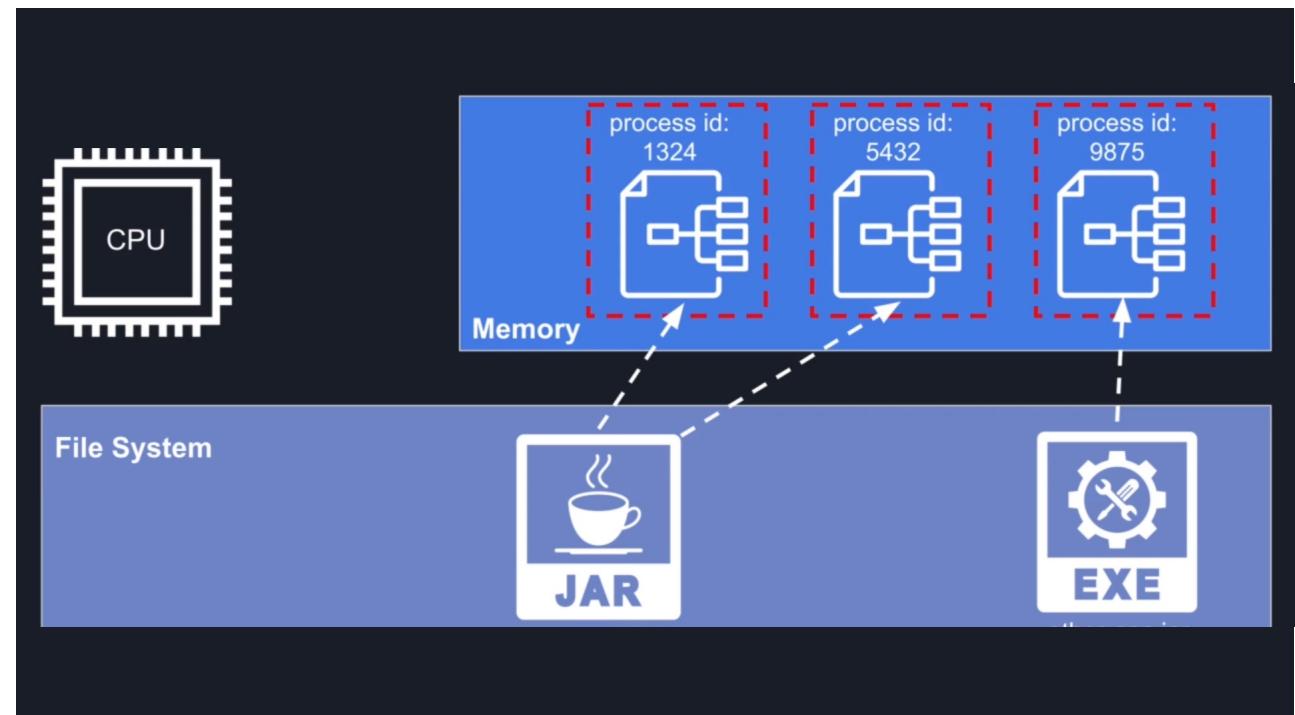
- Proceso
 - Instancia de una aplicación
 - En memoria



Proceso



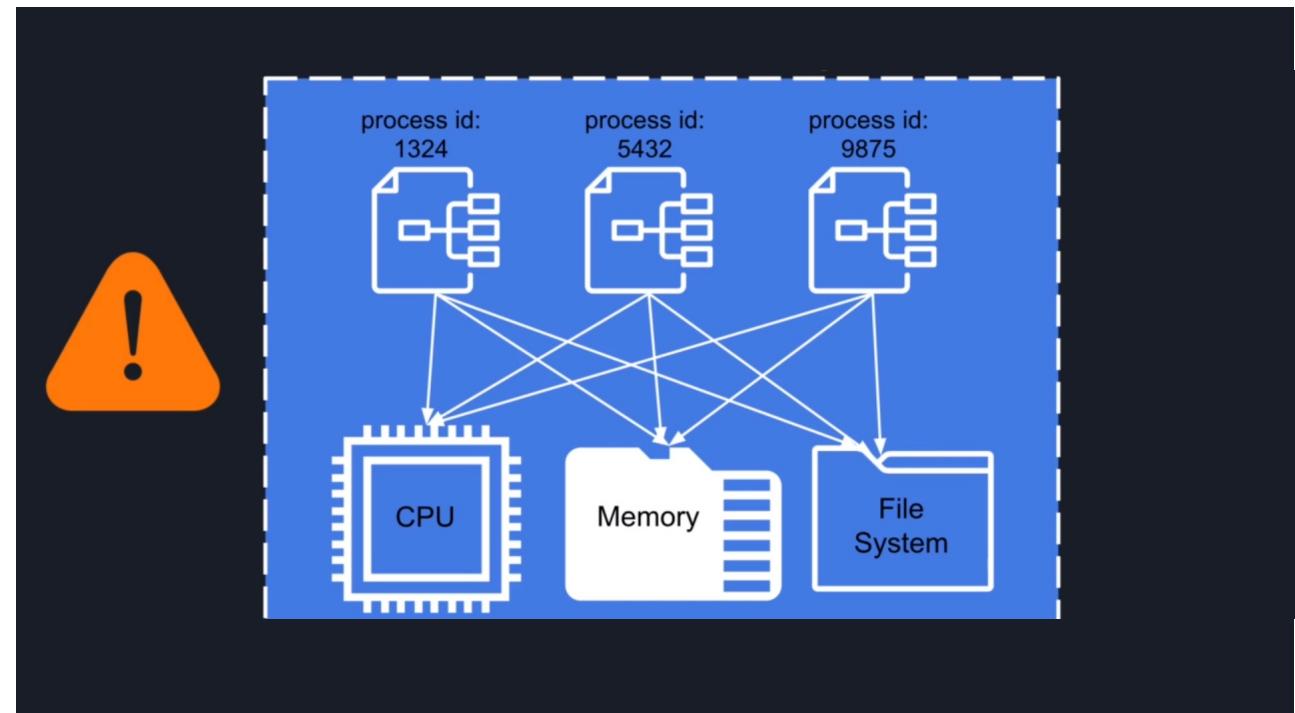
- Proceso
 - Instancia de una aplicación
 - En memoria
 - Aislados entre sí



Proceso



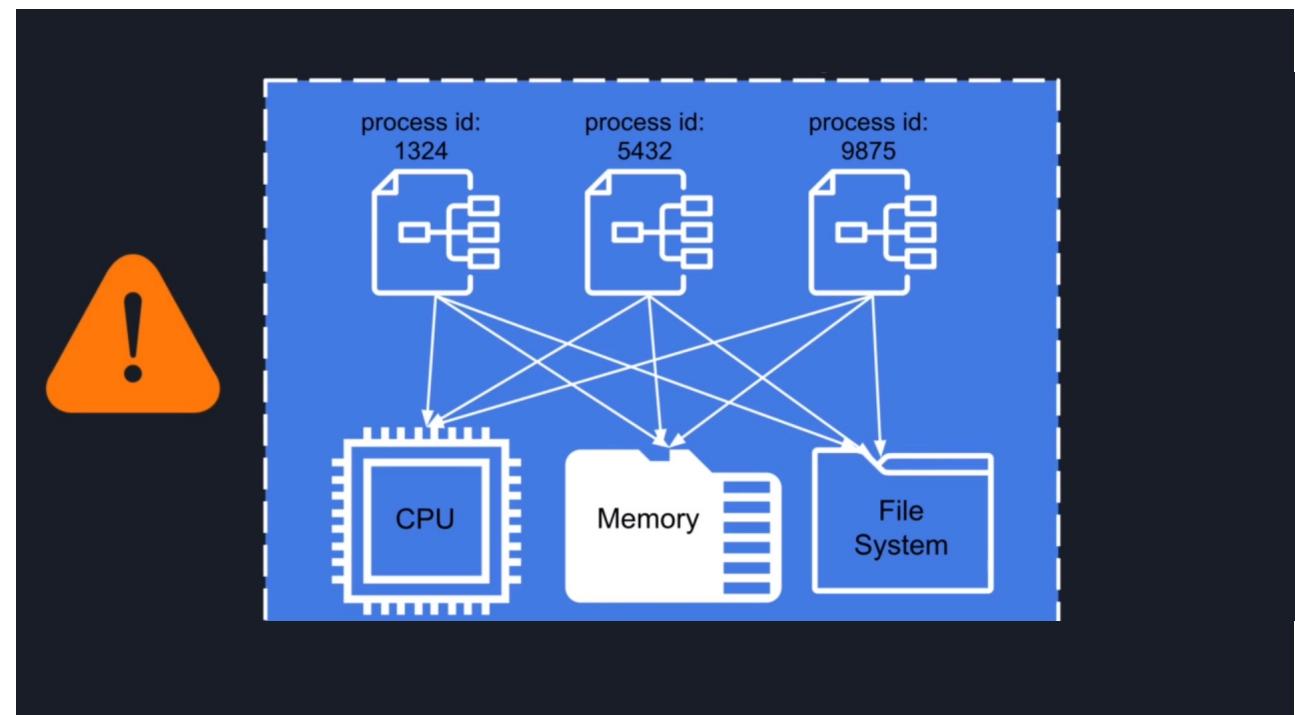
- Comunicación entre procesos
 - Network
 - Filesystem
 - Memoria



Proceso



- **No es un sistemas distribuido**
- Los procesos están en la misma computadora
- Comparten los mismos recursos
- No pueden escalar más allá de la capacidad de la computadora dónde se ejecutan



Procesos en computadoras diferentes



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

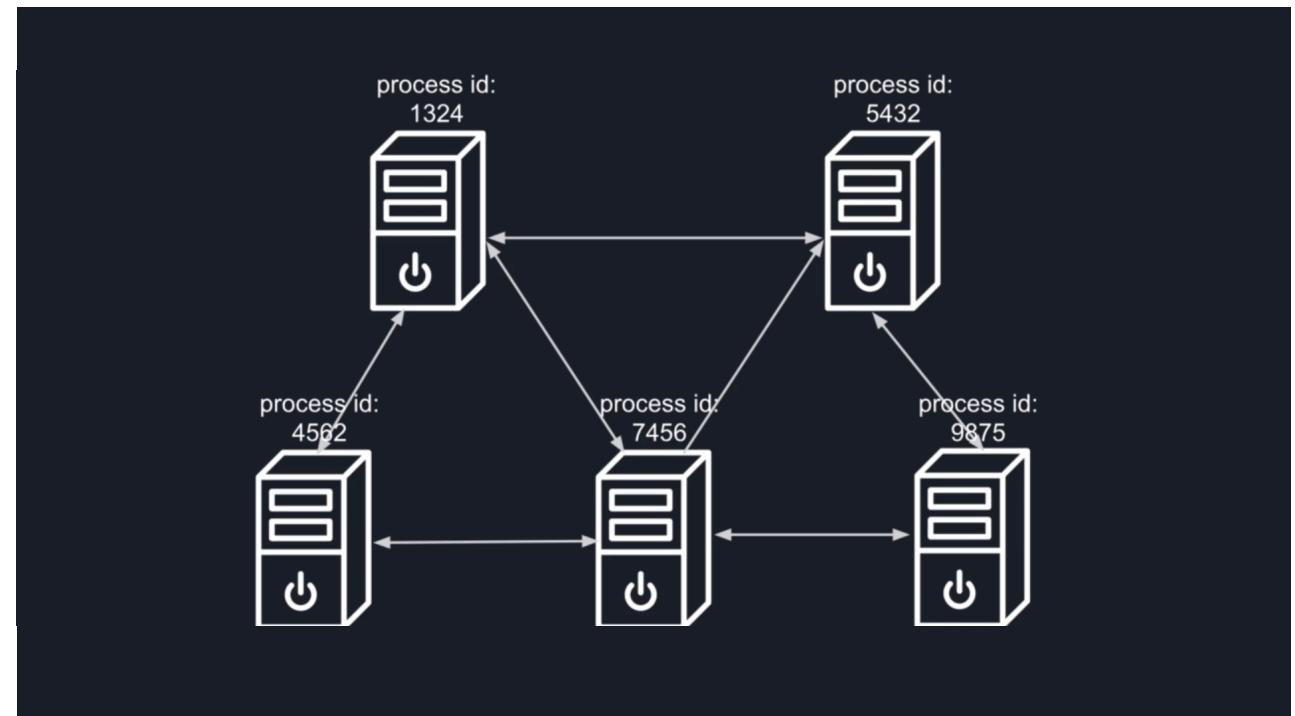
- Podemos poner los procesos en distintas computadoras para desacoplarlos en el uso de recursos



