

---

# IIC2523

# Sistemas Distribuidos

— Hernán F. Valdivieso López —  
(2025 - 2 / Clase 08)

---

# Elección de Líder

# Temas de la clase

1. Algoritmos de elección de líder.
2. Algoritmo *Bully*.
3. Algoritmo *Raft*: consenso de información a través de un líder.
4. *Ring-based Election*.

# Algoritmos de elección de líder

---

# Elección de líder

- ◆ Consiste en elegir un nodo/proceso único para desempeñar un **rol particular** en un sistema distribuido:
  - ◆ Coordinador.
  - ◆ Iniciador de algún protocolo/algoritmo.
  - ◆ Ser el servidor para sincronizar relojes.
  - ◆ Acceder a un recurso crítico.
  - ◆ Entre otros.

# 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  $electd_x$  que contiene el identificador del proceso elegido como líder.

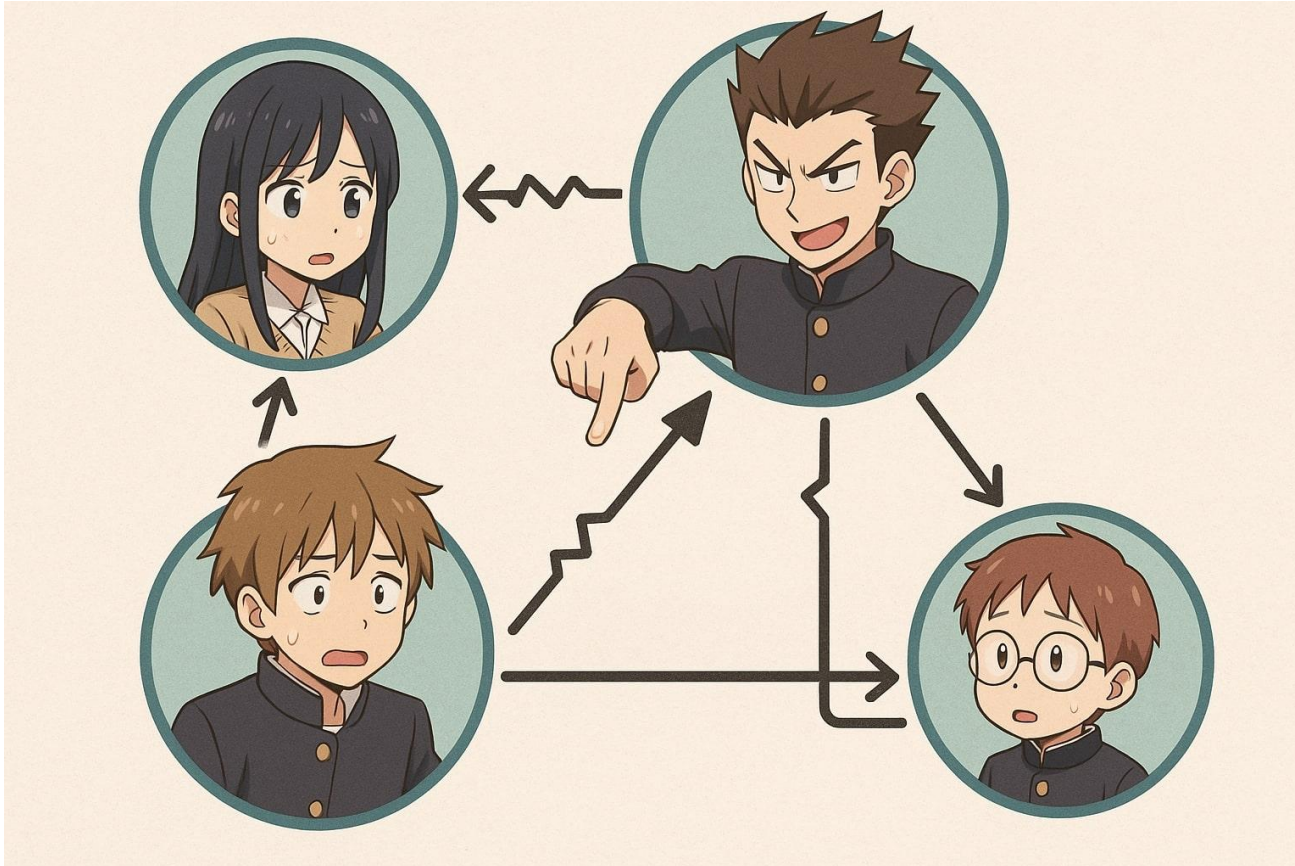
# Elección de líder - Características

- ◆ 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  $electd_x$  que contiene el identificador del proceso elegido como líder.
- ◆ Se espera que todo algoritmo de elección de líder cumple con 2 propiedades:
  - ◆ **Safety:** Un proceso  $P_x$  participante del algoritmo tiene  $electd_x$  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  $electd_x$  o fallan.
    - ◆ Siempre habrá un líder elegido eventualmente.

# Algoritmos *Bully*



# Elección de líder - Algoritmo *Bully*



# Elección de líder - Algoritmo *Bully*

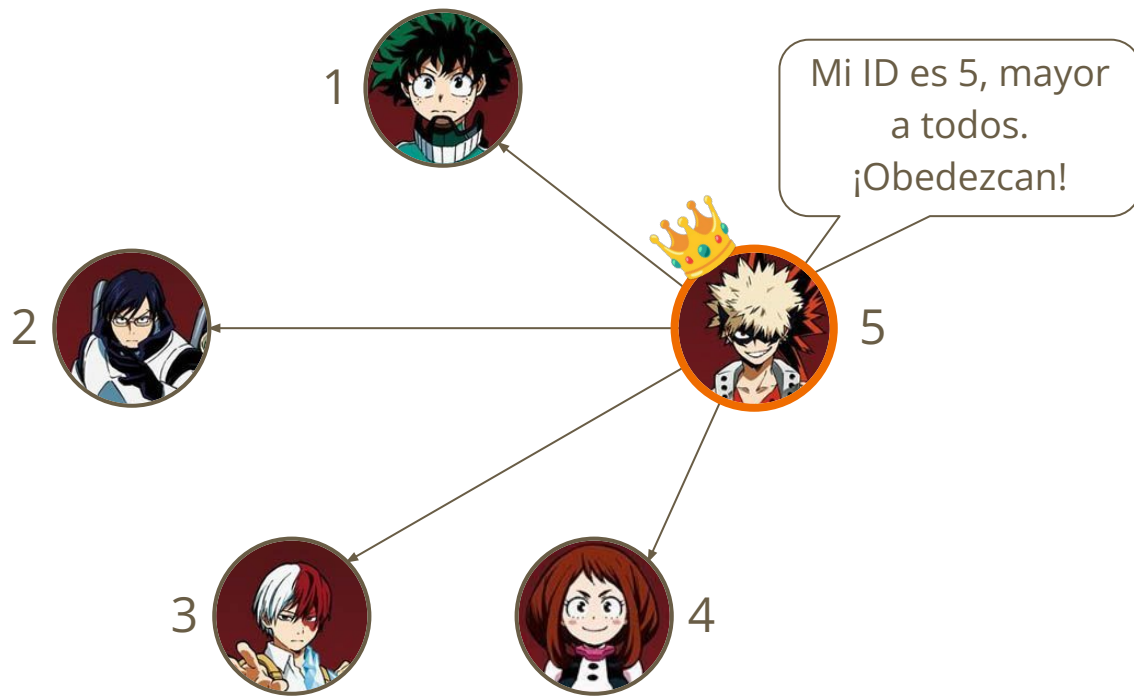
- ◆ Propuesto por Garcia-Molina en 1982.
- ◆ Es un algoritmo síncrono que utiliza tiempos de espera (*timeouts*) para detectar fallas en los nodos.
  - ◆ Se establece *delay* máximo de transmisión de mensajes ( $T_{trans}$ )
  - ◆ Se establece *delay* máximo de procesamiento de mensajes ( $T_{process}$ )
  - ◆ Si un nodo demora más de  $2 \times T_{trans} + T_{process}$  en responder, se asume una falla.
- ◆ Supuestos:
  - ◆ El identificador de cada nodo es un valor fijo que nunca cambia.
  - ◆ Cada nodo sabe qué nodo tienen identificadores más altos y puede comunicarse con todos ellos.

# Elección de líder - Algoritmo *Bully* - Funcionamiento

- ◆ Este algoritmo utiliza tres tipos de mensajes:
  - ◆ **Election**: Anunciar una elección.
  - ◆ **Ok**: Respuesta a un mensaje Election para detener la candidatura.
  - ◆ **Coordinator**: Enviado para anunciar la identidad del proceso elegido.

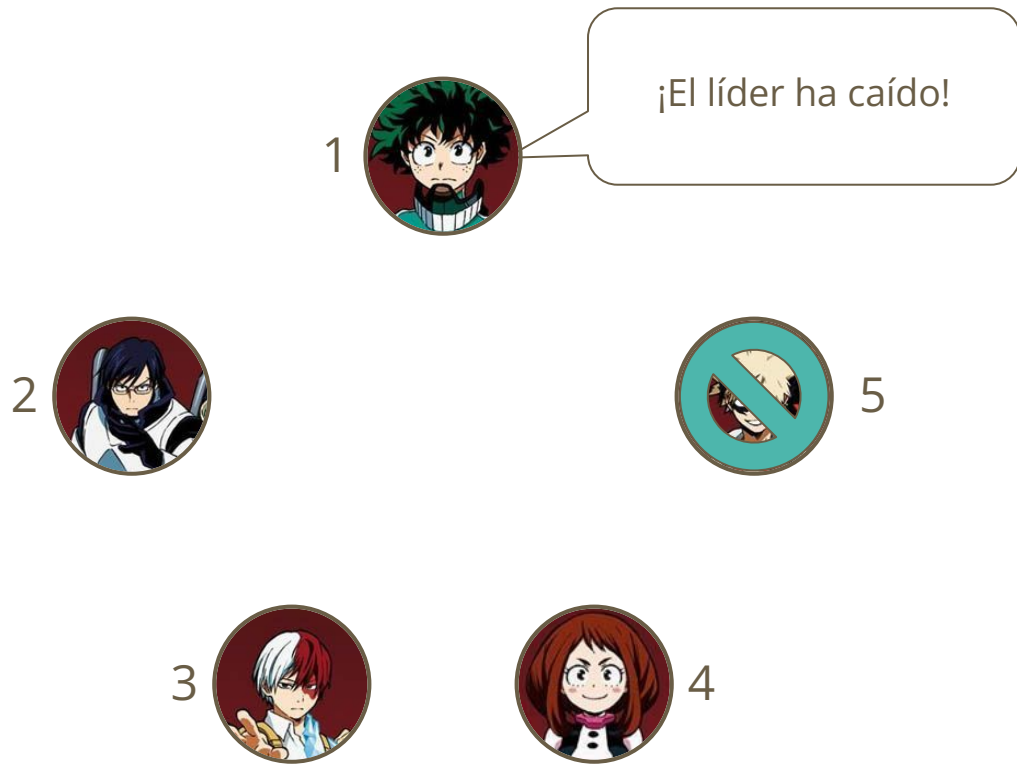
# Elección de líder - Algoritmo *Bully* - Funcionamiento

