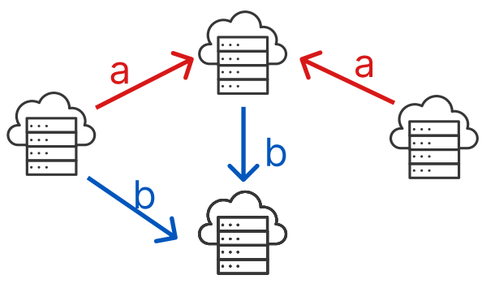
CTBP

Compact Transaction Broadcast Protocol

Chang Hyun Park
Ryang Jin Lee
CSE 310

Stony Brook University

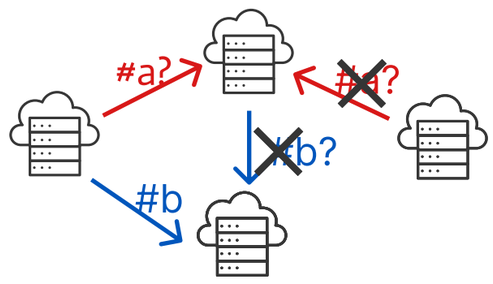
BACKGROUND & SOLUTION



Background

Flood Based Protocol:

- Unnecessary Duplication of Data Across Nodes
- Flooded Protocol



Solution

Our Implementation(CTBP):

- Use unique hashed tag to check if peer has the packet
- If so, do not send the packet
- “Smart Gossiping”

OUR IMPLEMENTATION

The CTBP protocol enhances traditional mempool systems by adding two new protobuf messages:

```
message SeenTx {
   bytes tx_key = 1;
   optional string from = 2;
}
```

```
message WantTx {
   bytes tx_key = 1;
}
```

a) Outbound Logic

SeenTx Mechanism: Broadcasts for "new" transactions. Possible to receive duplicates. Initial connections use SeenTx for pool synchronization.

RPC Transactions: Don't trigger SeenTx as their broadcast implies recognition.

WantTx Messages: Used for confirming transaction absence, sent point-to-point following a SeenTx or in specific response scenarios.

Transaction Requests: Avoids simultaneous WantTx requests for the same transaction, respecting network latency.

b) Inbound Logic

In-Memory Maintenance: Transactions are lost upon node shutdown. To accommodate restarted nodes, transaction pools track peer states based on each connection, not NodeID.

Handling Txs Messages: Distinguish between responses and broadcasts. Validate transactions and either reject, evict, or accept and broadcast a SeenTx (excluding the sender).

Responding to SeenTx Messages: Record peer recognition of transactions. Ignore messages if the transaction was previously rejected, already in possession, or recently evicted but now viable. If new, request it using WantTx or wait for a Txs message.

Receiving WantTx Messages: Respond with a Txs message if holding the transaction. If not, either send a WantTx or wait for the requesting peer to ask another peer.

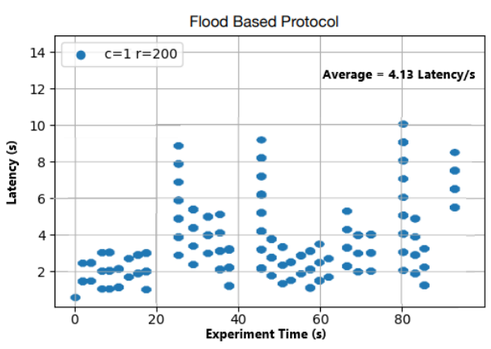
TESTING

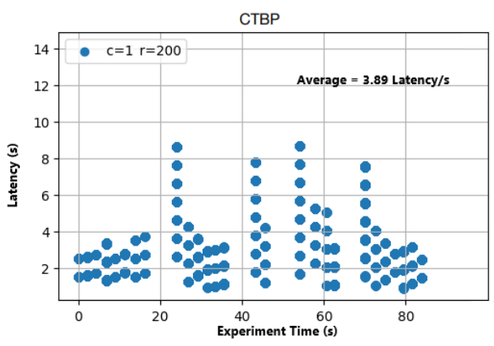
Testing was done by creating peer-to-peer network consisted of 200 nodes on cloud, specifically using open source framework Testground.

