Lección 12: Algoritmos de consenso

- Introducción
- Enunciado del problema
- Consenso: Algoritmo de una ronda
- Consenso: Algoritmo de los generales bizantinos
- Consenso: Algoritmo del rey

Introducción

- Los sistemas distribuidos son propensos a fallos
- La *replicación* es una estrategia habitual para conseguir que un sistema sea "más" fiable
  - Si replico y no hay fallos, todos los nodos deberían calcular el mismo resultado (suponiendo que los datos de entrada y el algoritmo de cálculo es el mismo)
  - Si replico y hay fallos, pueden generarse distintos resultados
    - Resultado "válido" determinado por medio de **algoritmos de consenso**
- Tipos de fallos a considerar:
  - *Fallos por caídas*: un nodo deja de enviar mensajes
  - *Fallos bizantinos*: un nodo envía mensajes arbitrarios

Enunciado del problema

Lamport, Shostak, Pease, 1982

- Un grupo de ejércitos bizantinos está rodeando una ciudad enemiga. El balance de fuerzas es tal que si todos los ejércitos atacan simultáneamente, pueden capturar la ciudad; en caso contrario, todos deben retirarse para evitar la derrota. Los generales tienen mensajeros fiables capaces de entregar con éxito cualquier mensaje enviado de un general a otro. El problema es que algunos de los generales son traidores y tienen como objetivo que los ejércitos sean derrotados.

  El objetivo es definir un algoritmo que facilite que todos los generales leales lleguen a un consenso sobre el plan de actuación. La decisión final será por votación de la mayoría sobre sus elecciones iniciales. Si empatan, la decisión es retirada.

Enunciado del problema

- Interpretación del problema:
  - Los generales son los nodos de cómputo
  - Los mensajeros son los canales de comunicación
  - Los generales pueden fallar, pero no los mensajeros
- Los generales traidores representan los fallos en el sistema
  - *Fallo por caída*: un traidor que simplemente deja de enviar mensajes
    - asumimos que podemos detectar cuándo un nodo cae
  - *Fallo bizantino*: un traidor que manda mensajes "confusos"
- La decisión final (consenso) está basada en la opinión de la mayoría

Consenso: Algoritmo de una ronda

- Cada general envía su plan a los demás generales y espera los planes de éstos

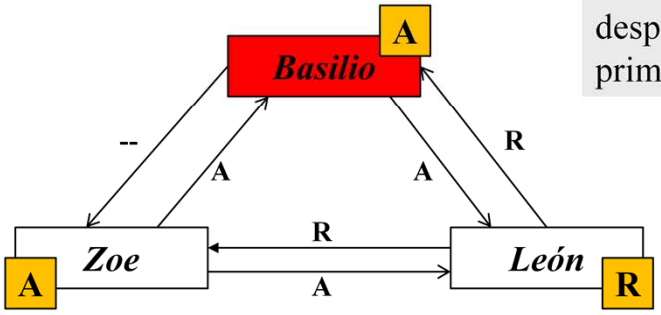
```
planType finalPlan
process General    planType array[generals] plan

P1: plan[myID]:= chooseAttackOrRetreat
P2: for all other generals G
P3:    send(G, myID, plan[myID])
P4: for all other generals G
P5:    receive(G, plan[G])
P6: finalPlan:= majority(plan)
```

Consenso: Algoritmo de una ronda

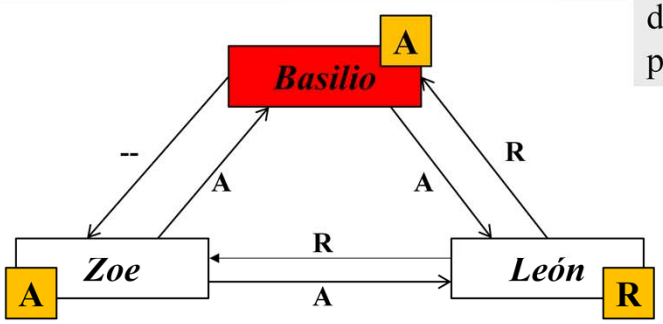
- Sean 3 generales, supongamos que uno de ellos es traidor (Basilio) y sufre un fallo por caída:

