
IIC2523

Sistemas Distribuidos

— Hernán F. Valdivieso López —
(2025 - 2 / Clase 07)

Recordatorios importantes

1. Responder la ECA, necesito ver cuanto tiempo está consumiendo esta tarea por favor.
2. La tarea, si bien se autoriza uso de IA de forma responsable y hacerlo en pareja, recuerden:
 - a. Inscribir su pareja hasta hoy a las 20:00
 - b. Citar todo uso de IA
 - c. Asegurar que el desarrollo de su tarea se mantenga en máximo **2 personas**. No comentar o permitir que su tarea se divulgue a más personas (cuidado con las salas de estudios con tanta gente).

Consenso en Sistemas Distribuidos

Temas de la clase

1. ¿Qué son los algoritmos de consenso?
2. Paxos
3. Problema de los generales bizantinos

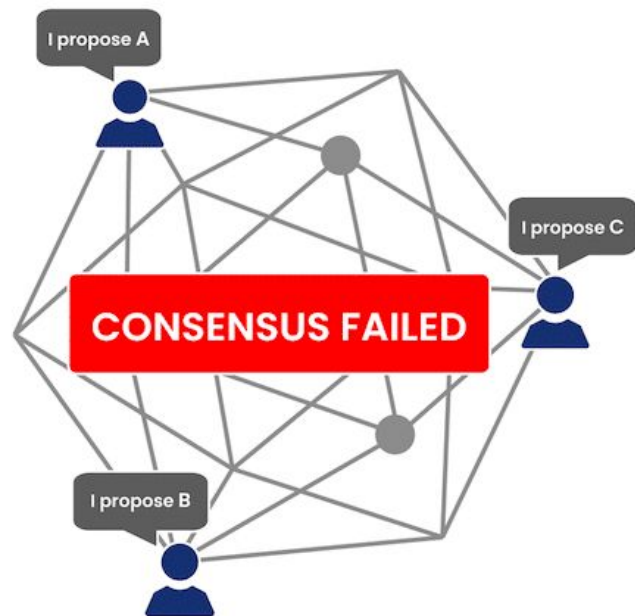
¿Qué son los algoritmos de consenso?

¿Por qué los necesitamos?

¿Qué son los algoritmos de consenso?

Algoritmos de consenso - ¿Por qué los necesitamos?

- ◆ Múltiples nodos deben acordar decisiones comunes.
 - ◆ Un valor.
 - ◆ Una modificación.
 - ◆ Eliminar algún dato.
- ◆ Fallos de red, nodos caídos o mensajes duplicados o contradictorios dificultan este proceso.
- ◆ Necesitamos algoritmos que aseguren consistencia, incluso ante fallos.



Algoritmos de consenso - ¿Que son?

Protocolo que permite a varios nodos acordar un valor común, aún en presencia de fallos.

- ◆ Los protocolos sirven para llegar a un consenso sobre cualquier decisión y no solo para un valor.
- ◆ Por simplicidad y generalidad, diremos que se busca acordar una **"operación"** o **"acción"** en vez de valor.

Protocolo que permite a varios nodos acordar una operación o acción en común, aún en presencia de fallos.

Algoritmos de consenso - ¿Que son?

Protocolo que permite a varios nodos acordar una operación o acción en común, aún en presencia de fallos.

Propiedades deseables:

- ◆ **Acuerdo (*Agreement*)**: todos los nodos activos aceptan la misma operación.
- ◆ **Terminación (*Termination*)**: todos los nodos activos eventualmente aceptan una operación.
- ◆ **Tolerancia a fallos (*fault tolerance*)**: funciona aun si algunos nodos no responden.

Algoritmos de consenso - ¿Que son?

Existen muchos algoritmos de consenso hoy en día, por ejemplo:

- ◆ Paxos, uno de los primeros creados por Leslie Lamport.
- ◆ Raft, una variación más sencilla y democrática de Paxos.
- ◆ *Proof of Work* utilizado en *Bitcoin* (también llamado consenso Nakamoto).
- ◆ *Proof of Stake*, considera una opción más eficiente energéticamente y potencialmente más escalable que *Proof of Work*. Utilizado por *Ethereum*.
- ◆ Muchos más... principalmente en Criptomoneda han surgido varias variaciones.

Algoritmo de consenso

Paxos

Paxos - Introducción

- ◆ Uno de los primeros algoritmos formales para consenso propuesto por Leslie Lamport.
- ◆ Paper original: [The Part-Time Parliament \(1990\)](#)

Lamport intentó hacerlo entretenido usando una historia ficticia de una isla griega llamada "Paxos", con un parlamento antiguo cuyos miembros eran versiones griegas de científicos reales.

Su intento de añadir humor fracasó. Nadie entendió el algoritmo. Los revisores dijeron que era interesante pero que debía eliminar la parte de Paxos.
- ◆ Versión posterior explicado para humanos: [Paxos Made Simple \(2001\)](#)

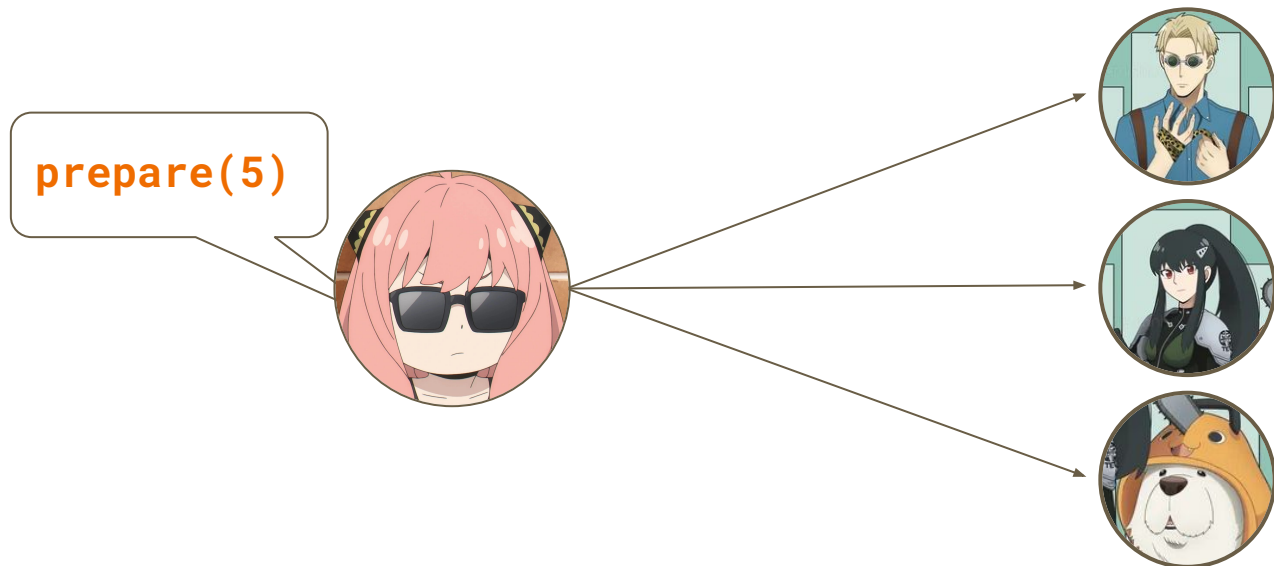
Paxos - Roles y fases

- ◆ Se proponen 3 roles:
 - ◆ **Proponente (*Proposer*)**: propone una acción a consolidar.
 - ◆ **Aceptante (*Acceptor*)**: acepta la acción y la guarda localmente.
 - ◆ **Aprendiz (*Learner*)**: commitea/consolida la acción de forma definitiva.
- ◆ En la práctica, un nodo puede cumplir más de un rol, siempre y cuando respete las reglas de cada rol.
- ◆ Hay 3 fases:
 - ◆ **Preparar propuesta (*preparar*)**.
 - ◆ **Aceptar propuesta (*accept*)**.
 - ◆ **Aprender propuesta (*learn*)**.

Paxos - Fase 1 - Preparar propuesta

1. El proponente envía un **prepare(n)** a cada aceptante.

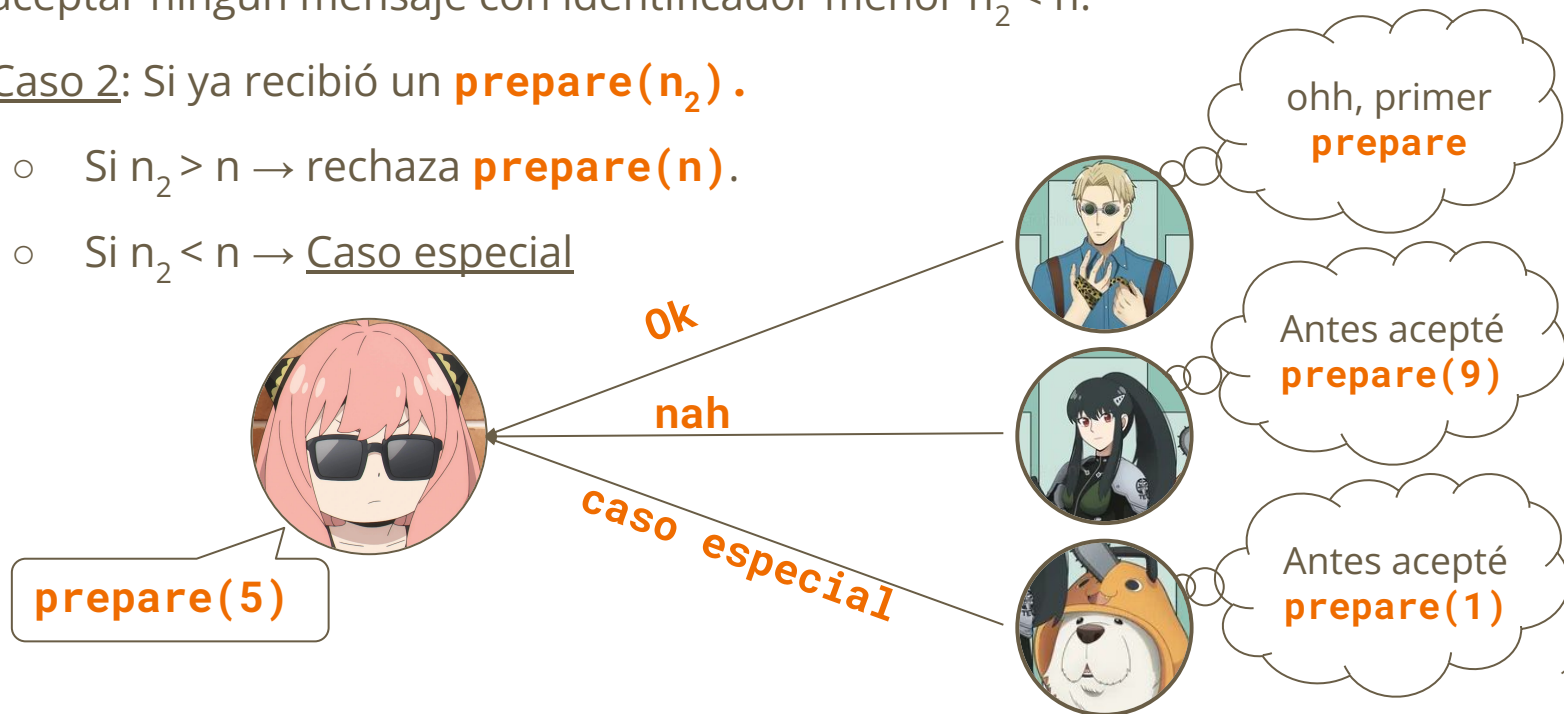
- ◆ **n** es un identificador único de la propuesta.
- ◆ En las implementaciones se puede ver como (timestamp, id_nodo)