- ◆ **Caso inicial:** como todos saben el identificador de cada proceso. Aquel con mayor identificador manda mensaje Coordinator para avisar que será el líder



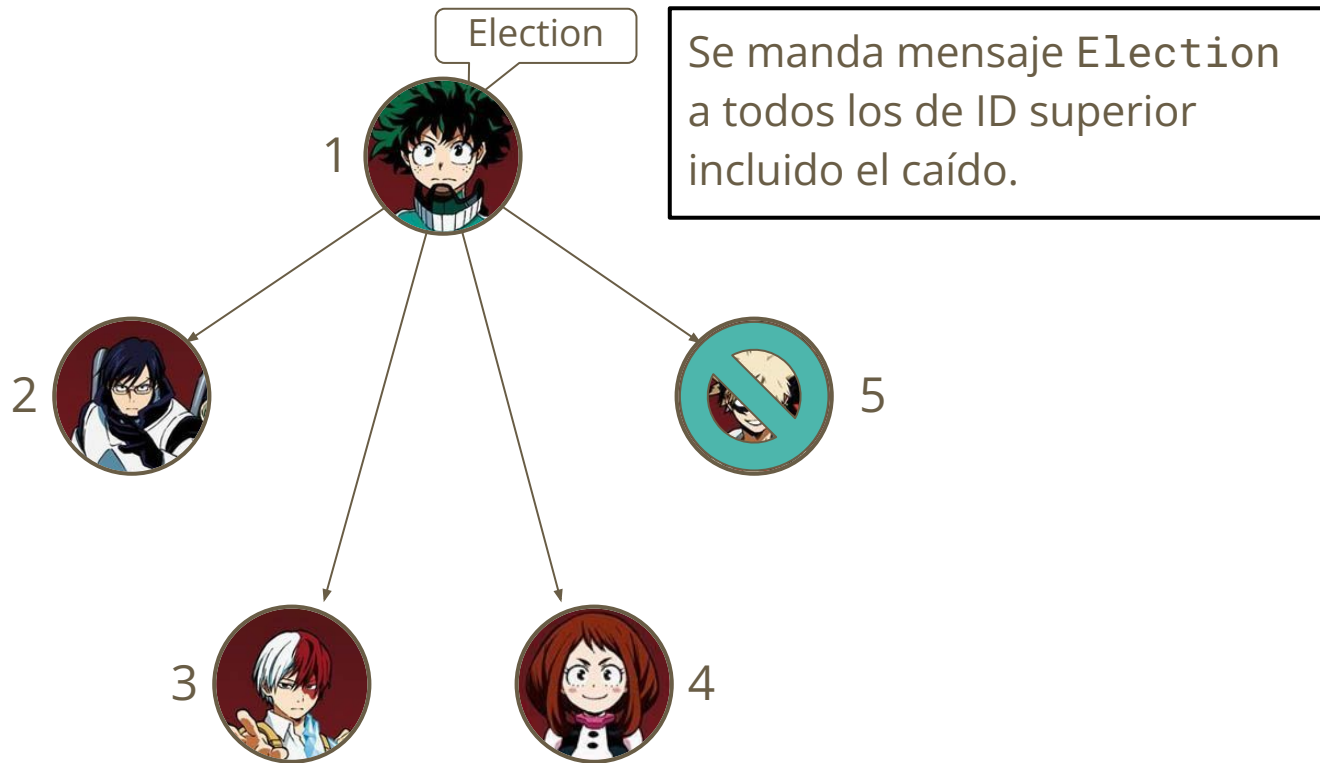
# Elección de líder - Algoritmo *Bully* - Funcionamiento

- ◆ **Caso el líder falla/muere.** Algún nodo se da cuenta de esto y comienza elección.



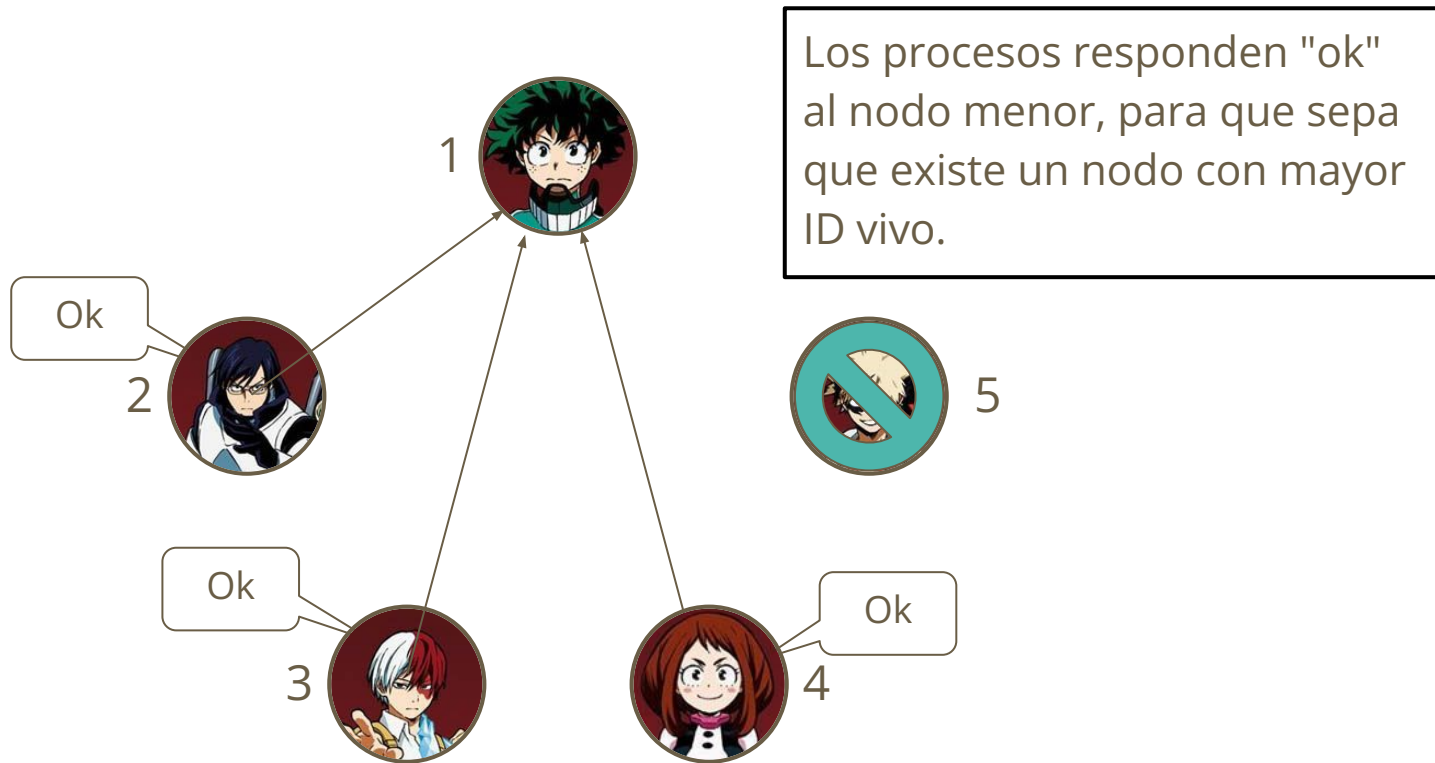
# Elección de líder - Algoritmo *Bully* - Funcionamiento

- ◆ **Caso el líder falla/muere.** Algún nodo se da cuenta de esto y comienza elección.



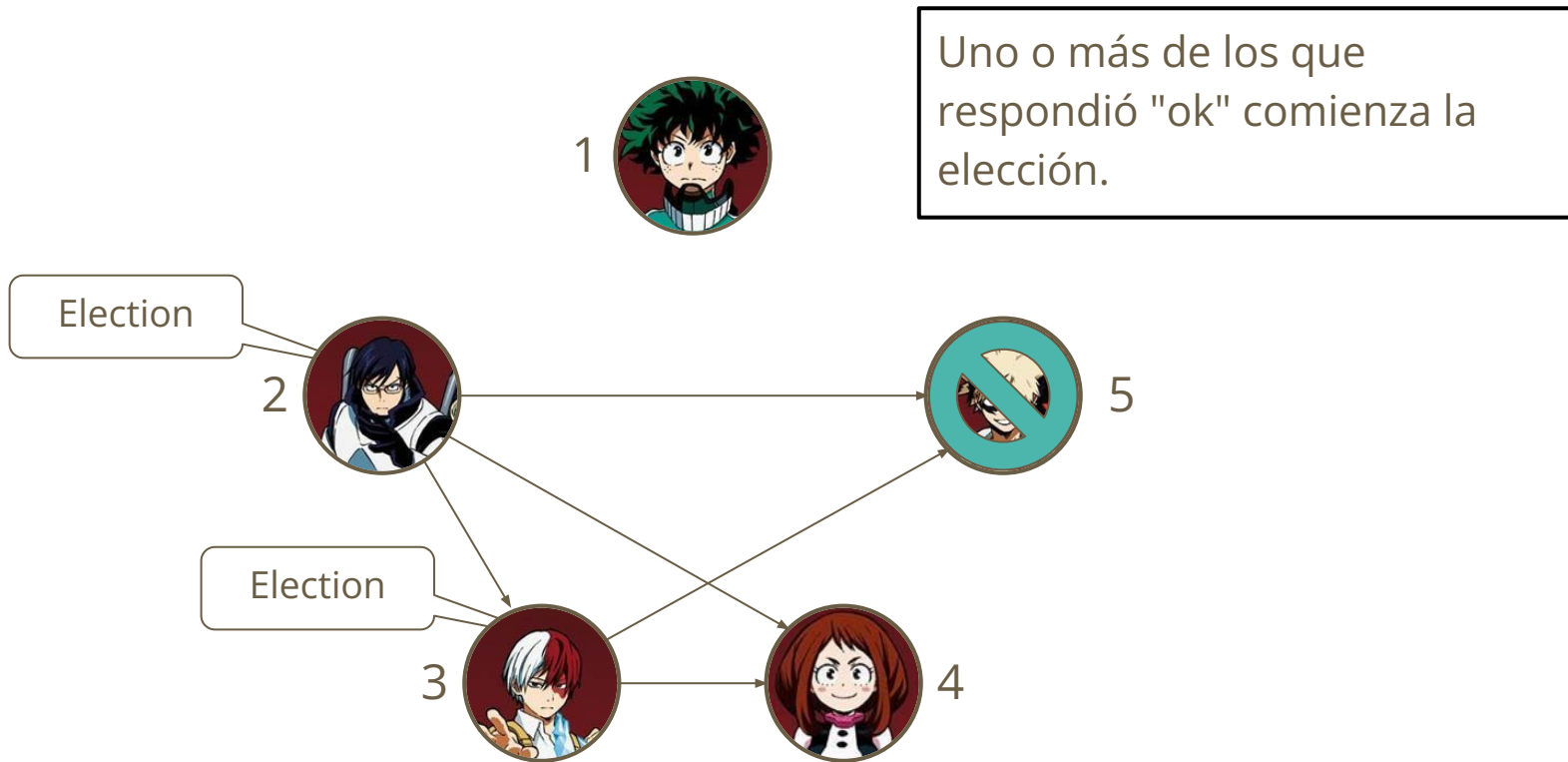
# Elección de líder - Algoritmo *Bully* - Funcionamiento

- ◆ **Caso el líder falla/muere.** Algún nodo se da cuenta de esto y comienza elección.



