

Design de protocolos

Aula para disciplina de Métodos Formais

Gabriela Moreira

Departamento de Ciência da Computação - DCC
Universidade do Estado de Santa Catarina - UDESC

19 de maio de 2025



Conteúdo

Design de protocolos

Problema dos dois generais

O problema dos generais bizantinos

Consenso

Outline

Design de protocolos

Problema dos dois generais

O problema dos generais bizantinos

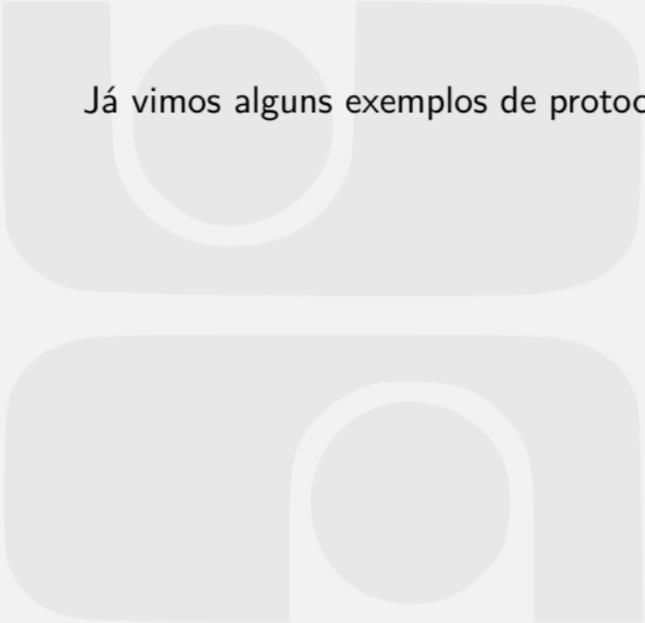
Consenso

Design de protocolos

A área de design de protocolos (*protocol design*) é possivelmente onde métodos formais que usam *model checking*, como Quint e TLA+, mais tem valor.

Protocolos

Já vimos alguns exemplos de protocolos:



Protocolos

Já vimos alguns exemplos de protocolos:

- Entregar a prova de cabeça pra baixo e pedir que todos desviem no mesmo momento é uma forma de garantir que todos tem o mesmo tempo de prova

Protocolos

Já vimos alguns exemplos de protocolos:

- Entregar a prova de cabeça pra baixo e pedir que todos desviem no mesmo momento é uma forma de garantir que todos tem o mesmo tempo de prova
- Na decolagem/pouso de um avião, precisamos abrir as janelas e levantar as mesinhas

Protocolos

Já vimos alguns exemplos de protocolos:

- Entregar a prova de cabeça pra baixo e pedir que todos desviem no mesmo momento é uma forma de garantir que todos tem o mesmo tempo de prova
- Na decolagem/pouso de um avião, precisamos abrir as janelas e levantar as mesinhas
- Sistemas de trocas em jogos ou, semelhantemente, sistemas de pagamento estilo mercado pago

Suposições (*Assumptions*)

Uma suposição é algo que é aceito como verdadeiro, sem provas.

Suposições (*Assumptions*)

Uma suposição é algo que é aceito como verdadeiro, sem provas.

Suposições são extremamente importantes para protocolos, e é necessário fazê-las explícitas.

- Quando alguém for implementar esse protocolo, deve garantir que as suposições valem para seu ambiente também

O que precisamos assumir nos nossos exemplos? - Provas

- Entregar a prova de cabeça pra baixo e pedir que todos desvirem no mesmo momento é uma forma de garantir que todos tem o mesmo tempo de prova

O que precisamos assumir nos nossos exemplos? - Provas

- Entregar a prova de cabeça pra baixo e pedir que todos desvirem no mesmo momento é uma forma de garantir que todos tem o mesmo tempo de prova
 - Assumimos que alunos não conseguem ler as questões de uma folha de papel virada.
 - Assumimos que os alunos não vão desobedecer e desvirar antes da hora

O que precisamos assumir nos nossos exemplos? - Aviões

- Na decolagem/pouso de um avião, precisamos abrir as janelas e levantar as mesinhas

O que precisamos assumir nos nossos exemplos? - Aviões

- Na decolagem/pouso de um avião, precisamos abrir as janelas e levantar as mesinhas
 - Nesse caso, o protocolo não é suficiente para garantir que os passageiros sobrevivam à uma emergência.
 - É uma medida de prevenção, não uma garantia.
 - Para que fornecesse uma garantia, teríamos que assumir que estar com as janelas abertas e mesinhas levantadas faça com que todos os passageiros possam desembarcar com segurança em qualquer emergência - o que não é verdade.

O que precisamos assumir nos nossos exemplos? - Trocas

- Sistemas de trocas em jogos ou, semelhantemente, sistemas de pagamento estilo mercado pago

O que precisamos assumir nos nossos exemplos? - Trocas

- Sistemas de trocas em jogos ou, semelhantemente, sistemas de pagamento estilo mercado pago
 - Assumimos que uma pessoa não pode fazer escolhas pela outra
 - Cada um só tem acesso e controle de sua própria conta



Outline

Design de protocolos

Problema dos dois generais

O problema dos generais bizantinos

Consenso

O problema dos dois generais - Contexto

Com imagens de (BROWN, 2022)

- 2 generais e seus exércitos acampam em montanhas ao redor de uma cidade inimiga
- O único jeito dos generais comunicarem entre si é enviando mensagens
 - Os mensageiros podem ser capturados pelo inimigo sem que os generais saibam!
- Para que a batalha seja vencida, os dois precisam atacar ao mesmo tempo. O ataque de um deles não é suficiente.

