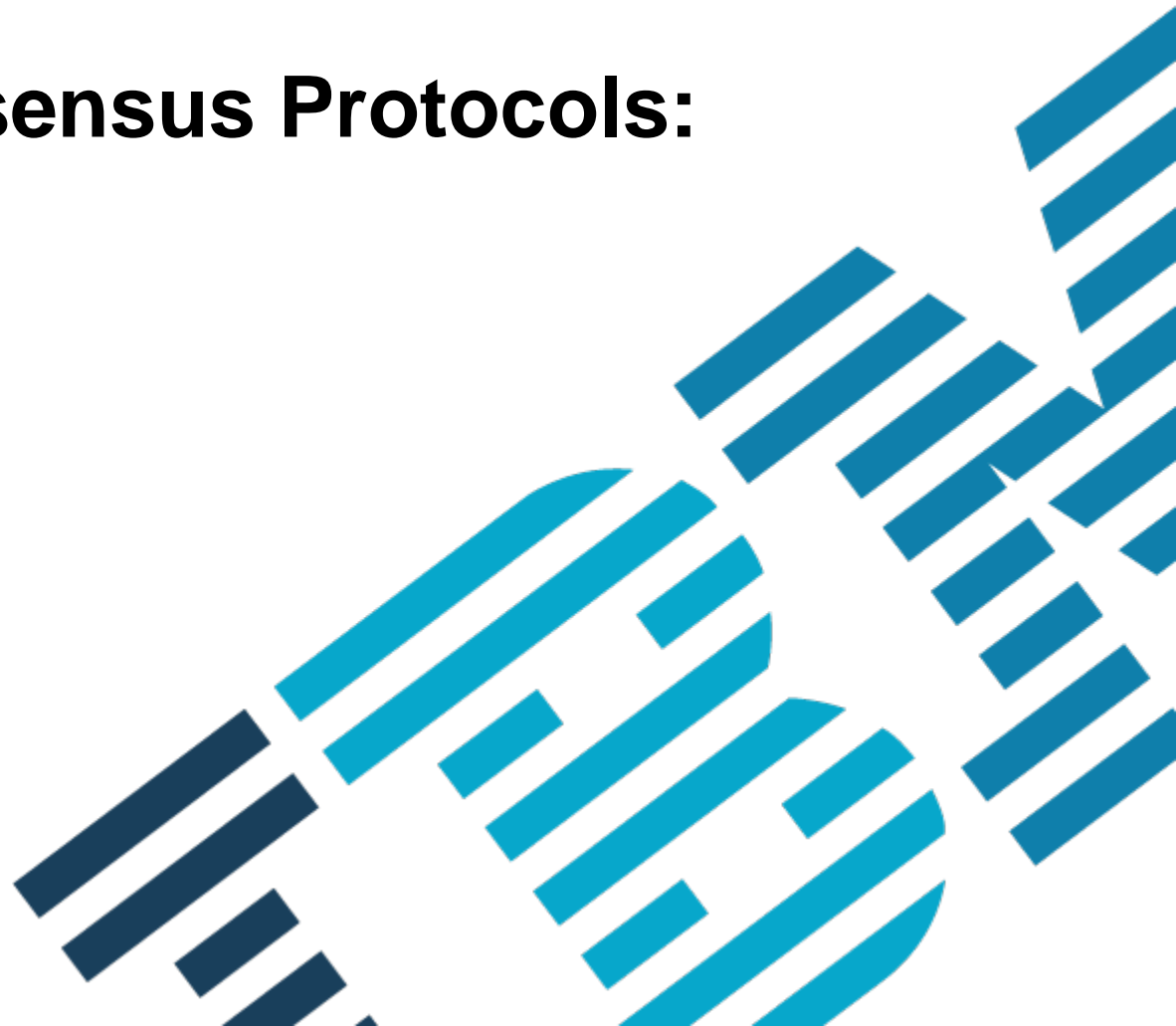


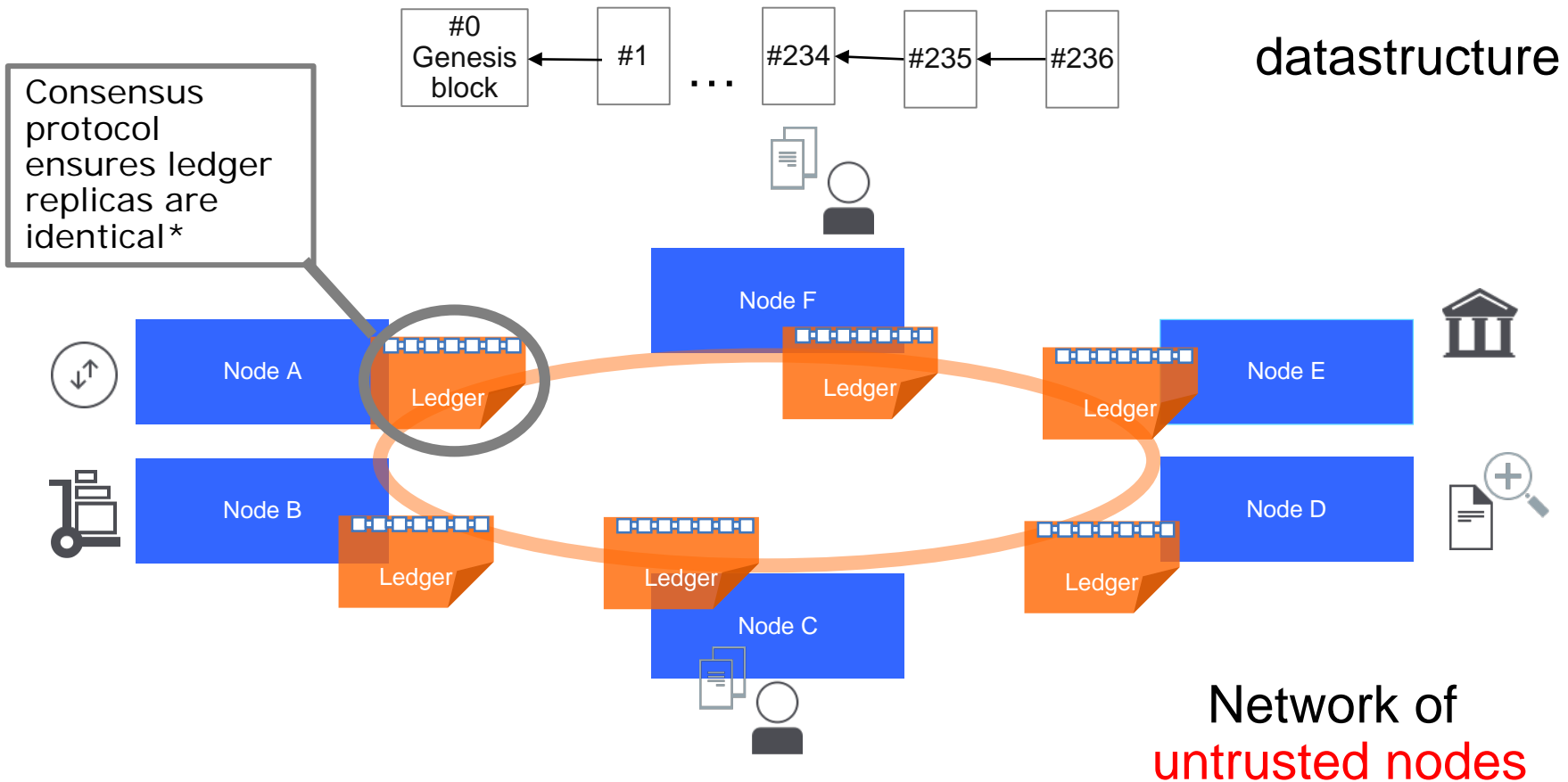
Blockchain Consensus Protocols: an Outlook

IFIP WG 10.4 meeting
Clervaux, Luxembourg



What is a Blockchain?

- **A chain (sequence, typically a hash chain) of blocks of transactions**
 - Each block consists of a number of (ordered) transactions
 - Blockchain establishes total order of transactions



Blockchain evolution (2009-present)

2009
Bitcoin



- **A hard-coded cryptocurrency application w. limited stack-based scripting language**
- **Proof-of-work-consensus**
- **Native cryptocurrency (BTC)**
- **Permissionless blockchain system**

Blockchain 1.0

2014
Ethereum



