

A Blockchain Consensus Protocol With Horizontal Scalability

K. Cong

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

August 29, 2017

Outline

① Introduction

- The dangers of centralisation

- Related work

- Research question

② System architecture

- System model

- Architecture overview

- Extended TrustChain

- Consensus protocol

- Transaction protocol

- Validation protocol

③ Experimental results

④ Conclusion

Outline

1 Introduction

The dangers of centralisation

Related work

Research question

2 System architecture

System model

Architecture overview

Extended TrustChain

Consensus protocol

Transaction protocol

Validation protocol

3 Experimental results

4 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”¹

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

The dangers of centralisation: Examples

- Facebook can predict your opinions and desires from your “likes” [1], and then create a “psychographic” profile for use in political campaigns [2]
- Baidu’s promoted search result on medical care caused death of a student [3]

Blockchain: a new hope?

- Blockchains are distributed (replicated) ledgers with no central control²
- They enable internet-scale consensus for the first time
- Some initial applications include:
 - Digital cash (e.g., Bitcoin)
 - Domain name system (e.g., Namecoin)
 - Storage rental (e.g., Filecoin)
 - General purpose (e.g., Ethereum)

²In practice, some central control exist.

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

Table: Summary of the scalability properties of many blockchain systems.

Not scalable	Somewhat scalable	Limited horizontal scalability	True horizontal scalability
Bitcoin Litecoin	Hyperledger ByzCoin	Elastico OmniLedger	CHECO (this work)

scalability
→

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 up to some 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