O problema dos dois generais

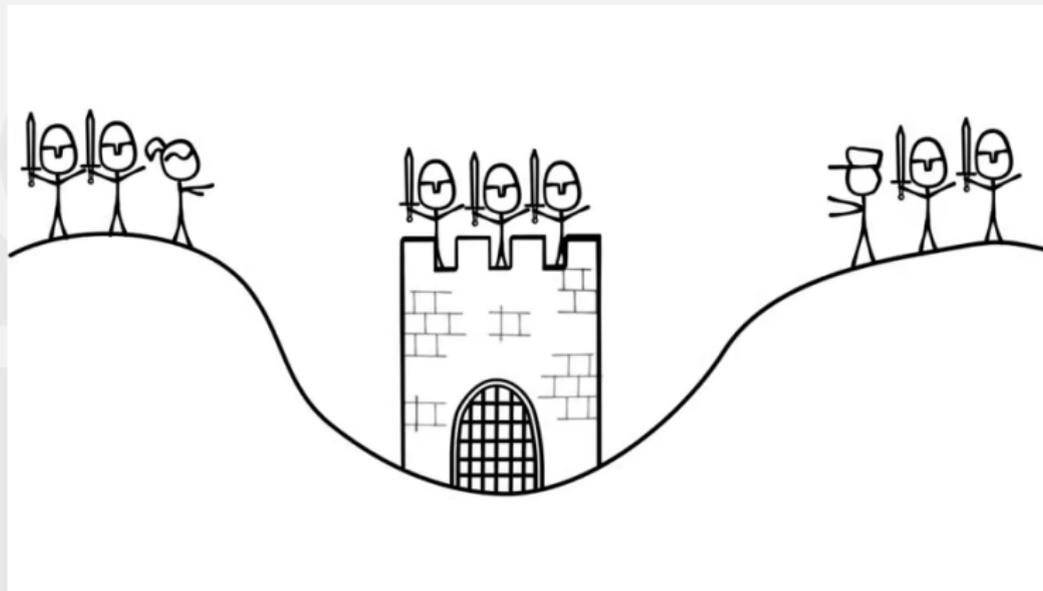
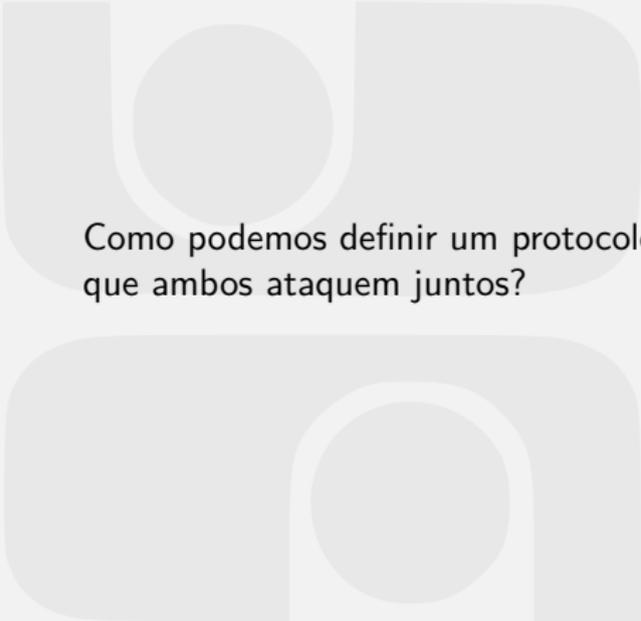


Figure 1: Fonte (BROWN, 2022)

O problema dos dois generais - protocolo



Como podemos definir um protocolo de troca de mensagens que garanta que ambos ataquem juntos?

Primeira mensagem

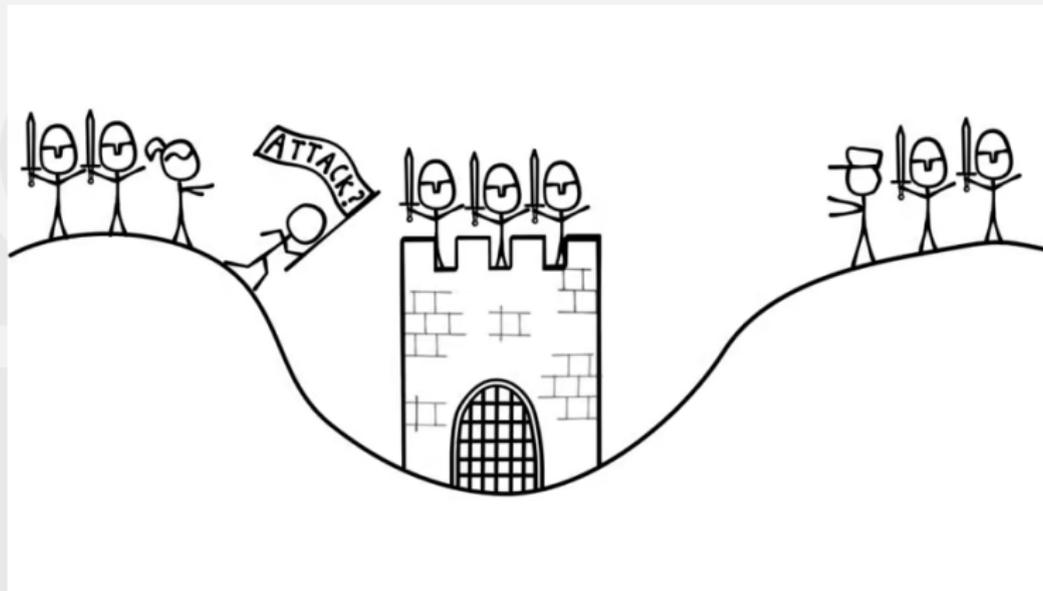


Figure 2: Fonte (BROWN, 2022)

Segunda mensagem

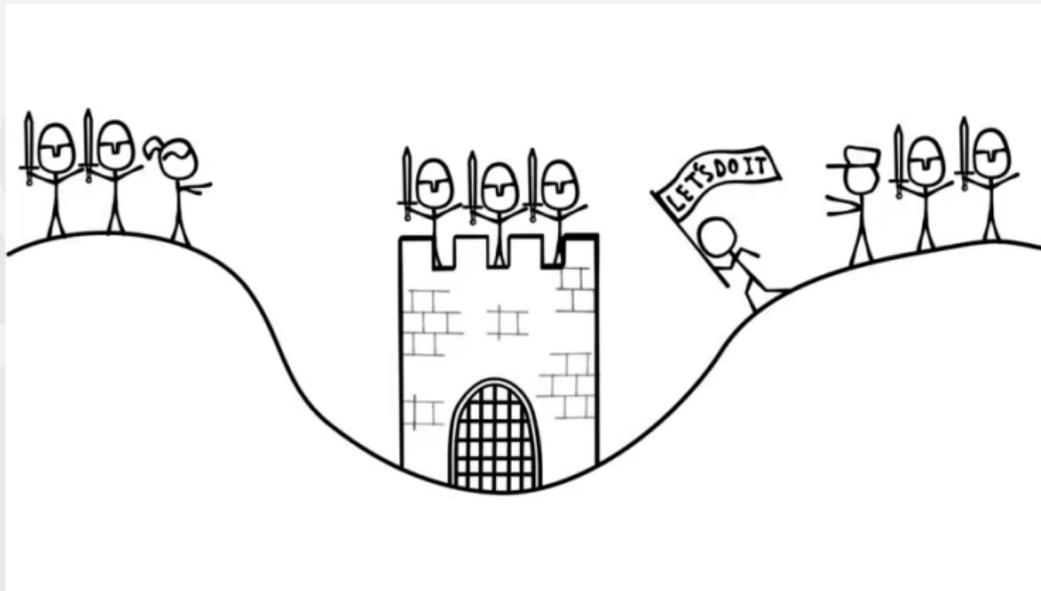


Figure 3: Fonte (BROWN, 2022)

