Os processos precisam entrar em acordo. Cada processo propõe um valor e os processos devem concordar sobre o correto. Usada por exemplo para decidir qual processo irá executar sua região crítica.

Acordo e problemas relacionados

- Processos devem concordar com um valor depois de um ou mais processos proporem qual deve ser o valor.

  Problemas aqui estudados
  - Generais Bizantinos
  - Consistência interativa
  - Multicast totalmente ordenado

Modelo do sistema e definições

- N processos se comunicando através de troca de mensagens.

Após uma quantidade de mensagens ser trocadas, deve-se chegar a um consenso.

- Chega-se a um consenso mesmo na presença de falhas.
- Comunicação confiável e processos podem falhar

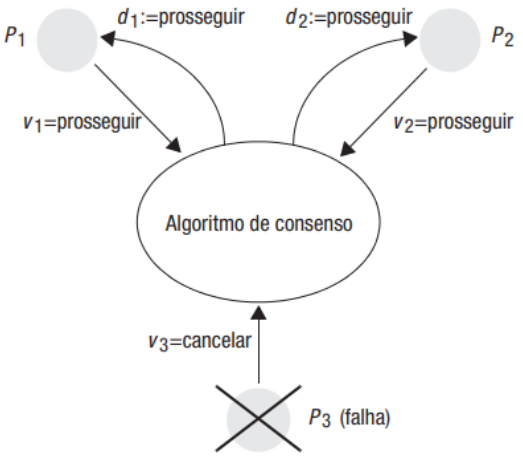
Deve-se garantir a entrega de mensagens mesmo havendo falhas de processos.

- Definição do problema:
  - Cada processo P_i começa indeciso e propõe um valor V_i
  - Valores são trocados entre os processos.
  - Cada processo configura a variável de decisão d_i e muda o estado para decidido.

Cada processo possui uma variável de decisão que é definida durante o acordo.

Após a decisão, a variável não pode mais ser alterada, e ao fim se chega a um consenso.

Consenso de três processos



A maioria "venceu", como o p3 falhou assim que ele voltar ao funcionamento será informado sobre o consenso.

Requisitos de um algoritmo de consenso

- Término: cada processo correto configura sua variável de decisão.
- Acordo: O valor da decisão é o mesmo para todos os processos corretos.
- Integridade: se todos os processos corretos propuseram o mesmo valor, então qualquer processo correto no estado decidido escolheu esse valor.

GARANTIA DE CONSENSO:

Deve-se garantir que:

1. Todos processos tomaram uma decisão;
2. Que a decisão foi a mesma em todos os processos.
3. Se todos tomaram a mesma decisão, qualquer processo no estado definido escolheu o mesmo valor.

Problema do consenso

- Cada processo em um grupo de multicast confiável, envia o seu valor

- proposto para os membros do grupo.

- Ao receber os N valores, cada processo avalia usando funções como:
  - Majority – retorna valor mais frequente
  - Maximum – retorna o maior valor
  - Minimum – retorna o menor valor
- Término é garantido pela confiabilidade

Todas as mensagens são entregues, então todos os processos ao final tem sua variável de decisão definida.

- Acordo é garantido pela função majority

Os processos usam a função *majority* para verificar o valor mais frequente e tomar sua decisão, então se garante que haverá um acordo.

- Integridade pelo multicast confiável.

Todos os processos corretos tomam a mesma decisão.

- Possibilidade de falhas aumentam a complexidade e não fica claro o término.

- Em falhas arbitrárias (bizantinas), processos falhos podem comunicar valores aleatórios para os outros.

Processos que comunicam valores diferentes uns para os outros. Essa "falha" pode ser mal intencionada.

- Pode ser mal intencionada.

proposto pelo comandante.

- Blockchain resolve o problema dos generais bizantinos!