① Introduction

The dangers of centralisation

Related work

Research question

② System architecture

System model

Architecture overview

Extended TrustChain

Consensus protocol

Transaction protocol

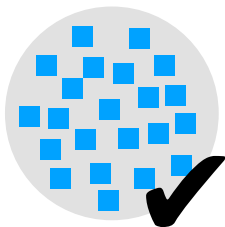
Validation protocol

③ Experimental results

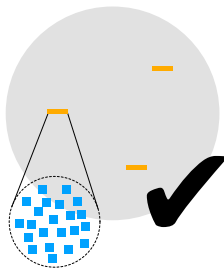
④ Conclusion

Intuition and idea explored in this thesis

- It is expensive to 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

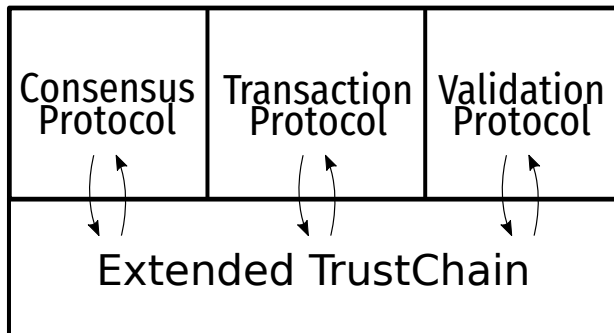


Early blockchains

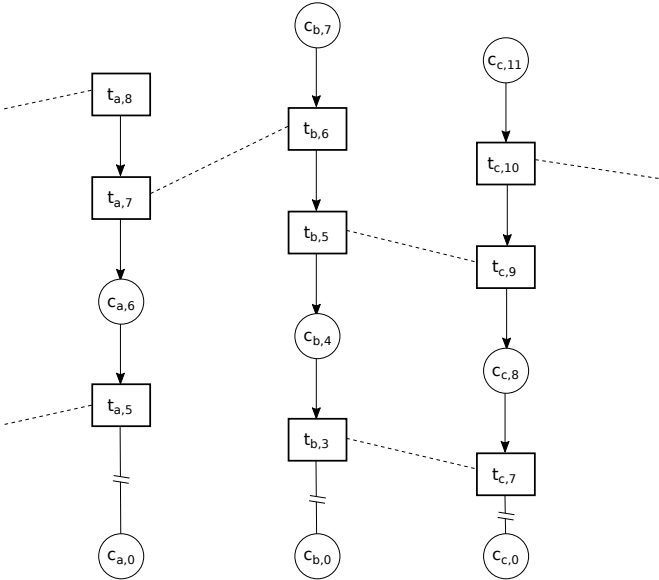


Our idea

The four components of CHECO

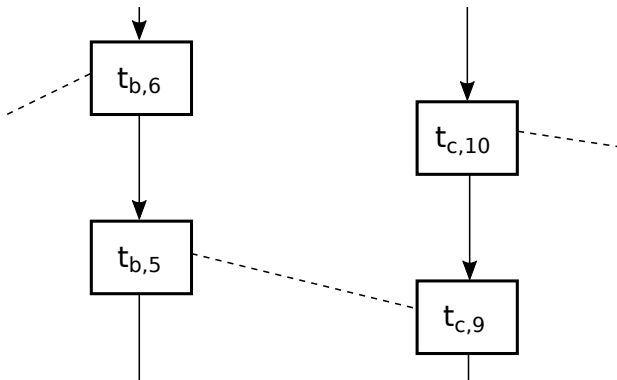


Extended TrustChain



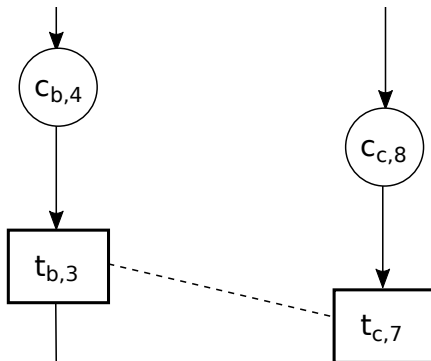
Extended TrustChain: Transaction (TX) block

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

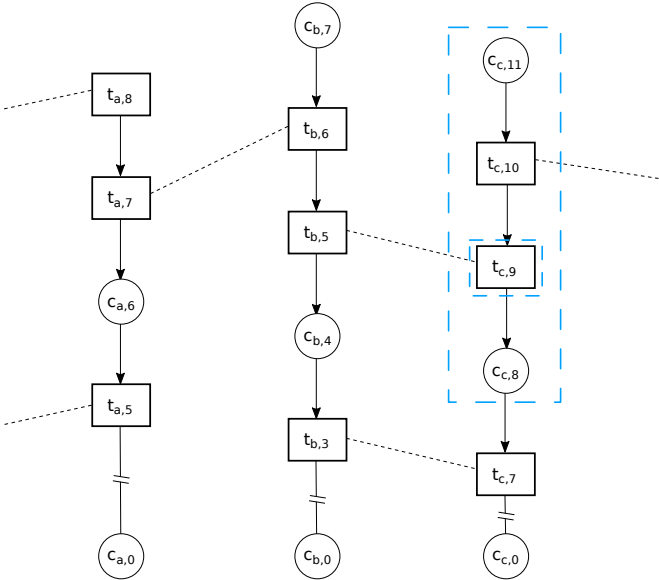


Extended TrustChain: Checkpoint (CP) block

- 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 agree on the system state using our consensus protocol



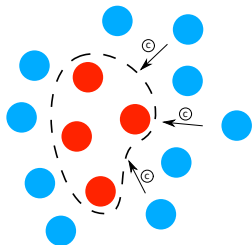
Extended TrustChain: Fragment of a TX block



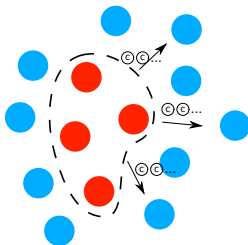
Consensus protocol

- Goal 1: reach consensus on a collection of CP blocks amongst all the nodes
- Goal 2: create CP blocks at the end of the protocol
- Uses an existing fault-tolerant consensus algorithm (HoneyBadgerBFT [4]) 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

