# BFT in Blockchains: From Protocols to Use Cases

XIN WANG, University of Maryland, Baltimore County
SISI DUAN, Tsinghua University, corresponding author
JAMES CLAVIN, University of Maryland, Baltimore County
HAIBIN ZHANG, Beijing Institute of Technology, corresponding author

A blockchain is a distributed system that achieves strong security guarantees in storing, managing, and processing data. All blockchains achieve a common goal: building a decentralized system that provides a trustworthy service in an untrustworthy environment. A blockchain builds a Byzantine fault-tolerant system where decentralized nodes run a protocol to reach an agreement on the common system state. In this article, we focus on the research of BFT protocols. In particular, we categorize BFT protocols according to both the system models and workflow. We seek to answer a few important questions: How has the research in BFT evolved in the past four decades, especially with the rise of blockchains? What are the driven needs for BFT research in the future?

CCS Concepts: • **Security and privacy → Distributed systems security**; • **Computer systems organization → Reliability**; **Availability**.

Additional Key Words and Phrases: blockchains, consensus, Byzantine fault tolerance, survey

## 1 INTRODUCTION

Blockchains are distributed systems that provide a secure and trustworthy service via a group of parties that do not have to trust each other [202]. Blockchains have many forms, ranging from cryptocurrencies such as Bitcoin [182] to more general-purpose systems such as Ethereum [226] and Hyperledger Fabric [30]. A Blockchain acts as a trusted service for maintaining a shared state and providing secure and reliable data storage and processing service. In recent years, blockchain implementation in real systems has increased and as of today, there are numerous startup companies and corporations using blockchain in a wide range of industries. Because of the strong security guarantees, blockchains can potentially disrupt any industry in which the data and the service need to be protected.

Blockchains can be generally categorized into permissionless blockchains (e.g., Bitcoin and Ethereum) and permissioned blockchains (e.g., Hyperledger Fabric). There is no barrier to entry with permissionless blockchains, anyone can participate so long as they can run the application necessary to participate in its network. By comparison, permissioned blockchains require participants to know each other's identities upfront, but they do not have to trust one another. Both types of blockchains achieve the same goal: developing a Byzantine fault-tolerant (BFT) system using consensus, sometimes referred to as the Byzantine generals problem [150]. Byzantine fault tolerance replicates the system state of a single node to multiple distributed nodes. The distributed nodes work to reach a consensus about one common state during normal-case operations, as well as when

Authors' addresses: Xin Wang, xinwang11@umbc.edu, University of Maryland, Baltimore County, 1000 Hilltop Cir, Baltimore, MD, 21042; Sisi Duan, duansisi@tsinghua.edu.cn, Tsinghua University, corresponding author; James Clavin, jclavin@umbc.edu, University of Maryland, Baltimore County, 1000 Hilltop Cir, Baltimore, MD, 21042; Haibin Zhang, Beijing Institute of Technology, corresponding author.

0360-0300/2021/1-ART1 $15.00
https://doi.org/10.1145/3503042

arbitrary failures and malicious attacks occur. Together the distributed nodes behave as a single node, thereby using replication to provide high availability of services and integrity of the data, despite failures and adversaries. Because blockchain builds a Byzantine fault tolerant service and given its rise in recent years, there has been renewed interest in studying and implementing distributed consensus protocols generally.

Yet the field of blockchain is still developing, partly because building a trustworthy blockchain demands expertise in cryptography, security, and the theories of distributed systems [62]. Also, the nature of what constitutes a blockchain is still being grasped. Different interpretations of what blockchains are, their capabilities, and how they should be used exist for many people. These differing interpretations have caused misunderstandings about the technology that threaten its chances of adoption. Furthermore, as developers and researchers explore blockchains in numerous application domains, the applications have varying system needs dependent upon the given context. For instance, financial applications may favor security and privacy over performance. In comparison, other less critical systems may demand high throughput and scalability in most cases, while still requiring the system to handle arbitrary failures and malicious attacks.

The purpose of this article is to provide a comprehensive review of the categories of blockchains, the underlying consensus mechanisms and principles, and the consensus protocols used and proposed by both industry and academia. In particular, we focus on the research of BFT protocols with their usage in both permissioned and permissionless blockchains. We aim to not only categorize the BFT research in the past four decades, but also to understand the evolvement of BFT research especially upon the rise of blockchains, and have an outlook of BFT research in blockchains. Based on a brief review of several known use cases of blockchains, we would also like to provide insights on what is the most appropriate model for each use case. Note that although the consensus is the key to the correctness of blockchains, blockchains do not only include consensus [44]. Studies on other aspects of blockchains could be found, e.g., blockchain interoperability [163] and smart contracts [114, 131, 233]. Our review focuses on BFT approaches. Therefore, the insights we provide, especially on the impact to the use cases, are based on the model and performance of the BFT mechanism.

**Why another survey about Blockchains and BFT?** Several survey papers for blockchains or BFT consensus are relevant to our paper [38, 62, 80, 183, 185, 193, 221, 222], and there are reviews of BFT used by industry systems and specific application areas [62, 73, 93, 157, 185, 221, 222, 227], including reviews that focus on BFT only [80, 93, 193]. Compared to previous survey papers, our article makes the following additional contribution. First, we categorize BFT into several types and analyze their properties from the perspective of blockchains, i.e., whether any protocols in the category are being used by real systems, whether the category is *suitable* for blockchains. Second, we provide insights on how to build a correct consensus protocol and how to understand the consensus protocols to build a correct blockchain system. Third, from the perspective of blockchains, we analyze the evolving trend of BFT research in the past four decades. Last but not least, based on the discussion of BFT categories and use cases, we provide suggestions on which consensus approach might be the best *fit* for each.

The rest of the paper is organized as follows. We begin with the introduction of the terms and categories of blockchains in Sec. 2. In Sec. 3 and Sec. 4, we provide a comprehensive review of the different categories of BFT used in both permissioned and permissionless blockchains. We review the use cases that are being explored by researchers and decision-makers in Sec. 5. Finally, in Sec. 6, we also summarize the features of the BFT approaches, summarize the evolving trend of BFT in the past four decades, and provide insights on when to use which type of blockchains from the BFT perspective.

## 2 BLOCKCHAINS: TERMS AND CATEGORIES

### 2.1 System Model

A blockchain involves a number of *nodes/replicas* as participants. The replicas are servers in the client-server model. Clients submit *transactions/requests* to the servers and expect a reply for each request. The transactions may involve certain operations to be executed by the replicas. The replicas *deliver* the transactions after reaching a consensus on the order of the transactions. Blockchains organize transactions into blocks. Each block has a fixed size and consists of multiple transactions. In some systems, transactions are organized into *batches*, where each batch consists of a certain number of transactions. In this paper, we use transactions, client requests, and operations interchangeably.

