ZLB: A Blockchain to Tolerate Colluding Majorities

Science and Engineering of Consensus 2024
Cornel Tech, New York

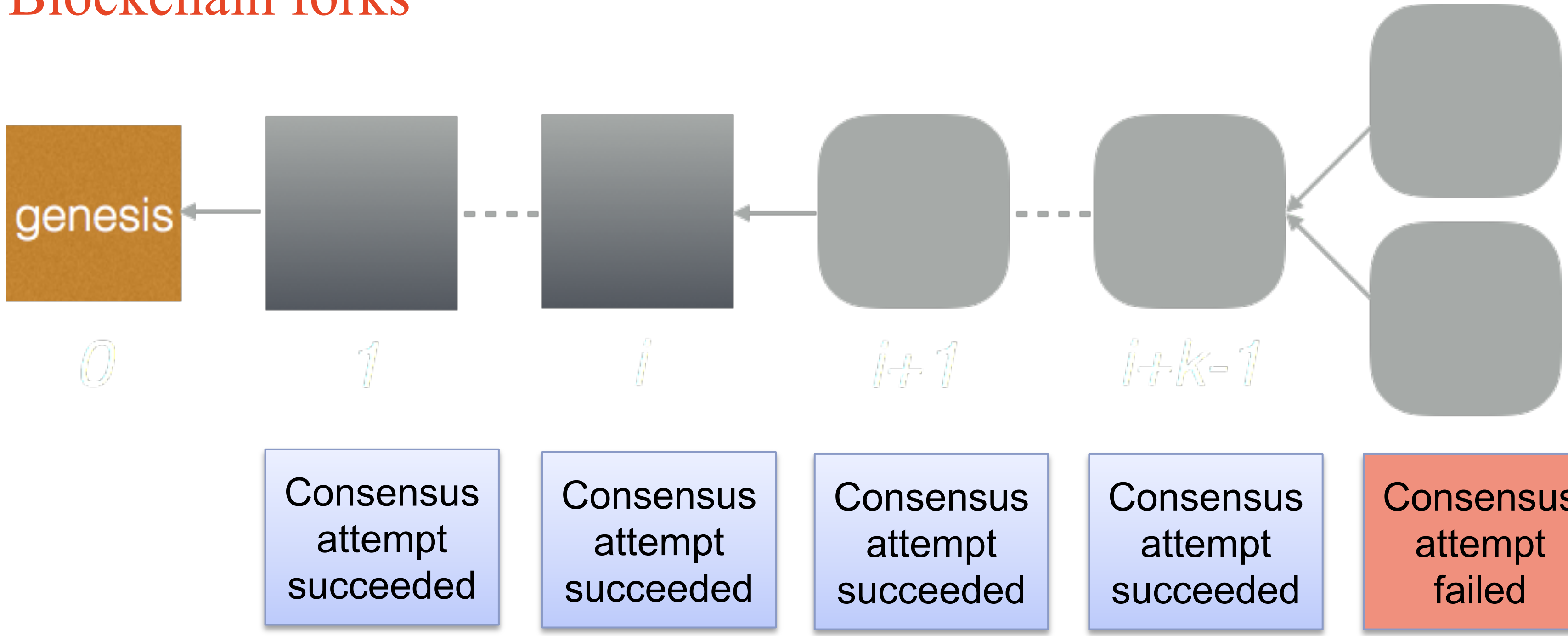
Alejandro Ranchal-Pedrosa and Vincent Gramoli

THE UNIVERSITY OF SYDNEY

Protocol Labs

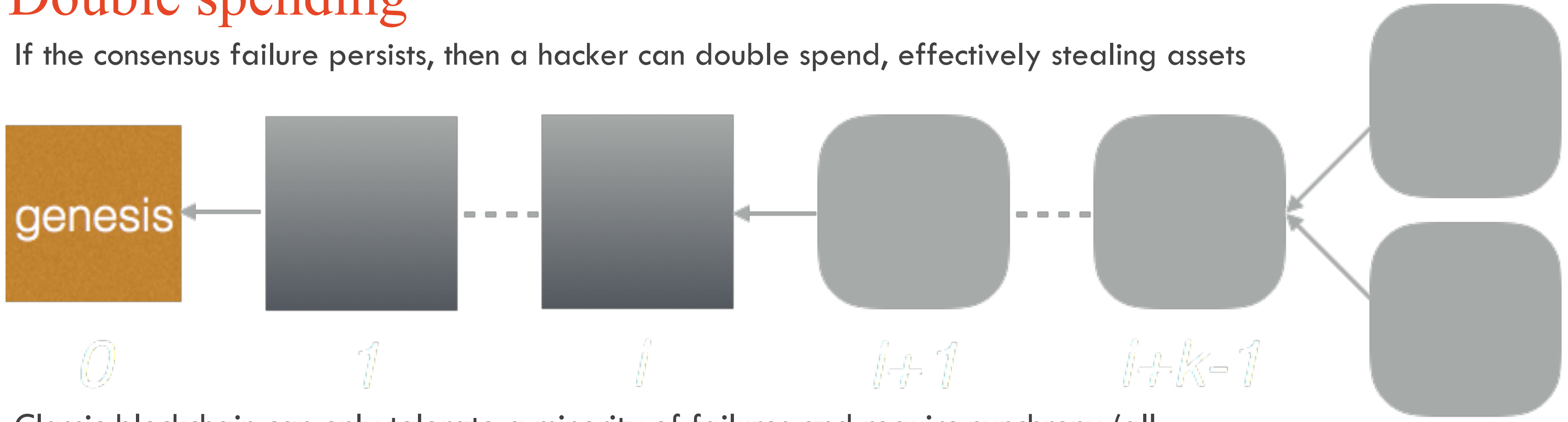
REDBELLY

Blockchain forks


genesis
0
1
i
i+1
i+k-1
Consensus attempt succeeded
Consensus attempt succeeded
Consensus attempt succeeded
Consensus attempt succeeded
Consensus attempt failed

Double spending

If the consensus failure persists, then a hacker can double spend, effectively stealing assets

