# CONSENSO

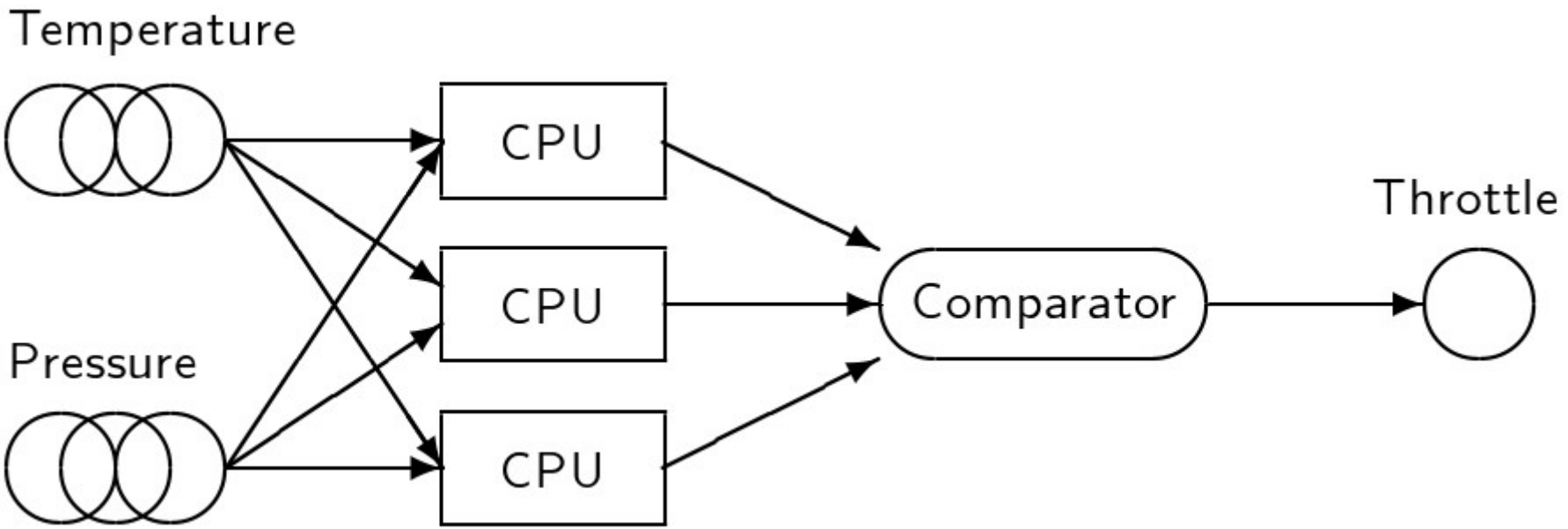
- Aplicação distribuída tem por **finalidade principal** aumentar a confiabilidade
  - Aplicação é replicada em diversas máquinas (nós)

- Deseja-se que a aplicação seja:
  - **Segura contra falhas**:
    - Garante que uma ou mais falhas não cause **prejuízo** ao sistema e usuários
    - A aplicação pode não estar mais funcional
  - **Tolerante a falhas**:
    - A aplicação continua **funcional** apesar de uma ou mais falhas

- Uma aplicação distribuída **não é automaticamente** segura à falhas ou tolerante a falhas
  - Considere o algoritmo de Ricart-Agrawala para exclusão mútua distribuída
    - A aplicação requer a colaboração de todos os nós
    - Caso um dos nós falhe, a aplicação entra em *deadlock*

- Considere uma arquitetura distribuída **confiável** em que:
  - Os dados de entrada podem provir da mesma fonte ou não
    - Dados podem ser originados em um único sensor
    - Dados podem vir de vários sensores replicados para aumentar a confiabilidade

- Cada um dos nós processa os dados de entrada e produz uma saída
- Considerando as saídas, uma determinada ação deve ser tomada



- Algumas questões devem ser observadas:
  - Quando **sensores são replicados**, eles **não indicam** exatamente **o mesmo valor**.
    - Os diversos nós podem receber dados de entrada ligeiramente diferentes
  - Sensores podem produzir **dados espúrios** que ultrapassam a faixa de valores consideradas pelos algoritmos
  - Se todos os nós **usarem o mesmo software**, o sistema **não é tolerante a falhas** decorrentes de ***bugs*** no software