A correct blockchain system tolerates Byzantine failures, i.e., arbitrary failures that could be caused by software bugs, hardware errors, and malicious attacks. In comparison, there are also other types of failures such as *crash failures*, where a crashed node simply stops executing any operations.

**Timing assumption.** The timing assumptions of the systems can be categorized into the three following models.

- *Synchronous model.* There exists a known upper bound for message delivery and processing time.
- *Partially synchronous model* [103]. There exists an unknown upper bound for message delivery and processing time.
- *Asynchronous model.* There does not exist an upper bound for message delivery and processing time.

**Adversary models.** The system may rely on different adversary models. It can roughly be divided into the following models [21].

- *Threshold Adversary (TA) Model.* The system has $n$ nodes in total and the adversary controls up to $f$ of them. A typical threshold is $n > 3f$.
- *Computational Threshold Adversary (CTA) Model.* The system has a total of $n_c$ computational power and the adversary controls up to $f_c$ computational power. A typical threshold is $n_c > 2f_c$.
- *Token Threshold Adversary (TTA) Model (Cryptocurrencies only).* The system has a total of $n_t$ tokens and the adversary controls up to $f_t$ tokens.

Besides Byzantine nodes, there may also exist a malicious *network scheduler* (considered in asynchronous systems only), which may delay the messages between any two nodes arbitrarily but cannot drop the messages. In other words, if a correct sender sends a message to another correct receiver, the message is eventually received by the receiver but there is no guarantee on the upper bound for message delivery time.

In the TA model, there is an important concept called **quorum**. Specifically, in the Byzantine failure model, if there are $n$ replicas, the system can tolerate $f = \lfloor \frac{n-1}{3} \rfloor$ replicas and the quorum size is $\lceil \frac{n+f+1}{2} \rceil$. In certain protocols or in the crash failure model, $f = \lfloor \frac{n-1}{2} \rfloor$ replicas and the quorum size is $\lceil \frac{n+1}{2} \rceil$. For a consensus protocol that considers TA model, it requires a quorum of replicas to reach a consensus before an operation is delivered.

Recent works also consider quorum based systems in asymmetric trust settings [61, 63, 85]. Specifically, conventional systems consider that all replicas have symmetric views on the number of failures in the system. Asymmetric quorum-based systems allow each replica to choose which nodes it trusts and which nodes can be considered faulty. Asymmetric quorum systems generalize conventional Byzantine systems by allowing replicas to maintain different quorum sizes. Similar idea has also been considered in practice such as the Stellar consensus [173].

**Smart contracts.** Smart contracts are small programs that can be written by any client and that are deployed to the blockchain in the form of a transaction. These contracts contain functions that can be called by users to update the smart contract, who will then submit a transaction to the system. After replicas reach a consensus about the order of the transaction, replicas then execute the smart contract's requirements accordingly and return the result to the clients. Popular smart contracts include Ethereum virtual machine (EVM) and Hyperledger Chaincode.

**System goals.** Blockchain achieves *atomic broadcast* in the Byzantine failure model. This is usually achieved using Byzantine fault-tolerant state machine replication (BFT-SMR). The correctness of the system can be defined as follows.

- *Agreement.* If a correct replica delivers a transaction $m$, then every correct replica delivers $m$.
- *Total Order.* If a correct replica delivers transaction $m_1$ before $m_2$, then any other correct replica delivers $m_1$ before $m_2$.
- *Liveness.* If a transaction is submitted to a sufficient number of correct replicas, then all correct replicas eventually deliver $m$.

The total order property is a safety rule while the agreement and liveness properties are liveness rules. The agreement property is not *required* in some BFT protocols [65, 230] (in fact, most partially synchronous BFT does not explicitly require agreement), as it can be achieved via other approaches, e.g., state transfer.

There is another closely related concept called *reliable broadcast*, where a designated sender sends a message to all replicas. It guarantees that correct replicas all deliver the message or no one delivers the message. If the system tolerates Byzantine failures, the problem is also called Byzantine broadcast.

**System parameters.** A blockchain consensus runs in *epochs* (sometimes referred to as rounds). In each epoch, the consensus is reached on the order of one transaction or one batch of transactions. The transaction is assigned with a *sequence number* (or height) that represents the order of the transaction in the transaction history. To reach a consensus, some protocols may run several *rounds* before the protocol terminates. In these types of protocols, replicas run the same *steps* several times before they reach a consensus on certain operations.

**System evaluation criteria.** Blockchain systems are evaluated experimentally according to several criteria: latency, throughput, server scalability, and client scalability.

- *Latency.* The period of time from the client sending a transaction to receiving a reply.
- *Throughput.* The number of transactions processed per second.
- *Server scalability.* The number of servers the systems can have to achieve decent throughput.
- *Client scalability.* The number of clients that can be handled by the system where the clients send/submit transactions concurrently.

Theoretically, latency can be evaluated as the number of steps in each protocol. Throughput can be evaluated by the number of messages/cryptographic operations the bottleneck server needs to process or the total number of messages in each consensus epoch.

There are also two other theoretical criteria:

- *Message complexity.* The total number of messages replicas need to exchange, which can be used to evaluate the server scalability.
- *Communication complexity.* The total size of the messages replicas needs to exchange, which can be used to measure the expected network bandwidth for the system.

Most protocols are evaluated experimentally using various workloads. Several solutions are also proposed with various benchmarks to evaluate different blockchain protocols [92, 228].

## 2.2 Categories

Blockchains are categorized into *permissionless* and *permissioned* blockchains. In permissionless blockchains, anyone can join the system. In permissioned systems, a node needs *permission* to join the system and nodes need to know the identities of each other. Most permissionless blockchains consider the computational threshold adversary model and assume a synchronous network. In contrast, most permissioned blockchains consider the threshold adversary model and a partially synchronous/asynchronous network. Some blockchains are hybrid and combine the features of both permissionless and permissioned blockchains.

Depending upon the members/organizations that participate in the blockchains, blockchains can be categorized into public blockchains, private blockchains, and consortium blockchains. In public blockchains, anyone can join the blockchain system. In comparison, the private blockchain is run by a single organization. Consortium blockchains allow multiple organizations to run the system. Most permissionless blockchains are public blockchains but some organizations have adopted them as private blockchains to improve performance over existing processes. In comparison, most permissioned blockchains are private or consortium blockchains.