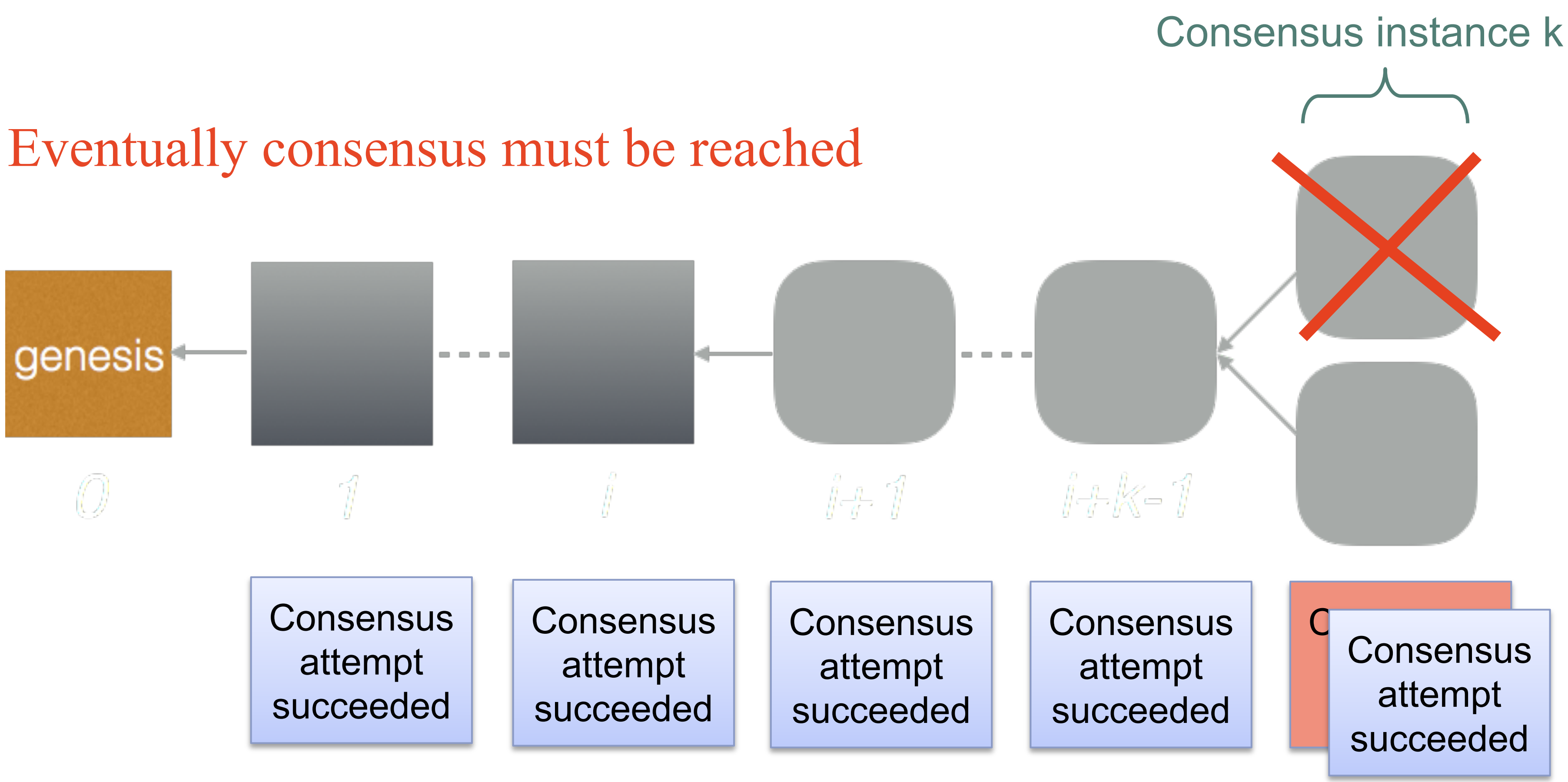


Classic blockchain can only tolerate a minority of failures and require synchrony (all messages to be delivered in less than a known bounded amount of time).

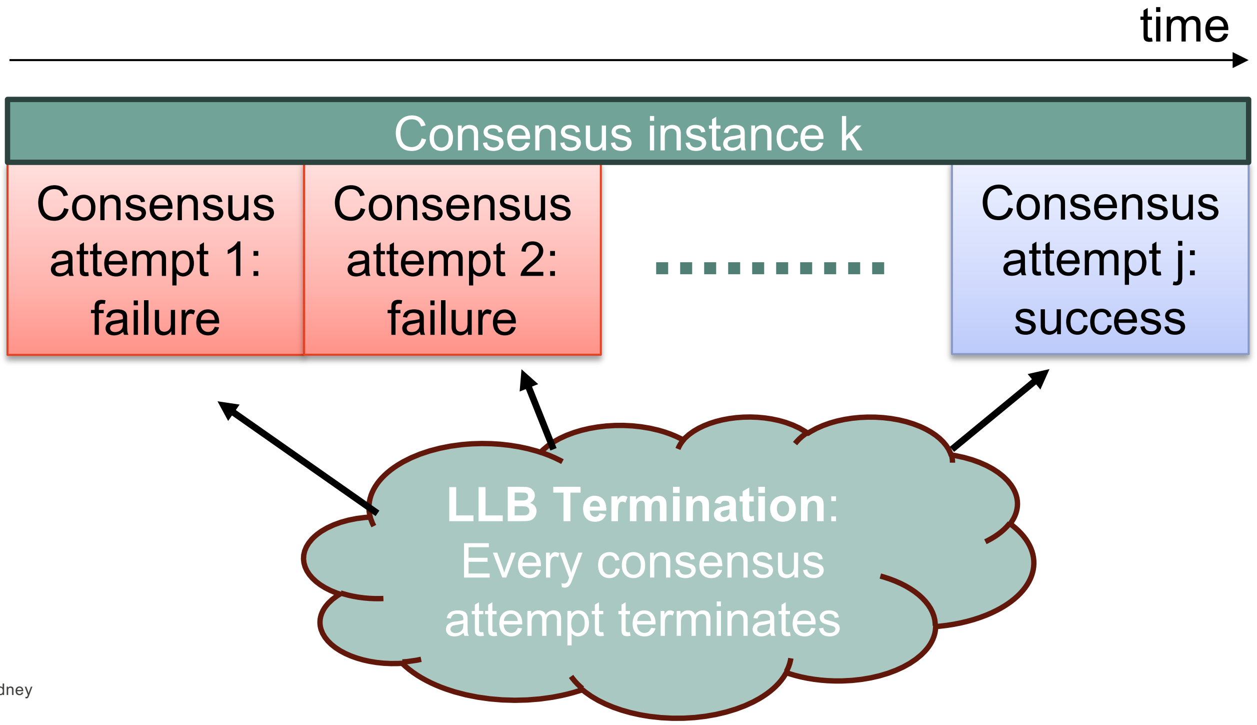
Consensus attempt failed

The balance attack or why forkable blockchains are ill-suited for consortium. C Natoli, V Gramoli. 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2017.

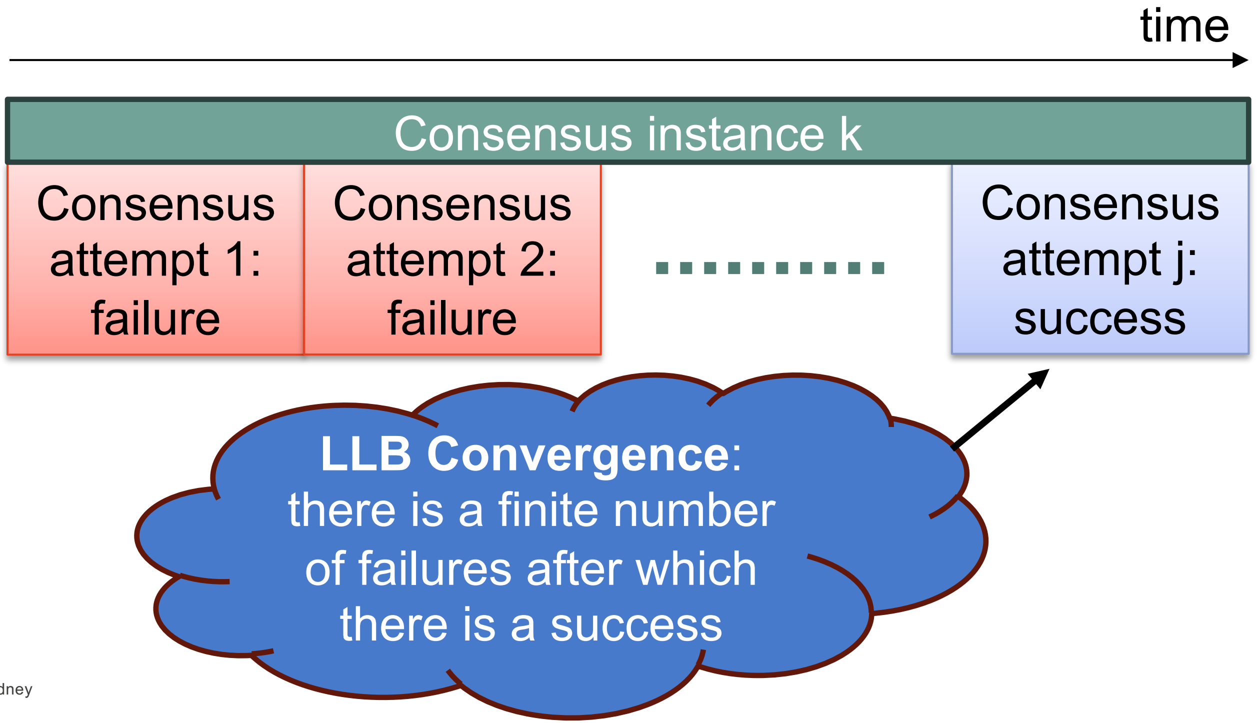
Eventually consensus must be reached


Consensus instance k
genesis
0
1
i
i+1
i+k-1
Consensus attempt succeeded
Consensus attempt succeeded
Consensus attempt succeeded
Consensus attempt succeeded
C
Consensus attempt succeeded