- Se **não usarem o mesmo software**, os resultados podem **diferir** para os mesmos dados de entrada:
  - Programadores são propensos a fazer interpretações equivocadas das especificações do software

- Nestas condições, um sistema tolerante a falhas deve decidir qual das respostas obtidas é a correta

- Em aplicações distribuídas, isto corresponde a um problema de **consenso**:
  - Cada um dos nós encontra um valor inicial
  - De todos os valores encontrados, cada nó deve optar por um resultado
  - Todos os nós devem optar pelo **mesmo valor**

- Se **não houver falhas**, a escolha é trivial:
  - Cada nó manda para os demais a opção inicial
  - Um algoritmo simples determina a resposta da maioria
  - Como todos os nós têm os mesmos dados (das escolhas) e o mesmo algoritmo de maioria
    - A opção da maioria **é igual** em todos os nós.

- Há **dois tipos de falhas** a serem consideradas:
  - **Falhas de quebra**:
    - O nó com a falha para de mandar mensagens
  - **Falhas Bizantinas**:
    - O nó com a falha manda mensagens aleatórias

# PROBLEMA DOS GENERAIS BIZANTINOS

- Usado para modelar a obtenção de consenso em problemas de comunicação
- **Enunciado**:
  - Um grupo de exércitos Bizantinos está cercando uma cidade inimiga
  - O equilíbrio de forças é tal que:
    - Se todos os exércitos **atacarem ao mesmo tempo**, eles conseguem capturar a cidade
    - Caso contrário, eles terão que se retirar para evitar a derrota
  - Os generais de cada exército tem mensageiros confiáveis que podem usar para enviar mensagens a qualquer outro general dos demais exércitos

- Entretanto, alguns generais podem ser **traidores** querendo que os exércitos sejam derrotados

- É preciso encontrar um algoritmo de modo que os generais **leais** possam chegar a um **consenso** sobre a estratégia adequada a ser usada

- A decisão final precisa ser a mesma da maioria dos votos iniciais

- Caso a decisão seja equilibrada, deve-se optar pela retirada

- No problema dos Generais Bizantinos, ambas as falhas ocorrem na seguinte forma:
  - **Falha de quebra** :
    - O general traidor pára de enviar o mensageiro em um determinado ponto da execução do algoritmo.
  - **Falha bizantina**:
    - O general traidor manda mensagens arbitrárias, não apenas as requeridas pelo algoritmo

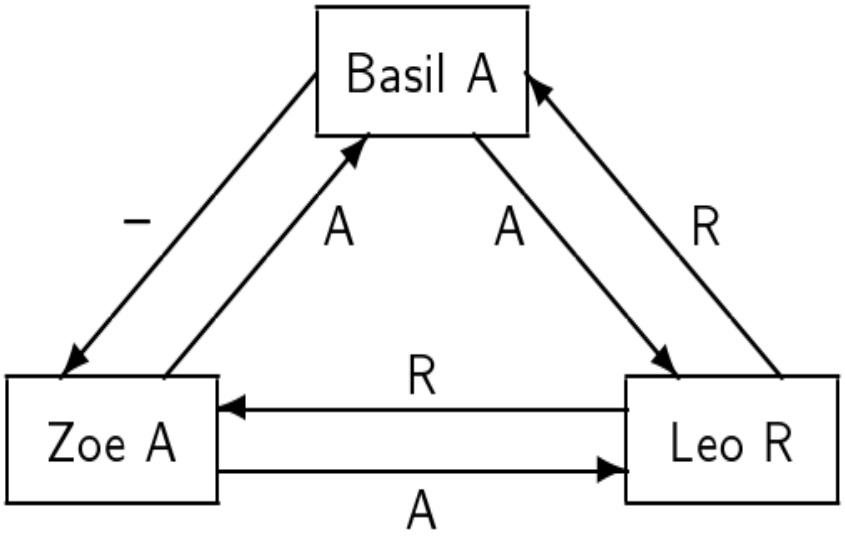
- As falhas de quebra podem ser determinadas estipulando um **tempo máximo de resposta** (*timeout*) para que cada nó envie a mensagem
  - Se não for possível estipular este prazo, o problema não tem solução