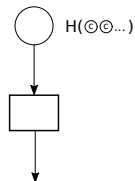
Consensus protocol



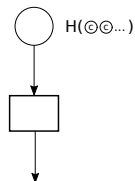
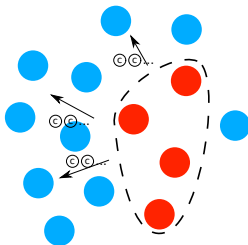
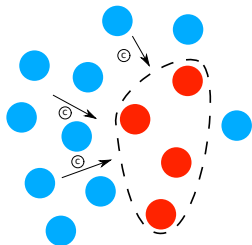
collect CP blocks



disseminate result



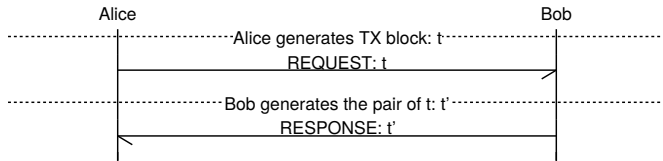
create new CP block



Consensus protocol: properties

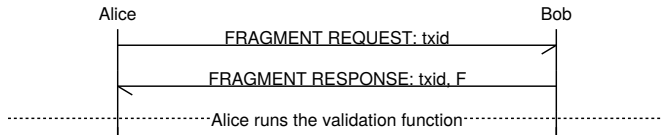
- *Agreement*: Every correct outputs the same set of facilitators.
- *Validity*: The consensus results is valid such that a new set of facilitators can be computed from it.
- *Fairness*: Every node with a CP block in the consensus result should have an equal probability of becoming a facilitator.
- *Termination*: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

- Result: we reach consensus on transactions via CP blocks
- 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

① Introduction

The dangers of centralisation

Related work

Research question

② System architecture

System model

Architecture overview

Extended TrustChain

Consensus protocol

Transaction protocol

Validation protocol

③ Experimental results

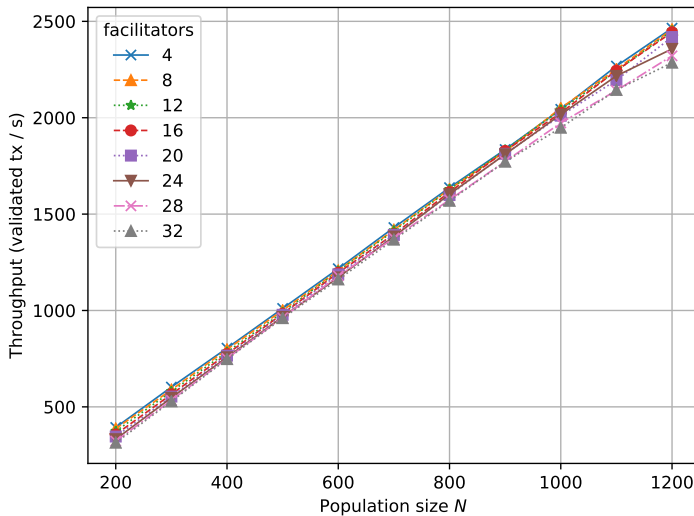
④ Conclusion

Implementation and experiment setup

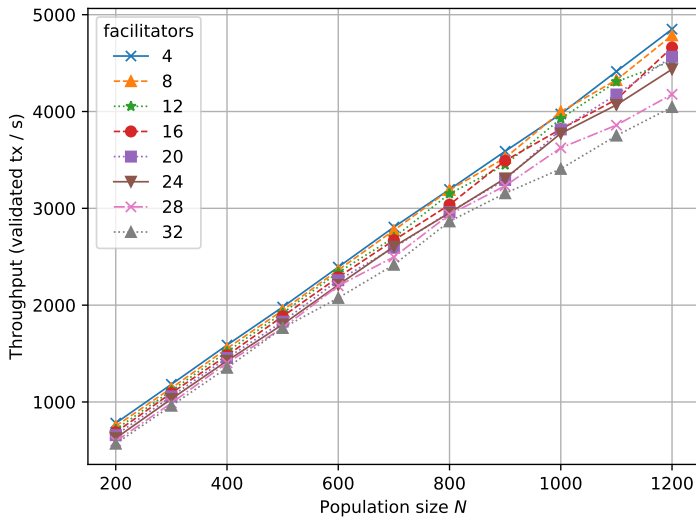
- Free and open source implementation on Github:
<https://github.com/kc1212/checo>
- SHA256 for hash functions and Ed25519 for digital signature
- Experiment on the DAS-5³
- Up to 1500 nodes