Terceira mensagem

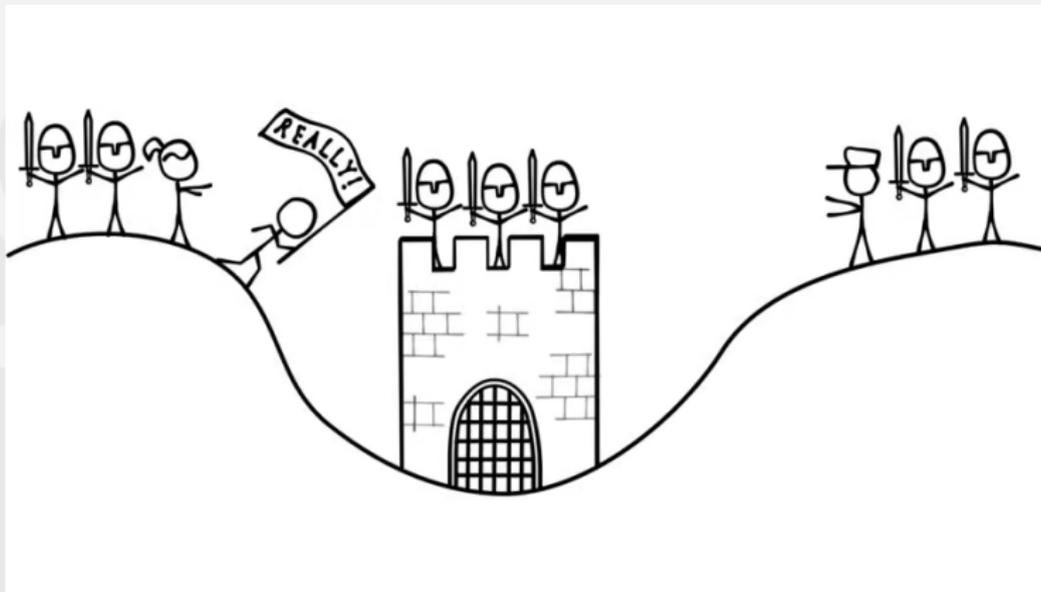


Figure 4: Fonte (BROWN, 2022)

Quarta mensagem

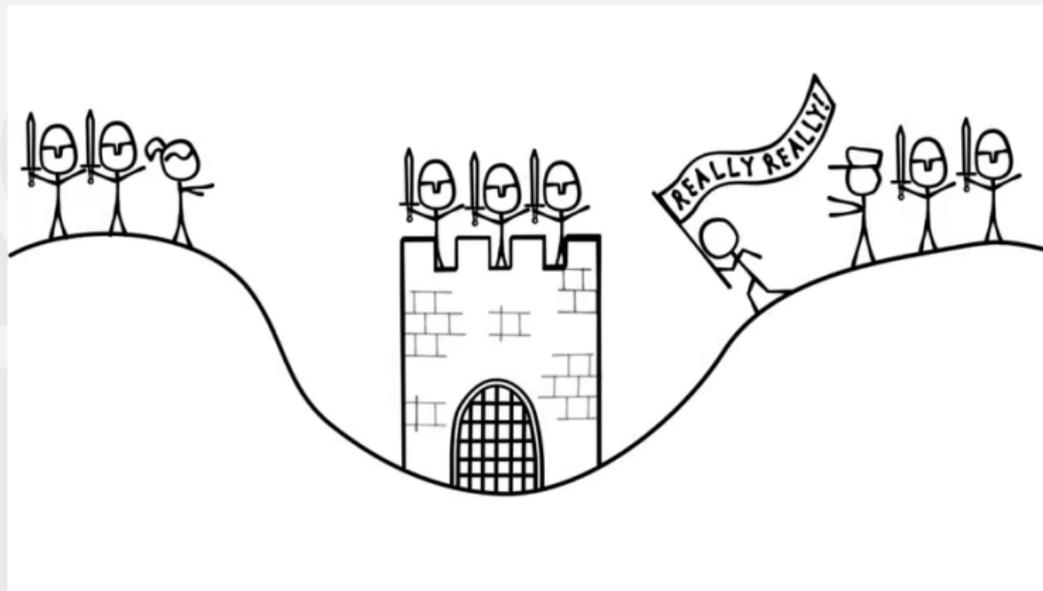
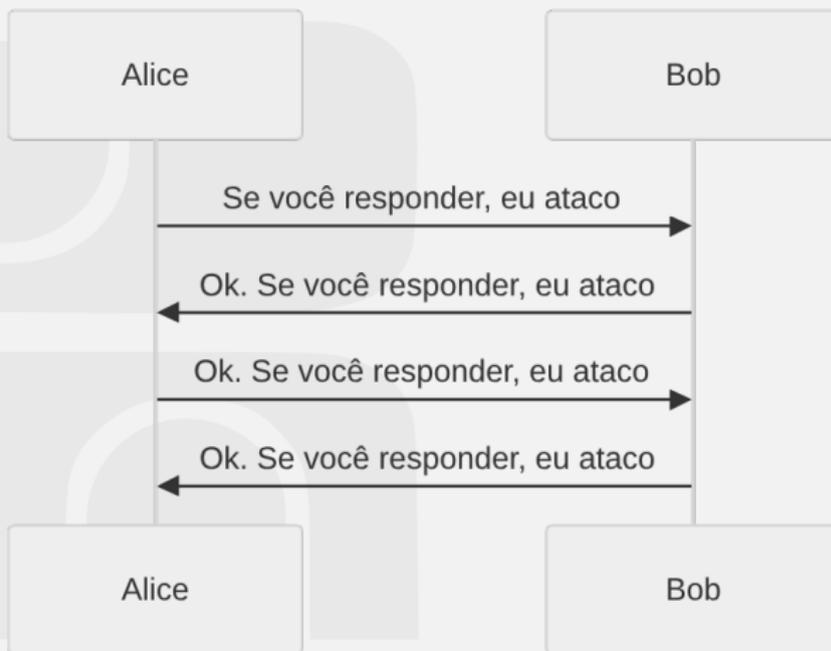


Figure 5: Fonte (BROWN, 2022)

Tentando algo como TCP/IP



Dois generais - Impossibilidade

O problema dos dois generais não tem solução!

- O melhor que podemos fazer é uma solução **estatística**: Envio uma mensagem de atacar amanhã as 9:00 e vou mandando mais mensagens de hora em hora até receber uma confirmação.



