Tendermint

Byzantine Fault Tolerance in the Age of Blockchains

Ethan Buchman
University of Guelph
MASc Defense

Outline

- Background
  - Consensus and Atomic Broadcast
  - Byzantine Fault Tolerance
  - Bitcoin
- Tendermint
  - Algorithm
  - Security
- Applications
  - Blockchain Application Logic
  - Blockchain Gateway Interface
- Experiments
  - Performance
  - Fault Tolerance
- Future Work

Consensus

"Then we are agreed nine to one that we will say our previous vote was unanimous!"

Impossible!

Impossibility of Distributed Consensus with One Faulty Process

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY A. LYNCH

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England

Abstract. The consensus problem involves an asynchronous system of processes, some of which may be unreliable. The problem is for the reliable processes to agree on a binary value. In this paper, it is shown that every protocol for this problem has the possibility of nontermination, even with only one faulty process. By way of contrast, solutions are known for the synchronous case, the "Byzantine Generals" problem.

Consensus vs Atomic Broadcast

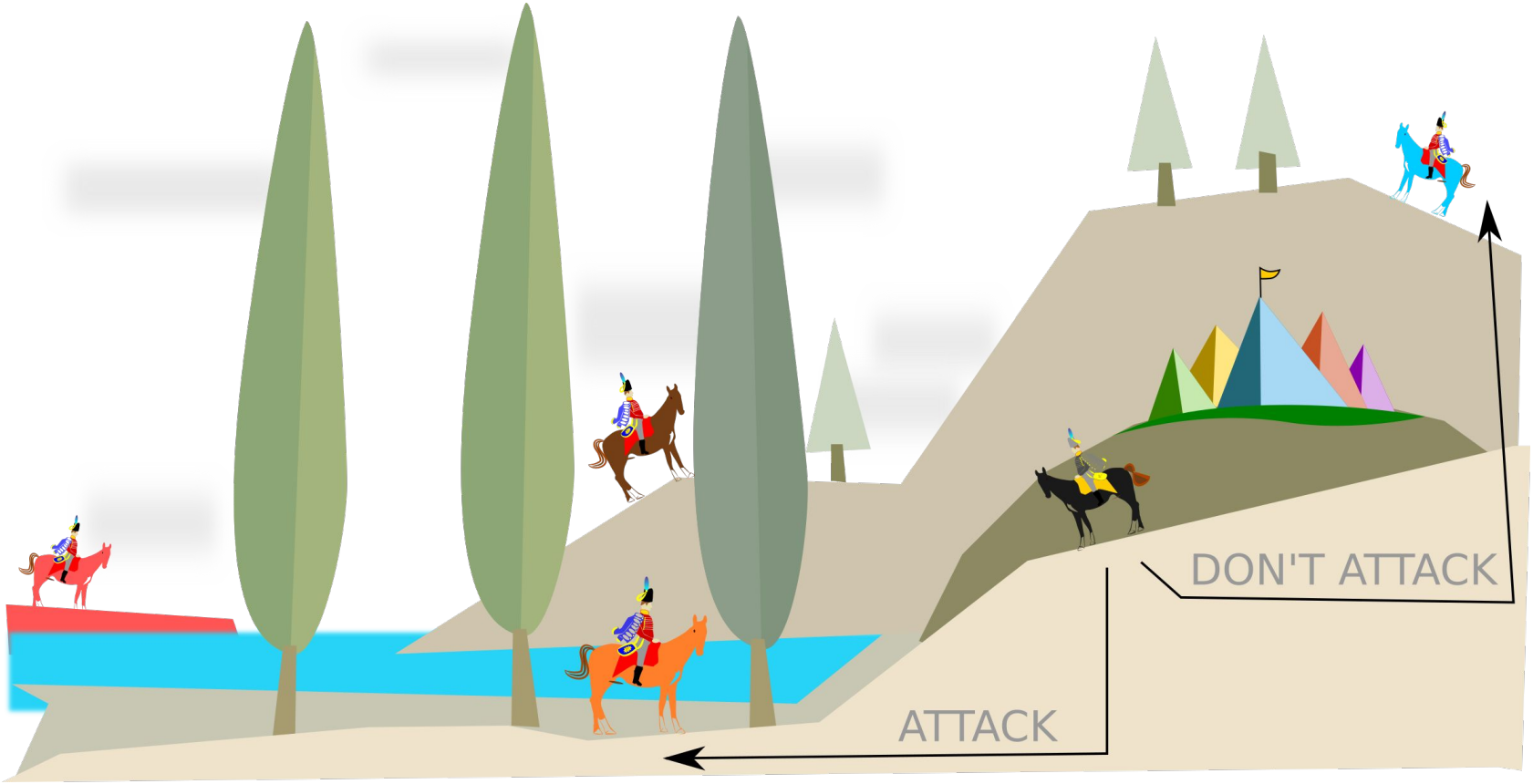
Consensus

- all correct processes eventually decide one value
- terminates
- eg. leader election, mutual exclusion lock

