Problema de los Generales Bizantinos

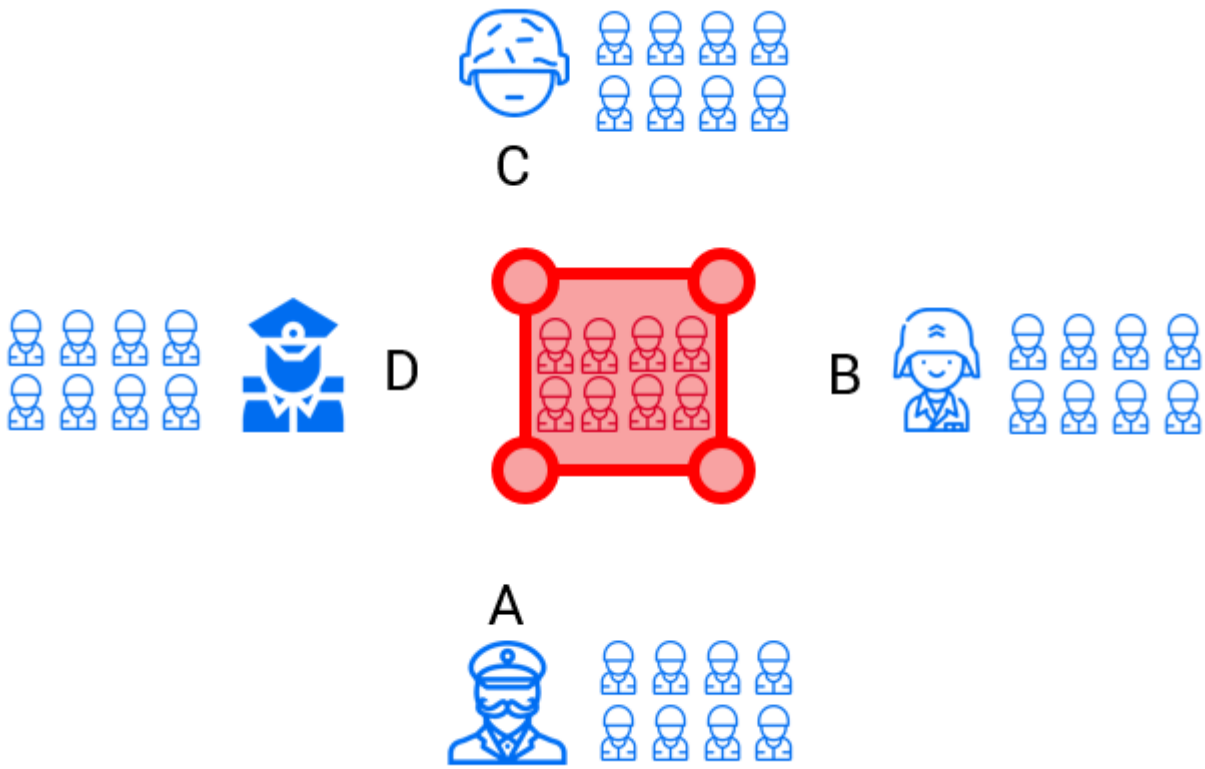
Benjamín Benavides - bbenavides1@uc.cl Sistemas distribuidos 2018-2

TLDR: Cómo un individuo puede estar completamente seguro que multiples entidades, separadas por distancia y posiblemente maliciosas, han llegado a un consenso antes de realizar una acción. Buen video explicativo (4 mins)

En qué consiste

Analogía del asedio

4 batallones del ejercito bizantino se encuentran cerca de un castillo enemigo y deben decidir si atacarlo o retirarse. Lo importante es que una vez tomada tomada la decisión, todos deben hacer lo mismo: Si deciden atacar, **todos** deben hacerlo o serán derrotados.



Supongamos que nosotros somos el general A. Lo que queremos es una forma de estar totalmente seguros que estamos de acuerdo en la decisión con los demás generales de modo de no ser derrotados.

- Podríamos hacer señales con fuego, pero podrían ser vistas por el enemigo. Es necesario que los mensajes sean dirigidos.
- Podríamos mandar mensajes a caballo, pero estos podrían perderse o ser asesinados por el enemigo en el camino. Es necesario poder verificar que un mensaje fue recibido.
- El mensaje podría ser interceptado y cambiado por el enemigo. Es necesario poder validar un mensaje (ej: una firma).

- Un general podría ser un espía y mandar decisiones distintas a cada general. Es necesario poder asegurar consistencia.

En computación

Ahora pensemos que cada general es un computador en una red. Algunos ejemplos prácticos que requieren consenso son: sincronización de relojes, page rank (algoritmo que usa google para determinar la relevancia de un sitio), sistemas de distribución eléctrica, sistemas de control de vuelo aéreo, etc.

Un nodo "traidor" no es necesariamente un *hacker* (aunque podría serlo). Puede ser simplemente un computador que funciona correctamente, pero que cuando se queda sin memoria ram, envía *null* en lugar de lo que debería.

Las fallas en un sistema se pueden clasficar en 2 grandes grupos:

- **falla *fail-stop***: Cuando un nodo deja de funcionar, pero otros nodos en la red pueden detectarlo y estar de acuerdo sobre su estado. Una vez detectado el problema se puede resetear el nodo o activar otro nodo redundante que reemplace al nodo con problemas para que el sistema siga funcionando.
- **falla bizantina**: Aquella que presenta sintomas distintos a observadores distintos.

En términos prácticos, un programa/sistema *Byzantine Fault Tolerant* (BFT) es aquel que puede seguir funcionando aunque algunas de sus partes tengan fallas bizantinas.

Soluciones

- PBFT (Practical Bizantine Fault Tolerant): funciona cuando menos de 1/3 de los nodos tienen fallas bizantinas. Utilizado para crear una version de NFS que es BFT siendo tan solo un 3% más lenta.
- También existen soluciones que permiten cualquier número de nodos con fallas bizantinas, pero requieren que las firmas de los mensajes no sean falsificables. Esto se puede lograr usando criptografía (usando llaves públicas y privadas)

Fuentes

- Documental "Hidden Secrets of Money: From Bitcoin to hashgraph"
- Wikipedia, Byzantine_fault_tolerance
- Wikipedia, Tolerancia a faltas Bizantinas
- Wikipedia, Consensus
- CS198.2x Blockchain Technology Week 1: Practical Byzantine Fault Tolerance
- Alvaro Videla: Failure Modes in Distributed Systems