³<http://www.cs.vu.nl/das5/>

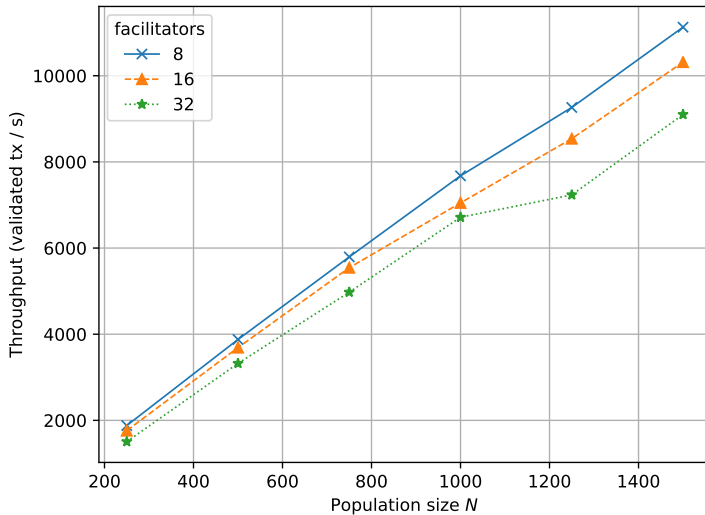
Validated transaction throughput (random neighbour)



Validated transaction throughput (fixed neighbour)



Stress test (fixed neighbour)



Outline

1 Introduction

The dangers of centralisation

Related work

Research question

2 System architecture

System model

Architecture overview

Extended TrustChain

Consensus protocol

Transaction protocol

Validation protocol

3 Experimental results

4 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 untangling consensus and validation
- Global-consensus on transactions is achieved via consensus on CP blocks

Bibliography

-  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>.
-  [Online]. Available: https://en.wikipedia.org/wiki/Death_of_Wei_Zexi.
-  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

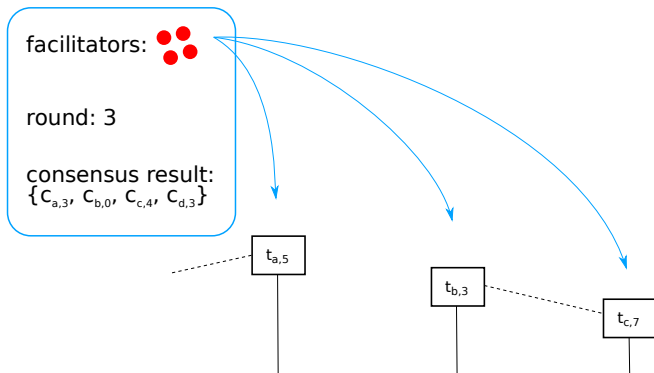
CP block

- 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

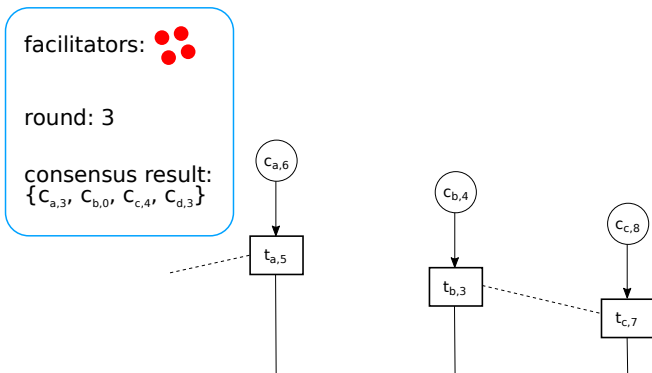
Background on ACS

- Asynchronous common subset
- A simplification of HoneyBadgerBFT [4]
- 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