Atomic Broadcast

- all correct processes eventually decide on the same order of values
- continuous
- eg. replicating a transaction log

The Byzantine Generals



Bitcoin

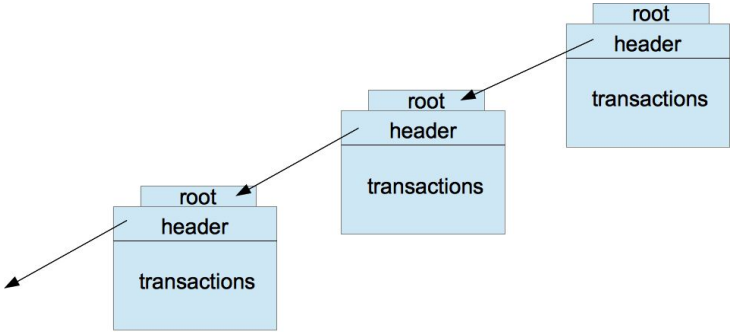
Bitcoin: A Peer-to-Peer Electronic Cash System

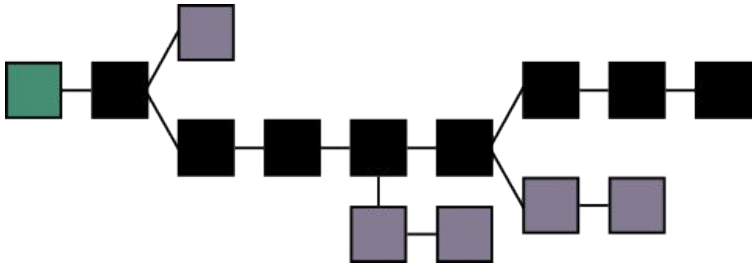
Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

“a solution to the double spending problem of decentralized digital currency”

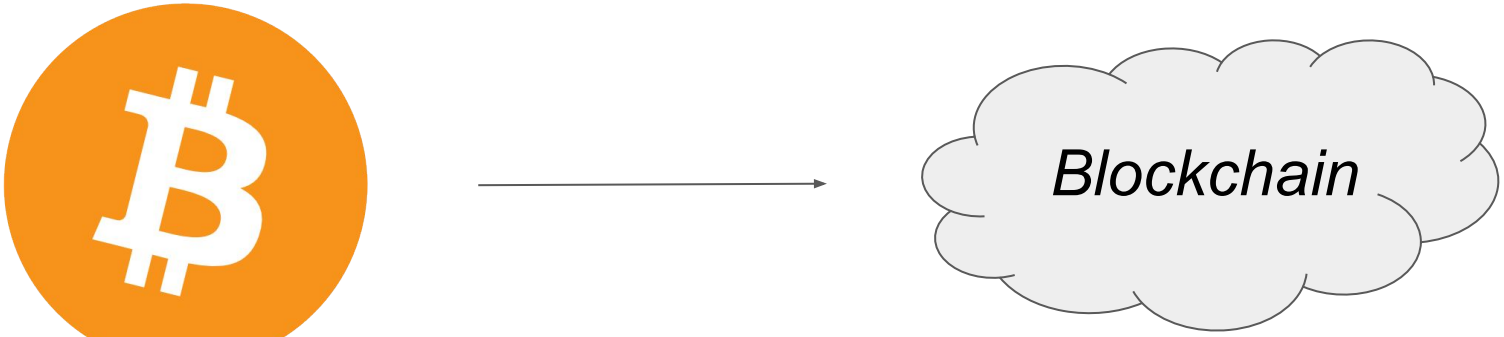
Bitcoin: BFT Atomic Broadcast

- Randomness
  - proposers chosen by cryptographic random lottery
  - “Proof-of-Work”
- Weak synchrony
  - latency << time between lottery winners
- “Economic security”
  - participation is expensive
  - byzantine behaviour has an opportunity cost (longest chain wins)
- Variable set of processes





Blockchain



- BFT Atomic Broadcast
  - BFT Consensus
  - Hash-linked blocks
- Application utilizes public keys
- Economic security (optional)

