

# 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, 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
- ▶ Most are motivated exclusively by profit
- ▶ Not always in the interest of the “users”<sup>1</sup>

---

<sup>1</sup>Typically users of some free service X are, in fact, used by X. 

# 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)

2018-04-23

# A Blockchain Consensus Protocol With Horizontal Scalability

## └ Introduction

### └ The dangers of centralisation

### └ Blockchain: a new hope?

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)

- Explain blockchain systems—consensus

# Blockchain: not there yet

- ▶ All blockchain systems have a consensus algorithm
- ▶ Early consensus algorithms (PoW) 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
  - ▶ Perun
- ▶ On-chain solution
  - ▶ Parameter tuning
  - ▶ BFT consensus (e.g. Tendermint, ByzCoin)
  - ▶ Sharding (e.g. Elastico, OmniLedger)



## 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, e.g., PoW
- ▶ 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

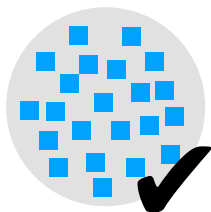
Validation protocol

## Experimental results

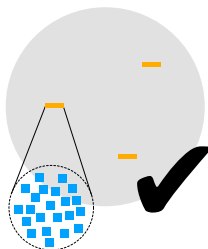
## Conclusion

# 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



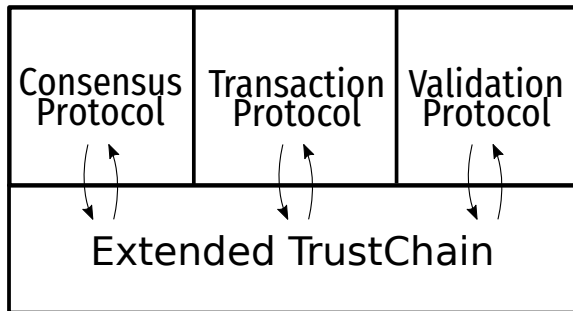
Early blockchains



Our idea

# Architecture overview

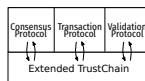
The four components of CHECO



# A Blockchain Consensus Protocol With Horizontal Scalability

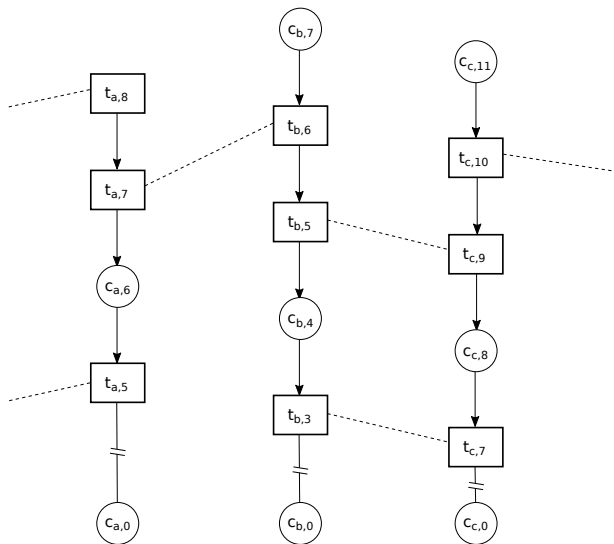
- └ System architecture
  - └ Architecture overview
    - └ Architecture overview

The four components of CHECO



- The primary data structure is the Extended TrustChain, extension of our prior work
- The three protocols the tasks as their name suggests
- They are independent and run concurrently
- The only synchronisation happens via the Extended TrustChain
- But in no part of those protocol do we lock the Extended TrustChain

# Extended TrustChain



2018-04-23

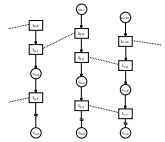
# A Blockchain Consensus Protocol With Horizontal Scalability

System architecture

Extended TrustChain

Extended TrustChain

Extended TrustChain

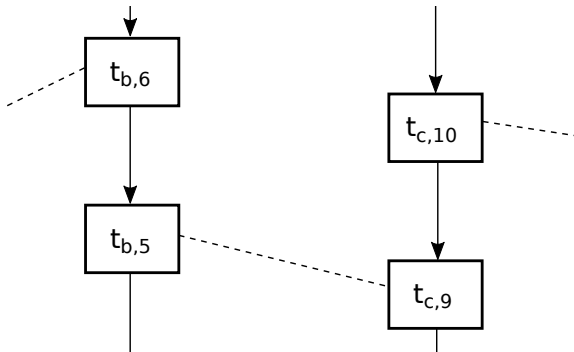


- In this example there are three nodes
- Each node maintains their personal hash chain and genesis block
- Squares are TX blocks and circles are CP blocks
- Explain the block content in caption
- The dotted line represent pairs of TX blocks
- Geared with the understanding of our data structure, we are ready to talk about the consensus protocol