Paxos - Fase 1 - Preparar propuesta

2. Los aceptantes **activos** responden al **prepare(n)**

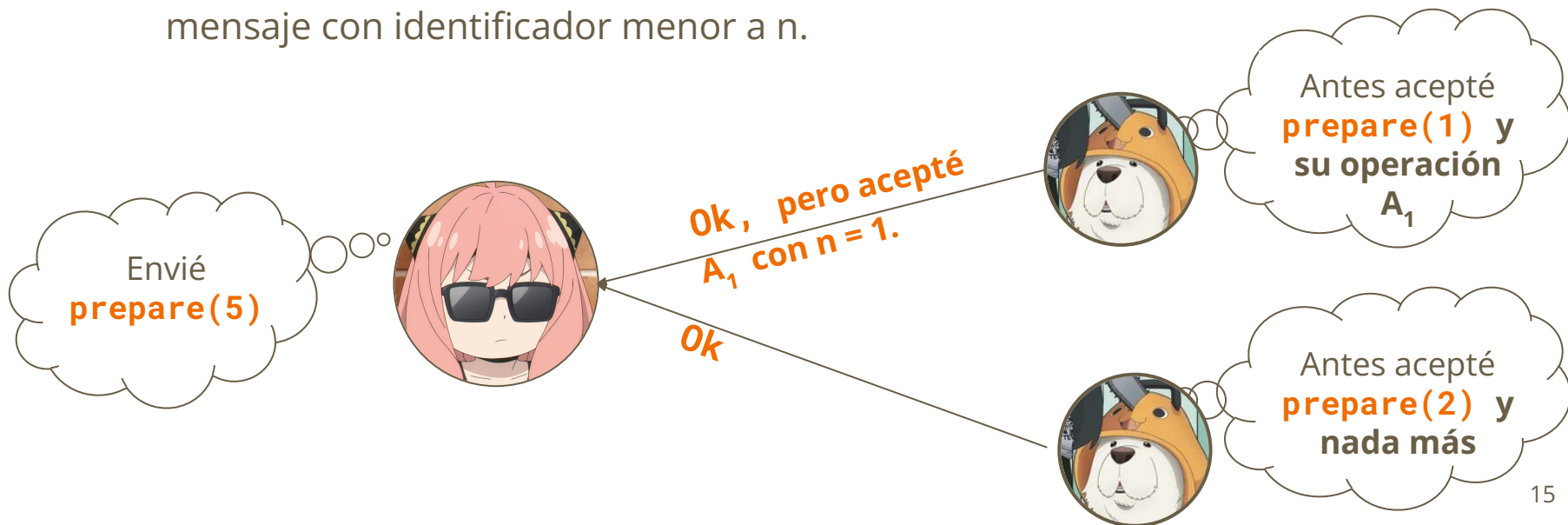
- ◆ Caso 1: Si es su primer **prepare(n)**: responde con "ok" y promete no aceptar ningún mensaje con identificador menor $n_2 < n$.
- ◆ Caso 2: Si ya recibió un **prepare(n₂)**.
 - Si $n_2 > n \rightarrow$ rechaza **prepare(n)**.
 - Si $n_2 < n \rightarrow$ Caso especial



Paxos - Fase 1 - Preparar propuesta

2. Los aceptantes **activos** responden al **prepare(n)**

- ◆ Caso especial: si un aceptante **ya aceptó una acción (A_x) previamente**, la debe compartir junto con su identificador.
- ◆ De todas formas responde con "ok" y promete no aceptar ningún mensaje con identificador menor a n.



Paxos - Fase 1 - Preparar propuesta (Resumen)

1. **Proponente** envía `prepare(n)`

- ◆ Se la aceptan.
- ◆ Se la rechazan.
- ◆ Se la acepta, pero le avisan si una operación previa fue aceptada.

2. **Aceptante** recibe `prepare(n)`

- ◆ La acepta y promete no aceptar nada con identificador menor a n .
- ◆ La rechaza o simplemente no responde.
- ◆ En caso de haber aceptado una operación con identificador menor, la comparte.

Paxos - Fase 2 - Aceptar propuesta

Propuesta fallida

- ◆ 50% o menos de todos los nodos de la red responden con "ok".
- ◆ Se vuelve a fase 1 con un identificador mayor.



Paxos - Fase 2 - Aceptar propuesta

Propuesta exitosa

- ◆ **Más** del 50% de todos los nodos de la red responden con "ok".
- ◆ Se envía operación a los demás nodos. Existen 2 posibles casos.



Paxos - Fase 2 - Aceptar propuesta

Propuesta exitosa - Caso 1: Nadie aceptó una operación previa

- ◆ Se manda la operación que el nodo proponente desea junto con el identificador del **prepare(n)** aceptado previamente.



Paxos - Fase 2 - Aceptar propuesta

Propuesta exitosa - Caso 2: Uno o más nodos ha aceptado una operación previa.



Paxos - Fase 2 - Aceptar propuesta

Propuesta exitosa - Caso 2: Uno o más nodos ha aceptado una operación previa.

- ◆ Se manda la operación asociada al identificador más alto junto con el identificador del **prepare(n)** aceptado previamente.



Paxos - Fase 2 - Aceptar propuesta (Resumen)

Propuesta fallida

- ◆ 50% o menos de todos los nodos de la red responden con "ok".
- ◆ Se vuelve a fase 1 con un identificador mayor.

Propuesta exitosa

- ◆ Más del 50% responden con "ok".
- ◆ Si nadie tiene una operación aceptada, se manda la que el proponente quiera.
- ◆ En otro caso, se reenvía la operación asociada al identificador más grande.

Paxos - Fase 3 - Aprender propuesta

- ◆ Los *learners* verifican la operación aceptada.
- ◆ Si hay más del 50% de los nodos aceptantes activos con la misma operación. Se "aprende" dicha operación de forma permanente.
- ◆ Si los nodos *learners* no verifican. No hay consolidación.



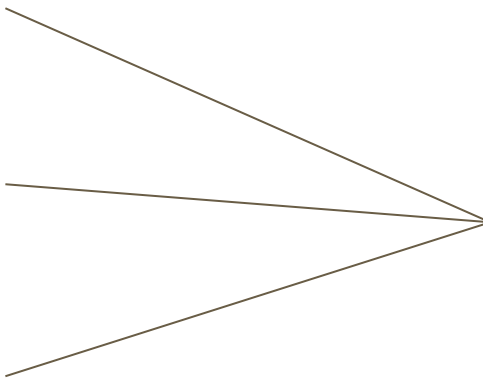
accept (A_2 , 1)



accept (A_2 , 1)



accept (A_1 , 5)

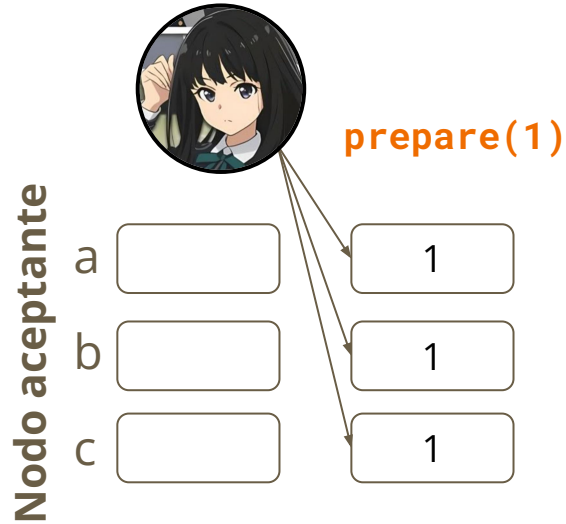


Me quedo con
accept (A_2 , 1)



