

# A Blockchain Consensus Protocol with Horizontal Scalability

Kelong Cong  
kelong.cong@epfl.ch  
EPFL

Zhijie Ren  
z.ren@tudelft.nl  
TU Delft

Johan Pouwelse  
peer2peer@gmail.com  
TU Delft

IFIP Networking, Zurich 2018

# Outline

## Introduction

- The dangers of centralisation

- Related work

- Research question

## System architecture

- System model

- Architecture overview

- Extended TrustChain

- Consensus protocol

- Transaction protocol

- Validation protocol

## Experimental results

## Conclusion

# Outline

## Introduction

- The dangers of centralisation

- Related work

- Research question

## System architecture

- System model

- Architecture overview

- Extended TrustChain

- Consensus protocol

- Transaction protocol

- Validation protocol

## Experimental results

## Conclusion

# The dangers of centralisation

- ▶ Technological advancements give us convenience
- ▶ But it puts central authorities in control
- ▶ Many are motivated by profit or the goals of the local government
- ▶ The interest of the authorities do not align with the users

# Blockchain: a new hope?

- ▶ Blockchains are distributed (replicated) ledgers with no central control
- ▶ They enable internet-scale consensus for the first time
- ▶ Some initial applications include:
  - ▶ Digital cash (e.g., Bitcoin, Litecoin)
  - ▶ Domain name system (e.g., Namecoin)
  - ▶ Storage rental (e.g., Filecoin)
  - ▶ General purpose (e.g., Ethereum)

# Blockchain: not there yet

- ▶ All blockchain systems have a consensus algorithm
- ▶ Early consensus algorithms 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 users → more transactions per second globally

## Related work

- ▶ Off-chain solution
  - ▶ Lightning Network<sup>1</sup>
  - ▶ Perun<sup>2</sup>
  - ▶ Easy to deploy but application specific
- ▶ On-chain solution
  - ▶ Parameter tuning
  - ▶ BFT consensus (e.g. Tendermint<sup>3</sup>, ByzCoin<sup>4</sup>)
  - ▶ Sharding (e.g. Elastico<sup>5</sup>, OmniLedger<sup>6</sup>)

---

<sup>1</sup><https://lightning.network/>

<sup>2</sup><https://perun.network>

<sup>3</sup><https://www.tendermint.com/>

<sup>4</sup>KJGKGF, USENIXSecurity16

<sup>5</sup>LNZBGS, CCS16

<sup>6</sup>KJGGSF, S&P18

## Related work

State-of-the-art—Sharding:

- ▶ Split state into multiple shards
- ▶ Shards run consensus algorithm in parallel

Challenges:

- ▶ Choosing and evolving the shard size
- ▶ Perform atomic inter-shard transactions
- ▶ Parameter choice highly depends on the application



## Research question

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

- ▶ Blockchain consensus protocol—application neutral
- ▶ Fault-tolerant—tolerate a number of malicious nodes
- ▶ Horizontal scalability—more nodes in the network leads to higher transaction throughput
- ▶ Global consensus—all node should agree on a global state

# Outline

## Introduction

The dangers of centralisation

Related work

Research question

## System architecture

System model

Architecture overview

Extended TrustChain

Consensus protocol

Transaction protocol

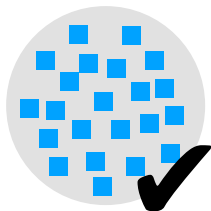
Validation protocol

## Experimental results

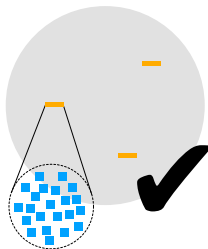
## Conclusion

# Intuition and idea explored in this thesis

- ▶ It is expensive to verify and reach consensus on all transactions
- ▶ Our idea: we decouple consensus and validation
- ▶ A single digest represents an arbitrarily large number of transactions
- ▶ Reach consensus on the small digest
- ▶ Nodes then independently check the validity of the transactions of interest