# Elección de líder - Algoritmo *Bully* - Funcionamiento

- ◆ **Caso el líder falla/muere.** Algún nodo se da cuenta de esto y comienza elección.





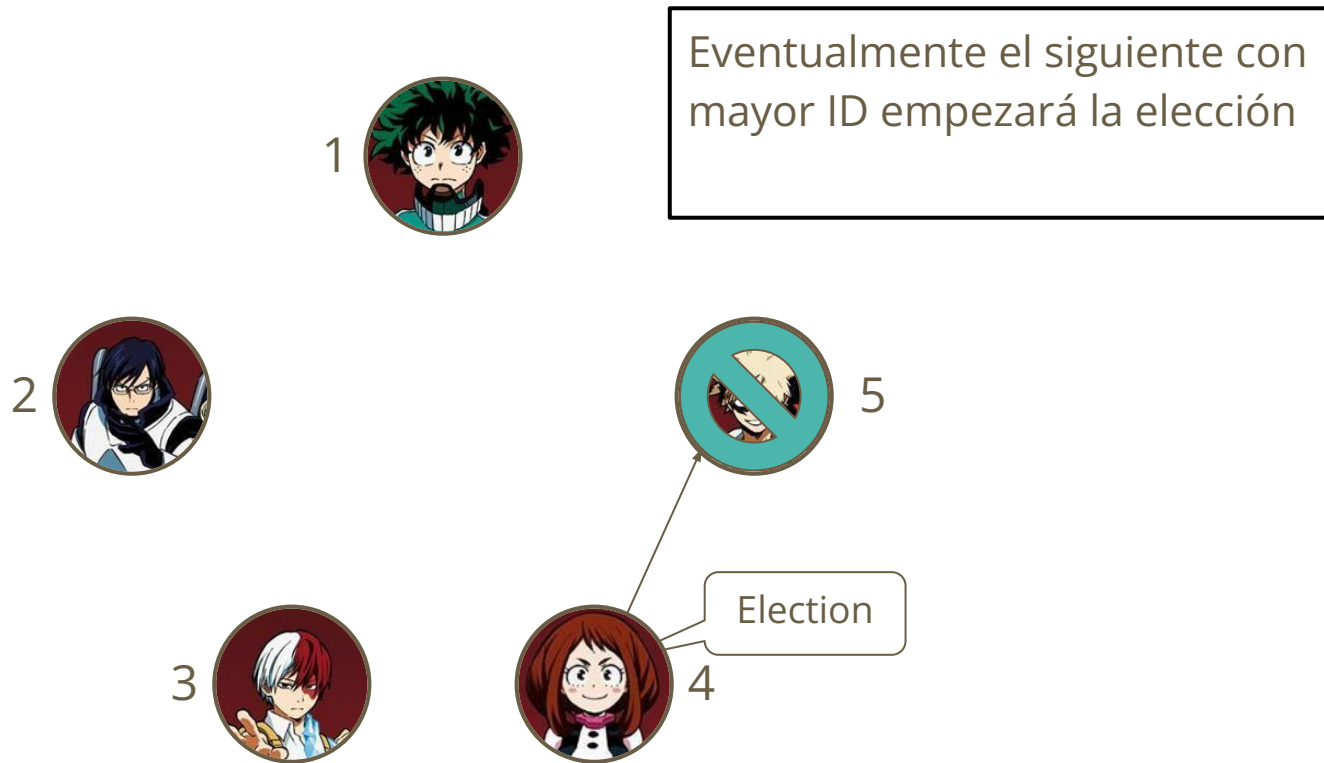
# Elección de líder - Algoritmo *Bully* - Funcionamiento

- ◆ **Caso el líder falla/muere.** Algún nodo se da cuenta de esto y comienza elección.



# Elección de líder - Algoritmo *Bully* - Funcionamiento

- ◆ **Caso el líder falla/muere.** Algún nodo se da cuenta de esto y comienza elección.



# Elección de líder - Algoritmo *Bully* - Funcionamiento

◆ **Caso el líder falla/muere.** Algún nodo se da cuenta de esto y comienza elección.



# Elección de líder - Algoritmo *Bully* - Resumen

- ◆ Proceso con mayor identificador es líder siempre.
- ◆ Si líder se cae, algún proceso comienza la elección. Ese proceso contacta **exclusivamente** a todos con ID superior.
  - ◆ Si al menos uno con ID superior responde (Ok), entonces el candidato se retira de la elección. El proceso que respondió comienza la elección eventualmente.
  - ◆ Si ningún proceso con ID superior responde, entonces el que comenzó la elección está capacitado de ser líder y avisa a todos los procesos activos.

# Elección de líder - Algoritmo *Bully* - Observaciones

- ◆ Es un algoritmo que, en el peor caso, se mandan demasiados mensajes.
  - ◆ El nodo con menor id manda  $N-1$  mensajes de Election.
  - ◆ El siguiente manda  $N-2$  mensajes de Election.
  - ◆ El siguiente manda  $N-3$  mensajes de Election.
  - ◆ Hasta llegar al segundo mejor proceso que manda 1 mensaje de Election.

# Elección de líder - Algoritmo *Bully* - Observaciones

- ◆ Es un algoritmo que, en el peor caso, se mandan demasiados mensajes.
  - ◆ El nodo con menor id manda  $N-1$  mensajes de Election.
  - ◆ El siguiente manda  $N-2$  mensajes de Election.
  - ◆ El siguiente manda  $N-3$  mensajes de Election.
  - ◆ Hasta llegar al segundo mejor proceso que manda 1 mensaje de Election.
- ◆ En una red distribuida, no es garantía que siempre se sepa todos los nodos participantes y sus ID, por lo cual puede ser necesario enviar Election a muchos más nodos (incluso con ID menor) para tener mayor seguridad ser el con mayor ID.
  - ◆ Si un nodo con ID mayor envía Election a un nodo con ID menor, simplemente el con ID menor no responde.

# Raft

Consenso de datos a través de  
la elección de un líder  
democráticamente

---

# Raft - Introducción

- ◆ Propuesto el 2014 por Diego Ongaro y John Ousterhout.
- ◆ Trabajo original: [In Search of an Understandable Consensus Algorithm](#)
- ◆ Citando su paper:

*Raft is a consensus algorithm for managing a replicated log. It produces a result equivalent to (multi-)Paxos, and it is as efficient as Paxos, but its structure is different from Paxos; this makes Raft more understandable than Paxos...*



# Raft - Estados y fases

◆ Se proponen 3 estados:

- ◆ **Seguidor (*Follower*)**: estado por defecto, solo responde a peticiones.
- ◆ **Candidato (*candidate*)**: busca votos para ser el líder.
- ◆ **Líder (*leader*)**: encargado de replicar las acciones para lograr consolidar.

◆ Hay 2 fases:

- ◆ Elección de líder.
- ◆ Replicación de *logs*.

# Raft - Estados y fases

◆ Se proponen 3 estados:

- ◆ **Seguidor (*Follower*)**: estado por defecto, solo responde a peticiones.
- ◆ **Candidato (*candidate*)**: busca votos para ser el líder.
- ◆ **Líder (*leader*)**: encargado de replicar las acciones para lograr consolidar.

◆ Hay 2 fases:

- ◆ Elección de líder.
- ◆ Replicación de *logs*.

◆ Hay un término que usaremos mucho: "***term***" que lo veremos como el "periodo de mandato" de un líder. Es un número que solo va creciendo.

◆ Todo nodo tiene un registro del *term* que lo compartirá con los demás nodos. El *term* más grande manda en todo momento.

# Raft - Fase 1 - Elección de líder

- ◆ Cada nodo tiene un *election timeout* que es un número aleatorio entre 150 a 300 milisegundos. Es un intento de asegurar un número distinto por nodo.
- ◆ **Postulación:** cuando se acaba el *election timeout*, el nodo se postula como candidato.
  - a. Aumenta el *"term/mandato"*.
  - b. El nodo vota por sí mismo.
  - c. Le pide a todos los demás nodos que voten por él.
  - d. Si logra mayoría, se vuelve líder.
  - e. Si no logra mayoría, espera reiniciar su *election timeout* y vuelve a paso A.

# Raft - Fase 1 - Elección de líder

◆ **Votación:** todo nodo que reciba solicitud de voto, debe responder:

1. Si ya votó en el *term* actual o el *term* del candidato es menor al *term* del votante, **vota rechazo**.
2. Si el nodo votante no tiene *logs* todavía, **vota a favor**.
3. En otro caso, compara los *logs* (registro de acciones) del nodo votante y del candidato.
  - a. Si el último *log* del candidato fue en un *term* mayor al nodo votante, **vota a favor del candidato**.
  - b. Si el último *log* del candidato fue en un *term* igual al nodo votante, **solo se acepta si su *log* es igual o más largo que el nodo votante**.
  - c. Si el último *log* del candidato fue en un *term* menos al nodo votante, **vota rechazo**.

# 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 - [estado inicial]



$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 2

*Timeout*: 150



$[(A_1, 1), (A_2, 1)]$

*Term*: 2

*Timeout*: 200



$[]$

*Term*: 0

*Timeout*: 210



$[(A_1, 1), (A_2, 1)]$

*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 1 - El más actualizado comienza votación.



Voten por mi

$[(A_1, 1), (A_2, 1), (A_3, 2)]$

Term: 3

Timeout: 150



$[(A_1, 1), (A_2, 1)]$

Term: 2

Timeout: 200



$[]$

Term: 0

Timeout: 210



$[(A_1, 1), (A_2, 1)]$

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 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 2 - [estado inicial]



$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 2

*Timeout*: 200



$[(A_1, 1), (A_2, 1)]$

*Term*: 2

*Timeout*: 210



$[]$

*Term*: 0

*Timeout*: 150



$[(A_1, 1), (A_2, 1)]$

*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 2 - El menos actualizado comienza votación.



