IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 10)

Resumen Unidad 2
Algoritmos de Coordinación

Temas de la clase

1. Sincronización de relojes
    a. Cristian, Berkeley, NTP, Reloj Lógico, Reloj Vectorial.
2. Estado global: *Chandy Lamport*
3. Consenso
    a. Paxos, Bizantinos (SM, OM).
4. Elección de líder
    a. *Bully*, *Ring-Based*, Raft.
5. Algoritmos de Exclusión Mutua Distribuida
    a. *Token Ring*, Servidor Central y *Multicast*.

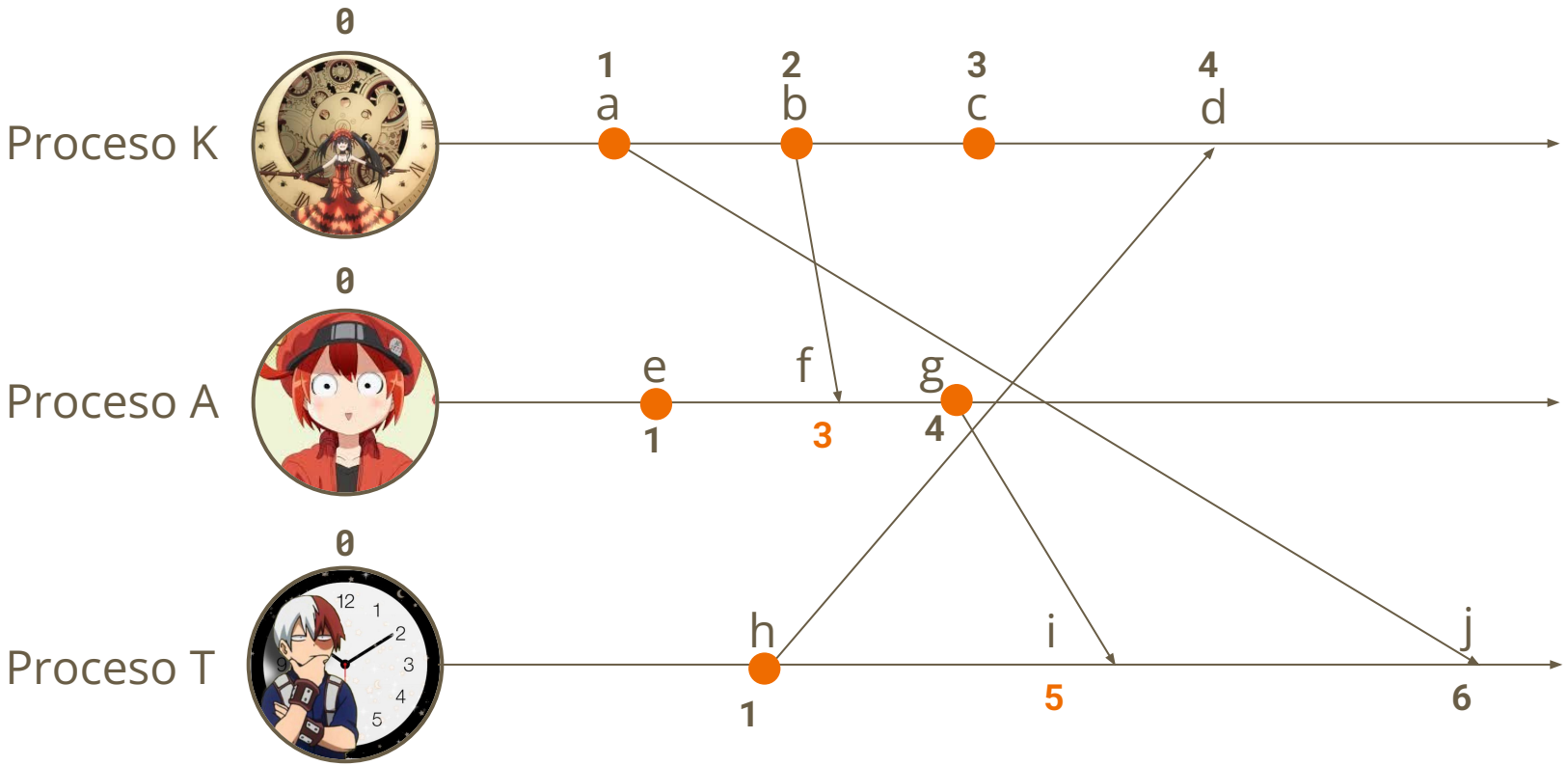
Sincronización de Relojes

Sincronización de relojes

- Los sistemas distribuidos, por su naturaleza, operan con relojes físicos individuales que pueden diferir con el tiempo debido a variaciones en el *hardware* y otros factores.
    - Todo reloj físico que no sea atómico se va a desincronizar eventualmente.
- La sincronización busca mitigar estas diferencias para asegurar un orden temporal consistente de los eventos.
- Puede ser mediante relojes lógicos o sincronizando relojes físicos.

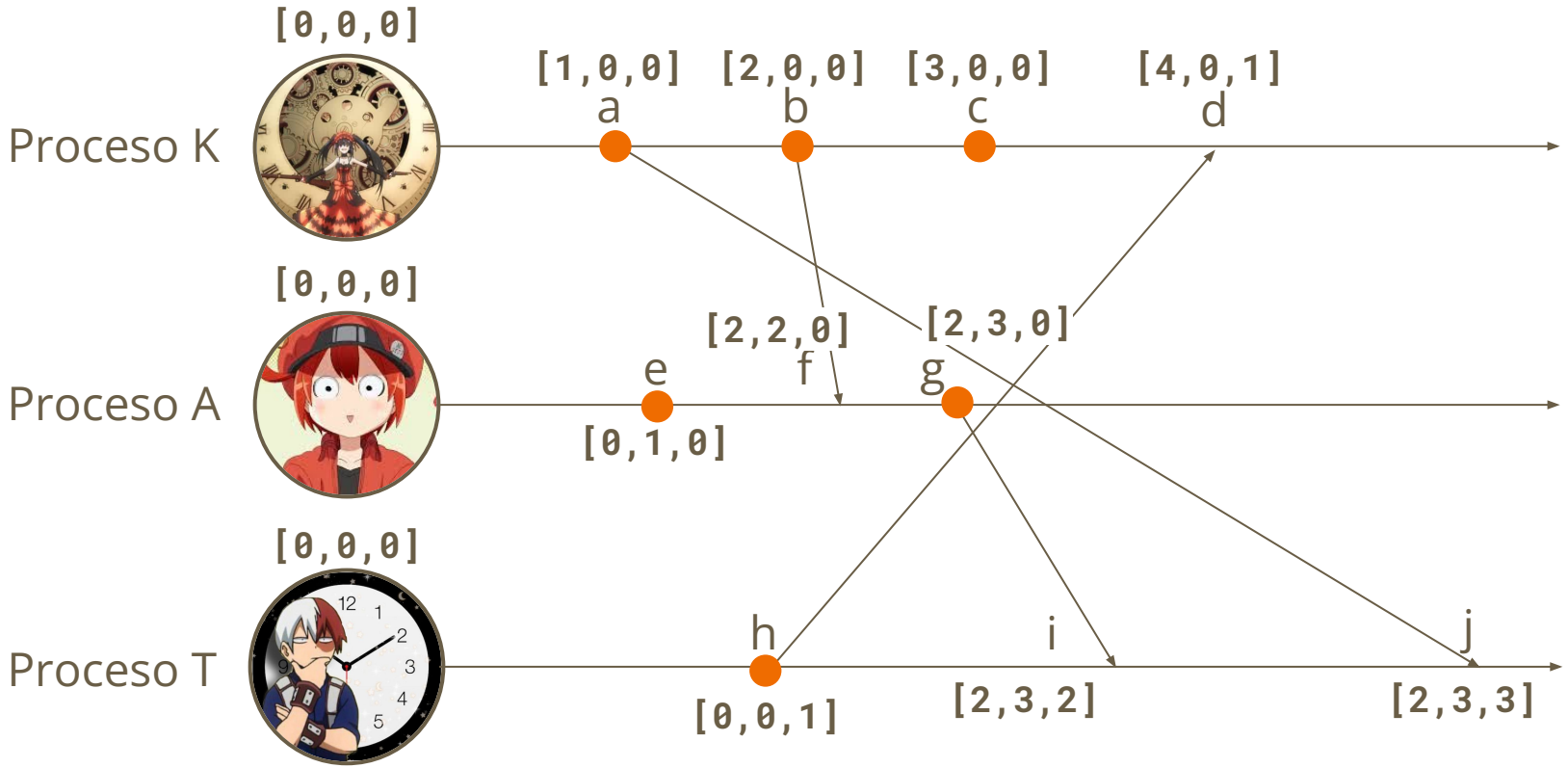
Sincronización de relojes - Algoritmos (Reloj Lógico)