- **Distributed applications (smart contracts) in a domain-specific language (Solidity)**
- **Proof-of-work-consensus (transition to Proof of Stake?)**
- **Native cryptocurrency (ETH)**
- **Permissionless blockchain system**

Blockchain 2.0

2017
Hyperledger
Fabric



- **Distributed applications (chaincodes) in different general-purpose languages (e.g., golang, Java, Node)**
- **Modular/pluggable consensus**
- **No native cryptocurrency**
- **Multiple instances/deployments**
- **Permissioned blockchain system**

Blockchain 3.0

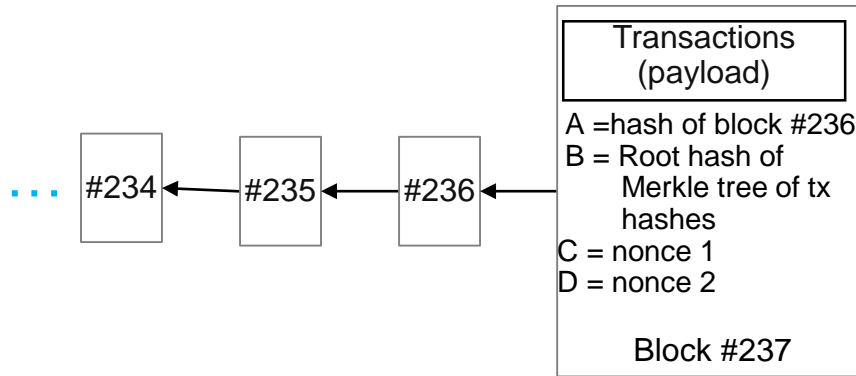
Consensus: growing the chain

- **How does the chain grow?**

Most popular blockchain technique (used also in Bitcoin):