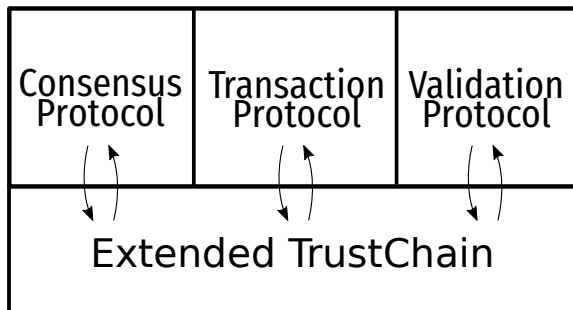
Early blockchains



Our idea

# Architecture overview

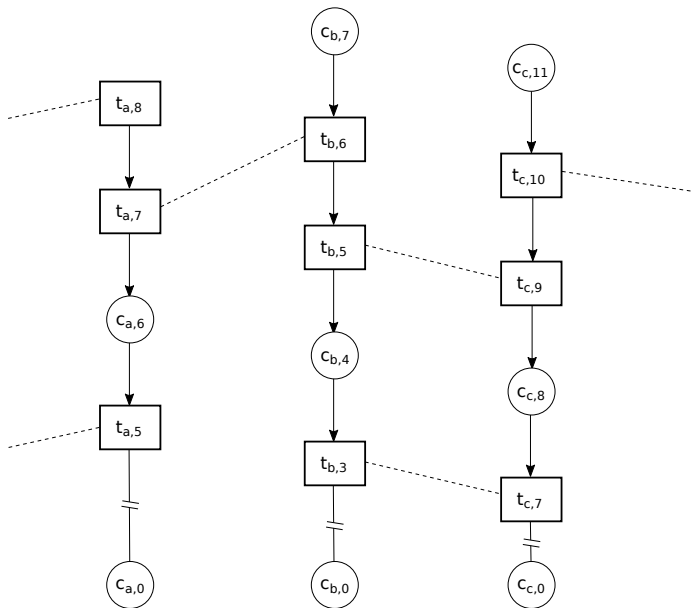
The four components of CHECO<sup>7</sup>



---

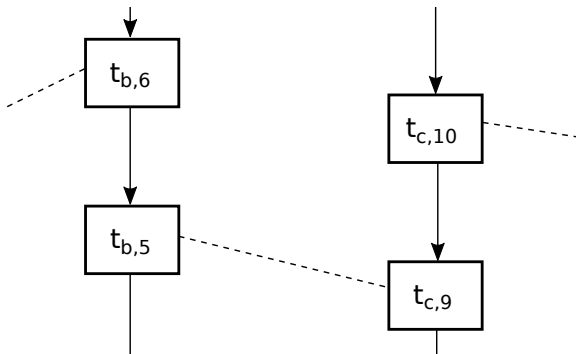
<sup>7</sup>CHECKpoint Consensus

# Extended TrustChain



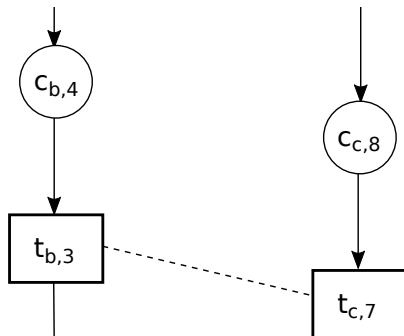
## Extended TrustChain: Transaction (TX) block

- ▶ Goal: record transactions
- ▶ A transaction is an agreement on a piece of data, i.e. digitally signed by both parties
- ▶ It is represented by a pair of TX blocks