Outline

Design de protocolos

Problema dos dois generais

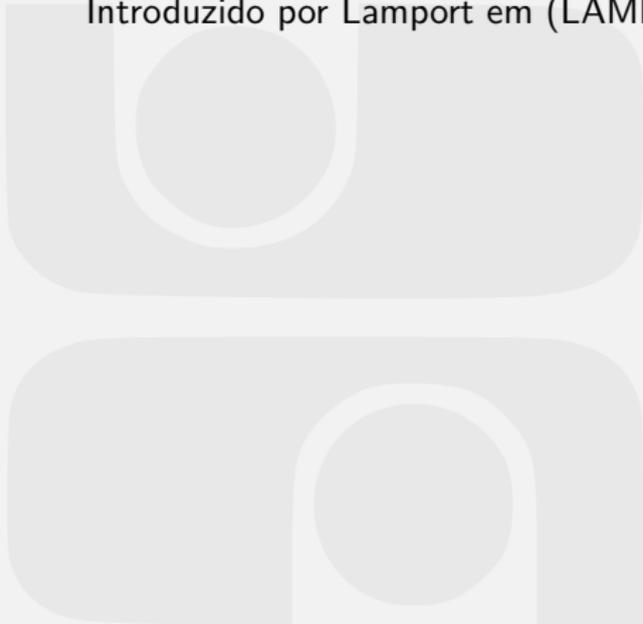
O problema dos generais bizantinos

Consenso



Histórico e motivação

Introduzido por Lamport em (LAMPOR; SHOSTAK; PEASE, 1982)





Histórico e motivação

Introduzido por Lamport em (LAMPORT; SHOSTAK; PEASE, 1982)

- sim, o mesmo Lamport criador de TLA+, uma década antes dele criar o TLA+.

Histórico e motivação

Introduzido por Lamport em (LAMPORT; SHOSTAK; PEASE, 1982)

- sim, o mesmo Lamport criador de TLA+, uma década antes dele criar o TLA+.

Um sistema de computadores confiável deve ser capaz de lidar com a falha de um ou mais de seus componentes. Um componente com falhas pode ser capaz de enviar **informações conflitantes** para diferentes partes do sistema.

Histórico e motivação

Introduzido por Lamport em (LAMPOR; SHOSTAK; PEASE, 1982)

- sim, o mesmo Lamport criador de TLA+, uma década antes dele criar o TLA+.

Um sistema de computadores confiável deve ser capaz de lidar com a falha de um ou mais de seus componentes. Um componente com falhas pode ser capaz de enviar **informações conflitantes** para diferentes partes do sistema.

- Lidar com esse tipo de falha é definido de forma abstrata pelo problema dos generais bizantinos

Histórico e motivação

Introduzido por Lamport em (LAMPORT; SHOSTAK; PEASE, 1982)

- sim, o mesmo Lamport criador de TLA+, uma década antes dele criar o TLA+.

Um sistema de computadores confiável deve ser capaz de lidar com a falha de um ou mais de seus componentes. Um componente com falhas pode ser capaz de enviar **informações conflitantes** para diferentes partes do sistema.

- Lidar com esse tipo de falha é definido de forma abstrata pelo problema dos generais bizantinos
- A capacidade de lidar com esse problema é chamada de *Byzantine Fault Tolerance* (BFT).

Histórico e motivação

Introduzido por Lamport em (LAMPORT; SHOSTAK; PEASE, 1982)

- sim, o mesmo Lamport criador de TLA+, uma década antes dele criar o TLA+.

Um sistema de computadores confiável deve ser capaz de lidar com a falha de um ou mais de seus componentes. Um componente com falhas pode ser capaz de enviar **informações conflitantes** para diferentes partes do sistema.

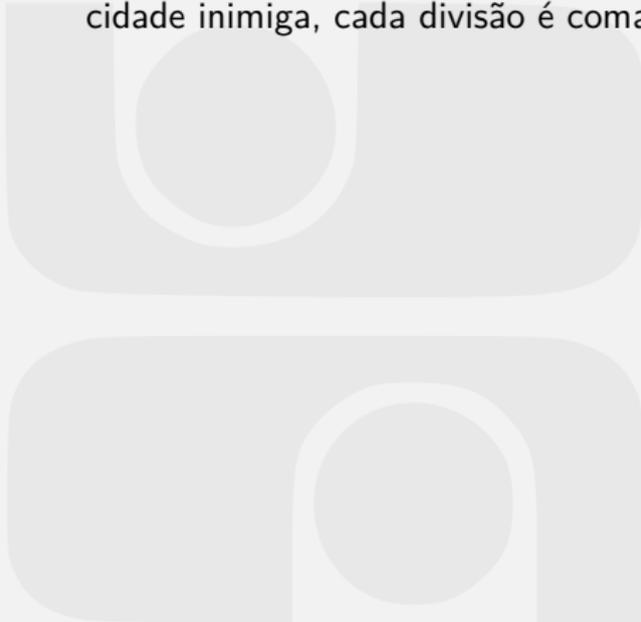
- Lidar com esse tipo de falha é definido de forma abstrata pelo problema dos generais bizantinos
- A capacidade de lidar com esse problema é chamada de *Byzantine Fault Tolerance* (BFT).

Recurso em vídeo: (COLOHAN, 2016)



O problema dos generais bizantinos

Várias divisões do exército bizantino estão acampadas ao redor de uma cidade inimiga, cada divisão é comandada por um general.





O problema dos generais bizantinos

Várias divisões do exército bizantino estão acampadas ao redor de uma cidade inimiga, cada divisão é comandada por um general.

- Os generais se comunicam entre si somente através de mensageiros

O problema dos generais bizantinos

Várias divisões do exército bizantino estão acampadas ao redor de uma cidade inimiga, cada divisão é comandada por um general.

- Os generais se comunicam entre si somente através de mensageiros
- Após observar o inimigo, eles devem decidir um plano de ação (i.e. atacar ou recuar)
 - Assim como no problema dos dois generais, um ataque com poucas divisões pode falhar



O problema dos generais bizantinos

Várias divisões do exército bizantino estão acampadas ao redor de uma cidade inimiga, cada divisão é comandada por um general.

- Os generais se comunicam entre si somente através de mensageiros
- Após observar o inimigo, eles devem decidir um plano de ação (i.e. atacar ou recuar)
 - Assim como no problema dos dois generais, um ataque com poucas divisões pode falhar
- Alguns generais podem ser traidores!
 - Traidores querem impedir os generais leais de entrarem em acordo sobre o plano de ação

O problema dos generais bizantinos

Várias divisões do exército bizantino estão acampadas ao redor de uma cidade inimiga, cada divisão é comandada por um general.

- Os generais se comunicam entre si somente através de mensageiros
- Após observar o inimigo, eles devem decidir um plano de ação (i.e. atacar ou recuar)
 - Assim como no problema dos dois generais, um ataque com poucas divisões pode falhar
- Alguns generais podem ser traidores!
 - Traidores querem impedir os generais leais de entrarem em acordo sobre o plano de ação

Precisamos de um algoritmo que garanta:

- A: Todos os generais leais devem decidir o mesmo plano de ação
- B: Um pequeno número de traidores não podem fazer com que os generais adotem um plano de ação ruim