a) Latency

CTBP shows better metrics in Latency, with

- Lower Average Latency
- Greater Consistency
- Fewer and Lower peaks





b) Mempool Size

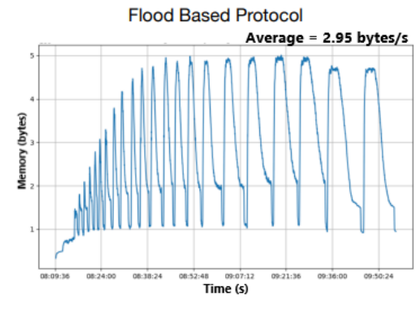
CTBP shows better metrics in Mempool Size, with

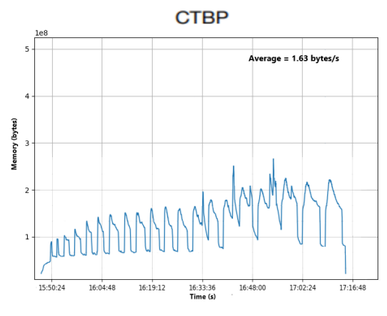
- Faster Average Transaction process
- Lower Peaks

c) Memory Usage

CTBP shows better metrics in Latency, with

- Lower Average Latency
- Greater Consistency
- Fewer and Lower peaks

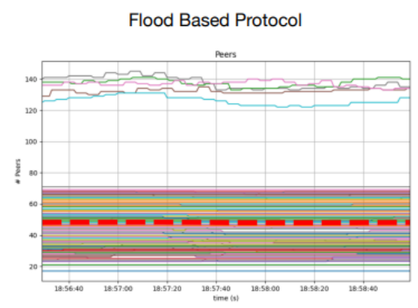


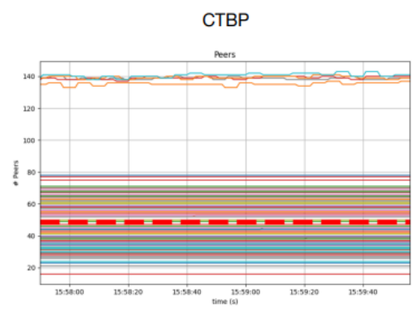


d) Rate of Transaction Processing

CTBP shows better metrics in Latency, with

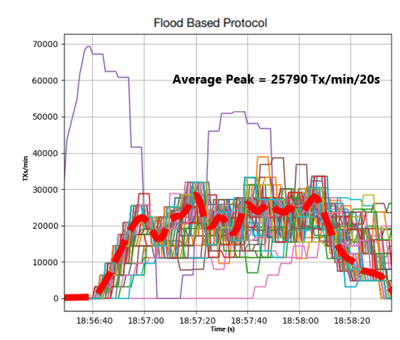
- Lower Average Latency
- Greater Consistency
- Fewer and Lower peaks

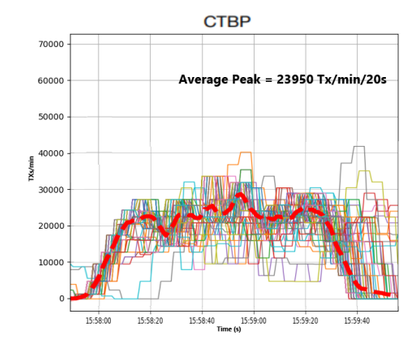




e) Peers

Both graph demonstrate a similar rate of peer connectivity over time, ensuring that new implementation of CTBP did not introduce unstableness for peer connection.





CONCLUSION & POSITIVE EFFECTS

- **Network Scale**
- **Data Transmission Efficiency**
- **Cost Savings**
- **Financial Impact**

References

1. Tendermint: https://tendermint.com/
2. Tendermint Source Code: https://github.com/tendermint/tendermint
3. Implentation of CTBP: https://github.com/mattverse/tendermint/pull/1
4. TestGround: https://github.com/testground/testground