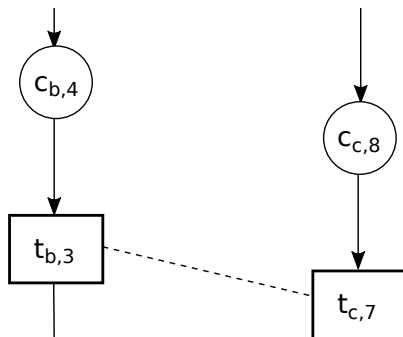
# Extended TrustChain: Transaction (TX) block

- ▶ Goal: record transactions
- ▶ A transaction is represented by a pair of TX blocks, i.e. a contract signed by both parties

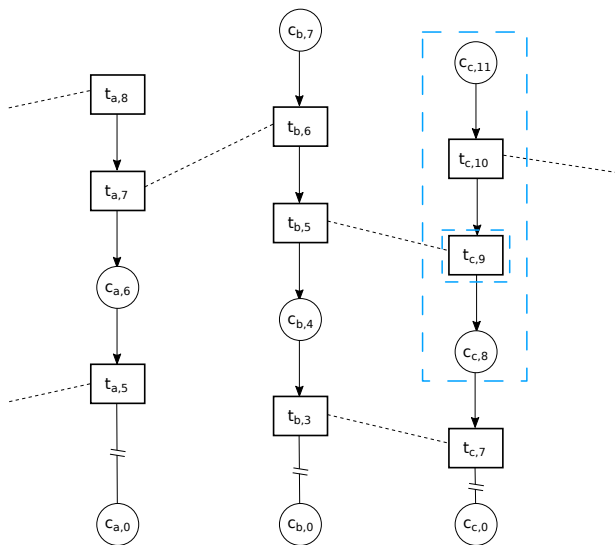


## Extended TrustChain: Checkpoint (CP) block

- ▶ 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



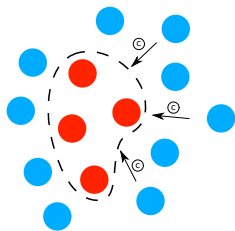
# Extended TrustChain: Fragment of a TX block



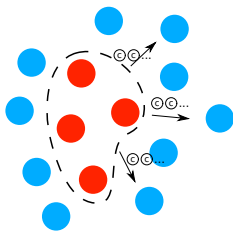
# Consensus protocol

- ▶ 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 [1]) 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

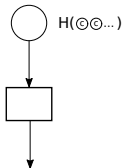
# Consensus protocol



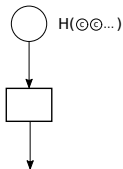
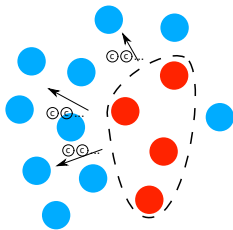
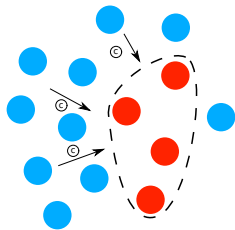
collect CP blocks



disseminate result



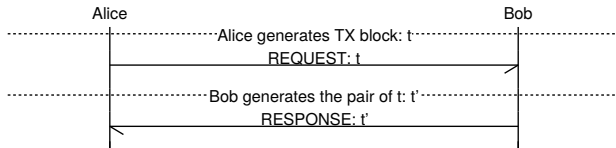
create new CP block  
elect new facilitators



# Consensus protocol: properties

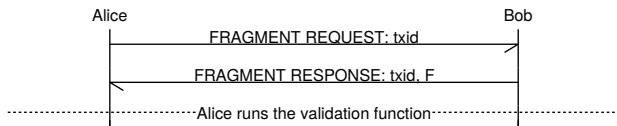
- ▶ *Agreement*: Every correct outputs the same set of facilitators.
- ▶ *Validity*: The consensus result is valid such that a new set of facilitators can be computed from it.
- ▶ *Fairness*: Every node with a CP block in the consensus result should have an equal probability of becoming a facilitator.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

