Overview
Agreement (= Consensus) problems
- why & where needed
- definition
Byzantine generals
- in synchronous systems
- in asynchronous systems
And finally
- impossibility results!
- practical implications

Consensus algorithms
used when it is necessary to agree on actions:
- in transaction processing
  - commit or abort transaction?
- mutual exclusion
  - which process should enter the critical section?
- in control systems
  - proceed or abort based on sensor readings?

The model & assumptions
The model
- N processes
- message passing
- synchronous or asynchronous
- communication reliable
Failures!
- process crashes
- arbitrary (Byzantine) failures
  - processes can be treacherous and lie
The algorithm
- works in presence of certain failures

Consensus: main idea
Initially
- processes begin in undecided state
- propose an initial value from a set D
Then
- processes communicate, exchanging values
- attempt to decide
- cannot change the decision value in decided state
The difficulty
- must reach decision even if crash has occurred
- or arbitrary failure!

Consensus for three processes

Consensus: requirements

- **Termination**: Eventually each correct process sets its decision value.
- **Agreement**: Any two correct processes must have decided on the same decision value.
- **Integrity**: If all correct processes propose the same value, then any correct process that has decided must have chosen that value.

Towards a solution

For simplicity, assume no failures:
- Processes multicast its proposed value to others.
- Wait until all N values collected (including own).
- Decide through majority vote (\(\perp\) special value if none)
  - Can also use minimum/maximum.

It works since:
- All processes end up with the same set of values.
- Majority vote ensures Agreement and Integrity.

But what about failures?
- Process crash - stops sending values after a while.
- Arbitrary failure - different values to different processes.

Byzantine generals

The problem [Lamport 1982]
- Three or more generals are to agree to attack or retreat.
- One (commander) issues the order.
- The others (lieutenants) decide.
- One or more generals are treacherous (= faulty!)
  - Propose attacking to one general, and retreating to another.
  - Either commander or lieutenants can be treacherous.

Requirements:
- Termination, Agreement as before.
- Integrity: If the commander is correct then all correct processes decide on the value proposed by commander.

Consensus in synchronous system

Uses basic multicast:
- Guaranteed delivery by correct processes assuming the sender does not crash.
- Admits process crash failures.
- Assumes up to f of the N processes may crash.

How it works...
- f+1 rounds.
- Relies on synchrony (timeout!).

Consensus in synchronous system

Initially:
- Each process proposes a value from a set D.

Each process:
- Maintains the set of values \(V_r\) known to it at round \(r\).

In each round \(r\), where \(1 \leq r \leq f+1\), each process:
- Multicasts the values to each other (only values not sent before, \(V_r - V_{r-1}\)).
- Receives multicast messages, recording any new value in \(V_r\).

In round \(f+1\):
- Each process chooses minimum \(V_{f+1}\) as decision value.

Why it works?
- Sets timeout to maximum time for correct process to multicast message.
- Can conclude process crashed if no reply.
- If process crashes, some value not forwarded...

At round \(f+1\):
- All correct process arrive at the same set of values.
- Hence reach the same decision value (minimum).
- At least \(f+1\) rounds needed to tolerate \(f\) crash failures.

What about arbitrary failures?
**Byzantine generals...**

Processes exhibit arbitrary failures
- up to f of the N processes faulty

In a synchronous system
- can use timeout to detect absence of a message
- cannot conclude process crashed if no reply
- impossibility with $N \leq 3f$

In asynchronous system
- cannot use timeout to reliably detect absence of a message
- impossibility with even one failure!!

**Impossibility with three generals**

Assume synchronous system
- 3 processes, one faulty
- if no message received, assume ⊥.
- proceed in rounds
- messages ‘3:1:u’ meaning ‘3 says 1 says u’

Problem! ‘1 says v’ and ‘3 says 1 says u’
- cannot tell which process is telling the truth!
- goes away if digital signatures used...

Show
- no solution to agreement for $N=3$ and $f=1$
- Can generalise to impossibility for $N \leq 3f$

**Three Byzantine generals**

<table>
<thead>
<tr>
<th>Process</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$ (Commander)</td>
<td>Sends illegal value to $p_2$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>Cannot tell which value sent by commander</td>
</tr>
<tr>
<td>$p_3$</td>
<td>Sends 2:1:u to $p_2$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>Sends 1:v to $p_1$ and 4:1:v to $p_2$</td>
</tr>
</tbody>
</table>

Faulty processes are shown in Colour

$p_3$ sends illegal value to $p_2$
$p_2$ cannot tell which value sent by commander

**Impossibility with three generals**

So, if the solution exists
- $p_2$ decides on value sent by commander (v) when the commander is correct
- and also when commander faulty (w), since cannot distinguish between the two scenarios

Apply the same reasoning to $p_1$
- $p_2$ must decide on x when commander faulty

Thus
- contradiction! since $p_2$ decides on w, $p_1$ on x if commander faulty
- no solution exists

**But...**

Solution exists for 4 processes with one faulty
- commander sends value to each of the lieutenants
- each lieutenant sends value it received to its peers
- if commander faulty, then correct lieutenants have gathered all values sent by the commander
- if one lieutenant faulty, the each correct lieutenant receives 2 copies of the value from the commander

Thus
- correct lieutenants can decide on majority of the values received
- Can generalise to $N \geq 3f + 1$

**Four Byzantine generals**

<table>
<thead>
<tr>
<th>Process</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$ (Commander)</td>
<td>Sends 2:3:w to $p_2$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>Decides majority(w,x,y) = w</td>
</tr>
<tr>
<td>$p_3$</td>
<td>Decides majority(y,x,w) = w</td>
</tr>
<tr>
<td>$p_4$</td>
<td>Decides majority(y,x,w) = w</td>
</tr>
</tbody>
</table>

Faulty processes are shown red

$p_3$, $p_4$, and $p_1$ decide, (no majority exists)
In asynchronous systems...

No guaranteed solution exists even for one failure!!!
[Fisher, Lynch, Paterson '85]
- does not mean never reach consensus in presence of failures
- but that can reach it with positive probability

But...
- Internet asynchronous, exhibits arbitrary failures and uses consensus?

Solutions exist using
- partially synchronous systems
- randomisation [Aspnes&Herlihy, Lynch, etc]

Summary

Consensus algorithms
- fundamental to achieve co-ordination
- crash or arbitrary (=Byzantine) failures
- several impossibility results

Solutions for synchronous systems
- if at most $f$ crash failures, in $f+1$ rounds
- if $N$ processes, $f$ faulty, $N \geq 3f + 1$

Solutions for asynchronous systems
- no guaranteed solution even for one failure!
- randomised solutions possible

Exercise