Ilustração

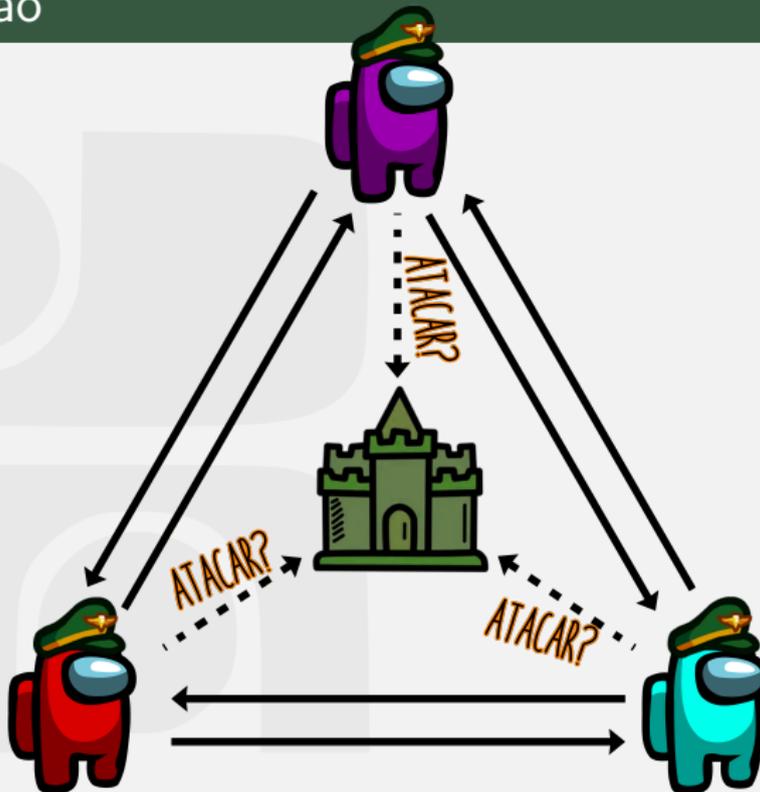


Ilustração II

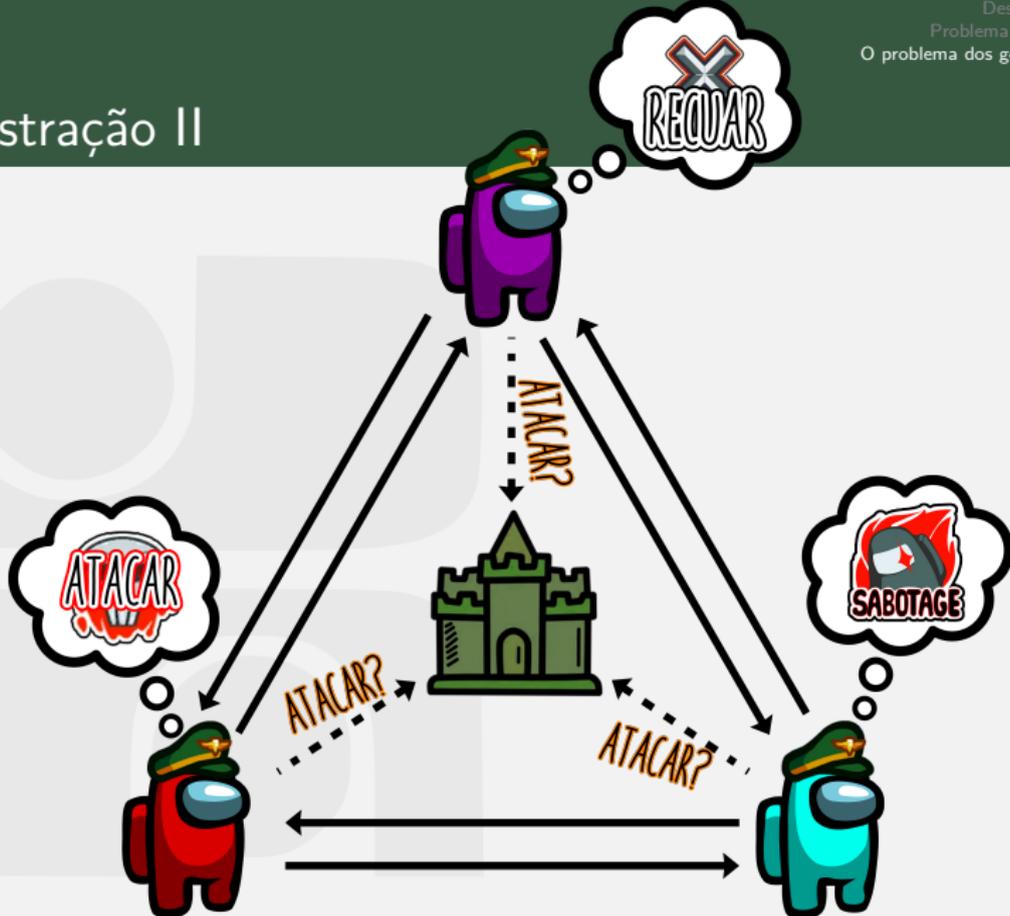


Ilustração III

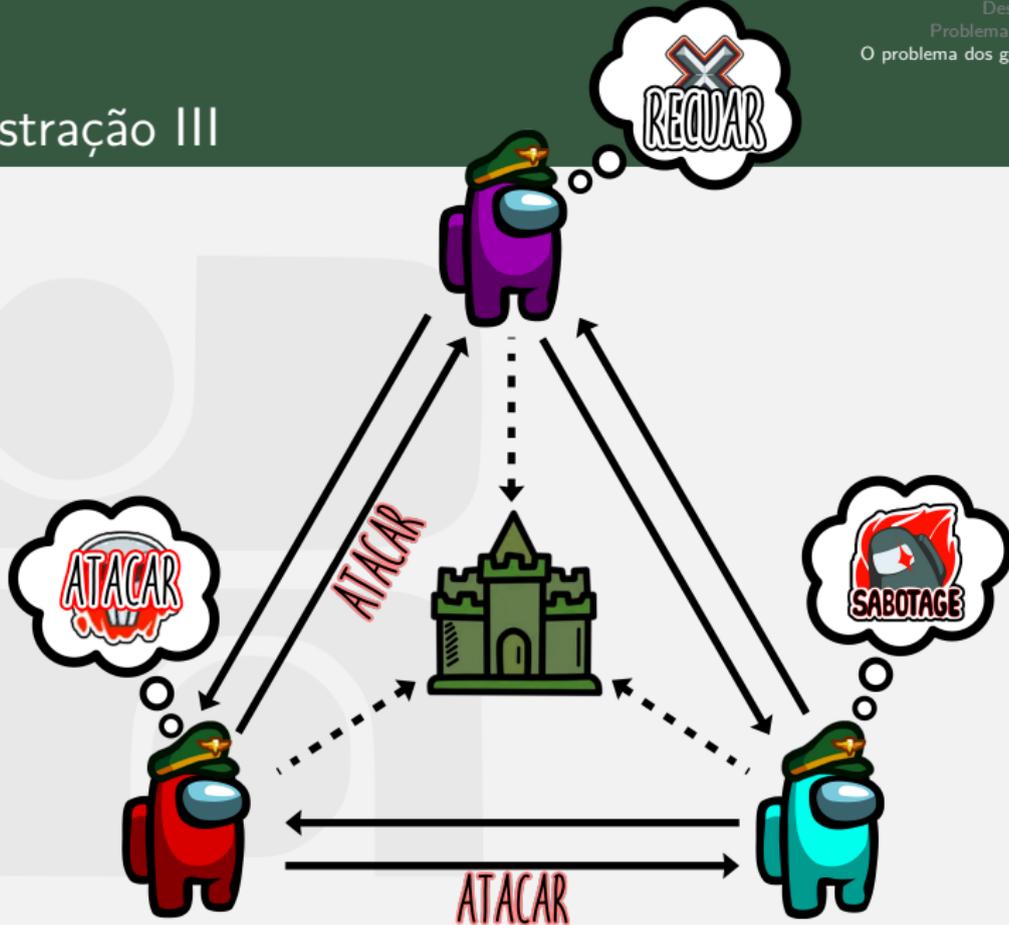