Relojes Lógicos de Lamport



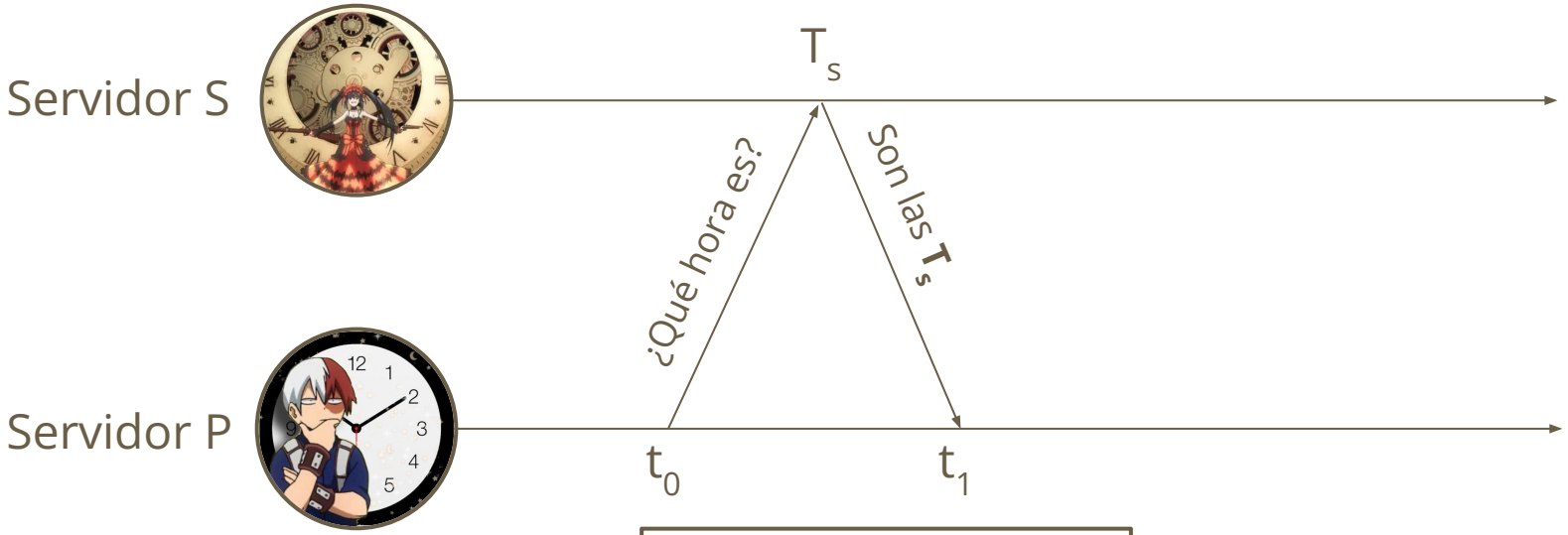
Sincronización de relojes - Algoritmos (Reloj Lógico)

Relojes Vectoriales



Sincronización de relojes - Algoritmos (Reloj Físico)

Método de Cristian



$$T_{servidor\ p} = T_s + \frac{(t_1 - t_0)}{2}$$

Sincronización de relojes - Algoritmos (Reloj Físico)

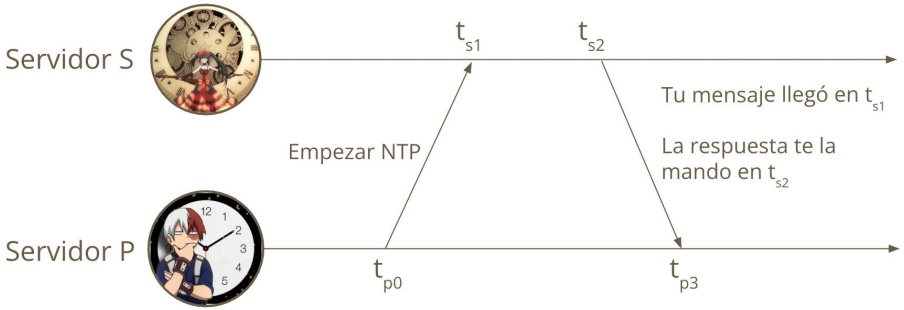
Algoritmo de Berkeley - Funcionamiento

1. El líder aplica el método de Cristian para estimar la hora actual de cada nodo seguidor considerando el *delay* de comunicación.
2. Calcula el promedio entre las horas de todos los nodos, incluido el líder
3. Líder ajusta su reloj al promedio calculado.
4. Se manda la diferencia entre la hora estimada y la hora promedio a cada nodo.
5. Los nodos aplican ajuste a sus relojes según la diferencia indicada.