$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 2

*Timeout*: 200



$[(A_1, 1), (A_2, 1)]$

*Term*: 2

*Timeout*: 210



Voten por mi

$[]$

*Term*: 1

*Timeout*: 150



$[(A_1, 1), (A_2, 1)]$

*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 2 - El menos actualizado comienza votación.



Tu *term* actual es menor al mío.  
**¡Rechazado!**

$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 2

*Timeout*: 200



Tu *term* actual es menor al mío.  
**¡Rechazado!**

$[(A_1, 1), (A_2, 1)]$

*Term*: 2

*Timeout*: 210



Voten por mi

$[]$

*Term*: 1

*Timeout*: 150



Tu *term* actual es menor al mío.  
**¡Rechazado!**

$[(A_1, 1), (A_2, 1)]$

*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 - [estado inicial]



$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 2

*Timeout*: 220



$[(A_1, 1), (A_2, 1)]$

*Term*: 1

*Timeout*: 150



$[]$

*Term*: 0

*Timeout*: 210



$[(A_1, 1), (A_2, 1)]$

*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 - Solo *Term* actualizado.



$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 2

*Timeout*: 220



Voten por mi

$[(A_1, 1), (A_2, 1)]$

*Term*: 2

*Timeout*: 150



$[]$

*Term*: 0

*Timeout*: 210



$[(A_1, 1), (A_2, 1)]$

*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 - Solo *Term* 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*: 2

*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*: 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 4 - [estado inicial]



$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 3

*Timeout*: 220



$[(A_1, 1), (A_2, 1)]$

*Term*: 1

*Timeout*: 250



$[\ ]$

*Term*: 0

*Timeout*: 210



$[(A_1, 1), (A_2, 1)]$

*Term*: 1

*Timeout*: 150

# 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 4 - *Term* y *logs* no tan actualizado.



$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 3

*Timeout*: 220



$[(A_1, 1), (A_2, 1)]$

*Term*: 1

*Timeout*: 250



$[]$

*Term*: 0

*Timeout*: 210



Voten por mi

$[(A_1, 1), (A_2, 1)]$

*Term*: 2

*Timeout*: 150

# 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 4 - *Term* y *logs* no tan actualizado.



Tu *term* actual es  
menor al mío.  
**¡Rechazado!**

$[(A_1, 1), (A_2, 1), (A_3, 2)]$

*Term*: 2

*Timeout*: 220



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

$[(A_1, 1), (A_2, 1)]$

*Term*: 1

*Timeout*: 250



No tengo *logs*  
**¡Voto por ti!**

$[]$

*Term*: 0

*Timeout*: 210



Voten por mi

$[(A_1, 1), (A_2, 1)]$

*Term*: 2

*Timeout*: 150



# Raft - Fase 1 - Elección de líder (Resumen)

- ◆ Todos parten como *seguidores* y votan una vez por *term*/mandato.
- ◆ **Postulación:** se acaba el *election timeout* de un nodo y se convierte en candidato.
- ◆ **Votación:** solo se rechaza si el nodo candidato está menos actualizado que el nodo votante. En caso de empate, se acepta al nodo candidato.
- ◆ **Elección de líder:** el candidato logra tener mayoría de votos.



# Raft - Fase 1 - Elección de líder (Peeero)

Siendo el líder...

🤔 ¿Cómo evito perder el trono, es decir, cómo evito que empiece otra votación? 🤔

**Heartbeat:** el líder manda constantemente un mensaje a todos los seguidores para reiniciar su *election timeout*.

# Raft - Fase 1 - Elección de líder (Peeero)

Siendo el líder...

🤔 ¿Cómo evito perder el trono, es decir, cómo evito que empiece otra votación? 🤔

**Heartbeat:** el líder manda constantemente un mensaje a todos los seguidores para reiniciar su *election timeout*.

🤔 Y si me desconecto y vuelvo... ¿cómo me doy cuenta que empezó una votación? 🤔

Cuando se comienza una votación, el *term* aumenta. Cualquier nodo que ve un *term* mayor al suyo se vuelve seguidor de inmediato y actualiza su *term* al valor más grande recibido.

# Raft - Fase 2 - Replicación de *logs*

## *Logs*

- ◆ Lista de tuplas. Cada tupla tiene la operación y el *term* en que se creó la operación.

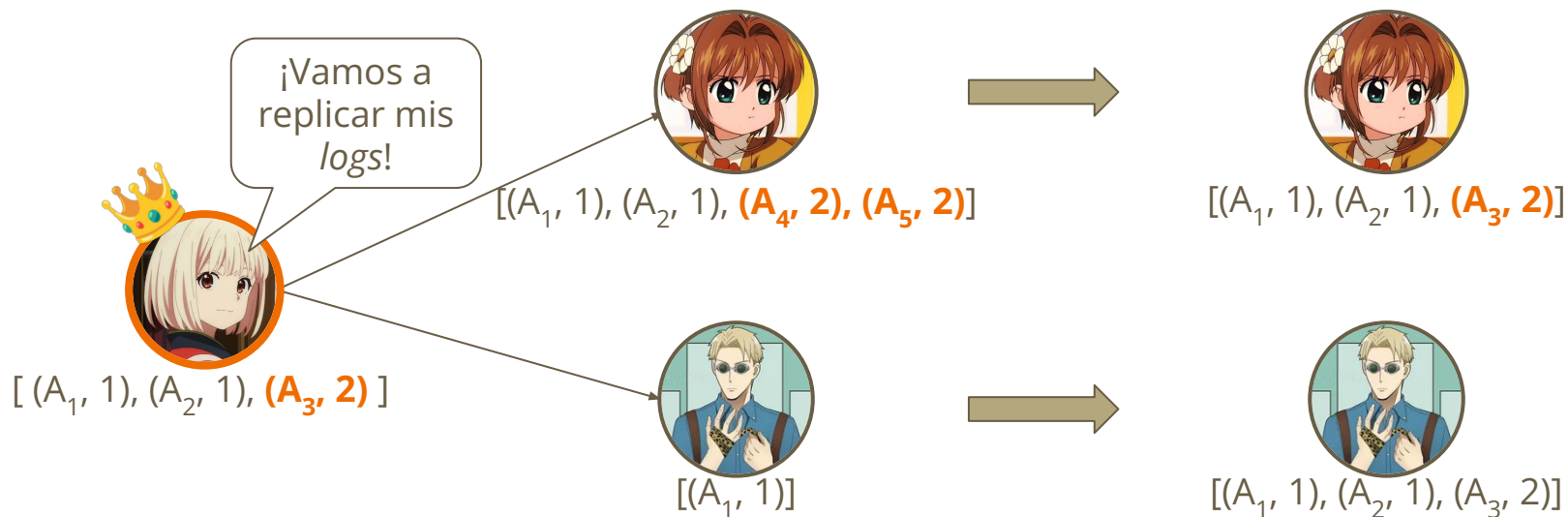
## Replicación

- ◆ El líder recibe operaciones, la guarda en su *log*.
- ◆ Si un seguidor recibe una operación, la redirige al líder.
- ◆ 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*

## 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.

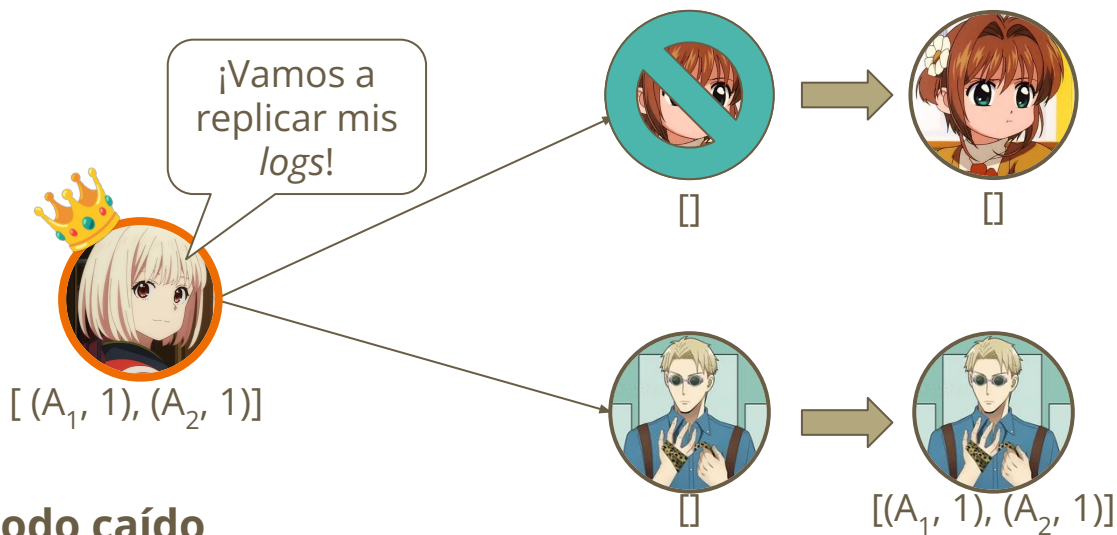


# Raft - Fase 2 - Replicación de *logs*

## 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.**

Term 1



Las acciones  $A_1$  y  $A_2$  serán consolidadas.

