Multi-agent simulation of attacks on distributed protocols: application to order fairness in Hyperledger Fabric

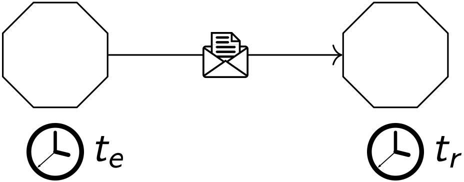
Ongoing work at CEA-LIST LICIA

Erwan Mahe - 2024 04 05 @ GT MFS

Summary

Distributed Ledgers & Order Fairness

Communication models

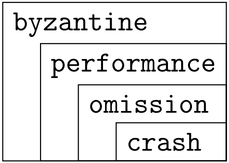


- asynchronous: $t_r - t_e \in]0, +\infty]$ so may never be received
- synchronous: $\exists\ \Delta \in]0, +\infty[$ s.t., $t_r - t_e \in]0, \Delta]$
- eventually/partially synchronous: $\exists\ GST \in]0, +\infty[$, $\exists\ \Delta \in]0, +\infty[$ s.t., $(t_e > GST) \Rightarrow (t_r - t_e \leq \Delta)$

Impossibility of distributed consensus with one faulty process
- Fischer, Lynch & Paterson - Journal of the ACM 1985

Consensus in the presence of partial synchrony
- Dwork, Lynch & Stockmeyer - Journal of the ACM 1988

Failure models

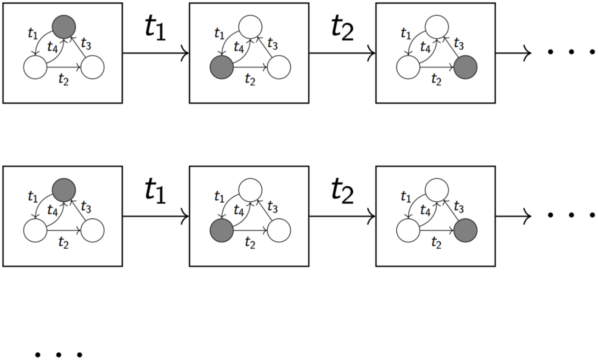


- crash: terminates prematurely
- omission: some events are not delivered
- performance: concerns timing constraints
- byzantine: unexpected behavior

What Good Are Models and What Models Are Good ?

- Schneider - Distributed Systems 2nd Ed 1993

Principle of Distributed Ledgers



- distributed ledgers are replicated SMs
- key-value store content ⇔ (global) state ⇔ any local state
- state change ⇔ transaction delivery
- coherence ⇐ eventually delivering the same transactions in the same order

Context

In the following we consider:

- Blockchains as the means to implement Distributed Ledgers (i.e., transactions are sequentially batched into blocks)
- non-revocable blockchains (i.e., once a block/a transaction is delivered there are no rollbacks)

Properties of interest:

- most protocols/algorithms involved in Blockchains are Byzantine Fault Tolerant
- but tolerance w.r.t. specific properties (often related to consistency and liveness)
- consistency refers to the fact that, eventually, every node agrees on the same list of transactions
- but nothing is said about the actual order that is agreed upon

Order-related fairness properties for distributed ledgers:

- pertinent (frontrunning/sandwich attack → MEV bots[1])
- only recently formalized (2020 paper)
- not upheld by most existing protocols

[1] ∼675 million$ gains on Ethereum alone between 2020 and 2022 forbes.com/sites/jeffkauflin/2022/10/11/the-secretiveworld-of-mev-where-crypto-bots-scalp-investors-for-big-profits/

Definition of order fairness

Given n nodes and any two pairs (t, t') of delivered transactions, $\texttt{before}(t, t')$ counts the number of times, across all n nodes, that t is received before t' and:

- *receive-order fairness* :=
  if $\texttt{before}(t, t') > n/2$ then t must be delivered before t'
- *block-order fairness* :=
  if $\texttt{before}(t, t') > n/2$ then t must not be delivered in a block after that in which t' is delivered
- *differential-order fairness* :=
  if $\texttt{before}(t, t') - \texttt{before}(t', t) > 2 * f$ with f a specific Byzantine threshold, then t must be delivered before t'

`Order-Fairness for Byzantine Consensus`
- Kelkar, Zhang, Goldfeder & Juels - CRYPTO 2020