Basilio "peta" después del primer mensaje



Consenso: Algoritmo de una ronda

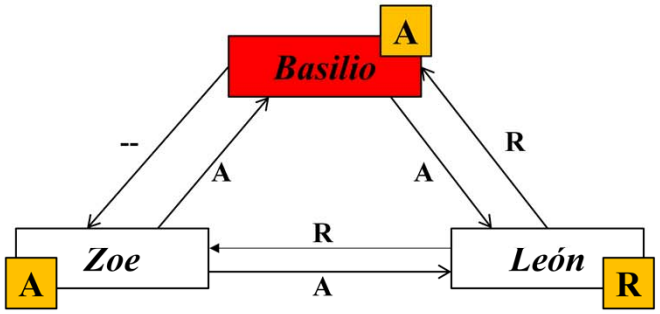
Basilio "peta" después del primer mensaje



Zoe	
General	**Plan**
Basilio	-
León	R
Zoe	A
Majority	**R**

León	
General	**Plan**
Basilio	A
León	R
Zoe	A
Majority	**A**

Consenso: Algoritmo de una ronda



Zoe	
General	Plan
Basilio	-
León	R
Zoe	A
Majority	R

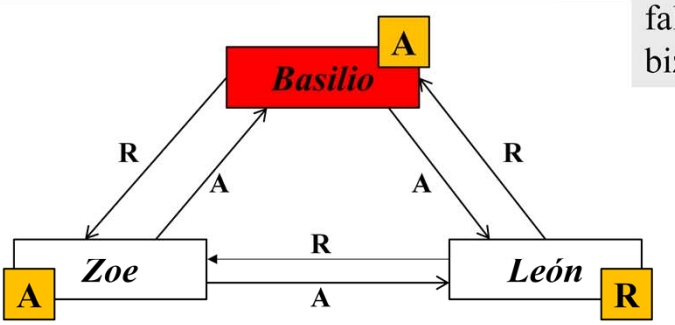
León	
General	Plan
Basilio	A
León	R
Zoe	A
Majority	A

Decisiones de los generales leales

Plan final = R

Consenso: Algoritmo de una ronda

Tampoco para fallos bizantinos



Zoe	
General	**Plan**
Basilio	R
León	R
Zoe	A
Majority	**R**

León	
General	**Plan**
Basilio	A
León	R
Zoe	A
Majority	**A**

Consenso: Algoritmo de los generales bizantinos

- El algoritmo anterior atribuye la misma importancia a los planes de los distintos generales
- ¿Por qué no dar más importancia a los planes de los generales leales? → deberíamos conocer su identidad
  - En un sistema distribuido no se puede conocer la identidad de los generales traidores (nodos que fallarán)
- Por tanto, la solución pasa por asegurar que los planes de los traidores no afecten al consenso de los leales
  - Envío de planes basado en "doble vuelta":
    - "Primera vuelta": generales envían su decisión
    - "Segunda vuelta": generales envían los planes recibidos de otros generales

Consenso: Algoritmo de los generales bizantinos

```
planType finalPlan
planType array[generals] plan
planType array[generals, generals] reportedPlan
planType array[generals] majorityPlan
```

```
process General

P1: plan[myID]:= chooseAttackOrRetreat

P2: for all other generals G                    // First round
P3:     send(G, myID, plan[myID])
P4: for all other generals G
P5:     receive(G, plan[G])

P6: for all other generals G                    // Second round
P7:     for all other generals G' except G
P8:             send(G', myID, G, plan[G])
P9: for all other generals G
P10:    for all other generals G' except G
P11:            receive(G, G', reportedPlan[G,G'])

… // First and second Vote
```

G me dice que G' le dijo...