# ALGORITMO DA SOLUÇÃO TRIVIAL

```
tipoPlano planoFinal
tipoPlano planos[numGenerais]
planos[IDLocal] = escolhaAtaqueRetirada();
Para todos os outros generais G
        send(G, IDLocal, planos[IDLocal]);
Fim_Para
Para todos os outros generais G
        receive(G, planos[G]);
Fim_Para
planoFinal =  maioria(planos);
```

- Valores para tipoPlano podem ser A (Ataque) ou R (Retirada)
- Cada general escolhe uma estratégia e envia para os demais
- Analisando a escolha dos demais, a estratégia final deve ser de acordo com a maioria
- Vamos analisar o que ocorre em um cenário de falha de quebra:
  - Considerando 3 generais, sendo:
    - 2 leais (Leo e Zoe)
    - 1 traidor (Basil)
  - Zoe e Basil escolhem ataque e Leo escolhe Retirada
  - Zoe e Leo funcionam normalmente

- Basil quebra após enviar uma mensagem a Leo, mas antes de enviar a Zoe, resultando nas tabelas:




| Leo | |
|---|---|
| general | plan |
| Basil | A |
| Leo | R |
| Zoe | A |
| majority | A |

| Zoe | |
|---|---|
| general | plans |
| Basil | – |
| Leo | R |
| Zoe | A |
| majority | R |

- A decisão final de Leo é **Ataque**, enquanto que para Zoe a decisão é **Retirada**!

# ALGORITMO DOS GENERAIS BIZANTINOS

- O problema do algoritmo anterior vem do fato que não usa o fato de que certos generais são leais e outros não
- Em um sistema distribuído, não há como um nó determinar quais são os traidores diretamente
- É preciso assegurar que a estratégia escolhida pelos generais traidores não afete a escolha de estratégia dos generais leais
- Para tanto, são introduzidas rodadas extras de envio de mensagens

- Na **primeira rodada** de mensagens, cada nó deve enviar a sua própria estratégia

- Na **segunda rodada**, cada nó deve enviar o que recebeu dos outros generais sobre as suas estratégias

    - Os generais leais sempre replicam exatamente as mesmas estratégias que receberam

    - Além disso, para reduzir as mensagens, cada general não precisa enviar:
        - A sua própria escolha para si mesmo
        - A escolha feita de um general para o próprio

```
tipoPlano planoFinal
tipoPlano planos[numGenerais]
tipoPlano planRep[numGenerais][numGenerais]
tipoPlano planMaioria[numGenerais]
planos[IDLocal]= escolhaAtaqueRetirada();
 // Primeira rodada de mensagens
Para todos os outros generais G
        send(G, IDLocal, planos[IDLocal]);
Fim_Para
Para todos os outros generais G
        receive(G, planos[G]);
Fim_Para
```

```
// Segunda rodada de mensagens
Para todos os outros generais G
    Para todos os outros generais H exceto G
        send(H, IDLocal,G, planos[G]);
    Fim_Para
Fim_Para
Para todos os outros generais G
    Para todos os outros generais H exceto G
        receive(G,H, planRep[G][H]);
    Fim_Para
Fim_Para
```

```
// Define primeiro voto de cada nó pela maioria
Para todos os outros generais G
    planMaioria[G] = maioria(planos[G], planRep[*][G])
Fim_Para
planMaioria[IDLocal] = planos[IDLocal];
// Define o consenso, pela maioria
planoFinal =  maioria(planMaioria);
```

- Vamos analisar um cenário de falha de quebra com 3 generais, descrito anteriormente:
  - Zoe e Leo, generais leais, decidem por ataque e retirada

- Basil, general traidor, decide por ataque. Envia a mensagem para Leo, mas quebra antes de enviar a Zoe

- Os dados com Leo, ficam:

| Leo | | | | |
|---|---|---|---|---|
| general | plan | reported by | | majority |
| | | Basil | Zoe | |
| Basil | A | | – | A |
| Leo | R | | | R |
| Zoe | A | – | | A |
| majority | | | | A |