The Longlasting Blockchain (LLB) Problem


time
Consensus instance k
Consensus attempt 1: failure
Consensus attempt 2: failure
Consensus attempt j: success
LLB Termination: Every consensus attempt terminates

The Longlasting Blockchain (LLB) Problem

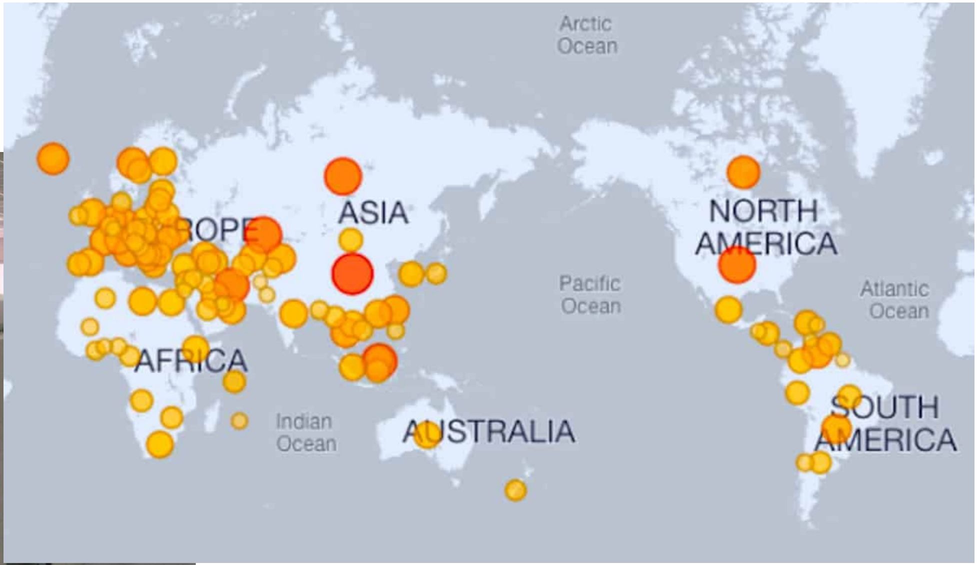

time
Consensus instance k
Consensus attempt 1: failure
Consensus attempt 2: failure
Consensus attempt j: success
LLB Convergence: there is a finite number of failures after which there is a success

How to tolerate a majority of failures?

Failure Paradigm Shift

From liveness violations in closed networks: distributed systems (datacenters, distributed databases, cloud) experiencing mostly omissions faults.

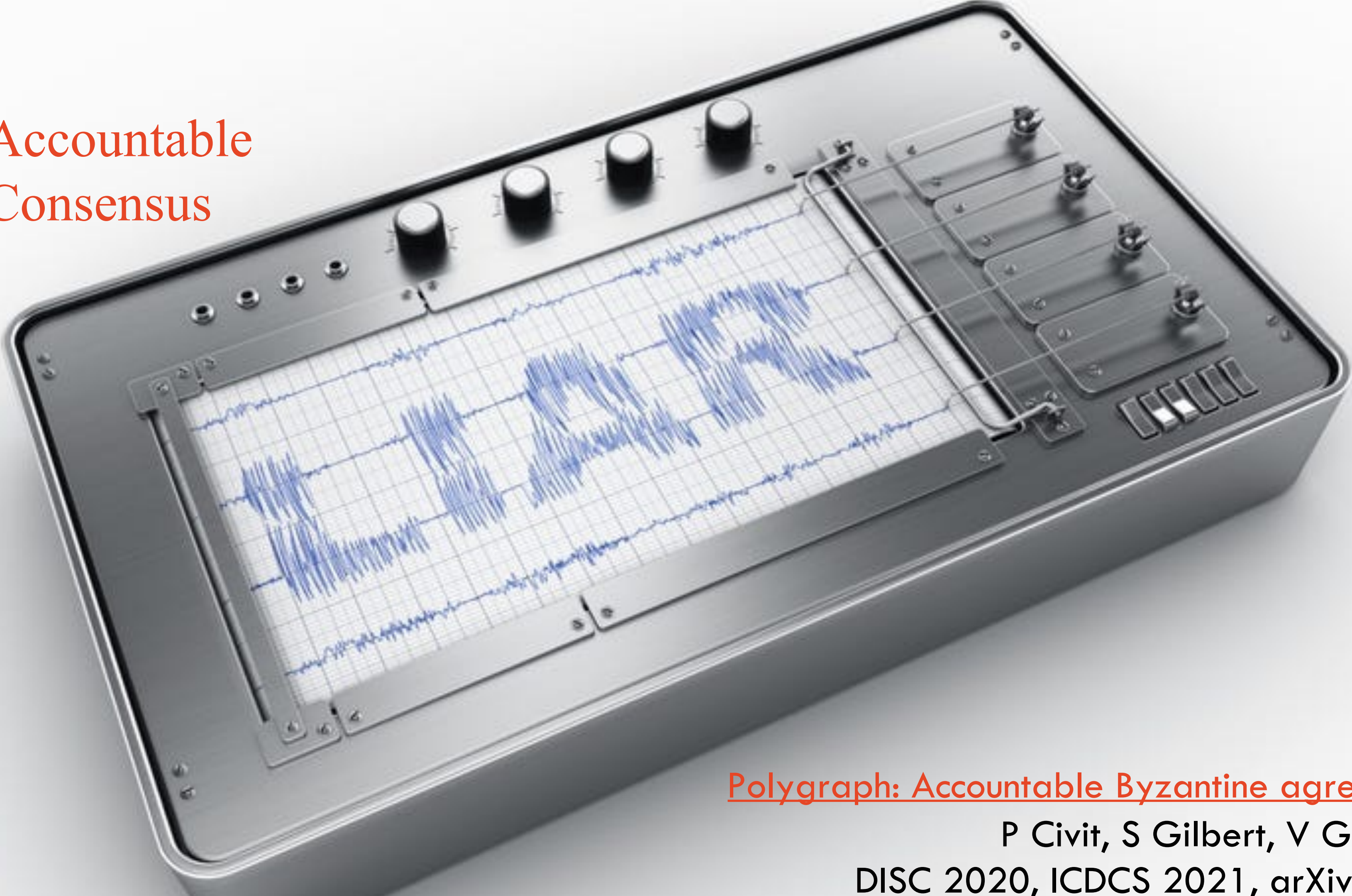
Failure Paradigm Shift



To safety violations in open networks: blockchain networks are open networks that incentivize nodes to be active (either by contributing or stealing).

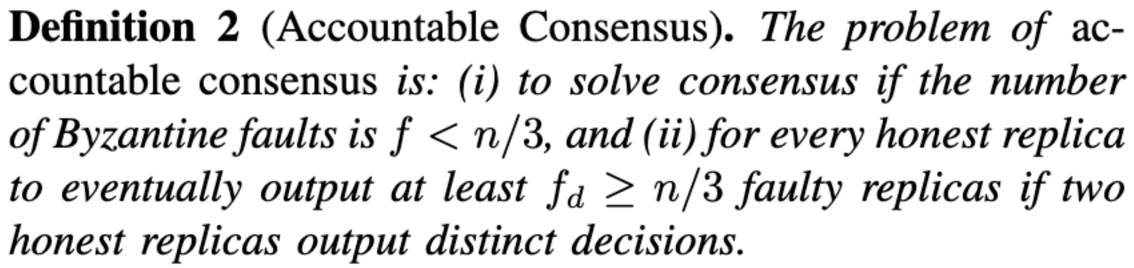
From liveness violations in closed networks: distributed systems (datacenters, distributed databases, cloud) experiencing mostly omissions faults.

Accountable Consensus



Polygraph: Accountable Byzantine agreement.
P Civit, S Gilbert, V Gramoli.
DISC 2020, ICDCS 2021, arXiv 2019.

Accountable Consensus



Definition 2 (Accountable Consensus). *The problem of* accountable consensus *is: (i) to solve consensus if the number of Byzantine faults is $f < n/3$, and (ii) for every honest replica to eventually output at least $f_d \geq n/3$ faulty replicas if two honest replicas output distinct decisions.*

Polygraph: Accountable Byzantine agreement.
P Civit, S Gilbert, V Gramoli.
DISC 2020, ICDCS 2021, arXiv 2019.