**Cryptographic Hash Chain.** All blockchains adopt the same approach to represent the total order of the transactions, i.e., cryptographic hash chain, as illustrated in Fig. 1. For each block k, the node that proposes the block generates a hash of the block and then appends a digital signature to the block. Before the next node starts to propose a new block, it must verify the hash and the signature. If the previous block is verified, the hash and the signature of the previous block and the transactions of the current block will be used to generate the hash of the new block. The list of the transactions over time becomes a hash, or block, chain. Together with the underlying consensus protocol, such a hash chain ensures that no one can manipulate the contents of any block, and the sequence of transactions form a total order which all nodes have reached an agreement on. In most permissionless blockchains, ideally, the blocks should be proposed by different nodes to ensure the correctness of the system. In contrast, in BFT-based permissioned blockchains, a node does not necessarily have to append a digital signature or verify the digital signature of the previous proposer, as the underlying consensus protocol already guarantees the total order of the transactions.

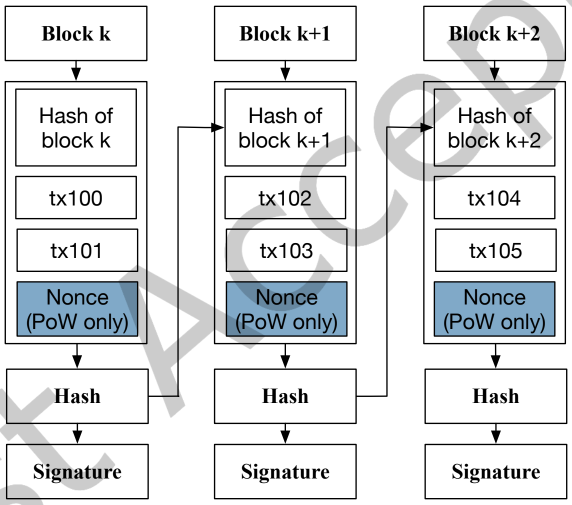


Fig. 1. Cryptographic hash chain.

**Permissionless Blockchains.** Most permissionless blockchains adopt a Proof-of-X, or -Something, strategy, as discussed in greater detail later. In the case of Bitcoin, this is Proof-of-Work (PoW). PoW is a mathematical puzzle that needs to be solved, the process of which is also known as mining. Once mined, a node proposes a new block and is rewarded if the proposal is finalized. The drawback to this approach is that the throughput is limited, and the energy consumption is high. Furthermore, nodes may 'collaborate' and form mining pools in order to solve PoW faster than others. As a result, a PoW system is reduced to a limited number of mining pools, making the blockchain less decentralized and therefore less secure.

**Permissioned Blockchains.** Most permissioned blockchains are based on BFT-SMR. Less energy is consumed by BFT-based systems because nodes actively reach an agreement by running a BFT protocol to determine the total order of transactions. A BFT protocol allows nodes to exchange messages in several steps before an agreement is reached. BFT protocols can be leader-based or leaderless. In leader-based protocols, the leader proposes the order of the transactions and other nodes choose to agree or disagree. When the leader fails, another new leader will be selected. In comparison, in leaderless protocols, all nodes can propose and those ones that the nodes reach an agreement on will be merged in the end. Compared with permissionless blockchains, BFT is provably secure and the system performance is usually orders of magnitude faster.

**Hybrid Blockchains.** Several hybrid blockchains combine the two types of blockchains to enjoy the benefits of both [105, 117, 141, 143, 155, 165, 232]. Most of these approaches rely on PoW and its variants to build an open blockchain where anyone can participate. But to address the low performance and high energy consumption of PoW, different permissioned techniques are applied. For instance, each node that solves PoW may be allowed to propose multiple blocks instead of one so as to reduce the frequency of mining [105]. In another case, PoW can be used as a membership protocol to select a small group of nodes called committees. The committees can then represent all nodes in the system to determine the order of the transactions, possibly using BFT to reach consensus among committees [141, 143, 165].

## 3 BFT PROTOCOLS

BFT, as the only generic approach to handle arbitrary failures, has been studied extensively since it was proposed in the 80s [150]. After the first-ever practical BFT protocol, called PBFT, was developed [65], numerous practical protocols have been proposed. BFT can be categorized into three types according to the timing assumption: *synchronous*, *asynchronous*, and *partially synchronous* protocols.

BFT assumes the TA model. It is known that in the synchronous model, a Byzantine broadcast problem can be solved using $n \geq f + 1$ replicas [94, 106, 107, 109, 109] and a BFT agreement protocol needs $n \geq 2f + 1$ replicas to tolerate $f$ Byzantine failures [19, 23, 135]. This is the same with crash fault-tolerant protocols in the partially synchronous model [130, 149, 187] that are widely used in real systems such as Google's Chubby [53] and Apache Zookeeper [124]. In partially synchronous and asynchronous models, a BFT system has to have $n \geq 3f + 1$ replicas. With additional tools such as trusted hardware, the requirement could also be reduced to $n \geq 2f + 1$. Although most permissionless blockchains consider a synchronous model, real networks are not synchronous. Furthermore, to the best of our knowledge, synchronous BFT protocols have never been implemented in any real system. Therefore, in this article, we only consider partially synchronous and asynchronous protocols.

**System components.** A BFT protocol consists of two components: operation of the protocol and garbage collection (also known as checkpoint sub-protocol). During the operation of the protocol, replicas reach an agreement on the order of client requests (transactions). In different BFT categories, protocols may further have different sub-protocols for the operation of the protocol. During the garbage collection, replicas generate stable checkpoints, delete unnecessary system logs, and release memory space. The garbage collection process is relatively independent of the operation of the protocol. Although the garbage collection sub-protocol may affect system performance and durability of the system [45], the same garbage collection approach can be applied to most BFT protocols without affecting the correctness of the protocols. In this article, we focus on the operation of the protocol, compare and contrast the BFT protocols in terms of the operation.