Comunicación por red



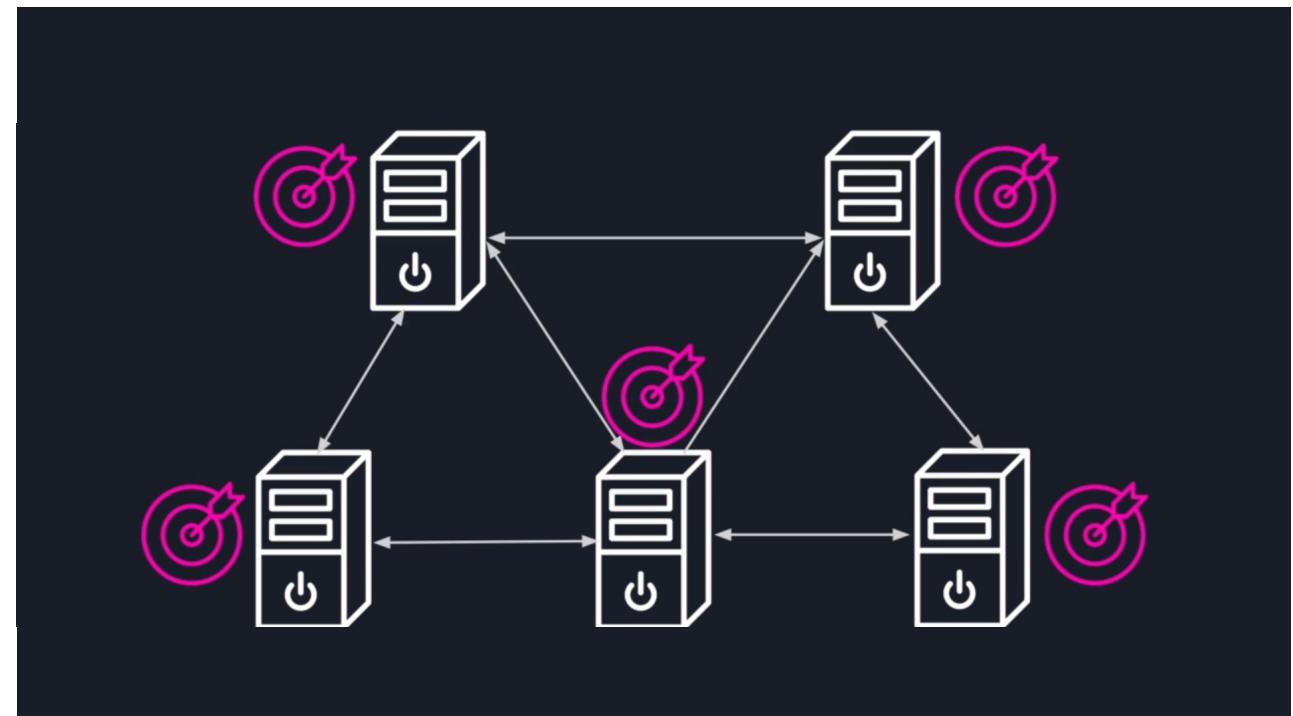
- La única opción de comunicación para los procesos corriendo en diferentes computadoras es utilizar la red
- La red no es confinable por definición
 - Puede no estar disponible
 - Tener una gran latencia
 - Estar congestionada



Tienen una meta común



- Mantienen una vista común del mundo
- Trabajan juntos para lograr un objetivo común
- Aparecen ante el usuario como una única entidad
- Si no tuvieran una meta común serían una colección de procesos separados

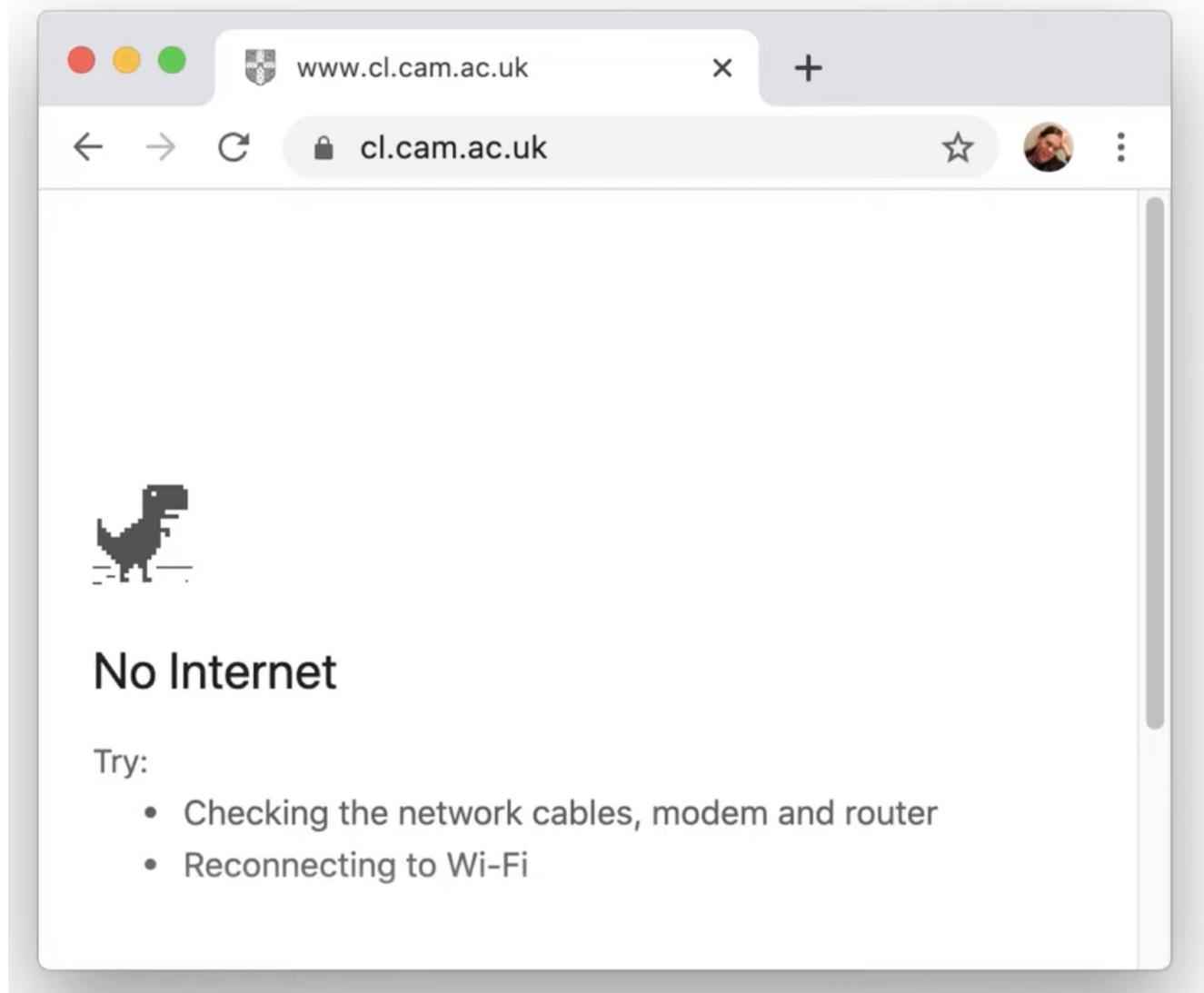


Por qué hacer un Sistema Distribuido?



- Sistemas inherentemente distribuidos
 - Telefonía
- Para mayor reliability
 - Tolerancia a fallos
- Para mayor performance
 - Location
 - Escalabilidad Horizontal
- Para poder resolver problemas mayores

Problemas con los sistemas distribuidos



Por qué no hacer un Sistema Distribuido?



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Problemas
 - La comunicación puede fallar
 - La red es inherentemente no confiable
 - Mayor probabilidad de fallos
 - Más procesos que pueden fallar
 - Más discos, etc
 - Son no-deterministas
 - Comportamiento emergente

Por qué no hacer un Sistema Distribuido?



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Es más difícil diseñar e implementar un sistema distribuido que funcione que uno centralizado
 - Tolerancia a fallos
 - Escalabilidad
 - Coordinación
 - Comunicación
 - Error handling
 - Etc.