Sincronización de relojes - Algoritmos (Reloj Físico)

Protocolo de Tiempo de Red (NTP)

- NTP realiza este proceso varias veces y selecciona aquel con menor retraso para aplicar su desfase al servidor.
- Este proceso se puede hacer con diferentes servidores para obtener más valores.



$$\text{Retraso} = (T_{s1} - T_{p0}) + (T_{p3} - T_{s2}) = (T_{p3} - T_{p0}) - (T_{s2} - T_{s1})$$

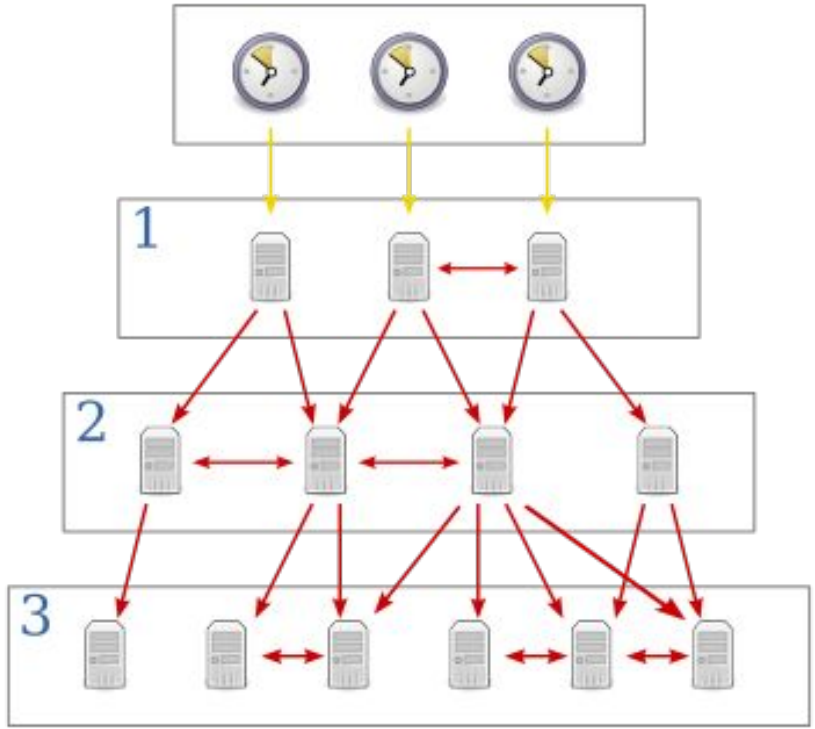
$$\text{Desfase} = \frac{(T_{s1} - T_{p0}) + (T_{s2} - t_{p3})}{2}$$

Sincronización de relojes - Algoritmos (Reloj Físico)

Protocolo de Tiempo de Red (NTP)

Funciona a partir de estratos.

- Estrato 0: relojes más eficientes.
- Estrato N se sincroniza entre ellos y con el estrato anterior
- A medida que N es mayor, la sincronización será menos precisa.

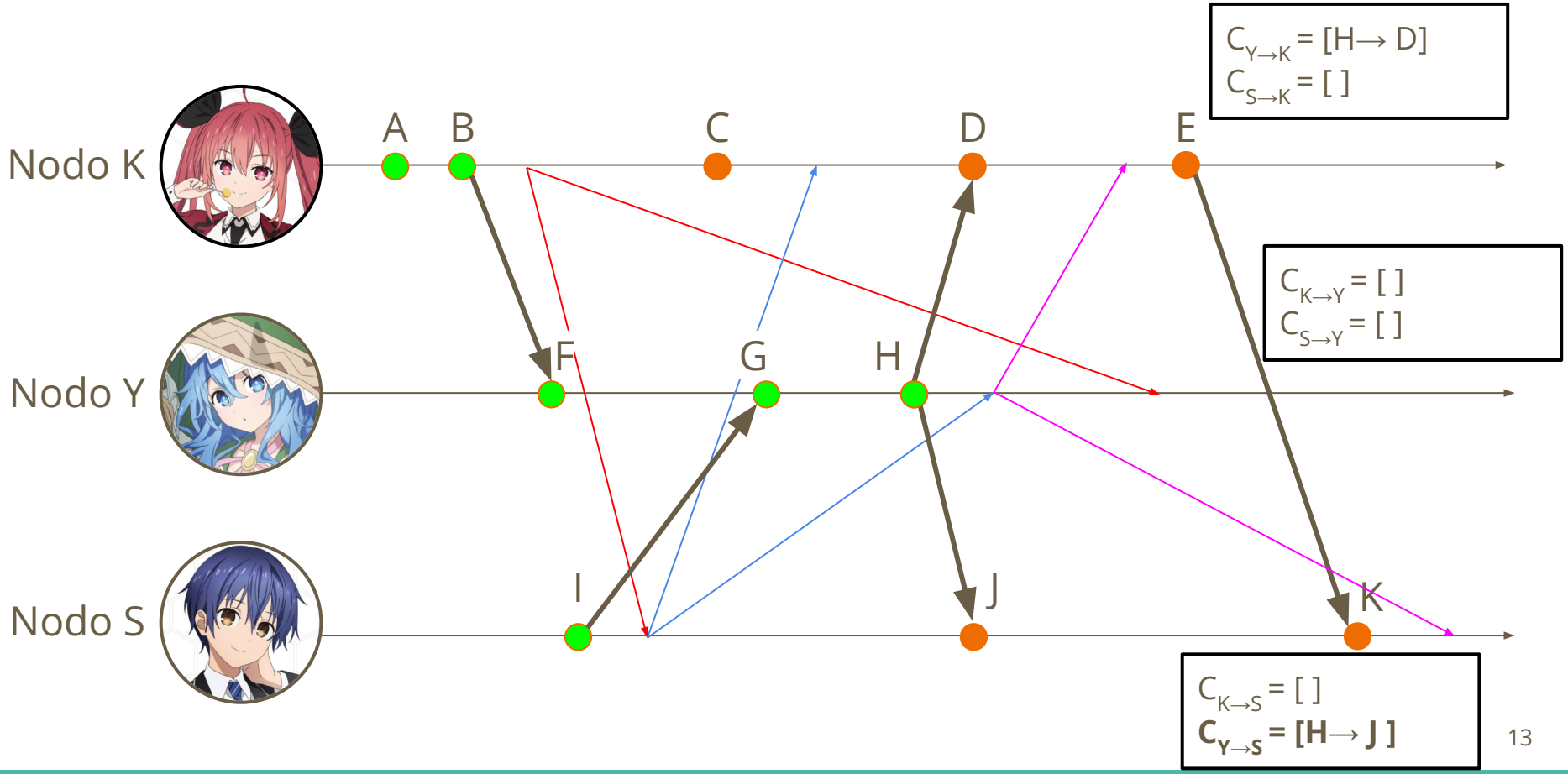


Fuente: Network Time Protocol

Estado Global

Chandy Lamport

Algoritmo de Chandy-Lamport - Ejemplo



Consenso

Algoritmos de consenso - ¿Que son?