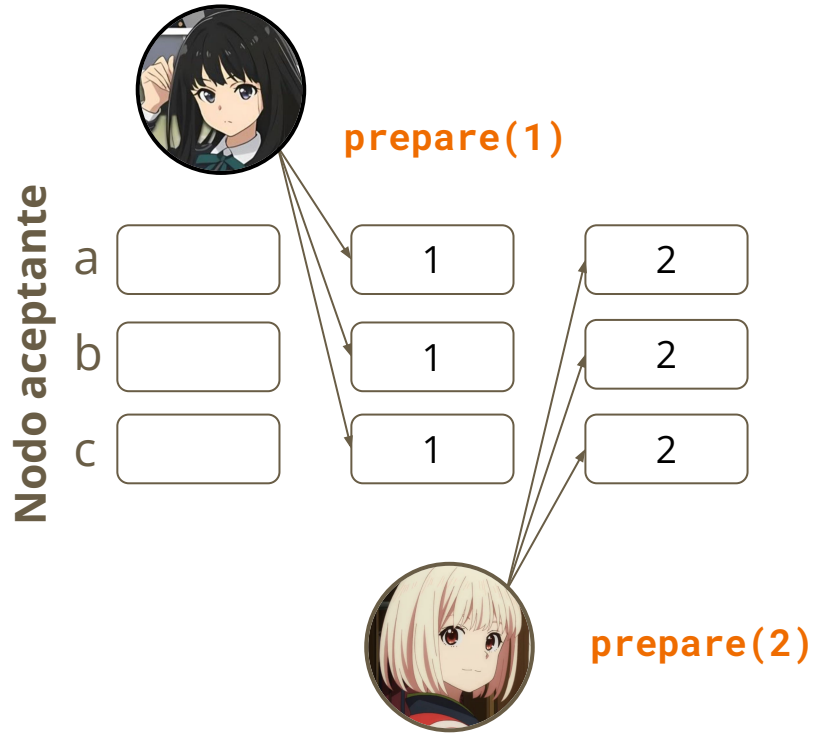
Paxos - Casos interesantes

Ejemplo 1. Interrumpir a un proponente



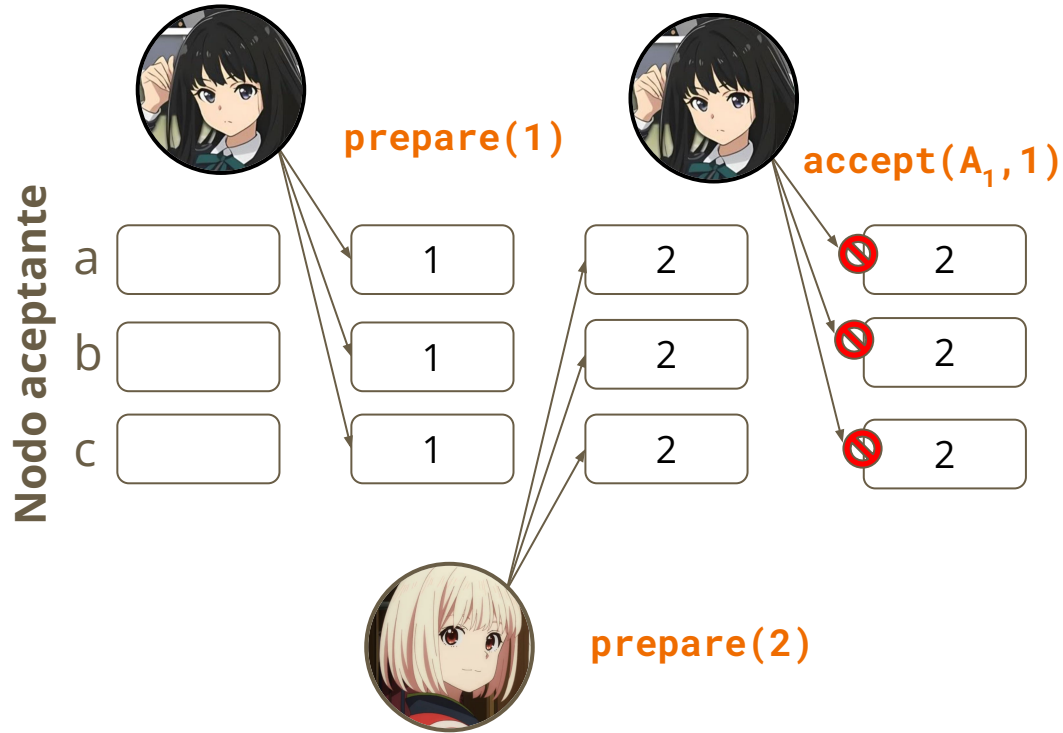
Paxos - Casos interesantes

Ejemplo 1. Interrumpir a un proponente



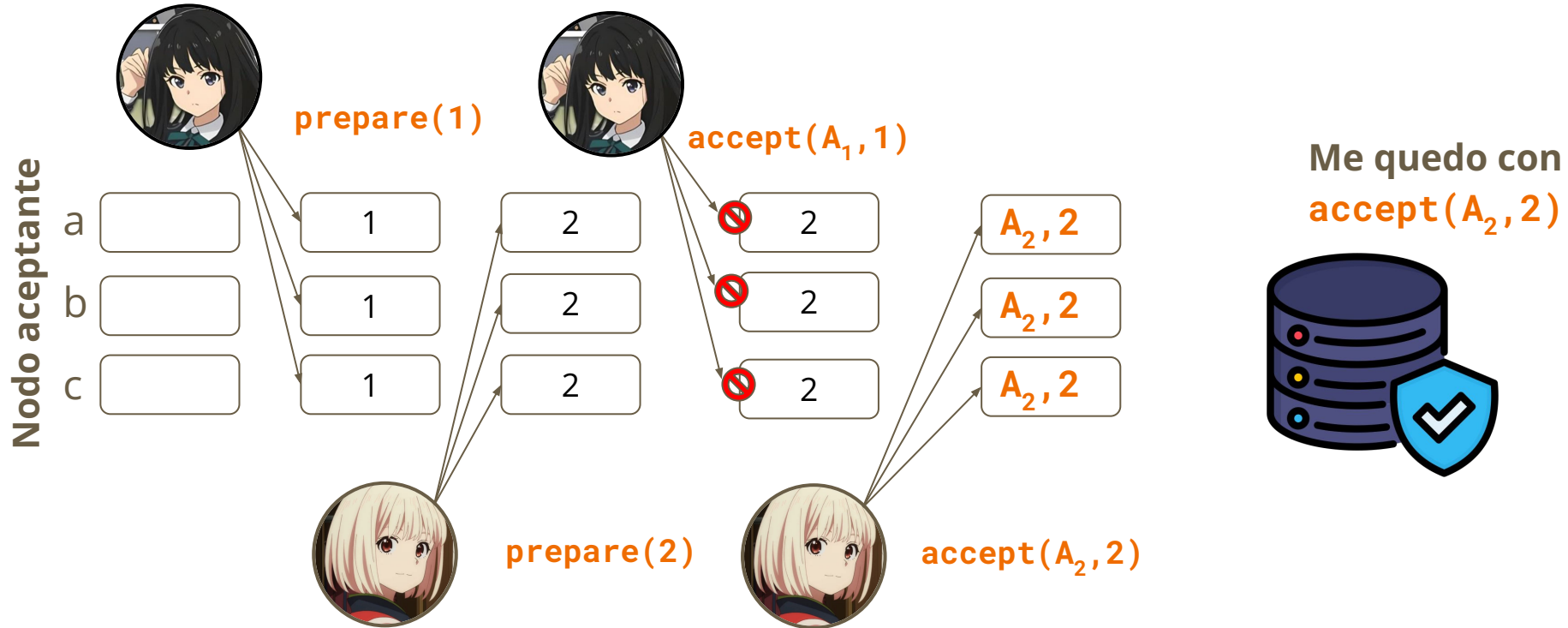
Paxos - Casos interesantes

Ejemplo 1. Interrumpir a un proponente



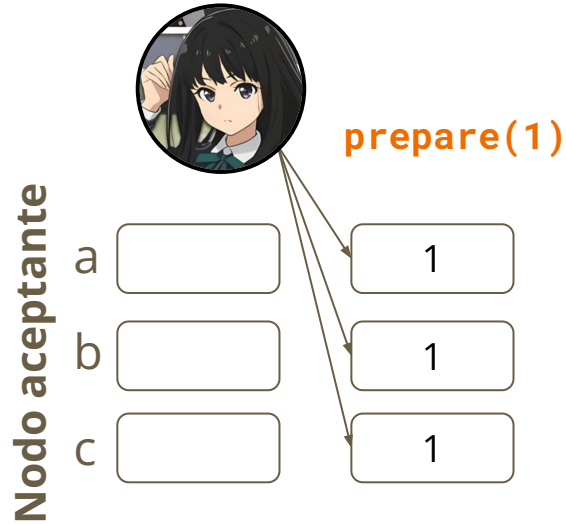
Paxos - Casos interesantes

Ejemplo 1. Interrumpir a un proponente



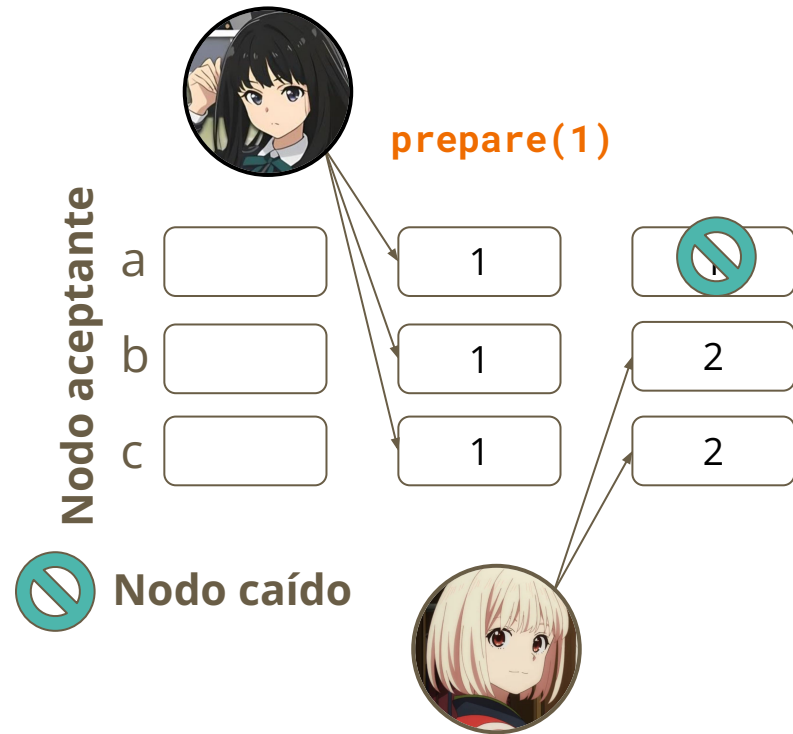