Ilustração IV

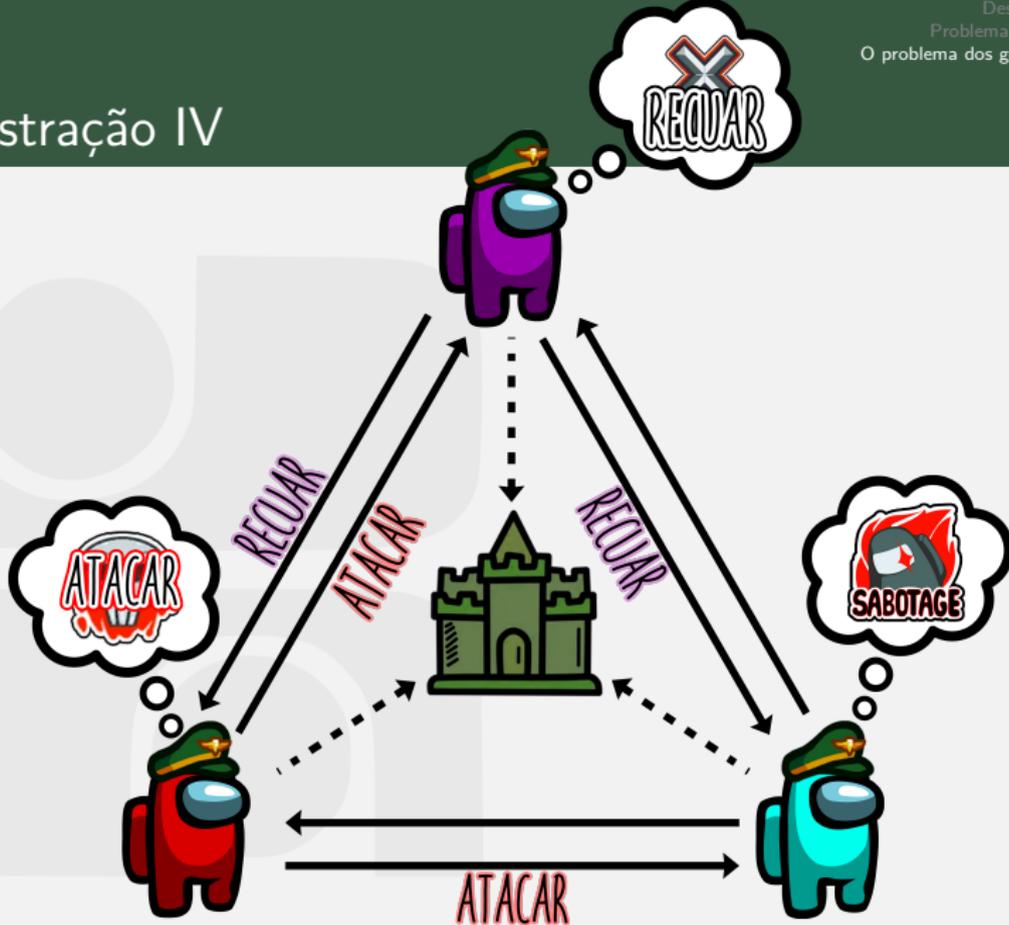


Ilustração V

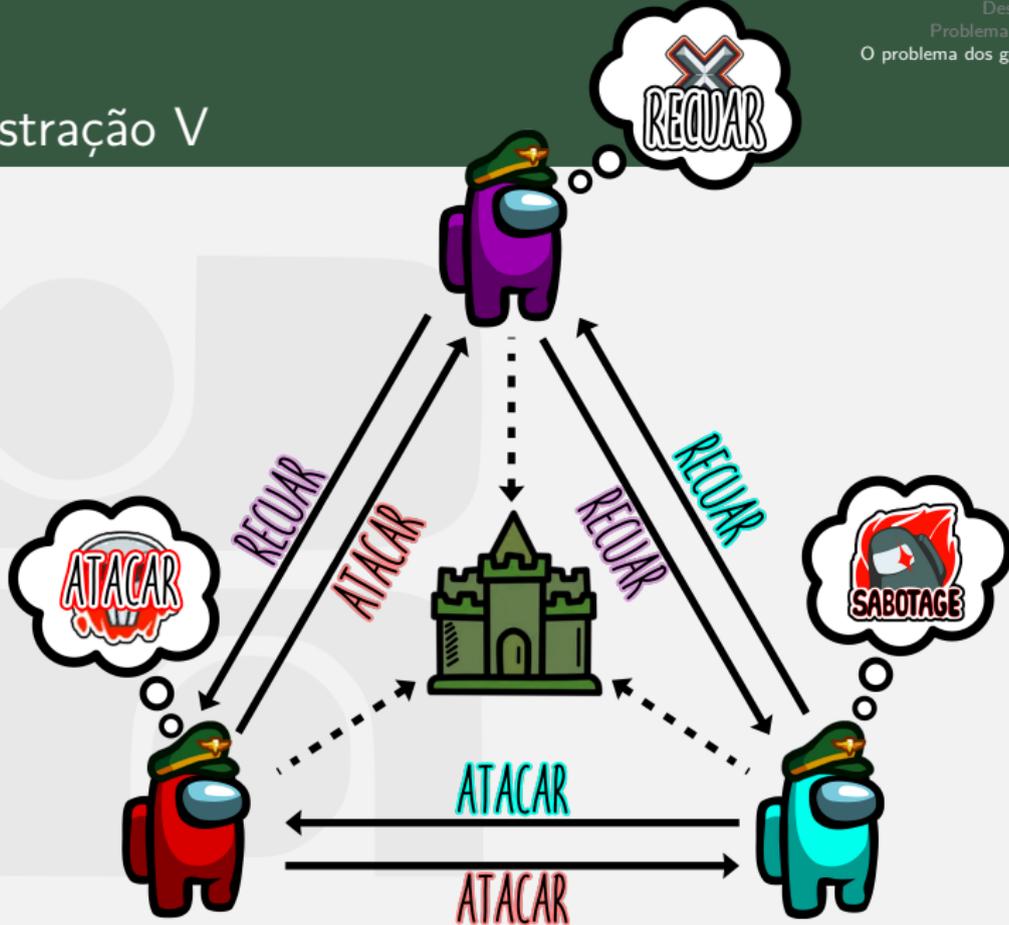
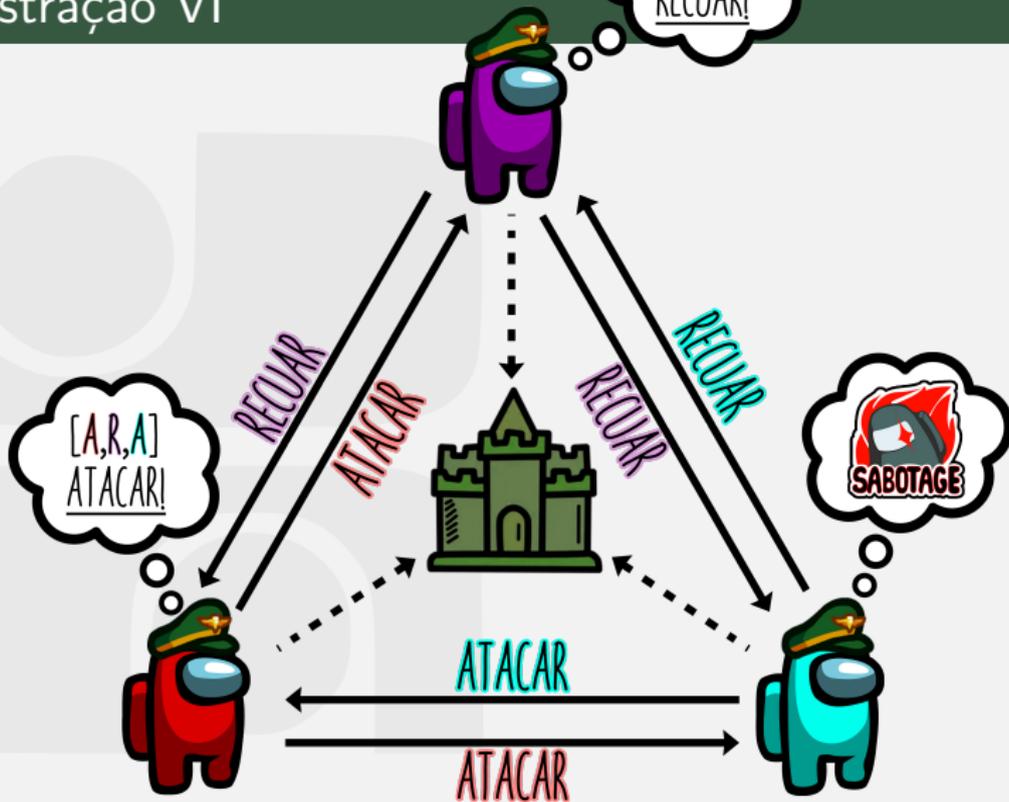