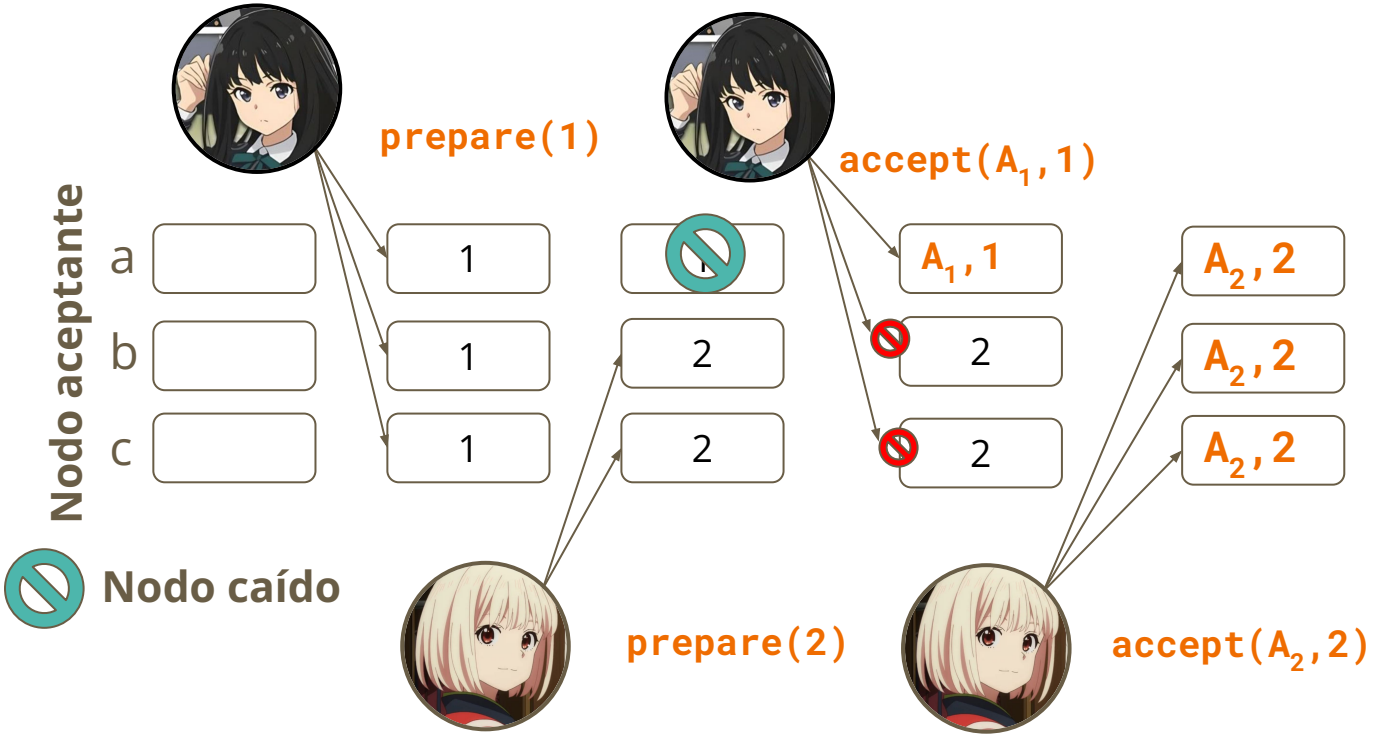
Protocolo que permite a varios nodos acordar una operación o acción en común, aún en presencia de fallos.

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.

Paxos - Casos interesantes

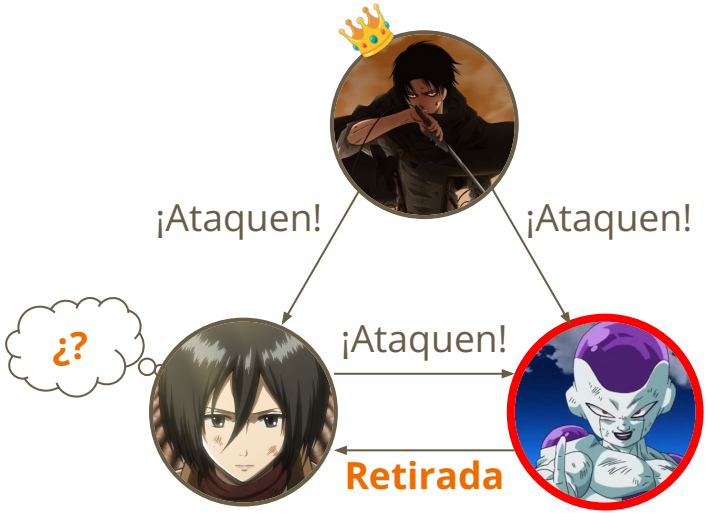
Ejemplo 2. Sobreescribir acción que tenía consenso



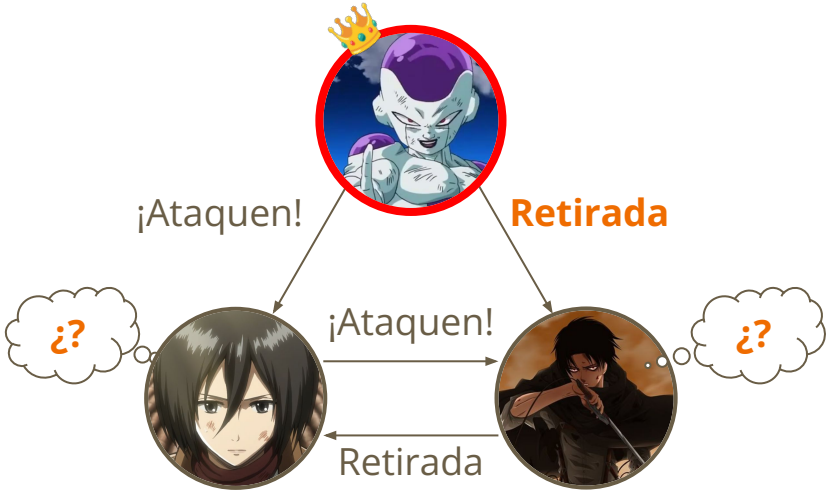
Me quedo con
accept(A_2,2)