Blockchain at the intersection of global trends

1) “Cloud”
= architectural
decentralization

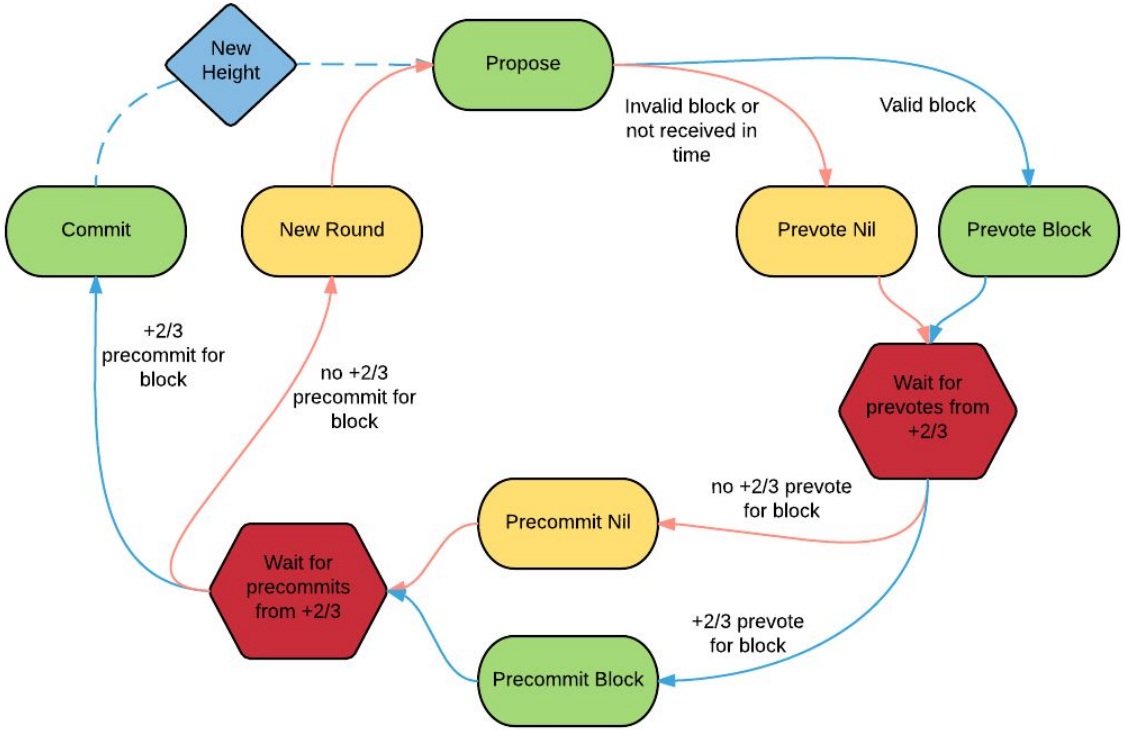
2) “Sharing economy”
= political
decentralization

3) “Cyber security”

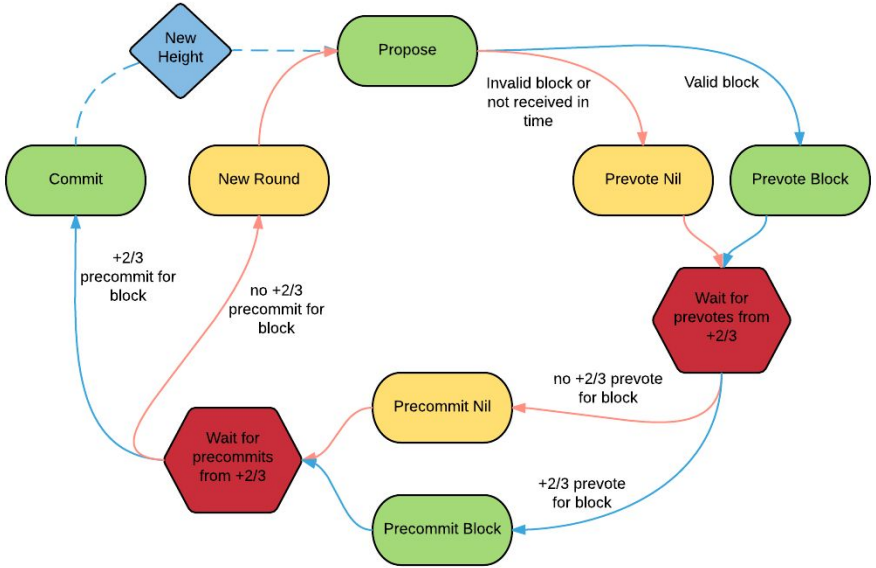
Outline

- Background
  - Consensus and Atomic Broadcast
  - Byzantine Fault Tolerance
  - Bitcoin
- Tendermint
  - Algorithm
  - Security
- Applications
  - Blockchain Application Logic
  - Blockchain Gateway Interface
- Experiments
  - Performance
  - Fault Tolerance
- Future Work