Paxos - Casos interesantes

Ejemplo 2. Sobrecribir acción que tenía consenso



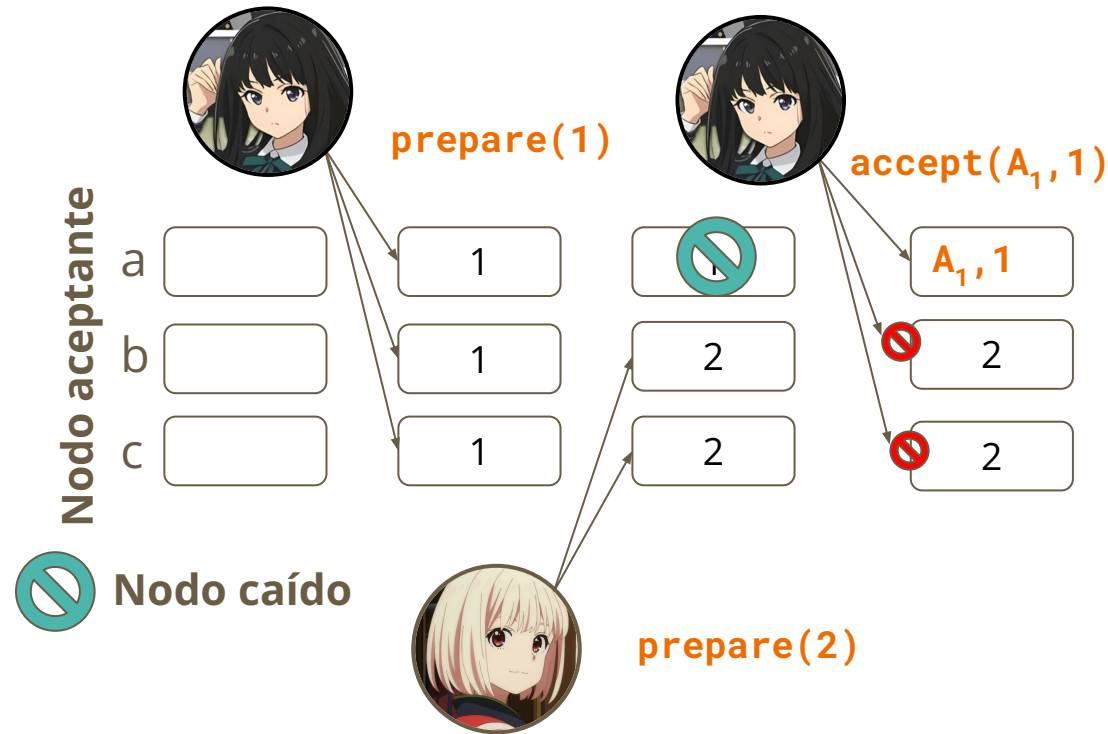
Paxos - Casos interesantes

Ejemplo 2. Sobrecribir acción que tenía consenso



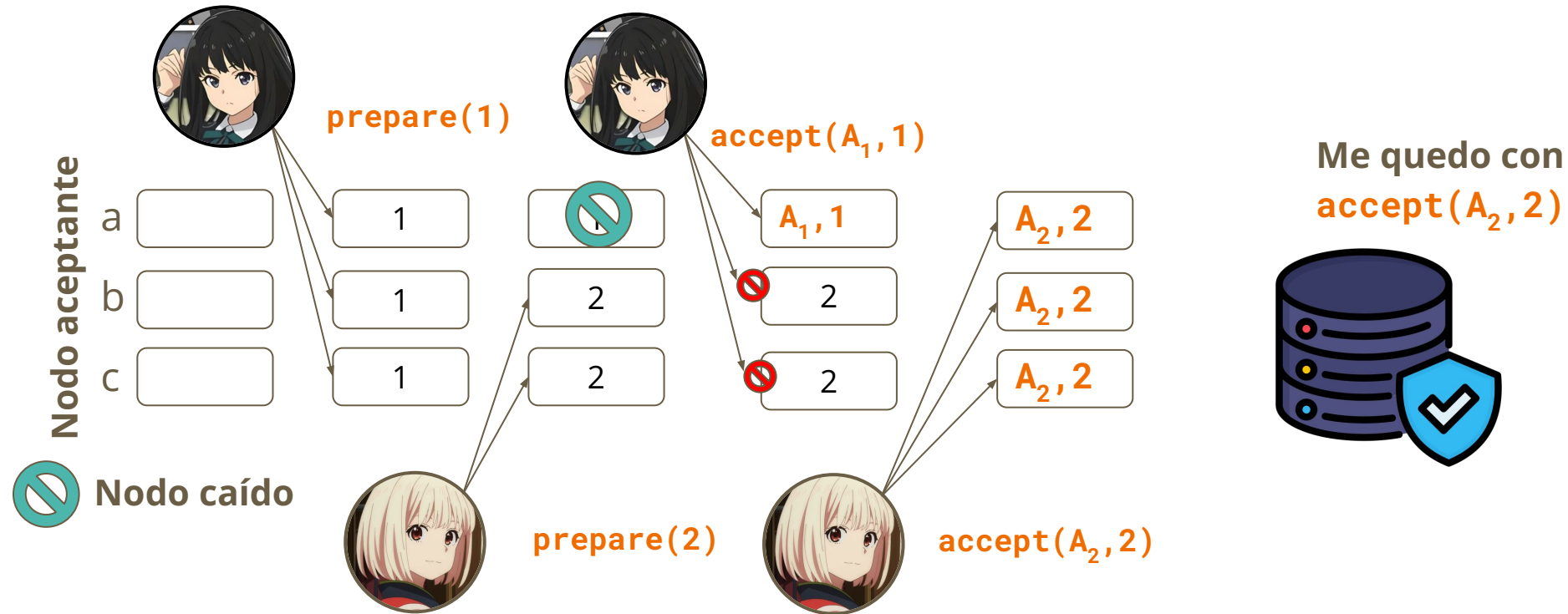
Paxos - Casos interesantes

Ejemplo 2. Sobrecribir acción que tenía consenso



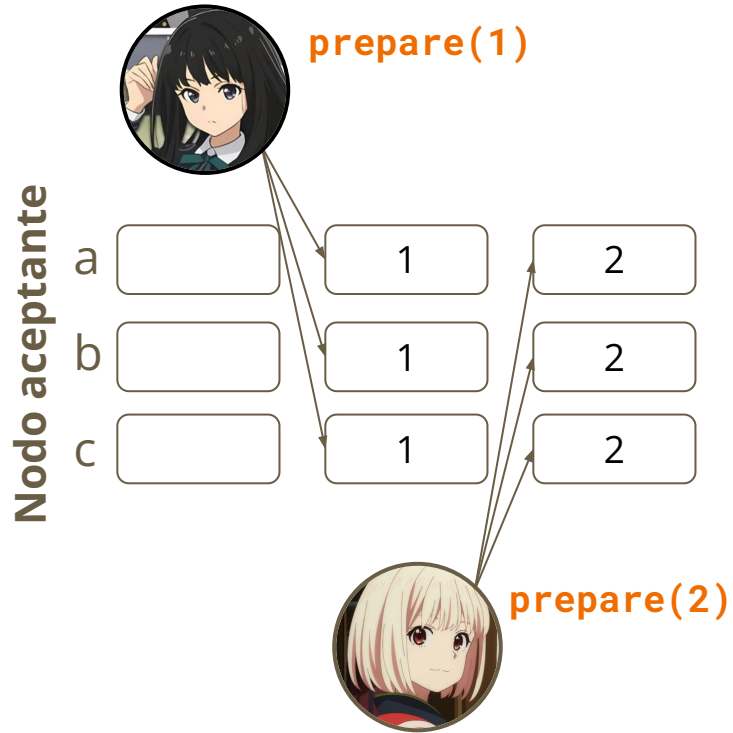
Paxos - Casos interesantes

Ejemplo 2. Sobrecribir acción que tenía consenso



