# Blockchain Consensus Protocol with Horizontal Scalability

K. Cong

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology

August 26, 2017

# Outline

# Outline

# The dangers of centralisation

- Technological advancements give us convenience
- But it puts central authorities in control
- Many are motivated by profit
- Not always in the interest of the "users"[1]

[1] Typically users of some free service X are, in fact, used by X.

# The dangers of centralisation: Examples

- Baidu's promoted search result on experimental medical care caused death of a student [1]
- Facebook can predict your opinions and desires better than your spouse [2]
- With intimate knowledge of the individuals, Facebook creates "psychographic" profiles in political campaigns [3]

# Blockchain: a new hope?

- Blockchains are distributed ledgers
- They enable large scale consensus
- An alternative to central authorities for the first time
- Some applications include:
  - Digital cash (e.g., Bitcoin)
  - Domain name system (e.g., Namecoin)
  - File sharing (e.g., Filecoin)
  - General purpose (e.g., Ethereum)

# Blockchain: not there yet

- Early blockchain systems do not scale
- Bitcoin is limited to 7 transactions per second
- 100,000 transaction backlog in May 2017
- We require horizontal scalability for ubiquitous use
- More nodes = better performance and throughput

# Related work

Table: Summary of the scalability properties of many blockchain systems. Scalability gets better from left to right.

| **Not scalable** | **Somewhat scalable** | **Limited horizontal scalability** | **True horizontal scalability** |
|---|---|---|---|
| Bitcoin<br>Ethereum<br>etc. | Hyperledger<br>ByzCoin<br>Solidius | Elastico<br>OmniLedger<br>Lightning Network | CHECO (this work) |

# Related work

State-of-the-art: Sharding

- Split state into multiple shards
- Shards run consensus algorithm in parallel
- Difficult to perform atomic inter-shard transactions
  - Elastico: not possible
  - OmniLedger: via Atomic Commit protocol
- Diminishing return

# Research question

How can we design a *blockchain consensus protocol* that is *fault tolerant*, *scalable* and able to reach *global consensus?*

# Outline

# Intuition

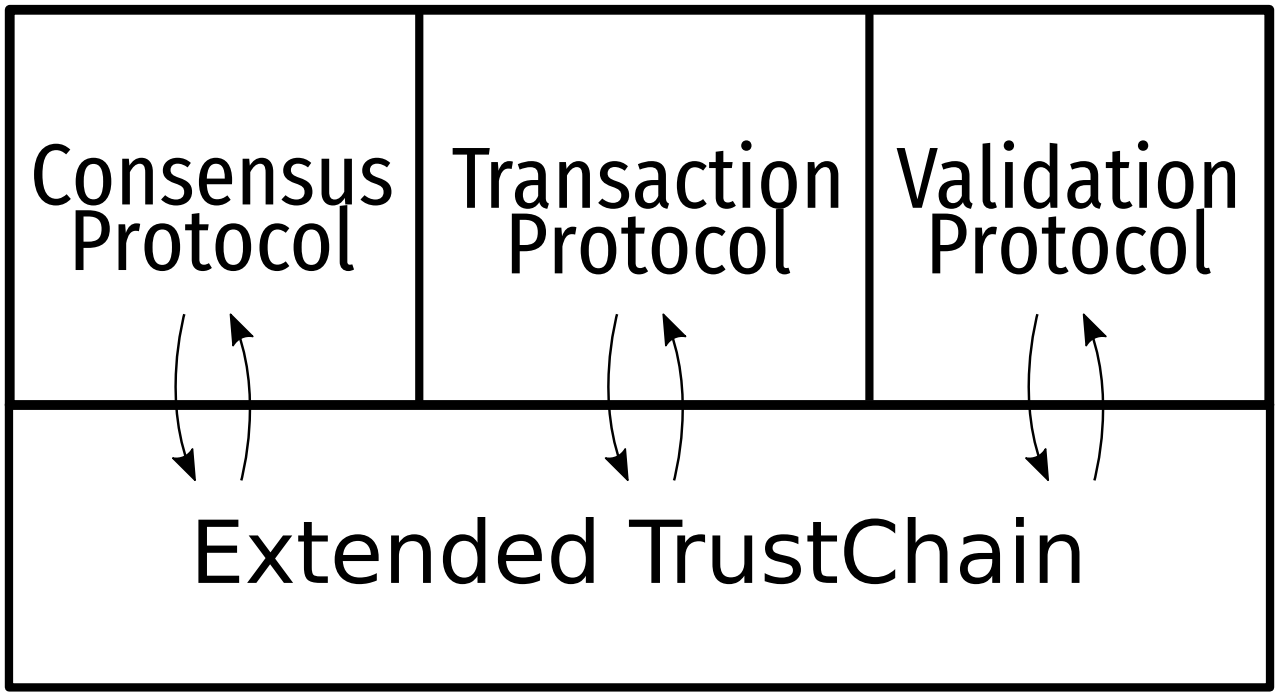
- Traditionally: to reach consensus and check the validity of all transactions is expensive
- Our idea: we decouple consensus and validation
- Use a single digest to represent an arbitrarily large number of transactions
- Reach consensus on the small digest
- Nodes then independently check the validity of the transactions of interest