Tendermint Consensus



Tendermint Consensus



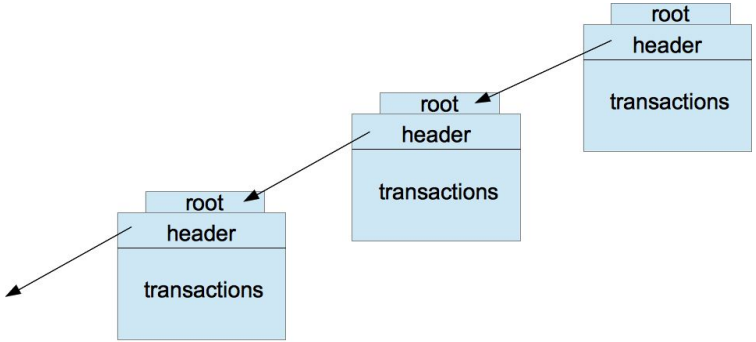
Locking Rules

- Locked - a validator is locked on a block once they precommit it
- Prevote-the-Lock - must prevote/propose the locked block
- Unlock-on-Polka - may unlock only after seeing polka at higher round than the round they locked in

Tendermint Blockchain

Block contains:

- Header
  - Block Height
  - Previous Block Hash
  - Commit Round
  - Commit Time
  - Merkle root of validator set
  - Merkle root of LastCommit
  - Merkle root of Transactions
  - Merkle root of application state (from last block's txs)
- LastCommit - +⅔ precommits from last block
- Transactions



Blocks are an optimization

- *amortize the cost of consensus over many transactions*
- *provide high frequency checkpointing via hashes*

Tendermint Security

- Safety & Liveness - guaranteed by algorithm
- Optimal BFT - requires ⅓+ Byzantine failures to violate safety/liveness
- Accountability - identify & punish those that attack the network
  - Legal security if identifiable persons
  - Economic security if using security deposits
- Antifragility - accountability enables re-birth after failure

Outline

- Background
  - Consensus and Atomic Broadcast
  - Byzantine Fault Tolerance
  - Bitcoin
- Tendermint
  - Consensus
  - Blockchain
  - Security
- Applications
  - Blockchain Application Logic
  - Blockchain Gateway Interface
- Experiments
  - Performance
  - Fault Tolerance
- Future Work

Blockchain Application Logic

- Bitcoin
  - Programmable money
  - “Functional programming” - no state, contracts renewed every transaction
  - Forth like, purposefully not Turing-complete

- Ethereum
  - “Smart contracts”
  - “Contract-oriented” - stateful contracts live independently on the blockchain
  - Turing complete (Ethereum Virtual Machine)

- Something more flexible?!