`Quick Order Fairness`
- Cachin, Micic, Steinhauer & Zanolini - FC 2022

Motivation for empirical evaluation & simulation

Theoretically:

- *receive-order fairness* is impossible to uphold
- *block-order fairness* only considered in Aequitas [Kelkar et al - CRYPTO 2020]
- *differential-order fairness* only considered in algo from [Cachin et al - FC 2022]

In practice:

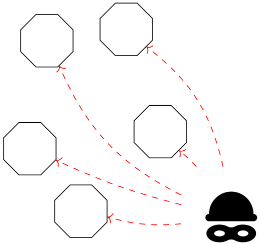
- no protocols used in the industry consider order-fairness
- how to evaluate vulnerabilities related to these properties ?

Adversary-augmented simulation:

- scalable w.r.t. system and properties
- fine-grained parameterization of system and attacker
- observation of attack effects

Adversary-Augmented Simulation

The Adversary



An external (w.r.t. the system) entity characterized by:

- its assumptions
- its goals
- and its capabilities

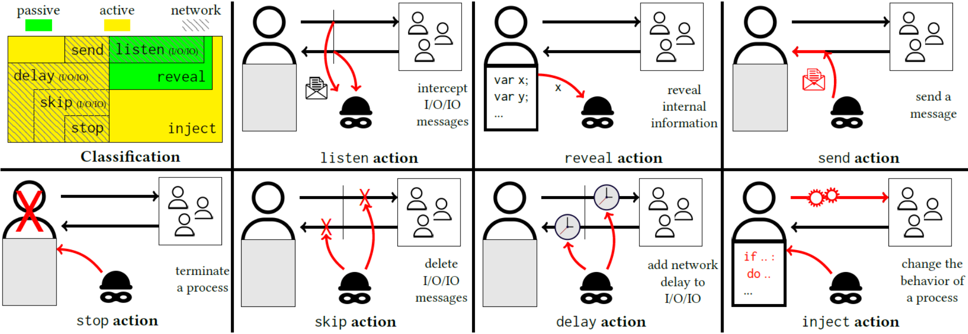
`The role of the adversary model in applied security research`

- Do, Martini & Choo - Computers & Security vol 81 2019

Our adversary model

Assumptions	Goals	Capabilities
Environment (system & assumptions): - Communication Model - Failure Model Resources (binding capabilities): - Available Information - Limited budget w.r.t. resources	property violation	adversarial actions

Adversarial actions



- `listen` : network eavesdropping, sniffing, snooping
- `reveal` : access with read permission, side-channel, memory scanning
- `skip` & `delay` : Denial of Service, man-in-the-middle (control over infrastructure)
- `inject` : admin access, code-injection (buffer overflow etc.)

Enabled actions w.r.t. assumptions

Fail. \ Comm.	Synch.	Async.	Event. Synch.
Crash	reveal stop delay $t+\delta<\Delta$	reveal delay	reveal stop delay $o \geq GST \Rightarrow t+\delta<\Delta$
Omission	reveal skip delay $t+\delta<\Delta$	reveal delay	reveal skip delay $o \geq GST \Rightarrow t+\delta<\Delta$
Performance	reveal delay	reveal delay	reveal delay
Byzantine	inject	inject	inject

Also limited w.r.t. resources assumptions
(e.g., related to Byzantine thresholds i.e., cannot apply actions to more than f distinct nodes)

Use case

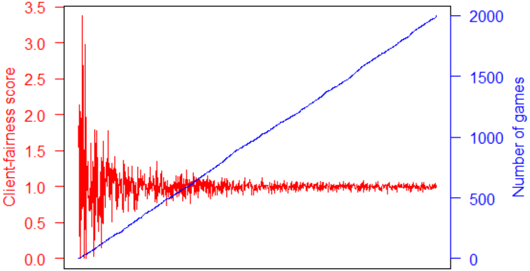
Application layer & goal of the adversary

Let us consider a use case with clients competing to solve successive puzzles:

- a new puzzle is revealed regularly
- upon solving a puzzle, a client sends a transaction with the solution
- for any given puzzle, the first delivered transaction that contains its solution determines the winner

In a concrete execution, over g repeated puzzles:

- if $\%g(c)$ denotes the percentage of games won by client c
- and if n_c denotes the number of clients
- then, supposing all clients have the same aptitude, the game is client-fair iff $\%g(c)$ converges towards $\frac{1}{n_c}$ as g increases

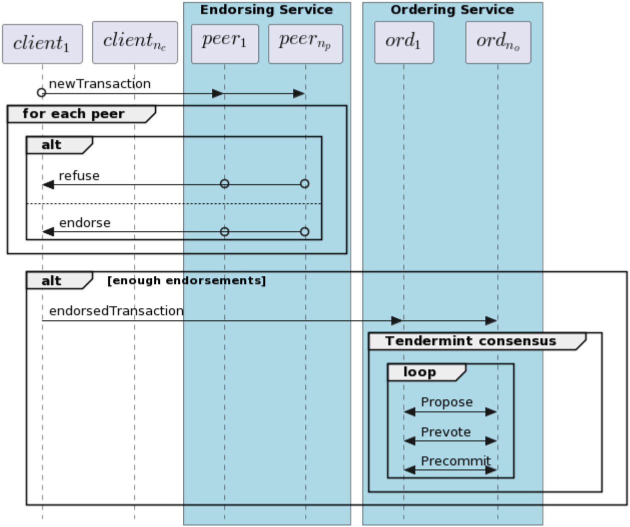


Goal: $\texttt{score}(c) = n_c * \%g(c)$ converges to value $< 1 - \varepsilon$ for a specific target client c

e.g.:

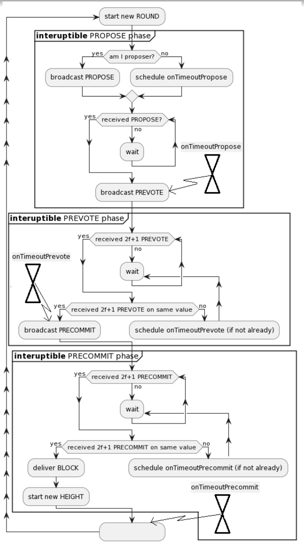
$$\phi = (g > 1500) \wedge (\texttt{score}(c) < 0.75)$$

Hyperledger Fabric with Tendermint



https://www.hyperledger.org/projects/fabric

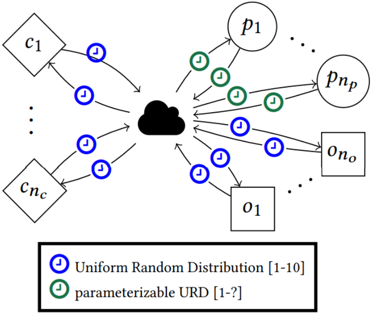
Rough sketch of Tendermint



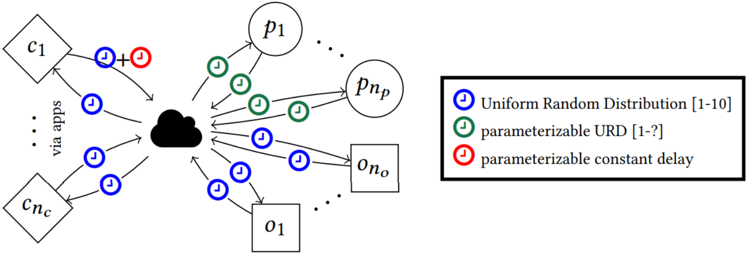
https://tendermint.com

Parameterization

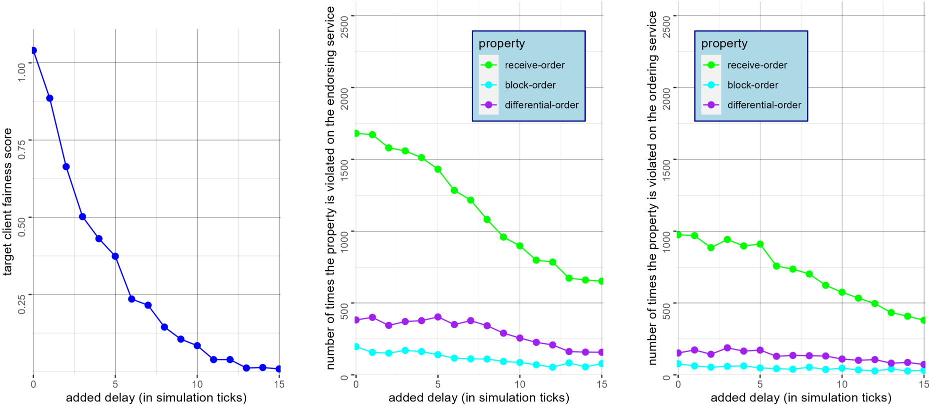
- 3 clients
- $55 = 3 * 18 + 1$ orderers
- 50 peers (25 required endorsements)
- a new puzzle reveal every 10 ticks
- solvable in at most 5 ticks by each client
- baseline communications delays distribution (see right)