The Zero Loss Blockchain Solution

1) We need to get rid of the synchrony assumption, so that consensus can be reached despite network attacks

2) We bound the number of participants that run consensus at a time to ensure consensus is reached deterministically

3) We use an **accountable consensus algorithm** that tolerates any number of failures to identify misbehaving participants

4) We remove or replace these participants when identified and repeat the step above to finally reach a state where consensus is reached successfully.

The Zero Loss Blockchain Solution

We assume partial synchrony as consensus is unsolvable with asynchrony

1) We need to get rid of the synchrony assumption, so that consensus can be reached despite network attacks

2) We bound the number of participants that run consensus at a time to ensure consensus is reached deterministically

3) We use an **accountable consensus algorithm** that tolerates any number of failures to identify misbehaving participants

4) We remove or replace these participants when identified and repeat the step above to finally reach a state where consensus is reached successfully.

The Zero Loss Blockchain Solution

We use DBFT with the superblock optimization as in Redbelly to scale to large number of nodes

1) We need to get rid of the synchrony assumption, so that consensus can be reached despite network attacks

2) We bound the number of participants that run consensus at a time to ensure consensus is reached deterministically

3) We use an **accountable consensus algorithm** that tolerates any number of failures to identify misbehaving participants

4) We remove or replace these participants when identified and repeat the step above to finally reach a state where consensus is reached successfully.

Red Belly: A secure, fair and scalable open blockchain. T Crain, C Natoli, V Gramoli. IEEE Symposium on Security and Privacy (S&P), 466-483, 2021

The Zero Loss Blockchain Solution

1) We need to get rid of the synchrony assumption, so that consensus can be reached despite network attacks

2) We bound the number of participants that run consensus at a time to ensure consensus is reached deterministically

3) We use an **accountable consensus algorithm** that tolerates any number of failures to identify misbehaving participants

4) We remove or replace these participants when identified and repeat the step above to finally reach a state where consensus is reached successfully.

We use the Polygraph algorithm to solve the accountable consensus problem for any number of faults

Red Belly: A secure, fair and scalable open blockchain. T Crain, C Natoli, V Gramoli. IEEE Symposium on Security and Privacy (S&P), 466-483, 2021

The Zero Loss Blockchain Solution

1) We need to get rid of the synchrony assumption, so that consensus can be reached despite network attacks

2) We bound the number of participants that run consensus at a time to ensure consensus is reached deterministically

3) We use an **accountable consensus algorithm** that tolerates any number of failures to identify misbehaving participants

4) We remove or replace these participants when identified and repeat the step above to finally reach a state where consensus is reached successfully.

We converge towards a state where sufficiently many honest participants reach consensus

Red Belly: A secure, fair and scalable open blockchain. T Crain, C Natoli, V Gramoli. IEEE Symposium on Security and Privacy (S&P), 466-483, 2021

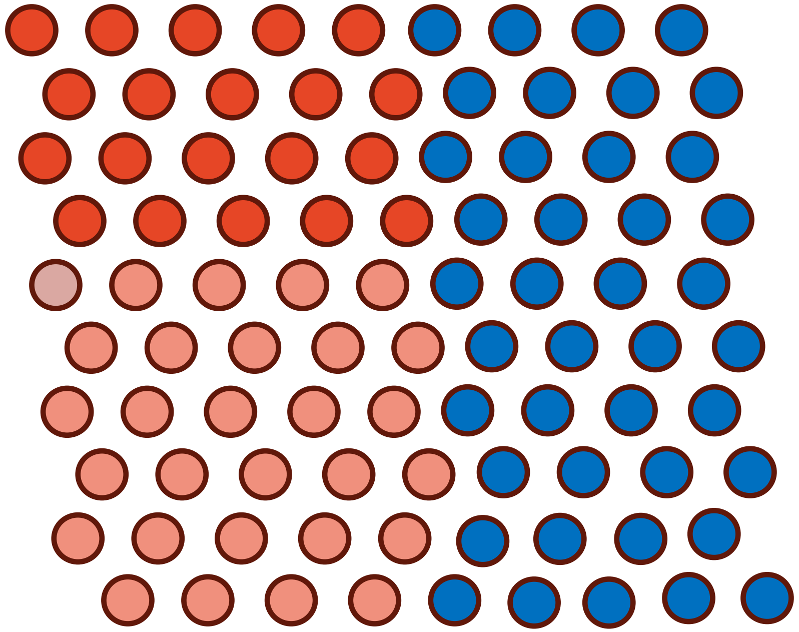
Assumptions

1) A slowly adaptive adversary selects at most $f < 5n/9$ faulty consensus participants among n participants during each consensus instance.

2) Among the f faulty nodes:
   - $t < n/3$ are Byzantine (arbitrary failures)
   - f-t are alive-but-corrupt (faulty but active) *

3) We assume a public-key infrastructure that associates participants to their public key

4) We assume a large pool of $m >> n$ nodes where $2n/3$ are honest and m-$2n/3$ are alive-but-corrupt from which new consensus participants are selected.

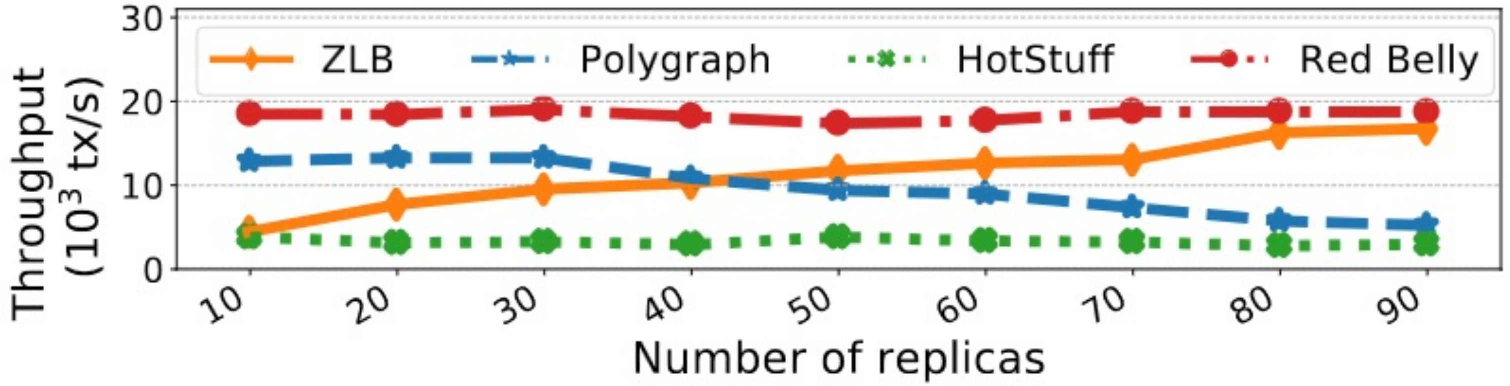


n = 90, f = 49 (t = 29, f-t = 20)

* Flexible Byzantine fault tolerance. D. Malkhi, K. Nayak, and L. Ren. CCS 2019.

