BlockFin

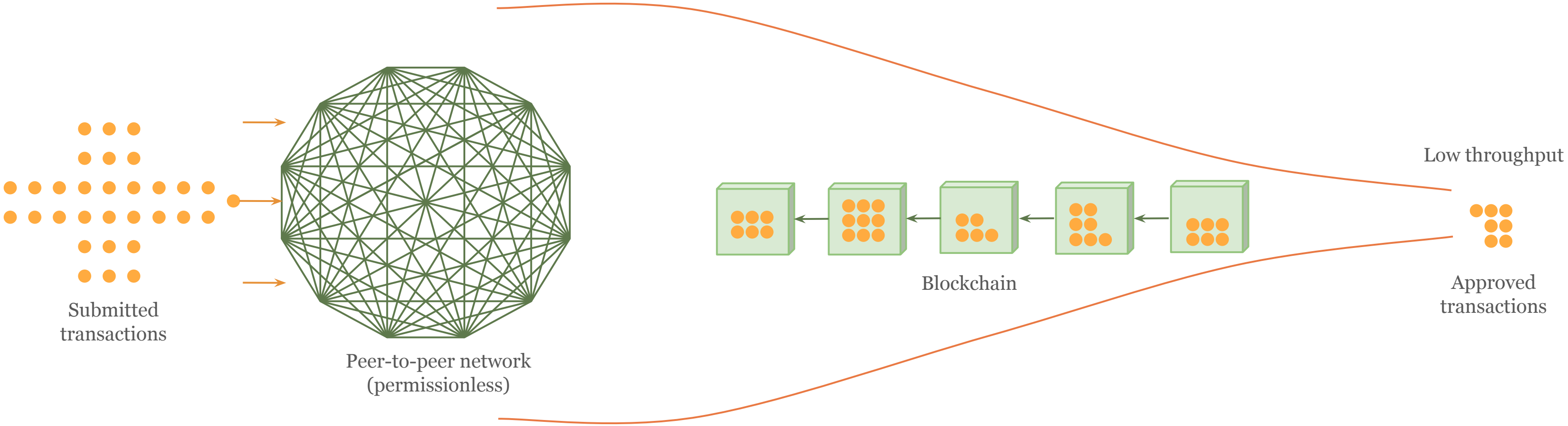
A Fork-Tolerant, Leaderless Consensus Protocol

April 2018

@storecoin

What are the most desirable features in a blockchain?

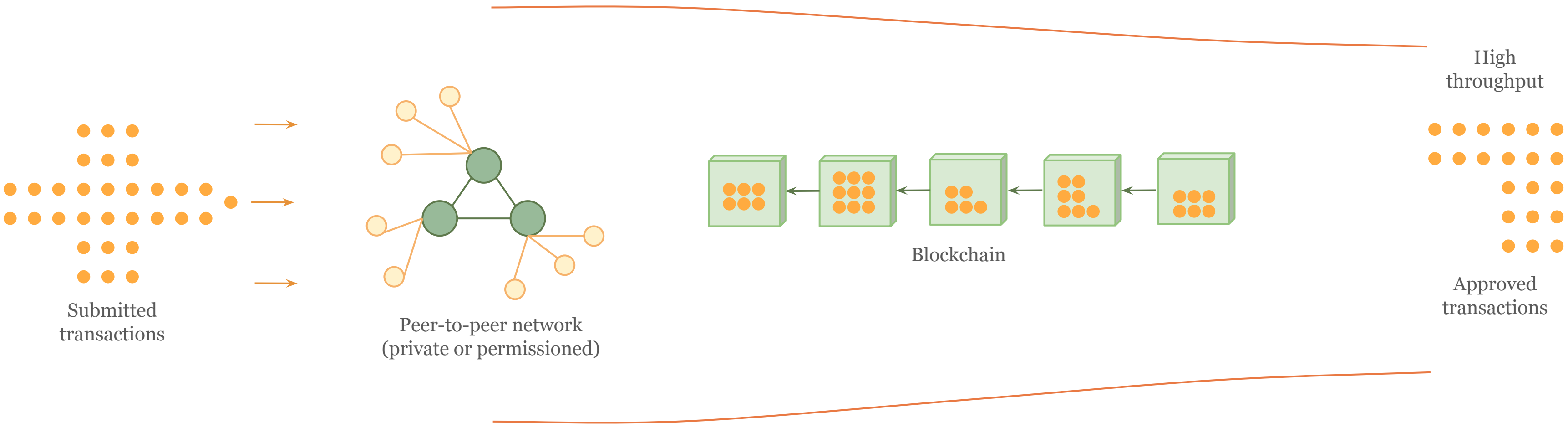
Scalability (throughput) and *decentralization* (censorship resistance), but they are at odds with each other



Highly decentralized (Bitcoin, Ethereum, etc.) systems typically have low throughput

What are the most desirable features in a blockchain?