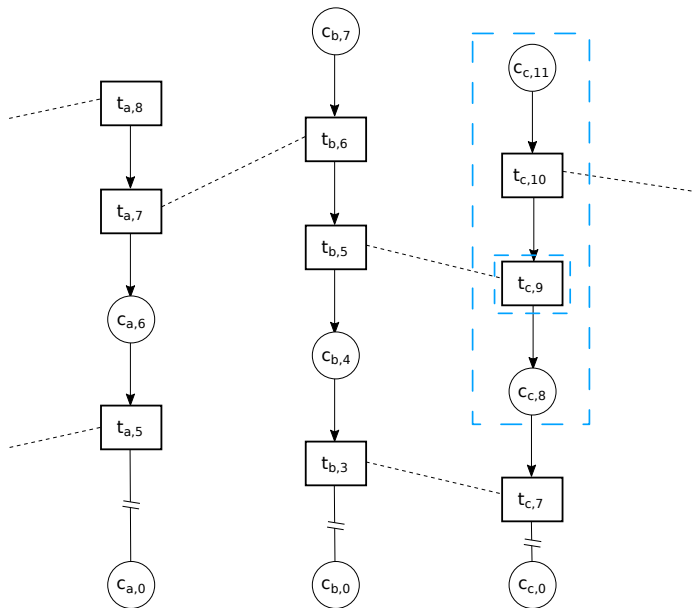


## Extended TrustChain: Checkpoint (CP) block

- ▶ Goal: represent the state of the chain using a single digest
- ▶ A collection of CP blocks from all the nodes represent the state of the system
- ▶ Nodes become aware of the system state by running our consensus protocol



# Extended TrustChain: Fragment of a TX block





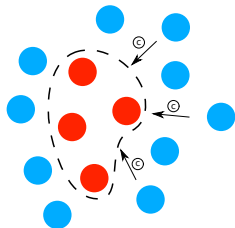
# Consensus protocol

- ▶ Goal 1: reach consensus on a collection of CP blocks amongst all the nodes
- ▶ Goal 2: create new CP blocks at the end of the protocol
- ▶ Uses an existing fault-tolerant consensus algorithm (HoneyBadgerBFT<sup>8</sup>) as the building block
- ▶ But it cannot be used in a large network due to high communication complexity
- ▶ We overcome this limitation by selecting a small number of *facilitators* from the network to run HoneyBadgerBFT

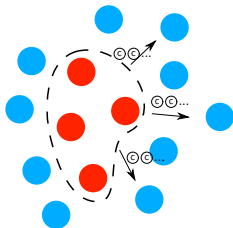
---

<sup>8</sup>MXCSS, CCS16

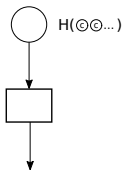
# Consensus protocol



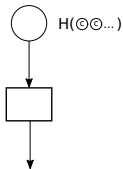
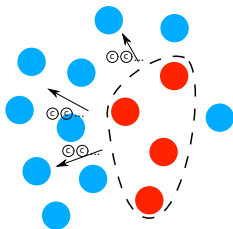
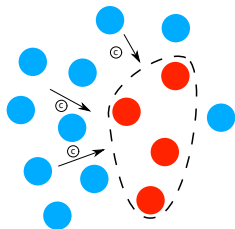
collect CP blocks



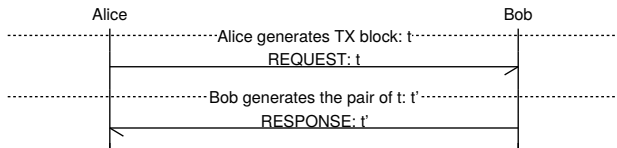
disseminate result



create new CP block  
elect new facilitators

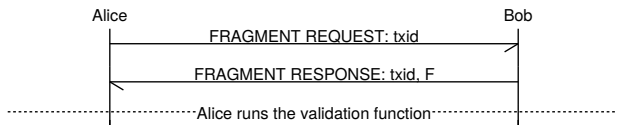


# Transaction protocol



- ▶ Two TX blocks are generated on the chains of Alice and Bob
- ▶ No guarantee that nodes follow this protocol
- ▶ Contrary to early blockchain systems, we do not broadcast transactions