Las tiene el líder y un seguidor (2 de nodos de 3),







 **Nodo caído**

# 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.**

Term 1

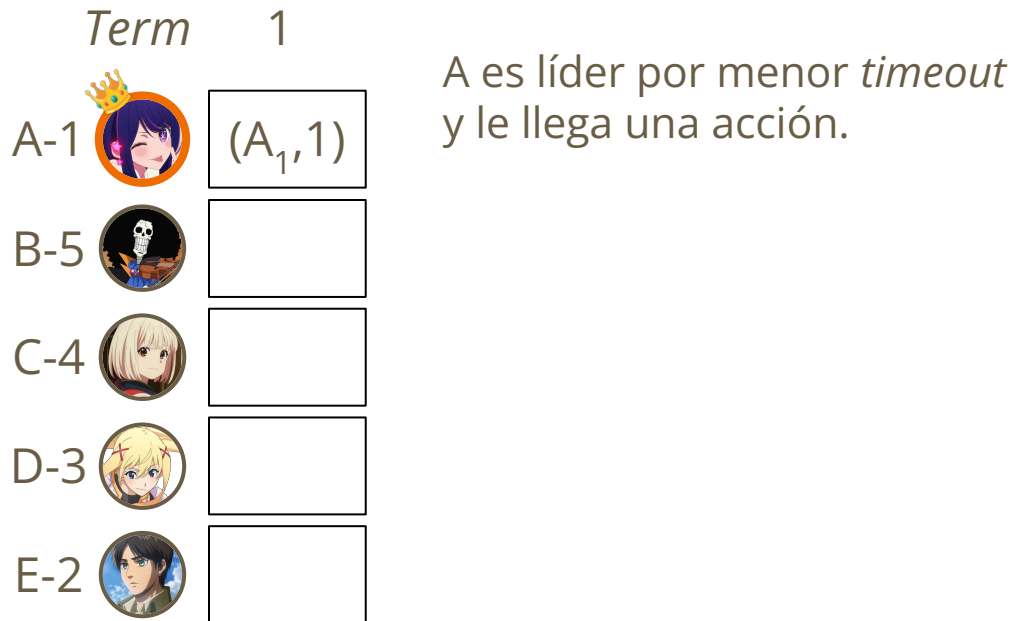
A-1		<input type="text"/>
B-5		<input type="text"/>
C-4		<input type="text"/>
D-3		<input type="text"/>
E-2		<input type="text"/>

Cada nodo tiene un ID (la letra) y un *election timeout* distinto (el número después de la letra)

# 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.**

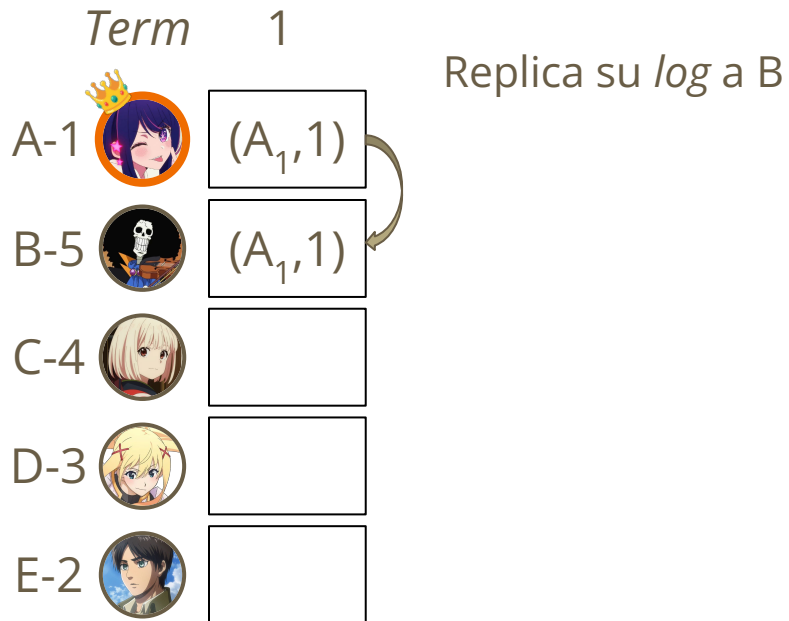




# Raft - Fase 2 - Replicación de *logs*

## 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.**








# 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.**



Term	1	2
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 		
D-3 		
E-2 		






A se cae temporalmente.  
E, segundo con menor *timeout*, logra mayoría gracias al voto de C y E.

# 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.**



Term	1	2
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 		
D-3 		
E-2 		(A <sub>2</sub> ,2)






Llega acción a E.

# 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.**



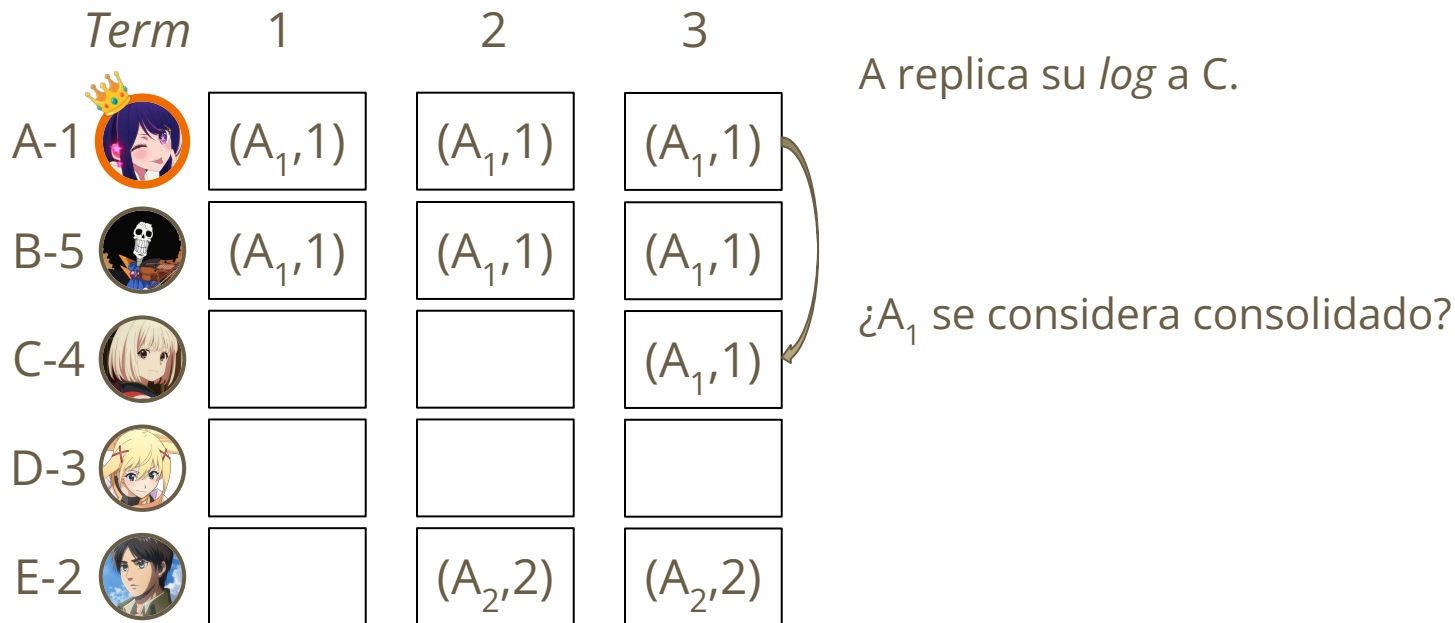
Term	1	2	3
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			
D-3 			
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)

E se cae temporalmente.  
A recupera liderazgo por  
menor *timeout* y su  
último log es más  
actualizado que C y D.

# 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.**



# 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.**

<i>Term</i>	1	2	3
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			(A <sub>1</sub> ,1)
D-3 			
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)






¿A<sub>1</sub> se considera consolidado?

**No, estamos en *term* 3, y esta acción fue creada en *term* 1.**

# 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.**

<i>Term</i>	1	2	3
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			(A <sub>1</sub> ,1)
D-3 			
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)

¿A<sub>1</sub> se considera consolidado?






**No, estamos en *term* 3, y esta acción fue creada en *term* 1.**

**Incluso... puede pasar lo siguiente**

# 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.**

Term	1	2	3	4
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
D-3 				
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)



## Nodo caído






A se cae por un buen tiempo.  
E se vuelve líder porque tiene menor *timeout* y su último *log* tiene el *term* más actual.



# 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.**

Term	1	2	3	4
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>2</sub> ,2)
C-4 			(A <sub>1</sub> ,1)	(A <sub>2</sub> ,2)
D-3 				(A <sub>2</sub> ,2)
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)



Nodo caído

E replica su *logs* a todos los nodos activos

Ahora A<sub>1</sub> ni siquiera está en la mayoría.

Pero A<sub>2</sub> ahora sí está en mayoría, debería ser consolidado...

# 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.**

	Term	1	2	3	4
A-1		(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5		(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>2</sub> ,2)
C-4				(A <sub>1</sub> ,1)	(A <sub>2</sub> ,2)
D-3					(A <sub>2</sub> ,2)
E-2			(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)



Nodo caído

E replica su *logs* a todos los nodos activos

Ahora A<sub>1</sub> ni siquiera está en la mayoría.






Pero A<sub>2</sub> ahora sí está en mayoría, debería ser consolidado... No

# 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.**



Term	1	2	3
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			
D-3 			
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)

Volvamos a esta situación

E se cae temporalmente.  
A recupera liderazgo por menor *timeout* y su último log es más actualizado que C y D.

# 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.**



Nodo caído






Llega una acción a A.

Term	1	2	3
A-1 	$(A_1, 1)$	$(A_1, 1)$	$(A_1, 1)(A_3, 3)$
B-5 	$(A_1, 1)$	$(A_1, 1)$	$(A_1, 1)$
C-4 			$(A_1, 1)$
D-3 			
E-2 		$(A_2, 2)$	$(A_2, 2)$

# 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.**

<i>Term</i>	1	2	3	4
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
D-3 				
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)



## Nodo caído






A se cae por un buen tiempo.

E se vuelve líder porque tiene menor *timeout* y su último *log* tiene el *term* más actual.

# 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.**

Term	1	2	3	4
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>2</sub> ,2)
C-4 			(A <sub>1</sub> ,1)	(A <sub>2</sub> ,2)
D-3 				(A <sub>2</sub> ,2)
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)








**Nodo caído**

E replica su *logs* a todos los nodos activos.

# 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.**

Term	1	2	3	4
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>2</sub> ,2)
C-4 			(A <sub>1</sub> ,1)	(A <sub>2</sub> ,2)
D-3 				(A <sub>2</sub> ,2)
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)



## Nodo caído

Si A vuelve y E se cae

El último *log* de A tiene un *term* mayor a todos.  
A es líder y puede hacer que A<sub>2</sub> se sobrescriba.

# 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.**



# 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.**

# Raft - Fase 2 - Replicación de *logs*

## Consolidación (indirecta)

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






Term	1	2	3
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			(A <sub>1</sub> ,1)
D-3 			
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)

Volvamos a esta situación del ejemplo anterior.

# Raft - Fase 2 - Replicación de *logs*

## Consolidación (indirecta)

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

Term	1	2	3
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			(A <sub>1</sub> ,1)
D-3 			
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)






Volvamos a esta situación del ejemplo anterior.

A va a replicar su *logs* a C y D.

# Raft - Fase 2 - Replicación de *logs*

## Consolidación (indirecta)

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

Term	1	2	3
A-1 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)
B-5 	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)	(A <sub>1</sub> ,1)
C-4 			(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)
D-3 			(A <sub>1</sub> ,1)( A <sub>3</sub> , 3)
E-2 		(A <sub>2</sub> ,2)	(A <sub>2</sub> ,2)

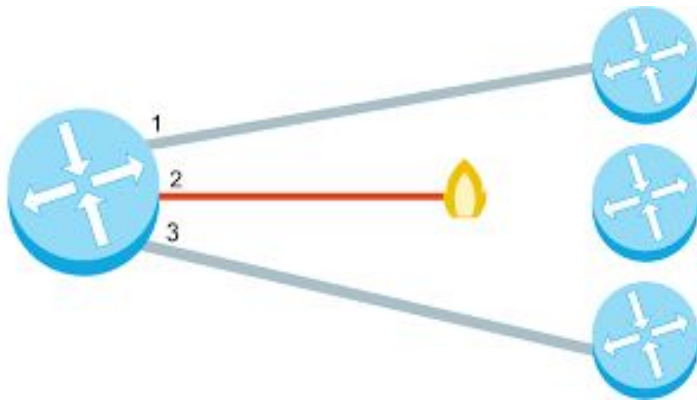
A<sub>3</sub> fue consolidada directamente. Por lo tanto, A<sub>1</sub> fue consolidada indirectamente.