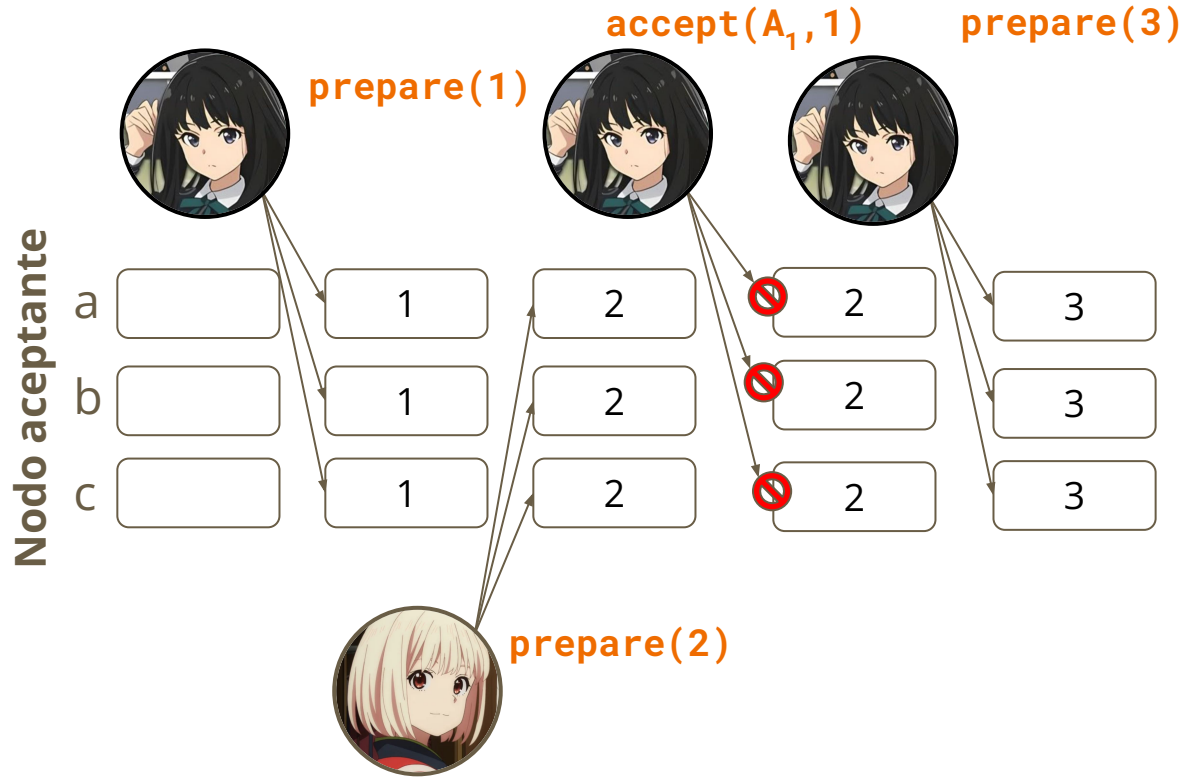
Paxos - Casos interesantes

Ejemplo 3. *LiveLock*



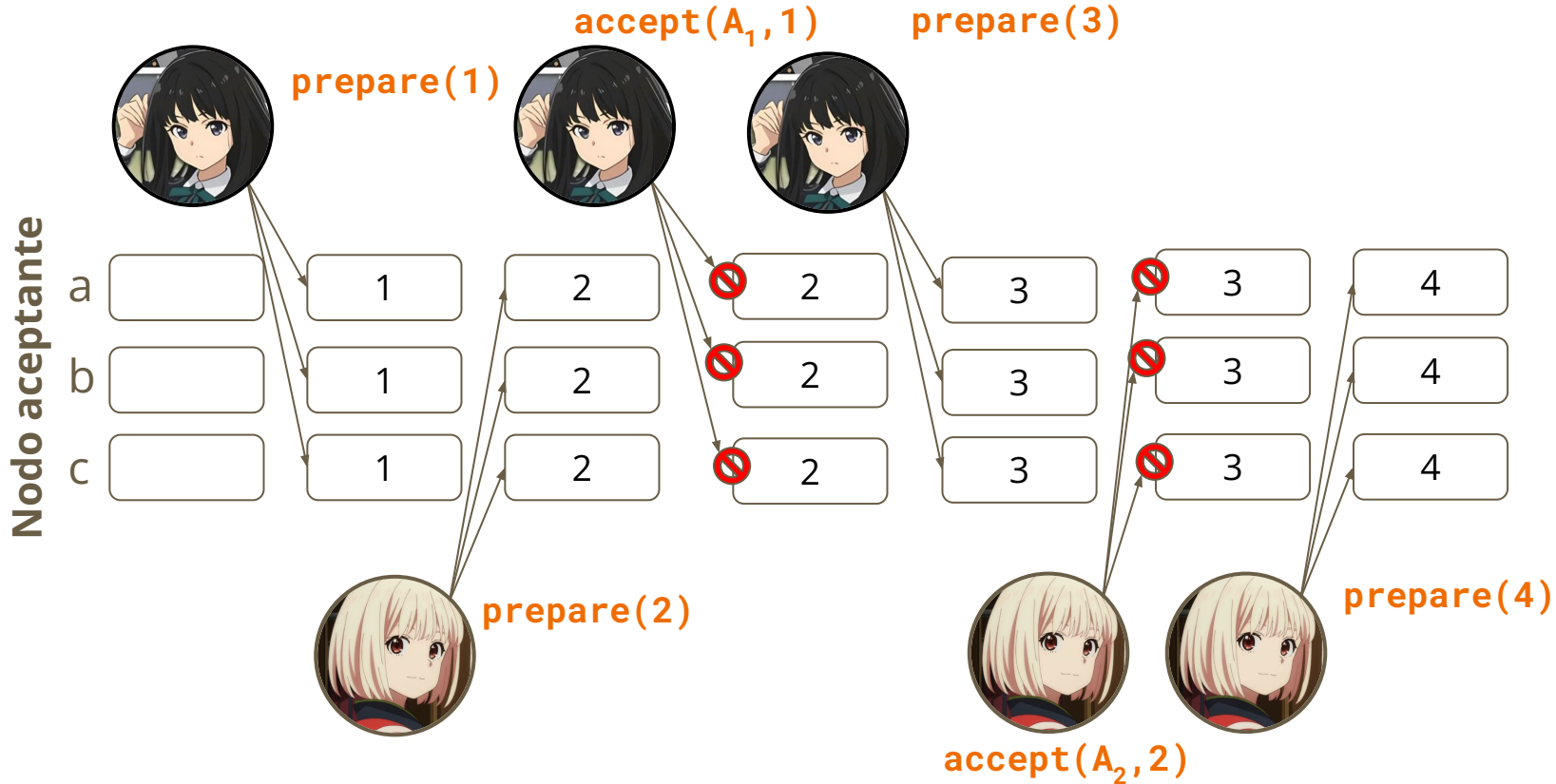
Paxos - Casos interesantes

Ejemplo 3. *Livelock*



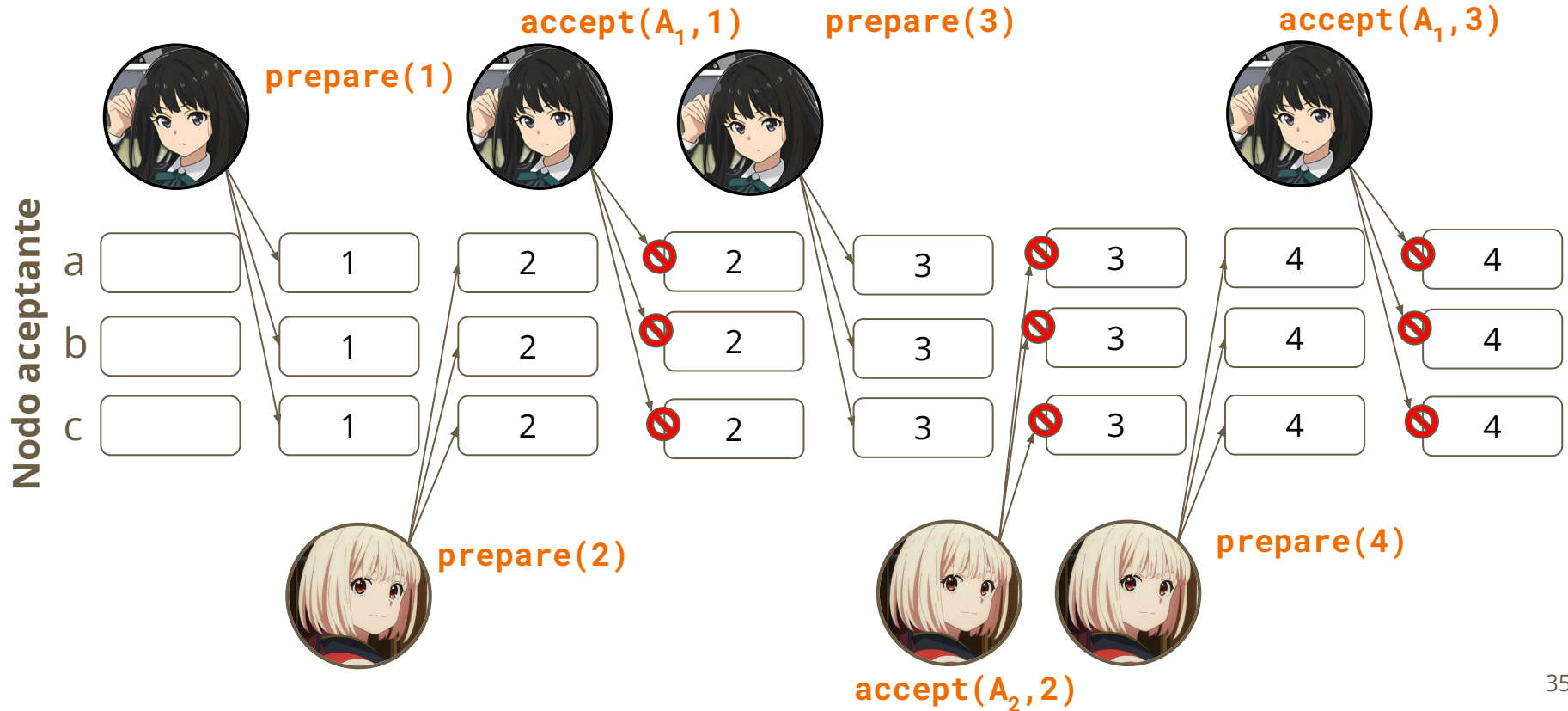
Paxos - Casos interesantes

Ejemplo 3. *Livelock*



Paxos - Casos interesantes

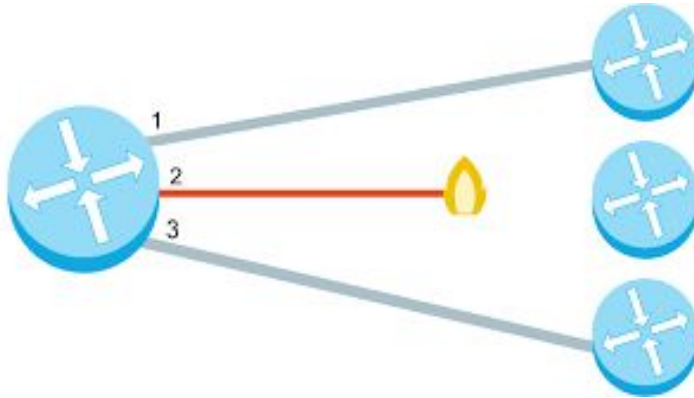
Ejemplo 3. Livelock



Paxos - Garantías

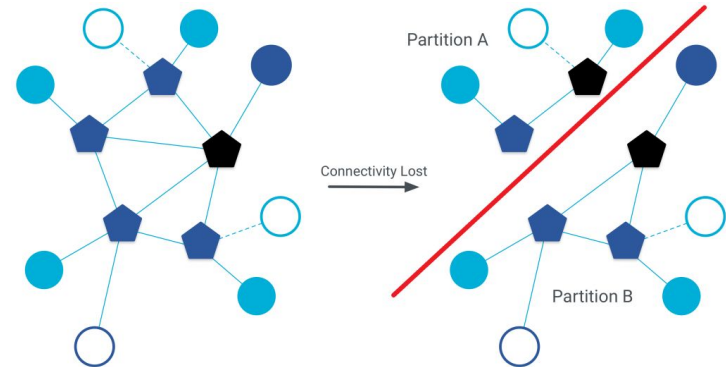
◆ Tolerancia a fallos

Pueden fallar $(N-1) // 2$ nodos.



