Problema dos Generais Bizantinos

No estudo de sistemas distribuídos é importante estudar o que pode acontecer quando um ou mais computadores deixam de funcionar, ou pior, passam a funcionar de maneira inesperada, ou pior ainda, passam a falhar de maneira intermitente.

Esta questão foi proposta há mais de 30 anos, e recebeu um nome bem legal: *O problema dos generais bizantinos*, ou também a *falha bizantina*.

A falha bizantina é a classe mais geral e mais difícil de detectar. Por exemplo, a falha-parada é fácil de detectar. Já a falha bizantina pode gerar dados arbitrários, fingindo ser um componente correto.

O problema dos generais bizantinos é um problema teórico na Ciência da Computação que vem sendo estudado há décadas, e que teve uma possível boa resposta na recém implementação do protocolo Bitcoin. Uma referência clássica para o problema é *The Byzantine Generals Problem* de LESLIE LAMPORT, ROBERT SHOSTAK, e MARSHALL PEASE. Ela pode ser recuperada na Internet.

Na CC trata-se de obter consenso em uma rede distribuída (sem hierarquia), com nodos funcionais e *bem comportados* embora possa haver nodos *mal comportados*. Este caso engloba tanto os nodos maliciosos quanto aqueles com problemas no funcionamento.

No caso dos generais bizantinos, imagina-se um cenário de guerra no qual os exércitos bizantinos cercam uma cidade inimiga. Cada exército dos invasores é comandado por um general. A garantia de vitória só existe se todos os exércitos (ou sua maioria) atuar coordenadamente. Se apenas um exército resolver invadir sem o auxílio dos demais, certamente será derrotado.

Para facilitar o raciocínio define-se um dos m generais como sendo o comandante. É ele que pode dar a ordem de ataque. Todos os demais $m-1$ exércitos são comandados por um tenente general. Todos os generais só podem se comunicar por mensageiros e cada mensageiro só pode levar uma de duas ordens: atacar ou retirar.

Mais ainda, os generais podem ser leais ou traidores e o objetivo dos traidores é evitar que os leais se ponham de acordo. Para fazer isso, os traidores oferecem informação errônea. Por exemplo, se o comandante é traidor, pode mandar ordens diferentes para os diferentes tenentes. Se um tenente é traidor, poderia tentar enganar a outros tenentes de modo a que eles achassem que o comandante seria traidor ou seja que o comandante mandou a ordem contrária à que realmente enviou.

Para resolver satisfatoriamente o problema precisamos achar algoritmos que tenham os seguintes objetivos:

- Todos os tenentes leais devem tomar a mesma decisão
- Se o comandante é leal, todos os tenentes leais executam a ordem que este lhes enviou.

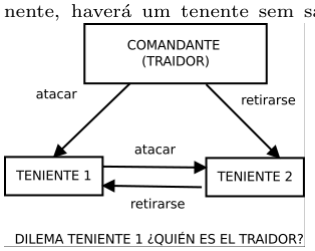
Normalmente para chegar a uma solução, se impõe algumas condições adicionais

- cada mensagem enviada chega corretamente (os mensageiros são ok)
- cada receptor de mensagem conhece quem a enviou.
- a ausência de mensagem pode ser detectada
- havendo ausência de mensagem, tem-se uma ação *default*. Isso ocorre para evitar o problema do comandante ser traidor e não mandar nenhuma ordem

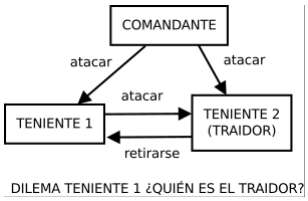
No artigo acima citado, os autores propõe alguns algoritmos de solução:

Mensagens orais e todos falam com todos A estratégia começa buscando saber se o comandante é traidor: Todos os tenentes se reenviam a infomação que o comandante lhes enviou. Se o tenente é leal, a informação que ele retransmitirá será a que lhe enviou o comandante. A consequencia de usar mensagens orais (não assinadas) é que um tenente traidor pode dizer o oposto do que o comandante lhe disse.

Caso de 3 generais Suponhamos que o comandante é traidor. Se ele manda uma mensagem diferente a cada tenente, haverá um tenente sem saber o que fazer

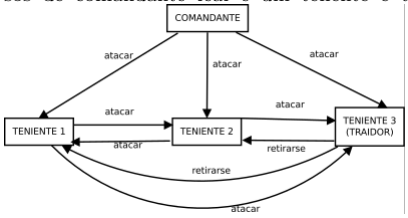


Supondo que um tenente é traidor, e ele manda informação diferente da que recebeu: O outro tenente não saberá o que fazer:

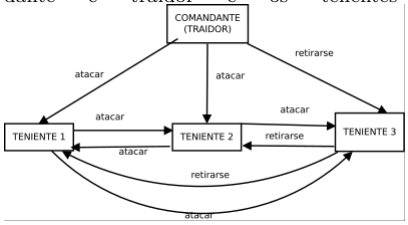


A conclusão é que não se pode resolver o problema se se permite que em 3 generais um seja traidor. Isto se deve ao fato de que não há generais suficientes para formar uma opinião de consenso.

