Beyond Blockchains

Adi Seredinschi

Informal Systems

Distributed Algorithms // EPFL Fall’20
My Team

- Vienna
- Lausanne (Innovation Park)
- Berlin
- Toronto
- Belgrade
- Paris
- Zug
Verifiable distributed systems and organizations.

We envision an open-source ecosystem of cooperatively owned and governed distributed organizations running on reliable distributed systems.

what we do

Systems (of Machines)
Informal designs, implements, and formally verifies distributed systems and protocols, including blockchain systems like Tendermint and Cosmos.

Organizations (of Humans)
Informal develops tools to simplify the operation of organizations by leveraging open-source development, plaintext data, and distributed version control systems.
Roadmap

1. **Blockchains & BFT Consensus**
   Tendermint Core

2. **Beyond Blockchains**
   Inter-Blockchain Communication (IBC)
Consensus vs. Blockchain

Consensus: “processes propose values and have to agree on one among these values”

Models:
- **Benign**: crash-stop processes (P, <>P algorithms)
- **Today**: Byzantine processes
  - e.g., buggy, malicious & adversarial, rational
  - Authenticated links (dig. sigs. assumption)

Blockchain
- Can mean different things
- Often, the whole stack is the “blockchain”
- Builds on a consensus core -> total order
  - Multiple instances of consensus
- Also known as: **Replicated State Machine**
Basic Tendermint BFT Consensus

Properties

Validity Predicate-based Byzantine Consensus (Crain et al, 2017)

1. **Validity**: A decided value is *valid*, i.e., it satisfies a predicate \(\text{valid}()\).
2. **Agreement**: No two correct processes decide differently.
3. **Termination**: All correct processes eventually decide on a value.
4. **Integrity**: No correct process decides more than once (w.r.t. a consensus instance).

Tendermint Algorithm Overview

- Similar in spirit to Consensus algorithm III
- We assume a correct “majority”:
  - > 2/3 processes are correct (quorums)
  - < 1/3 processes may be Byzantine
  - N = 3f + 1
- Processes take turns in the role of proposer
  - Round-based model
  - Each round has a predefined proposer
  - Goal: proposer locks everyone on a value

Consensus algorithm III

- A uniform consensus algorithm assuming:
  - A correct majority
  - A <>P failure detector
- > ½ processes are correct
- N = 2f + 1
- Benign (non-Byzantine case)

System model

- Partially synchronous system model (DLS88)
  - Communication between correct processes is reliable and timely (bounded with Delta) after GST

- At most f processes can be faulty (Byzantine faults)

- Gossip communication:
  - If a correct process receives a message m at time t, all correct processes will receive m before max(t, GST) + Delta

![Diagram showing the relationship between communication and time with GST (Global Stabilization Time)]
Rounds

Round
- Proposal, Prevote, Precommit → decision
- Has a predefined proposer process

Locking
- Locked values means PRECOMMIT was sent
- Two variables keep track of the last locked value:
  - lockedValue
    - Retains the value itself; initially nil
  - lockedRound,
    - Initially -1
Algorithm Overview (good case)
Algorithm Overview (complete)
Tendermint consensus algorithm

P1’s round

lockedValue = nil
lockedRound = -1
validValue = nil
validRound = -1
step = Propose

lockedValue = v
lockedRound = r
slow

<PROPOSAL, r, v, vr>
<PREVOTE, r, id(v)>
<PRECOMMIT, r, id(v)>

decide
Novelties

1. **Gossip layer** (instead of all-to-all links)
2. **Light client** (e.g., a mobile phone)
3. **Robustness** (**Jepsen** tests)
4. **ABCI** — interface b/t consensus and application layer

Open Challenges

1. **Rust implementation**
   (Focus on correctness: Model-based testing, Mocking)
2. **Formal verification**
   (TLA+, Stainless, Prusti, Isabelle)
3. **Inter-blockchain Communication** — **IBC**
Roadmap

1. **Blockchains & BFT Consensus**
   Tendermint Core

2. **Beyond Blockchains**
   Inter-Blockchain Communication (IBC)
   Quick Overview
IBC: Problem statement

1. **WHAT** is communicated?
2. **WHO** communicates?
3. **HOW** to perform the communication?

**Application:** Token Transfer

- Consensus Algorithm
- Networking (links, broadcast)
1. What is communicated?

Application: Token Transfer

Consensus Algorithm  Chain Store
Networking (links, broadcast)

(Australian Open Ledger)

Balances:
_Alice: 10  
_Bob: 20  
...
What is communicated?

Application: Token Transfer

- Consensus Algorithm
- Chain Store
- Networking (links, broadcast)

Application: Token Transfer

- Chain Store
- Consensus Algorithm
- Networking (links, broadcast)

[Credit Suisse and UBS logos]
2. Who communicates?

Recall that this is a replicated state machine

https://hostingfacts.com/where-website-live/
2. Who communicates?

This is not efficient...
2. Who communicates?

**Application:**
- **Token Transfer**

**Consensus Algorithm**

**Chain Store**

**Networking (links, broadcast)**

---

**Application:**
- **Token Transfer**

**Consensus Algorithm**

**Chain Store**

**Networking (links, broadcast)**

---

**IBC Relayer**

**read/write**
3. How?
Analogy with TCP/IP
3. How? Assuming no Byzantine faults

1. Read state from node
2. Construct packet
3. Write packet to node
4. Replicate via consensus to all nodes
5. Write into store
6. Read state from node
Student Projects

- **AT2** ~ implementation of consensus-less payments
- **IBC** ~ a “TCP/IP” for interconnecting ledgers
- **Rust** ~ Implementation of Tendermint modules (consensus, mempool, fast sync) using Prusti and Rust.
- **Stainless** ~ Implementation of Tendermint modules (consensus, mempool) using Stainless and Scala.
- **Facebook Libra** ~ comparative analysis of consensus algorithms.
- **Mempool** (performance analysis); adversarial engineering.

Complete list:
[https://dcl.epfl.ch/site/education#collaborative_projects](https://dcl.epfl.ch/site/education#collaborative_projects)

Thank you!

adi@informal.systems