Consenso: Algoritmo de los generales bizantinos

```
planType finalPlan
planType array[generals] plan
planType array[generals, generals] reportedPlan
planType array[generals] majorityPlan
```

```
process General

P1: plan[myID]:= chooseAttackOrRetreat

P2: for all other generals G                    // First round
P3:     send(G, myID, plan[myID])
P4: for all other generals G
P5:     receive(G, plan[G])

P6: for all other generals G                    // Second round
P7:     for all other generals G' except G
P8:             send(G', myID, G, plan[G])
P9: for all other generals G
P10:    for all other generals G' except G
P11:            receive(G, G', reportedPlan[G,G'])

… // First and second Vote
```

G me dice que G' le dijo...

Consenso: Algoritmo de los generales bizantinos

```
… // First and Second Round

P12: for all generals G                                  // First vote
P13:    majorityPlan[G]:= majority(plan[G] ∪ reportedPlan[*,G])

P14: majorityPlan[myID] := plan[myID]                    // Second vote
P15: finalPlan:= majority(majorityPlan)
```

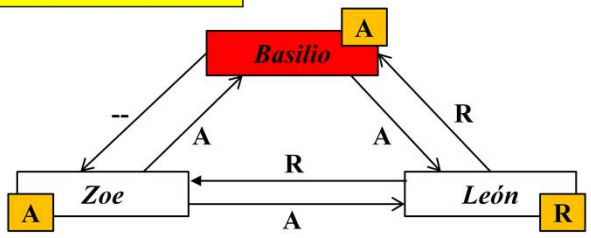
Consenso: Algoritmo de los generales bizantinos

- Los siguientes escenarios pretenden demostrar que:
  - **Ejemplo 1**: El algoritmo evita decisiones inconsistentes como consecuencia de fallos por caídas
  - **Ejemplo 2**: El algoritmo puede llevar a decisiones inconsistentes si se trata de fallos bizantinos (2 generales leales, 1 general traidor)
  - **Ejemplo 3**: El algoritmo puede garantizar decisiones consistentes si se trata de fallos bizantinos (3 generales leales, 1 general traidor)

Consenso: Algoritmo de los generales bizantinos

Ejemplo 1: caída en "primera vuelta"

Basilio "peta" después del primer mensaje



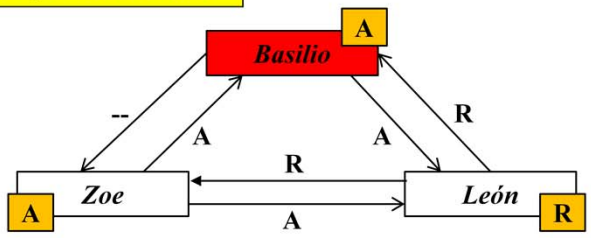
Zoe				
General	**Plan**	**Reported by**		**Majority**
		Basilio	León	
Basilio				
León				
Zoe				
Majority				

León				
General	**Plan**	**Reported by**		**Majority**
		Basilio	Zoe	
Basilio				
León				
Zoe				
Majority				

Consenso: Algoritmo de los generales bizantinos

Ejemplo 1: caída en "primera vuelta"

Basilio "peta" después del primer mensaje



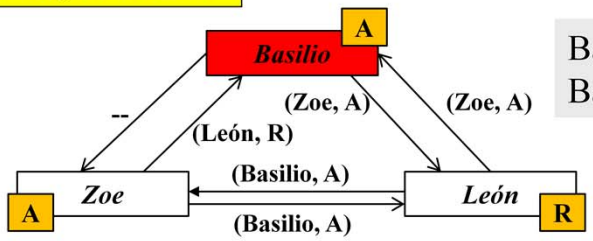
Zoe				
General	**Plan**	**Reported by**		**Majority**
		Basilio	León	
Basilio	-		**A**	**A**
León	R	-		**R**
Zoe	A			**A**
Majority				**A**

León				
General	**Plan**	**Reported by**		**Majority**
		Basilio	Zoe	
Basilio	A		-	**A**
León	R			**R**
Zoe	A	-		**A**
Majority				**A**

Basta con que un general envíe un mensaje para que todos lo sepan

Consenso: Algoritmo de los generales bizantinos

Ejemplo 1: caída en "segunda vuelta"