- Estratégia final: Ataque

- Os dados com Zoe, ficam:

| Zoe | | | | |
|---|---|---|---|---|
| general | plan | reported by | | majority |
| | | Basil | Leo | |
| Basil | – | | **A** | A |
| Leo | R | – | | R |
| Zoe | A | | | A |
| majority | | | | A |

- Estratégia final: Ataque

- Em um segundo cenário, supondo que o traidor:
  - Mande todas as mensagens da primeira rodada
  - Envie apenas a primeira mensagem da segunda rodada e depois quebre
- Dados de Leo:

| Leo | | | | |
|---|---|---|---|---|
| general | plan | reported by | | majority |
| | | Basil | Zoe | |
| Basil | A | | A | A |
| Leo | R | | | R |
| Zoe | A | A | | A |
| majority | | | | A |

Estratégia Final: Ataque

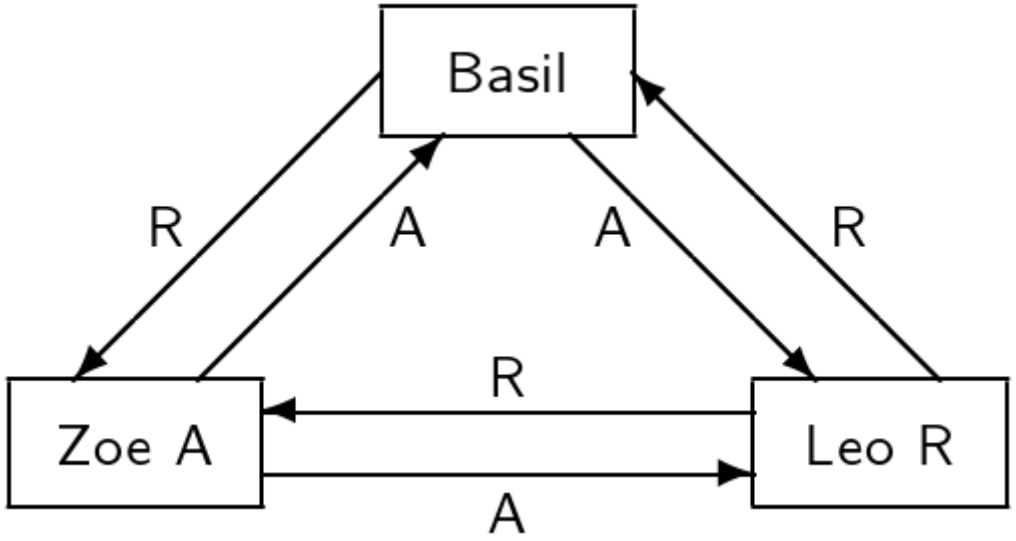
- Dados de Zoe:

| Zoe | | | | |
|---|---|---|---|---|
| general | plan | reported by | | majority |
| | | Basil | Leo | |
| Basil | A | | A | A |
| Leo | R | – | | R |
| Zoe | A | | | A |
| majority | | | | A |

Estratégia Final:
Ataque

## Falhas Bizantinas com 3 generais

- Vimos que o algoritmo consegue o consenso no caso de falhas de quebra de um general em três
- Vamos ver um cenário em que o traidor ocasiona uma falha bizantina, ou seja:
    - Uma mensagem aleatória é enviada
- A primeira rodada de mensagens fica:



- Usando o algoritmo trivial (uma rodada) temos decisões diferentes:

| Leo | |
|---|---|
| general | plans |
| Basil | A |
| Leo | R |
| Zoe | A |
| majority | A |

| Zoe | |
|---|---|
| general | plans |
| Basil | R |
| Leo | R |
| Zoe | A |
| majority | R |

- Para o algoritmo dos generais bizantinos (duas rodadas):
  - Basil indica o mesmo plano para Zoe e Leo na primeira rodada
  - Indica o plano correto de Leo para Zoe
  - Indica o plano errado de Zoe para Leo

| Leo | | | | |
|---|---|---|---|---|
| general | plans | reported by | | majority |
| | | Basil | Zoe | |
| Basil | A | | A | A |
| Leo | R | | | R |
| Zoe | A | **R** | | R |
| majority | | | | R |

| Zoe | | | | |
|---|---|---|---|---|
| general | plans | reported by | | majority |
| | | Basil | Leo | |
| Basil | A | | A | A |
| Leo | R | R | | R |
| Zoe | A | | | A |
| majority | | | | A |

  - Neste caso, decisões diferentes são tomadas.

