On the way to Omniledger: adding transaction batching and ByzcoinX to skipchains

Raphaël Dunant

DEDIS lab

Supervisor: Linus Gasser

Responsible: Prof. Bryan Ford
Omniledger improvements

- Block state storing
- Transaction Batching
- Improve decentralised Signing
Summary

- Introduction (done)
- Collections
- ByzcoinX Quick Answers
- Simulation results
- Conclusion (results, lessons learned, etc.)
Patricia Merkle Tree

\[ H_{\text{root}} = h(H_{1,0}, H_{1,1}) \]

\[ H_{1,0} = h(H_{2,0}, H_{2,1}) \]

\[ H_{2,0} = h(K_1, V_1) \]

\[ K_1 = \text{spider} \]

\[ V_1 = \text{eight} \]

\[ H_{2,1} = h(K_4, V_4) \]

\[ K_4 = \text{snake} \]

\[ V_4 = \text{zero} \]

\[ H_{1,1} = h(K_0, V_0) \]

\[ K_0 = \text{cat} \]

\[ V_0 = \text{four} \]
ByzcoinX: Decentralized Witness Cosigning

Phase 1: Announcement
(send message-to-witness, optional)

Phase 2: Commitment
(collect aggregate commit)

Phase 3: Challenge
(send collective challenge)

Phase 4: Response
(collect aggregate response)
Objectives

- Understand and Document complete collection library
- Improve drastically running time of ByzcoinX
- Have nice, documented, tested code
Collections Code Cleaning

Packages collection is a Merkle-tree based data structure to securely and verifiably store key/value associations on untrusted nodes. The library in this package focuses on ease of use and flexibility, allowing to easily develop applications ranging from simple client-server storage to fully distributed and decentralized ledgers with minimal bootstrapping time.

```go
package collection

// Collection represents the Merkle-tree based data structure.
// The data is defined by a pointer to its root.

type Collection struct {
    root  *node
    fields []Field
```
ByzcoinX Quick Answers

Phase 1: Announcement
(send message-to-witness, optional)

Phase 2: Commitment
(collect aggregate commit)

Threshold = 5
Simulation results

- 50 Nodes, 6 Subleaders, Default Leafs Timeout: 417ms, average of 10 tries
- Threshold: \[ \lceil \frac{2}{3} \cdot 50 \rceil = 34 \]
- 2.9 GHz, 4 Core, 4MB cache, 8GB DDR3-1600 RAM
Future work

• Collections
  – Store on hard drive
  – Handle transactions conflicts more finely

• ByzCoinX
  – Add backward Compatibility
  – Rework Timeouts
  – Improve security

• Add more unit tests
Conclusion

- Complete, working and reusable collections code
- Quick ByzcoinX performances
- Will be used in production
- Scalable and tested
- Can still get better
- Personal improvement