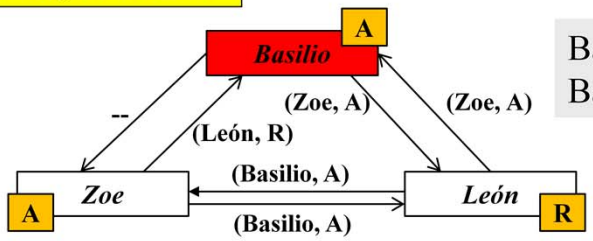
Basilio envía 1ª vuelta
Basilio envía parte 2ª

Zoe				
General	**Plan**	**Reported by**		**Majority**
		Basilio	León	
Basilio				
León				
Zoe				
Majority				

León				
General	**Plan**	**Reported by**		**Majority**
		Basilio	Zoe	
Basilio				
León				
Zoe				
Majority				

Consenso: Algoritmo de los generales bizantinos

Ejemplo 1: caída en "segunda vuelta"



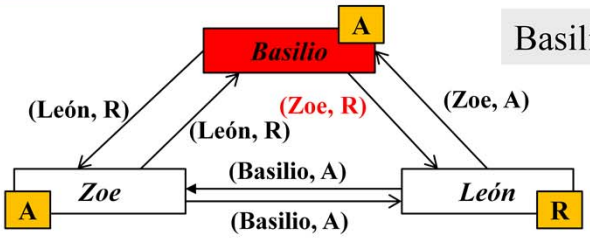
Basilio envía 1ª vuelta
Basilio envía parte 2ª

Zoe				
General	**Plan**	**Reported by**		**Majority**
		Basilio	León	
Basilio	**A**		**A**	**A**
León	R	-		**R**
Zoe	A			**A**
Majority				**A**

León				
General	**Plan**	**Reported by**		**Majority**
		Basilio	Zoe	
Basilio	A		A	**A**
León	R			**R**
Zoe	A	A		**A**
Majority				**A**

Consenso: Algoritmo de los generales bizantinos

Ejemplo 2: Fallo bizantino en "segunda vuelta"



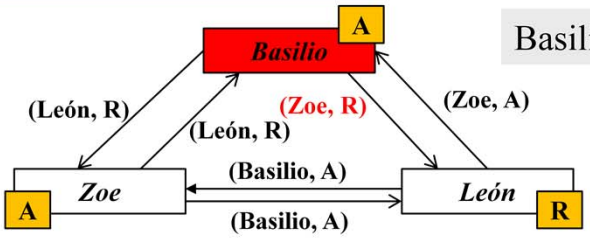
Basilio miente sobre Zoe

Zoe				
General	**Plan**	**Reported by**		**Majority**
		Basilio	León	
Basilio	**A**			
León	R			
Zoe	A			
Majority				

León				
General	**Plan**	**Reported by**		**Majority**
		Basilio	Zoe	
Basilio	A			
León	R			
Zoe	A			
Majority				

Consenso: Algoritmo de los generales bizantinos

Ejemplo 2: Fallo bizantino en "segunda vuelta"



Basilio miente sobre Zoe

Zoe				
General	**Plan**	**Reported by**		**Majority**
		Basilio	León	
Basilio	**A**		**A**	**A**
León	R	**R**		**R**
Zoe	A			**A**
Majority				**A**

León				
General	**Plan**	**Reported by**		**Majority**
		Basilio	Zoe	
Basilio	A		A	**A**
León	R			**R**
Zoe	A	**R**		**R**
Majority				**R**

¡Inconsistencia en las decisones de los generales leales!

Consenso: Algoritmo de los generales bizantinos

Ejemplo 3: Fallos bizantinos

- Escenario para 4 generales:
  - Basilio (A), Juan (A) y León (R) son leales
  - Zoe es el traidor
- Estructura de datos de Basilio con los mensajes recibidos por los generales leales:

Basilio					
General	**Plan**	**Reported by**			**Majority**
		Juan	León	Zoe	
Basilio	**A**				**A**
Juan	**A**		**A**	?	**A**
León	**R**	**R**		?	**R**
Zoe	?	?	?		?
Majority					?