# Transaction protocol



- ▶ Two TX blocks are generated on the chains of Alice and Bob
- ▶ No guarantee that nodes follow this protocol

# 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

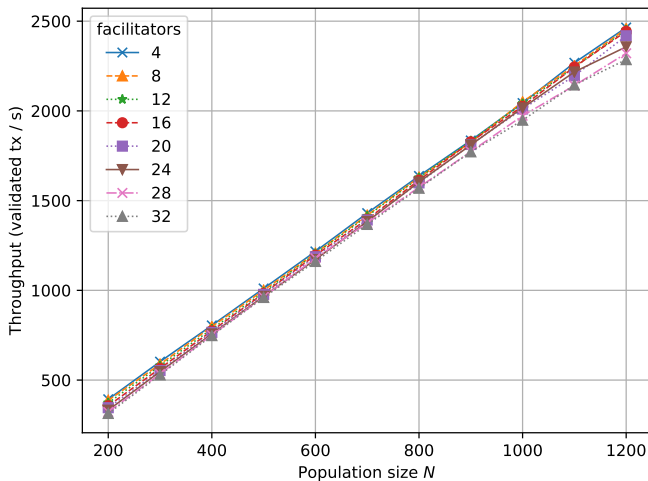
# Implementation and experiment setup

- ▶ 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>2</sup>
- ▶ Up to 1500 nodes

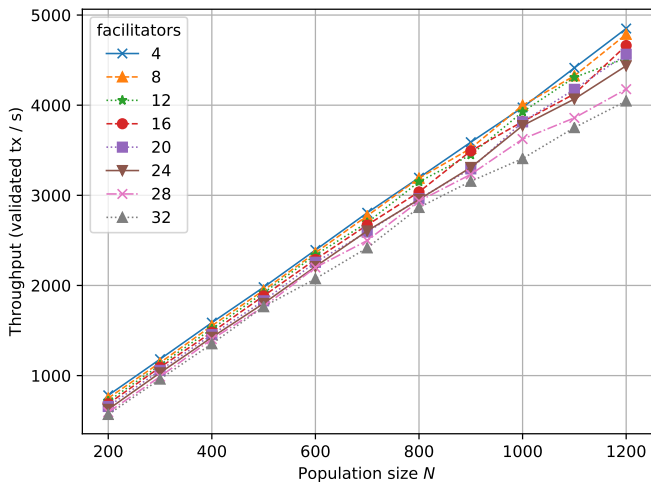
---

<sup>2</sup><http://www.cs.vu.nl/das5/>

# Validated transaction throughput (random node)



# Validated transaction throughput (fixed neighbour)



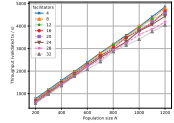
2018-04-23

# A Blockchain Consensus Protocol With Horizontal Scalability

└ Experimental results

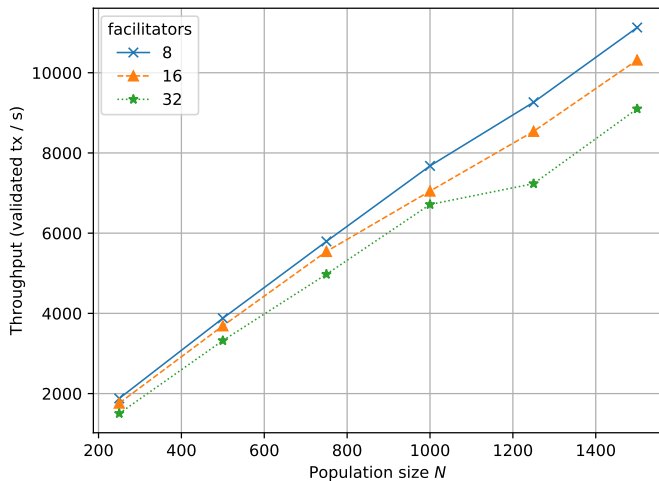
└ Validated transaction throughput (fixed

Validated transaction throughput (fixed neighbour)



No need to request for fragment every time a TX needs to be validated.  
Upon receiving a fragment, validate as many TX as possible.

# Stress test (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 work answers the research question.

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

- ▶ Fault-tolerance is achieved using HoneyBadgerBFT
- ▶ Horizontal-scalability is achieved by separating consensus and validation, demonstrated experimentally
- ▶ Global-consensus on transactions is achieved via consensus on CP blocks

# Bibliography

- [1] 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.