Proof-of-Work (PoW)

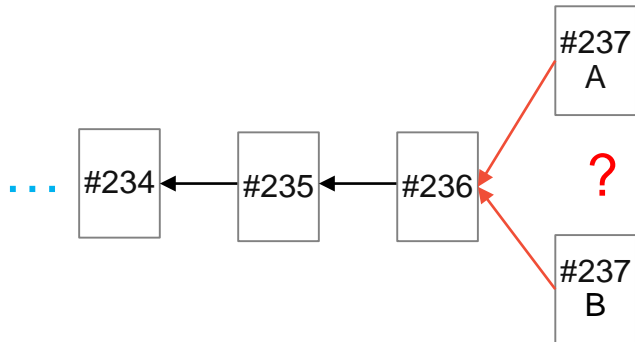
Growing Proof-of-Work (PoW)-based Blockchain



$$h = \text{hash of Block \#237} = \text{SHA256}(A||B||C||D)$$

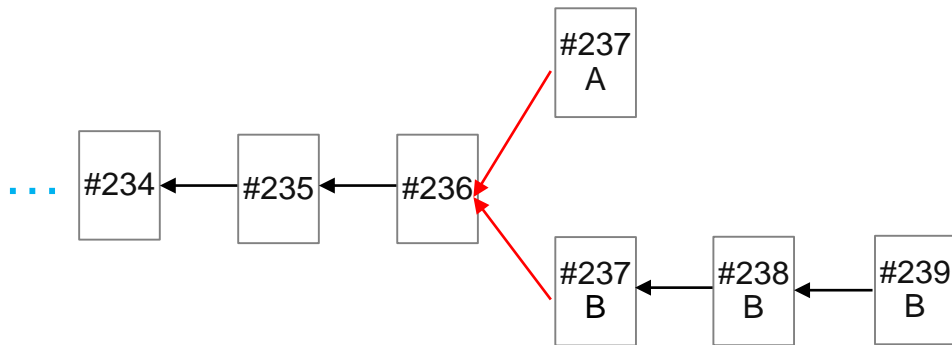
- Block “mining”:
 - Every participant (“miner”) tries to find nonces
 - such that the hash of the block h **is lower than a 256-bit target**
- Bitcoin
 - Target dynamically adjusted every 2016 blocks
 - 1 block generated roughly every 10 minutes
 - This currently requires roughly 2^{80} expected hashes per block

Forks

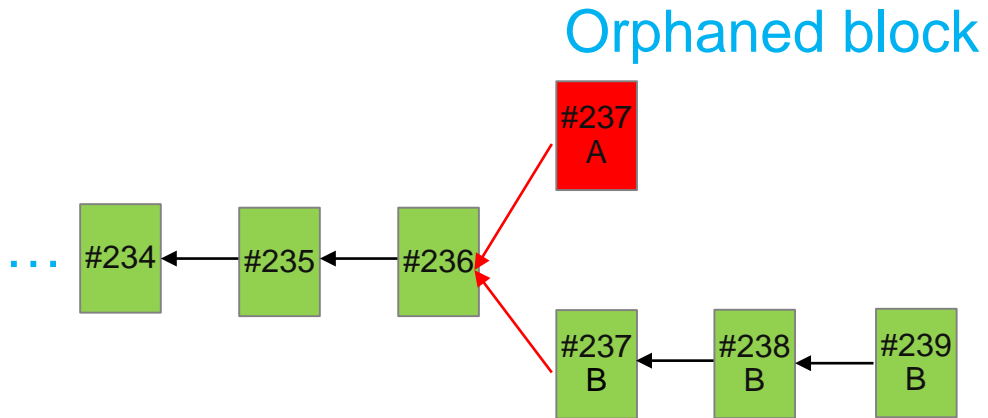


- If multiple miners mine the next block, consensus (on the next block) might be broken
PoW acts as an unreliable concurrency control mechanism – it may fail in this
- Hence, Bitcoin miners adopt a **conflict resolution policy**
 - They will temporarily store both 237A and 237B
 - A fork being extended longer (in fact with more work) eventually prevails

Example (longest/most difficult chain wins)



Example (longest/most difficult chain wins)



Implications and the performance issues

PoW way of extending the ledger heavily and negatively impacts system scalability and overall throughput

- Bitcoin: With 1 block every 10 minutes and fixed block size of 1 MB
 - Peak throughput: **only 6-7 tx/sec**
 - Latency (of 6 block confirmations): **about 1h**
 - **Enormous energy consumption!**
- <https://digiconomist.net/bitcoin-energy-consumption>
 - 71 TWh/year → 8GW of power
 - More than Switzerland, 0.32% of world electricity consumption
 - 987 kWh per transaction!
 - Average US household in 2016 → 897 kWh per month

Better performance by tuning PoW parameters?