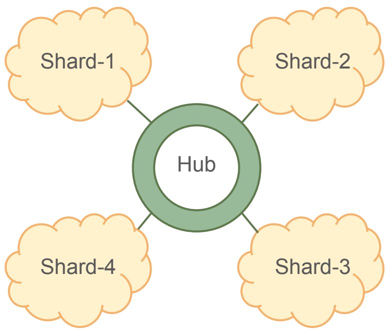
Scalability (throughput) and *decentralization* (censorship resistance), but they are at odds with each other



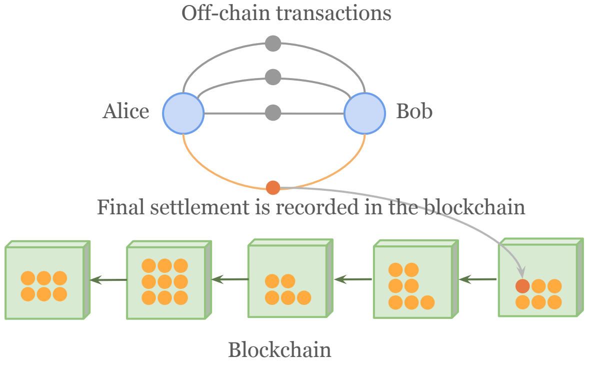
Private (Ripple) or permissioned (EOS) networks have high throughput, but poor decentralization

Common approaches to address scalability

Sharding and *off-chain transactions* are commonly employed, but they bring their own challenges



Cross-shard communication
and settlement are expensive



No visibility or record of
private transactions

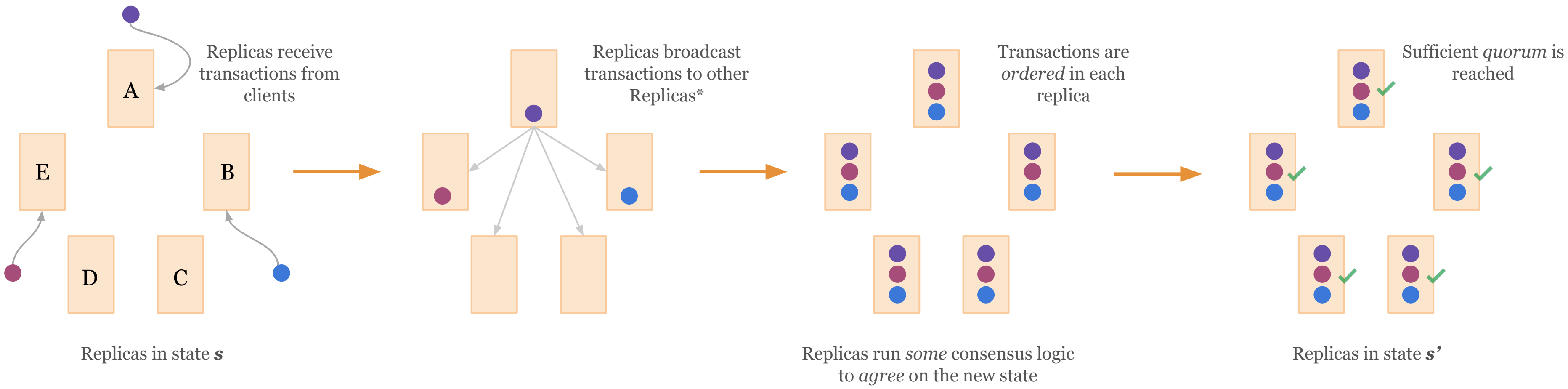
Cross-shard communication and shard-rebalancing are expensive. Off-chain transactions require additional setup and they have their own lifecycle overheads

Simple *on-chain* solution is desirable

Highly *scalable* and highly *decentralized* - tie the odd ends together

Why typical consensus process is slow?