Experimental Evaluation

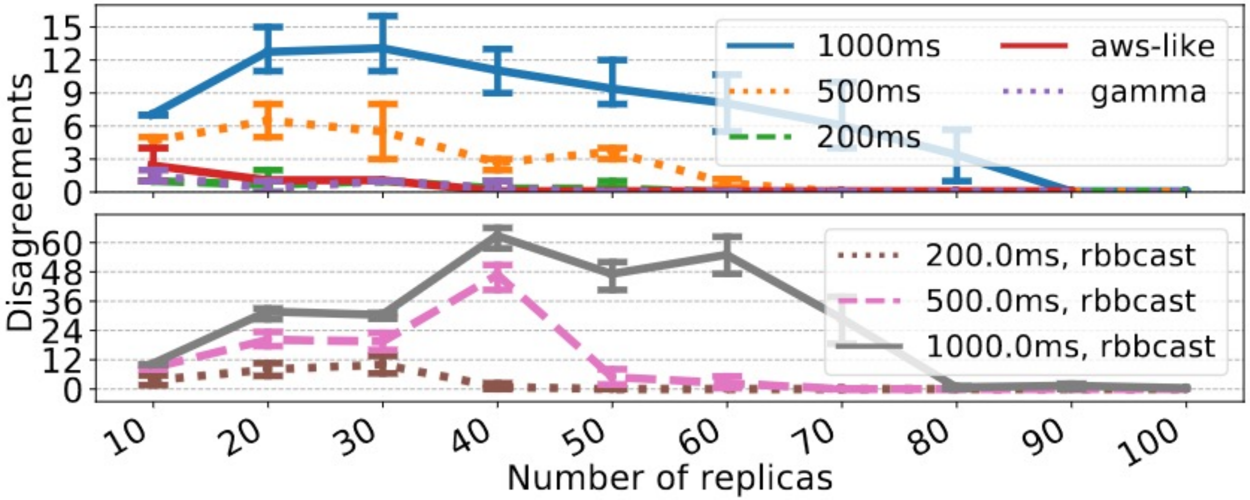
Performance of Zero Loss Blockchain (ZLB)



Geodistributed experiments: California, Oregon, Ohio, Frankfurt and Ireland.
Up to 90 machines, each with 4 vCPUs and 7.5 GiB of memory on AWS
Plotted values averaged over 3-5 runs. Transactions are ~400-bytes UTXO.
Facebook Libra was delivering 11 TPS, so we only tested the raw SMR HotStuff in its default implementation in C++.

Number of Disagreements under Attacks

Disagreeing decisions for various uniform delays and for delays generated from a Gamma distribution and a distribution that draws from observed AWS latencies, when equivocating while voting for a decision (top), and while broadcasting the proposals (bottom), for $f = \lceil 5n/9 \rceil - 1$.
Error bars are 95% confidence interval.



Zero Loss Payment Application

Zero Loss Payment Application Assumptions

1) Assets are fungible

2) Attacker can fork into a branches maximum and transfer a maximum gain G

3) Each consensus participant deposits $0.3G/n$ coins that are kept for m blocks

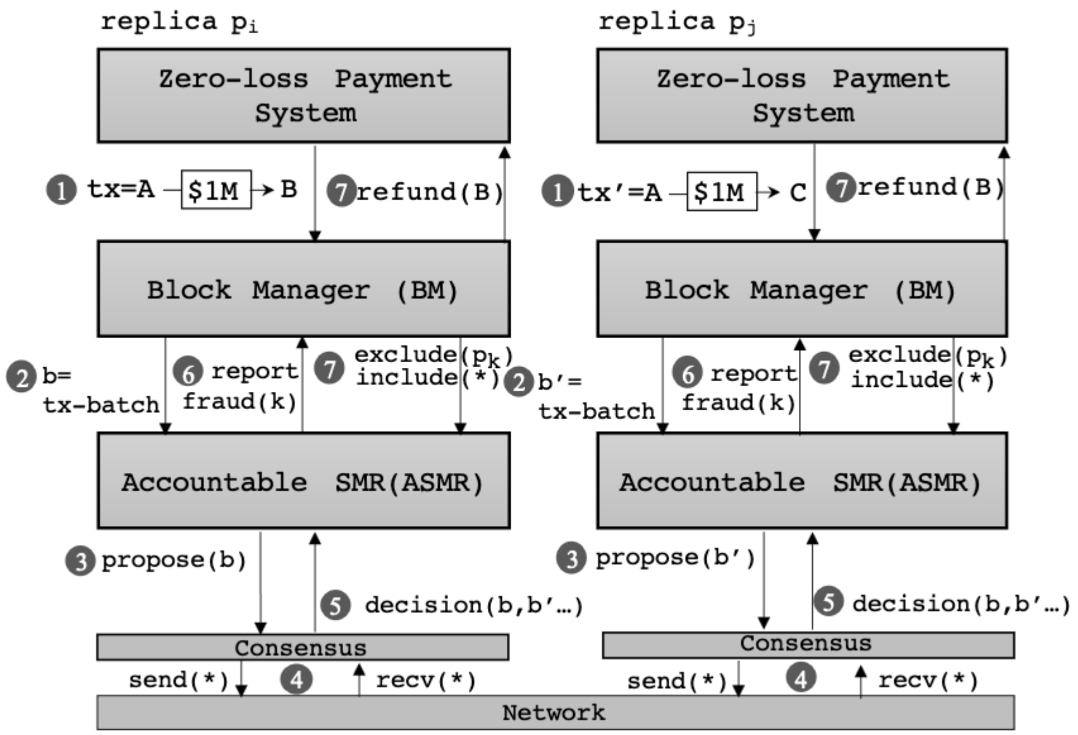
4) A transaction is committed if it reaches block depth m

5) Byzantine replicas cannot communicate infinitely faster than honest replicas in different partitions

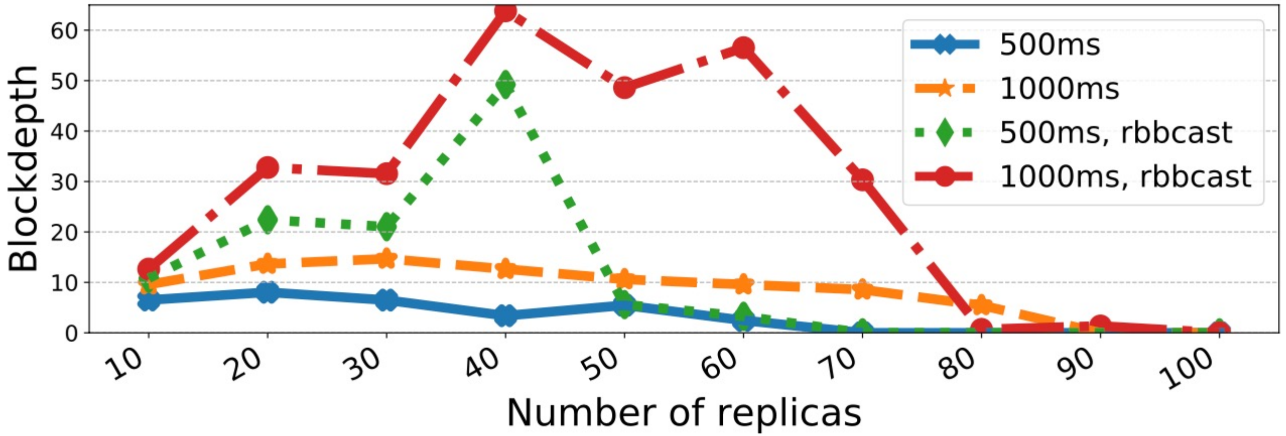
Zero Loss Payment Application

Consider that Alice (A) has 1M coins initially and tries to double spend.
Alice hacks the code of node p_k to lead p_i and p_j to disagree about b and b' blocks.

The Accountable SMR detects the misbehavior of p_k and uses the deposit of p_k to refund Bob (B).



Zero Loss Payment Application



Minimum finalization blockdepth m to obtain zero-loss payment app
Failures $f = \lceil 5n/9 \rceil - 1$

Conclusion

Prior to this work, blockchains could not tolerate a majority of failures.

ZLB tolerates a majority of failures by building upon:
- Recent advances on the Accountable Consensus problem
- The failure paradigm shift between closed networks and open blockchain networks

ZLB leverages the superblock optimization to scale well and achieves similar performance as Redbelly with 90 geodistributed consensus participants. It outperforms the raw state machine replication of HotStuff by 5.6 times.

It allows to implement a zero-loss payment system, which rolls back conflicting transactions with deposited coins, which prevents double spending.

More information

Redbelly Network

Backup

Why f < 5n/9 ?

