Byzantien Fault Tollerance

Original paper by Leslie Lamport, Robert Shostak and Marshall Pease, 1982: https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf

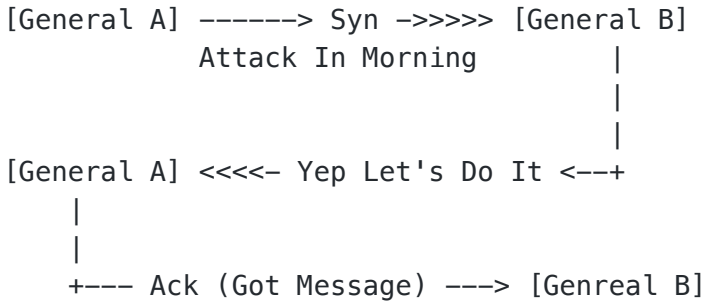
Practical Byzantine Fault Tolerance and Proactive Recovery" by Miguel Castro and Barbara Liskov: http://www.pmg.csail.mit.edu/papers/bft-tocs.pdf

Honney Badger BFT, 2016: https://eprint.iacr.org/2016/199.pdf Go Implementation: https://github.com/anthdm/hbbft

Two Generals Problem

1. 2 generals with armies.
2. Opponent is larger than each of the 2 armies but smaller than the combined 2 armies.
3. Incentive to cheat.

Come up with a solution (Like TCP/IP Syn / SynAck / Ack)



With the network as the "opponent" there is no way to solve this. *General B* never knows until a successful `Ack` what state he should be in.

There is no way to get consensus.

Let's reverse this and let's assume that the nodes are the corrupting element.

Lamport's '82 paper has a solution. Very Ugly - Very Slow. Speed is `O(n^(f+1))`. Puts proven bounds on number of bad nodes and what can be done to solve this.

Is this valuable:

1. Boeing 767, 777 both use a solution for fly by wire.

2. Blockchains/Cryptocurrency is using this.
3. Data distribution over a faulty network.
4. IBM Hyperledger (Merk, IBM Food Trust etc).
5. Zilliqa uses an optimized version pBFT with a PoW round every ~100 blocks.

Multi Generals and Consensus

After Consensus:

```
General A         - A D D A A
General B         - A D D A A
General C         - D A A A A
General D         - A D D A A
General E         - A A D A A
                    ^ <<<<<<<<<< This one!
```

Communicate 1 decision from 1 general to all the all the other generals. Solves 1 communication.

How many traitors can we deal with?

Trivial Case - 1 general.

2 - general. If 1 traitor, then no conses.

Leader general says Attack, then how if you as a non-leader know. You could ask your piers. Walk through cases. Ask Piers

1. Case with peer as a traitor.
2. Case with leader as a traitor.

No Solution w/ 3 generals and 1 traitor case.

How many generals to tolerate 1 traitor. Generals = 3, 1 traitor. 3m+1. – Or look at it as you have to have less than $1/3$ traitors.

Proof: Assume that a solution exists. 12 Generals 4 Traitors. Simulate each one. Determine that you don't have a distinct answer. The lack of a unique solution means that you can't have this many. Simulate with 12 and 5 ... same result. Simulate with all cases with 12 and 3 - you can get a unique solution.

Algorithm that works:

1. 0 Traitors - then everybody shares - all agree - all good

2. M Traitors - less than $1/3$ - sends order. Record order. Add up all the total number of orders - and when you have a set of answers. Use majority.

Demo of this with a traitor.

2 traitors means you will need 7 generals.

m = number of traitors.

Traitors	Order
m = 0	0
m = 1	n^2
m = 2	n^3
m = 3	n^4

messages.

Practical Byzantine Fault Tolerance

For pBFT to work, we assume that the amount of malicious (lying) nodes in the network cannot simultaneously equal or exceed $1/3$ of the total number of nodes during any one time of vulnerability. The larger the number of nodes in the system the more robust the system. With `n` nodes as long as at most (n-($1/3$)) are malicious or faulty at the same time the pBFT algorithm provides both speed and safety. Eventual consensus of the replies is achieved due to linearizability.

Each round of pBFT consensus (called views) is in 4 distinct phases. This model follows more of a "Commander and Lieutenant" system than a pure Byzantine Generals' Problem, where all generals are equal, due to the presence of a leader node. The phases are below.

1. A client sends a request to the leader node to invoke a service operation.
2. The leader node uses "hear-say" to multicasts the request to the other nodes.
3. The nodes execute the request and then send a reply to the client.
4. The client awaits f + 1 (f represents the maximum number of nodes that may be faulty) replies from different nodes with the same result. This result is the result of the operation.

The requirements for the nodes are that they are deterministic and start in the same state. The final result is that all honest nodes come to an agreement on the order of the record and they either accept it or reject it.

The leader node can be changed in a round robin type format during every view and can even be replaced with a protocol called view change if a specific amount of time has passed without the leader node multicasting the request. A majority of honest nodes can also decide whether a leader is faulty and remove them with the next leader in line as the replacement.

The primary advantages of the pBFT model is its ability to provide transaction finality without the need for confirmations like in Proof-of-Work models. If a proposed block is agreed upon by the nodes in a pBFT system, then that block is final. This addresses a lot of the energy usage concerns of Eth/BTC.

The model only works well with small consensus group sizes and has a large amount of communication overhead. The paper mentions using digital signatures and MACs (Method Authentication Codes) for authenticating messages. This is extremely inefficient and will not scale. Also you can’t prove the authenticity of the messages. Digital signatures are a better solution.