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

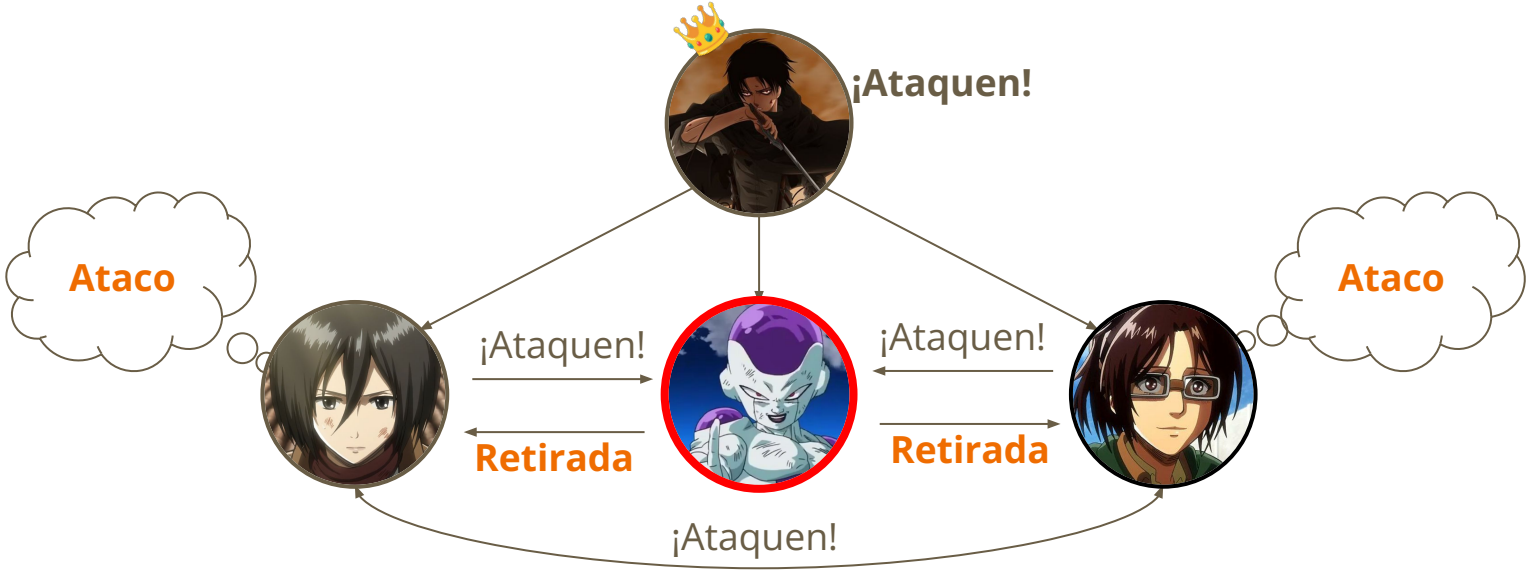


Coordinador traidor

Problema de los generales bizantino - Soluciones

Mecanismo de mensajes orales (OM)

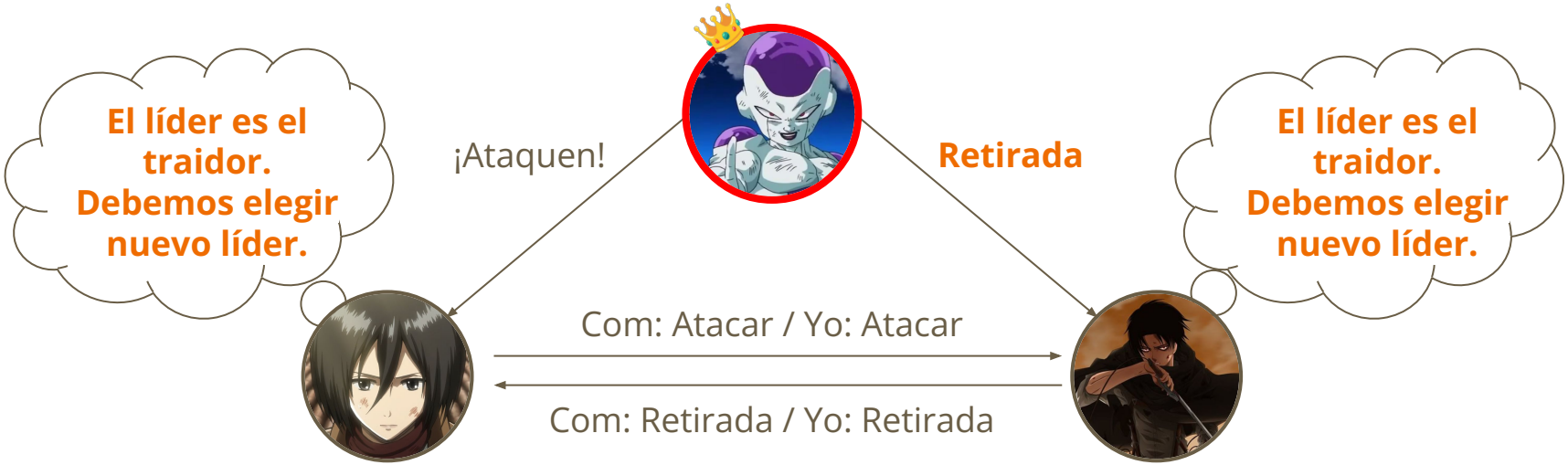
- Una vez se recibe el mensaje del líder, se propaga a todos. Me quedo con el mensaje más repetido. En caso de empate, se debe definir una regla determinista.
- Si hay n traidores, se necesitan $3n + 1$ nodos en total.



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.