# System model

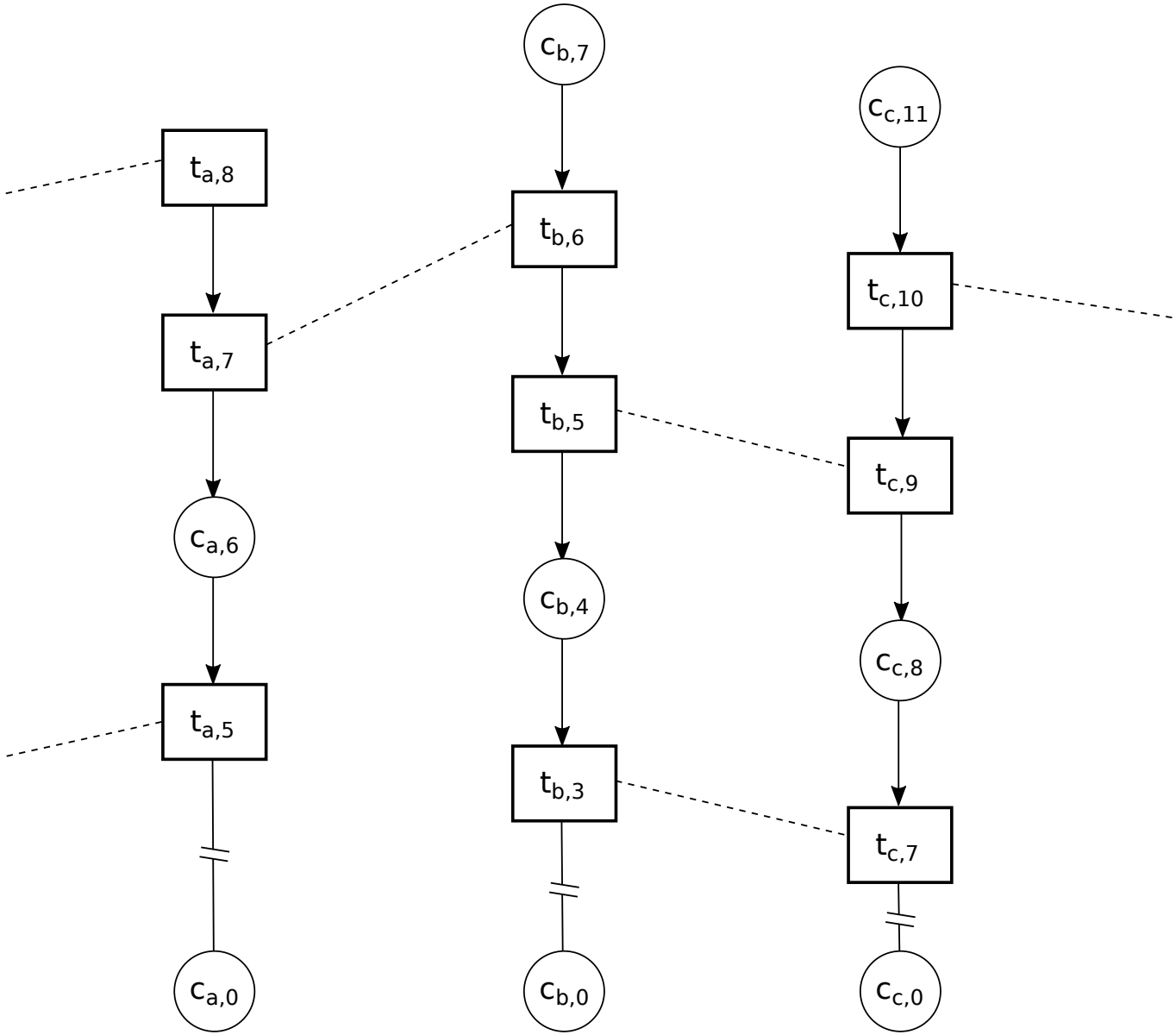
- $N$ is the population size
- Purely asynchronous channels with eventual delivery
- $n$ nodes are facilitators
- $t$ nodes are malicious, i.e. Byzantine
- $n \geq 3t + 1$
- $N \geq n + t$