In Polygraph, honest replicas identify f_d >= n/3 faulty replicas

Even if honest replicas do not agree on the same replicas, they will eventually ban the same set of replicas within the static period of the adversary to obtain n’ consensus participants

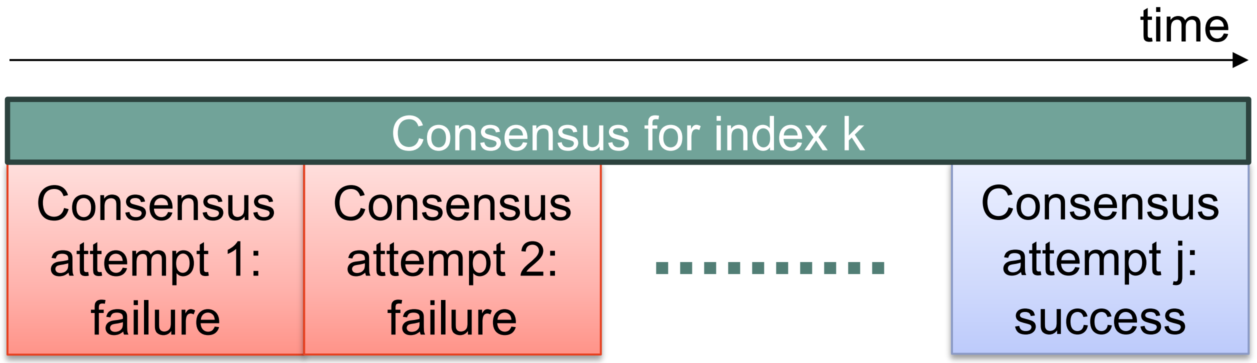
Since n’<= n-f_d and f’ = f-f_d, we have that the number of faulty participants not excluded is f’ < n’/3 (1)

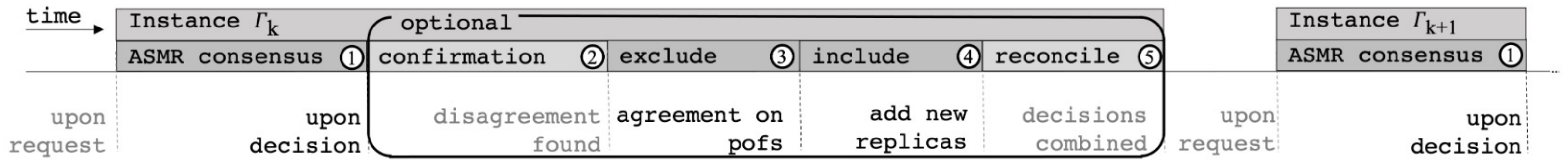
Since n’/3 <= (n-f_d)/3 and f_d >= n/3, this means that n’/3 <= n/3-n/9 (2)

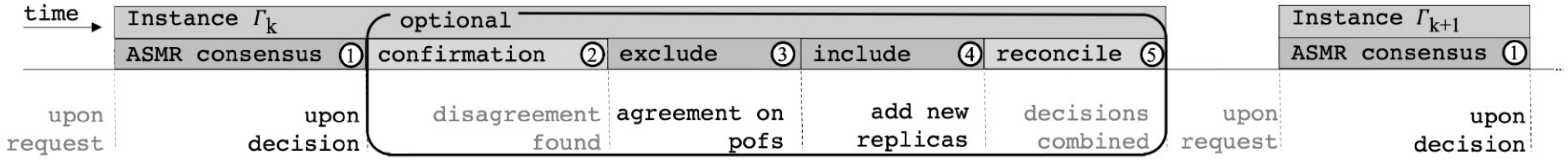
The conjunction of (1) and (2) leads to f’<2n/9.

By choosing f < 5n/9, we guarantee that f – f_d < 2n/9.

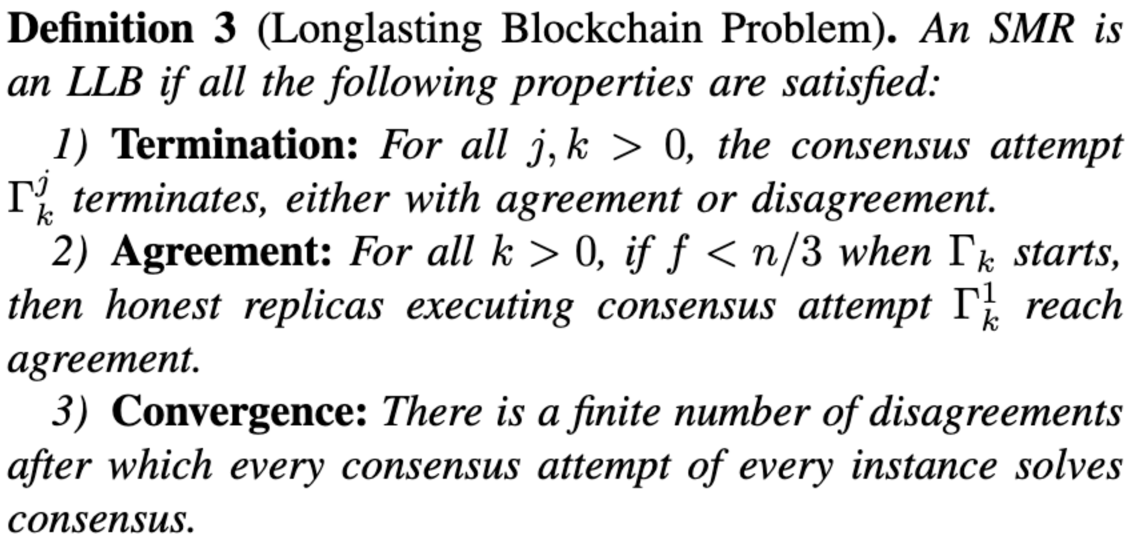
Longlasting Blockchain Problem


time
Consensus for index k
Consensus attempt 1: failure
Consensus attempt 2: failure
Consensus attempt j: success




time
Instance Γ_k
optional
ASMR consensus ①
confirmation ②
exclude ③
include ④
reconcile ⑤
Instance Γ_{k+1}
ASMR consensus ①
upon request
upon decision
disagreement found
agreement on pofs
add new replicas
decisions combined
upon request
upon decision

Longlasting Blockchain Problem



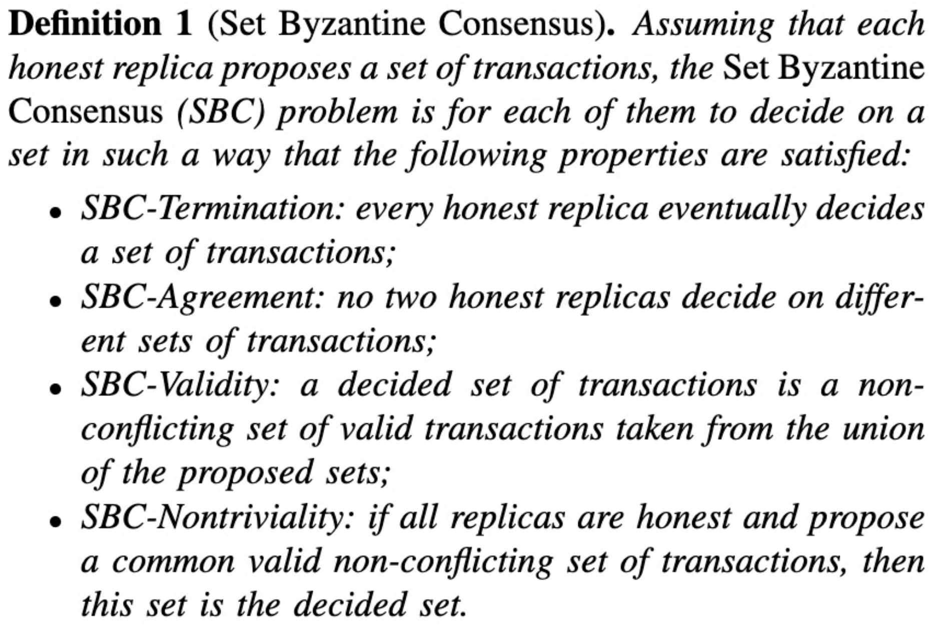
Definition 3 (Longlasting Blockchain Problem). *An SMR is an LLB if all the following properties are satisfied:*

