DTU Compute
Institut for Matematik og Computer Science

Project Poster
02228 - Fault-Tolerant Systems

Raft as a solution to the consensus problem

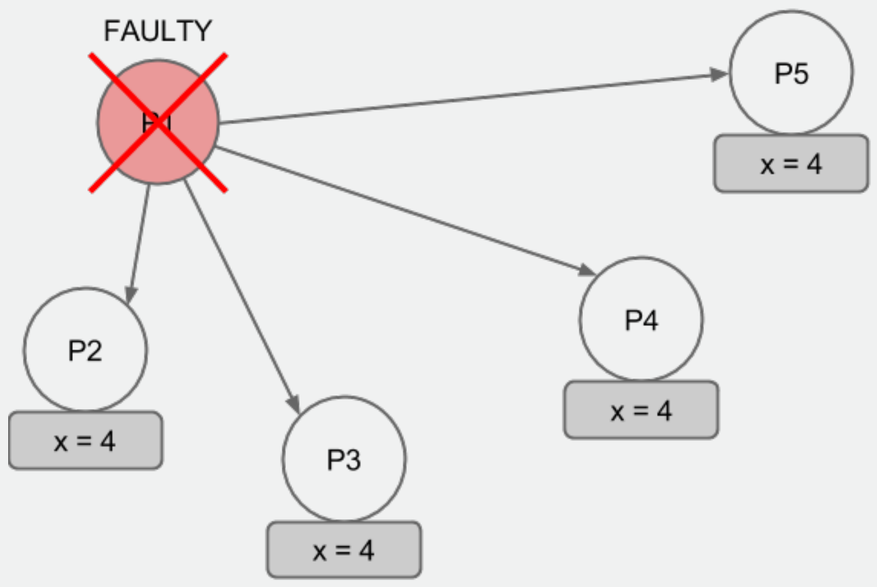
Consensus:
To decide on a common value amongst several processes in a distributed system.

Current solutions:
Suffer from complex specifications and variations resulting in buggy implementations.

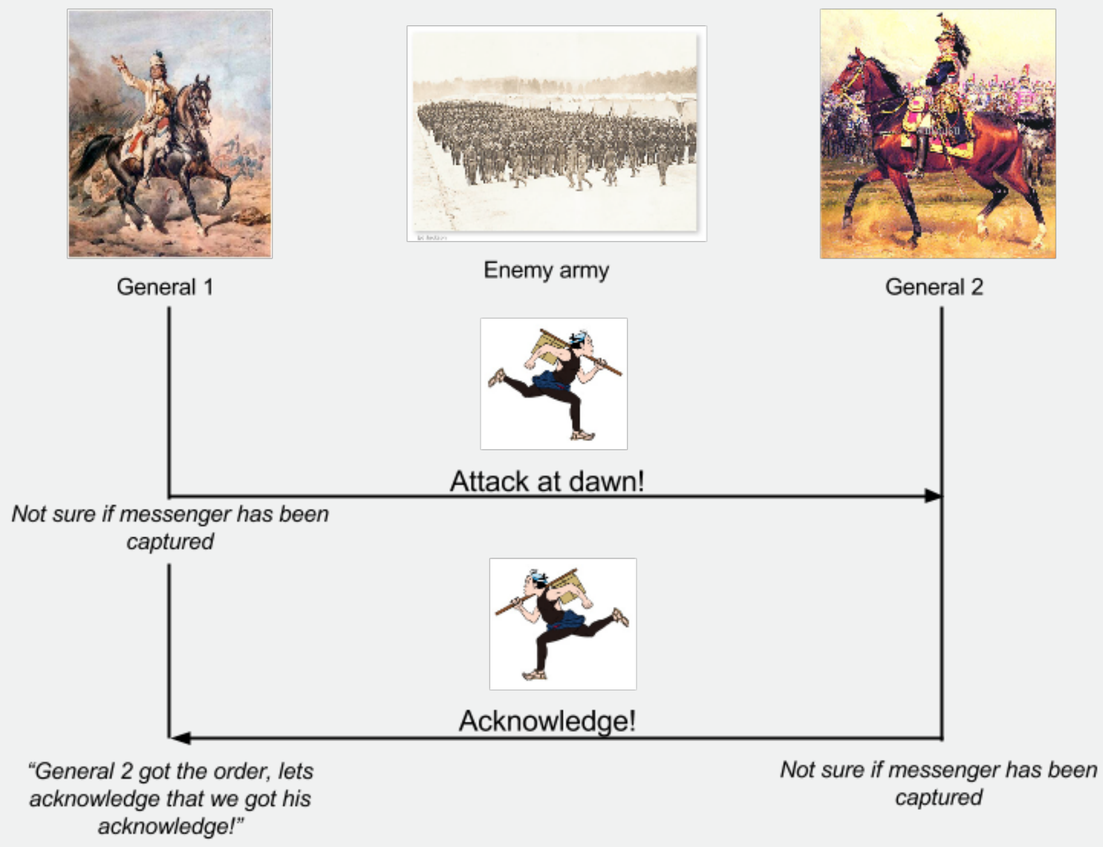
Two Generals Problem

Relation to consensus:
Two generals want to agree on when to attack the enemy.