◆ Tolerancia a particiones.

Si se divide la red en 2 grupos, mientras uno tenga la mayoría, el algoritmo funciona.




Paxos - Observaciones

- ◆ La interacción entre proponentes o caídas de nodos no asegura que el primer "prepare" que logra consenso consolide la operación final.
- ◆ Paxos, en su forma original, está diseñado para acordar una sola acción, mientras que variaciones, como Multi-Paxos, optimizan el proceso para secuencias de acciones.
- ◆ Aunque existe el *livelock*. Hay alternativas en la implementación del algoritmo para asegurar respuesta, por ejemplo, *timeout* para llegar a consenso o retrasar mensajes aleatoriamente.

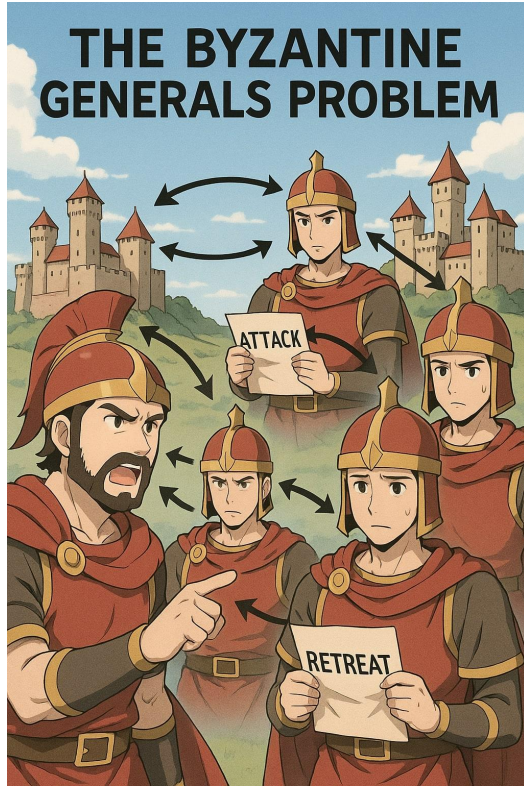
Paxos - Vida Real

- ◆ **Google Analytics** utiliza Paxos en su servicio de bloqueo distribuido "Chubby" para mantener la consistencia de las réplicas en caso de fallo.
- ◆ **Amazon DynamoDB** (base de datos NoSQL) utiliza el algoritmo Paxos para la elección de líderes y el consenso.
- ◆ **Apache Cassandra** (base de datos NoSQL) utiliza Paxos únicamente para la *feature* de *Light Weight Transaction* (un tipo de transacción condicional).

Problema de los generales bizantino

¿Qué ocurre cuando hay un
traidor? 

Problema de los generales bizantino

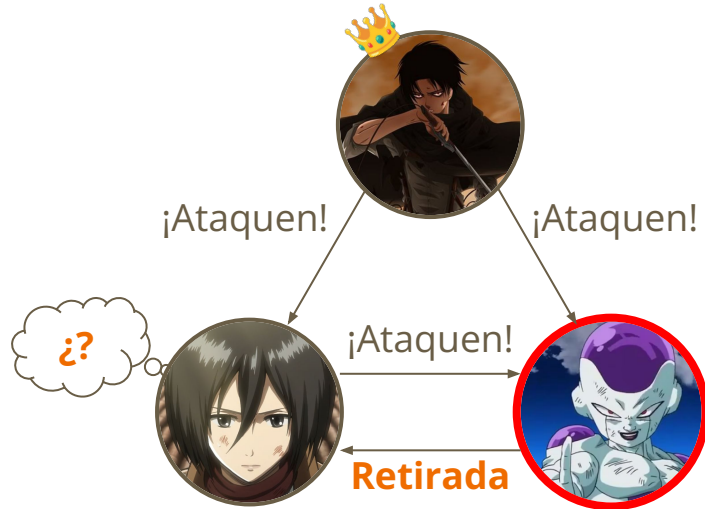


Problema de los generales bizantino

- ◆ Existe uno o más nodos que se comportan de manera **impredecible o maliciosa respecto al protocolo**: puede enviar mensajes contradictorios, omitir pasos, o actuar de forma arbitraria, haciendo difícil que los nodos correctos lleguen a un consenso confiable.
- ◆ Muchos algoritmos (como paxos) asume fallas de nodos o particiones de red, pero no que un nodo entregue un valor distinto al indicado por el protocolo.
- ◆ Cuando hay 1 nodo coordinador (líder) y 2 nodos (seguidores). Con que un nodo sea traidor ya no se logra consenso.

Problema de los generales bizantino

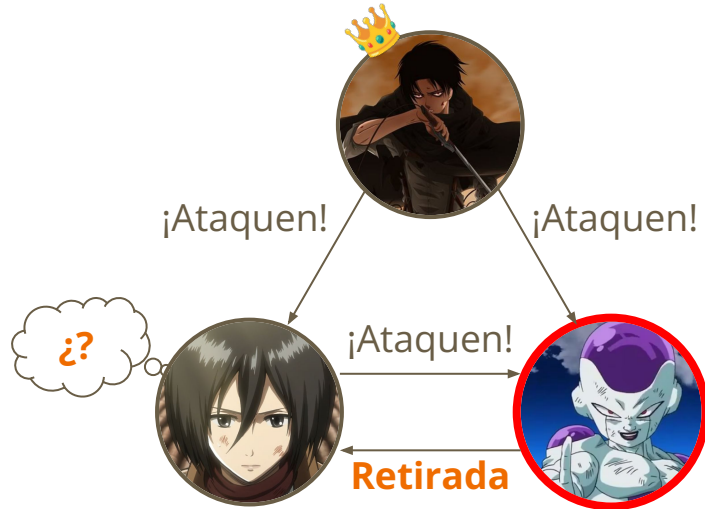
- ◆ Cuando hay 1 nodo coordinador (líder) y 2 nodos (seguidores). Con que un nodo sea traidor ya no se logra consenso.



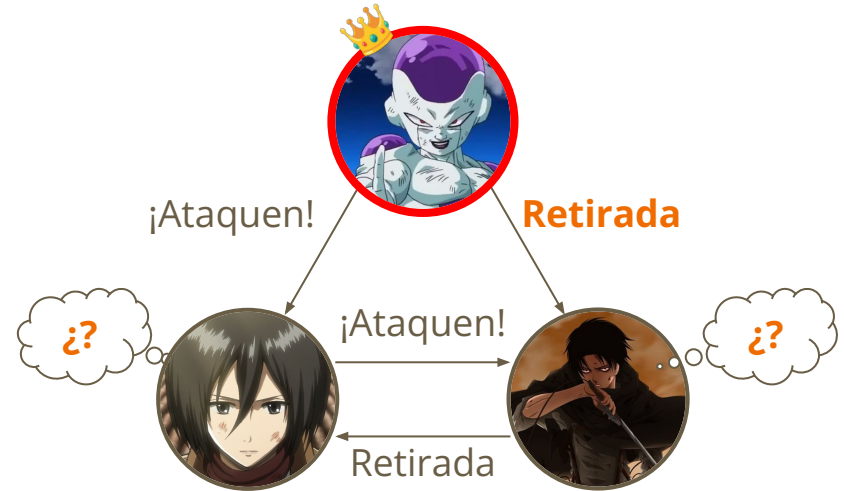
Seguidor traidor

Problema de los generales bizantino

- ◆ Cuando hay 1 nodo coordinador (líder) y 2 nodos (seguidores). Con que un nodo sea traidor ya no se logra consenso.