### 3.1 BFT in the partially synchronous model

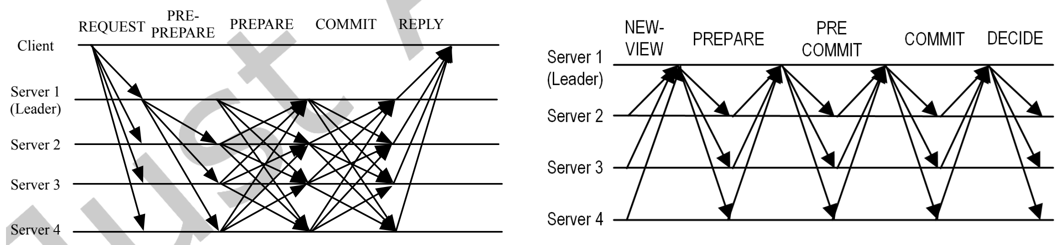
Most BFT protocols assume the partially synchronous model. The protocols are usually leader-based, i.e., there is only one leader at a time and the replicas reach a consensus about the identity of the current leader. Such protocols usually involve two major subprotocols: *normal-case operation* and *view change*. Replicas run the normal-case operations according to the protocol when the leader is correct. If the leader is suspected to be faulty, replicas run the view change subprotocol to elect a new leader. To detect whether the leader is faulty, replicas need to set up timers and ensure that the system makes progress. For instance, each replica sets up a timer for the first pending transaction in its queue. If the transaction has not been delivered before the timer expires, the replica will run the view change subprotocol. If a sufficient large fraction (i.e., a quorum) of replicas start the view change subprotocol, a correct new leader will eventually be selected.

According to the FLP theorem [108], synchronous or partially synchronous protocols may suffer from zero throughput in an asynchronous environment and make no progress at all. According to the theorem, the safety (agreement and total order) of such protocols will not be violated. Therefore, the safety of most partially synchronous protocols can be proved in an asynchronous environment. The liveness, however, requires partially synchrony, i.e., the network eventually becomes synchronous. Otherwise, the protocol may not make progress at all, achieving zero throughput. Despite the zero throughput issue, in a normal network with no network scheduler, partially synchronous protocols are usually very fast, achieving low latency and high throughput. Therefore, the majority of BFT protocols in the literature assume partially synchrony.

We categorize partially synchronous protocols into the following types according to the characteristics of the workflow: *broadcast-based BFT*, *hybrid BFT*, *chain-based BFT*, *trusted hardware-based BFT*, *scalable BFT*, and *others*, the description of which are shown in Table 1. A protocol may fall into multiple categories. For instance, most trusted hardware-based BFT protocols are also broadcast-based BFT. Therefore, our intention is not to draw a line between different categories, but rather to show the similarities and design motivation for the protocols.

| Category | Description |
|---|---|
| Broadcast-based BFT | Protocols that involve all-to-all or one-to-all communication |
| Hybrid BFT | Protocols that switch between different sub-protocols |
| Chain-based BFT | Protocols where replicas are organized into a chain-based topology |
| Trusted hardware-based BFT | Protocols that involve trusted based hardware component |
| Scalable BFT | Group-based or hierarchical protocols that are designed to scale the number of replicas involved |
| Others | Protocols aiming at other goals such as confidentiality |

Table 1. Classification of partially synchronous BFT protocols. A protocol may fall into multiple categories.



(a) Normal-case operation of PBFT [65].

(b) Normal-case operation of HotStuff [230]. Client requests and replies are ignored.

Fig. 2. Normal-case operation of broadcast-based BFT.

**Broadcast-based BFT.** Broadcast-based protocols involve several steps of *all-to-all* or *one-to-all* communication, i.e., each replica may need to communicate with other replicas. This is a conventional message pattern employed by BFT protocols. As illustrated in Fig. 2(a), the normal-case operation of PBFT involves three steps of all-to-all communication. The leader first assigns an order to a batch of transactions and sends a PRE-PREPARE message to the replicas. The replicas acknowledge the transaction by sending PREPARE messages to each other. A replica collecting a total of $2f + 1$ matching PREPARE messages will send a COMMIT message to other replicas. Finally, if a replica receives $2f + 1$ matching COMMIT messages, a consensus is reached by sufficient correct replicas. The replica can then deliver the transactions, execute the corresponding operations, and send a reply to the client. Since the protocol is leader-based, other replicas also need to *monitor* the correctness of the current leader. If more than $f + 1$ replicas suspect the current leader, they trigger a view change and exchange another step of messages to elect a new leader. This view change may continue until a correct leader is in charge.

The first practical BFT protocol, PBFT, is a broadcast-based protocol [65], and it is to-date still one of the most practical, with variants that are widely used in permissioned blockchains [62] and hybrid blockchains [89, 165, 190]. However, PBFT has some performance issues in some cases, and solutions have been proposed, each with varying costs [27, 32, 76]. The root issue is that malicious replicas in the system can *collude* and manipulate the value of the timer and make it extremely large. This is because the value of the timer may not be *appropriate* in a partially synchronous environment so the protocol resets the timer (by doubling the value) during each view change. After the timer becomes sufficiently large, a faulty leader can then make the system process certain client requests before the timer expires so as not to be suspected by the replicas. Other client requests can be delayed arbitrarily. In this way, the system becomes extremely slow and nothing goes 'wrong'. Although this type of attack focuses on PBFT, such an attack can likely be applied to most broadcast-based partially synchronous protocols.

Numerous works enhance the performance of PBFT [18, 32, 52, 82, 99, 122, 144, 171, 209, 217]. A common approach is to shift tasks from replicas to the clients to reduce the message and communication complexity of the replicas [18, 82, 122, 144, 171, 217]. This approach is inspired by the concept of quorum systems [175]. For instance, QU [18] builds a system that requires $n \geq 5f + 1$. It requires clients to directly communicate with the replicas to read and update data. A client only communicates with a quorum of replicas and validates the results before proceeding to the next step. HQ [82] further optimizes the approach and lowers the lower bound of $n$ to $n \geq 3f + 1$. Such approaches rely on correct clients for the system to make progress. Therefore, replicas also need to authenticate themselves before delivering an operation, which can prevent faculty clients from making replicas inconsistent. Such an approach acts as a leaderless system which reduces the message complexity of PBFT-like protocols from $O(n^2)$ to $O(n)$. The drawback, however, is that client requests cannot be batched so the system performance may still be limited in practice.

The approach can be further improved using a leader-based workflow. Specifically, the leader communicates with the replicas and replicas do not necessarily have to communicate with each other [122, 144, 171, 230]. For instance, in Zyzzyva [144], the leader first sends a message to the replicas. The replicas directly reply to the client. If the client receives matching replies from all replicas, the operation is delivered. Otherwise, replicas need to further exchange messages with each other. Fab [171] uses a similar workflow. This reduces the number of steps of PBFT from three to one in the failure-free case and two in case of backup failure(s). The message complexity can be reduced to $O(n)$ in the failure-free case but the complexity under failures is still $O(n^2)$. Unfortunately, Zyzzyva and Fab were later on found to have safety issues due to their design of the view change and solutions have been given accordingly [20]. The message complexity remains the same for both Zyzzyva and Fab. The difference is that the view change subprotocol involves more steps.

