Distributed Systems

Consensus

Johan Montelius

Consensus

- Nodes in a system sometimes need to make a decision as a group.
  - **Agreement**: no two nodes should decide differently
  - **Termination**: all (correct) nodes should eventually decide
  - **Integrity:** a node is not allowed to change decision
  - **Validity**: a decided value must be a value proposed by someone

Why consensus

- Nodes need to take coordinated decisions
  - bank transactions
  - air traffic control
  - mobile handover
- often called something else
  - atomic broadcast
  - leader election
  - mutual exclusion

How do we reach consensus

How do we reach consensus

- Synchronous systems
  - not a problem
- Asynchronous systems
  - takes time but not a problem (if that is not a problem)
- Asynchronous systems with crashing nodes
  - not that easy

Fisher, Lynch and Patterson

- There is no algorithm that will guarantee to always reach consensus in an asynchronous system even if only one node can crash.
- But ... we are working in asynchronous systems and nodes can crash!
  - if it's impossible then let's ignore it.

Is there a way around this

- crashing nodes
  - fail stop
- idea
  - detect crashing nodes
- If we know which nodes that have crashed then we can reach a consensus among the non-crashed.

Failure detectors

- Assume we have a asynchronous system where each node has an oracle that can determine if nodes have crashed.
- The oracles are called failure detectors.
- Oracles are very expensive and you can probably not buy them but more on this later....
- What oracles do we have?

Completeness

- Completeness (strong)
  - Every crashed node is eventually <u>*suspected*</u> by all correct nodes

Accuracy

- strong
  - no correct node is ever *suspected* by any node
- weak
  - there exists a correct node that is never...
- eventually strong/weak:
  - there is a time after which...

Failure detectors

- perfect (P)
  - complete and strong accuracy
- strong (S)
  - complete and weak accuracy
- eventual perfect ($\Diamond$P)
  - complete and eventual strong accuracy
- eventual strong ($\Diamond$S)
  - complete and eventual weak accuracy

Failure detectors and consensus

- Given a eventual strong, (◇S), failure detector (uniform) consensus can be solved in a asynchronous network of n nodes even if (n-1)/2 nodes fail by crashing.
- Can we implement a eventual strong failure detector in a asynchronous network?

How hard can it be?

- Every now and then send a message to the node, if no reply within 10 seconds, it's dead.
- How to choose:
  - now and then
  - 10 seconds
- Failure detectors
  - suspect nodes to have failed

... but we could

- .. implement one that might work in practice
  - hopefully there will be a time after which there exists a correct node that is not suspected by any correct node <u>for sufficiently long time for the consensus to be formed</u>

```
In round r from 0...
    leader is (r mod n)+1
    phase 1
        send estimate and when you
        adopted this to leader
    phase 2
        leader collects (n+1)/2 estimates
        estimate is set to latest estimate
        received, send new estimate to all
    phase 3
        adopt new estimate and send ack, or
        if leader suspected to have crashed
        send nack
    phase 4
        leader waits for (n+1)/2 messages,
        if all ack then reliably broadcast
        decided
```

This is too much....

- That's too complicated
  - we don't have time for this
  - too many messages
  - no guarantee that it will ever terminate
- Not that bad if there are no failures!
  - leader sends estimate to all
  - all reply with **ack**
  - leader *reliably broadcast* **decide**
  - **... or even simpler**

Why have we survived so far

- non-distributed systems
- client/server systems where consensus is not an issue
- people are used to inconsistent systems
- small systems where failures do not happen
- vital systems do use these techniques

What will change

- If you have several hundred nodes connected into one service, crashes will be part of the weekly procedure.
- More and more systems are vital need to be fault tolerant.

Summary

- Consensus
  - in general unsolvable if we have a dead-line
  - in practice solvable using non-perfect failure detectors
- Failure detectors
  - not perfect
  - we need to handle this
  - we can