- **Limited benefits, potentially weaker security**
 - shorter block generation times (increasing block frequency)?
 - larger blocks?
 - Different conflict resolution rules?
- From Gervais et al. CCS'16 paper <https://eprint.iacr.org/2016/555>

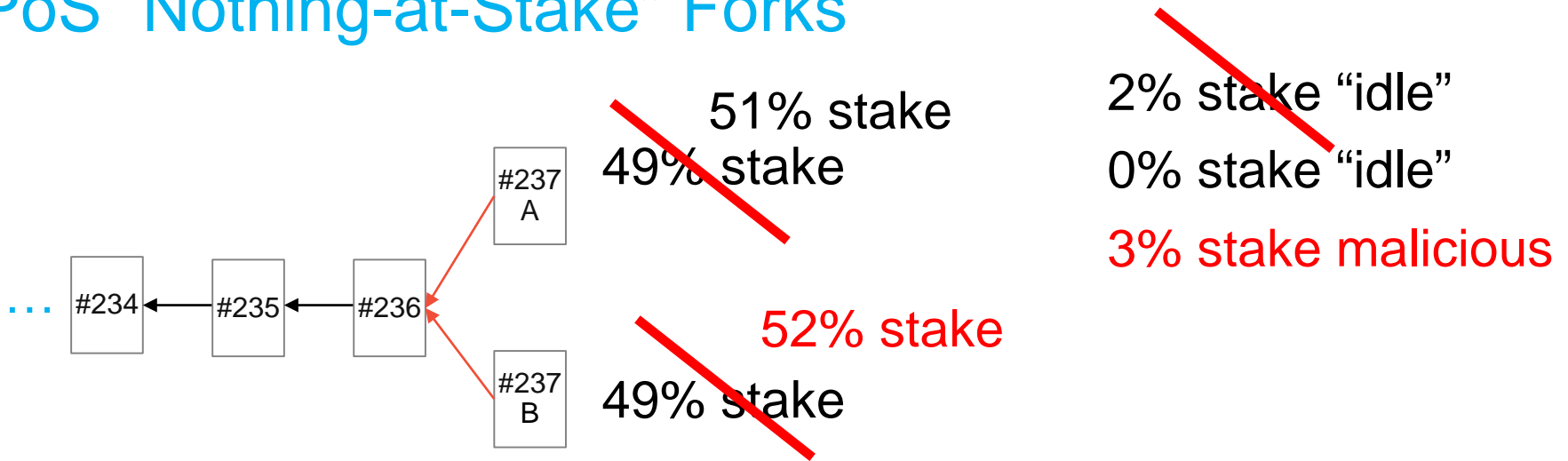
	Bitcoin	Litecoin	Dogecoin	Ethereum
Block interval	10 min	2.5 min	1 min	10-20 seconds
Public nodes	6000	800	600	4000 [11]
Mining pools	16	12	12	13
t_{MBP}	8.7 s [8]	1.02 s	0.85 s	0.5 - 0.75 s [12]
τ_s	0.41%	0.273%	0.619%	6.8%
s_B	534.8KB	6.11KB	8KB	1.5KB

- Bitcoin 6 blocks (1hour) ~ Ethereum 37 blocks (9-10 minutes)
- PoW blockchains can attain up to 60 tps with Bitcoin-like probability of stale blocks

Boosting consensus: Enter Proof of Stake (PoS)

- PoS usually sits on top of PoW tree datastructure
- Allows nodes with more stake/weight to form blocks more often effectively lowering the difficulty
- “Nothing at stake” problem?
- Centralization?

PoS “Nothing-at-Stake” Forks



- PoS breaks ties selecting forks (branches) with more stake on them
- Very susceptible to “double-spend” attacks in absence of penalties
- Example with 3% of stake double spending

Casper – Friendly Finality Gadget

- Ethereum PoS
- Buterin/Griffith
 - <https://arxiv.org/abs/1710.09437>

- Leverages BFT techniques to limit the effects of forks and to address nothing at stake problem
 - Byzantine Fault Tolerant (BFT) agreement to settle on a single block and penalize equivocating nodes

- Relies on node synchrony as well

- Announced as early as in 2015, still being figured out...