Thank you

Any questions?

# 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

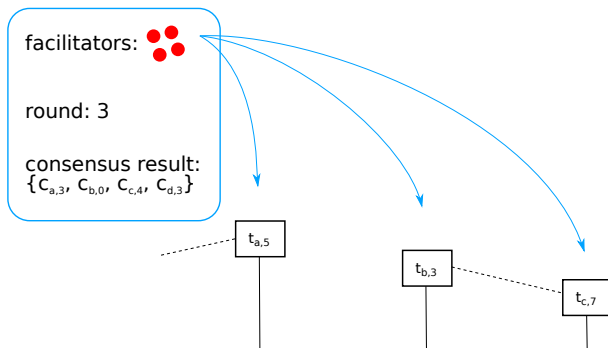
# CP block

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 [1]
- ▶  $n$  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

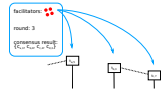


2018-04-23

# A Blockchain Consensus Protocol With Horizontal Scalability

- Extras
  - Consensus protocol
    - Consensus protocol: part 1

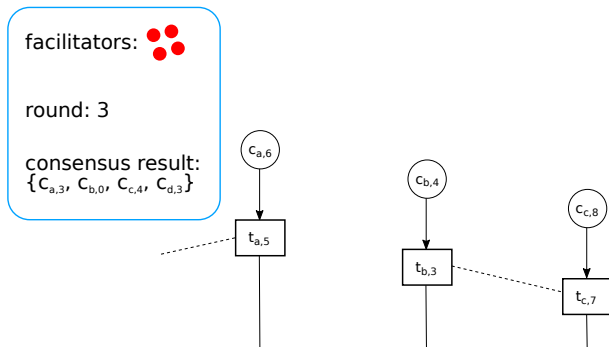
Consensus protocol: part 1



Suppose we are in a state where  $C_{r-1}$  has just been agreed by some facilitators but not yet propagated.



## Consensus protocol: part 2



2018-04-23

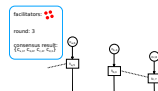
# A Blockchain Consensus Protocol With Horizontal Scalability

└ Extras

└ Consensus protocol

└ Consensus protocol: part 2

Consensus protocol: part 2



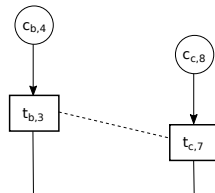
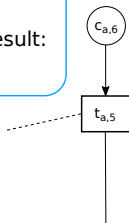
Nodes receive consensus result  $\mathcal{C}_{r-1}$ , first  $n$  nodes ordered by  $H(\mathcal{C}_{r-1} || pk)$  become  $\mathcal{F}_{r-1}$ , send the new CP blocks to  $\mathcal{F}_{r-1}$ .

# Consensus protocol: part 3

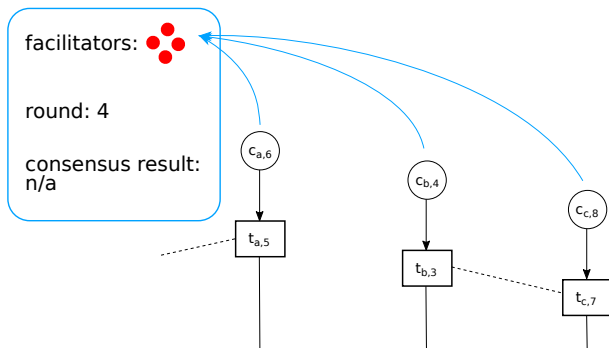
facilitators: lottery  
 $\{c_{a,3}, c_{b,0}, c_{c,4}, c_{d,3}\}$

round: 4

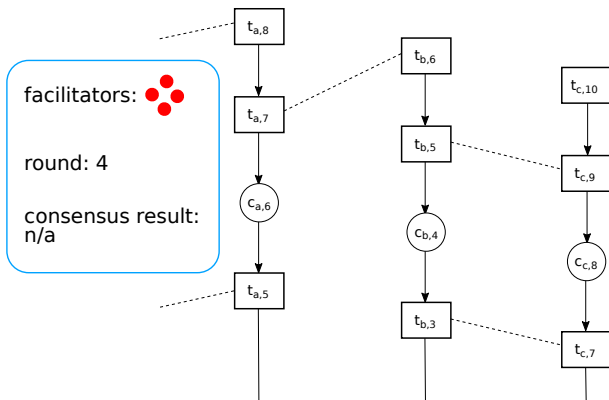
consensus result:  
n/a



# Consensus protocol: part 4



# Consensus protocol: part 5



2018-04-23

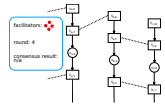
# A Blockchain Consensus Protocol With Horizontal Scalability