1) **Termination:** *For all $j, k > 0$, the consensus attempt Γ_k^j terminates, either with agreement or disagreement.*

2) **Agreement:** *For all $k > 0$, if $f < n/3$ when Γ_k starts, then honest replicas executing consensus attempt Γ_k^1 reach agreement.*

3) **Convergence:** *There is a finite number of disagreements after which every consensus attempt of every instance solves consensus.*

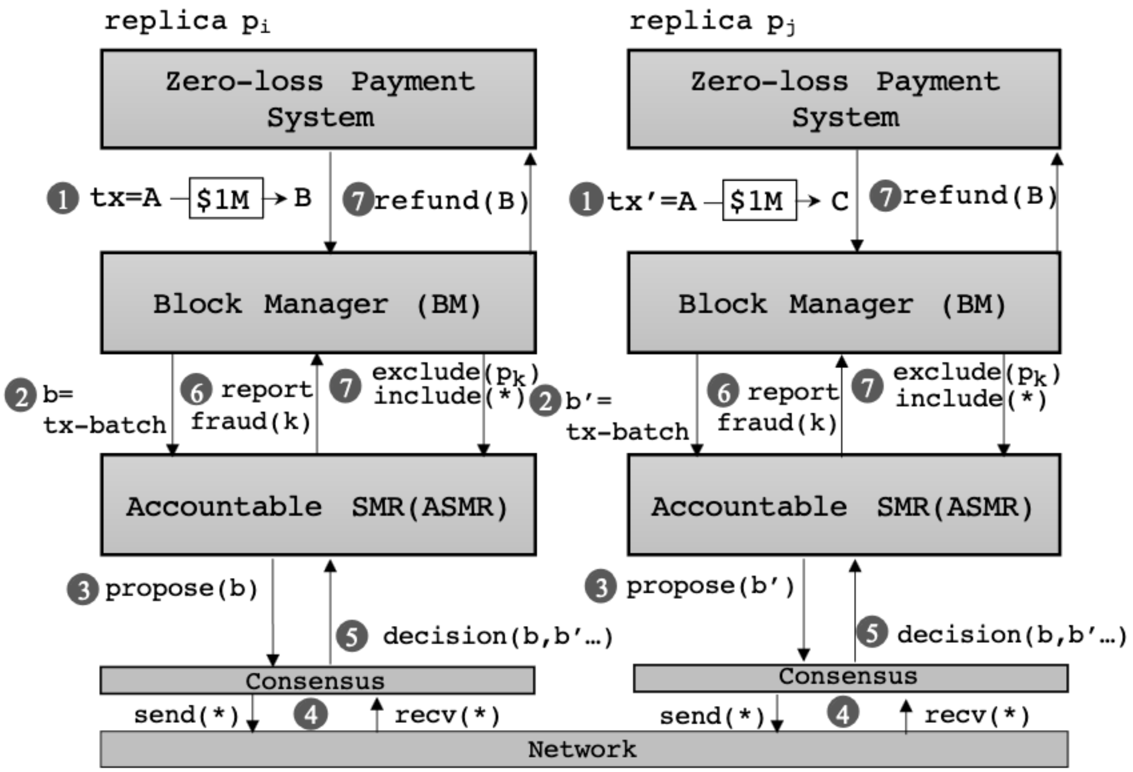
Set Byzantine Consensus

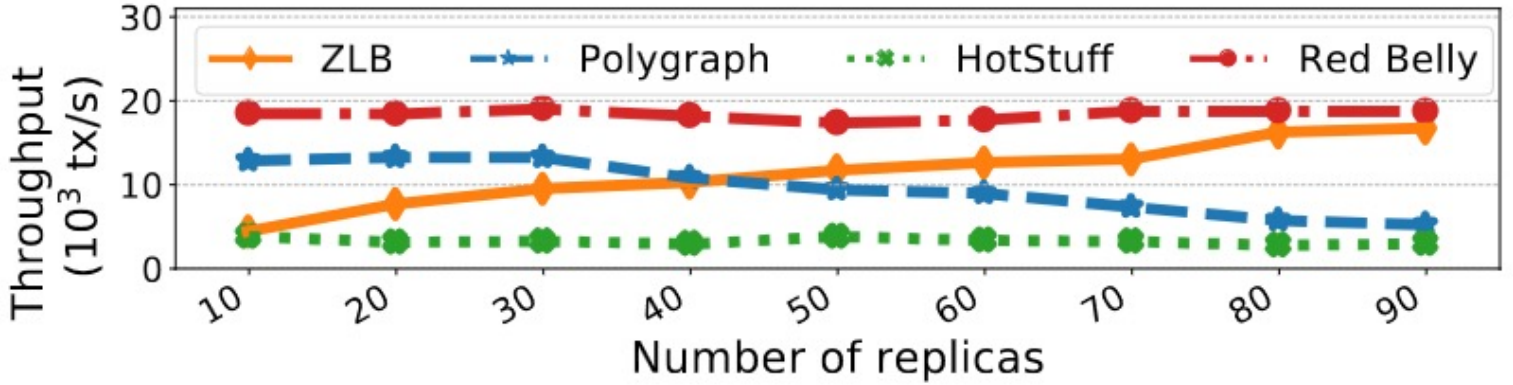


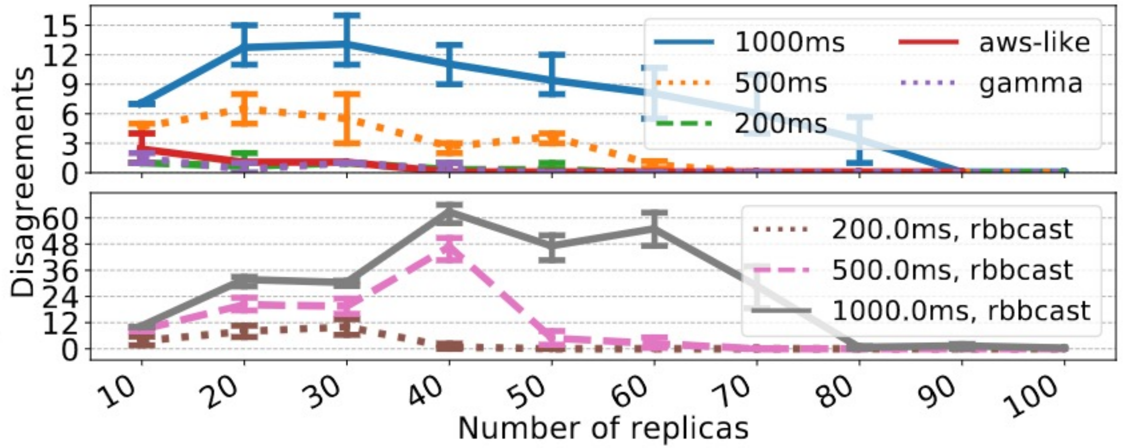
Definition 1 (Set Byzantine Consensus). *Assuming that each honest replica proposes a set of transactions, the* Set Byzantine Consensus *(SBC) problem is for each of them to decide on a set in such a way that the following properties are satisfied:*

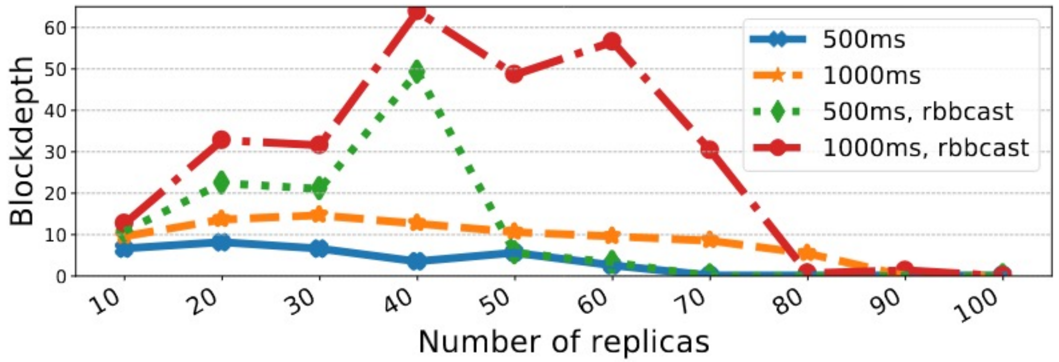
- *SBC-Termination: every honest replica eventually decides a set of transactions;*
- *SBC-Agreement: no two honest replicas decide on different sets of transactions;*
- *SBC-Validity: a decided set of transactions is a non-conflicting set of valid transactions taken from the union of the proposed sets;*
- *SBC-Nontriviality: if all replicas are honest and propose a common valid non-conflicting set of transactions, then this set is the decided set.*