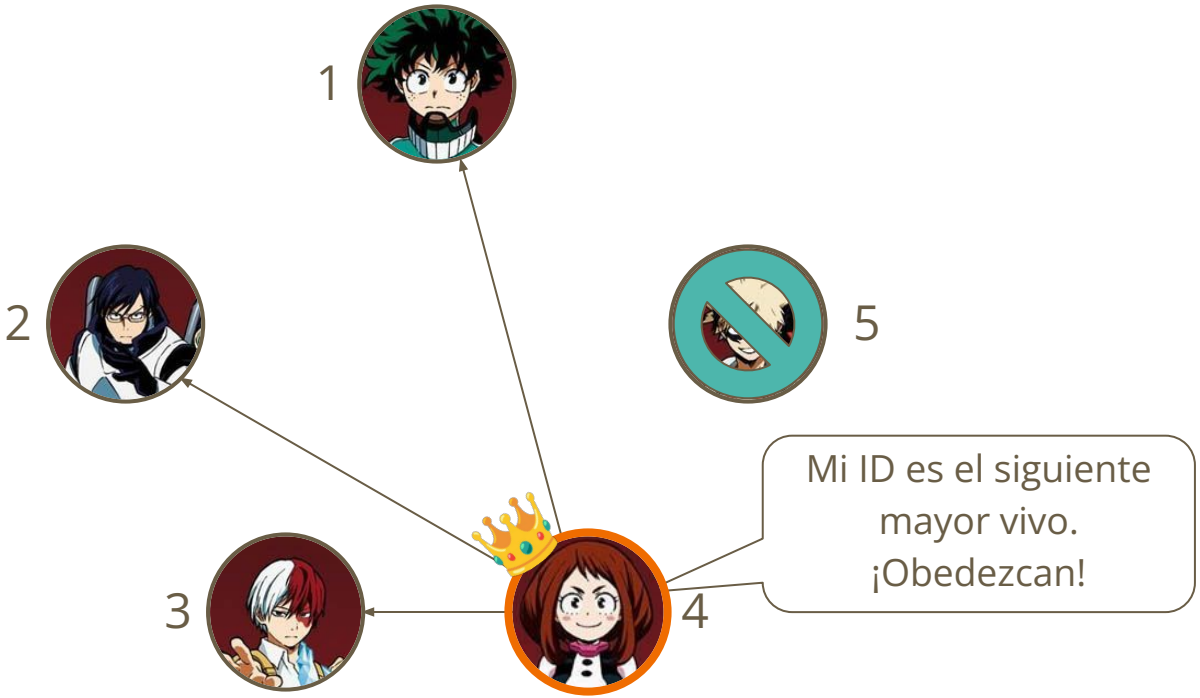


Elección de Líder

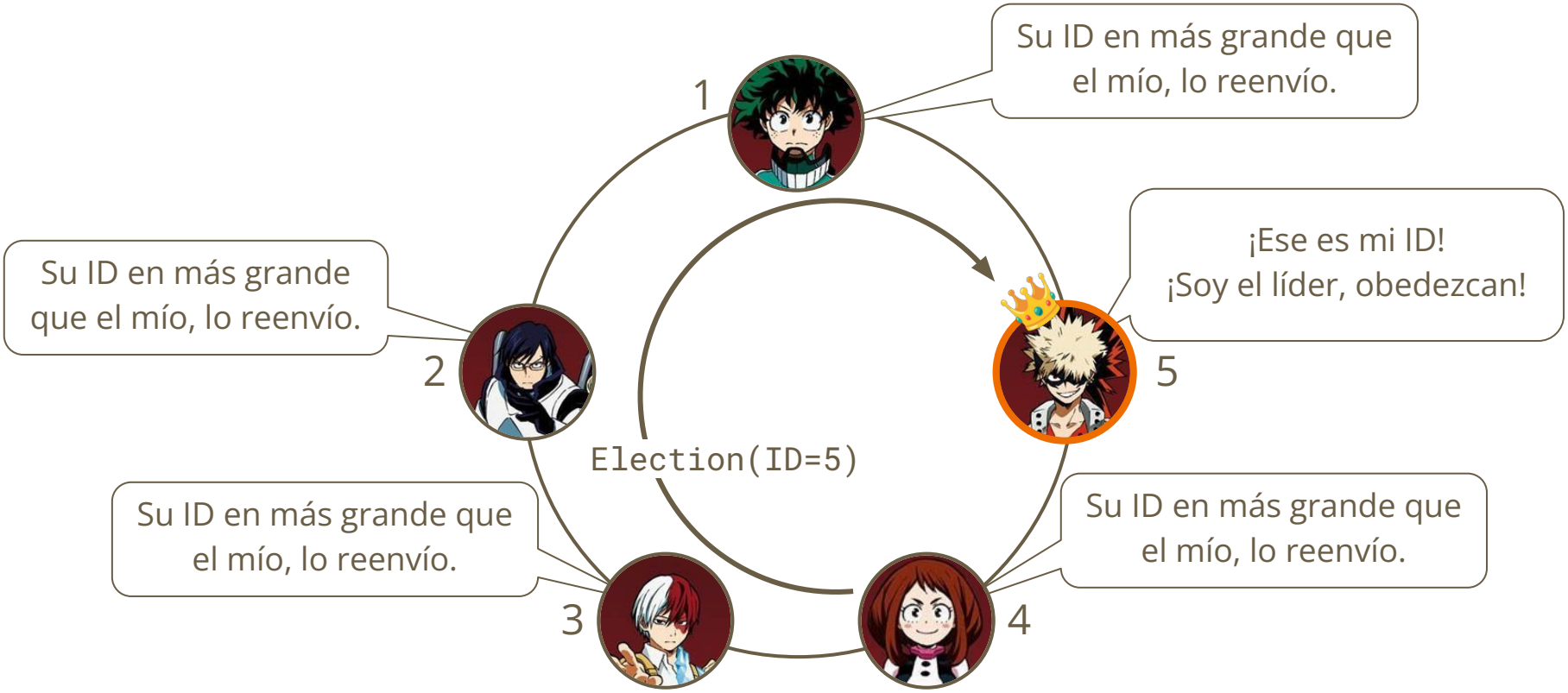
Elección de líder - Características

- Cada proceso P_x debe tener un identificador único que puede ser directamente su PID, algún valor dinámico, alguna propiedad, una combinación de valores, etc.
- Cada proceso P_x tiene una variable $elected_x$ que contiene el identificador del proceso elegido como líder.
- Se espera que todo algoritmo de elección de líder cumple con 2 propiedades:
  - ***Safery***: Un proceso P_x participante del algoritmo tiene $elected_x$ definido como "por definir" o como el ID del proceso que será el líder.
    - Nunca hay dos líderes válidos a la vez.
  - ***Liveness****:* Todos los procesos P_x participan y, finalmente, establecen su variable $elected_x$ o fallan.
    - Siempre habrá un líder elegido eventualmente.

Algoritmo *Bully*



Algoritmo *Ring-based*



Raft - Fase 1 - Elección de líder

Considerando la tupla (A_x, N) la acción X que se registró en el *log* durante el *term* N.

Caso 1 - El más actualizado comienza votación.

Voten por mi

[(A_1, 1), (A_2, 1), (A_3, 2)]
Term: 3
Timeout: 150

Tu último registro está en *term 2* vs el mio que es *term* 1.
¡Voto por ti!

[(A_1, 1), (A_2, 1)]
Term: 2
Timeout: 200

No tengo *logs*
¡Voto por ti!

[ ]
Term: 0
Timeout: 210

Tu último registro está en *term 2* vs el mio que es *term* 1.
¡Voto por ti!

[(A_1, 1), (A_2, 1)]
Term: 1
Timeout: 250

Raft - Fase 1 - Elección de líder

Considerando la tupla (A_x, N) la acción X que se registró en el *log* durante el *term* N.

Caso 3 - *Log* no tan actualizado

[(A_1, 1), (A_2, 1), (A_3, 2)]
Term: 2
Timeout: 220

[(A_1, 1), (A_2, 1)]
Term: 2
Timeout: 150

[ ]
Term: 0
Timeout: 210

[(A_1, 1), (A_2, 1)]
Term: 2
Timeout: 250

Raft - Fase 1 - Elección de líder

Considerando la tupla (A_x, N) la acción X que se registró en el *log* durante el *term* N.

Caso 3 - *Log* no tan actualizado

[(A_1, 1), (A_2, 1), (A_3, 2)]
Term: 2
Timeout: 220

[(A_1, 1), (A_2, 1)]
Term: 3
Timeout: 150

[ ]
Term: 0
Timeout: 210

[(A_1, 1), (A_2, 1)]
Term: 2
Timeout: 250

Raft - Fase 1 - Elección de líder

Considerando la tupla (A_x, N) la acción X que se registró en el *log* durante el *term* N.

Caso 3 - *Log* no tan actualizado

Tu último *log tiene un term* menor al mío.
¡Rechazado!

[(A_1, 1), (A_2, 1), (A_3, 2)]
Term: 2
Timeout: 220

Voten por mi

[(A_1, 1), (A_2, 1)]
Term: 3
Timeout: 150

No tengo *logs*
¡Voto por ti!