Mientras exista mayoría de nodos (requisito del algoritmo), el líder será A, C o D que tienen A<sub>1</sub> y A<sub>3</sub>.

# Raft - Garantías (las mismas que Paxos)

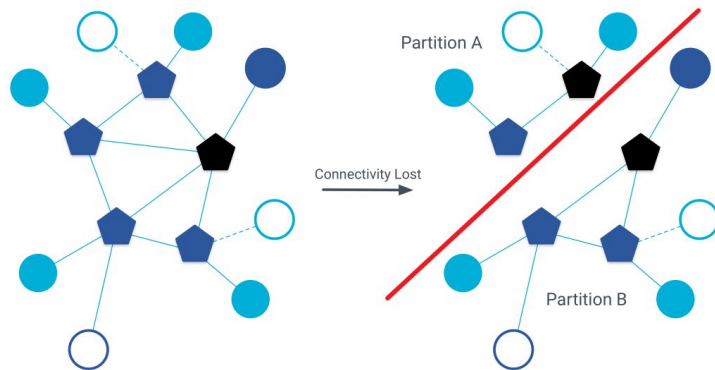
## ◆ Tolerancia a fallos

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.



# Raft - Vida real

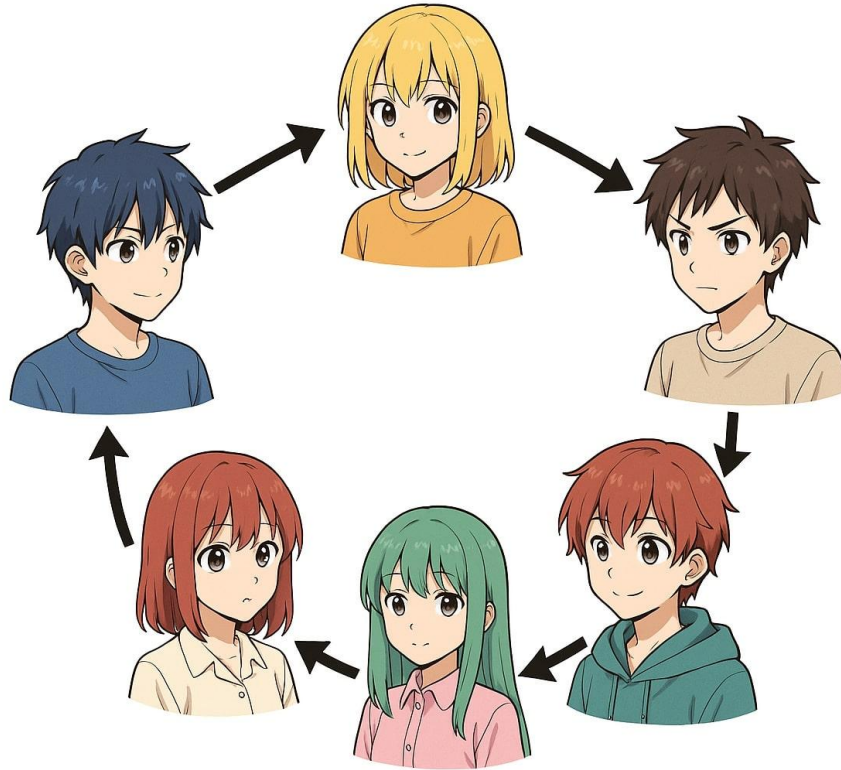
- ◆ **etcd** (base de datos clave-valor) utiliza Raft para gestionar su log replicado; es la base de Kubernetes, Docker Swarm, entre otros.
- ◆ **Neo4j** (base de datos de grafos) garantiza consistencia y seguridad mediante Raft. eBay ocupa Neo4j.
- ◆ **ScyllaDB** (base de datos distribuida NoSQL) usa Raft internamente para metadata y transacciones. Discord y Disney+ ocupan ScyllaDB.

# *Ring-based Election*

Solo lo vemos si nos da el tiempo, sino lo estudiaremos con calma el lunes 8 de septiembre.

---

# Elección de líder - Algoritmo *Ring-based*





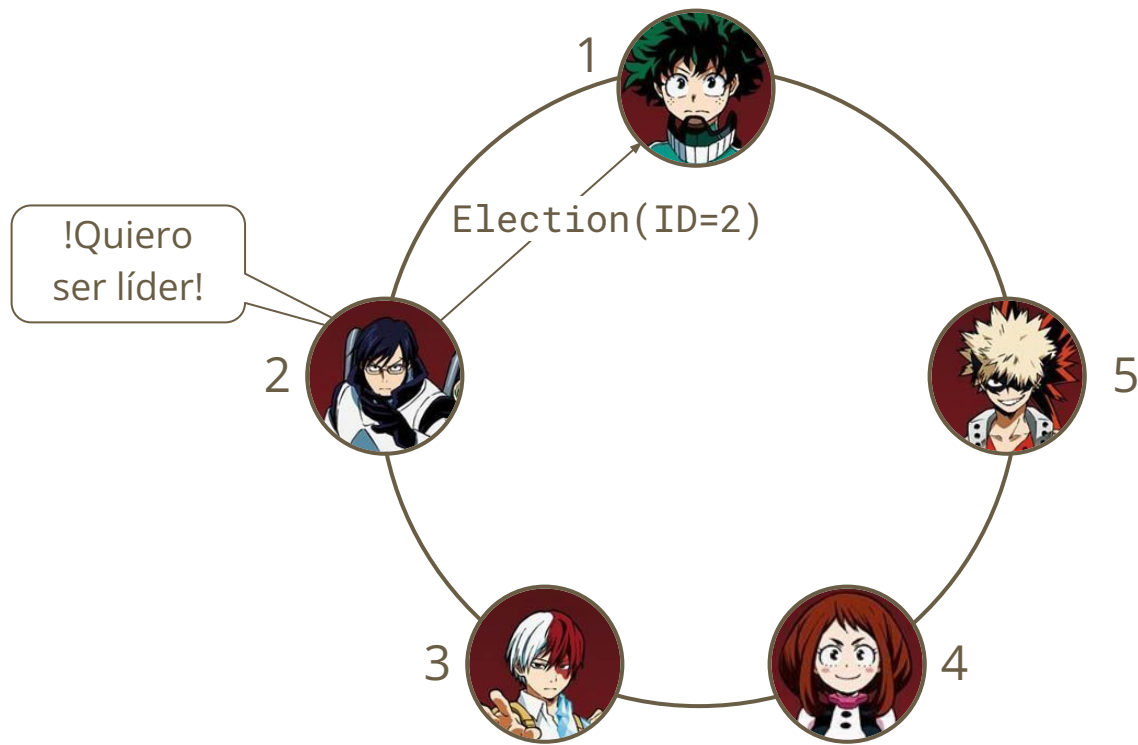
# Elección de líder - Algoritmo *Ring-based*

- ◆ Propuesto por Chang y Roberts en 1979.
- ◆ Los procesos están organizados en forma de un anillo y cada proceso conoce solo el vecino más cercano, no toda la red.
  - ◆ Cada proceso  $P_i$  tiene una comunicación al siguiente proceso en el anillo,  $P_{i+1 \bmod N}$
- ◆ Es un algoritmo asíncrono y, por defecto, no es tolerante a fallos.
- ◆ Supuestos:
  - ◆ El identificador de cada proceso es un valor fijo que nunca cambia.
  - ◆ Cada proceso NO sabe el identificador de los demás procesos.

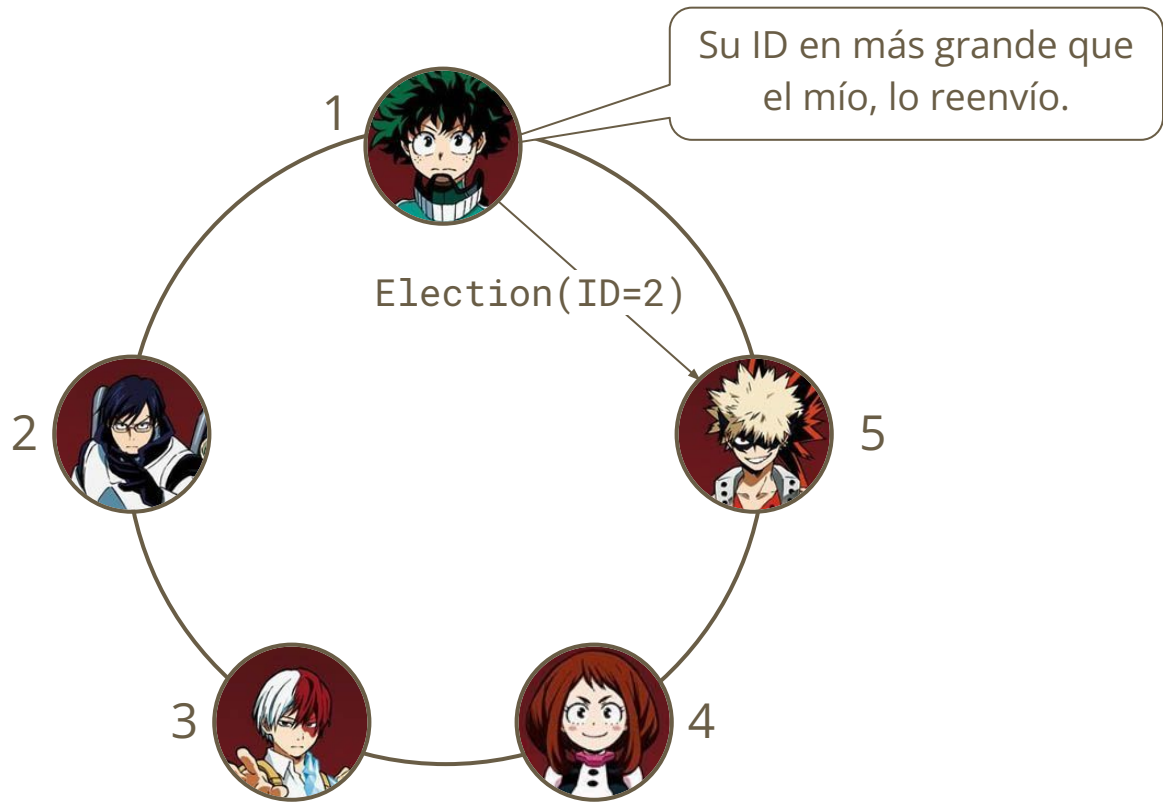
# Elección de líder - Algoritmo *Ring-based* - Funcionamiento

- ◆ Este algoritmo, por defecto, utiliza 2 tipos de mensajes:
  - ◆ **Election**: Anunciar una elección.
  - ◆ **Elected**: Enviado para anunciar la identidad del proceso elegido.