Blockchain Gateway Interface

APPLICATION PLATFORMS

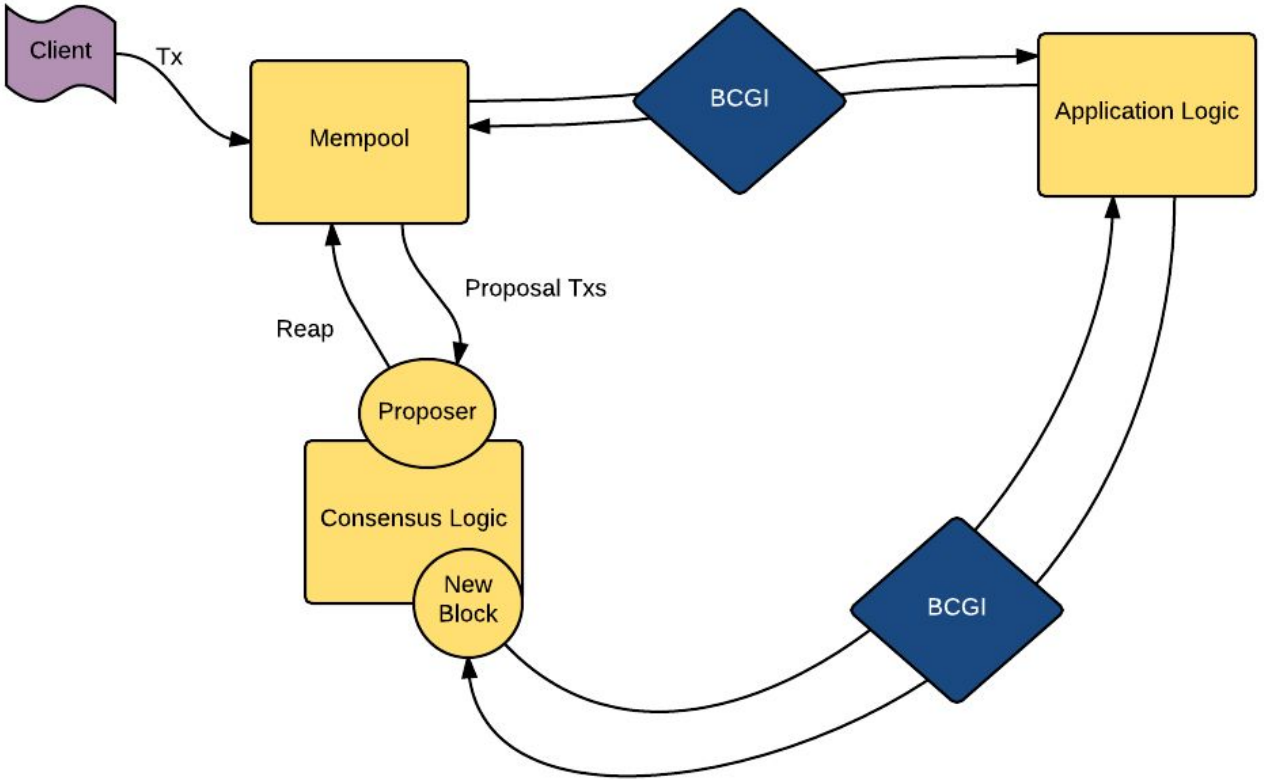
CGI

SOCKET PROTOCOLS

BCGI

SECURITY & NETWORKING PLATFORMS

Blockchain Gateway Interface



Outline

- Background
  - Consensus and Atomic Broadcast
  - Byzantine Fault Tolerance
  - Bitcoin
- Tendermint
  - Algorithm
  - Security
- Applications
  - Blockchain Application Logic
  - Blockchain Gateway Interface
- Experiments
  - Performance
  - Fault Tolerance
- Future Work

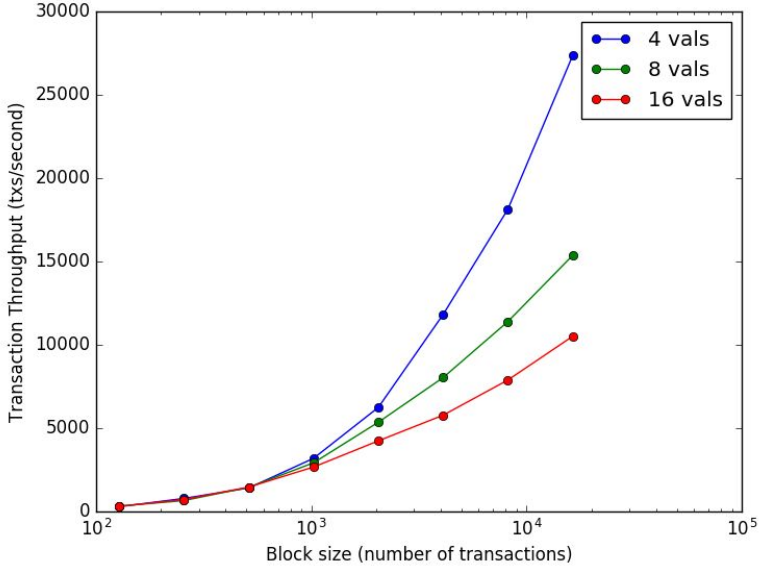
Tendermint Performance

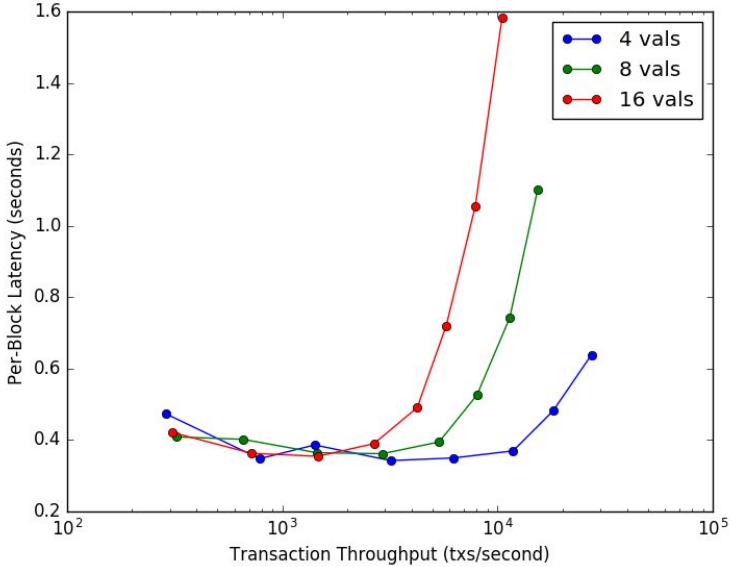
Experimental Methods

- Amazon EC2 instances
  - t2.medium: 2 vCPU, 4 GB RAM
  - c3.8xlarge: 32 vCPU, 60 GB RAM
- Single datacenter or multi datacenter
- Docker and docker-machine
- 250 byte transactions
- Network monitoring via websockets records events
- 2 to 64 validators