Puede conocerse con certeza la decisión de cada general leal, independientemente de lo que diga el general traidor

Consenso: Algoritmo de los generales biza

R a Basilio y Leo
A a Juan

- Supongamos que Zoe envía en "primera vuelta" mensajes contradictorios:
  - Los mensajes intercambiados entre los generales leales en "segunda vuelta" son correctos

Basilio					
General	**Plan**	**Reported by**			**Majority**
		Juan	León	Zoe	
Basilio	**A**				**A**
Juan	**A**		**A**	?	**A**
León	**R**	**R**		?	**R**
Zoe	**R**	**A**	**R**		**R**
Majority					**R**

El traidor puede influir en la decisión final de los generales leales, pero entre estos existirá consenso (R).

Consenso: Algoritmo de los generales bizantinos

- ¿Cuál sería la decisión final si todos los generales leales hubieran inicialmente elegido el mismo plan (A)?

Basilio					
General	**Plan**	**Reported by**			**Majority**
		Juan	León	Zoe	
Basilio	**A**				**A**
Juan	**A**		**A**	?	**A**
León	**A**	**A**		?	**A**
Zoe	**?**	**?**	**?**		**?**
Majority					**?**

Consenso: Algoritmo de los generales bizantinos

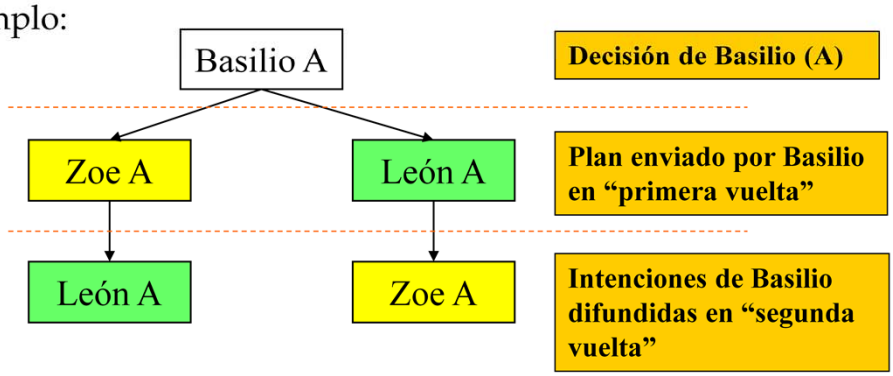
- ¿Y si el traidor manda mensajes contradictorios, pero la mayoría de estos mensajes es "atacar" (A)?

Basilio					
General	**Plan**	**Reported by**			**Majority**
		Juan	León	Zoe	
Basilio	**A**				**A**
Juan	**A**		**A**	?	**A**
León	**R**	**R**		?	**R**
Zoe	**A**	**A**	**R**		**?**
Majority					**?**

Consenso: Algoritmo de los generales bizantinos

- **Corrección del algoritmo** basada en *Knowledge Trees (KT)*
  - Un KT representa la información en el sistema sobre el general en su nodo raíz

- Ejemplo:



Consenso: Algoritmo de los generales bizantinos

- Sean 3 generales (uno de ellos traidor, Basilio), el algoritmo de los generales bizantinos será correcto ante fallos por caída si:
  - para cualquier escenario, los generales leales siempre llegan a la misma conclusión sobre los planes del traidor

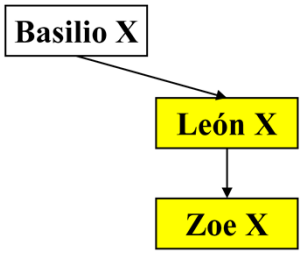
Consenso: Algoritmo de los generales bizantinos

- Sean 3 generales (uno de ellos traidor, Basilio), el algoritmo de los generales bizantinos será correcto ante fallos por caída si:
  - para cualquier escenario, los generales leales siempre llegan a la misma conclusión sobre los planes del traidor
- **Caso 1**: Basilio no manda ningún mensaje.

Consistente !

