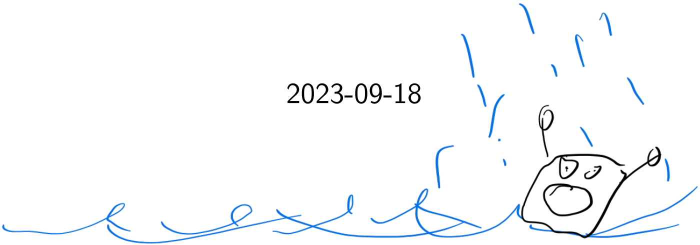
CPSC 465/565 Theory of Distributed Systems

James Aspnes

2023-09-18

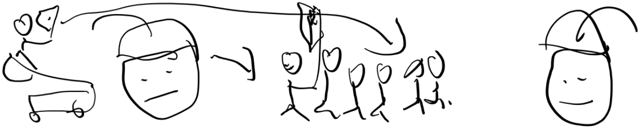


Today's exciting topics

Synchronous agreement with failures, including:

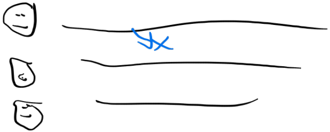
- Coordinated attack!
- Upper and lower bounds on time!
- Byzantine agreement!

Two generals problem (Gray 1978, Akkoyunlu et al. 1975)

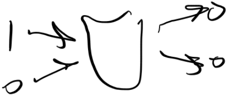


- Two generals separated by a dangerous enemy camp.
- Attack or retreat?
  - Both attack: Victory!
  - Both retreat: Maybe victory tomorrow.
  - One attacks, one retreats: Shameful defeat!
- Model: message-passing with omission failures
- Question: Can we reach agreement always?

Formal version



- **Coordinated attack problem** (*n*-general version)
- Synchronous message passing with omission failures.
  - Any message can be lost.
  - Sender doesn't know about lost messages.
- Requirements:
  - **Agreement**: All processes output same 0-1 value.
  - **Termination**: Protocol finishes in bounded rounds.
  - **Validity**:
    - If all processes have the same input
    - and no messages are lost,
    - all processes output the common input.



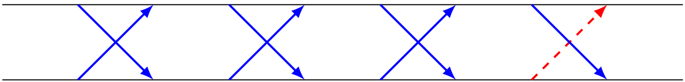
Impossibility of coordinated attack

$\Xi \underset{p}{\sim} \Xi'$ iff $\Xi|p = \Xi'|p$

Claim: No protocol survives unlimited message losses.

0 q

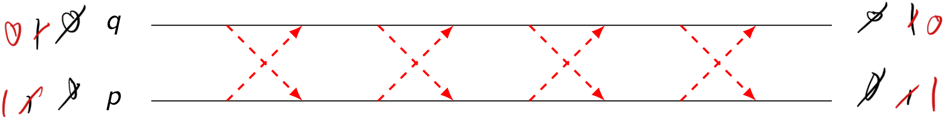
0 p



Proof:

- Fix a supposedly correct protocol with processes p and q.
- Consider execution Ξ_0 with all 0 inputs and no lost messages.
  - Validity $\Rightarrow$ both processes decide 0.
  - Termination $\Rightarrow$ finite sequence of messages.
- What happens if we delete last message from p to q?
  - New execution $\Xi_0^{-1} \sim_p \Xi_0$.
  - So p still decides 0.
  - Agreement $\Rightarrow$ q still decides 0.
- Now delete last message from q to p to get Ξ_0^{-2}.
  - Now $\Xi_0^{-2} \sim_q \Xi_0^{-1}$.
  - So both p and q decide 0 in Ξ_0^{-2}.

Impossibility of coordinated attack



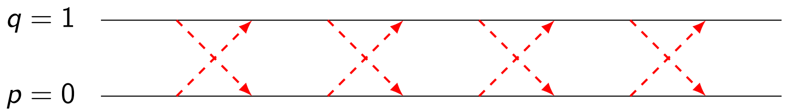
Remove messages one at a time until all messages are lost.