Tendermint Performance

1 datacenter (AWS - t2.medium)

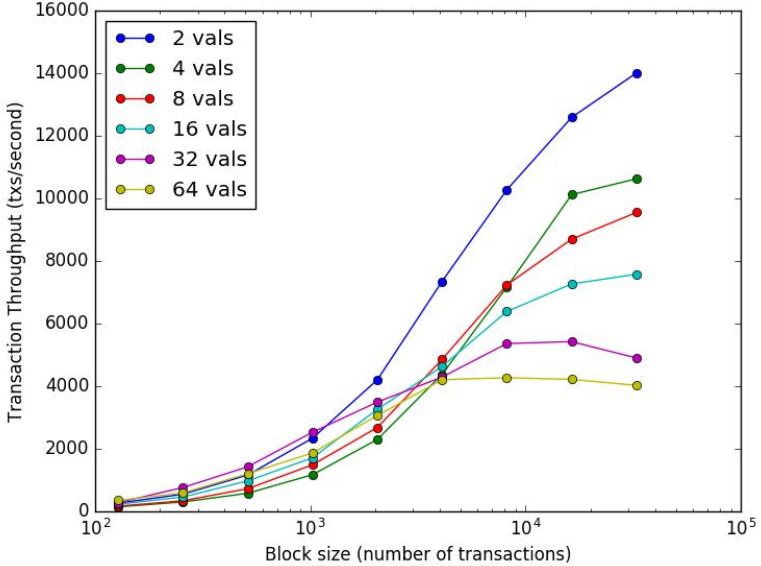


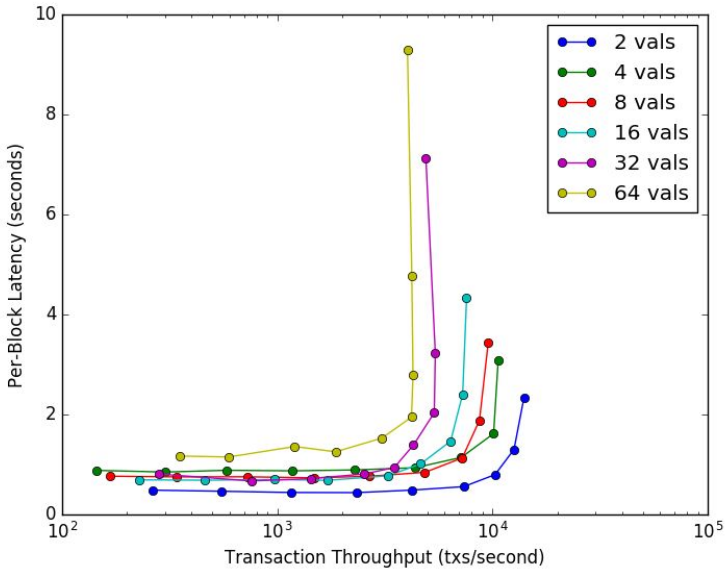


https://github.com/tendermint/network_testing

Tendermint Performance

7 datacenters on 5 continents (AWS - t2.medium)