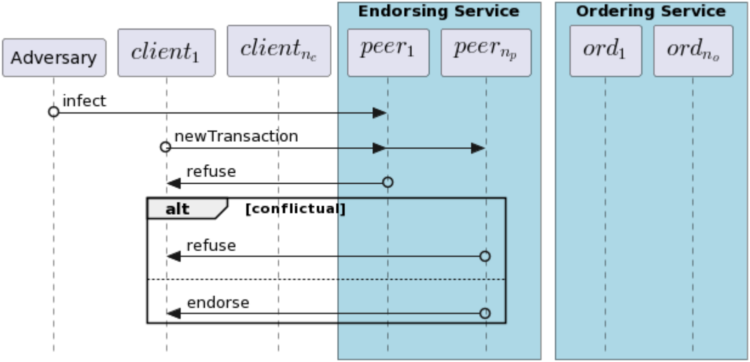
Delay attack principle



Delay attack results



Peer sabotage principle



Peer sabotage principle

$\mathcal{P}_p(t < z) :=$ probability that the client receives an endorsement from peer p for transaction t before timestamp z

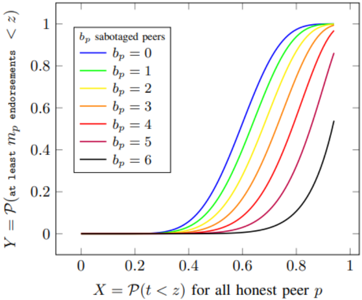
If i.i.d. variables we have a X such that $\forall p \in S_p$, $\mathcal{P}_p(t < z) = X$ and $\mathcal{P}_p(t \geq z) = 1 - X$

Among n_p trials, the probability of having exactly $k \leq n_p$ peers endorsing t before z is:

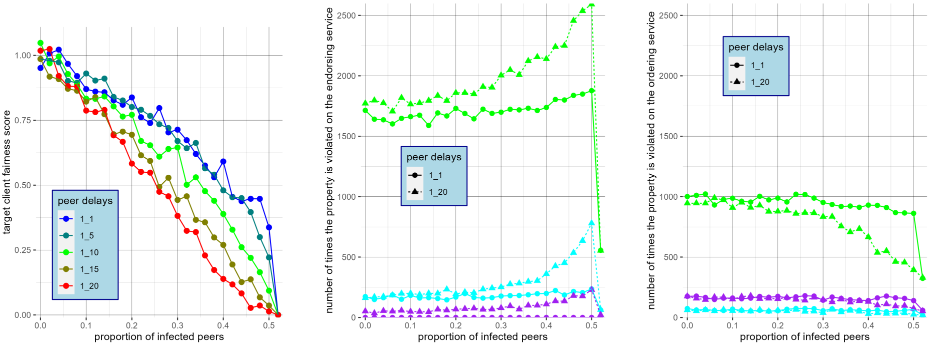
$$\mathcal{P}(k_{\texttt{ endorsement}} < z) = \binom{n_p}{k} * X^k * (1 - X)^{n_p - k}$$

Given $b_p \leq n_p - m_p$ the number of sabotaged peers, the probability Y of having at least $m_p \leq n_p$ distinct endorsements before z is:

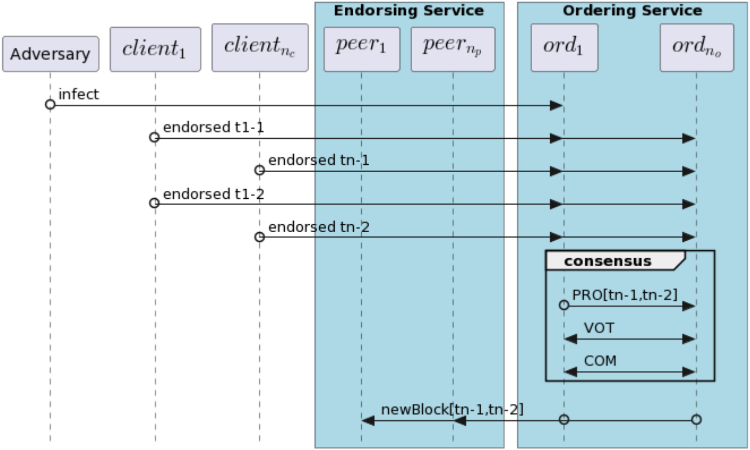
$$Y = \sum_{k=m_p}^{np-bp} \binom{n_p - b_p}{k} * X^k * (1 - X)^{n_p - b_p - k}$$



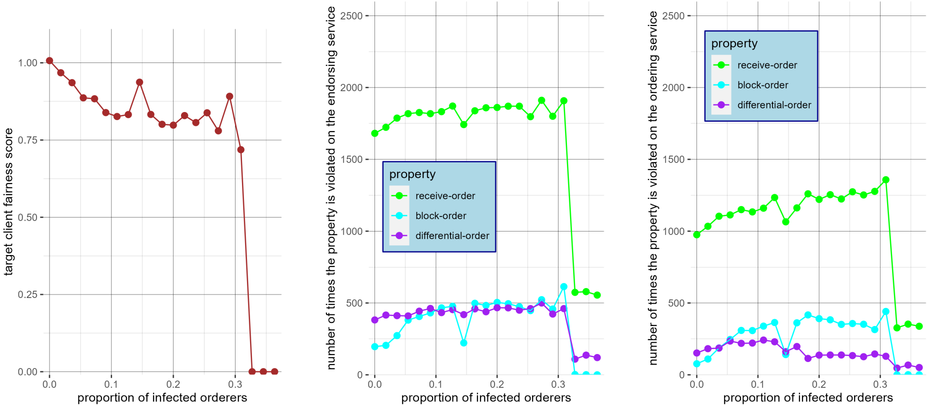
Peer sabotage results



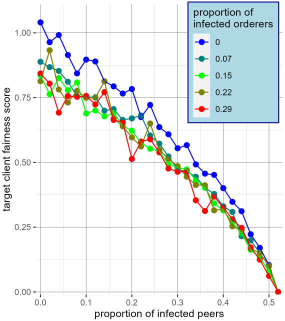
Orderer sabotage principle

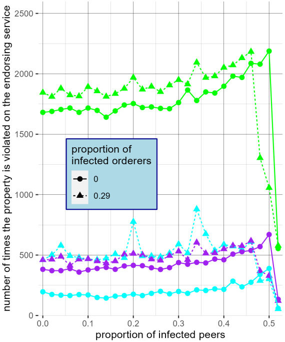


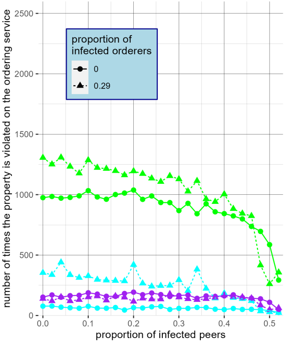
Orderer sabotage results



Peer & orderer sabotage results







Resources

WIP article:

- https://arxiv.org/abs/2403.14342

MAX code & means to reproduce the experiments:

- adversarial model in P2P layer : https://gitlab.com/cea-licia/max/models/networks/max.model.network.stochastic_adversarial_p2p
- distributed ledger interface and puzzle use case : https://gitlab.com/cea-licia/max/models/ledgers/max.model.ledger.abstract_ledger
- Tendermint model : https://gitlab.com/cea-licia/max/models/ledgers/max.model.ledger.simplemint
- HF model : https://gitlab.com/cea-licia/max/models/ledgers/max.model.ledger.simplefabric
- experiments : https://gitlab.com/cea-licia/max/models/experiments/max.model.experiment.fabric_tendermint_client_fairness_attack

Conclusion

Contributions:

- an adversary model for multi-agent simulation of attacks on distributed protocols
- implementation in a simulator
- design & implementations of attacks on client-fairness on HF
- evaluation of impact on order fairness

Thank you for your attention
Any questions ?