Los sistemas distribuidos y la Red



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita



Complejidad

- Operadores de Red
 - Wifi
 - IS Provider
 - Satellite
 - Submarine cable
- Comunicación Física
 - Corriente eléctrica
 - Ondas de radio
 - Laser
 - Disco rígido

Ancho de bando y latencia

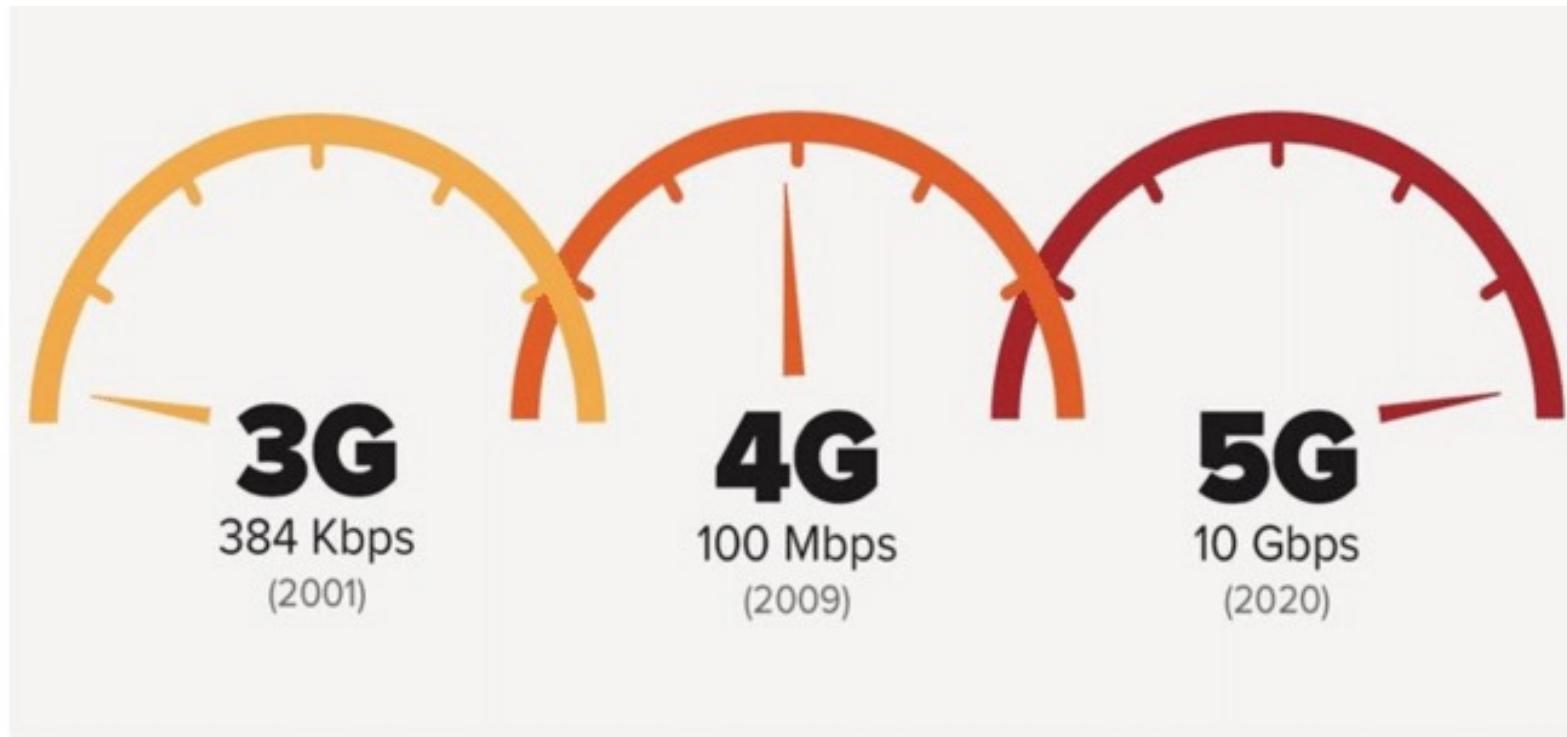


- Latencia:
 - En el mismo edificio 1 ms aprox
 - De un continente al otro 100 ms aprox
 - Hard drives 1 day aprox
- Ancho de Banda
 - 3G celular 384 kbps aprox
 - 4G celular 100 Mbps aprox
 - 5G celular 10 Gbps aprox
 - Discos rígidos en una Van 50TB/box - 5 Gbps aprox

Ancho de banda y latencia

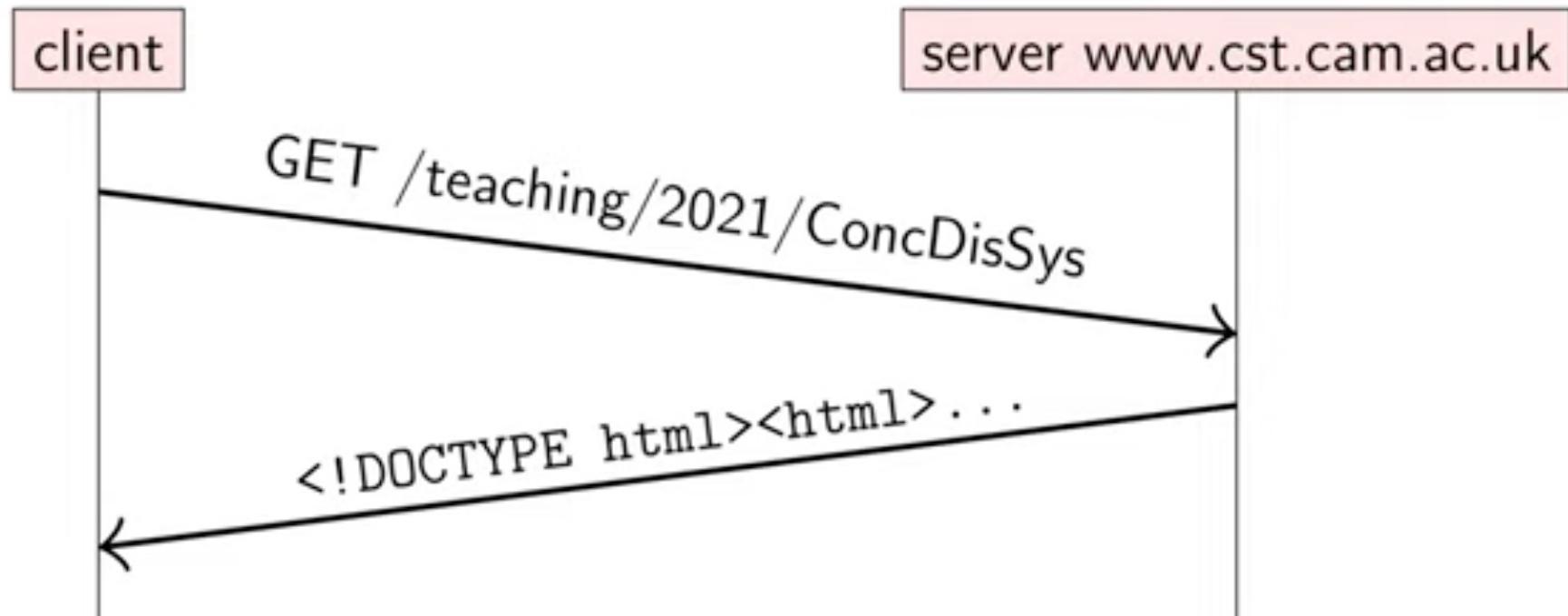


- Latencia:
 - En el mismo edificio 1 ms aprox



- DISCOS rígidos en una van 1 Gbit/s aprox

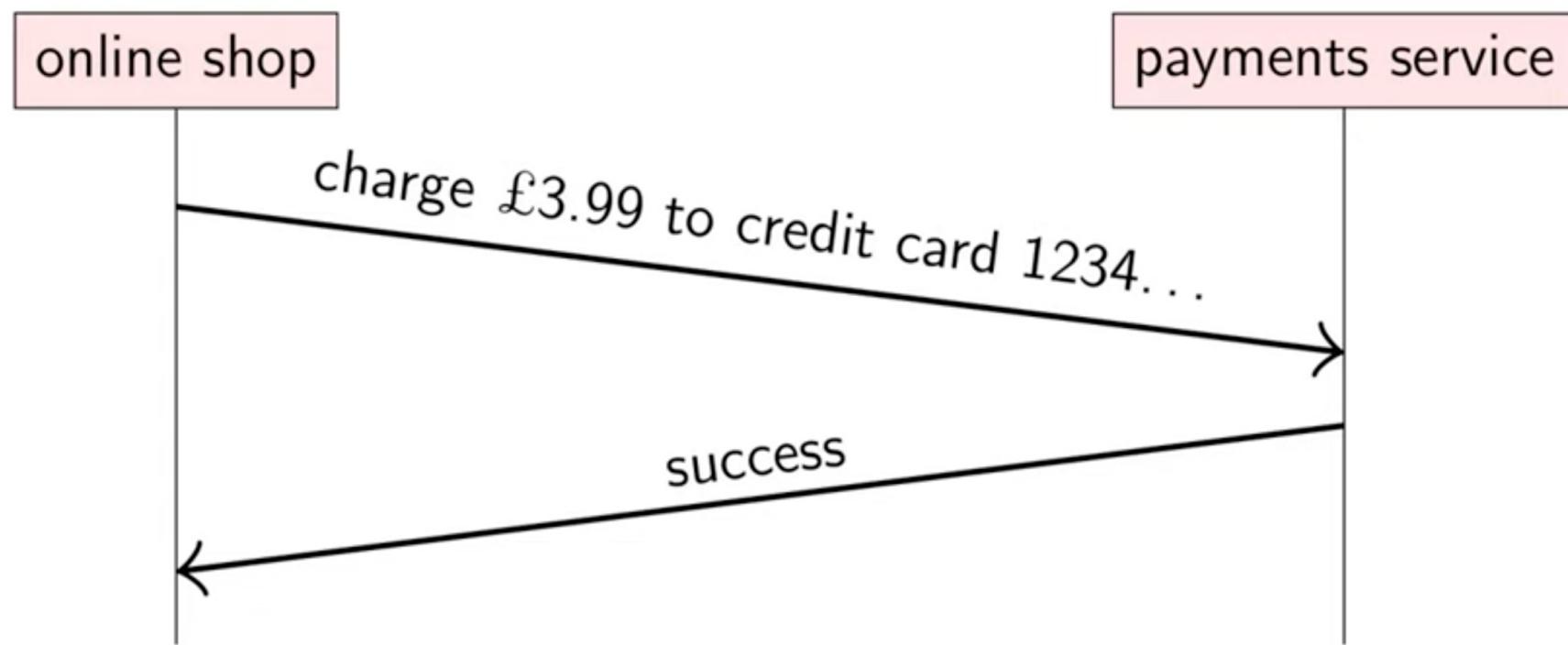
Sistema Distribuido: ejemplo



Sistema Distribuido: ejemplo



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita



Llamada a procedimiento remoto (RPC)



```
// Online shop handling customer's card details
Card card = new Card();
card.setCardNumber("1234 5678 8765 4321");
card.setExpiryDate("10/2024");
card.setCVC("123");

Result result = paymentsService.processPayment(card,
    3.99, Currency.GBP);

if (result.isSuccess()) {
    fulfilOrder();
}
```

Llamada a procedimiento remoto (RPC)