# Architecture overview

The four components of CHECO



# Extended TrustChain



# Extended TrustChain: TX block

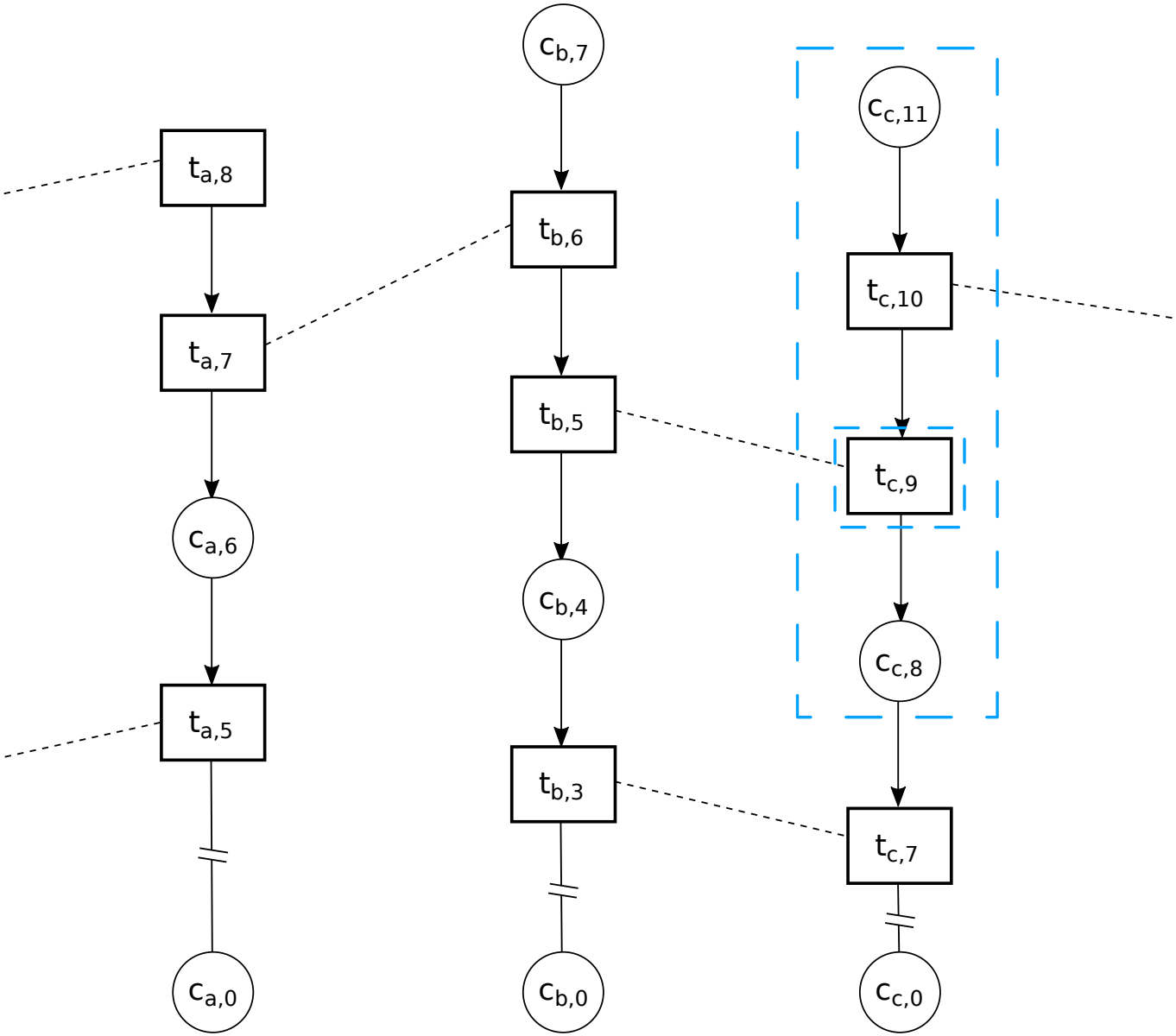
1. Hash pointer to the previous block
2. Sequence number
3. Transaction ID
4. Public key of the counterparty
5. Transaction message $m$
6. Signature the five items above

A transaction is represented by a *pair* of TX blocks

# Extended TrustChain: CP block

1. Hash pointer to the previous block
2. Sequence number
3. Digest of consensus result, i.e. a set of CP blocks
4. Round number $r$
5. Signature on the four items above

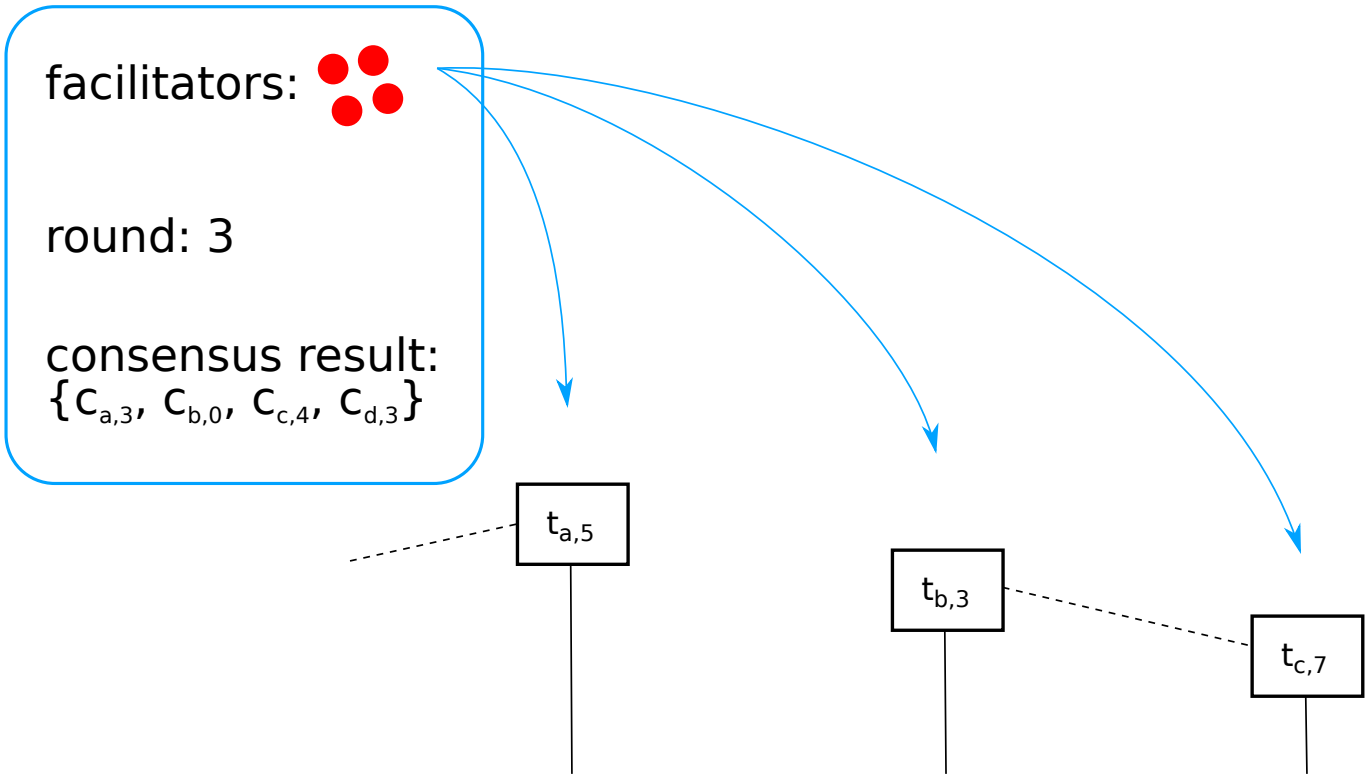
# Extended TrustChain: Fragment of a TX block



