

Lectures 4 & 5: The Asynchronous Model and the FLP Impossibility Theorem

(Lecture Series on Foundations of Blockchains)

Relaxing the Synchronous Assumption

Synchronous model: (i) shared global clock (common notion of time);

(ii) every msg sent at time t arrives by time $t + t_l$.

↳ not OK for modeling the Internet (which has outages, denial-of-service (DoS) attacks, etc.)

Note: if there is a known bound Δ

On the maximum message delay, still in the synchronous model.

(can port over old protocols by inflating time steps)

(similarly if a priori known bound Δ on max difference between nodes' clocks)

Unsatisfying: (i) didn't force us to deal with unexpected outages / attacks (ii) produces stupid protocols in which nodes mostly idle

The Asynchronous Model

Note: 2 adversaries:
- Byz. nodes
- adversarial msg delivery

The Basic Idea: (i) no shared clock; (ii) no bound on the max msg delay.
(minimal assumption: every msg arrives, eventually)

Precise Model:

- pool M of outstanding (not-yet-delivered) messages
- while (TRUE):
 - one message (r, m) is delivered (to recipient r)
recipient message content
 - r can add any number of messages to M

(think of or chosen by
"adversary" who wants to
break a consensus protocol)

Assume: ① M initialized to $\{(i, 1)\}_{i=1}^n$ *← dummy msgs to get started*
② every msg eventually delivered (otherwise trivial impossibility via starvation)

Byzantine Agreement

Protocol: specifies what messages a node sends upon receipt of a new message (as a function of what the node knows — its private input + messages it has seen thus far)

The Problem: Byzantine agreement.

Setup: every node i has a private input $v_i^* \in V_i$. (for FLP, can even take $V = \{0,1\}$)

- ① termination: each node i eventually halts, with some output v_i .
- ② agreement: all honest nodes output same v_i .
- ③ validity: if all honest nodes have same input v^* , all honest v_i 's equal v^* .

The FLP Impossibility Theorem

Theorem: [Fischer-Lynch-Paterson 1985] For every $n \geq 2$, even for $f=1$, no deterministic protocol for Byzantine agreement satisfies termination, agreement, and validity in the asynchronous model.

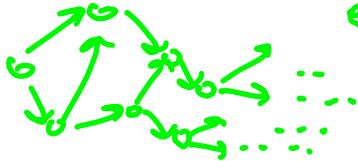
Note: formally separates what is possible in synchronous, asynchronous settings.

Comment #1: holds even if restrict adversary to one crash fault!

Comment #2: implies same impossibility for Byzantine broadcast + SMR.

Workarounds:

- allow randomized protocols (nodes can flip coins)
- use less extreme notion of asynchrony (see Lecture 6, partially synchronous model)



vertices = configurations
edges = transitions / message deliveries

Configurations

π = an allegedly
correct deterministic FLP
protocol

Definition: a configuration C records the state of every node (private input + sequence of msgs received thus far) and the message pool M . (not delivery of a msg (r, m) causes state transition $(\rightarrow C')$)

Definition: 0-configuration = no matter what adversary does from here, all honest nodes output 0. (Similarly, 1-configuration.)

ambiguous configuration: adversary has options forcing all-0s, forcing all-1s.

Proof plan: show there exists an infinite sequence $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow \dots$ given by Lemma 1 of ambiguous configurations. (i.e., might run forever, contradicting termination)

Π =
allegedly
current
protocol

An Initial Ambiguous Configuration

Lemma: There is a choice of private inputs s.t. the initial configuration C_0 is ambiguous.

Proof: Let X_i = initial configuration when private inputs of first i nodes are 1, last $n-i$ nodes are 0 : 

Note: (i) X_0 is a 0-configuration. } because T satisfies validity!
(ii) X_n is a 1-configuration.

Claim: One of the X_i 's is ambiguous.

Completing the Contradiction

Claim: one of the X_i 's is ambiguous.

111 - - - 1000 - - - 0
|
i
n-i

Proof of Claim: (implies lemma 1)
must exist!

let i be such that (i) X_{i-1} a 0-configuration j (ii) X_i not a 0-config.

To finish proof: will argue that X_i must be ambiguous (ie, can't be a 1-config.)

① adversary has a strategy that forces all -1s output (since X_i not a 0-config.)

② suppose node i Byzantine, adversary executes Π as if an honest node with private input 0

$\Rightarrow X_i \Rightarrow \Pi$ proceeds exactly as if stated in $X_{i-1} +$ all nodes honest

\Rightarrow all honest nodes output 0 (because X_{i-1} a 0-configuration + honesty is one possible adversary strategy)

Extending the Ambiguous Sequence

Lemma 2: let C_i be ambiguous, and (r, m) a message in C_i 's msg pool.
Then, there exists a sequence of message deliveries such that:
(i) last step = delivery of (r, m) ;
(ii) leads to an ambiguous configuration.



Lemma 1 + Lemma 2 → FLP Theorem

let C_0 = ambiguous configuration promised by lemma 1.

To define C_{i+1} from C_i :

- let (r, m) = oldest message in C_i 's pool

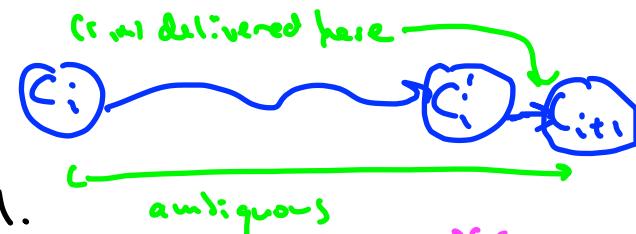
- let C_{i+1} = result of lemma 2

\Rightarrow all of $C_0, C_1, C_2, C_3, \dots$ ambiguous (by lemma 2)

(along with all of the intermediate configurations)

Note: if (r, m) added to pool at or before configuration C_i
 \Rightarrow delivered at or before $C_{i+(m)}$ size of msg pool immediately after (r, m) added to it

Proof Setup



Fix ambiguous configuration C_i , msg (r, m) in C_i 's pool.