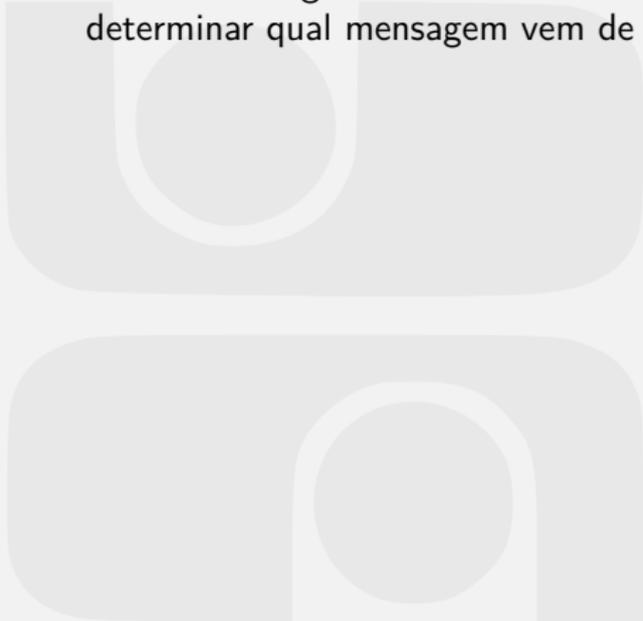
Ilustração VI





Impossibilidades

Mesmo se os generais leais souberem que há um traidor, não há como determinar qual mensagem vem de um traidor.



Impossibilidades

Mesmo se os generais leais souberem que há um traidor, não há como determinar qual mensagem vem de um traidor.

Segundo (LAMPORT; SHOSTAK; PEASE, 1982), mesmo que o problema pareça simples, sua dificuldade é indicada pelo fato de que, se os generais puderem apenas enviar mensagens orais, nenhuma solução funciona sem que **mais de 2/3** dos generais seja leal.

Impossibilidades

Mesmo se os generais leais souberem que há um traidor, não há como determinar qual mensagem vem de um traidor.

Segundo (LAMPORT; SHOSTAK; PEASE, 1982), mesmo que o problema pareça simples, sua dificuldade é indicada pelo fato de que, se os generais puderem apenas enviar mensagens orais, nenhuma solução funciona sem que **mais de 2/3** dos generais seja leal.

- Nenhuma solução com menos de $3m + 1$ generais no total consegue lidar com m traidores.

Impossibilidades

Mesmo se os generais leis souberem que há um traidor, não há como determinar qual mensagem vem de um traidor.

Segundo (LAMPORT; SHOSTAK; PEASE, 1982), mesmo que o problema pareça simples, sua dificuldade é indicada pelo fato de que, se os generais puderem apenas enviar mensagens orais, nenhuma solução funciona sem que **mais de 2/3** dos generais seja leal.

- Nenhuma solução com menos de $3m + 1$ generais no total consegue lidar com m traidores.
- Prova por contradição, mostrando que se houvesse uma solução pra esse caso, poderíamos usá-la para resolver a instância com três generais, o que é impossível.

Impossibilidades

Mesmo se os generais leis souberem que há um traidor, não há como determinar qual mensagem vem de um traidor.

Segundo (LAMPORT; SHOSTAK; PEASE, 1982), mesmo que o problema pareça simples, sua dificuldade é indicada pelo fato de que, se os generais puderem apenas enviar mensagens orais, nenhuma solução funciona sem que **mais de 2/3** dos generais seja leal.

- Nenhuma solução com menos de $3m + 1$ generais no total consegue lidar com m traidores.
- Prova por contradição, mostrando que se houvesse uma solução pra esse caso, poderíamos usá-la para resolver a instância com três generais, o que é impossível.

Se considerarmos mensagens assinadas (i.e. com criptografia) que não podem ser forjadas, o problema se torna mais simples e temos solução para o caso de três generais.

Modelos com diferentes parâmetros

Quando temos esse tipo de dependência do comportamento em uma suposição, é interessante checar/simular nosso modelo com diferentes parâmetros, que obedecem ou não a suposição.

