Lecture 8 – Byzantine fault tolerance – Tien Dat Le

Byzantine fault tolerance is a class of distributed systems that is dependable against Byzantine failure. In this kind of failure context, components may fails and there is imperfect information on whether a component is failed. In particular, a computer component may behave inconsistently as both working and failed to failure-detection systems.
Byzantine General's problem is referred to as a general problem in distributed systems to solve the problem of reaching consensus with the appearance of Byzantine failure.
Practical Byzantine Fault Tolerance is a new replication algorithm that is able to tolerate Byzantine faults. The algorithm offers both safety and liveness provided that at most [(n-1)/3] out of n replicas is faulty. The algorithm works in asynchronous distributed systems like in the Internet. The algorithm ensures safety property by guaranteeing that all non-faulty nodes agree on a total order of execution of request despite failures.
The algorithm is defined in three phases: pre-prepare, prepare and commit. The pre-prepare and prepare phases are used to totally ordered requests sent in the same view event when the primary, which is the one proposed the ordering of requests is faulty. Commit is the phase to commit the state change. As a node may fail and need to restore the state later from all the faulty nodes, the nodes have to maintain a buffer of all the messages within a boundary and a provable checkpoints. Buffer will be reset when a new checkpoint is created. When the primary replicas is faulty, a view change will be triggered to change to another primary backup.
Both PBFT and PoW is designed to solve the consensus problem in asynchronous distributed system. There are trade-offs between the two systems. While PBFT seems to not scale compare to PoW, it offers better performance regards of performance and power consumption, the consensus in PBFT will reach eventually and there is a proof of correctness. However, PoW prove to scale to work with thousands of nodes, make the systems truly decentralized.