# Consensus protocol—Background on ACS

- Asynchronous common subset
- A simplification of HoneyBadgerBFT [4]
- $n$ nodes
- $t$ nodes may be malicious
- Input: every node proposes a set of values, e.g., $\{A, B\}, \{B, C\}, \ldots$
- Output: set union of the majority, e.g., $\{A, B, C, \ldots\}$

# Consensus protocol

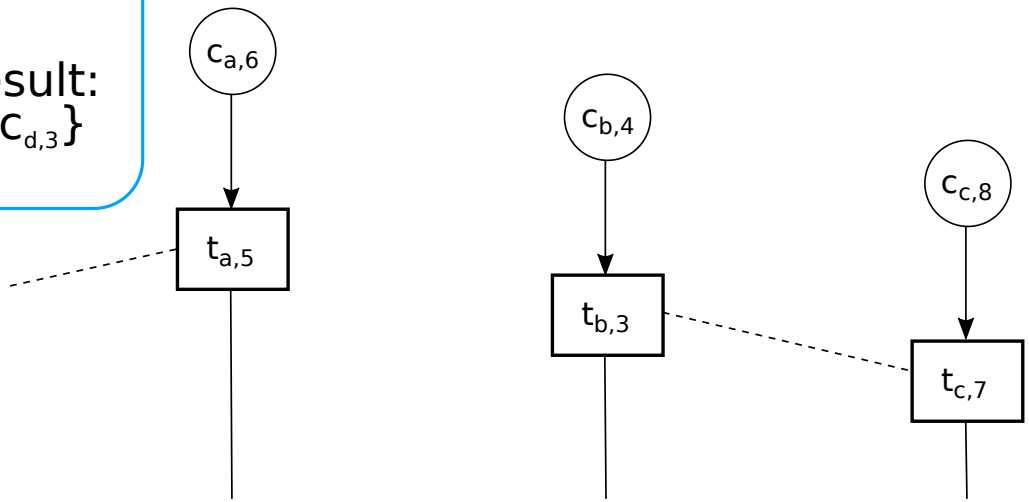


# Consensus protocol

facilitators: 