Modelos com diferentes parâmetros

Quando temos esse tipo de dependência do comportamento em uma suposição, é interessante checar/simular nosso modelo com diferentes parâmetros, que obedecem ou não a suposição.

```
1 module TendermintModels {
2   import TendermintTest(
3     Corr = Set("p1", "p2", "p3"),
4     Faulty = Set("p4"),
5     // ...
6   ) as n4_f1 from "./TendermintTest"
7
8   import TendermintTest(
9     Corr = Set("p1", "p2"),
10    Faulty = Set("p3", "p4"),
11    // ...
12  ) as n4_f2 from "./TendermintTest"
13
14  import TendermintTest(
15    Corr = Set("p1", "p2", "p3"),
16    Faulty = Set("p4", "p5")
17  ) as n4_f3 from "./TendermintTest"
18 }
```



Relação com nosso trabalho 1

No trabalho 1, podemos garantir que os personagens sempre sobrevivem se houver um único inimigo.

- Poderíamos tentar encontrar o número mínimo de inimigos para que os personagens sempre morram.

Relação com nosso trabalho 1

No trabalho 1, podemos garantir que os personagens sempre sobrevivem se houver um único inimigo.

- Poderíamos tentar encontrar o número mínimo de inimigos para que os personagens sempre morram.

Assim, nosso protocolo (estratégia) para o trabalho funciona para batalhas contra um inimigo.



Relação com nosso trabalho 1

No trabalho 1, podemos garantir que os personagens sempre sobrevivem se houver um único inimigo.

- Poderíamos tentar encontrar o número mínimo de inimigos para que os personagens sempre morram.

Assim, nosso protocolo (estratégia) para o trabalho funciona para batalhas contra um inimigo.

- O fato de haver apenas um inimigo é uma suposição

Relação com nosso trabalho 1

No trabalho 1, podemos garantir que os personagens sempre sobrevivem se houver um único inimigo.

- Poderíamos tentar encontrar o número mínimo de inimigos para que os personagens sempre morram.

Assim, nosso protocolo (estratégia) para o trabalho funciona para batalhas contra um inimigo.

- O fato de haver apenas um inimigo é uma suposição
- Se formos apresentar esse protocolo (estratégia) para alguém, precisamos deixar clara essa suposição

Relação com nosso trabalho 1

No trabalho 1, podemos garantir que os personagens sempre sobrevivem se houver um único inimigo.

- Poderíamos tentar encontrar o número mínimo de inimigos para que os personagens sempre morram.

Assim, nosso protocolo (estratégia) para o trabalho funciona para batalhas contra um inimigo.

- O fato de haver apenas um inimigo é uma suposição
- Se formos apresentar esse protocolo (estratégia) para alguém, precisamos deixar clara essa suposição
- Simular/checar o protocolo (estratégia) em um ambiente onde a suposição não é satisfeita nos ajuda a entender por que ela existe.

Relações do trabalho 1 com sistemas distribuídos I

- 1 Inimigos são como generais bizantinos.
 - Podemos controlar o que os personagens fazem, mas não podemos controlar os inimigos.
 - Não podemos assumir que generais bizantinos estão seguindo um protocolo, assim como não podemos assumir que inimigos seguem uma estratégia.
- 2 Protocolos funcionam a partir de suposições, como a suposição de que há somente um inimigo.
 - Até 1 inimigo, propriedade garantida
 - 2 ou mais inimigos, propriedade não é garantida
- 3 Em muitas situações, um protocolo deve funcionar independentemente da ordem de chegada das mensagens.
 - Assim como a estratégia do trabalho deve funcionar para todas as possíveis ordens (iniciativas).



Relações do trabalho 1 com sistemas distribuídos II

- ④ Em protocolos, é comum identificarmos um “caminho feliz”. Uma ordem em que as coisas acontecem onde não é necessário usar as partes mais complicadas do protocolo.
 - No nosso jogo, a ordem “bardo, necromante, sacerdote, monstro” é um caminho feliz.
- ⑤ Em muitos (senão todos) protocolos de sistemas distribuídos, temos *timeouts*: ações que são disparadas após certo tempo para prevenir problemas como *deadlocks* por mensagens perdidas.
 - Esses *timeouts* são parecidos com o ritual do inimigo no nosso trabalho: ou as coisas acontecem dentro de um certo tempo, ou a ação tomada será aquela do *timeout* (todos os personagens morrem).



Outline

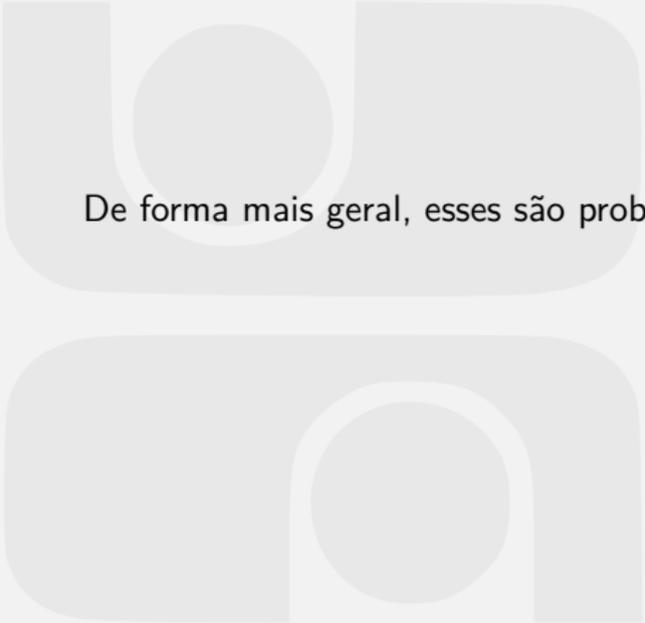
Design de protocolos

Problema dos dois generais

O problema dos generais bizantinos

Consenso

Consenso



De forma mais geral, esses são problemas de consenso.

Consenso

De forma mais geral, esses são problemas de consenso.

Não fiz figurinhas pra esse, vamos ver as imagens do vídeo da Heidi Howard pro Computerphile (HOWARD, 2016).

Referências

BROWN, S. A. **The two generals problem**. Disponível em:

<<https://linuxblog.io/the-two-generals-problem/>>.

COLOHAN, C. **L6: Byzantine fault tolerance**. Disponível em:

<https://www.youtube.com/watch?v=_e4wNoTV3Gw>.

HOWARD, H. **Consensus & organising coffee - computerphile**.

Disponível em:

<<https://www.youtube.com/watch?v=jn3DBzr--0k>>.

LAMPORT, L.; SHOSTAK, R.; PEASE, M. The byzantine generals problem. **Acm transactions on programming languages and systems**, p. 382–401, 1982.

Design de protocolos

Aula para disciplina de Métodos Formais

Gabriela Moreira

Departamento de Ciência da Computação - DCC
Universidade do Estado de Santa Catarina - UDESC

19 de maio de 2025