HotStuff [230] is another recent addition to the BFT family of protocols. It is inspired by the concept of consistent broadcast, a classic approach in distributed systems [57]. As illustrated in Fig. 2(b), the leader communicates with the backups and collects votes from them. This continues three times before replicas deliver the corresponding operation, which can be mapped to the three steps in PBFT for ensuring correctness. This increases the number of steps in PBFT from three to seven but reduces the message complexity from $O(n^2)$ to $O(n)$. If replicas can utilize pipelining to compensate for the number of steps (which is also called Chained HotStuff), system throughput can be improved. Specifically, replicas can start processing the next transaction before the previous one is delivered.

Several broadcast-based BFT protocols have been used in the industry. For instance, a variant of HotStuff called LibraBFT [39] has been used by the Libra blockchain (now rebranded as Diem) [4]. Tendermint [52] (a variant of PBFT) has been used by Hyperledger Burrow [6] and Tendermint core [16]. Symbiont Assembly [15] uses a variant of BFT-SMaRt [209], another variant of PBFT.

▷ **Remark.** The broadcast-based BFT is the most popular type of BFT protocol due to its robustness. Specifically, it allows replicas to exchange messages before reaching a consensus so all replicas essentially share the same information. The major drawback is the unavoidable $O(n^2)$ message complexity which limits the server scalability. Although the complexity can be reduced to $O(n)$, latency can be increased significantly and replicas still need to perform extensive cryptographic operations to authenticate each other, although they do not necessarily have to communicate with each other. Threshold cryptography and its variants can be used to reduce the number of cryptographic operations. The performance in real systems, however, is yet to be carefully evaluated.

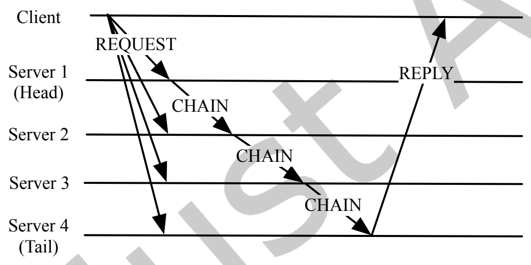
Note that some blockchains adopt non-BFT protocols or protocols that have not been formally analyzed or proven. For instance, Hyperledger Fabric provides the options to use Kafka [146] or Raft [187]. The Quorum blockchain [5] and R3 Corda [51] also provide an option to use Raft as the consensus protocol. The private version of Kadena blockchain [9] uses the ScalableBFT [225] protocol which is heavily inspired by Tangaroa [77], a BFT version of Raft. Raft and Kafka are two crash fault-tolerant protocols that do not tolerate Byzantine failures, and Tangaroa is neither safe nor live [62]. In other words, any protocols used in blockchains should be carefully reviewed before being deployed in real systems.

**Hybrid BFT.** A common belief is that there is no one-size-fits-all BFT, i.e., each protocol has its own trade-offs. The motivation for hybrid BFT is to use multiple protocols and switch between them to achieve the *best* performance and security guarantees according to the network situations, e.g., the number of failures in the system [37, 97, 118, 133, 186]. In those cases when failures are rare, the system can use a cheap protocol to guarantee great performance. When failures occur where a transaction cannot be delivered, the system then switches to a more expensive one to guarantee correctness. Aliph [118] proposes the first formal framework that switches between different protocols to provide a safe and resilient approach for replicas to transfer the system state from the previous protocol instance to the next one. In this way, even during a protocol switch, correct replicas are still consistent. CheapBFT [133] proposes a lightweight protocol using trusted hardware that tolerates $f$ failures using $n \geq f + 1$ replicas. It falls back to MinBFT [218] under certain conditions which requires $n \geq 2f + 1$ replicas. ADAPT [37] utilizes a similar idea. It further includes a run-time evaluation process powered by machine learning to evaluate the protocol characteristics and performance. The evaluation process can then *predict* the *best* protocol to use. Turtle consensus [186] and cost-sensitive consensus [97] take a different approach inspired by having moving target defense (MTD). Turtle consensus switches between protocols epoch by epoch to prevent a denial-of-service attack. Cost-sensitive consensus assumes an intrusion detection system (IDS) that oversees the system. It builds a software component and a formalized cost model that takes signals from the IDS to evaluate the *cost* for running different BFT protocols. The cost represents how vulnerable the current protocol is according to the network/system condition and whether the system should switch to a new one. The IDS does not have to be trusted for the system to be correct.

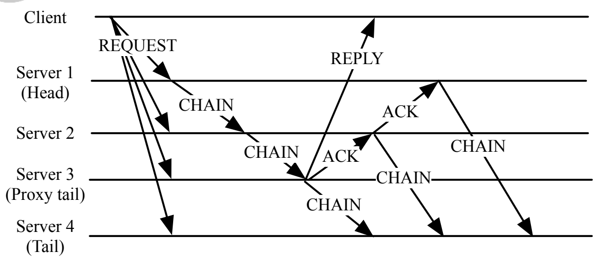
▷ **Remark.** Hybrid BFT provides a flexible interface to switch between different protocols, which balances between the performance and security trade-offs. Although such approaches are not known to be implemented in any real system, hybrid BFT allows one to reuse different protocol implementations and potentially build a robust system. The major challenge is two-fold. On the one hand, switching between protocols incurs additional latency. On the other hand, it is questionable whether the *selected* protocol is indeed the *best* according to the system conditions.

**Chain-based BFT.** Different from broadcast-based BFT, chain-based replication organizes replicas in a chain [98, 215]. A replica in the middle of the chain only communicates with the previous node (predecessor) and the subsequent node (successor) of the chain. The head (first replica in the chain) only communicates with its successor and the tail (last replica in the chain) only communicates with its predecessor. Such a communication pattern was previously studied in the crash failure model [170, 216] and later on extended to the Byzantine failure model. Aliph [118] also has a chain-based subprotocol that works only when all replicas are correct, as shown in Fig. 3(a). The major benefit of such a communication pattern is that every replica only communicates with at most two other replicas, so the protocol has $O(n)$ message complexity. The major challenge is that in the presence of even one replica failure, the chain breaks and replicas cannot reach an agreement. Shuttle [215] solves the problem using an oracle called *Olympus*. Olympus can be viewed as a trusted service that generates configurations for the system. When failures occur in the current chain, Olympus will issue a new configuration and replace all replicas in the current chain. Since Olympus becomes a single point of failure, it can be made replicated to achieve fault tolerance. In comparison, BChain [98] takes a different approach that detects failures in a distributed way, the normal-case operation of which is shown in Fig. 3(b). Specifically, the first $2f + 1$ out of $n$ (assuming $n = 3f + 1$) replicas form a chain. The rest $f$ replicas serve as backups and only learn the results from the first $2f + 1$ replicas passively in the normal-case operation. If the chain breaks where replicas cannot receive certain messages before they time out, replicas can 'suspect' each other. As a result, some replicas will be moved to the backups and certain replicas from the backups will be moved back to the chain. Replicas in the backup group will be recovered in the hope that the new replicas moved from backups to the chain are correct. Therefore, BChain is the only known robust chain-based BFT protocol. A variant of it has been used in Hyperledger Iroha permissioned blockchain [7].