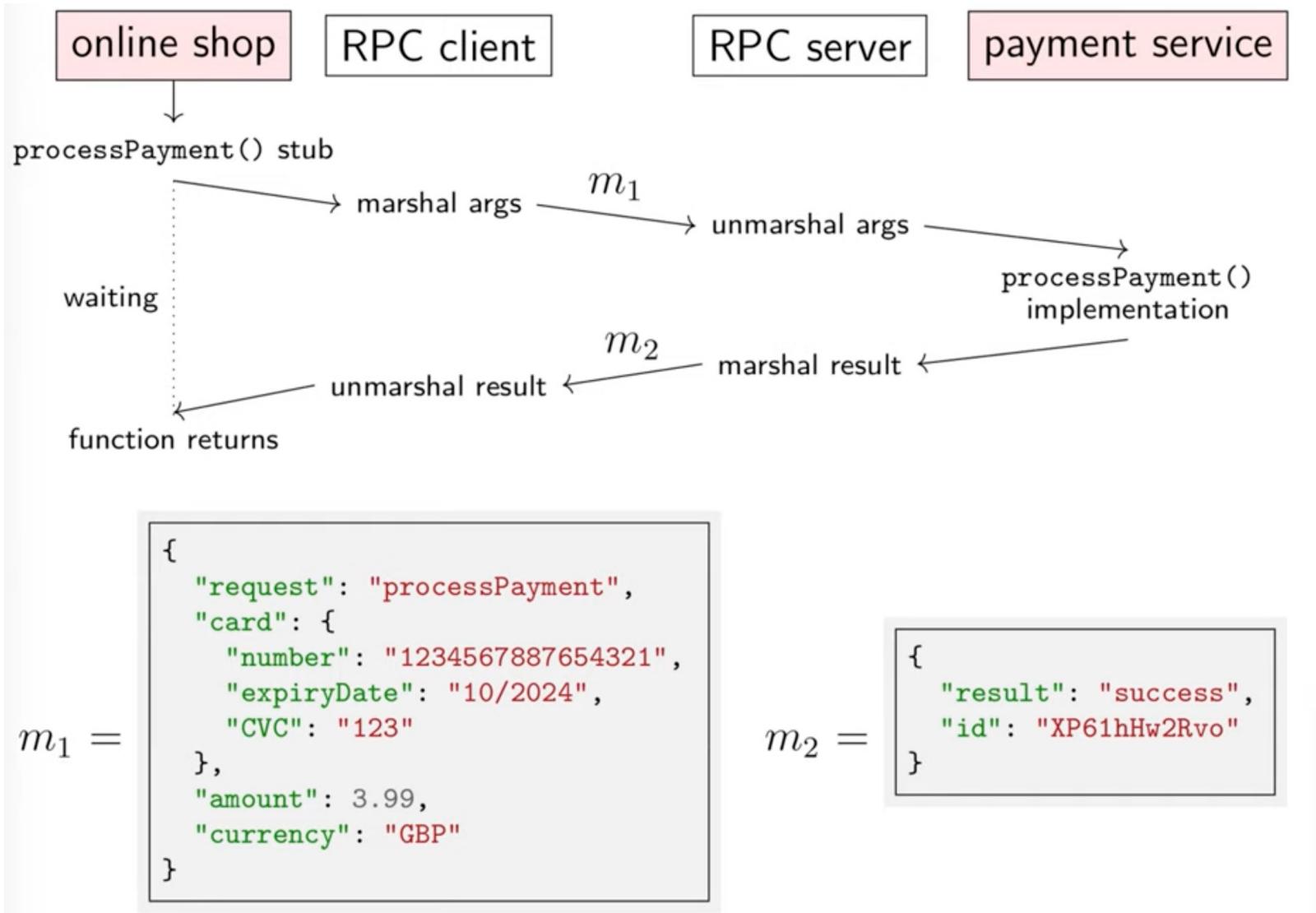
```
// Online shop handling customer's card details
Card card = new Card();
card.setCardNumber("1234 5678 8765 4321");
card.setExpiryDate("10/2024");
card.setCVC("123");

Result result = paymentsService.processPayment(card,
    3.99, Currency.GBP);

if (result.isSuccess()) {
    fulfilOrder();
}
```

La implementación está en otro nodo!

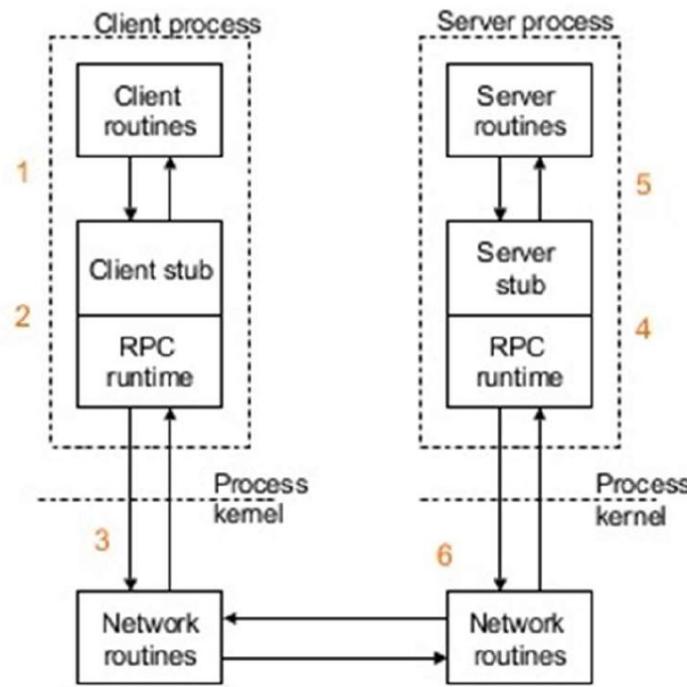
Llamada a procedimiento remoto (RPC)



Llamada a procedimiento remoto (RPC)



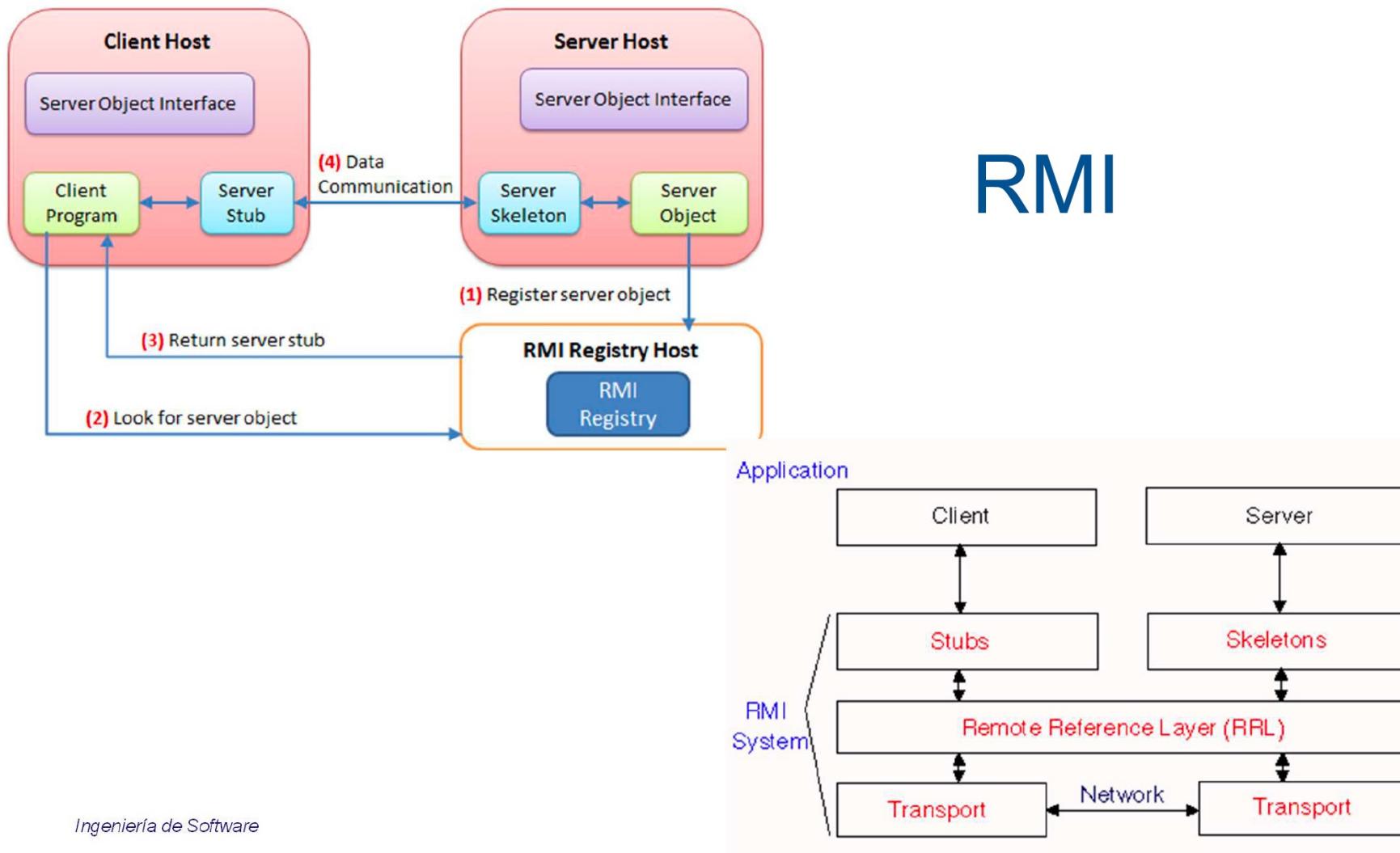
RPC: The basic mechanism



1. Client calls a local procedure on the client stub
2. The client stub acts as a proxy and **marshalls** the call and the args.
3. The client stub sends this to the remote system (via TCP/UDP)
4. The server stub **unmarshalls** the call and args from the client
5. The server stub calls the actual procedure on the server
6. The server stub **marshalls** the reply and sends it back to the client

Source: R. Stevens, *Unix Network Programming (IPC)*
Vol 2, 1998

Llamada a procedimiento remoto (RPC)



Llamada a procedimiento remoto (RPC)



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Idealmente RPC hace que una llamada a una función remota sea igual a una llamada local

“Location transparency”

- En la práctica
 - El servicio remote falla
 - Los mensajes se pierden
 - Los mensajes se demoran
 - Puedo reintentar? Hasta cuando?

RPC: Historia



- ▶ SunRPC/ONC RPC (1980s, basis for NFS)
- ▶ CORBA: object-oriented middleware, hot in the 1990s
- ▶ Microsoft's DCOM and Java RMI (similar to CORBA)
- ▶ SOAP/XML-RPC: RPC using XML and HTTP (1998)
- ▶ Thrift (Facebook, 2007)
- ▶ gRPC (Google, 2015)
- ▶ REST (often with JSON)
- ▶ Ajax in web browsers

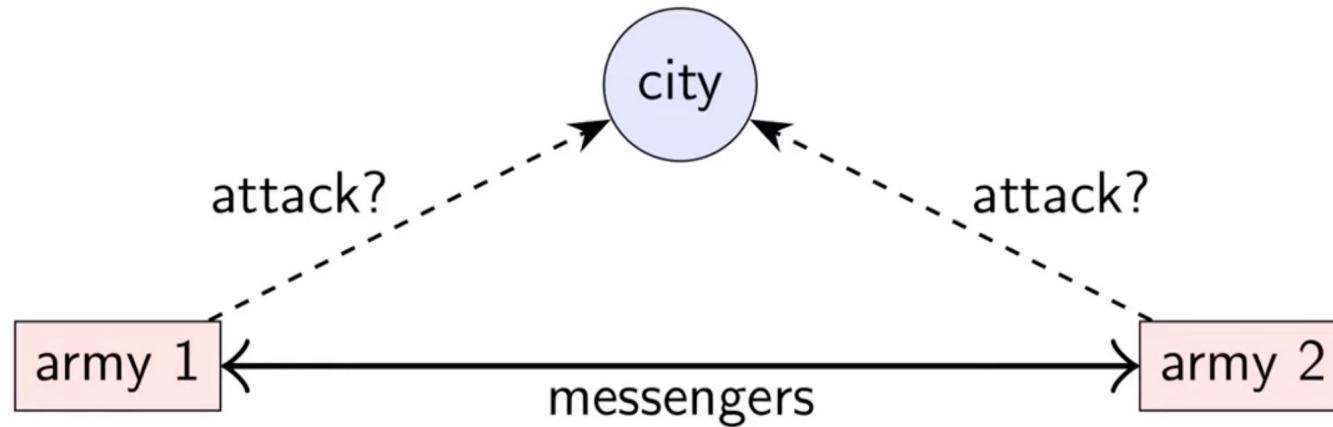
Modelos de Sistemas distribuidos



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Descripciones de las suposiciones que hacemos al diseñar sistemas distribuidos
- Importantes para entender que puede fallar
 - Nodos
 - La red
- Necesitamos ser precisos sobre qué estamos trabajando