# Validation protocol



- ▶ To check that the transaction protocol is correctly followed
- ▶ Alice needs the fragment of the TX on Bob's hash chain
- ▶ Validation function checks whether the fragment is OK and contain the transaction
- ▶ Can be generalised—any node may run the validation protocol on any transaction (does not need to be their own)

# Validation protocol: properties

*Consensus on CP blocks → consensus on transactions*

- ▶ CP blocks of the fragments are “anchored” due to the consensus protocol
- ▶ It is difficult to modify the fragment once “anchored”
- ▶ Since the transaction protocol and the validation protocol only use point-to-point communication, we achieve horizontal scalability.

# Outline

## Introduction

- The dangers of centralisation

- Related work

- Research question

## System architecture

- System model

- Architecture overview

- Extended TrustChain

- Consensus protocol

- Transaction protocol

- Validation protocol

## Experimental results

## Conclusion

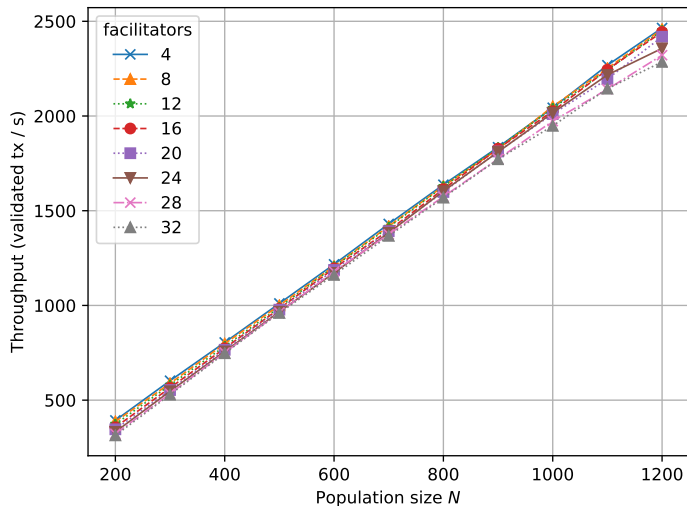
# Implementation and experiment setup

- ▶ 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<sup>9</sup>
- ▶ Up to 1200 nodes
- ▶ A third of the facilitators are malicious

---

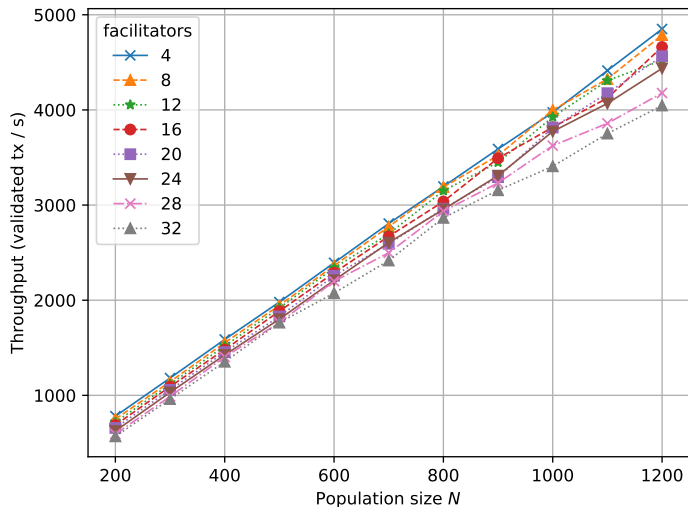
<sup>9</sup><http://www.cs.vu.nl/das5/>

# Validated transaction throughput (random node)





# Validated transaction throughput (fixed neighbour)



# Outline

## Introduction

The dangers of centralisation

Related work

Research question

## System architecture

System model

Architecture overview

Extended TrustChain

Consensus protocol

Transaction protocol

Validation protocol

## Experimental results

## Conclusion

# Conclusion

Our contribution:

- ▶ We design a blockchain consensus protocol—CHECO with horizontal scalability
- ▶ We formally analyse CHECO to ensure correctness
- ▶ Our experimental evaluation show strong evidence of our claim.