**Falhas Bizantinas com 4 generais**

- Neste caso, estamos considerando a existência de mais um general leal

- Neste cenário temos:
  - Basil, John e Leo – generais leais
    - Basil e John escolhem Ataque
    - Leo escolhe retirada

  - Zoe – general traidor

- Vamos analisar os dados parciais obtidos por Basil

| Basil | | | | | |
|---|---|---|---|---|---|
| general | plan | reported by | | | majority |
| | | John | Leo | Zoe | |
| Basil | A | | | | A |
| John | **A** | | **A** | ? | A |
| Leo | R | R | | ? | R |
| Zoe | ? | ? | ? | | ? |
| majority | | | | | ? |

- Observe que as mensagens enviadas por Zoe não podem mudar o resultado da maioria para John e Leo

- Vamos considerar agora que Zoe mande mensagens na primeira rodada indicando:
  - Para John que seu plano é Ataque
  - Para os demais indique que seu plano é Retirada

| Basil | | | | | |
|---|---|---|---|---|---|
| general | plans | reported by | | | majority |
| | | John | Leo | Zoe | |
| Basil | A | | | | A |
| John | A | | A | ? | A |
| Leo | R | R | | ? | R |
| Zoe | **R** | **A** | **R** | | R |
| | | | | | R |

- Como os demais generais são leais, todos terão exatamente as mesmas opções indicadas por Zoe.

- Todas as quatro opções são iguais para os 3 generais.
- Escolherão a mesma estratégia!

## Complexidade do algoritmo dos Generais Bizantinos

- Para cada traidor adicional, é necessário acrescentar uma rodada adicional de mensagens
- O número total de generais deve ser $3t+1$, sendo $t$ o número de traidores
- Cada general manda:
  - Na primeira rodada: (n-1) mensagens
  - Na segunda rodada: (n-1)*(n-2) mensagens
  - Na terceira rodada: (n-2)*(n-3) mensagens
  - e assim por diante

- Como são $n$ generais, o total de mensagens enviadas é:

$$n \cdot \left[ (n-1) + \sum_{k=1}^{t} (n-k) \cdot (n-k-1) \right]$$

- Assim, com o aumento do número de traidores, a quantidade de mensagens cresce rapidamente

| **Traidores** | **Generais** | **Mensagens** |
|---|---|---|
| 1 | 4 | 36 |
| 2 | 7 | 392 |
| 3 | 10 | 1790 |
| 4 | 13 | 5408 |

# ALGORITMO REI

- Modificação do algoritmo dos generais bizantinos
- Requer menos mensagens quando o número de traidores aumenta
- Requer um general leal adicional por traidor, ou seja:
  - Número total de generais é $4t+1$, sendo $t$ número de traidores
- O algoritmo baseia-se no fato de que poucos traidores não afetarão o consenso se houver uma maioria esmagadora
- Para um traidor, há duas etapas, cada uma com duas rodadas de envio de mensagens

- Em **cada etapa**:
  - Um dos nós é escolhido Rei
  - Cada nó envia para os demais o plano que escolheu
  - É escolhido um plano por maioria e guardado o número de votos
  - Se o nó for Rei
    - Envia a sua maioria para os demais
    - A sua maioria passa a ser o seu próprio plano
  - senão
    - Recebe a maioria do Rei

- Se a própria maioria teve cotação esmagadora:
  - O sua maioria passa a ser o seu plano
- senão
  - A maioria do rei passa a ser o seu plano

- Na execução da segunda etapa, o Rei deve ser um general diferente da primeira
- Como há apenas um traidor, pelo menos um dos Reis será leal