El problema de los dos generales



army 1	army 2	outcome
does not attack	does not attack	nothing happens
attacks	does not attack	army 1 defeated
does not attack	attacks	army 2 defeated
attacks	attacks	city captured

Desired: army 1 attacks *if and only if* army 2 attacks

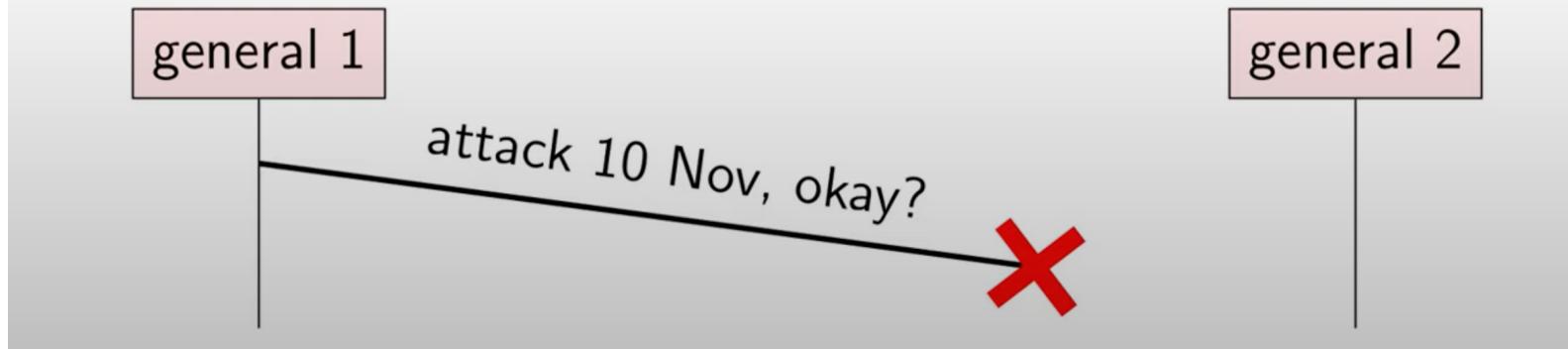
El problema de los dos generales



El problema de los dos generales



From general 1's point of view, this is indistinguishable from:



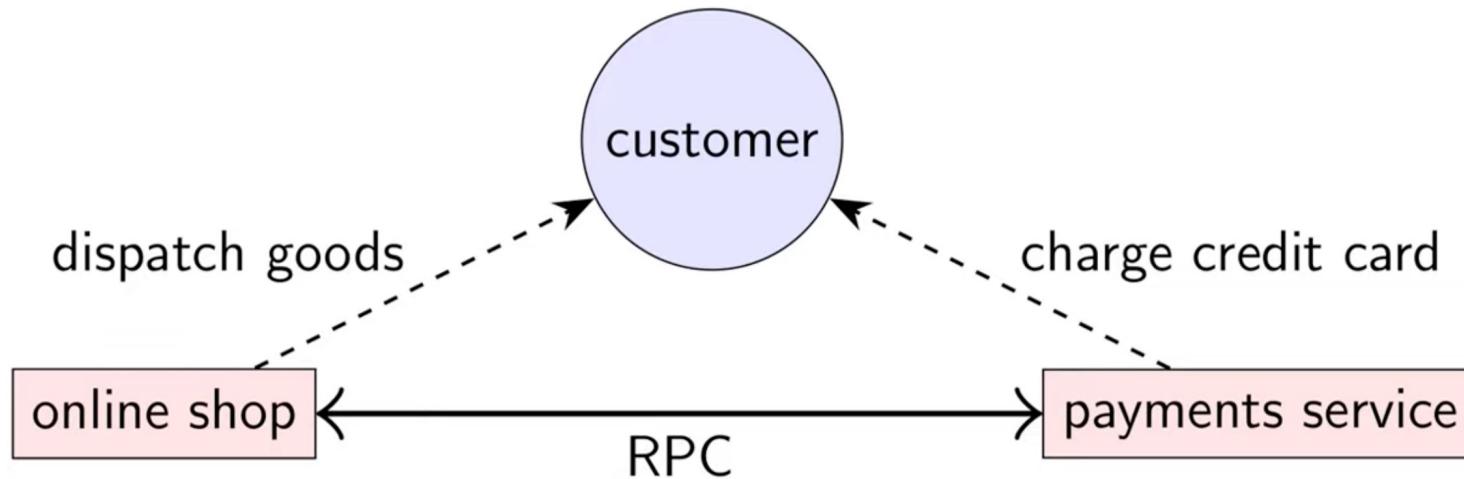
El problema de los dos generales



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Qué debería hacer el general 1?
 - General 1 siempre ataca
 - Enviar varios mensajeros?
 - Y si capturan todos ellos? General 1 en riesgo
 - General 1 solo ataca si recibe respuesta del general 2
 - General 1 está seguro
 - General 2 está en riesgo (igual que el general 1 antes)
- No hay conocimiento común de las cosas

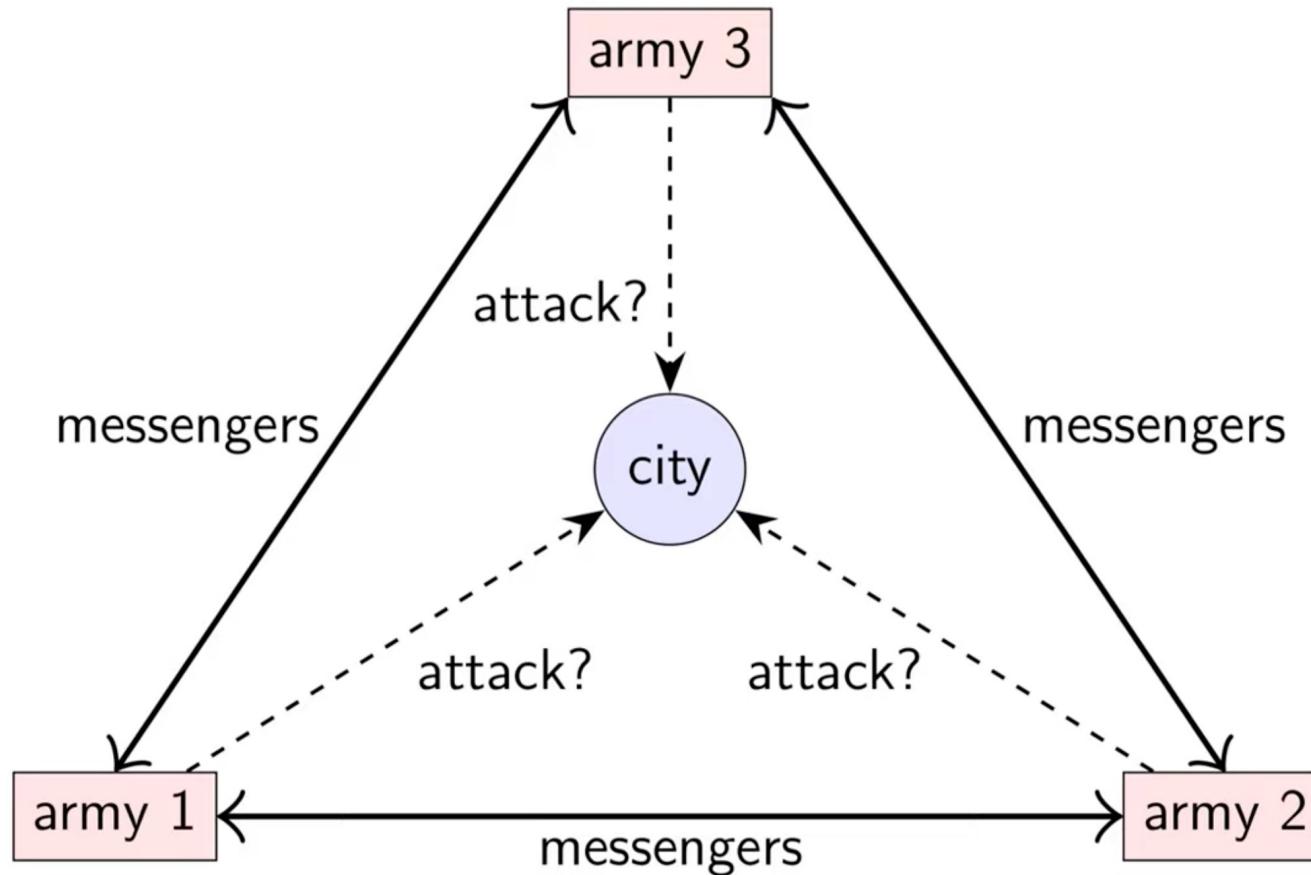
El problema de los dos generales



online shop	payments service	outcome
does not dispatch	does not charge	nothing happens
dispatches	does not charge	shop loses money
does not dispatch	charges	customer complaint
dispatches	charges	everyone happy

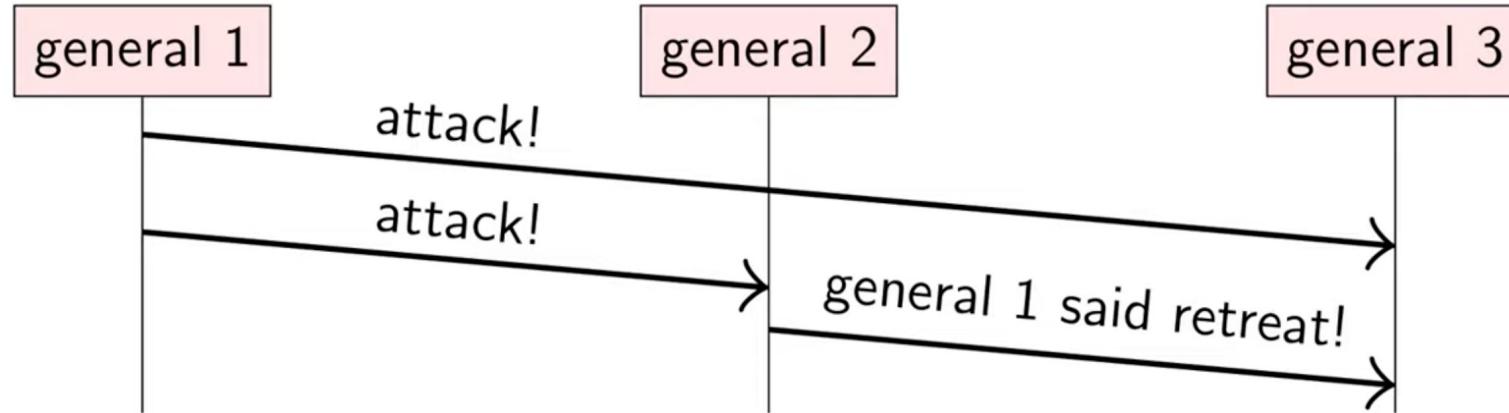
Desired: online shop dispatches *if and only if* payment made