Impossibility:
The generals will never reach an agreement since they are not sure about the messenger's reliability.



An example of a distributed system containing a faulty leader, which would cause the system to become unavailable.

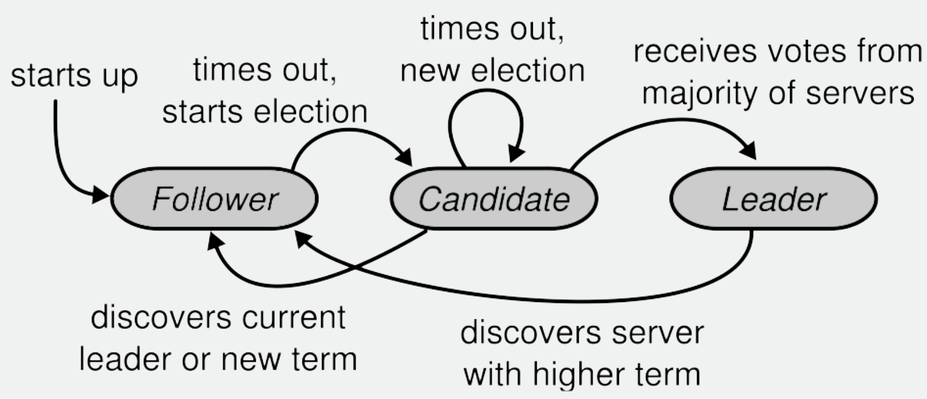


The Two Generals problem illustrated.

A fault tolerant and self-repairing system

When you might have found a suitable solution to the consensus problem and implemented it, but you also want it to be fault tolerant.

What happens when a process fails?
The system should of course still be operational under presence of failure. Solving this requires a fault tolerant consensus algorithm. A new solution to this is **Raft**!



A state machine showing the state a process can be in.
From the Raft paper - "In Search of an Understandable Consensus Algorithm".

Components of Raft

Log replication:
- Commands issued by a user through a client is replicated throughout the system.

Leader election:
- A leader is elected to be the link between the client and the rest of the network.
- The leader frequently sends out heart-beat (empty) messages in order to keep track of faulty processes.

Safety:
- There can only be at most one leader.
- If a leader crashes a new leader is elected.
- If a faulty process wakes up the leader ensures that their log is updated.

Test and behavioural driven development

- The algorithm is mainly described by its behaviour without any formal specification or pseudo code.
- From this tests were derived during the implementation. Each test was served as an implementation step.

Results

- Raft is showcased in an visualised implementation, describing its fault tolerance features in a nice and concise manner.
- The user is able to provide the program with various parameters for testing different heart beat timers, election timeouts etc.

```
------------------ Running Raft ------------------
Election timeout: between 4000 and 8000 ms
Heartbeat: every 1000 ms
RPC Delay: between 100 and 200 ms
--------------------------------------------------
Server 1  (follower)  term: 1  logEntries: 3 commitIndex: 3 4911
[v->X->5, t->1], [v->Y->2, t->1], [v->Z->3, t->1]

Server 2  (follower)  term: 1  logEntries: 2 commitIndex: 2 -15101
[v->X->5, t->1], [v->Y->2, t->1]

Server 3  (follower)  term: 1  logEntries: 3 commitIndex: 3 3691
[v->X->5, t->1], [v->Y->2, t->1], [v->Z->3, t->1]

Server 4  (follower)  term: 1  logEntries: 3 commitIndex: 3 6979
[v->X->5, t->1], [v->Y->2, t->1], [v->Z->3, t->1]

Server 5  (leader)    term: 1  logEntries: 3 commitIndex: 3 6681
[v->X->5, t->1], [v->Y->2, t->1], [v->Z->3, t->1]
```

Contact: Anders Nielsen, s103457@student.dtu.dk
Joachim K. Friis, s093256@student.dtu.dk
GitHub: https://github.com/anderslime/ft-raft