Seguidor traidor

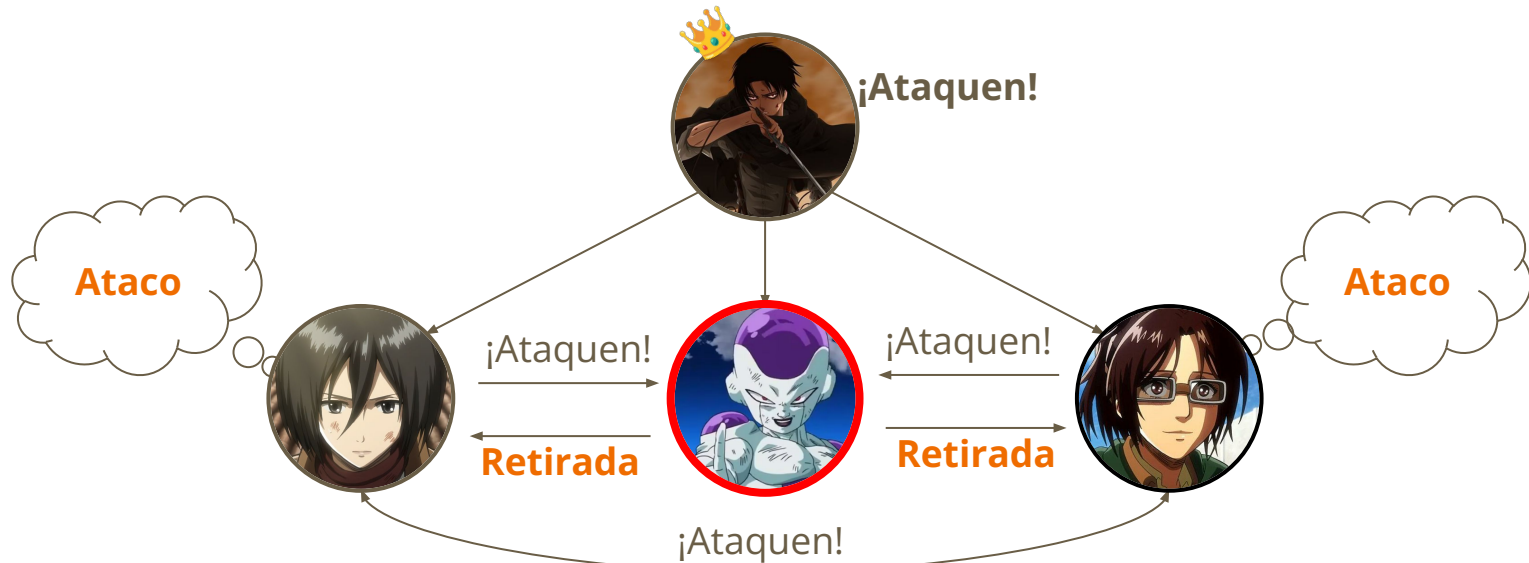


Coordinador traidor

Problema de los generales bizantino - Soluciones

Mecanismo de mensajes orales (OM)

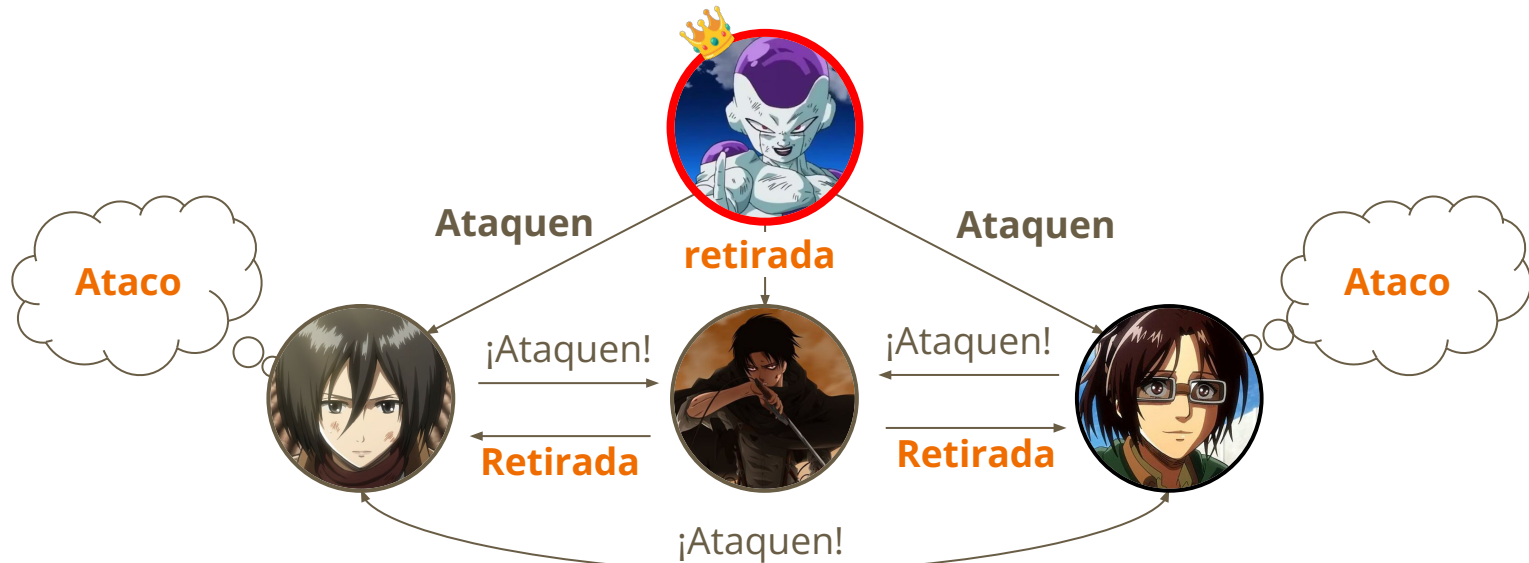
- ◆ Una vez se recibe el mensaje del líder, se propaga a todos. Me quedo con el mensaje más repetido. En caso de empate, se debe definir una regla determinista.
- ◆ Si hay n traidores, se necesitan $\underline{3n + 1}$ nodos en total.



Problema de los generales bizantino - Soluciones

Mecanismo de mensajes orales (OM)

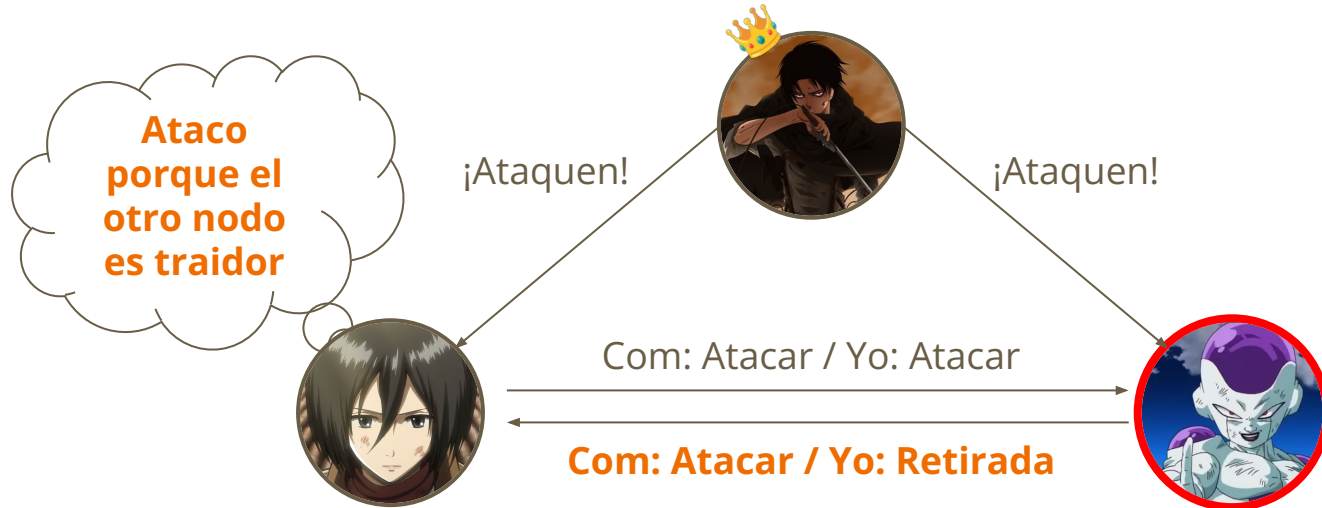
- ◆ Una vez se recibe el mensaje del líder, se propaga a todos. Me quedo con el mensaje más repetido. En caso de empate, se debe definir una regla determinista.
- ◆ Si hay n traidores, se necesitan $\underline{3n + 1}$ nodos en total.



Problema de los generales bizantino - Soluciones

Mecanismo de mensajes firmados (SM)

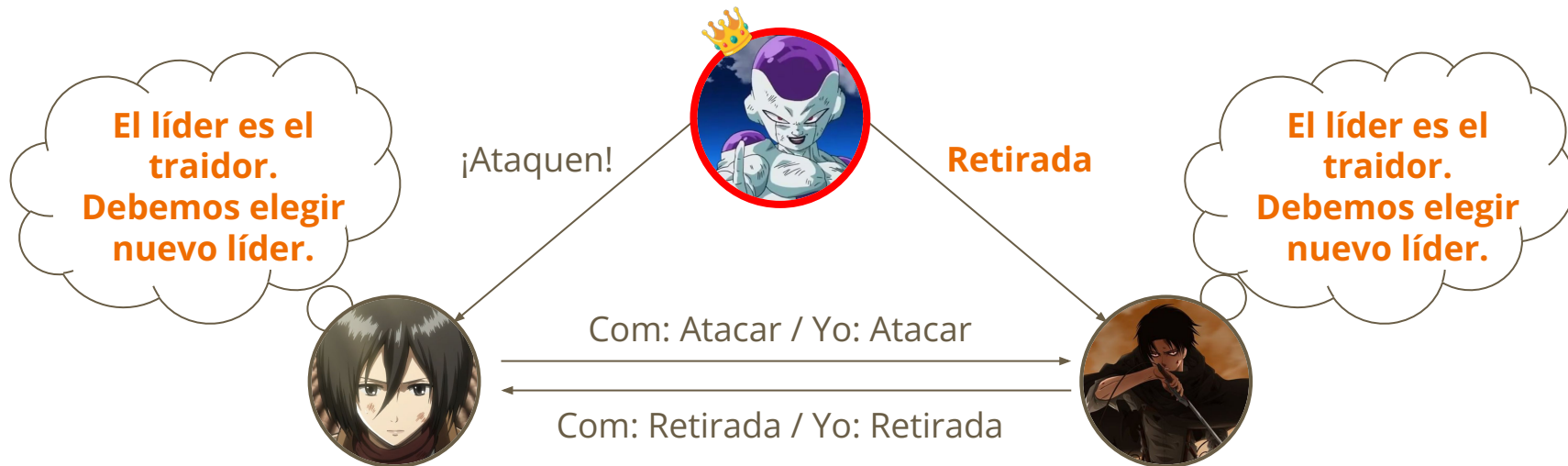
- ◆ El líder firma su mensaje, los demás nodos copian el mensaje, lo firman y envían ambos mensaje (el original y la copia). Se asume que la firma no se puede falsificar.
- ◆ Si hay n traidores, se necesitan $2n + 1$ nodos en total.