Los generales bizantinos

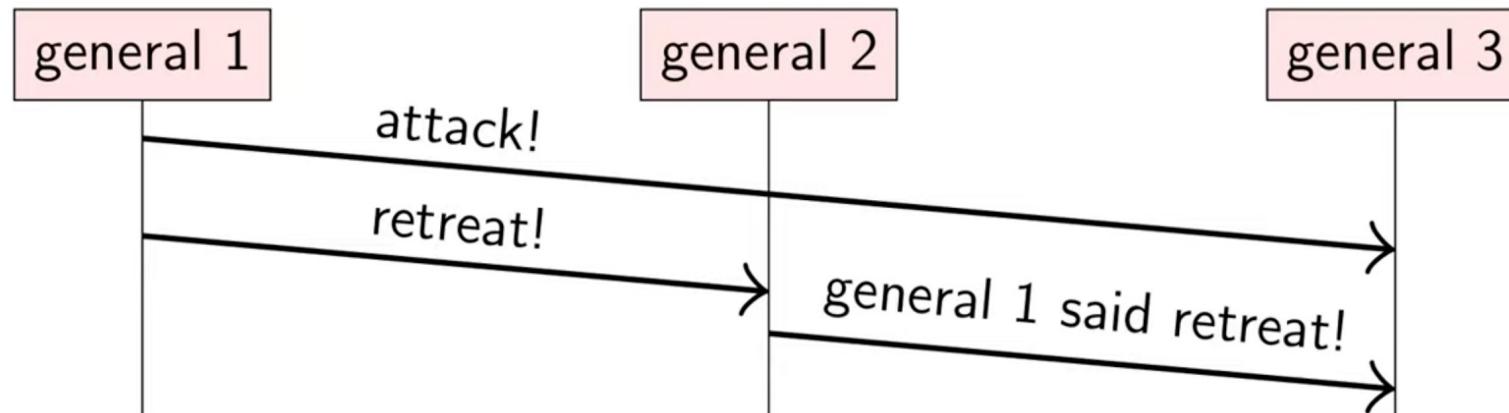


Problem: some of the generals might be traitors

Los generales bizantinos



From general 3's point of view, this is indistinguishable from:



Los generales bizantinos



- Hasta f generales maliciosos
- Los generales honestos no saben quienes son los generales maliciosos
- Los generales maliciosos pueden actuar juntos
- A pesar de estas condiciones los generales honestos deben acordar en un plan
- Teorema:
 - Se necesitan $3f+1$ en total para tolerar f generales maliciosos.
- Criptografía puede ayudar

Modelos de Sistemas distribuidos



- Vimos dos experimentos mentales
 - El problema de los 2 generales
 - Modelo de las redes
 - El problema de los generales bizantinos
 - Modelos del comportamiento de los nodos
- En la realidad ambos, la red y los nodos pueden ser defectuosos
- Suposiciones en la que se basan los algoritmos distribuidos
 - Comportamiento de la Red (ej: perdida de mensajes)
 - Comportamiento de los nodos (ej: caída de nodos)
 - Timing (ej: latencia)

Comportamiento de la Red



- Suponemos comunicación point-to-point entre 2 nodos con uno de los siguientes modos:
 - Reliable
 - Un mensaje llega solo y solo es enviado
 - Fair-loss
 - Los mensajes se pueden perder.
 - Si se reintenta lo suficiente los mensajes llegan eventualmente.
 - Arbitrary
 - Puede haber un adversario maligno que puede interferir con los mensajes de manera arbitraria.
- Network partition
 - La comunicación se interrumpe o se demora arbitrariamente



Comportamiento de los Nodos



- Cada nodo ejecuta un algoritmo asumiendo uno de los siguientes:
 - Crash-stop
 - Falla para siempre
 - Crash-recovery
 - Falla momentáneamente, pierde el estado en memoria, puede continuar su ejecución en algún momento después de la falla.
 - Byzantine
 - Comportamiento malicioso incluso fallar.
- Un nodo que no es defectuoso es llamado “Sano”

Suposiciones de “Timing”



- Suponemos uno de los siguientes para la red y los nodos:
 - Síncrono
 - La latencia de los mensajes no es mayor que un límite superior conocido.
 - Los nodos ejecutan algoritmos a una velocidad conocida.
 - Parcialmente síncrono
 - El Sistema es asíncrono por momentos (finitos pero desconocidos) y síncrono el resto del tiempo
 - Asíncrono
 - Los mensajes pueden demorarse arbitrariamente
 - Los nodos pueden pausarse arbitrariamente
 - No existen garantías de ningún tipo

Problemas de sincronismo en la práctica



- Las redes tiene una latencia previsible, pero no siempre
 - La perdida de mensajes pueden requerir los retries
 - El congestionamiento puede causar que los mensajes se demoren
 - Puede haber cambios en las rutas de la red
- Los nodos tiene velocidad predecible, pero no siempre
 - Scheduling a nivel SO
 - Garbage collector
 - Page faults, swaps

NOTA: RTOS pueden dar garantías de scheduling pero la mayoría de los DS no corren en RTOS

Modelos



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Las suposiciones vistas son la base para todos los algoritmos distribuidos.
- Si las suposiciones no son las correctas los algoritmos serán defectuosos

Disponibilidad



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Disponibilidad/Availability
 - Fracción del tiempo que el sistema está funcionando correctamente (uptime)
 - “two nines” = 99% up = down 3.7 días/año
 - “three nines” = 99.9% up = down 8.8 horas/año
- SLO: Service-Level Objective
- SLA: Service-Level Agreement

Reliability: ejercicio



- Cual es la Confiabilidad/Reliability y la probabilidad de fallo de un sistema con tres components con
 - $R_1 = 0.9$
 - $R_2 = 0.8$
 - $R_3 = 0.5$
- $F = 1 - R$

Tolerancia a Fallos



- Falla/Failure
 - El sistema como un todo no funciona
- Fallo/Fault
 - Fallo en la red
 - Fallo en un nodo
- Tolerancia a fallos
 - El sistema continua funcionando a pesar de la existencia de fallos
 - Se supone un numero máximo de fallos concurrentes
- Diseñar el sistema sin Single Point of Failure:
 - Nodo/enlace cuyo fallo lleva a una falla total del sistema

Tolerancia a Fallos



- Detectores de fallos
 - Algoritmo que determina/detecta si un nodo falla
- Detector de fallos perfecto
 - Determina que un nodo es defectuoso solo y solo el nodo falla
- Problema
 - No puedo diferenciar entre: nodo defectuoso, temporalmente no disponible, mensajes perdidos, mensajes demorados
- Detectores de fallos perfectos
 - Solo posibles si suponemos sistemas **síncronos** y nodos **crash-stop** con red **confiable**

Tolerancia a Fallos



- Detectores de fallos eventualmente perfectos
 - Puede decidir **temporalmente** que un nodo es defectuoso incluso si no lo es
 - Puede decidir **temporalmente** que un nodo está sano incluso si no lo está
 - Pero **eventualmente** determina que un nodo es defectuoso si y solo si lo es.
- La detección de fallos no es instantánea
- Implementación más común
 - Health checks / liveness probes
 - Envío mensaje, espero la respuesta un tiempo determinado, determino que el nodo falla si no recibo respuesta dentro de ese tiempo.
 - Puede requerir retries

Replicación

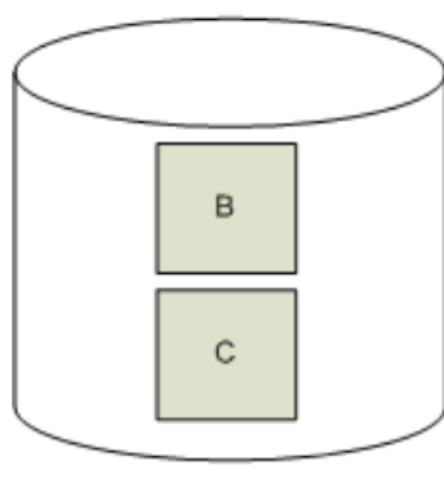


- Copiar los mismos datos en multiples nodos
 - Base de datos, file systems, caches, etc
- Un nodo con una copia de datos se llama Replica
- Motivación
 - Nos permite hacer el sistema Tolerante a Fallos
 - Si algunos nodos fallan, los otros pueden continuar operando
 - Distribuir la carga entre varios nodos
- Si los datos no cambian es fácil
 - Pero no es el caso más interesante

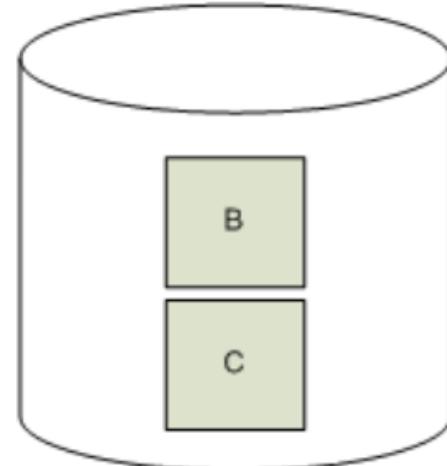
Replicación & Escalabilidad & High Availability



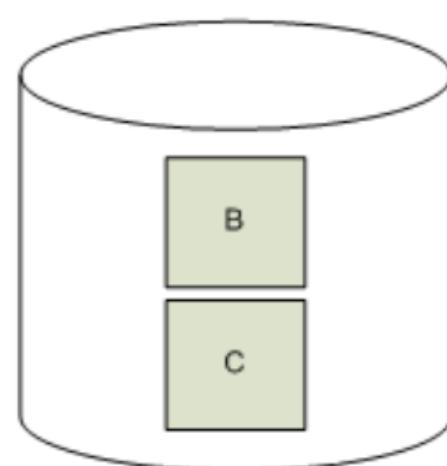
- Replico los datos para tener mayor confiabilidad/tolerancia a fallos



Node 1



Node 2

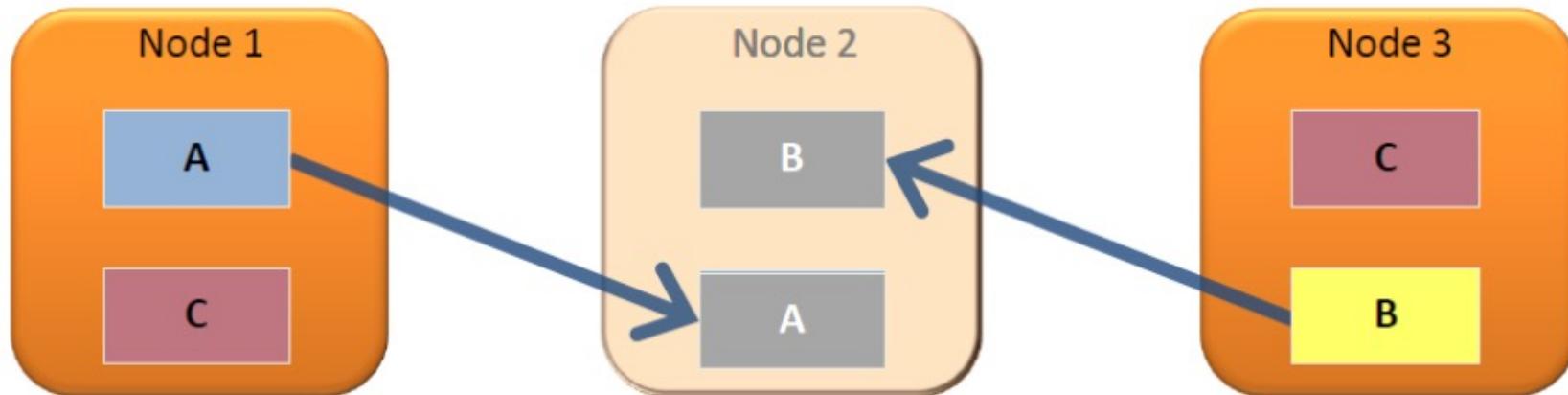


Node 3

Replicación & Escalabilidad & High Availability



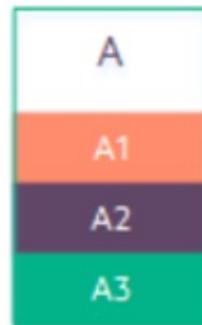
- Replico los datos para tener mayor confiabilidad/tolerancia a fallos
- Segmento los datos para escalar