Consenso: Algoritmo de los generales bizantinos

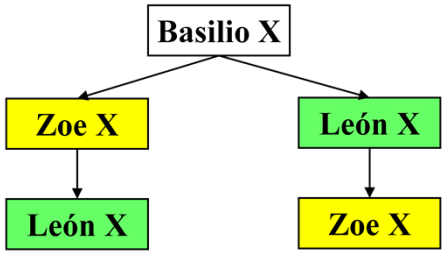
- Sean 3 generales (uno de ellos traidor, Basilio), el algoritmo de los generales bizantinos será correcto ante fallos por caída si:
  - para cualquier escenario, los generales leales siempre llegan a la misma conclusión sobre los planes del traidor
- **Caso 2**: Basilio envía un único mensaje en "primera vuelta" a un único general leal (por ejemplo, a León).



Consistente !

Consenso: Algoritmo de los generales bizantinos

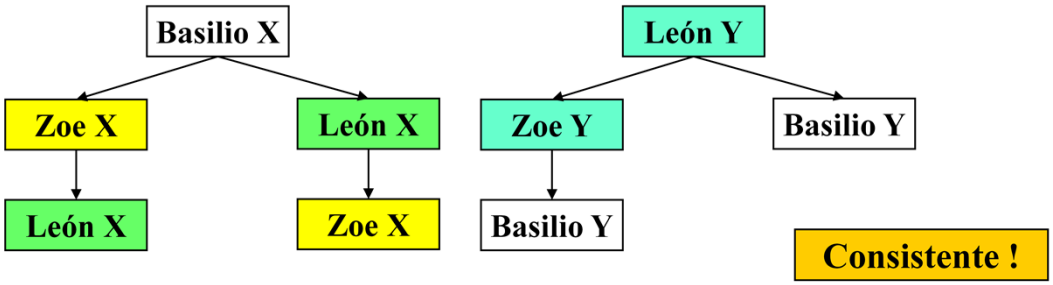
- Sean 3 generales (uno de ellos traidor, Basilio), el algoritmo de los generales bizantinos será correcto ante fallos por caída si:
  - para cualquier escenario, los generales leales siempre llegan a la misma conclusión sobre los planes del traidor
- **Caso 3**: Basilio envía correctamente los mensajes de "primera vuelta", pero cae antes de enviar los de "segunda vuelta"



Consistente !

Consenso: Algoritmo de los generales bizantinos

- Sean 3 generales (uno de ellos traidor, Basilio), el algoritmo de los generales bizantinos será correcto ante fallos por caída si:
  - para cualquier escenario, los generales leales siempre llegan a la misma conclusión sobre los planes del traidor
- **Caso 4**: Basilio envía correctamente los mensajes de "primera vuelta", pero no el segundo a Zoe ¿qué conoce Zoe de León?



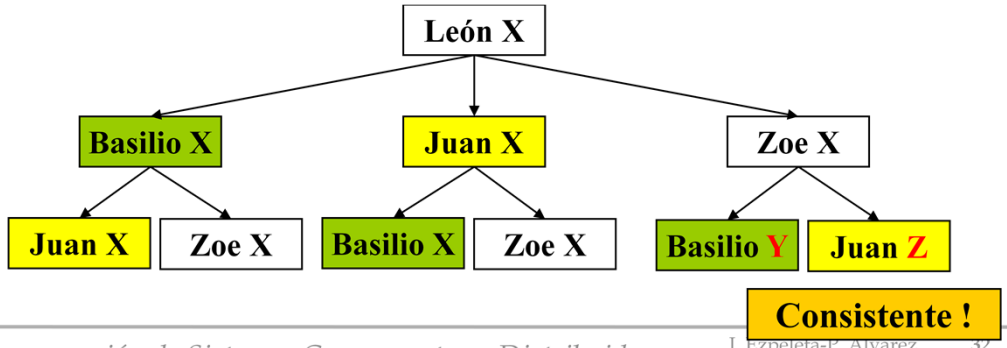
Consistente !