Problema dos generais bizantinos

- Três ou mais generais devem concordar com um ataque ou retirada.
- Um é o comandante e dá a ordem.
- Os generais devem decidir se atacam ou recuam.
- Um ou mais podem ser traidores (falho).
- Comandante traidor: faz um general atacar e outro recuar

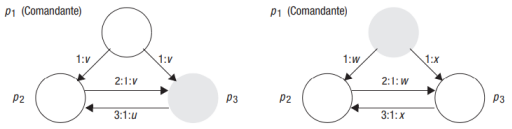
Neste caso nem todas as decisões podem ser íntegras, o problema é detectar se existem processos falhos.

- Requisitos
  - Término: cada processo configura sua variável de decisão
  - Acordo: O valor da decisão de todos os processos corretos é o mesmo.
  - Integridade: se o comandante está correto, todos os processos corretos decidem pelo valor

---

Problema dos generais bizantinos

Impossibilidade de acordo em caso de falha em um de três processos (mensagens não assinadas)



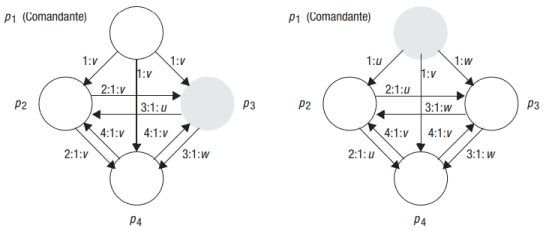
Os processos falhos aparecem em cinza

mensagens não assinadas: Um nó pode receber uma mensagem, alterar o conteúdo e enviar para o destino.

Sem assinatura não se pode saber quem é o comandante e não se chega a um consenso em caso de falha.

Solução com um processo falho

N=4, f=1

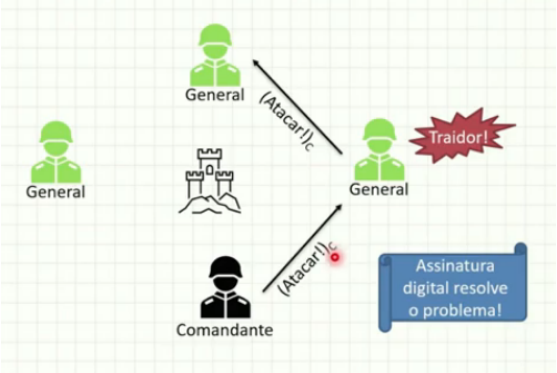


Os processos falhos aparecem em cinza

No caso de 4 processos sendo um deles falho, pode-se usar a função majority para se chegar a uma decisão.

Contudo no caso da falha no comandante, sem a assinatura não se pode usar majority então não há consenso.

Problema dos generais bizantinos



A maneira mais simples de resolver é usar **assinatura digital.** Dessa forma, a mensagem é assinada de uma maneira que caso o general traidor tente alterar consegue-se detectar a tentativa de alteração e é desconsiderada as mensagens vindas do traidor.

Consistência Interativa

- Cada processo propõe um único valor
- Processos corretos devem concordar com vetor de valores (vetor de decisão), com um valor para cada processo.

Um vetor com o valor de cada processo envolvido na tomada de decisão.

- Objetivo: cada processo do conjunto obter a mesma informação sobre seus respectivos estados.

O vetor de decisão precisa ser o mesmo para todos os processos envolvidos.

- Requisitos:
  - Término: cada processo configura sua variável de decisão.
  - Acordo: Vetor de decisão de todos os processos corretos é o mesmo.
  - Integridade: Se p, está correto, todos os processos corretos decidem por v, como o i-ésimo componente de seus vetores. ???

Consenso em um sistema síncrono

- Síncrono- Tempo máximo para envio de uma mensagem é conhecido.

- Suposição: No máximo f dos N processos apresentam falhas por colapso.

No máximo f processos falhos dentro de n processos.

- Cada processo reúne os valores propostos pelos outros processos.
- Uso do multicast para a entrega dos valores.