Replicación & High Availability



Projection A with segments 1, 2, and 3



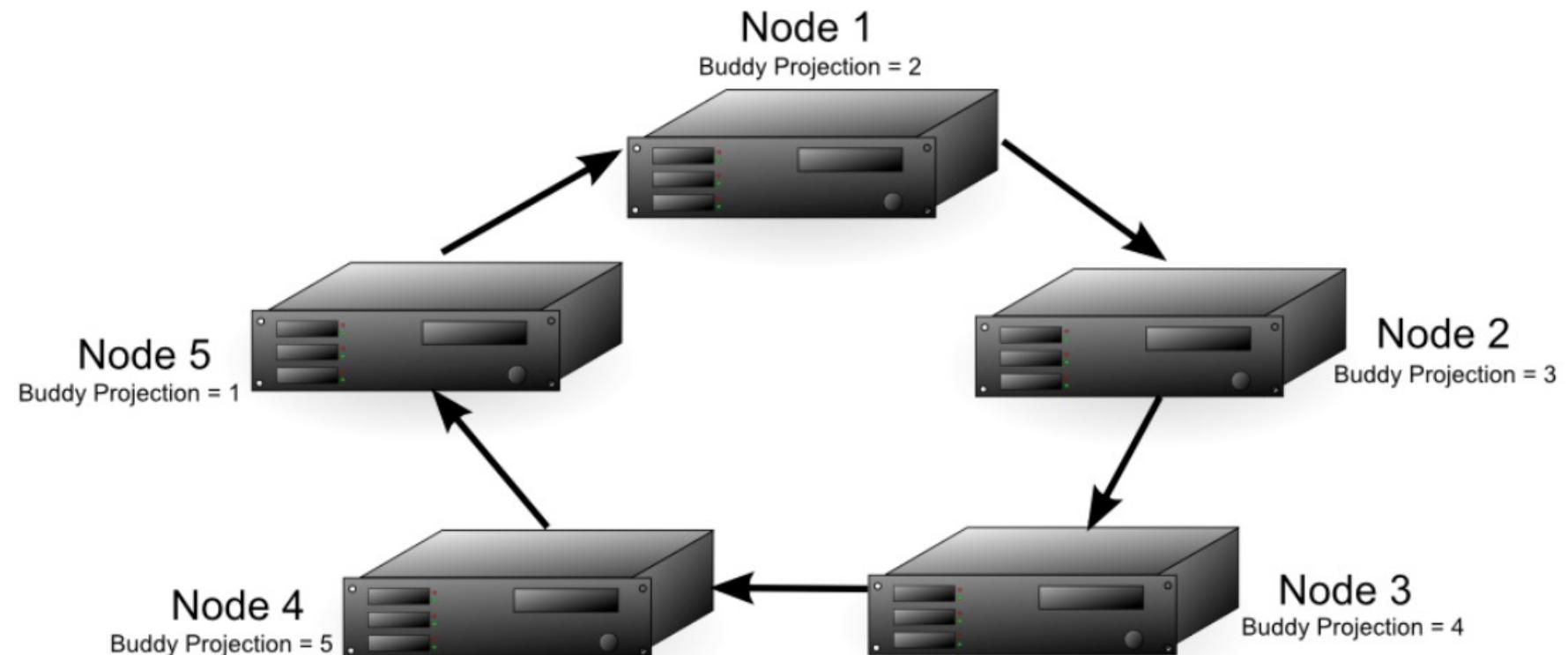
Projection A's buddy projection (A_BP) with segments A1_BP, A2_BP, and A3_BP



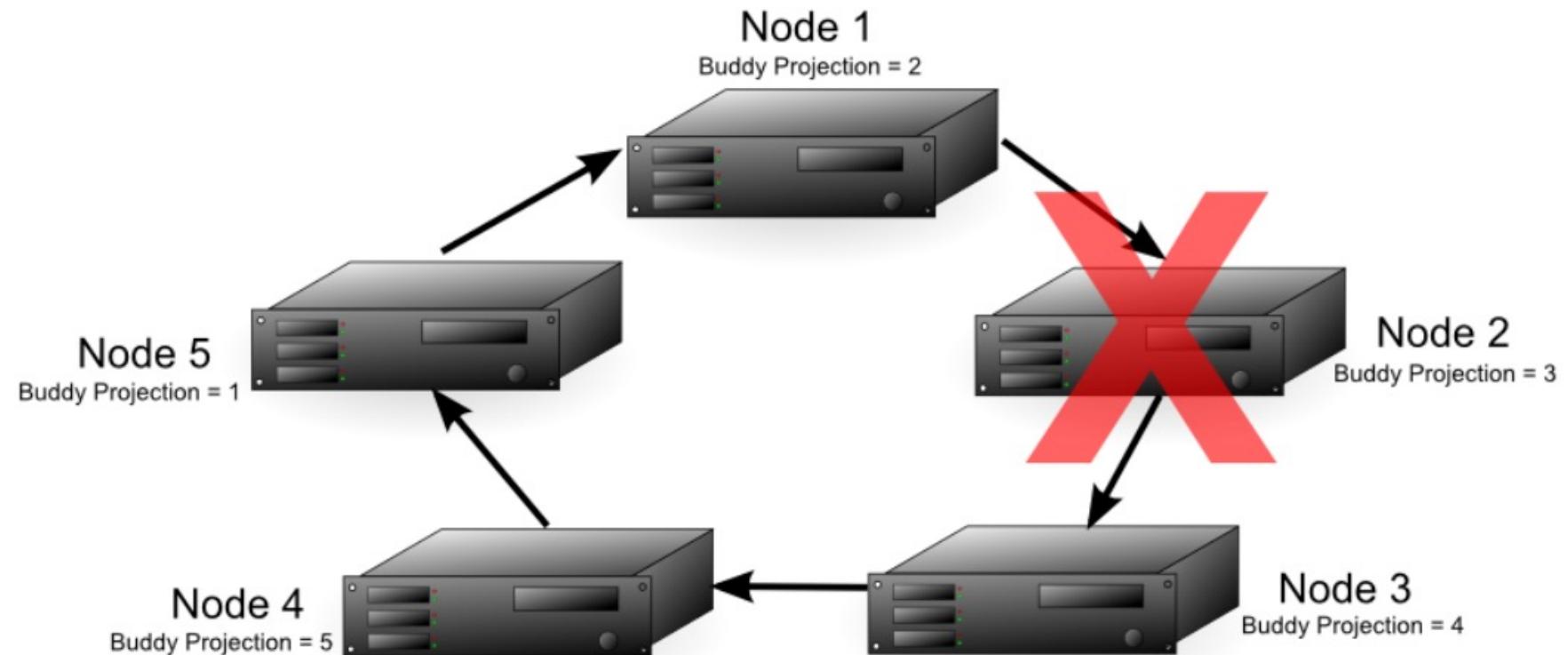
Segments distributed across 3 nodes



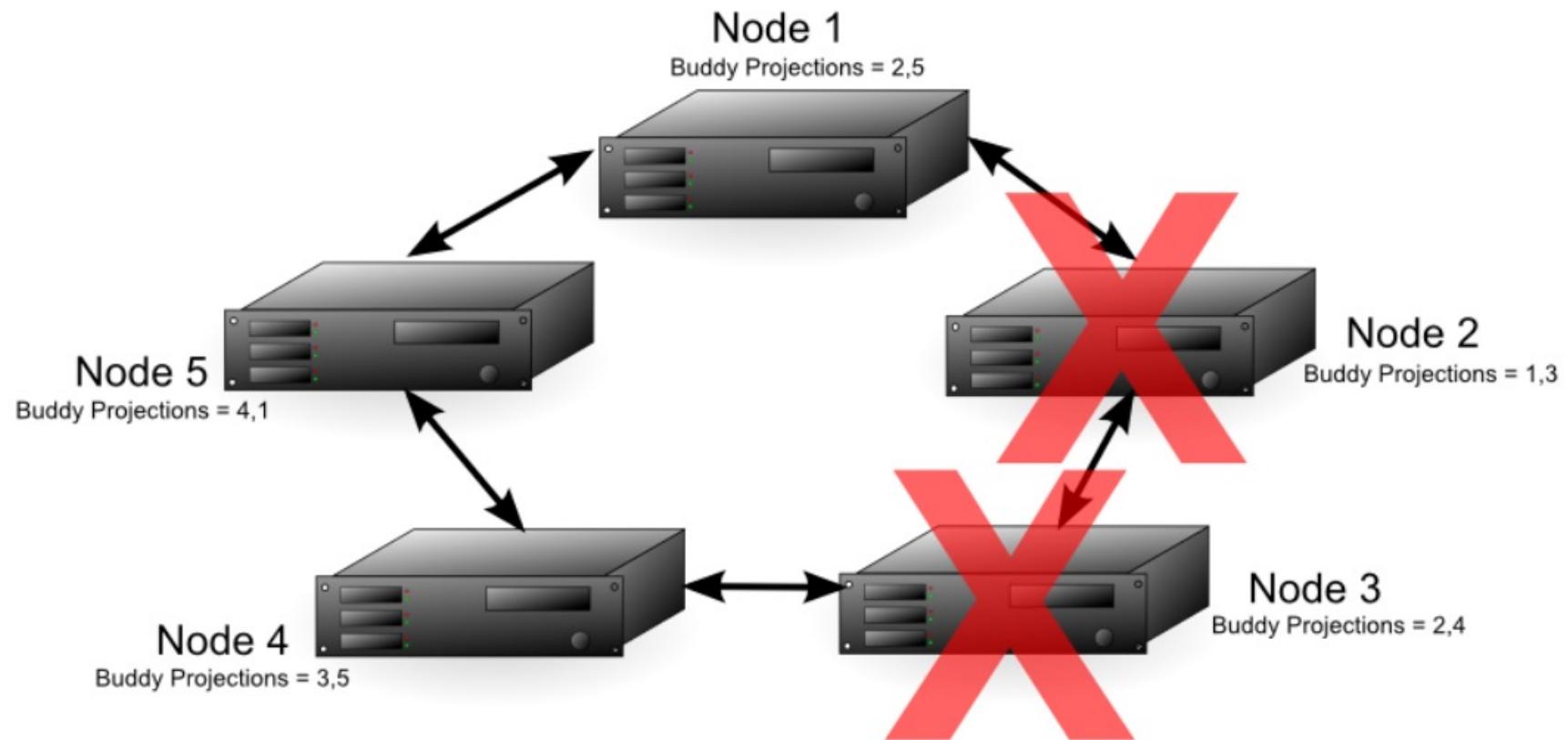
Replicación & High Availability



Replicación & High Availability



Replicación & High Availability



Retries



- Idempotencia
 - Una función es idempotente si $f(x) = f(f(x))$
 - $Y = Y + 1$ (NO)
 - $Y = Y \cup \{ \text{id} \}$ (YES)
- Opciones de retry
 - At-most-once: envío una sola vez
 - At-least-once: envío hasta que recibo acknowledge
 - Exactly-once: at-least-once + idempotencia (o deduplicación)

El teorema CAP



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Un sistema distribuido no puede ser Consistente, disponible (Available) y tolerante a las Particiones de la red.