▷ **Remark.** Compared with broadcast-based BFT, chain-based BFT enjoys the benefits of high throughput and great server scalability at the cost of higher latency. The latency can be compensated using the pipelining pattern. Similar to the pipelining approach used by broadcast-based BFT, a replica can start processing new requests before the previous one has been delivered. Since the message/communication complexity for chain-based replication is already lower than the broadcast-based BFT, the pipelining pattern makes it achieve an even higher gain for throughput improvement. The major challenge for chain-based replication is that failures can easily break the normal-case operation and replicas need to stop processing client requests before the chain stabilizes again. Therefore, such approaches work best when failures are less frequent.



(a) Normal-case operation of Aliph Chain [118], which requires all replicas to be correct.



(b) Normal-case operation of BChain [98].

Fig. 3. Normal-case operation of chain-based BFT.

**Trusted hardware-based BFT.** The motivation for using trusted hardware is to shift some tasks from the replicas (software) to trusted hardware. This has the effect of either preventing or monitoring equivocation [75] of replicas so they are 'forced' to behave correctly or act as crash faulty nodes. The benefit is that the system can tolerate $f$ failures using fewer replicas. A list of the available approaches is summarized in Table 2.

Correia, Neves, and Verissimo (CNV) proposed a component named *Trusted Multicast Ordering Service* (TMO) based on *Trusted Timely Computing Base* (TTCB) [78], a prior existing concept [81]. TMO (or TTCB) serves as an *ordering service* which authenticates the messages and directly multicasts the messages in the communication channels. Since malicious replicas do not have *control* over the channel, they cannot equivocate and therefore cannot behave arbitrarily. The faulty nodes, however, can become silent and act as crash failures. The system is then similar to a crash fault-tolerant one, which can tolerate $f$ Byzantine failures using at least $2f + 1$ replicas.

A2M [72] introduces *Attested Append-Only Memory*, and implements a local counter at each replica that records the messages transmitted in the protocol. Functionally, A2M acts as a message authenticator and a reference monitor. Similar to TMO, A2M also enforces that a replica only sends consistent messages to other replicas. TrInc [153] and MinBFT/MinZyzzyva [218] improve upon A2M by focusing upon the design of the trusted hardware in terms of storage space and design simplicity. CheapBFT proposes *Counter Assignment Service in Hardware* (CASH), an even simpler version of A2M. CheapBFT also includes CheapTiny, a lightweight protocol that only requires $f + 1$ replicas to tolerate $f$ Byzantine failures. Specifically, the system still has $2f + 1$ replicas and $f$ of them are *inactive* during the normal-case operation. If the $f + 1$ active replicas fail to reach an agreement, CheapBFT falls back to MinBFT to achieve correctness. FastBFT [162] optimizes the broadcast-based workflow of such protocols by creating a hierarchy of replicas. It builds a Trusted Execution Environment (TEE) based on a trusted component. Similar to other approaches, the TEE has a monotonically increasing counter to *mark* new messages and make replicas only send consistent messages. FastBFT also uses TEE for secret sharing between different hierarchies of replicas to authenticate the messages. Therefore, FastBFT can be viewed as a more scalable trusted hardware-based protocol compared with prior works.

ByzID [96] uses a slightly different approach. It builds a trusted component from the concept of the specification-based intrusion detection system (IDS). The component includes several specifications that define the *correct behavior* of each replica. Different from other approaches, the trusted hardware passively monitors the replicas instead of actively participating in the protocols. When a replica violates the specifications, the IDS generates an alert and triggers replica recovery. This has the benefit that replicas can continue processing even when the IDS fails. When the IDS fails, the system is correct only when at most $f$ faulty replicas crash since the system requires $2f + 1$ replicas to tolerate $f$ failures.

| Protocol | $n$ | Trusted hardware type/name | Functions |
|---|---|---|---|
| CNV [78] | $2f + 1$ | Trusted Multicast Ordering Service (TMO) | Authenicated channel |
| A2M [72] | $2f + 1$ | Attested append-only memory | Message authentication and reference monitor |
| TrInc [153] | $2f + 1$ | Trusted incrementer | Message authentication and reference monitor |
| MinBFT [218] | $2f + 1$ | Unique Sequential Identifier Generator(USIG) | Message authentication and reference monitor |
| CheapBFT [133] | $f + 1$ | Counter Assignment Service in Hardware (CASH) | Message authentication and reference monitor |
| ByzID [96] | $2f + 1$ | Specification-based IDS | Monitoring the message flow |
| FastBFT [162] | $2f + 1$ | Trusted Execution Environment (TEE) | Secret sharing and reference monitor |

Table 2. Comparison of trusted hardware-based BFT.

▷ **Remark.** Trusted hardware-based approaches can greatly enhance the resilience of the replicas, making a BFT protocol more performant, as well as greatly simplified and therefore more similar to a crash fault-tolerant one. The cost of tolerating failures can also be reduced from $3f + 1$ to $2f + 1$. The major drawback, however, is that the trusted hardware has to be reliable since most approaches require the trusted hardware to actively participate in the protocols. There is always a debate on *how many* tasks should be shifted from the software to the trusted hardware. Since all known TEEs have their vulnerabilities, implementing a fully-fledged BFT protocol cannot guarantee system correctness. Therefore, a common belief is that the functions implemented at the trusted hardware should be lightweight and the protocol should not rely on the trusted hardware to make progress.

Although to the best of our knowledge none of these protocols have been implemented in real systems, certain blockchain systems use a similar idea, e.g., Proof-of-Elapsed-Time (PoET).