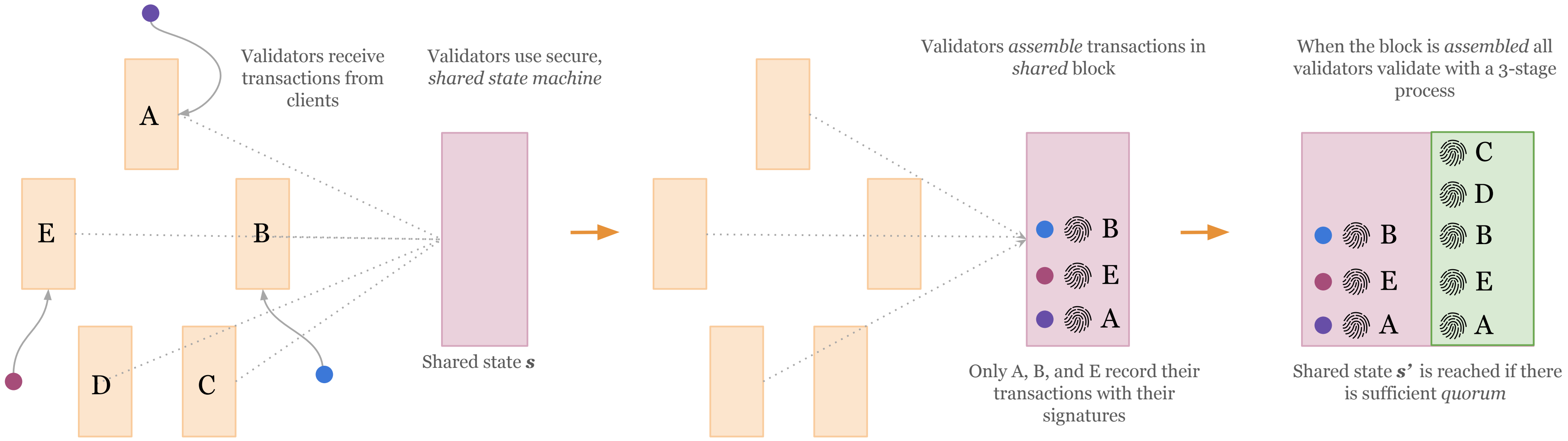
All replicas (validators) maintain their *private* states *locally*. The replicas *broadcast* their states to their peers to *reach* consensus. The larger the network, the slower the consensus process becomes



* For brevity, only the broadcast from replica A is shown here. Replicas B and E work similarly

BlockFin consensus engine

Leaderless, fork-tolerant, highly scalable, and *highly decentralized*



Leaderless — All validators participate in assembling and validating blocks
Fork-tolerant — Account-based transaction model avoids double-spend attack, thus allowing parallel block assembly and validation
Highly scalable — Two-tier peer-to-peer network provisions *shared state machine*, which minimizes communication overhead. Along with fork-tolerance, this results in high throughput
Highly decentralized — All validators participate in assembling and validating blocks, making it truly decentralized
No-fee transactions — The transaction fee is paid for by annual inflation, making transactions free for merchants and developers
Community block rewards — The block rewards are shared by all validators encouraging participation and up-time

BlockFin - *Implicit* Consensus and Finality

- Implicit consensus
  - BlockFin protocol doesn't have *explicit markers* for the completion of pre-commit, commit, and seal stages. This is because doing so requires leader election and timing assumptions
  - The validators *never collectively agree* on the block state because of the impossibility of doing so in large networks
  - *Implicit decision* involves protocol-following validators *decide* on the state the block depending on signatures collected. For example, when validators look at a block that has ⅔ + *valid* pre-commit signatures they *agree individually* that the block has completed pre-commit stage
  - By the property, "***Agreement*** - If an honest node proposes a value v, then all honest nodes agree with v", we can conclude that if an honest validator agrees on the block state, then all honest validators agree with the same state
  - Implicit decisions lead to *implicit consensus*. This is powerful because even clients can choose to follow the BlockFin protocol and *implicitly agree* on the block state and hence the transactions included in it without relying on a leader or delegates to make such decisions

- Implicit finality
  - Implicit consensus leads to implicit finality for both the blocks as well transactions included in them
  - A *committed* block is *finalized* in that it is *irreversible*. Any attempt to reverse or alter it by an adversary fails the *agreement* property described above
  - There is no waiting for N confirmed blocks (for example, 6 confirmed blocks in Bitcoin) before a block is *finalized*. As soon as ⅔ + *valid* commit signatures are collected for the block, the block is automatically finalized
  - Due to the asynchronous nature in which validators sign various stages of block validation, no timing guarantees are provided. Since multiple blocks are assembled and validated in parallel, sub-second block confirmation times are realistic when at least ⅔ + validators are online
  - By extension, the transactions in a finalized block are also *automatically* finalized. The individual transactions may be *approved* or *declined* specifically, but they are finalized nevertheless