- Get $\Xi_0 \sim_p \Xi_0^{-1} \sim_q \Xi_0^{-2} \sim_p \Xi_0^{-3} \sim_q \ldots \Xi_0^{-m}$, where
  - Each Ξ_0^{-k} has both processes output 0, and
  - Ξ_0^{-m} delivers no messages.

Now do the same thing starting with Ξ_1 with 1 inputs:

- Get $\Xi_1 \sim_p \Xi_1^{-1} \sim_q \Xi_1^{-2} \sim_p \Xi_1^{-3} \sim_q \ldots \Xi_1^{-m'}$, where
  - Each Ξ_1^{-k} has both processes output 0, and
  - $\Xi_1^{-m'}$ delivers no messages.

Impossibility of coordinated attack



$1 \neq 0$

Finally, construct Ξ_{01} where

Ξ_{00}^{-m} $\Xi_{01}^{-m'}$

- p has input 0
- q has input 1
- All messages are lost, so $\Xi_0^{-m} \sim_p \Xi_{01} \sim_q \Xi_1^{-m'}$.

Since p decides 0 in Ξ_0^{-m}, p decides 0 in Ξ_{01}.

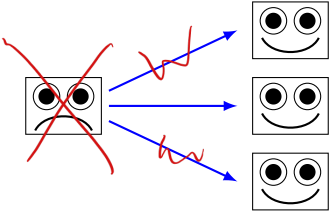
Since q decides 1 in $\Xi_1^{-m'}$, q decides 1 in Ξ_{01}.

So Ξ_{01} violates agreement.

(For $n > 2$, same argument, just lose more messages.)

Synchronous model with crash failures

Crash failure: Faulty process dies!



- Some subset of messages in crash round are delivered.
- No messages in subsequent rounds are delivered.
- $f \leq t$ processes fail in an execution.
  - $f =$ actual failures
  - $t =$ maximum failures ("tolerance")

This is a stronger model than omission failures.

Synchronous agreement with crash failures

Almost the same requirements as coordinated attack:

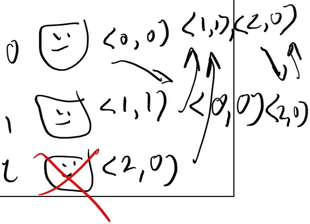
- **Agreement**: All non-faulty processes output same value.
- **Termination**: Protocol finishes in bounded rounds.
- **Validity**: All inputs equal $\Rightarrow$ all non-faulty processes output input.
  - Equivalent version: common output is somebody's input.
  - We'll require this even if there are failures.

With crash failures this turns out to be possible.

Dolev-Strong 1983

Synchronous agreement with up to t crash failures in $t+1$ time.

$S_i \leftarrow \{\langle i, \mathsf{input}_i \rangle\}$ `// S stores known id-input pairs`
for $r \leftarrow 1$ **to** $t+1$ **do**
  Send S_i to all processes (including myself)
  Receive messages S_k
  $S_i \leftarrow \bigcup S_k$
return $f(S)$



This works for any reasonable f (smallest input, smallest id, ...).

Note f is same for all processes.

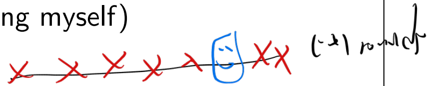
Easy properties:

- Termination: Loop only runs for $t+1$ rounds.
- Validity: f picks a value that started off as somebody's input.

Dolev-Strong: agreement

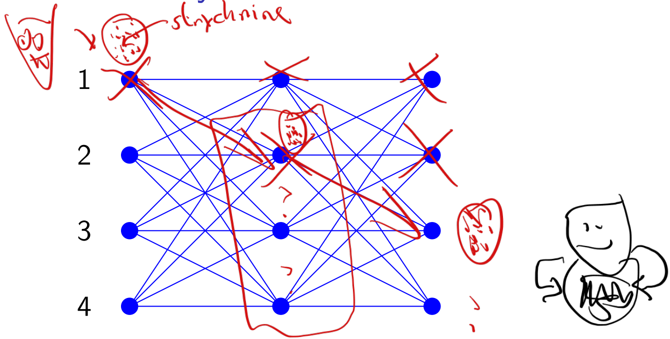
Agreement requires a bit more of an argument.

```
S_i ← {⟨i, input_i⟩} // S stores known id-input pairs
for r ← 1 to t + 1 do
    Send S_i to all processes (including myself)
    Receive messages S_k
    S_i ← ⋃ S_k
return f(S)
```