Future perspective:

- ▶ Improve fault-tolerance using ByzCoinX from OmniLedger
- ▶ Design and evaluate UTXO-style transactions (new paper to appear in Crypto Valley Conference)
- ▶ Assume a permissionless model

# 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

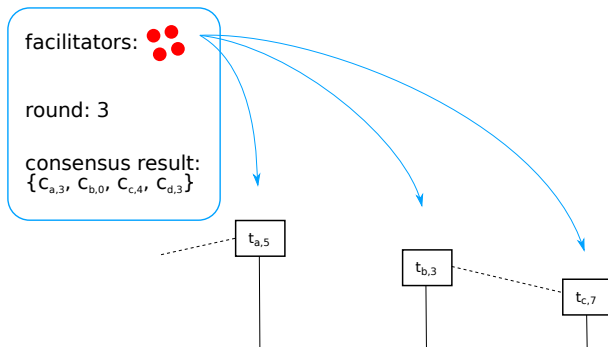
# CP block

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

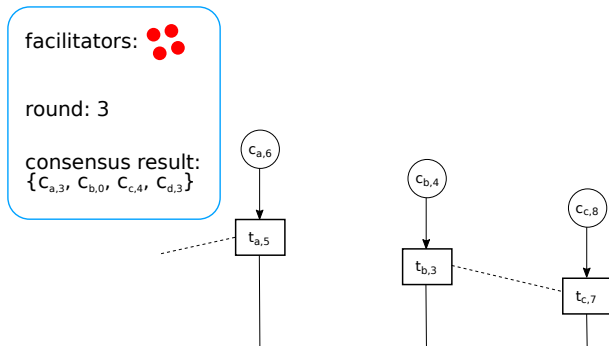
# Background on ACS

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

# Consensus protocol: part 1

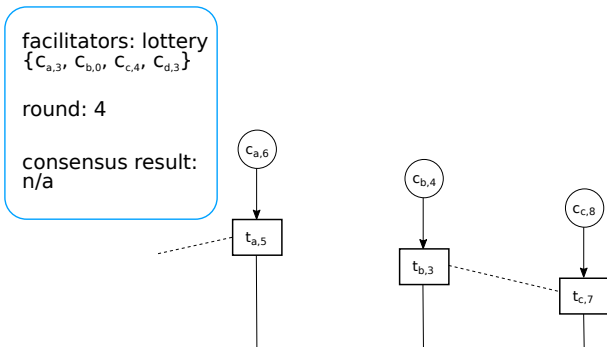