El teorema CAP



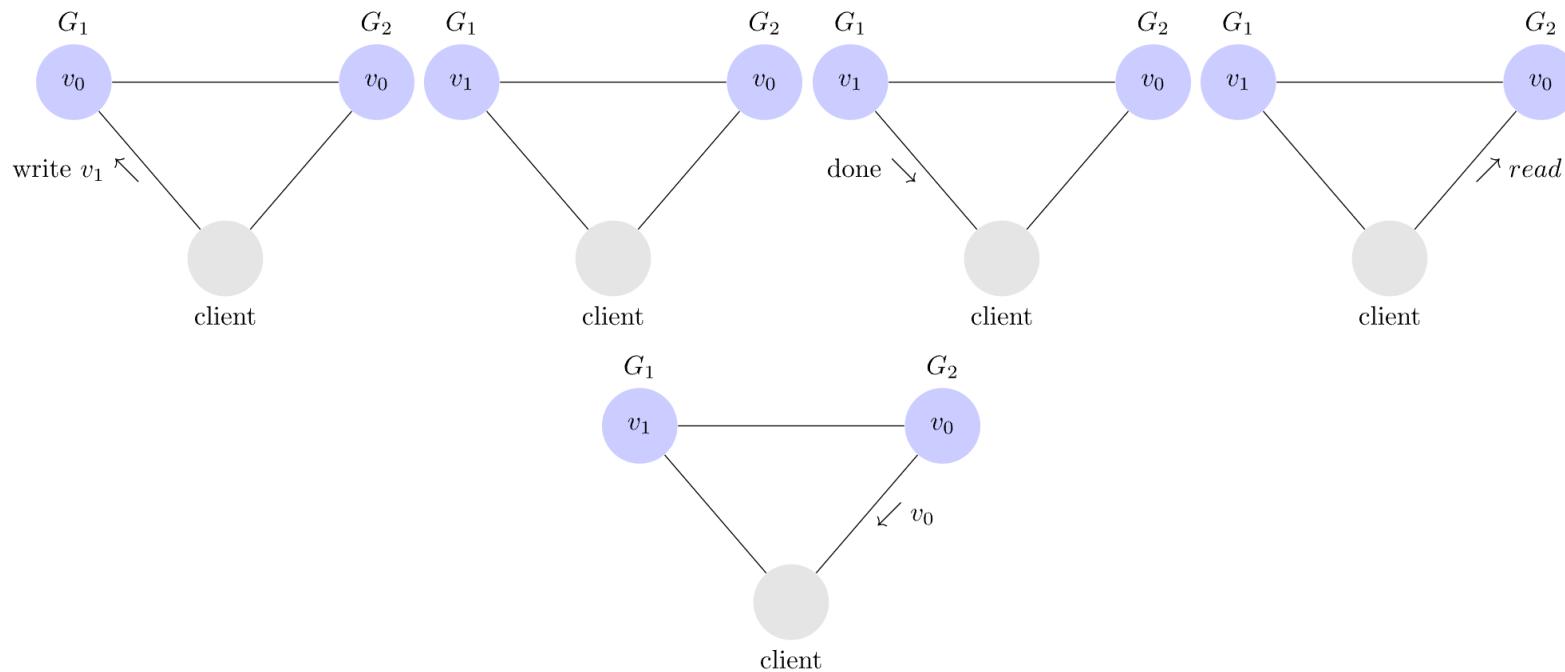
UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Consistencia
 - Un sistema es consistente si cualquier READ que sucede después de un WRITE debe devolver ese valor o el resultado de un WRITE posterior

El teorema CAP



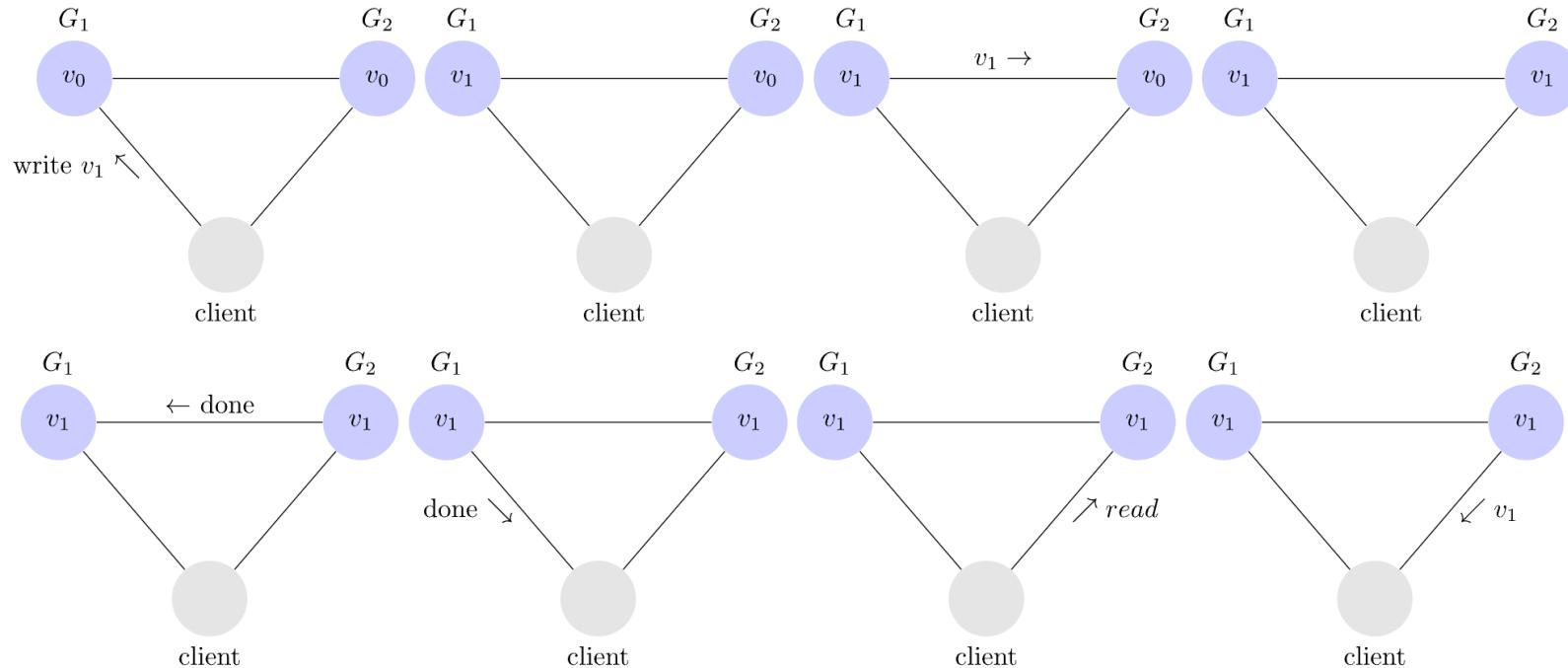
- Incosistente



El teorema CAP



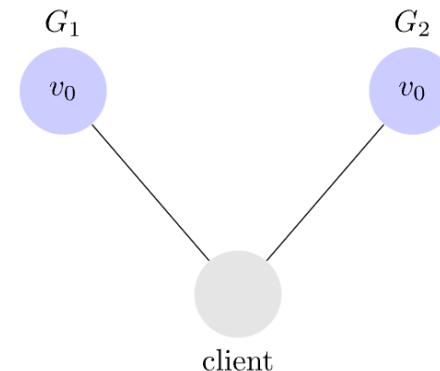
- Consistente



El teorema CAP



- Disponibilidad
 - Toda petición a un nodo que no está fallando debe obtener respuesta
- Tolerancia a las particiones de la red
 - La red puede perder cualquier cantidad de mensajes de manera arbitraria

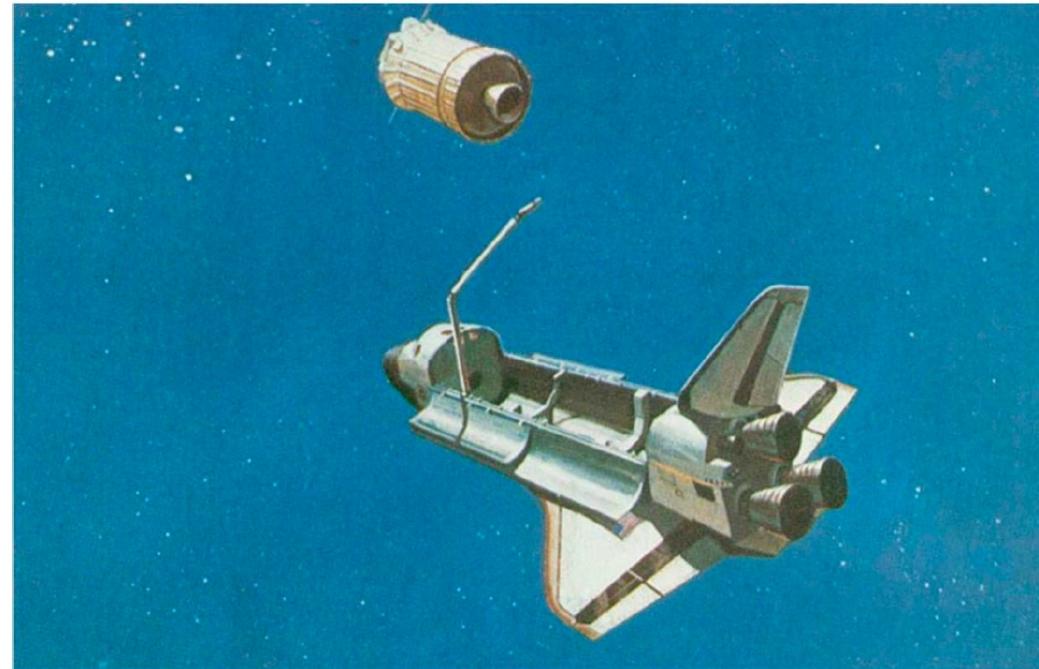


Modelos



UNIVERSIDAD
CATÓLICA DE CÓRDOBA
Universidad Jesuita

- Ejemplo de sistema distribuido para mejorar la confiabilidad
 - PASS IBM 1981
 - Sistema de votación entre 4 computadoras



Ejemplo de Sistema Distribuido