Easy case: delivering (r, m) now (at C_i) yields an ambiguous configuration.

~~⊕~~ assume delivering (r, m) now yields a 0-configuration (say) analogous proof works for a 1-configuration

Definition: a configuration C is 0^* if: ← defined w.r.t.
the protocol π ,
msg (r, m)

(i) C reachable from C_i without delivering (r, m)

(ii) if (r, m) delivered at C , transition to a 0-configuration

\Rightarrow define 1^* and $ambiguos^*$ configurations analogously

Note: lemma 2's conclusion (\Rightarrow) there exists an ambiguous* configuration C_i^* .

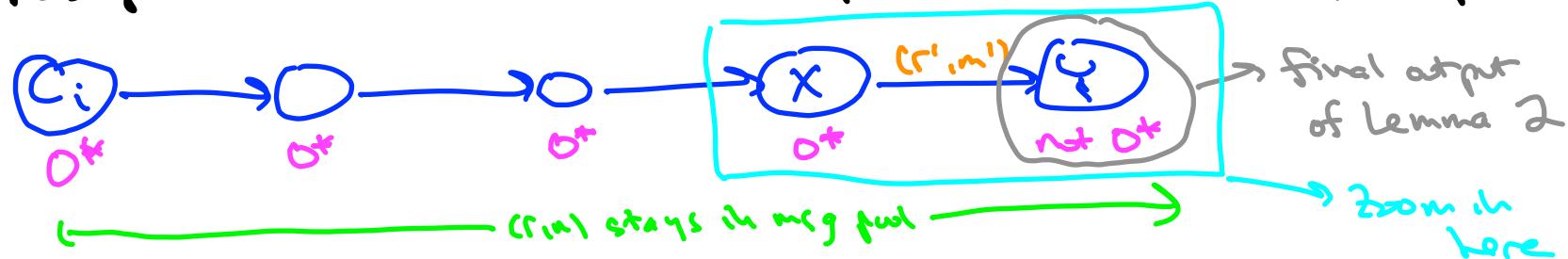
Note: assumption ~~⊕~~ $\Rightarrow C_i$ is a 0^* -configuration.

Hunting for a Non- O^* -Configuration

Goal: find an ambiguous*-configuration. Assumption: C_i is a O^* -configuration.

Note: must exist at least I^* - or ambiguous*-configuration.
- if all reachable configurations were O^* , C_i would be a O -config, not an ambiguous config.

Notation: let $Y = \text{non-}O^* \text{ configuration closest to } C_i$ (i.e., fewest # of message deliveries). let X be Y 's predecessor in shortest $C_i \rightarrow Y$ path.

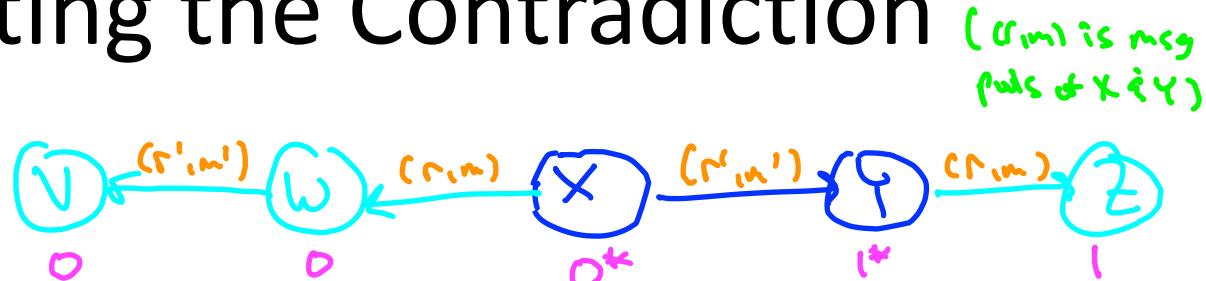


Claim: Y cannot be a I^* -configuration (so, must be ambiguous*).

Completing the Contradiction

Proof of Claim:

Assume (for contradiction)
 γ is a \mathbb{I}^{**} -config.



Point: (i) msg order $(r, m), (r', m')$ \longleftrightarrow O-config (V) $\xrightarrow{*}$
(ii) msg order $(r', m'), (r, m)$ \longleftrightarrow I-config (Z) $\xrightarrow{**}$

Case 1: $r \neq r'$. All nodes see same sequence of received msgs in (i), (ii)
 \Rightarrow all (honest) nodes behave identically (+ halt w/same output) in (i), (ii) [contradiction]

Case 2: $r = r'$. Suppose r is Byzantine.
→ indistinguishable to all honest nodes (See identical sequences
of messages), same output either way

- Scenario #1: r receives m before m' $\xrightarrow{(4*)}$ final outputs = 0, r follows Π honestly
- Scenario #2: r receives m' before m $\xrightarrow{(***)}$ final outputs = 1, r acts as if
received m before m' & is then following Π honestly,
Contradiction!

Discussion

- ① Impossibility results make clear the compromises + trade-offs required
(e.g., need to choose between safety + liveness with unbounded msg delays)
- ② clarifies which assumptions matter
(e.g., fkl in synchronous model, synchronous vs. asynchronous models)
- ③ guides you to the "right" model to reason about consensus/blockchain protocols
(e.g., the partially synchronous model)
- ④ how cool are these proofs?!?

Next: Tardos and the partially synchronous model.