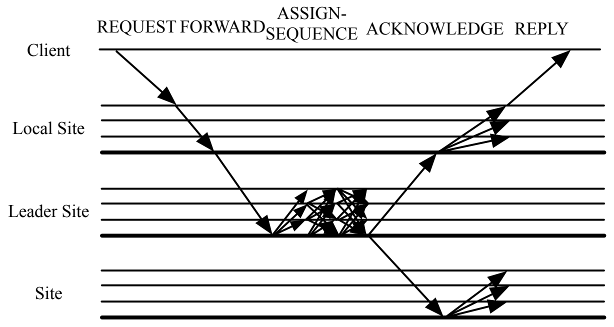


Fig. 4. Normal-case operation of Steward protocol [28].

**Parallel BFT.** BFT (or even CFT) is considered *slow* since it achieves a global total order of client requests. Therefore, the motivation for parallel BFT is to relax the requirement to order and execute the client requests one after the other to speed up performance. Specifically, the approaches usually run concurrent BFT instances to order and execute non-overlapping operations [32, 134, 145, 154]. CBASE protocol separates BFT agreement from execution [145]. It adds a *parallelizer* that uses application-related rules to identify requests that can be executed in parallel. This allows parallel execution of requests without compromising the correctness of the protocols (i.e., safety). In comparison, Eve uses an *execute-verify* approach [134]. The primary replica first groups requests into batches. Replicas then run a mixer to partition each batch and then execute the requests in parallel. Meanwhile, replicas run BFT to order the requests. After replicas reach an agreement on the order of all batches, replicas verify whether the results can be finalized by learning the results. SAREK is a parallel BFT approach that divides replica states into partitions [154]. Each replica runs multiple BFT instances, each of which orders request for one partition. SAREK provides an approach that deals with both operations that involve only a single partition operation and those that involve multiple partitions to achieve the total order of client requests. Different from these approaches where the purpose is to enhance the performance, RBFT [32] utilizes parallel instances to solve the performance attack of PBFT where faulty replicas can collude to slow down the system [27, 76]. RBFT runs $f + 1$ PBFT instances in parallel and one of them becomes the *master instance*. The master instance is the only one that executes the requests. If the master instance performs slower than other instances, a new instance will be selected.

▷ **Remark.** The concept of parallel BFT is not only limited to BFT, but is also widely studied in CFT [25, 168, 169, 200]. It has also recently been used in permissionless blockchains [231]. The approaches utilize modern multi-core computing architecture to enhance system performance. The major limitation is that the parallel execution of requests is dependent on the underlying application. Therefore, the *parallelizer* becomes the key to the correctness of the entire system. Furthermore, previous experience of deploying parallel CFT protocols in real systems shows that the parallel pattern may not achieve the expected performance [67]. Therefore, whether parallel BFT approaches are practical in real systems is yet to be discovered.

**Scalable BFT.** Scalability is known to be a major challenge for blockchains [219], and this type of challenge is different for permissioned and permissionless blockchains. BFT has high client scalability as the protocols have high system throughput. BFT, however, suffers from low server scalability due to its high message complexity. In other words, BFT protocols usually cannot scale to a large number of replicas. Several works explore enhancing the server scalability of BFT protocols and fall into one are of two categories: *hierarchical BFT* and *partition-based*

*BFT*. Both types of approaches reduce all-to-all communication to enhance the performance, a common approach in distributed systems [57, 86].

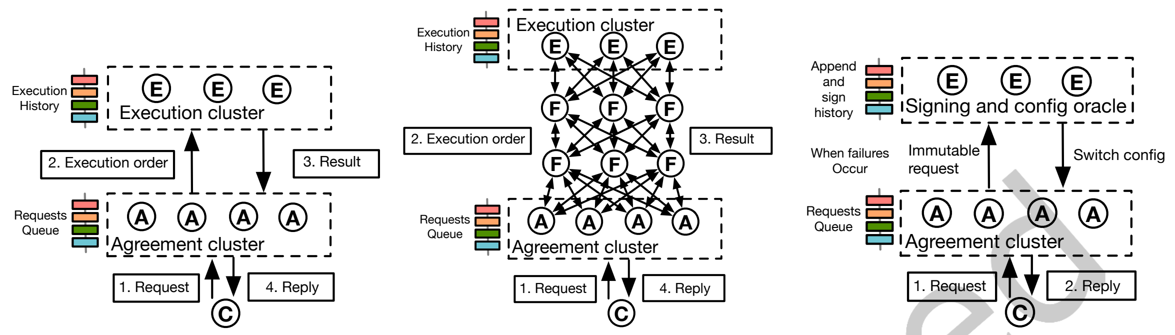
Hierarchical BFT creates a logical hierarchy of replicas. Such protocols usually involve a few phases of different BFT instances at different hierarchies [28, 69, 115, 127, 140, 162]. For instance, Steward [28] organizes replicas into several groups called sites. The normal-case operation is shown in Fig. 4. Each site has $n \geq 3f + 1$ replicas that tolerate $f$ Byzantine replicas. One site serves as a *leader site*, which can be changed if the current leader site fails. Every client contacts a local site to send a request. The client request is forwarded to the leader site if the local site is not the leader site. The leader site runs a BFT protocol (specifically PBFT) and assigns a global order to the request. The order is sent to all sites using a Paxos-like protocol. Finally, all replicas adopt the global order. The underlying idea is that each site itself is fault-tolerant so each site can logically act as a *replica* that may fail by crashing at the global level. To make sure other sites can validate the messages generated by a site, replicas in the site have to collectively generate message authentication using threshold signatures. At the global level, sites can then run a crash fault-tolerant protocol to broadcast messages. Since at the global level, only site representatives need to communicate with each other, this pattern can greatly reduce the all-to-all communication of conventional broadcast-based protocols and enhance the server scalability.

Partition-based BFT divides replicas into several parallel groups/partitions [46, 104, 147, 158, 198]. This approach could be extremely effective in applications such as file systems and storage systems [18, 82, 167, 175]. A few hybrid blockchains also utilize this idea [143, 232]. Performance can be greatly improved if write requests can be executed concurrently by different partitions. The major challenge for this approach is conflict resolution, such as when a write request may access multiple partitions [46, 158]. Concurrent access may degrade the performance since different partitions need to frequently resolve conflicts using expensive procedures. This is known to be a major issue in both the crash failure model [192] and the Byzantine model.