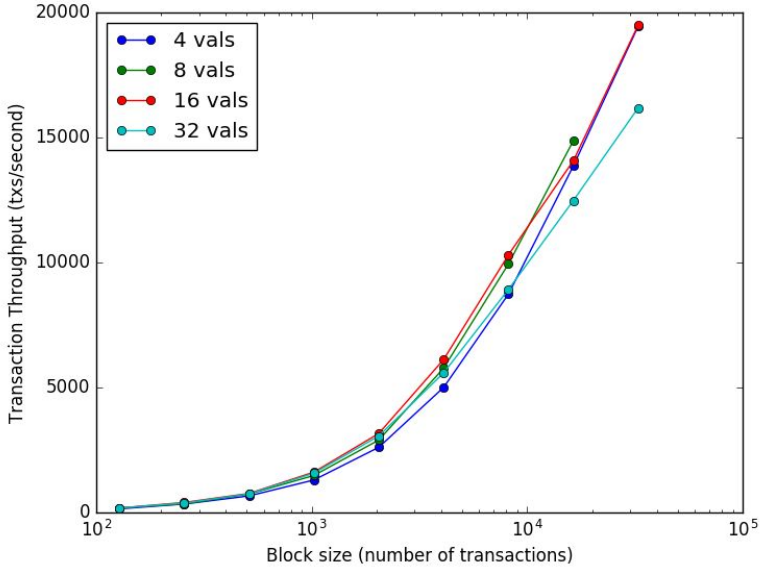


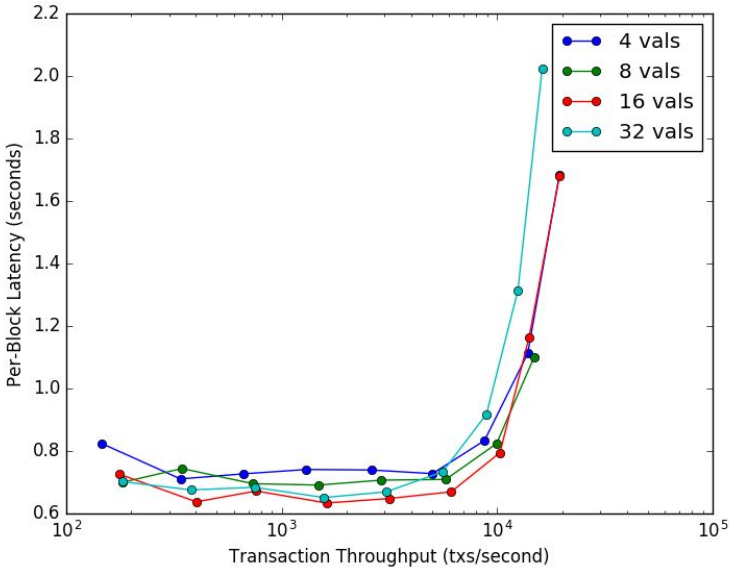


https://github.com/tendermint/network_testing

Tendermint Performance

7 datacenters on 5 continents (AWS - c3.8xlarge)





https://github.com/tendermint/network_testing

Tendermint Fault Tolerance

Experimental Methods

- N_fault = (N-1)/3

- Crash failures
  - random N_fault machines stopped for 3 seconds, restarted, every 3 seconds

- Network delay
  - random N_fault machines sleep for X ~ U(0,3000) ms before every read/write

- Byzantine faults
  - N_fault machines follow adjusted protocol:
    - Propose conflicting proposals to opposite halves of the network
    - Never vote nil
    - Prevote and precommit every proposal, right away

Tendermint Fault Tolerance (crash)

TimeoutPropose	Min	Max	Mean	Median	95^{th} % $-$ *ile*
500	434	15318	2179	1102	5575
1000	516	18149	2180	1046	5677
2000	473	15067	2044	1049	5479
3000	428	9964	2005	1096	5502

(a) 4 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % $-$ *ile*
500	618	126481	2679	990	5589
1000	570	9832	1763	962	5835
2000	594	8869	1658	968	5481
3000	535	10101	1633	959	5485

(b) 8 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % $-$ *ile*
500	782	21354	1977	1001	5930
1000	758	12659	1761	981	5642
2000	751	21285	2041	1005	6872
3000	719	72406	2395	991	5987

(c) 16 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % $-$ *ile*
500	760	24692	2591	1087	14025
1000	755	19696	2328	1119	9321
2000	852	21044	2178	1141	6514
3000	763	25587	2289	1119	6707

(d) 32 Validators

Tendermint Fault Tolerance (delay)

TimeoutPropose	Min	Max	Mean	Median	95^{th} % $-$ *ile*
1000	873	2796	1437	1036	2627
2000	831	4549	1843	1180	4036
3000	921	5782	2273	1251	5491
4000	967	6875	2700	1413	6781

(a) 4 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % $-$ *ile*
1000	870	2840	1449	1040	2786
2000	957	4268	1848	1076	4148
3000	859	5724	2156	1100	5649
4000	897	11859	3055	1093	11805

(b) 8 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % $-$ *ile*
1000	914	5595	1821	1135	5466
2000	950	7782	2490	1165	7650
3000	978	10305	3049	1163	9890
4000	1018	6890	2808	1174	6813

(c) 16 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % $-$ *ile*
1000	1202	8562	2219	1349	5733
2000	1196	7878	2549	1365	7579
3000	1164	10082	3003	1382	9805
4000	1223	17571	3696	1392	12014

(d) 32 Validators

Tendermint Fault Tolerance (byzantine)

TimeoutPropose	Min	Max	Mean	Median	95^{th} % − *ile*
1000	868	3888	1450	1086	3320
2000	929	4375	1786	1272	4166
3000	881	4363	1224	1099	1680
4000	824	8256	1693	1272	2607

