Timechain
A Time Synchronization Protocol based on Distributed Network

LEUNG TSZ HIN (1155079351)
SUPERVISED BY PROF. LYU RUNG TSONG MICHAEL
Recap

- Research on Network Time Protocol
  - How it works
  - Security concerns

- Proposed Timechain
  - Blockchain for timekeeping
  - Demonstration
Agenda

- Background research
- Timechain
- Testing and Evaluation
Consensus Algorithm

- Preventing Byzantine faults:
  - Ensuring the proposed block in the chain generated by a node is legitimate
  - Preventing malicious users from successfully derailing the system
### Consensus Algorithm

#### Proof of Work (PoW)

<table>
<thead>
<tr>
<th>Block content</th>
<th>Nonce</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>0001</td>
<td>888B19A43B151683C87895F6211D9F8640F97BDC8EF……</td>
</tr>
<tr>
<td>0002</td>
<td>0002</td>
<td>4FAC6DBE26E823ED6EDF999C63FAB3507119CF3CB……</td>
</tr>
<tr>
<td>0003</td>
<td>0003</td>
<td>446E21F212AB200933C4C9A0802E1FF0C410BBD75F……</td>
</tr>
<tr>
<td></td>
<td></td>
<td>......</td>
</tr>
<tr>
<td>1234</td>
<td>1234</td>
<td>03AC674216F3E15C761EE1A5E255F067953623C8B38……</td>
</tr>
</tbody>
</table>
Consensus Algorithm

- Proof of Stake (PoS)
  - Probability of creating a block depends on the amount of stake
  - Economic incentive
Consensus Algorithm

- Practical Byzantine Fault Tolerance (pBFT)

https://www.geeksforgeeks.org/practical-byzantine-fault-tolerancepbf/
Consensus Algorithm

- Computational overhead
- Network overhead
- Fault tolerance
- Transaction finality
- Scaling
Timing Mechanisms

- Physical Time
- Logical Time
Timing Mechanisms

- Lamport Clock

Timing Mechanisms

- Vector Clock

Timing Mechanisms

- Network Time Protocol
- Real time

Timing Mechanisms

- Berkeley’s Algorithm
- No trustworthy time source

https://www.geeksforgeeks.org/berkeleys-algorithm/
Timing Mechanisms

- Vector clock
- Lamport clock
- Decentralized
- Timechain
- Logical time
- Physical time
- Centralized
- Berkeley’s Algorithm
- NTP

Page 15
NTP Clock Selection

- Truechimers vs Falsetickers
- $[\theta_0 - \lambda, \theta_0 + \lambda]
- \(\theta_0\): Measured offset
- \(\lambda\): Root distance
  - \(\frac{1}{2}\) Round trip delay + root dispersion

\[\theta_0 - \lambda \quad \theta_0 \quad \theta_0 + \lambda\]
NTP Clock Selection

- Marzullo’s algorithm
  - intersection interval: the smallest interval containing points from the largest number of correctness intervals
NTP Clock Selection

Marzullo’s algorithm

https://en.wikipedia.org/wiki/Marzullo%27s_algorithm
Timechain

- Decentralized, distributed timekeeping
- Physical time + logical time
Consensus Algorithm

- Proof of Work (PoW)
  - Highly scalable
  - No stake
  - Better fault tolerance
### Timechain

#### Block

<table>
<thead>
<tr>
<th>Index</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hash</th>
<th>Previous Hash</th>
<th>Nonce</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hash</th>
<th>Previous Hash</th>
<th>Nonce</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hash</th>
<th>Previous Hash</th>
<th>Nonce</th>
</tr>
</thead>
</table>
Operation

Solving PoW

Node

New

Hash

Node

Solving PoW

$\text{time}$

$\text{time}$

$\text{time}$

$t_1$

$t_2$
Operation

Solving PoW

Node

New → Hash

Append

Solving PoW

Node

Verify

Append
Operation

Solving PoW

Sync

t_1

Broadcast

Node

Append

t_2

Verify

Node

Broadcast

Solving PoW

Append
Implementation

- Deployed working instance on 7 machines
- Timechain vs NTP in clock selection
- How to choose a time among the blocks
Implementation

- NTP Test:
Implementation

- NTP Test:
Implementation

- Timechain Test:
Nonce: d40139dd
Hash: 09026dc1d73210c08c7f315258cd5b93d12306a0f7453813a6179eb604fe26a0
PrevHash: 0cfcdb49513654c10ae8ed6625d404140d2cace29d7c68c11d4e4bd6998db68dcc

Index: 940
Timestamp: Tue Apr 23 2019 11:02:20 GMT+0800 (Hong Kong Standard Time) (1555988540510610344)
Nonce: 1081a123
Hash: 01b8fbb2160c30bf06dca2baf4570b670bec4d25f33f509a91c203af76b2097c
PrevHash: 09026dc1d73210c08c7f315258cd5b93d12306a0f7453813a6179eb604fe26a0

Index: 941
Timestamp: Tue Apr 23 2019 11:02:17 GMT+0800 (Hong Kong Standard Time) (1555988537101008640)
Nonce: 672a356f
Hash: 02504cb18494329b20d2b0a5661b6a330d1445c20e158048d0cebd43e4f2fac31
PrevHash: 01b8fbb2160c30bf06dca2bff4570b670bec4d25f33f509a91c203af76b2097c
Choosing a Time

- **Mean(n)**
  - n: number of samples
  - \(0 \left(\frac{4}{7}\right) + 2 \left(\frac{2}{7}\right) - 2 \times \left(\frac{1}{7}\right) = 0.285\) mins
Choosing a Time

- Median(n)
  - n: number of samples
Choosing a Time

- Average of timestamps of nodes between 25\textsuperscript{th} percentile and 75\textsuperscript{th} percentile
Conclusion

- Timechain capable to keep physical time and logical ordering
- There may exist better methods to compute a more accurate time
- Intended to provide a more creditable time source alongside with other timing mechanisms