- O algoritmo Rei está descrito a seguir:

```
tipoPlano planoFinal, minhaMaioria, planoRei
tipoPlano planos[numGenerais]
int votosMaioria, IDRei
planos[IDLocal]= escolhaAtaqueRetirada();
Repita duas vezes
      // Primeira rodada de mensagens
      Para todos os outros generais G
            send(G, IDLocal, planos[IDLocal]);
      Fim_Para
      Para todos os outros generais G
          receive(G, planos[G]);
      Fim_Para
```

```
 minhaMaioria = maioria(planos)
 votosMaioria = qtdadeVotosMaioria()
 // Segunda rodada de mensagens
Se for minha vez de ser Rei então
    Para todos os outros generais G
        send(G, IDLocal, minhaMaioria)
    Fim_Para
     planos[IDLocal]= minhaMaioria
senão
       receive(IDRei, planosRei);
Fim_se
```

```
            Se (votosMaioria>3) então
                planos[IDLocal]= minhaMaioria
            senão
                planos[IDLocal]=planoRei
            Fim_se
    Fim_Repita
    planoFinal =  planos[IDLocal]
```

- Vamos considerar um cenário com 5 generais:
  - Leais:
    - Basil e John escolhendo Ataque
    - Leo e Zoe escolhendo Retirada
  - Traidor: Mike – Manda dois avisos de cada (A e R)

- Após a primeira rodada de mensagens temos:

| Basil | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| A | A | R | R | R | R | 3 | |

| John | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| A | A | R | A | R | A | 3 | |

| Leo | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| A | A | R | A | R | A | 3 | |

| Zoe | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| A | A | R | R | R | R | 3 | |

- Tendo sido Zoe escolhido primeiro Rei, e como ninguém teve votação esmagadora, temos:

| Basil | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | | | | | | | R |

| John | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | R | | | | | | R |

| Leo | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | | R | | | | | R |

| Zoe | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | | | | R | | | |

- Na terceira rodada, todos obtém maioria esmagadora:

| Basil | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | R | R | ? | R | R | 4–5 | |

| John | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | R | R | ? | R | R | 4–5 | |

| Leo | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | R | R | ? | R | R | 4–5 | |

| Zoe | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | R | R | ? | R | R | 4–5 | |

- Qualquer que seja o voto de Mike, a opção da maioria não é alterada
- Observe também que o segundo Rei não vai alterar o consenso obtido
- Consideremos agora a situação em que o primeiro Rei é o traidor
- Independente do que foi enviado na primeira rodada, o Rei traidor pode mandar qualquer plano para os demais
    - Por exemplo: A para Leo e John e R para Basil e Zoe

- Após a segunda rodada, temos (não há maioria esmagadora):

| Basil | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | | | | | | | R |

| John | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | A | | | | | | A |

| Leo | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | | A | | | | | A |

| Zoe | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | | | | R | | | R |

- Na segunda etapa, o Rei é um general leal (por exemplo, Zoe)

| Basil | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | A | A | ? | R | ? | 3 | |

| John | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | A | A | ? | R | ? | 3 | |

| Leo | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | A | A | ? | R | ? | 3 | |

| Zoe | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| R | A | A | ? | R | ? | 3 | |

- Neste caso, a opção da maioria depende da opção de Mike, mas não será maioria em nenhum caso

- Suponhamos que o novo Rei obteve A

- Este plano será enviado aos demais (**igual para todos**)

- Logo todos terão consenso na mesma opção.

| Basil | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| A | | | | | | | A |

| John | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | A | | | | | | A |

| Leo | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | | A | | | | | A |

| Zoe | | | | | | | |
|---|---|---|---|---|---|---|---|
| Basil | John | Leo | Mike | Zoe | myMajority | votesMajority | kingPlan |
| | | | | A | | | |

## Complexidade do Algoritmo do Rei

- O algoritmo requer um general a mais: $4t+1$, sendo $t$ o número de traidores

- Requer **menos** mensagens:
  - Na primeira rodada: $n*(n-1)$ mensagens
  - Na segunda rodada: $(n-1)$ mensagens (apenas o Rei manda)
  - Além disso, são necessárias $t+1$ etapas de duas rodadas

  - Logo, o total é: $(t+1)*(n+1)*(n-1)$ mensagens

- Comparando os dois algoritmos:

| | **Generais Bizantinos** | | **Algoritmo Rei** | |
|---|---|---|---|---|
| **Traidores** | **Generais** | **Mensagens** | **Generais** | **Mensagens** |
| **1** | 4 | 36 | 5 | 48 |
| **2** | 7 | 392 | 9 | 240 |
| **3** | 10 | 1790 | 13 | 672 |
| **4** | 13 | 5408 | 17 | 1440 |