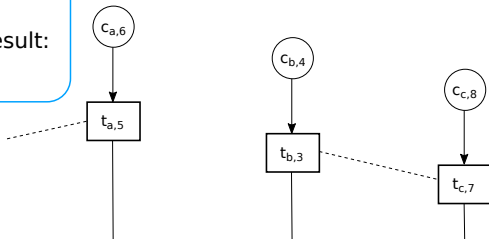


Consensus protocol: part 3

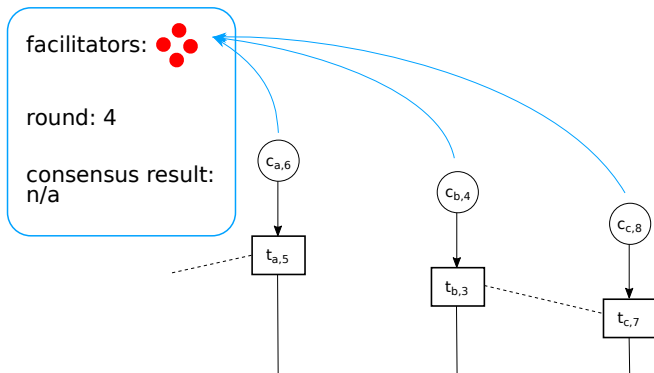
facilitators: lottery
 $\{c_{a,3}, c_{b,0}, c_{c,4}, c_{d,3}\}$

round: 4

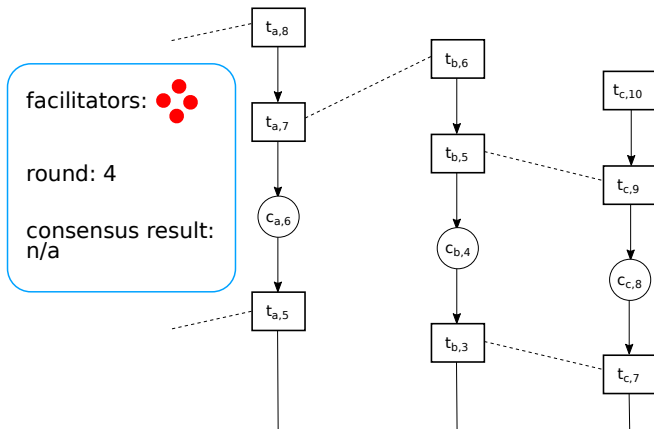
consensus result:
n/a



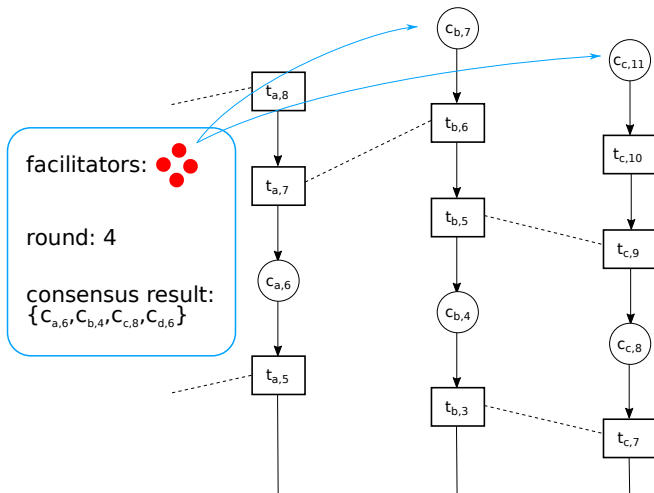
Consensus protocol: part 4



Consensus protocol: part 5



Consensus protocol: part 6



Future work

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