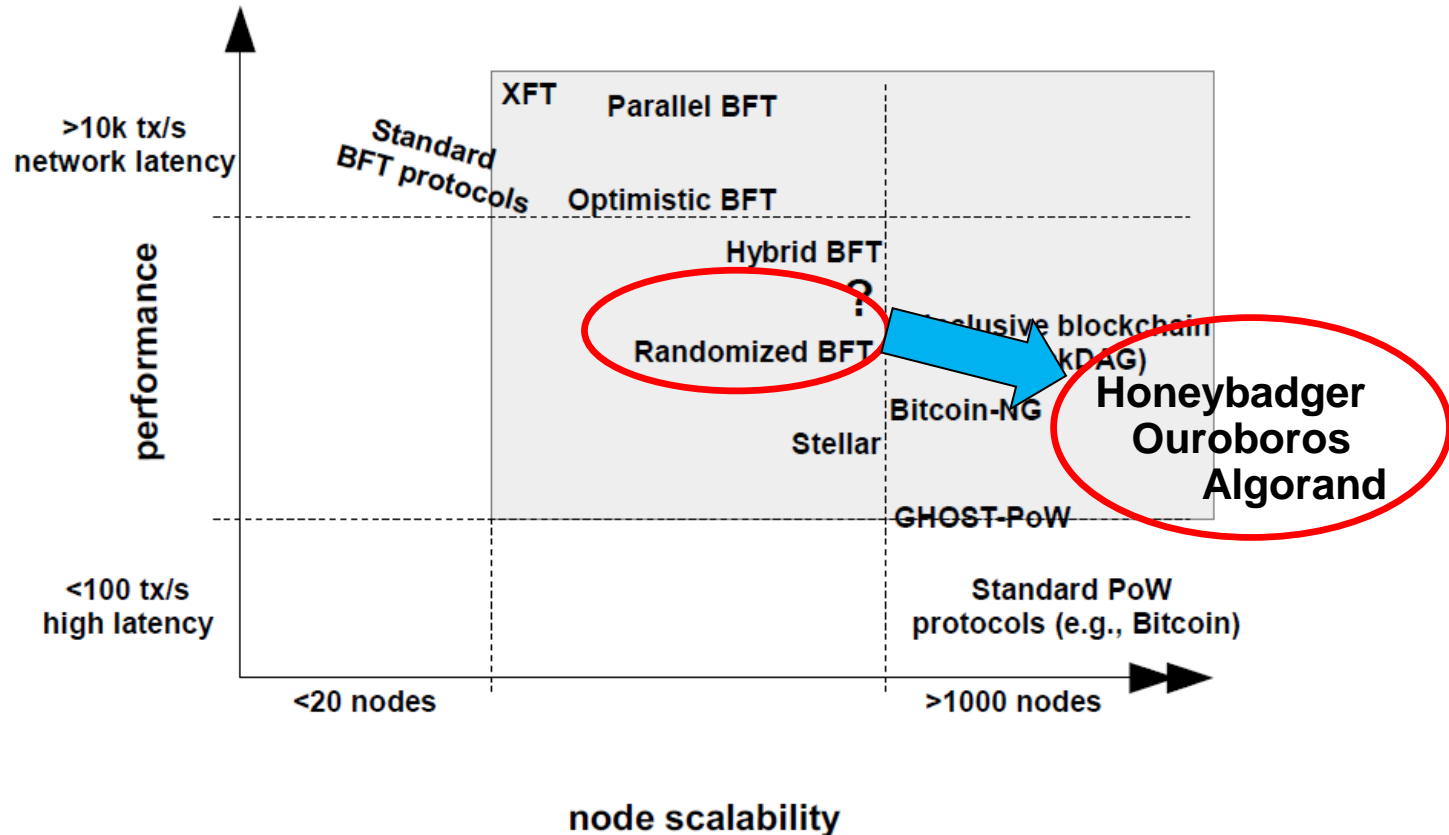
BFT in Blockchains

- BFT is known to matter for **permissioned** blockchains
- With PoS BFT importance extends also to **permissionless** blockchains
 - All PoS protocols resort in one way or another to BFT

PoW vs. BFT for Blockchain (simplified overview)

	Proof of Work (Bitcoin, Ethereum,...)	BFT state machine replication (Ripple, Stellar, Fabric,...)
Membership type	Permissionless	Permissioned
User IDs (Sybil attack)	Decentralized, Anonymous (Decentralized protection by PoW compute/hash power)	Centralized, all Nodes know all other Nodes (Centralized identity management or stake protect against Sybil attacks)
Scalability (no. of Nodes)	Excellent, >100k Nodes	Verified up to few tens (or so) Nodes Can scale to 100 nodes with certain performance degradation (scalability limits not well explored)
Scalability (no. of Clients)	Excellent	Excellent
Latency	Poor, up to 1h (Bitcoin) From 9-10 mins (Ethereum)	Depends on the implementation/deployment (order of ms)
Peak Throughput	from 7 tx/sec (Bitcoin)	>10k tx/sec with existing implem. in software [<20...100 nodes]
Power efficiency	>8 GW (Bitcoin)	Good (commodity hardware)
Temporary forks in blockchain	Possible (leads to double-spending attacks)	Not possible