## Consensus protocol: part 2

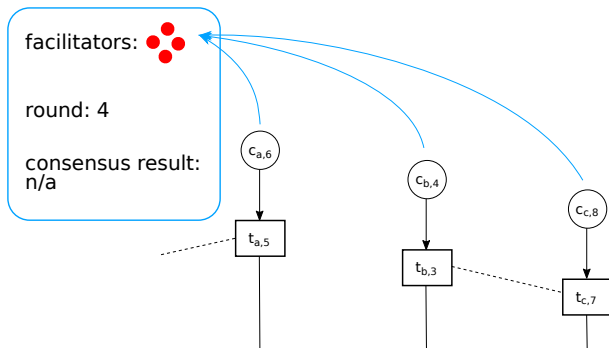




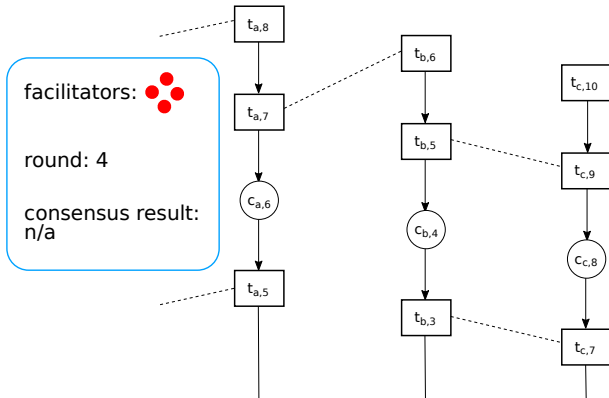
## Consensus protocol: part 3



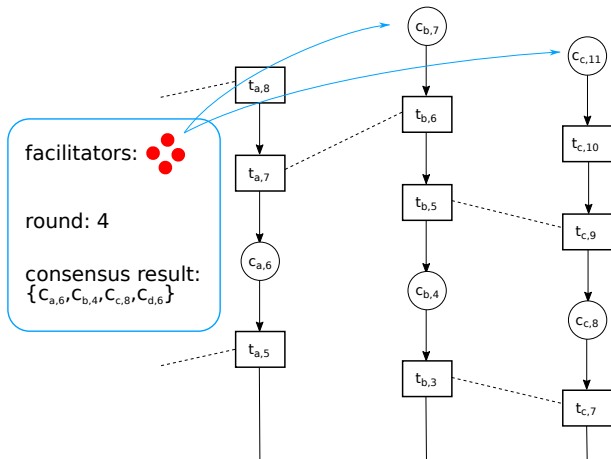
## Consensus protocol: part 4



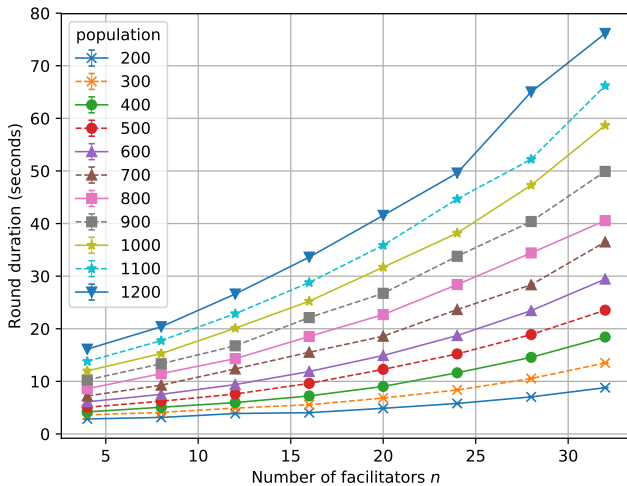
## Consensus protocol: part 5



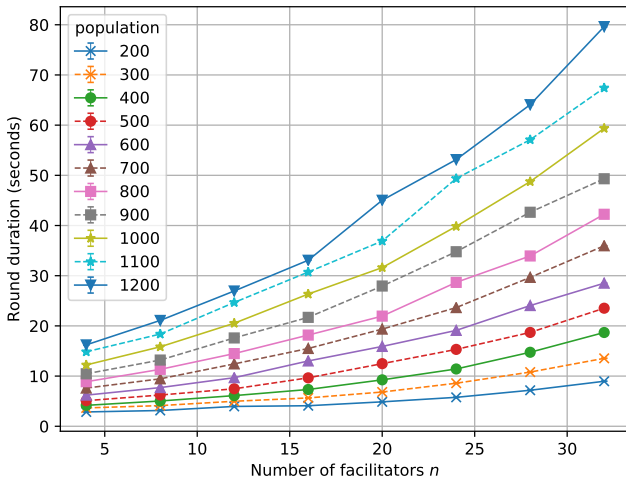
## Consensus protocol: part 6



# Effect of the number of facilitators (fixed neighbours)



# Effect of the number of facilitators (random neighbours)



## Stress test (fixed neighbour)