Problema de los generales bizantino - Soluciones

Mecanismo de mensajes firmados (SM)

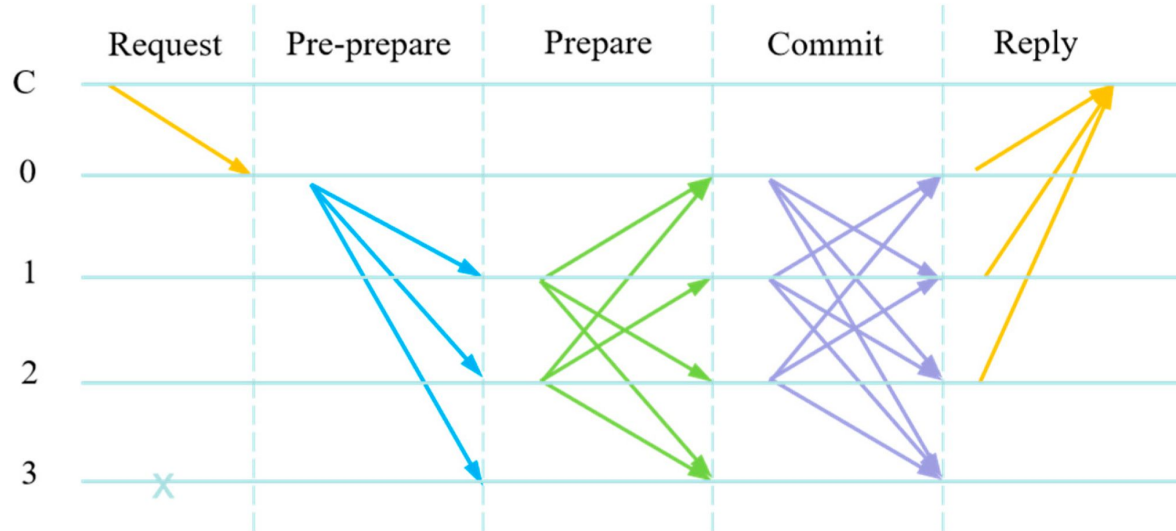
- ◆ El líder firma su mensaje, los demás nodos copian el mensaje, lo firman y envían ambos mensaje (el original y la copia). Se asume que la firma no se puede falsificar.
- ◆ Si hay n traidores, se necesitan $2n + 1$ nodos en total.



Problema de los generales bizantino - Vida Real

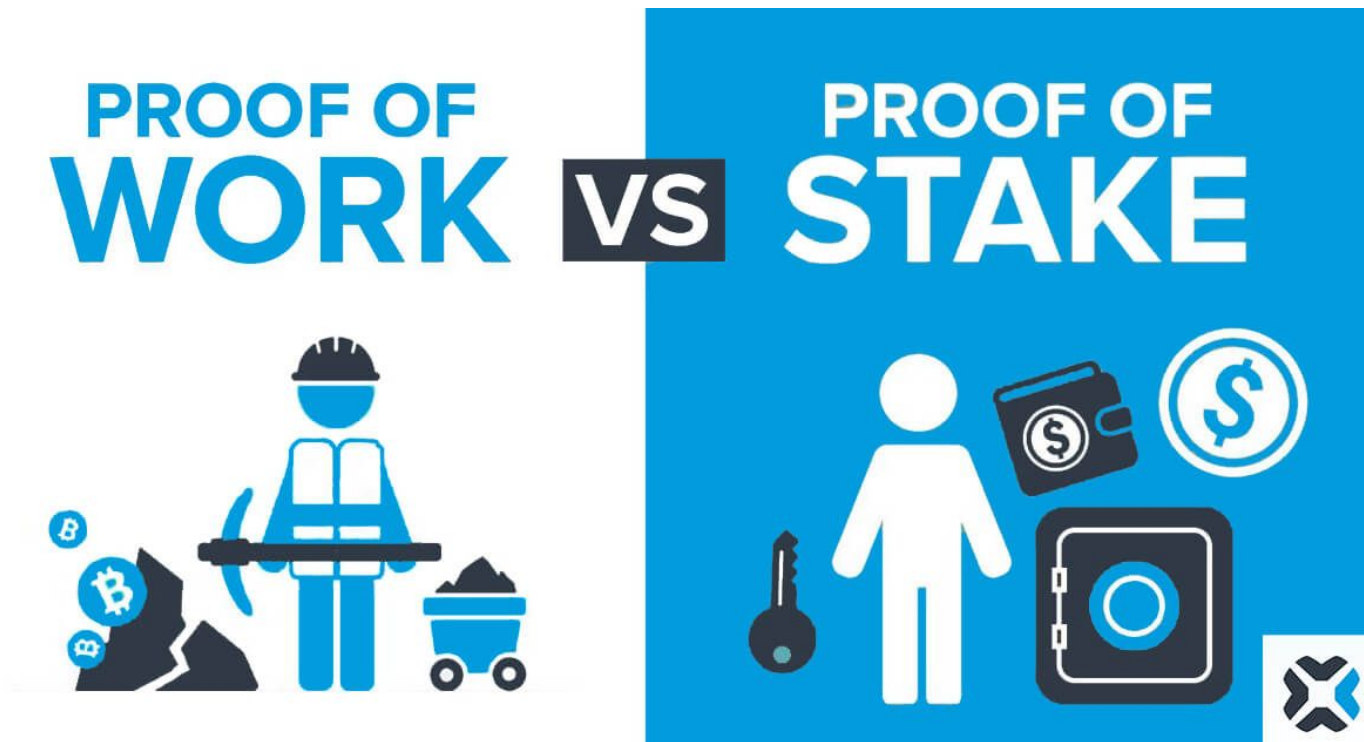
Tolerancia Práctica a Fallas Bizantinas (pBFT)

- ◆ Algoritmo de consenso que soporta fallas bizantinas propuesto en los 90.
- ◆ Aplica firma de mensaje, pero exige que si hay n fallas, se tenga al menos $3n+1$ nodos.



Problema de los generales bizantino - Vida Real

Proof of Work (PoW) y Proof of Stake (PoS) en Criptomoneda



Problema de los generales bizantino - Vida Real

Proof of Work (PoW) y Proof of Stake (PoS) en Criptomoneda

PoW

- ◆ Los nodos compiten para resolver problemas matemáticos complejos usando su poder computacional. El primero que lo logra puede crear un bloque en criptomoneda.
- ◆ Un traidor deberá ocupar demasiados recursos para solucionar el problema o bien tener el 51% de la red para que "acepten" su solución.

Problema de los generales bizantino - Vida Real

Proof of Work (PoW) y Proof of Stake (PoS) en Criptomonedas

PoW

- ◆ Los nodos compiten para resolver problemas matemáticos complejos usando su poder computacional. El primero que lo logra puede crear un bloque en criptomoneda.
- ◆ Un traidor deberá ocupar demasiados recursos para solucionar el problema o bien tener el 51% de la red para que "acepten" su solución.

PoS

- ◆ Los participantes "apuestan" una cantidad de sus criptomonedas para proponer un bloque de criptomoneda. Se elige uno entre los apostadores.
- ◆ Mediante un consenso con votos firmados, los demás validadores verifican el bloque propuesto.
- ◆ Si un validador intenta hacer trampa, los demás lo detectan y pueden rechazar su bloque. En ese caso, el validador pierde parte o la totalidad de lo apostado.

Problema de los generales bizantino - Vida Real

Proof of Work (PoW) y Proof of Stake (PoS) en Criptomonedas

PoW

- ◆ Los nodos compiten para resolver problemas matemáticos complejos usando su poder computacional. El primero que logra y puede crear una nueva criptomoneda.

PoS

- ◆ Los validadores mantienen una cierta cantidad de criptomonedas para validar un bloque. Los validadores eligen una cantidad de validadores basándose en la cantidad de criptomonedas que poseen.
- ◆ Si un validador intenta hacer trampa, los demás lo detectan y pueden rechazar su bloque. En ese caso, el validador pierde parte o la totalidad de lo apostado.

Si quieren saber más de Criptomonedas, existe el curso IIC3272 - Criptomonedas y Contratos Inteligentes [2019 y 2020].
Ahora se está dictando nuevamente, pero no sé cuando será nuevamente. En el enlace están los PPT y las clases grabadas

Poniendo a prueba lo que hemos aprendido 🙄

Respecto a los algoritmos de consenso, ¿cuál de las siguientes afirmaciones es **correcta**?

- I. Paxos es tolerante únicamente a caídas de nodos.
- II. Paxos, por defecto, no está diseñado para el problema de los generales bizantinos.
- III. En Paxos, una operación se considera consolidada cuando la mayoría de los nodos aceptantes la aceptan.

- a. Solo I
- b. Solo II
- c. Solo III
- d. II y III
- e. I, II y III

Poniendo a prueba lo que hemos aprendido 🙄

Respecto a los algoritmos de consenso, ¿cuál de las siguientes afirmaciones es **correcta**?

- I. Paxos es tolerante únicamente a caídas de nodos.
- II. Paxos, por defecto, no está diseñado para el problema de los generales bizantinos.
- III. En Paxos, una operación se considera consolidada cuando la mayoría de los nodos aceptantes la aceptan.


- a. Solo I
- b. Solo II**
- c. Solo III
- d. II y III
- e. I, II y III

Próximos eventos

Próxima clase

- ◆ Profundizaremos en algoritmos de elección de líder
 - ◆ ¿Qué formas tenemos de votar por un líder?
 - ◆ ¿Qué podemos hacer luego de tener un líder?

Evaluación

- ◆ El viernes se entrega la Tarea 1, si no la han empezado, haganlo con tiempo .
- ◆ El jueves se publica el control 3 que evalúa de la clase 5 hasta la clase 8.
 - ◆ *Spoiler*: Tendrán que poner en práctica si o si varios algoritmos.

IIC2523

Sistemas Distribuidos

— Hernán F. Valdivieso López —
(2025 - 2 / Clase 07)

Créditos (animes utilizados)

Lycoris Recoil



Shingeki no Kyojin



Spy x Family



Dragon Ball