4 generais Aqui é possível um consenso, através do seguinte algoritmo: Ao receber a ordem do comandante e as mensagens dos outros 2 tenentes os tenentes leais decidirão a ordem de consenso segundo uma função de maioria $M(v_1, v_2, v_3)$ devolvendo o mais frequente valor v_i, onde v_i é a ordem mandada pelos diferentes enviadores ao general que estamos avaliando. Vejamos os casos de comandante leal e um tenente é traidor:



e o esquema onde o comandante é traidor e os tenentes leais



m generais Para a generalização funcionar, tendo t traidores, se necessita que m seja maior ou igual a $3t+1$. O algoritmo se chama $OM(m)$ onde OM vem do inglês *Oral Messages* e usa a seguinte função de maioria: $M(v_1, v_2, ..., v_n)$ onde v_i é a ordem mandada ao general que estamos avaliando.

Mensagens assinadas e todos se comunicam com todos Neste cenário, as mensagens vão assinadas por serem mensagens escritas. Neste caso os traidores não os podem modificar e dizer que proveem do comandante. Nesta situação pode-se resolver o problema com 3 generais e um traidor. O algoritmo solução se chama SM (Signed Messages) e é o seguinte: Primeiro o comandante envia uma ordem assinada a todos os tenentes. Cada vez que um tenente recebe uma mensagem assinada ele a guarda, faz uma cópia, a assina, e reenvia a todos os tenentes que cuja assinatura não aparece na mensagem que chegou. Segundo este algoritmo os tenentes não receberão mais mensagens além de todas as combinações possíveis. Uma vez recebidas as mensagens, cada tenente toma a decisão baseando-se na ordem transmitida pela maioria. 0

Neste cenário, os comandantes traidores não descobertos imediatamente, já que assinam ordens contraditórias.

Nem todos podem se comunicar com todos Se falta alguma conexão, as coisas se complicam. Tudo depende agora de condições de conectividade: como sempre graus crescentes de redundância aumentam a confiabilidade. Na literatura citada, podem-se encontrar condições de conectividade necessárias e suficientes para cada caso.

Algumas observações:

- O problema dos generais bizantinos é o mesmo de se enviar dinheiro virtual sem um intermediário confiável. Bitcoin ofereceu a primeira solução prática aqui.
- No mundo real as conexões falham de maneira não deliberada. Para as detectar pode-se usar *códigos de detecção de erros*. No cenário de mensagens orais a conexão que falta pode ser considerada como um nodo traidor. Se se usam mensagens assinadas, uma falha de conexão se detectaria de maneira irrefutável.
- Para conhecer o enviante de mensagens orais só é possível se usarmos linhas fixas e não redes de comunicações. Com mensagens assinadas não há problema em reconhecer o emissor.
- A ausência de mensagem pode ser detectada usando *time-out*.
- No mundo real não se pode garantir que um erro aleatório não possa falsificar uma assinatura. No entanto, isso pode ter uma probabilidade tão baixa quanto se queira, usando-se métodos adequados.
- Evitar fraudes deliberadas se converte em um problema criptográfico. Portanto é fundamental escolher algoritmos de assinatura seguros.
- Deve-se detectar se uma mensagem é enviada duas (ou mais) vezes pela comprovação da assinatura. Assim, uma assinatura não deve poder ser gerada se o processo já a viu em instante anterior.

Para encerrar, a literatura apresenta dezenas de algoritmos que buscam sanar o problema da falha bizantina. Abundante literatura da NASA (por exemplo) mostra casos reais da falha acontecendo. Os aviões Boeing (777 e 787) usam o algoritmo com tempos de latência da ordem do microssegundo (por razões óbvias). Cada um desses algoritmos caracteriza o problema e busca a solução de acordo com os parâmetros dados pela tecnologia, plataforma, arquitetura e problema específicos.

☞ Para você fazer

Desenhe o grafo de uma situação em que há 5 generais e um deles é traidor. Estude o caso em que o comandante é traidor e o caso em que o traidor é um tenente. Conclua adequadamente o que acontece em cada um dos casos.