Extras

Consensus protocol

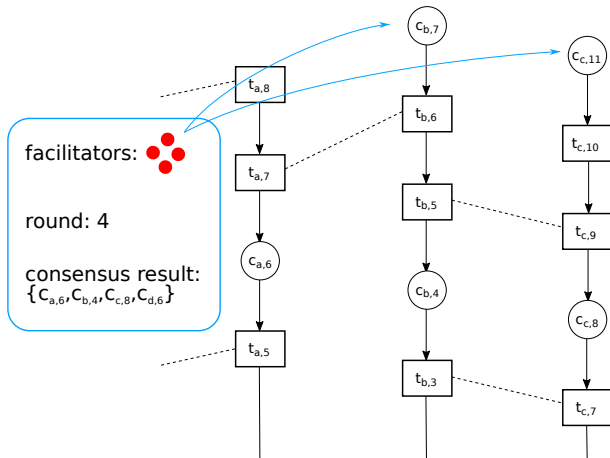
Consensus protocol: part 5

Consensus protocol: part 5



Transactions carry on as usual in round  $r$ , while facilitators are trying to reach consensus on the new CP blocks concurrently.

## Consensus protocol: part 6



2018-04-23

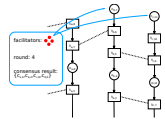
# A Blockchain Consensus Protocol With Horizontal Scalability

└ Extras

└ Consensus protocol

└ Consensus protocol: part 6

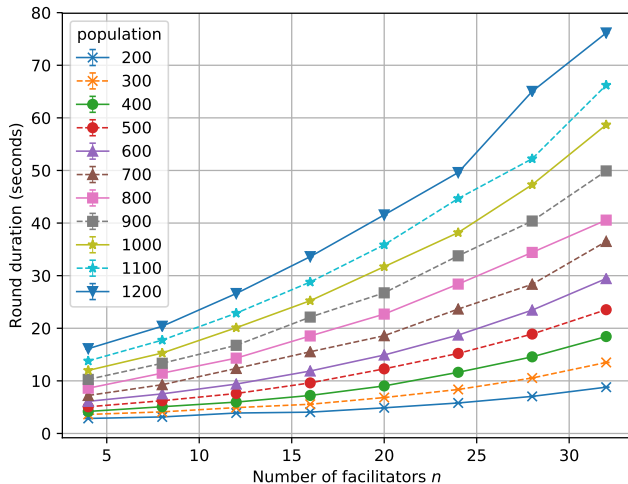
Consensus protocol: part 6



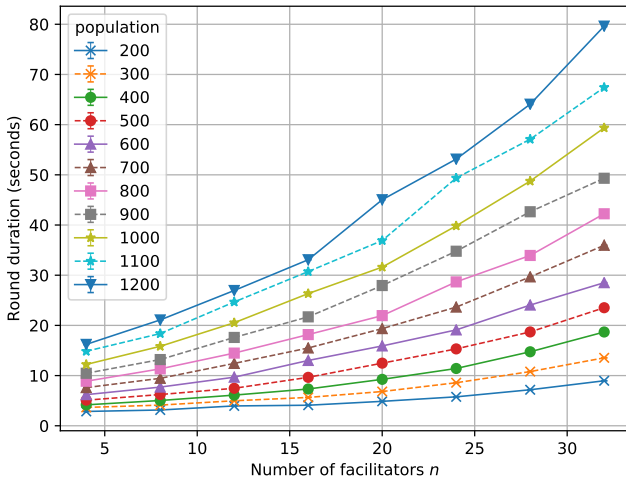
$\mathcal{F}_{r-1}$  agree and disseminate  $\mathcal{C}_r$ , CP blocks at round  $r - 1$  ( $c_{a,6}, c_{b,4}, c_{c,8}$ ) should be in  $\mathcal{C}_r$ .



# Effect of the number of facilitators (fixed neighbours)



# Effect of the number of facilitators (random neighbours)



# Future work

- ▶ Implement and experiment with a concrete application
- ▶ Analyse the system in the permissionless environment
- ▶ Improve fault tolerance