round: 3

consensus result:
$\{c_{a,3}, c_{b,0}, c_{c,4}, c_{d,3}\}$

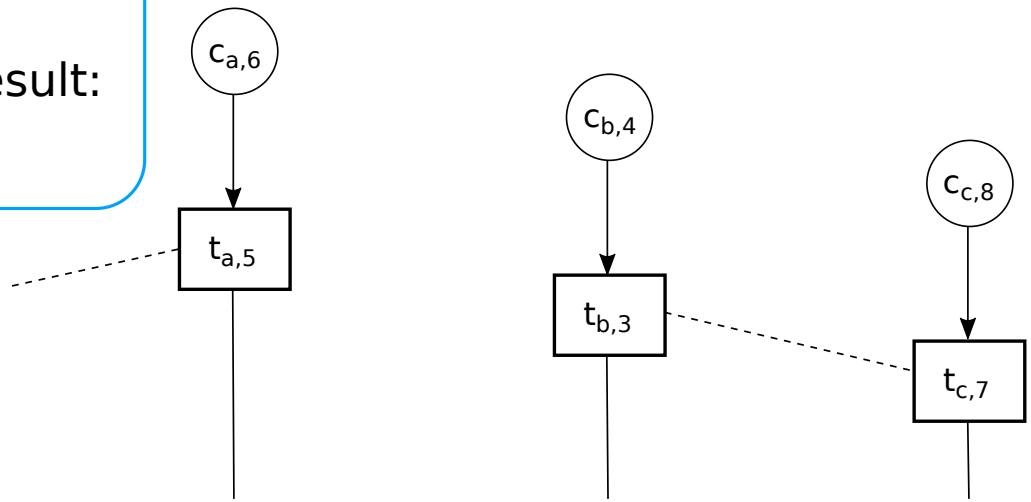


# Consensus protocol

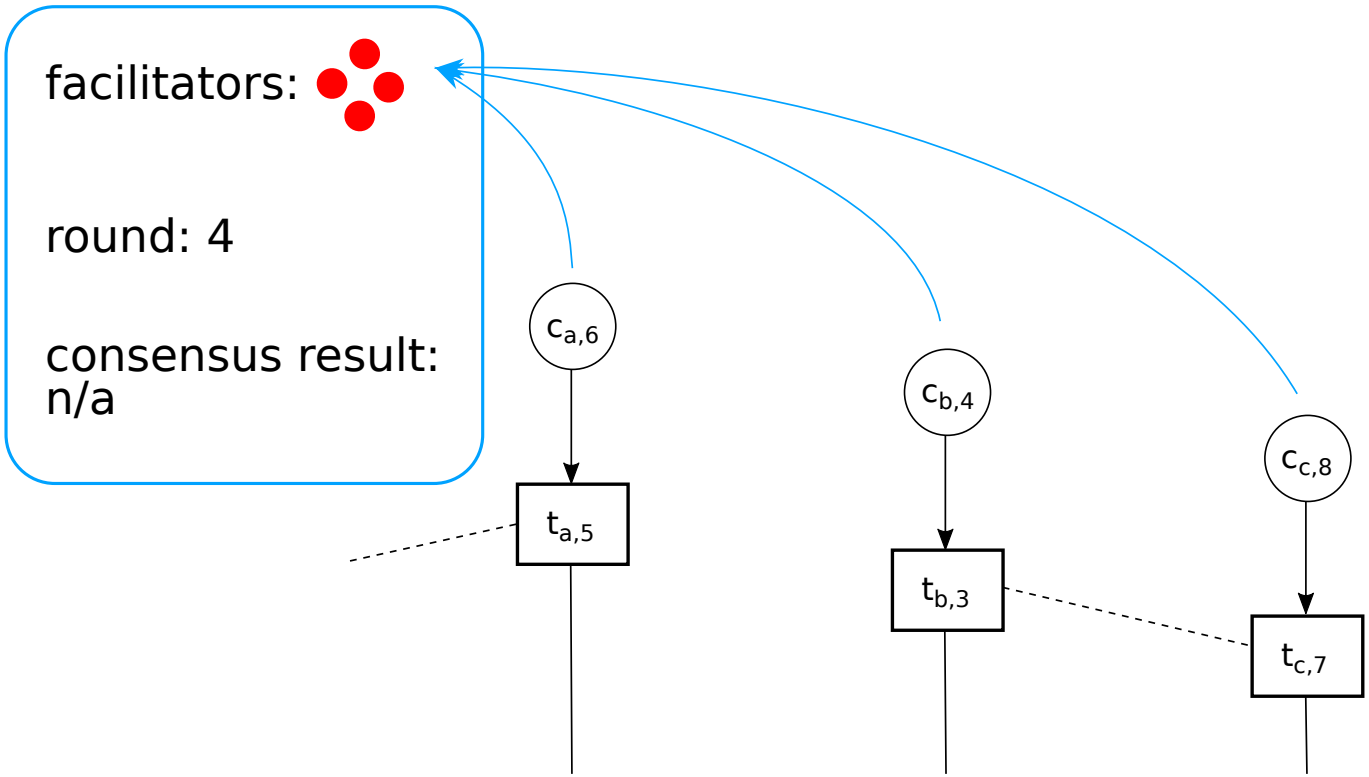
facilitators: lottery $\{c_{a,3}, c_{b,0}, c_{c,4}, c_{d,3}\}$