Challenge #4: Consensus modularity/pluggability



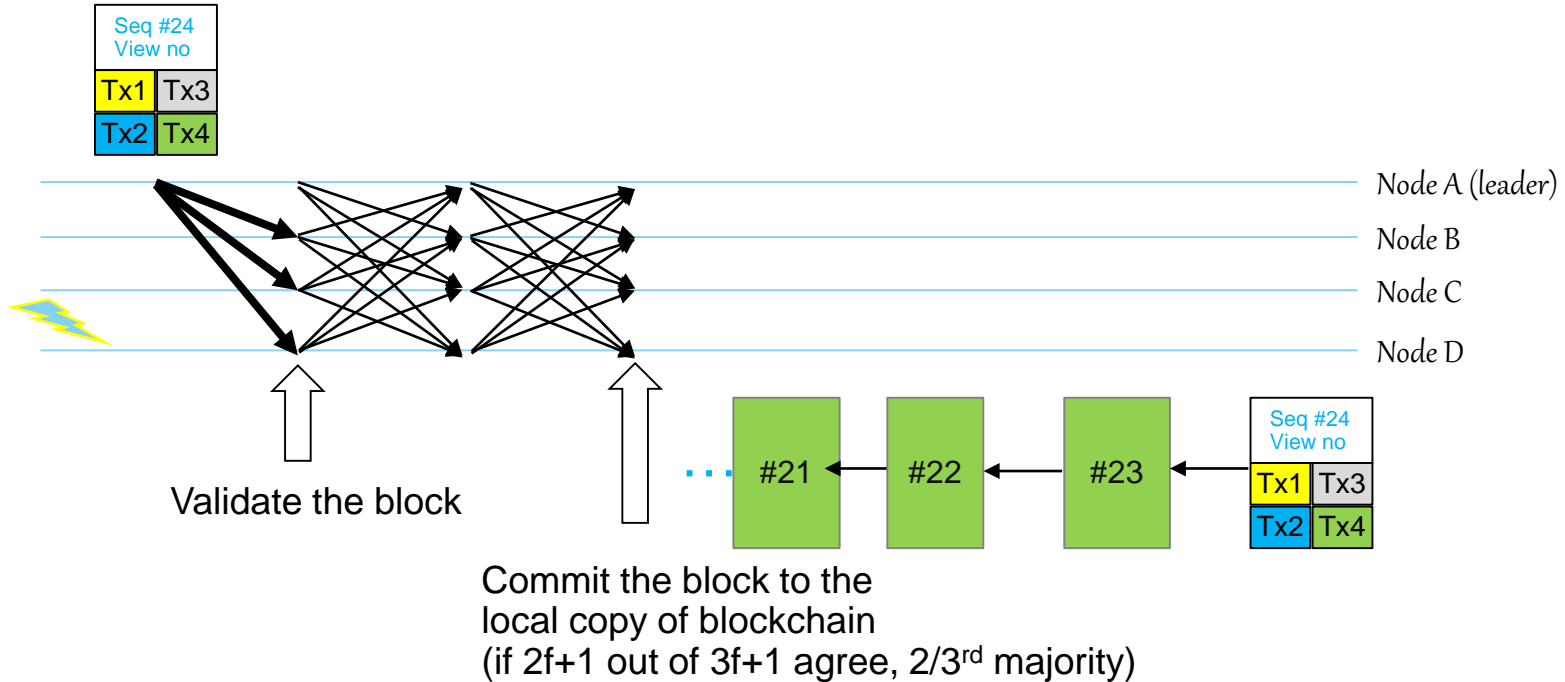
Open research problem:

Given the use case, network, no. of nodes
 What is the most suitable Blockchain consensus?

Outstanding challenges in BFT for blockchains?

- **Maximizing throughput on WANs**
 - Retaining acceptable latency, performance in clusters
- **Scaling to 100+ nodes, without sacrificing performance**
 - Supporting dynamic node reconfiguration
- **Robust but understandable protocols**
- **Scalable incentives**
 - Transaction fee payment on BFT that does not immediately become a bottleneck
- **Simplicity, provability and testing**