Usam multicast para envio dos valores.

- São necessárias f+1 rodadas para garantir que, no pior caso (f falhas), os processos corretos tenham condições de concordar.

Para se garantir consenso deve-se ter f+1 rodadas.

Consenso em blockchain

- O que é blockchain?
- Utiliza um algoritmo de consenso descentralizado chamado Proof of Work(PoW).
- Também conhecido como mineração.

O trabalho de consenso se chama mineração.

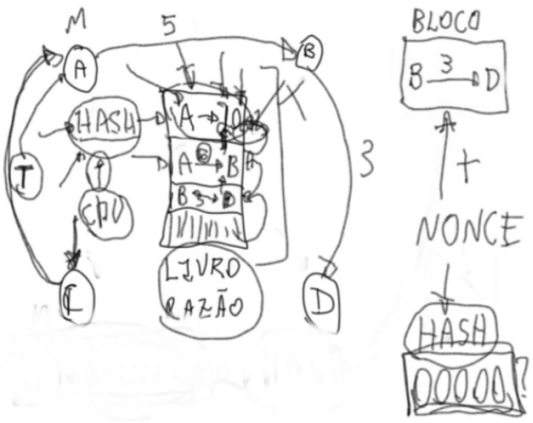
- Utiliza hash.

Para garantir a integridade dos dados.

- Os full nodes possuem toda a cópia do blockchain.

Eles guardam a cópia da blockchain (livro razão).

  - Responsável por validar as transações



Livro razão: Contém um registro do conjunto de operações financeiras que ocorreram.

Todos os nós participantes da blockchain tem conhecimento do livro razão.

Blockchain usa hash (cálculo matemático em cima de um conteúdo que gera um valor, qualquer alteração no conteúdo irá gerar um valor diferente).

Proof of work: exige que o minerador faça o cálculo da hash a partir de determinada regra (por exemplo cinco “0” no início).

Os **mineradores** calculam a transação + um nonce e o primeiro a conseguir chegar ao resultado desejado recebe uma “recompensa”, geralmente **bitcoins**. Além disso, ele adiciona a informação com a hash e a hash do bloco anterior (encadeamento) para evitar que

caso uma modificação feita num bloco anterior não haja quebra de cadeias.

Blockchain: Funcionamento

- Novas transações validadas são adicionadas a um banco de dados chamado mempool.

- O mempool é usado para criar um novo bloco candidato pelos mineradores.

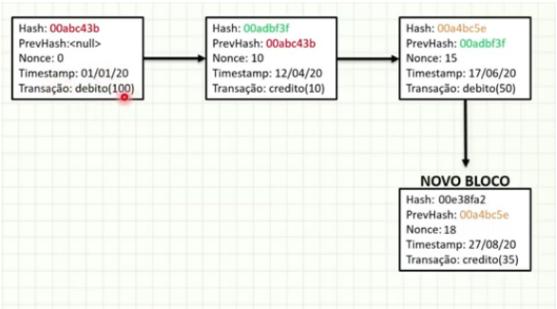
Usado como base para o cálculo de hash do bloco.

- O hash deste bloco candidato é calculado.

Além de adicionar a hash do bloco anterior.

- O próximo bloco é iniciado com este hash.
- Daí o nome blockchain (blocos encadeados através de hashes).

Blockchain: Funcionamento



Algoritmo de consenso: Proof of Work

- Usado pelos full nodes para o consenso da validade de um novo bloco gerado.

- O algoritmo adiciona um número aleatório chamado nonce ao final do bloco.

- Para aceitar um novo bloco de um minerador, exige-se que o mesmo descubra um nonce que gere um hash com uma determinada quantidade de zeros no seu início.
  Daí o custo da mineração.
- Este requisito exige um razoável custo computacional. Isso desestimula a tentativa de burlar o algoritmo gerando blocos falsos.

- Demonstração em python!