[ ]
Term: 0
Timeout: 210

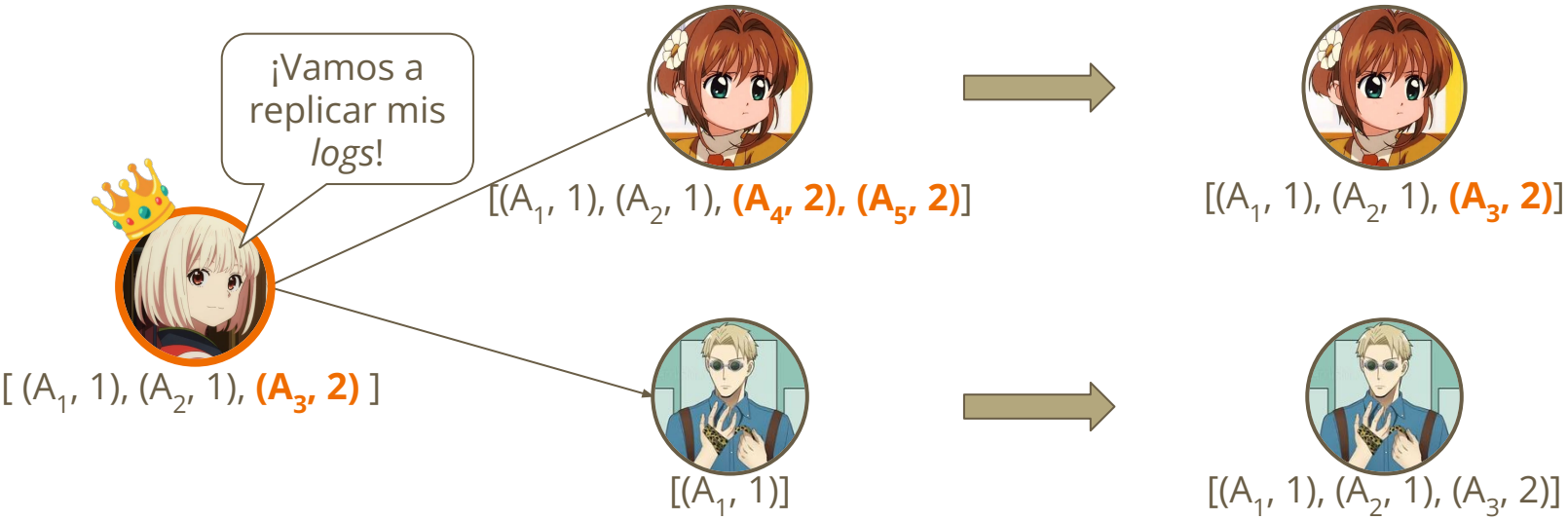
Nuestro último log es igual en *term*. El largo es el mismo.
¡Voto por ti!

[(A_1, 1), (A_2, 1)]
Term: 2
Timeout: 250

Raft - Fase 2 - Replicación de *logs*

Replicar

- Líder intenta replicar su *log* en seguidores.
- Si una posición del *log* tiene conflicto entre la tupla del líder y del seguidor, gana la tupla del líder.



Raft - Fase 2 - Replicación de *logs*

Consolidación (directa)

- Una acción/operación es consolidada solo si esta fue replicada **a la mayoría de los nodos en el mismo mandato que fue creada**.

Consolidación (indirecta)

- Una acción/operación es consolidada indirectamente si existe una operación del *log* que fue consolidada **directamente**.

Exclusión Mutua

Algoritmos de Exclusión Mutua Distribuida

- Existen 3 requisitos que se buscan de los algoritmos de exclusión mutua:
  - ***Safety***: Como máximo un solo nodo accede al recurso crítico a la vez.
  - ***Liveness***: Las solicitudes para entrar y salir de la sección crítica finalmente tienen éxito.
    - Implica la ausencia de *deadlock* y de inanición.
    - Inanición es el aplazamiento indefinido de la entrada de un nodo que la ha solicitado.
  - ***Ordering***: Si una solicitud para ingresar al recurso crítico ocurrió antes de otra, entonces el ingreso al recurso crítico se concede en ese orden.