Consenso: Algoritmo de los generales bizantinos

- Sean 4 generales (uno de ellos traidor, Zoe), el algoritmo de los generales bizantinos será correcto ante fallos bizantinos si:
  - para cualquier escenario, los generales leales siempre llegan a la misma conclusión sobre los planes del traidor

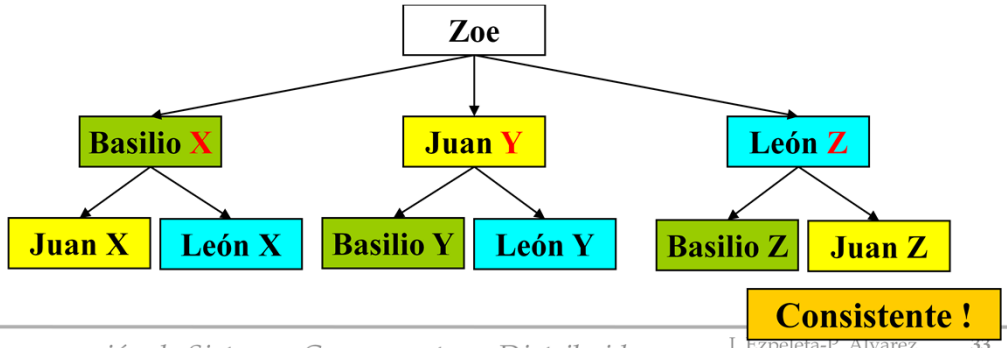
Consenso: Algoritmo de los generales bizantinos

- Sean 4 generales (uno de ellos traidor, Zoe), el algoritmo de los generales bizantinos será correcto ante fallos bizantinos si:
  - para cualquier escenario, los generales leales siempre llegan a la misma conclusión sobre los planes del traidor
- KT de un general leal:



Consenso: Algoritmo de los generales bizantinos

- Sean 4 generales (uno de ellos traidor, Zoe), el algoritmo de los generales bizantinos será correcto ante fallos bizantinos si:
  - para cualquier escenario, los generales leales siempre llegan a la misma conclusión sobre los planes del traidor
- KT del general traidor:



Consenso: Algoritmo de los generales bizantinos

- Por tanto, el algoritmo de los generales bizantinos **evita inconsistencia en la información** en:
  - Cualquier escenario que experimente fallos por caídas
  - Aquellos escenarios que experimenten fallos bizantinos si el número de generales es **>= 3t+1**, siendo **t** el número de generales traidores

- No obstante, este algoritmo **requiere un elevado flujo de mensajes**, especialmente conforme aumenta el número de generales

Consenso: Algoritmo del rey

- Basado en la idea de que un número pequeño de traidores no puede influir en las intenciones de la mayoría
  - El intercambio de mensajes es menor
  - Pero requiere que el número de generales sea (4t+1), siendo t el número de generales traidores

- En cada ronda el voto de un general tiene una mayor importancia (status de Rey)
  - La identidad del Rey no es conocida por el resto de los nodos
  - El Rey puede ser un general leal o traidor

Consenso: Algoritmo del rey

```
planType finalPlan, myMajority, kingPlan
planType array[generals] plan
Integer votesMajority, kingID
```

```
process General

P1: plan[myID]:= chooseAttackOrRetreat

P2: do TWO times
P3:   for all other generals G        //first and third round
P4:       send(G, myID, plan[myID])
P5:   for all other generals G
P6:       receive(G, plan[G])
P7:   myMajority:= majority(plan)
P8:   votesMajority:= number of votes for myMajority

...
```

5 generales (t = 1)

Consenso: Algoritmo del rey

```
P2: do TWO times
...

P9:   if my turn to be king        //second and fourth round
P10:      for all other generals G
P11:         send(G, myID, myMajority)
P12:      plan[myID]:= myMajority
      else
P13:      receive(kingID, kingPlan)
P14:      if (votesMajority > 3)
P15:          plan[myID]:= myMajority
          else
P16:          plan[myID]:= kingPlan

P17: finalPlan:= plan[myID]        // Final decision
```

mayor que (generales/2)+t

Consenso: Algoritmo del rey

- Escenarios de ejemplo con 5 generales (Miguel es el traidor)
  - Planes iniciales: Basilio y Juan (A); Zoe y León (R)