round: 4

consensus result: n/a



# Consensus protocol

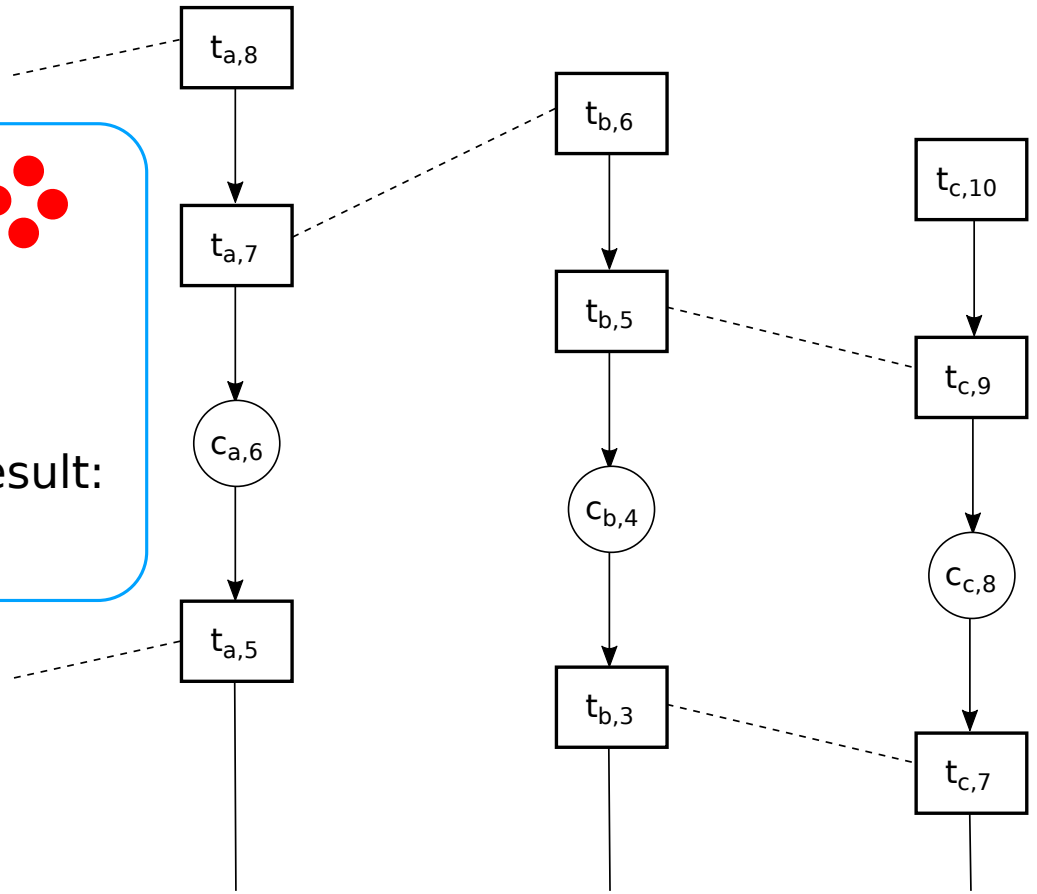


# Consensus protocol

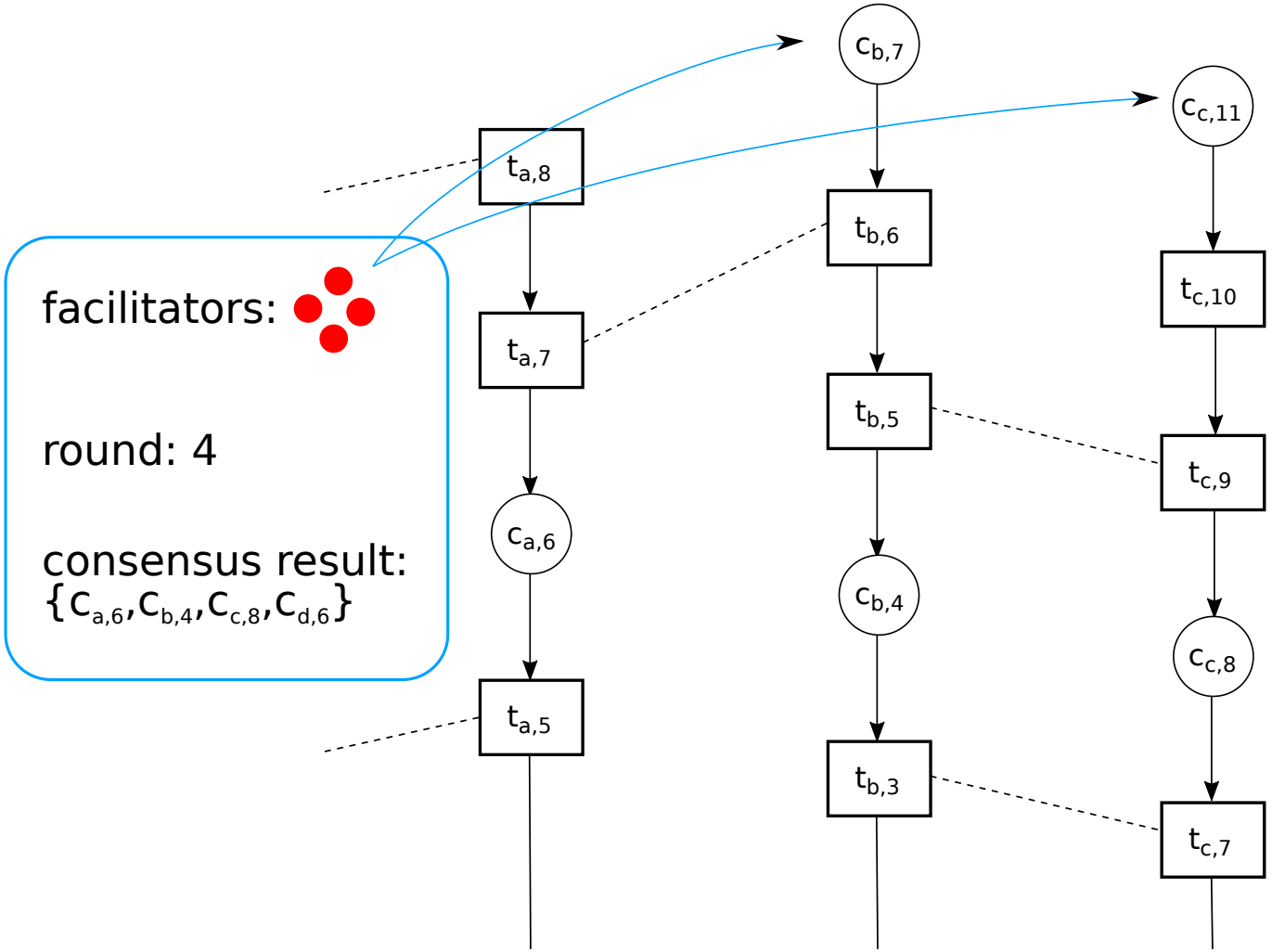
facilitators:

round: 4

consensus result: n/a



# Consensus protocol



# Consensus protocol: properties

The consensus protocol has the following properties in every round $r$.

1. *Agreement*: Every correct outputs the same set of facilitators.
2. *Validity*: The consensus results is valid such that a new set of facilitators can be computed from it.
3. *Fairness*: Every node with a CP block in the consensus result should have an equal probability of becoming a facilitator.
4. *Termination*: Every correct node eventually outputs a set of facilitators.

# Transaction protocol

- Request (`tx_req`) and response (`tx_resp`) protocol
- Two TX blocks containing the same *txid* are generated
- Non-blocking

# Validation protocol: overview

- Request (`vd_req`) and response (`vd_resp`) protocol
- Transactions are in three states—*valid*, *invalid* and *unknown*—defined by get_validity($\cdot$)
- The goal is to identify which state a given transaction is in satisfying the validation protocol properties

# Validity definition

**Function** get_validity($t_{u,i}, F$) validates the transaction $t_{u,i}$

Check that $v$ sent the correct $F$, otherwise return *unknown*.

$\langle \_, \_, txid, pk_v, m, \_ \rangle \leftarrow t_{u,i}$
**if** number of blocks of *txid* in $F \neq 1$ **then**
    **return** *invalid*

▷ TX exists

$\langle \_, \_, txid', pk'_u, m', \_ \rangle \leftarrow t_{v,j}$
**if** $m \neq m' \vee pk_u \neq pk'_u$ **then**
    **return** *invalid*

▷ no tampering

**return** *valid*

# Validation protocol

Send $\langle \texttt{vd\_req}, txid \rangle$ to $v$ to begin.