Algoritmo 1 - Servidor Central

Nodo K - Rol Servidor

Cola de peticiones

Fin [G H] Inicio

2. Quiero el recurso crítico

1. Quiero el recurso crítico

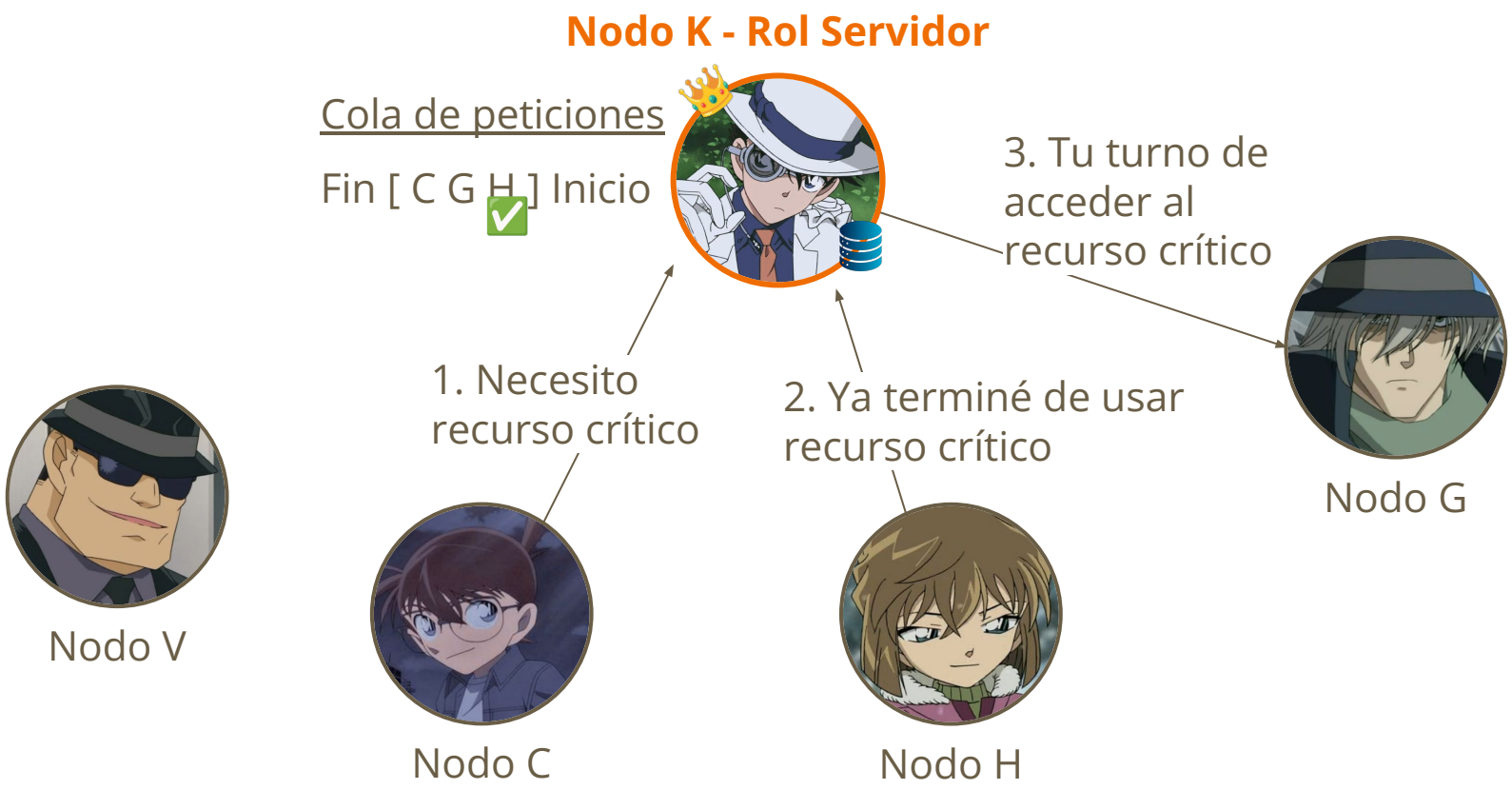
Nodo G

Nodo V

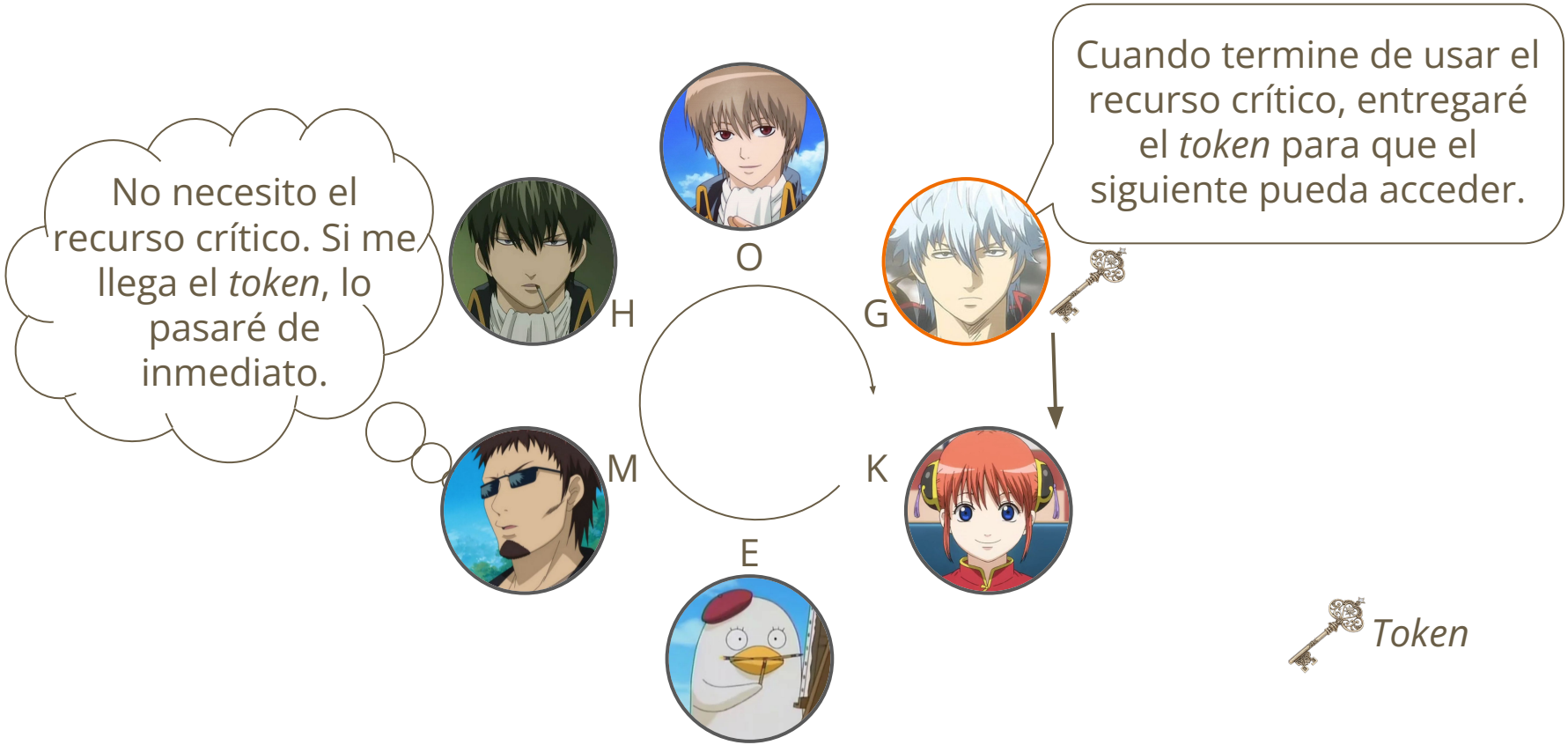
Nodo C

Nodo H

Algoritmo 1 - Servidor Central

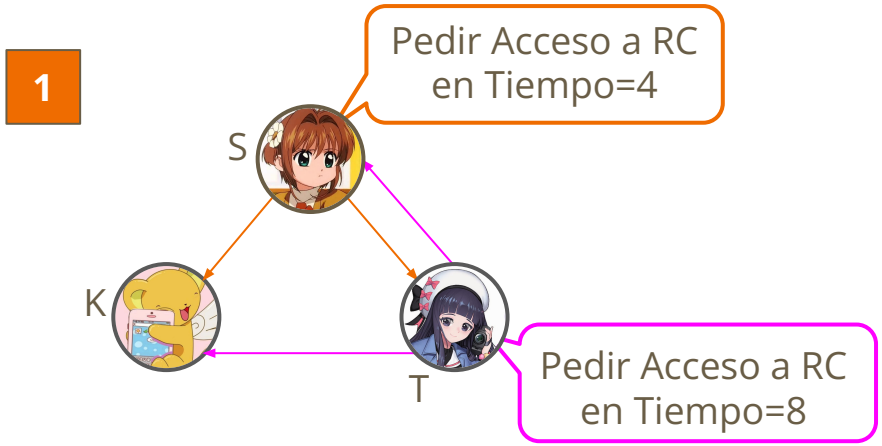


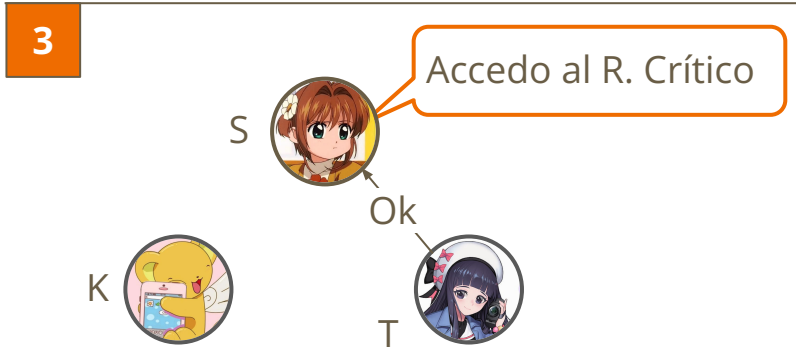
Algoritmo 2 - *Token-Ring*



Algoritmo 3 - *Multicast* de Ricart & Agrawala

RC = Recurso Crítico





Próximos eventos

Próxima clase (de vuelta de semana de receso)

- Nueva unidad: Fiabilidad en Sistemas Distribuidos
- En particular, profundizaremos más en lo que es tolerancia a fallos.
  - ¿Qué tipos de fallas existen?
  - ¿Qué otras técnicas hay para sobrellevar o detectar fallos?

Evaluación

- Mañana finaliza el plazo del control.
- Sábado hay prueba... y en términos de salas... todavía no hay sala.

IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 10)