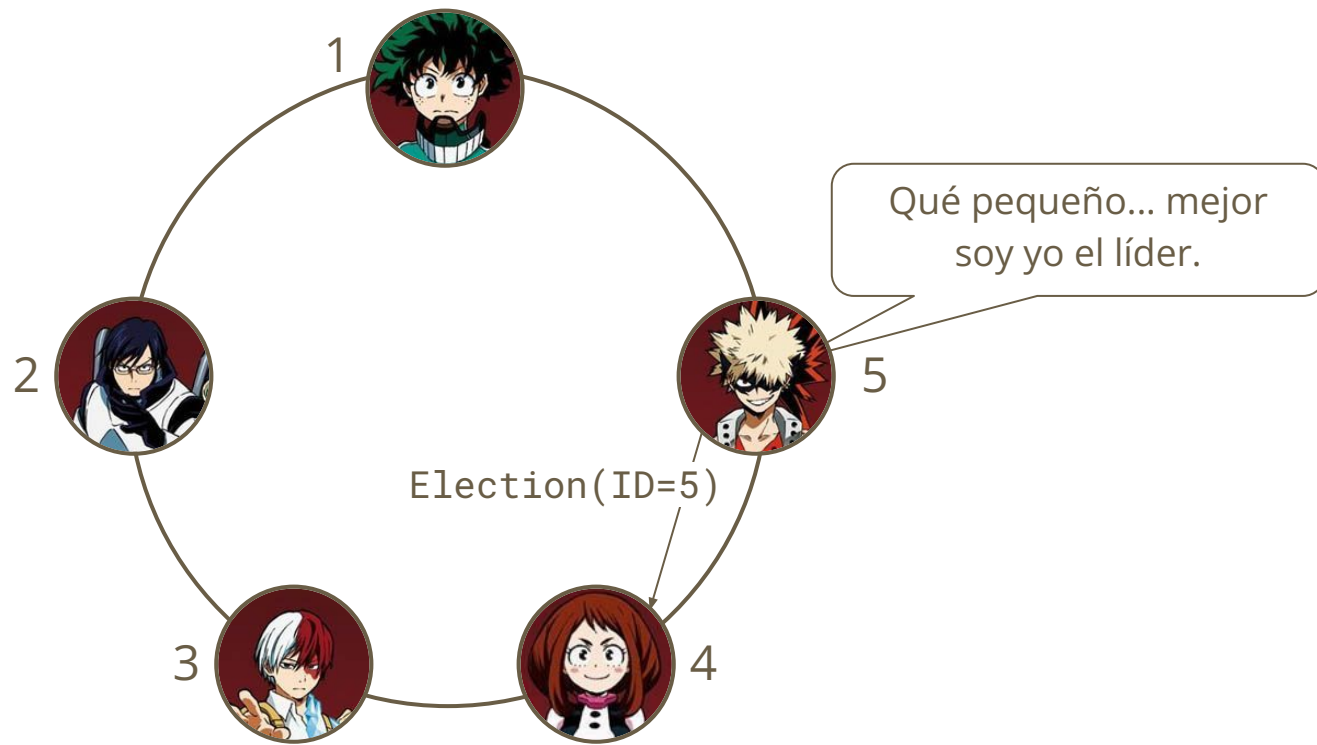
# Elección de líder - Algoritmo *Ring-based* - Funcionamiento



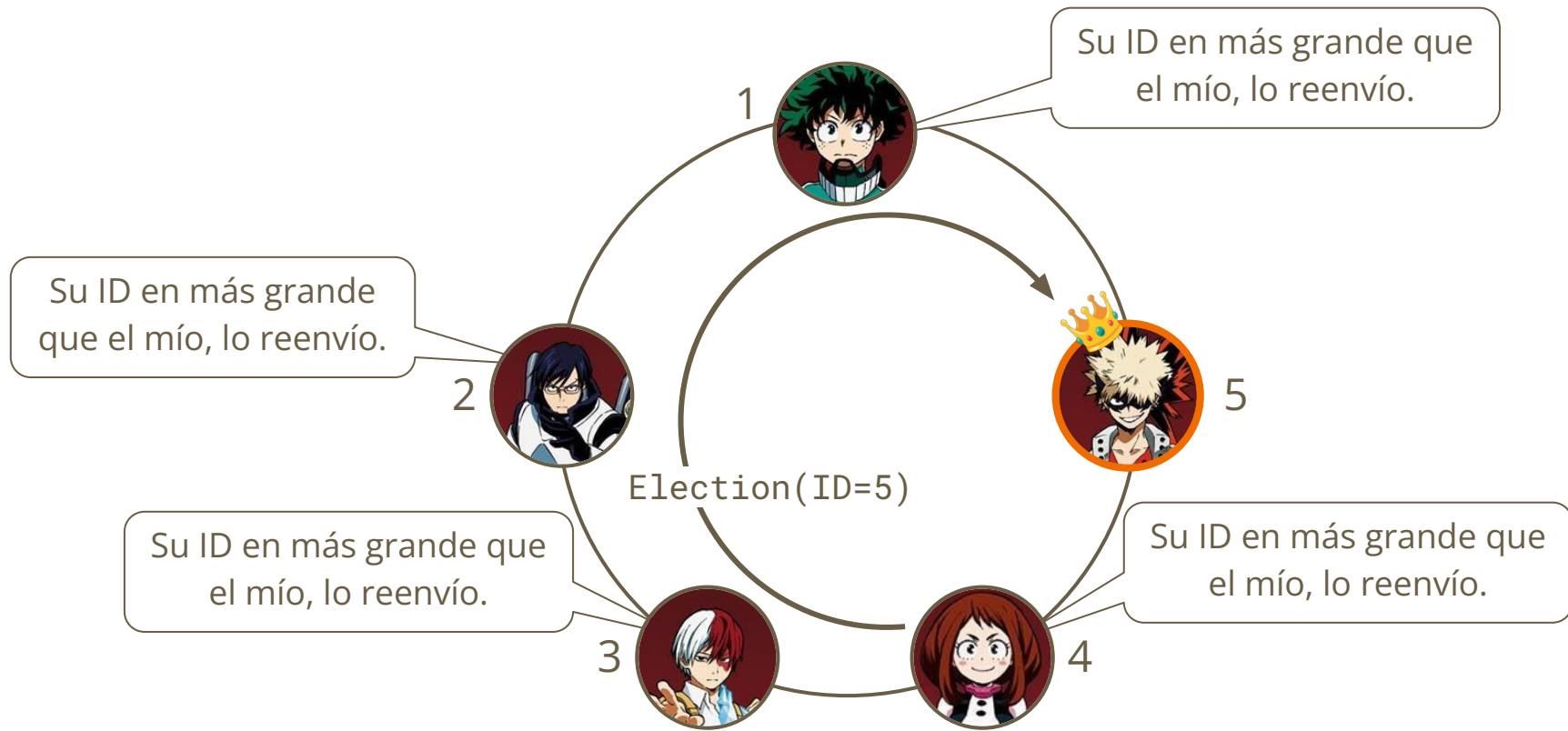
# Elección de líder - Algoritmo *Ring-based* - Funcionamiento



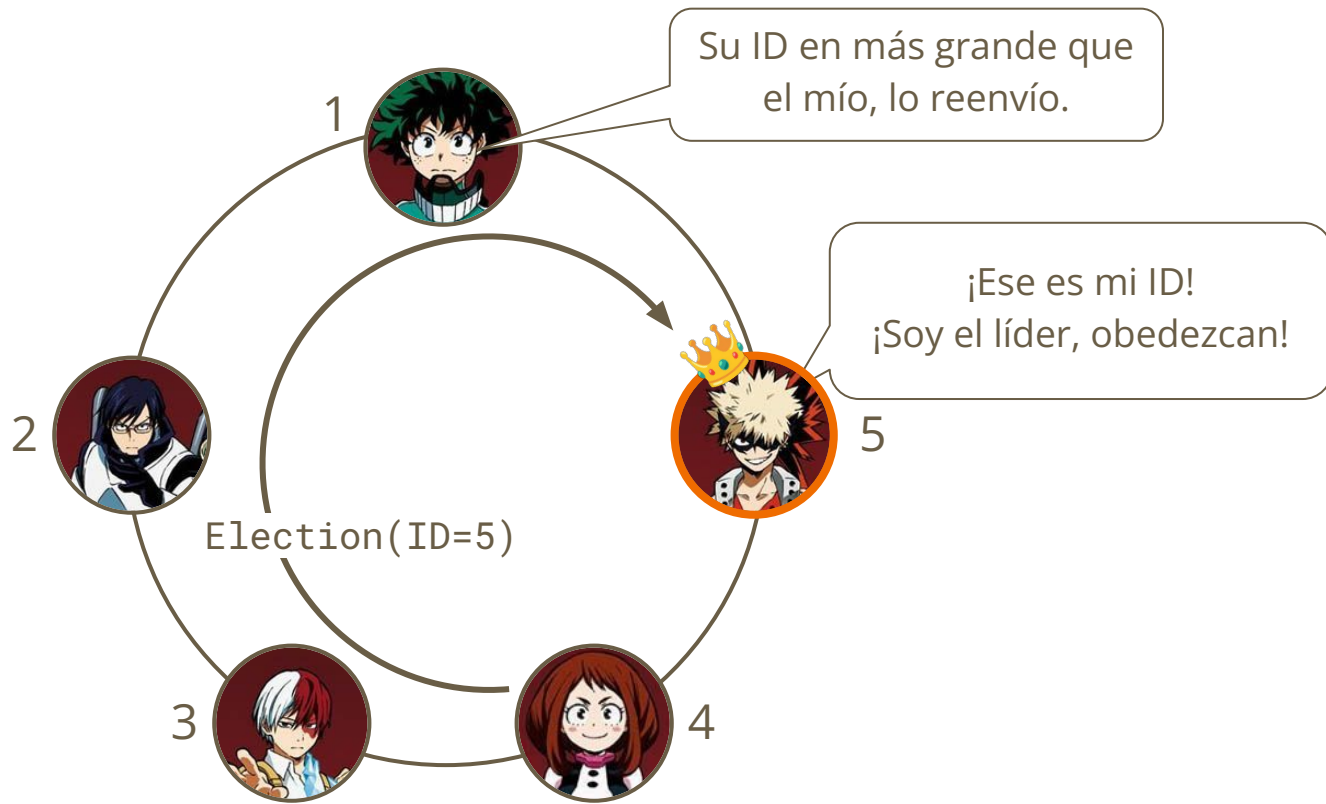
# Elección de líder - Algoritmo *Ring-based* - Funcionamiento