**Upon** $\langle \texttt{vd\_req}, txid \rangle$ from $v$
  $t_{u,i} \leftarrow$ the transaction identified by $txid$
  $F_{u,i} \leftarrow \text{agreed\_fragment}(t_{u,i})$
  send $\langle \texttt{vd\_resp}, txid, F_{u,i} \rangle$ to $v$
**Upon** $\langle \texttt{vd\_resp}, txid, F_{v,j} \rangle$ from $v$
  $t_{u,i} \leftarrow$ the transaction identified by $txid$
  set the validity of $t_{u,i}$ to $\text{get\_validity}(t_{u,i}, F_{v,j})$

# Validation protocol: properties

- *Agreement*: If any correct node decides on the validity of a transaction (except when it is *unknown*), then all other correct nodes are able to reach the same conclusion or *unknown*.
- *Correctness*: The validation protocol outputs the correct result according to the aforementioned validity definition.
- *Liveness*: Any valid (invalid) transaction is marked as validated (invalid) eventually.

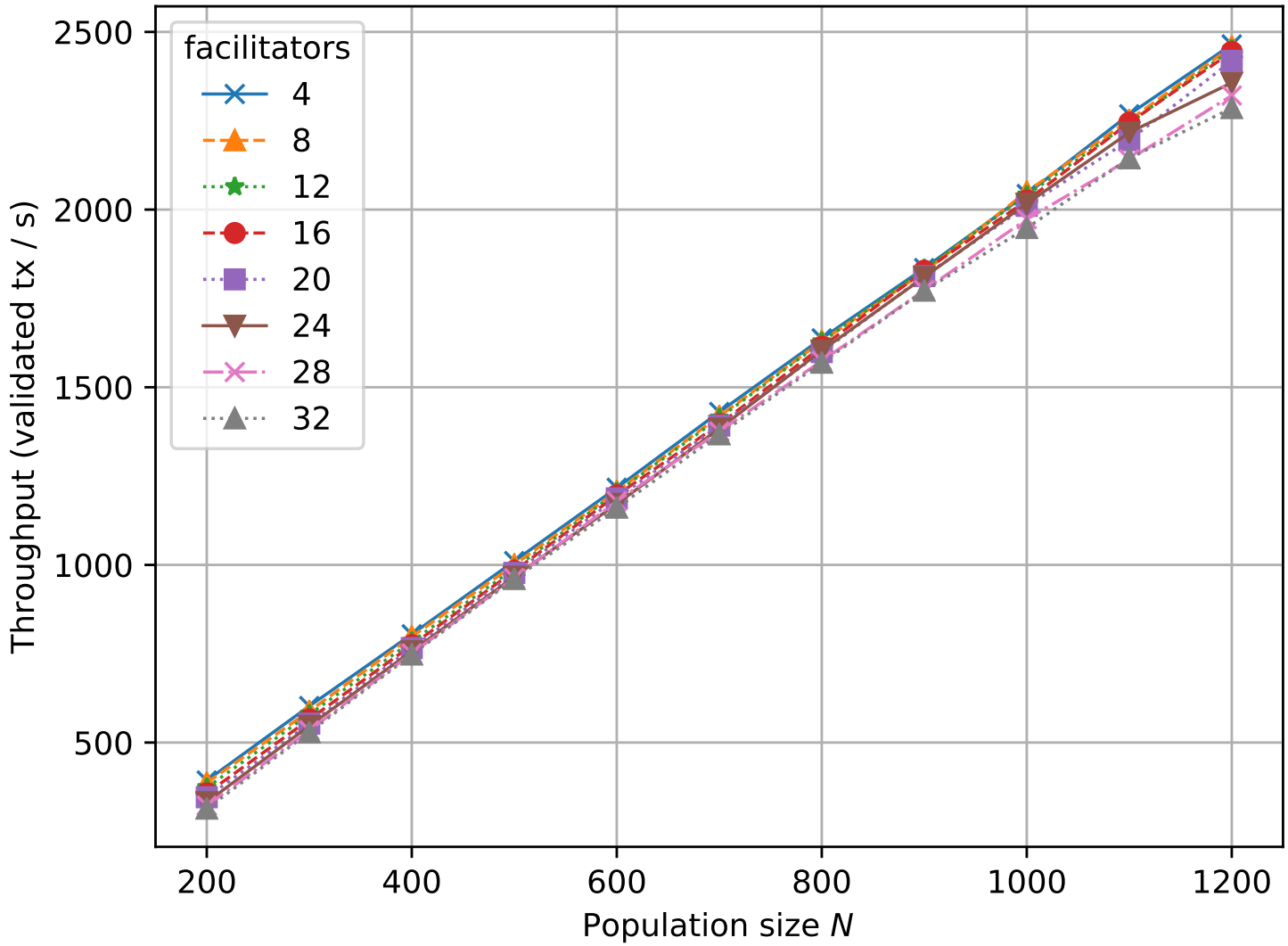
# Outline

# Implementation and experiment setup

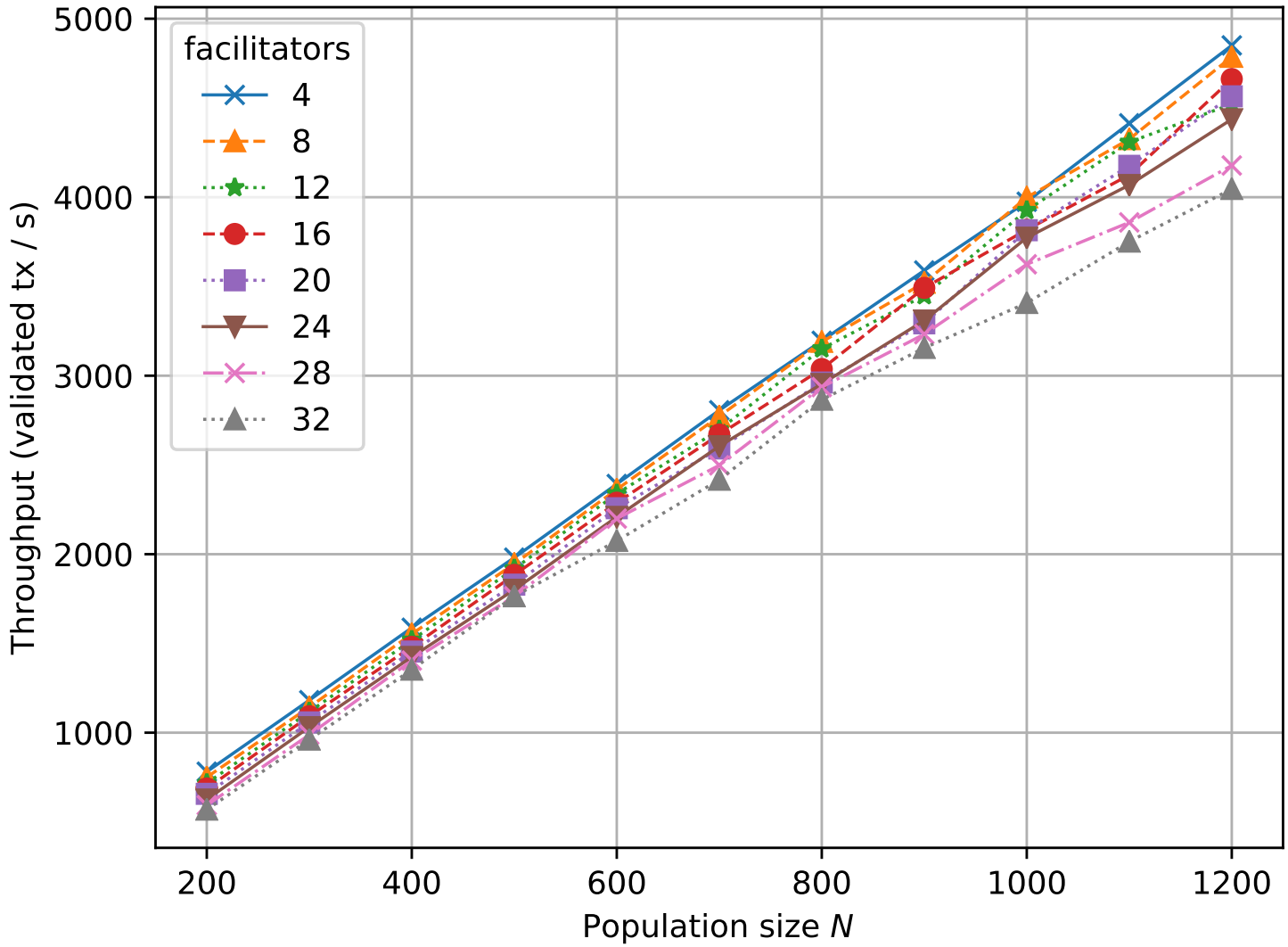
- Free and open source implementation on Github: `https://github.com/kc1212/checo`
- SHA256 for hash functions and Ed25519 for digital signature
- Experiment on the DAS-5[2]
  - Up to 30 machines
  - Each running 40 nodes

[2] `http://www.cs.vu.nl/das5/`

# Throughput vs population size (random neighbour)



# Throughput vs population size (fixed neighbour)



# Outline

# Conclusion

**Research question**

How can we design a *blockchain consensus protocol* that is *fault tolerant*, *scalable* and able to reach *global consensus?*

Our system achieve the following

- Fault tolerant up to $t$ nodes
- Horizontal scalability
- Global consensus on CP blocks

# Future work

- Improve fault tolerance