- Interés por la consistencia de las estructuras de datos de los generales leales:
  - **Ejemplo 1**: el primer Rey es un general leal (Zoe)
  - **Ejemplo 2**: el primer Rey es un general traidor (Miguel) y éste envía mensajes contradictorios

Consenso: Algoritmo del rey

Ejemplo 1

- Escenario con 5 generales (Miguel es el traidor):

Basilio							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
A	A	R	**R**	R	R	3	
Juan							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
A	A	R	**A**	R	A	3	
León							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
A	A	R	**A**	R	A	3	
Zoe							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
A	A	R	**R**	R	R	3	

Consenso: Algoritmo del rey

Ejemplo 1

- Zoe es el Rey (R)

Basilio							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
R							R
Juan							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
	R						R
León							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
		R					R
Zoe							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
				R			

Consenso: Algoritmo del rey

Ejemplo 1

- Tercera ronda de envío de planes:

Basilio							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
R	R	R	?	R	R	4-5	
Juan							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
R	R	R	?	R	R	4-5	
León							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
R	R	R	?	R	R	4-5	
Zoe							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
R	R	R	?	R	R	4-5	

Mayorías sólidas, decisión final (R)

Consenso: Algoritmo del rey

Ejemplo 2

- Supongamos que el primer Rey es el traidor (Miguel):

Basilio							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
R							R
Juan							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
	A						A
León							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
		A					A
Zoe							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
				R			R

Consenso: Algoritmo del rey

Ejemplo 2

- Tercera ronda de envío de planes:

Basilio							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
R	A	A	?	R	?	3	
Juan							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
R	A	A	?	R	?	3	
León							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
R	A	A	?	R	?	3	
Zoe							
Basilio	**Juan**	**León**	**Miguel**	**Zoe**	**myMajority**	**votesMajority**	**kingPlan**
R	A	A	?	R	?	3	

Consenso: Algoritmo del rey

Ejemplo 2

- Cuarta ronda, Zoe es el Rey (A):

Basilio							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
A							A
Juan							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
A							A
León							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
A							A
Zoe							
Basilio	Juan	León	Miguel	Zoe	myMajority	votesMajority	kingPlan
				A			

Decisión final (A)

Ejercicios

Considérese el caso de los generales bizantinos con cuatro procesos, denominados **A**, **B**, **C** y **T**, donde **T** es traidor y los otros tres son leales. Supongamos que **A** y **B** deciden retirarse, mientras que **C** decide atacar. Supongamos también que el traidor termina la primera vuelta de envío de mensajes, comunicando a **A** y **B** que piensa atacar y a **C** que piensa retirarse.

Se pide construir y completar para cada general leal la estructura de datos con la que tomará la decisión final (en la figura 1 se muestra el cuadro a completar para el general **A**). ¿Llegarán a consenso? Si es así ¿qué decidirán hacer?

En un sistema de control, un proceso denominado `Productor` genera cada cierto tiempo un número real, que se almacena en un *buffer* con capacidad para 100 reales, cuyo acceso se gestiona mediante un monitor, denominado `control`. Un proceso denominado `Consumidor` usa los números producidos como sigue: invoca a la operación `dame_media` del monitor recibiendo como respuesta la media aritmética de tres números consecutivos en el *buffer*,y luego muestra esa media. Así, la primera vez que se invoque será la media de los números primero, segundo y tercero; la segunda será la media de los números recibidos como segundo, tercero y cuarto, etc. Por ejemplo, si el productor ha generado la secuencia `1.0, 4.0, 3.0, 5.0, 5.0` y `6.0`, la primera invocación del consumidor recibirá `2.6` (media de `(1.0+4.0+3.0)/3.0`); la segunda invocación recibirá `4.0` (media de `(4.0+3.0+5.0)/3.0`), la tercera `4.3` (media de `(3.0+5.0+5.0)/3.0`), y así sucesivamente. El ejercicio pide:

1. Escribir en lenguaje algorítmico el código del monitor y de los procesos.
2. Dar una versión alternativa de manera que para la sincronización se use, en lugar de un monitor, un proceso `Control_buffer` y un espacio de tuplas Linda.