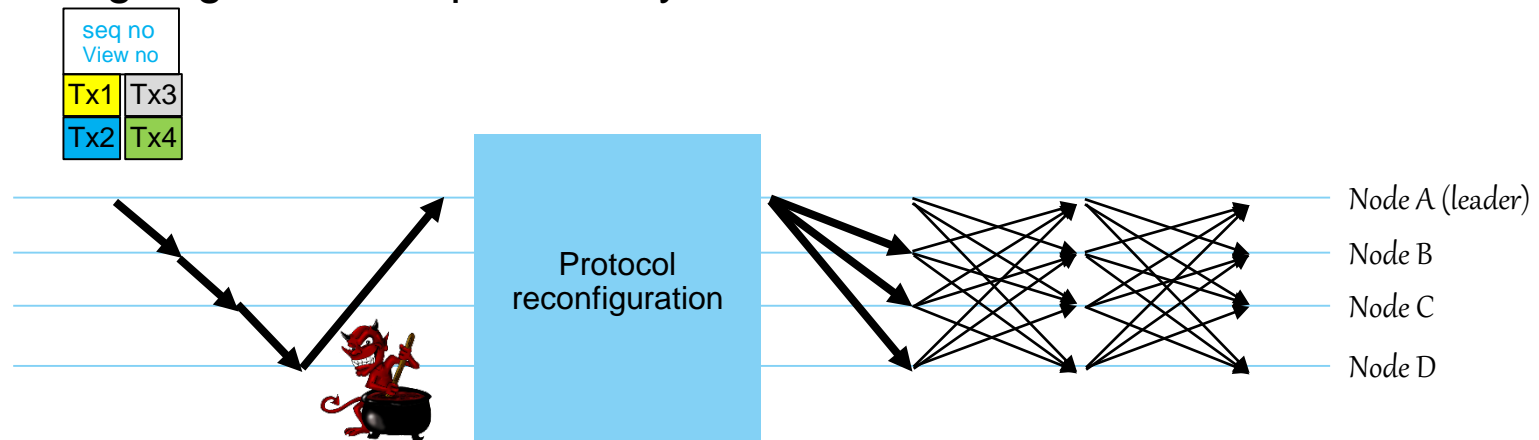
BFT Consensus 101 (example of PBFT [TOCS2002], implemented in Hyperledger Fabric v0.6)



In this example, all nodes have equal weights, the protocol can simply be adapted to weights/stakes

Maximizing throughput in clusters is fairly known

- LCR Crash fault tolerant protocol [Guerraoui et al, TOCS 2010]
 - Ring-based protocol
 - Throughput effectively limited only by replicas' NIC bandwidth
- The next 700 BFT protocols [Aublin et al, TOCS 2015]
 - Can run $O(n)$ BFT protocol of a basically arbitrary communication pattern including (a very load-balanced one) in the optimistic case
 - Backed by any BFT protocol (e.g., PBFT) to cover the worst case without redesigning the entire protocol/system



Maximizing throughput in WANs


- **Need to address the bottleneck at the leader**
- **Ring topology from clusters will not work in WANs**
 - Linear latencies may be too much
- **Alternatives?**
 - Gossip, tree-multicast, leaderless,...
- **Pipelined message patterns should be a norm**

Going to 100+ nodes [and reconfiguring them]

- **In 40 years of research we tried [unsuccessfully] to sell $n=4$...**
- **Now we suddenly need BFT with $n\sim 100$... $n\sim 1000$... $n\sim 1000000$**
 - That scale was not even tested for CFT
- **What can we do? (except sharding)**

Revisiting the assumptions

- XFT [Liu et al., OSDI 2016] <http://arxiv.org/abs/1502.05831>
- BFT assumes powerful adversary
 - controlling the network among correct nodes
 - and f Byzantine nodes out of $3f+1$ nodes
 - Simultaneous control over network and Byzantine nodes may be difficult to pull out in the blockchain setting
- XFT: at most f of partitioned nodes and Byzantine nodes at any time
 - Have the cost of CFT consensus without trusted hardware