(a) 4 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % − *ile*
1000	771	3445	1472	916	3288
2000	731	3661	1426	902	3339
3000	835	6402	1912	962	6155
4000	811	4462	1512	964	3592

(b) 8 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % − *ile*
1000	877	15930	2086	1024	5844
2000	808	5737	1580	1027	4155
3000	919	10533	1801	1110	4174
4000	915	5589	1745	1095	4181

(c) 16 Validators

TimeoutPropose	Min	Max	Mean	Median	95^{th} % − *ile*
1000	1594	11730	2680	1854	5016
2000	1496	17801	3430	1874	11730
3000	1504	15963	3280	1736	9569
4000	1490	24836	3940	1773	12866

(d) 32 Validators

Outline

- Background
  - Consensus and Atomic Broadcast
  - Byzantine Fault Tolerance
  - Bitcoin
- Tendermint
  - Algorithm
  - Security
- Applications
  - Blockchain Application Logic
  - Blockchain Gateway Interface
- Experiments
  - Performance
  - Fault Tolerance
- Future Work

Future Work

- Software
  - More testing!
    - Additional byzantine faults
    - Network partitions (https://aphyr.com/tags/jepsen)
  - Optimizations in block and vote propagation
  - Enforce locking rules during normal execution
  - Many, many, applications!

- Theory
  - Expand formalization to include more components
  - Validate formalization using model checkers
  - More comprehensive framework for Byzantine and Economic consensus

- Production
  - Gnuclear: scalable public tendermint blockchain

Appendix

Code!

Core
- github.com/tendermint/tendermint
- github.com/tendermint/tmsp

Tools
- github.com/tendermint/mintnet
- github.com/tendermint/netmon
- github.com/tendermint/network_testing

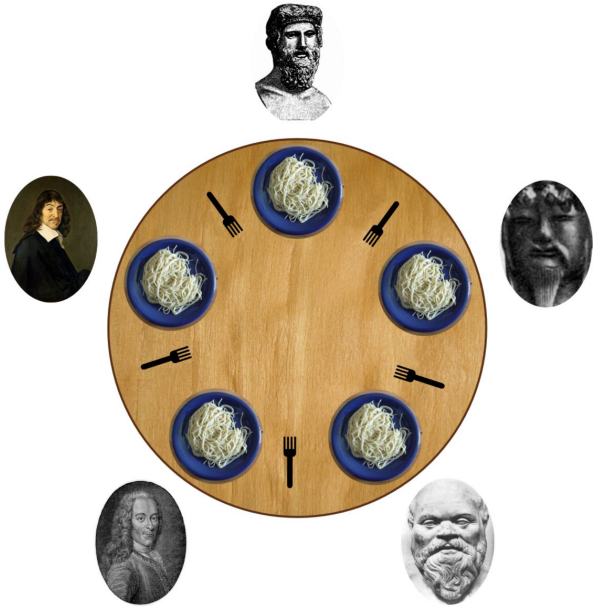
Apps
- github.com/tendermint/merkleeyes
- github.com/tendermint/governmint
- github.com/tendermint/basecoin
- github.com/eris-ltd/geth-tmsp
- github.com/eris-ltd/eris-db

Docs
- github.com/tendermint/tendermint/wiki
- github.com/ebuchman/thesis

Libraries
- github.com/tendermint/go-wire
- github.com/tendermint/go-rpc
- github.com/tendermint/go-p2p
- github.com/tendermint/go-merkle

Distributed Computing

What do *cigarette smokers*, *dining philosophers*, and a *couple of generals* have in common?



Consensus

- Termination - every correct process eventually decides
- Integrity - every correct process decides at most once
- Agreement - if one correct process decides *v1* and another decides *v2*, then *v1* = *v2*
- Validity - if a correct process decides *v*, at least one process proposed *v*

Atomic Broadcast

- Validity - if a correct process broadcasts *m*, it eventually delivers *m*
- Agreement - if a correct process delivers *m*, all correct processes eventually deliver *m*
- Integrity - *m* is delivered only once, and only if broadcast by its sender
- Total Order - if correct processes *p* and *q* deliver *m* and *m'*, then *p* delivers *m* before *m'* iff *q* delivers *m* before *m'*

} Reliable Broadcast

Synchrony vs Asynchrony

VS

Synchrony vs Asynchrony

Common Coin (or, that random trick)

Common Coin (or, that random trick)

Another Advantage of Free Choice:
Completely Asynchronous Agreement Protocols
(Extended Abstract)

Michael Ben-Or †

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

RANDOMIZED BYZANTINE GENERALS

Michael O. Rabin*

Hebrew University and Harvard University

Blockchain Gateway Interface