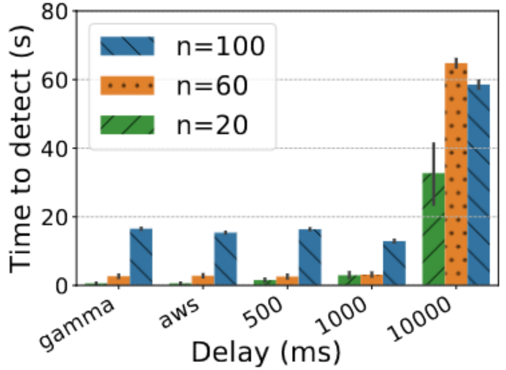
Red Belly: A secure, fair and scalable open blockchain. T Crain, C Natoli, V Gramoli. IEEE Symposium on Security and Privacy (S&P), 466-483, 2021

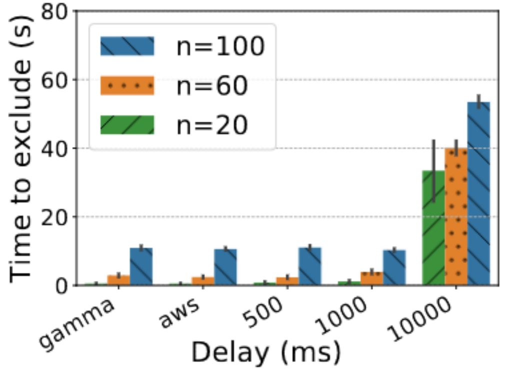

replica p_i
Zero-loss Payment System
1 tx=A → $1M → B
7 refund(B)
Block Manager (BM)
2 b= tx-batch
6 report fraud(k)
7 exclude(p_k) include(*)
Accountable SMR(ASMR)
3 propose(b)
5 decision(b,b'…)
Consensus
send(*)
4
recv(*)
replica p_j
Zero-loss Payment System
1 tx'=A → $1M → C
7 refund(B)
Block Manager (BM)
2 b'= tx-batch
6 report fraud(k)
7 exclude(p_k) include(*)
Accountable SMR(ASMR)
3 propose(b')
5 decision(b,b'…)
Consensus
send(*)
4
recv(*)
Network

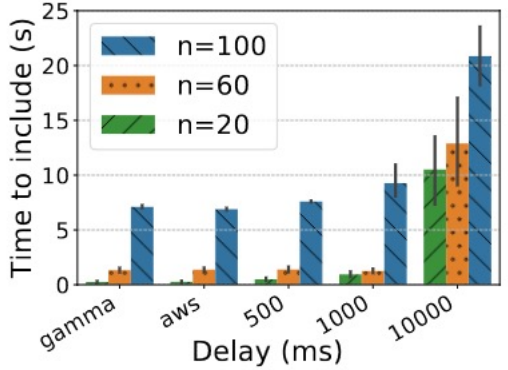

Throughput
(10^3 tx/s)
30
20
10
0
ZLB
Polygraph
HotStuff
Red Belly
10
20
30
40
50
60
70
80
90
Number of replicas

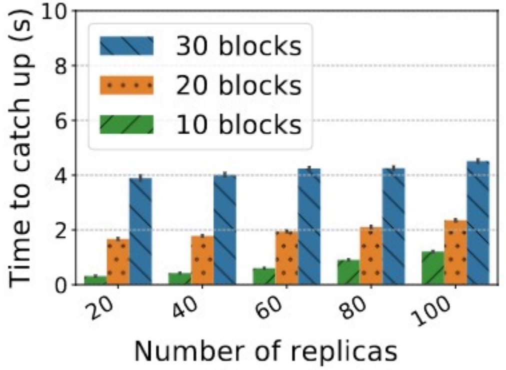

Disagreements
15
12
9
6
3
0
1000ms
500ms
200ms
aws-like
gamma
60
48
36
24
12
0
200.0ms, rbbcast
500.0ms, rbbcast
1000.0ms, rbbcast
10
20
30
40
50
60
70
80
90
100
Number of replicas


Blockdepth
0
10
20
30
40
50
60
500ms
1000ms
500ms, rbbcast
1000ms, rbbcast
10
20
30
40
50
60
70
80
90
100
Number of replicas


80
60
40
20
0
Time to detect (s)
n=100
n=60
n=20
gamma
aws
500
1000
10000
Delay (ms)


80
60
40
20
0
Time to exclude (s)
n=100
n=60
n=20
gamma
aws
500
1000
10000
Delay (ms)


25
20
15
10
5
0
Time to include (s)
n=100
n=60
n=20
gamma
aws
500
1000
10000
Delay (ms)


10
8
6
4
2
0
Time to catch up (s)
30 blocks
20 blocks
10 blocks
20
40
60
80
100
Number of replicas

Project Quality & Innovation

1) What is the significant gap in knowledge or problem?

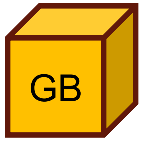
2) What is your innovative solution?

3) What methods and/or conceptual/theoretical framework will be used in the project?

4) What is the anticipated new knowledge that will be created by the project?

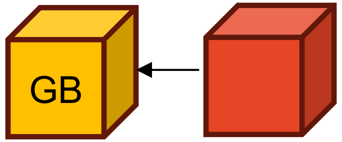
5) How might the research result in economic, environmental, social and/or cultural benefits to Australia?

Blockchain



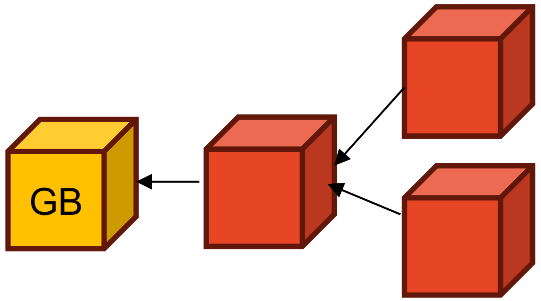
Initially, there is a genesis block (GB) that all nodes is aware of.

Blockchain



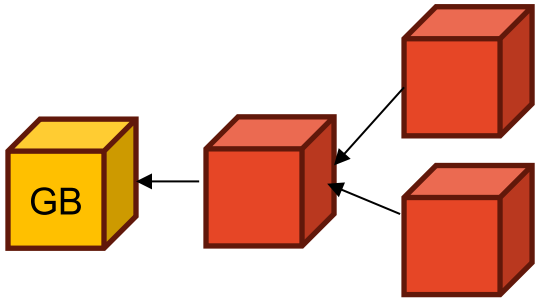
Then nodes try to append a new unique block at the next available index

Blockchain



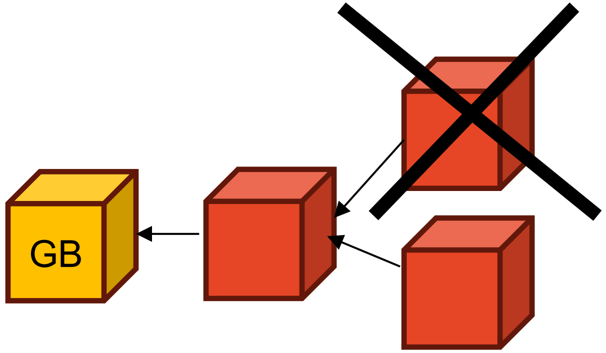
Nodes may disagree about the next block that should appended

Blockchain



This is why it is crucial to eventually agree on the uniqueness of the block to be appended

Blockchain



If not resolved rapidly, this leads to double spending (theft or loss of assets)