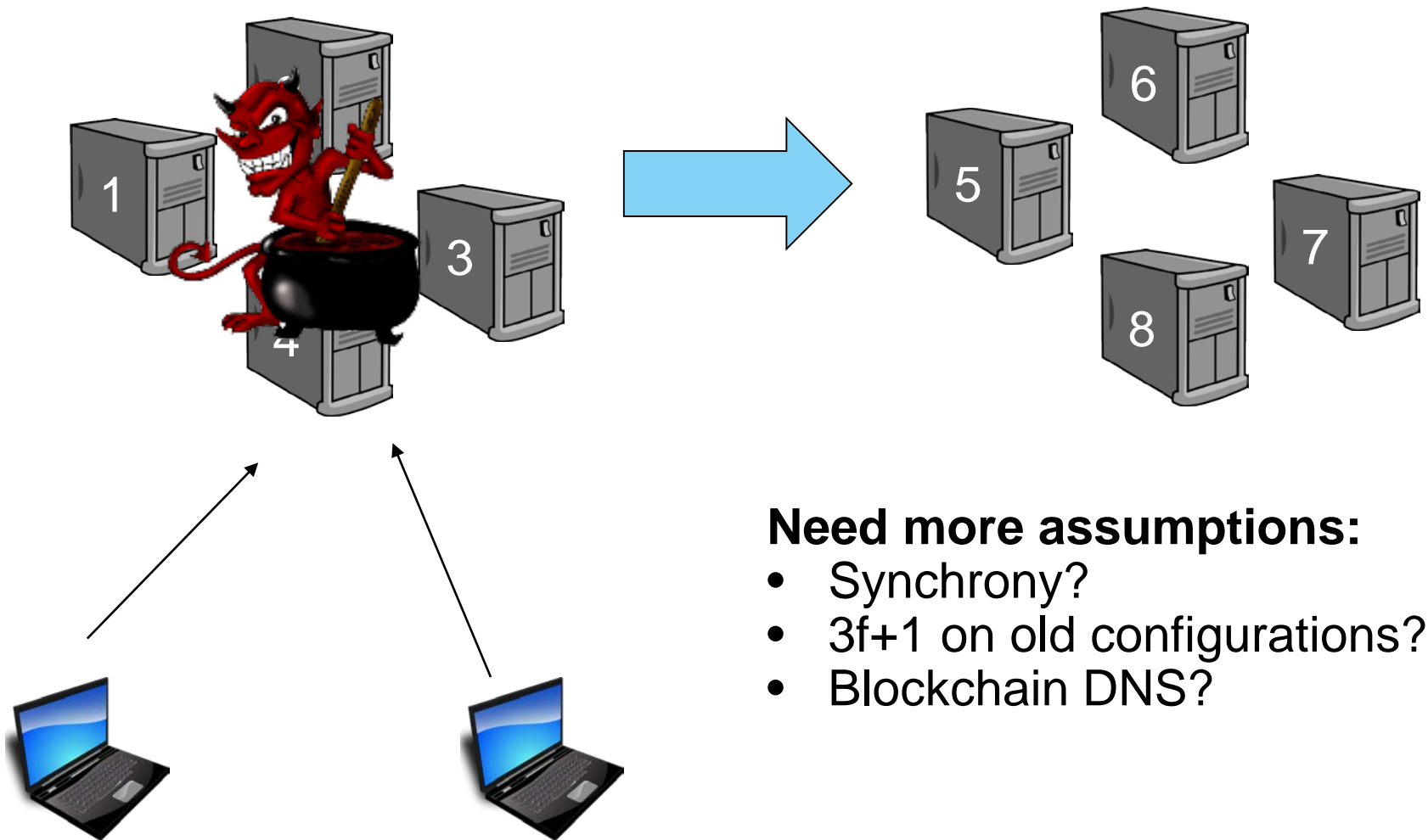
SMR model	CFT	XFT 	BFT
Number of Nodes	$2f+1$	$2f+1$	$3f+1$
Tolerating Byzantine Nodes	no	yes	yes
Performance	Good	Practically as good as CFT	Poor (compared to CFT)

- Most blockchains assume some sort of synchrony

Committees, subcommittees

- **Algorand [SOSP 2017]**
 - Claims to run on 500 000 nodes
 - Actually relies on PBFT-variant among a much smaller subcommittee ($n < 100$)
- **What are the limits on natively running BFT across all nodes?**
- **At which point probabilistic protocols outperform deterministic ones?**
 - And vice-versa

Reconfiguration is an issue



Need more assumptions:

- Synchrony?
- $3f+1$ on old configurations?
- Blockchain DNS?

We need robustness – but we need it understandable

Properties under faults, bad conditions are critical

- Prime, Spinning, Aardvark, RBFT, RAliph, have been around for some time now...
- But how accessible are these techniques to blockchain backend programmers?
- Good example is Spinning rotating leader paradigm
 - Simple, yet effective for a number of performance attacks
 - Unfortunately does not solve all issues

Understandable, testable code

- **SBFT implementation (PBFT variant written for Hyperledger Fabric)**
 - 1000 lines of Go code to implement OSDI/TOCS PBFT
- **Deterministic testing frameworks**
 - Every new feature/CR needs to come with a distributed test
- **Formal methods, model checking are great**
 - but there is a gap wrt reality and code that actually runs

It is not about ordering only

- **Applications can easily become a bottleneck**
- **In Hyperledger Fabric application bottleneck is currently at less than 10000tps**
- **We need scalable mechanisms for any per-transaction processing that BFT consensus is doing**

Near-term “Holy Grail” of blockchain scalability

Can we have a blockchain protocol
(single shard? :)

Scaling to hundreds of nodes

Sustaining VISA-like performance numbers?

(seconds latency, about 5k tps on average, few 10k tps peak throughput)

An even “Holier Grail”:

**can we do this with some notion of
transaction confidentiality?**

Ultimate “Holy Grail” of blockchain scalability

Can we have a blockchain protocol

Scaling to hundred(s) of nodes on WANs

With network bounded throughput

And network/speed of light bounded latency?

An even “More ultimate Holier Grail”:

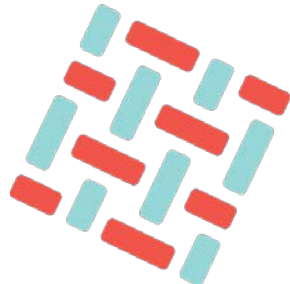
can we do this with some notion of

transaction confidentiality?

Thank you

You have the new best consensus protocol?

Consider integrating it in



HYPERLEDGER
FABRIC

Thank You!