- By Pigeonhole Principle, $\exists$ a round s with no new failures.
- Let S_i^r = value of S_i after r rounds.
- Claim: $S_i^s = S_j^s$ for all non-faulty i and j.
  - Proof: i and j receive the same messages in round s.
- Claim: $S_i^r = S_j^r$ for all non-faulty i, j and all $r \geq s$.
  - Proof: Induction on r: $S_i^{r+1} = \bigcup S_k^r = S_i^r = S_j^r$.
- $\Rightarrow f(S_i^{t+1}) = f(S_j^{t+1})$.

Are $t+1$ rounds necessary?



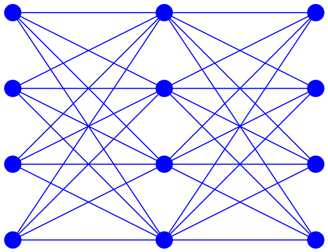
For the Dolev-Strong algorithm, yes:

- Suppose $f(S) = \max$.
- Start 1 with input 1, all others with 0.
- In each round r, crash process r and deliver S_r to $r+1$ only.

One process sees $\{0, 1\}$ and decides 1, while others decide 0.

Are $t + 1$ rounds necessary for any algorithm?

Yes! The lower bound is due to (Dolev and Strong, 1985).



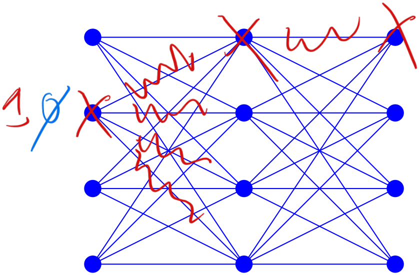
0 0 0 1 1 1

$\equiv_i \sim_p \equiv_{i+1}$

Strategy:

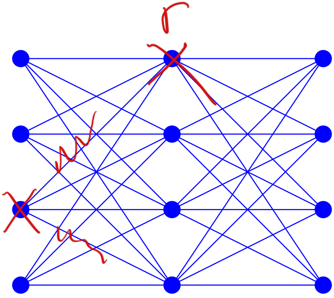
- Start with all-0 input execution.
- Build a chain of intermediate executions to change a 0 to a 1.
- Exploit indistinguishability to make each step in the chain preserve outputs.
- Arrive at all-1 input execution deciding 0.

How to hide a change



- Want to change input at process p.
- If we crash p, nobody else will know!
- But maybe they saw we crashed p.
  - Solution: remove p's outgoing messages one at a time.
  - For each removal, crash the receiver first.
- But then we see the receiver crashed!
  - Solution: Slaughter witnesses recursively.

Main lemma



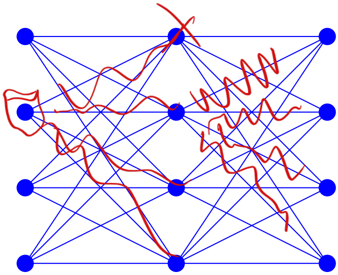
Claim: Let Ξ_0 be a t-round execution in which:

- At most one process crashes in rounds $1..r-1$.
- No process crashes in rounds $r..t$.

Then for any non-faulty q in Ξ_0, there is a sequence $\Xi_0, \Xi_1, \Xi_2, \ldots, \Xi_m$ such that:

- $\Xi_i \sim_p \Xi_{i+1}$ for some p that doesn't crash in either execution.
- Ξ_m has the same message pattern as Ξ_0, except
- q crashes fully (sends no messages) in round r in Ξ_m.