- Improve fork detection
- Analyse the system in the permissionless environment
- Concrete application

# Bibliography


- [Online]. Available: https://en.wikipedia.org/wiki/Death_of_Wei_Zexi.
- W. Youyou, M. Kosinski, and D. Stillwell, "Computer-based personality judgments are more accurate than those made by humans", *Proceedings of the National Academy of Sciences*, vol. 112, no. 4, pp. 1036–1040, 2015.
- [Online]. Available: https://www.theguardian.com/technology/2017/jul/31/facebook-dark-ads-can-swing-opinions-politics-research-shows.
- A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols", in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, pp. 31–42.

# Transaction protocol

Create $t_{u,h}$ and send $\langle \texttt{tx\_req}, t_{u,h} \rangle$ to start a transaction.

**Upon** $\langle \texttt{tx\_req}, t_{v,j} \rangle$ from $v$
  $\langle \_, \_, txid, pk_v, m, \_ \rangle \leftarrow t_{v,j}$ ▷ unpack $t_{v,j}$
  new_tx$(pk_u, m, txid)$ ▷ create and store $t_{u,i}$
  store $t_{v,j}$ as the pair of $t_{u,h}$
  send $\langle \texttt{tx\_resp}, t_{u,h} \rangle$ to $v$
**Upon** $\langle \texttt{tx\_resp}, t_{v,j} \rangle$ from $v$
  $\langle \_, \_, txid, pk_v, m, \_ \rangle \leftarrow t_{v,j}$ ▷ unpack $t_{v,j}$
  store $t_{v,j}$ as the pair of the TX with identifier $txid$

# Consensus protocol overview

1. Runs in rounds
2. In round $r$, $n$ out of $N$ act as facilitators
3. The facilitators collect CP blocks from all nodes
4. Facilitators run a Byzantine consensus algorithm to agree on a set of CP blocks
5. Disseminate the consensus result $\mathcal{C}_r$, i.e. the CP blocks
6. From $\mathcal{C}_r$, new facilitators are computed
7. Repeat