UCB CS 162 - Operating Systems and Systems Programming "Lecture 23 - Distributed Decision Making, Networking, TCP/IP"

Distributed Decision Making, Networking, TCP/IP

General's Paradox

The general's paradox is a problem in which two generals have to decide on a particular time to coordinate their attack. They can only exchange physical messages through messengers, who can be captured by the enemy. If they successfully decide on a time to attack, they will succeed. If they cannot decide on a time and attack at different times, they will fail.

The general's paradox doesn't have a solution. Since the messengers can be captured, you can never be sure that the last message you sent was received. So even if you send an ACK, you can never be sure that the other person got the ACK.

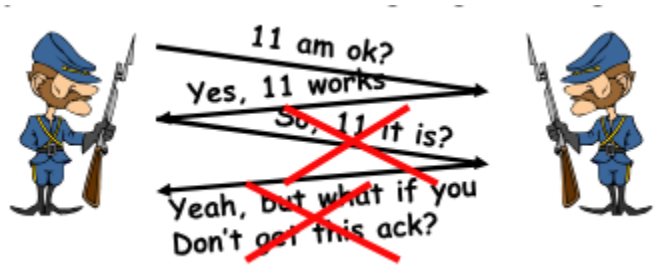

Figure 1: General's Paradox

Two-Phase Commit

The two-phase commit protocol was developed by Jim Gray, a Turing award winner and Berkeley alumni. The two-phase commit protocol solves a problem related to the General's paradox, since we cannot solve the general's paradox itself.

The 2PC algorithm has one coordinator node and n workers who are trying to decide whether to commit or abort a transaction. The transaction can be a database transaction, a distributed file system transaction, or any other action that can be committed or aborted.

The high-level overview of the algorithm is -

1. The coordinator asks all workers if they want to commit or abort
2. All workers decide individually if they are ready to commit or if they want to abort
3. All workers reply with their desired action
4. If the coordinator receives "commit" from all workers, it sends a "global-commit" message to each worker, which causes each worker to commit their transaction
5. If the coordinator receives at least one "abort", it sends a "global-abort" message to each worker, which causes each worker to abort their transaction

All workers and the coordinator have an internal log in which they record their decision so that they will remember their decision even after a crash.

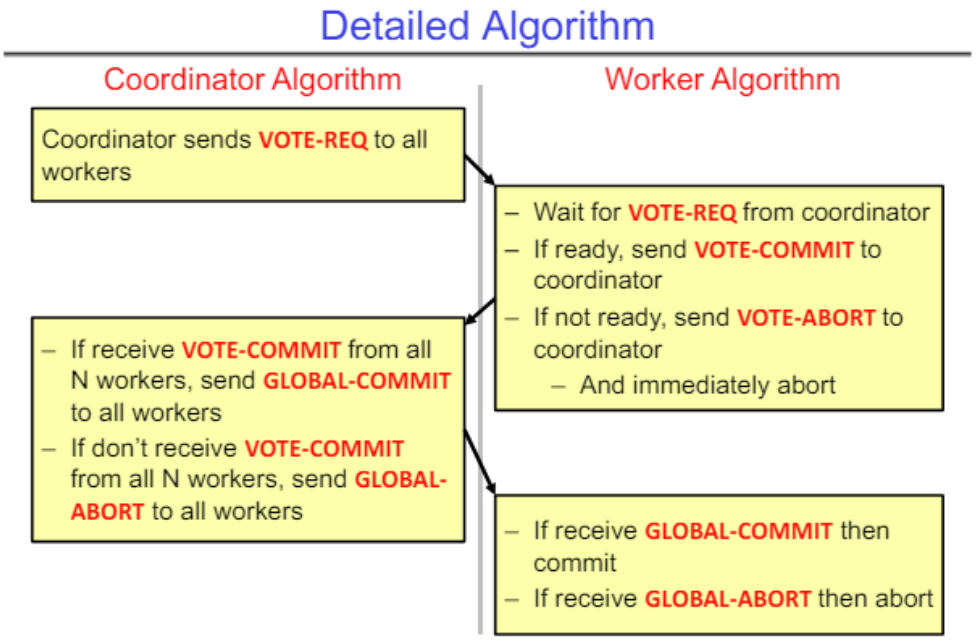

Figure 2: 2PC algorithm

A key drawback of the 2PC algorithm is that it is blocking. If a node fails or keeps failing, the other nodes are blocked and cannot make progress. A blocked node can also hold resources like locks, which make the blocking problem worse. There are

some alternatives to the 2PC algorithm, such as the 3 phase commit, which has 3 phases instead of 2, PAXOS, a complicated distributed decision making algorithm used by Google in its internal systems, etc.

Another limitation of the 2PC algorithm (and also the alternatives to it) is that it assumes all nodes are working correctly, and no node is malicious. If a distributed system can have malicious nodes, you need a different class of algorithms. This situation is demonstrated by the Byzantine general's problem.

Byzantine General's Problem

The Byzantine general's problem has n players (nodes), out of which one or more can be malicious. One of the nodes is the General (leader), and has to send a boolean message to all nodes. The outcome must be that all correctly working nodes perform the same action, regardless of how the malicious nodes try to throw them off.

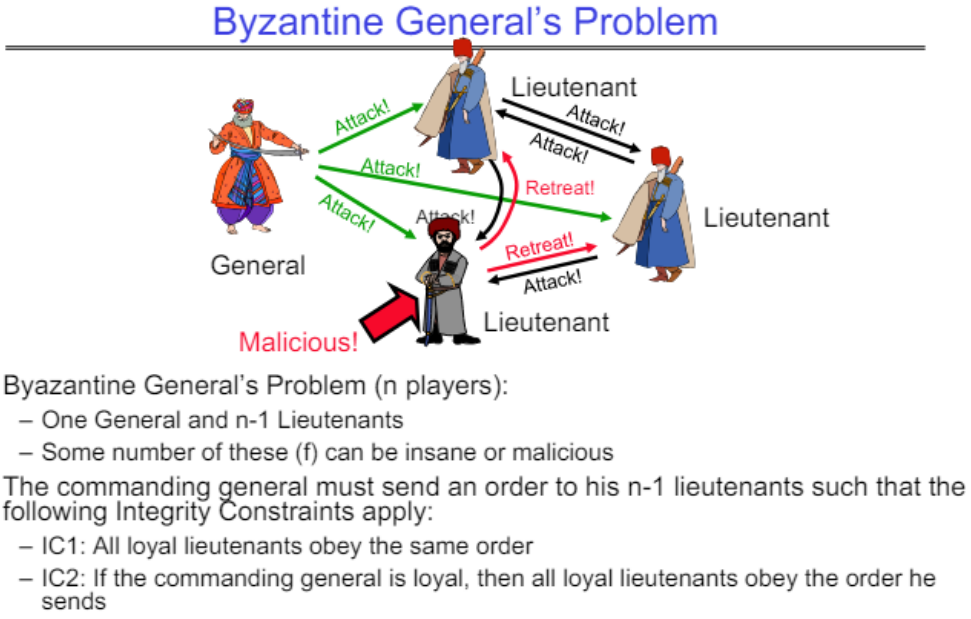

Figure 3: Byzantine General's Problem

The leader itself could be compromised. In this case, all the correctly functioning nodes still perform the same action, but that action may not be what the leader communicates to each node.

The lecture doesn't actually describe the algorithms that solve this problem, but today's algorithms are $ O(n^2) $, and block chain based algorithms can even be $ O(n) $ in the number of messages that need to be exchanged between the nodes.