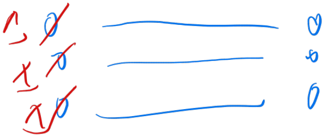
Proof of main lemma



By induction on $t - r$:

1. If $r = t$, remove q's outgoing messages one at a time.
   - Only the former recipient notices each step.
2. If $r < t$, pick a non-faulty recipient s.
   - 2.1 Use the lemma recursively to crash s.
   - 2.2 Remove the message to s.
   - 2.3 Use the lemma in reverse to uncrash s.
   - 2.4 Repeat until q has no outgoing messages in rounds r or higher.

Rest of proof



Given a t-round protocol that supposedly solves consensus:

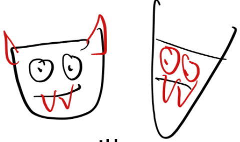
1. Use main lemma to crash a process at start of protocol.
2. Change its input from 0 to 1.
3. Repeat until all inputs changed.

Somewhere in this exponentially long sequence of executions, we violate agreement or violate validity.

$\Rightarrow$ Need at least $t + 1$ rounds.

(This is tight because Dolev-Strong algorithm uses $t + 1$ rounds.)

Byzantine failures



Worse than mere death: processes turn evil!

Byzantine agreement (Pease, Shostak, Lamport 1980).

- A Byzantine process can send any message it likes.
  - Allied to the adversary
  - Not bound by the protocol.
  - Seeks only to destroy.
- Constraints
  - Can't impersonate other processes.
  - Model is still synchronous.
- Name is offensive to Byzantine Empire (ended 1453).
  - May also annoy some Byzantine Orthodox Christians.
  - We are kind of stuck with it.

Where is your agreement, termination, and validity now?

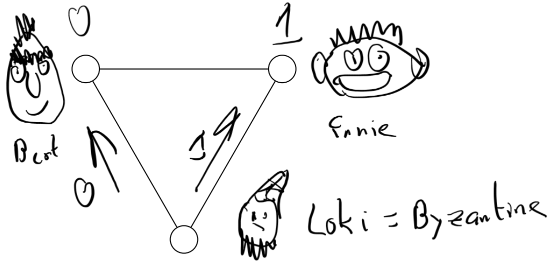
Requirements for Byzantine agreement

Revised to take into account evil nodes:

- Agreement: All non-faulty nodes output same value.
- Termination: Finish in bounded number of rounds.
- Validity: If all *non-faulty* nodes have same input, they all output this input.

Validity in particular means Byzantines can't hijack the protocol.

Impossibility results

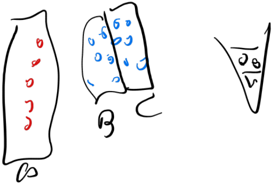


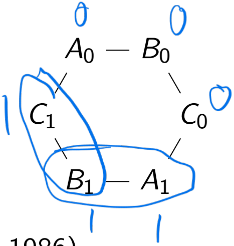
Dolev-Strong implies $t + 1$ rounds needed for t Byzantine failures.

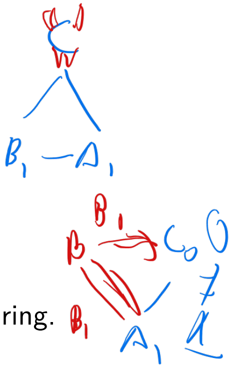
Pease et al. show $n \geq 3t + 1$ also required:

- Intuition: With 1 evil node out of 3 total nodes, evil node can play good nodes against each other.
- Generalizes to $t = n/3$ since each of these nodes can simulate $n/3$ nodes.
- Need more than a story to get a real proof.

Strategy for the Byzantine nodes







(Fischer, Lynch, Merritt 1986)

- Adversary simulates six non-faulty processes in a ring.
- Each copy X_i has input i.
- A_1 and B_1 think they are in an execution with evil C.
  - Validity means A_1 and B_1 both decide 1.
  - In general all X_i decide i.
- Now run evil B against good A_1 and C_0.
  - B acts like B_1 with A_1 and B_0 with C_0.
  - As in 6-process execution, A_1 decides 1, C_0 decides 0.
  - $\Rightarrow$ No agreement!

Next time

- Protocols for synchronous Byzantine agreement ($n \geq 3t + 1$).
- Impossibility of asynchronous agreement with one crash failure.