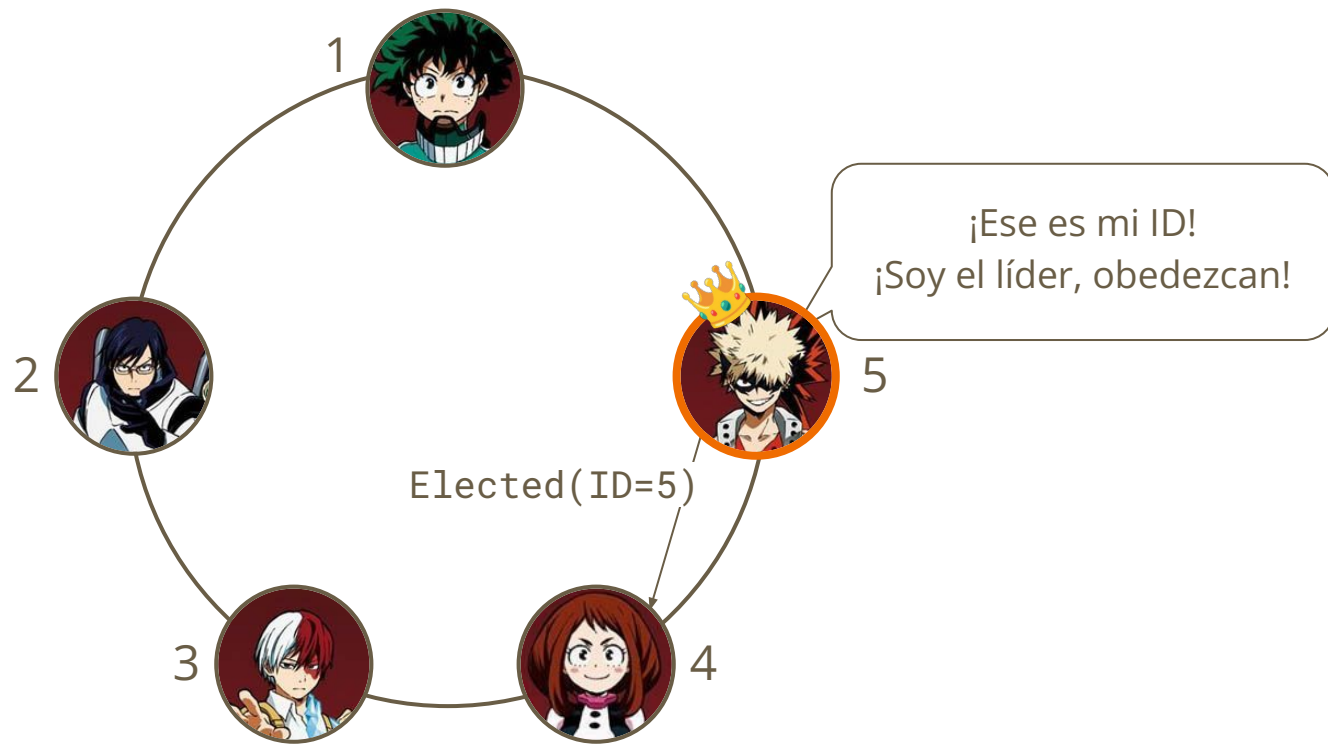
# Elección de líder - Algoritmo *Ring-based* - Funcionamiento



# Elección de líder - Algoritmo *Ring-based* - Funcionamiento

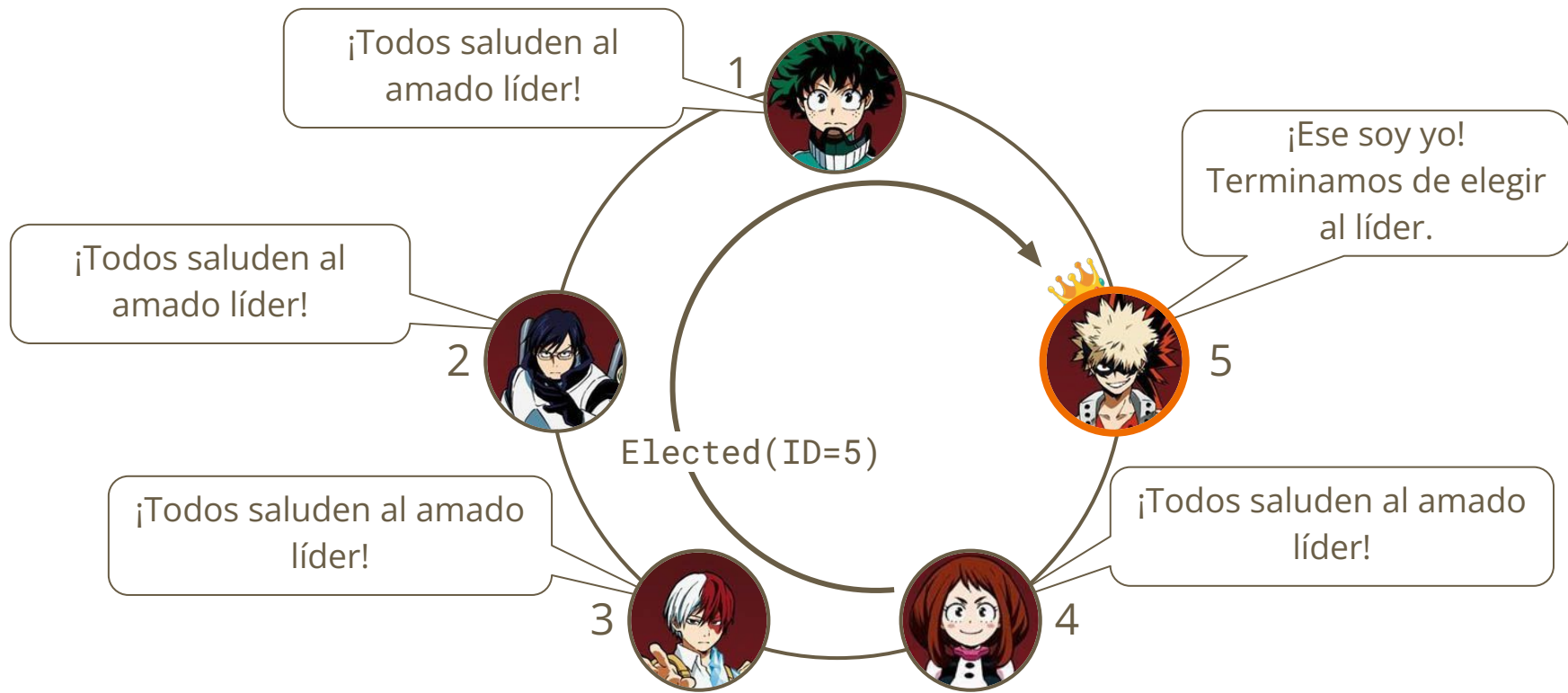


# Elección de líder - Algoritmo *Ring-based* - Funcionamiento





# Elección de líder - Algoritmo *Ring-based* - Funcionamiento



# Elección de líder - Algoritmo *Ring-based* - Resumen

- ◆ Cualquier proceso  $P_i$  se auto elige como líder, y manda un mensaje de `Election` a su vecino adjuntando su propio  $ID_i$ .
- ◆ Cuando proceso  $P_x$  recibe mensaje tipo `Election` con  $ID_i$ 
  - ◆ Si  $ID_i < ID_x$  se reemplaza  $ID_i$  por  $ID_x$  y se manda el mensaje `Election` con  $ID_x$
  - ◆ Si  $ID_i > ID_x$  se reenvía el mensaje al siguiente vecino sin cambiar nada.
  - ◆ Si  $ID_i == ID_x$  significa que el mensaje dió una vuelta completa. Se comienza a notificar que el líder será el proceso con  $ID_i$ , y se espera que el mensaje de una vuelta completa.

# Elección de líder - Algoritmo *Ring-based* - Observaciones

- ◆ En el peor caso que el proceso con mayor ID es justo el anterior al proceso que comienza la elección, se mandarán  $3 \times N - 1$  mensajes.
- ◆ Existen muchas variaciones a este tipo de algoritmo:
  - ◆ Comunicación bi-direccional.
  - ◆ Agregar *timeouts* para transformarlo en un algoritmo síncrono.
  - ◆ Conocer al "vecino del vecino" para poder saltar a algún nodo en caso de caídas.
  - ◆ Agregar mensajes de respuestas al Election.

# Poniendo a prueba lo que hemos aprendido

Respecto a la consolidación de datos utilizando el algoritmo Raft visto en clases, ¿cuál de las siguientes afirmaciones es **incorrecta**?

- I. Una operación que no fue consolidado durante un *term* nunca podrá consolidarse en mandatos posteriores.
- II. Si una operación logra almacenarse en la mayoría de los nodos luego de diferentes mandatos, se garantiza que dicha operación será consolidado.
- III. Si una operación se almacena en la mayoría de los nodos durante el mismo *term* este será consolidado.
- IV. Es posible consolidar más de una operación en un mismo *term*.

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

# Poniendo a prueba lo que hemos aprendido

Respecto a la consolidación de datos utilizando el algoritmo Raft visto en clases, ¿cuál de las siguientes afirmaciones es **incorrecta**?

- I. Una operación que no fue consolidado durante un *term* nunca podrá consolidarse en mandatos posteriores.
- II. Si una operación logra almacenarse en la mayoría de los nodos luego de diferentes mandatos, se garantiza que dicha operación será consolidado.
- III. Si una operación se almacena en la mayoría de los nodos durante el mismo *term* este será consolidado.
- IV. Es posible consolidar más de una operación en un mismo *term*.

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

# Próximos eventos

## Próxima clase

- ◆ Profundizaremos en algoritmos de exclusión mutua
  - ◆ ¿Cómo aseguramos que 1 nodo acceda a una zona crítica ahora que están repartidos y hay latencia en la comunicación?

## Evaluación

- ◆ El viernes se entrega la T1 y máximo plazo (con atraso) es el domingo a las 20:00.
- ◆ Mañana se publica el Control 3 que evalúa hasta esta clase.
  - ◆ Usenlo de estudio para la I1.
  - ◆ Su solución si o si se libera el jueves a las 20:00. La nota final puede tomar un poco más de tiempo si es que hay que hacer algún ajuste.

---

# IIC2523

# Sistemas Distribuidos

— Hernán F. Valdivieso López —  
(2025 - 2 / Clase 08)

---

# Créditos (animes utilizados)

Lycoris Recoil



Oshi no Ko



Sakura Card Captor



One Piece



Sword Art Online



Boku no Hero Academia



Shingeki no Kyojin



Konosuba



Spy x Family