BlockFin - *Explicit* Finality with Sealing

- Block sealing provides explicit finality
  - Block sealing is a slower, asynchronous, and batch-oriented stage, so its completion time is unpredictable
  - Until committed blocks are sealed, implicit consensus and finality are used by validators and clients alike. This calls for extra work (counting *valid* signatures), but only until the blocks are sealed
  - After the blocks are *sealed*, block and transaction finality are explicit
  - Whether explicit or implicit finality is used, the blocks are *irreversible* as soon as they are *committed*. The explicit and implicit finalities are just different ways to make the same decision

BlockFin consensus vs others in the wild

BlockFin Consensus	Other Consensus Protocols
Replicated State Machine: • Uses a two-tier P2P network to maintain *shared state* for participating replicas • Each replica *signs* its entry in shared blocks to identify itself and its decisions • The shared state is maintained in a *replicated* P2P datastore for resiliency and throughput • Replicas read from and write to shared state in P2P datastore • Replicas make shared decisions based on the shared state	**Replicated State Machine:** • Each replica maintains its own *local state* • Replicas use broadcast/gossip message primitives to share their local states with other replicas • Replicas make *local decisions* based on their local state
Communication Complexity: • Best case scenario is estimated to be $(4N + 2m)$ *per block* where N is the number of replicas and m, the number of transactions in the block • There are 3 primary stages in BlockFin validation -- a) add transaction to the block, b) pre-commit, and c) commit. Adding each transaction involves a read and a write, resulting in $2m$ operations. Pre-commit and commit are block level operations and each stage involves a read and write operation per replica, resulting in $4N$ operations • Worst case scenario requires *several* reads, followed by a write for each stage. This is not modeled yet, but still will be orders of magnitude lower than N^2 • Only the replicas that receive transactions add transactions to the block, so there is no unnecessary data duplication and resulting communication overheads among all replicas. This is the primary reason for the low complexity in BlockFin	**Communication Complexity:** • Usually N^2 where N is the number of replicas • Some implementations optimize this number for non-adversary scenarios or optimize the message *size* for speed • As N grows, so would the complexity. In practical Byzantine setup, consensus may be harder (even impossible) to achieve when number of replicas is large • This complexity leads to delegation election for block finality, thus leading to highly centralized deployments in practice

BlockFin consensus vs others in the wild

BlockFin Consensus	Other Consensus Protocols
New replicas joining the network: • The new replica connects to the P2P datastore and is up-to-date *instantly* because of the shared state • It can receive transactions and participate in block validations instantly for the same reason • The replicas can go offline sporadically without affecting their state. When they are back online, they can start their activities without any *sync* required • The nodes in P2P datastore can also go offline sporadically. They will sync to live state when they are back online	**New replicas joining the network:** • The new replica must *trust* one of the existing replicas to build its local state. It must download GBs of data (Bitcoin's blockchain is ~149GB and Ethereum's is ~57GB) before building its local state • The replicas must continually *sync* with each other to remain *current*
Attack Vectors: • The shared state makes P2P datastore target for DDoS attacks. But P2P datastore is replicated so there is no SPF • History rewinding (long range attack) is difficult because the entire blockchain is *public* all the time. Once the blocks are sealed, any attempts to modify them will trigger notifications to all replicas. Since each operation has replica's signature, it is easy to identify the offending replica	**Attack Vectors:** • Since practical implementations tend to be centralized (EOS for example has 21 *known* validators) they become targets for DDoS • If a replica has enough hash or cash power, it can mount long range attack because the blocks are built *privately* and *published* to the blockchain
Decentralization: • Practical deployments are truly decentralized because all replicas participate in block validation • Liveness is ensured via incentivization with block rewards. All nodes win portions of block reward for all blocks because BlockFin is a *community (leaderless) consensus* protocol. The incentivization ensures that at least ⅔ +1 replicas are online all the time	**Decentralization:** • Theoretically true decentralization is possible, but practical implementations have been highly centralized • Nondeterministic block rewards incentivize poorly • *Competitive (leader or delegation) consensus* models end up being highly centralized because stake-holders get bigger over time making it difficult for small players joining the network