Several other approaches are designed with the motivation in enhancing the scalability of broadcast-based BFT [119, 197, 212]. SBFT [119] optimizes the workflow of PBFT. Specifically, instead of having all replicas exchange messages with each other in the second and the third step of PBFT, SBFT assigns one or a few replicas that serve as *collector* that collects signatures from the replicas and distribute the aggregated signatures to the replicas. This can reduce the all-to-all communication and enhance server scalability. Mir-BFT [212] uses a different approach that utilizes parallel computing to enhance performance. Specifically, Mir-BFT allows replicas to run parallel PBFT instances to order different client requests at the same time. To avoid the potential conflict of the client requests ordered by different instances, Mir-BFT uses a rotating assignment of the hash space to partition client requests at different instances. Therefore, Mir-BFT can be viewed as a parallel version of partition-based BFT. Snowball protocol uses a gossip-based mechanism based upon probabilistic broadcast [197]. Specifically, each node only sends a message to a fraction of the replicas instead of all other replicas (a common communication pattern in BFT protocols). This pattern greatly reduces the communication complexity to enhance the throughput and enables the Snowball protocol to guarantee reliable broadcast with high probability.

▷ **Remark.** Scalable BFT reduces the amount of all-to-all communication to enhance server scalability. The major challenge for such approaches is that they essentially use a slightly different failure model, i.e., each group/site has to be fault-tolerant and has a strict requirement on the fraction of failures. For instance, Steward assumes that each site of $n$ replicas is fault-tolerant and cannot have more than $\lfloor \frac{n-1}{3} \rfloor$ faulty replicas. This is different than assuming the entire system cannot have more than one-third faulty replicas.

**Others.** XFT is a recent work that improves the resilience of classic PBFT protocols, i.e., it requires $n \geq 2f + 1$ instead of $n \geq 3f + 1$. Slightly different from other BFT protocols, XFT requires that the majority of the replicas are not partitioned, i.e., the protocol does not tolerate network partition. TwinBFT [91] presents an approach that also reduces the requirement of $n$ to at least $2f + 1$. Specifically, it builds two virtual machines on each physical machine (replica). The two virtual machines monitor each other so that even one fails, the other one can still guarantee that only consistent messages are sent.



(a) Separating execution (data) from agreement (metadata) [31, 56, 102, 223, 229].

(b) Separating execution from agreement while preserving confidentiality [102, 229].

(c) Separating configuration from agreement for maintaining a set of correct orderers [215].

Fig. 5. Different approaches that separate the roles of the nodes.

Several previous efforts separate the roles in BFT for different purposes [31, 56, 102, 215, 223, 229]. Wang et al. [223], Cachin et al. [31, 56], Duan et al. [102], and Yin et al. [229] use an architecture that separates the agreement of client requests from the execution of the operations. A simple architecture is illustrated in Fig. 5(a). The agreement cluster has at least $3f + 1$ nodes to tolerate $f$ Byzantine failures. The execution cluster has at least $2t + 1$ nodes that tolerate $t$ failures. The client request is first sent to the agreement cluster and the nodes assign an order to the request. Then the request is sent to the execution cluster where the nodes execute the client requests according to the order. Such an architecture separates the data from the agreement, which has benefits such as better performance and modularity [224]. The architecture was further extended by Duan et al. [102] and Yin et al. [229] to achieve confidentiality. The approaches add a *privay firewall* that filters messages between the agreement cluster and the execution cluster, as illustrated in Fig. 5(b). This can preserve confidentiality where even corrupted replicas cannot learn the contents in the client requests. The Shuttle protocol, previously mentioned in chain-based BFT, also uses a similar architecture via the configuration oracle [215], as illustrated in Fig. 5(c). The role of the oracle is to obtain the data copies from the agreement cluster and sign the data as proof during reconfiguration. In this way, the nodes in the agreement cluster could be reconfigured and new nodes can directly obtain the stable data from the oracle.

Weighted BFT protocols have been proposed to *reduce* quorum size and enhance performance [42, 210]. Weights are assigned to the replicas so the quorum size is determined by the weights of replicas instead of the number of replicas. This can reduce the number of messages each replica needs to collect so as to reduce latency and enhance throughput. The major challenge is how to calculate the weights. WHEAT [210] uses a voting-based approach that allows replicas to collaboratively determine the weights. In AWARE [42] each replica continuously monitors other replicas' communication link latencies. The distribution of weights is then calculated and refreshed periodically. Similar concepts have been used in other types of BFT as well. For instance, Algorand [117] utilizes weights as the coins owned by replicas as part of the protocol. Several permissionless blockchains (e.g., Proof-of-Authority [88]) use voting-based approaches to select a group of replicas to run the BFT protocols. FairLedger builds a BFT protocol inspired by blockchain in financial applications [152]. Specifically, it achieves fairness where each participant gets an equal opportunity to append transactions to the ledger. It builds an abstraction with a failure detector, which defines the rational behavior of replicas and allows replicas to remove faulty replicas from the system.

## 3.2 BFT in asynchronous model

The FLP theorem states that in an asynchronous environment, a consensus protocol may never achieve correctness [108]. In other words, partially synchronous or synchronous protocols may achieve zero throughput against an adversarial asynchronous network scheduler. Randomized asynchronous protocols solve the problem using a probabilistic approach to ensure that the protocols terminate with the probability of 1 [40]. The protocols usually involve a *common coin*, which is collectively generated by the replicas. Practically, the common coin protocol can be generated using threshold cryptography [58]. As a result, randomized asynchronous BFT protocols are robust against timing, performance, and denial-of-service (DoS) attacks. Therefore, asynchronous protocols are (arguably) the most appropriate solutions for blockchains, especially mission-critical applications.

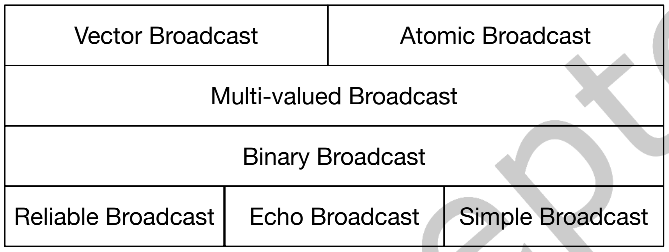


Fig. 6. The asynchronous protocol stack [57, 79, 180].

**Consensus stack.** Randomized asynchronous Byzantine fault-tolerant protocols can be built upon each other [57, 79, 180]. Conventionally, at the lowest level are two fundamental primitives: *reliable broadcast* (e.g., Bracha's broadcast [49]) and *echo broadcast* (a two-step version of Bracha's broadcast). Since several recent binary consensus protocols greatly simplify previous ones, we further add *simple broadcast* to the bottom of the stack, as illustrated in Fig. 6. In a simple broadcast, one replica simply broadcasts a message to all other replicas, which may include message authentication. *Binary consensus* can be built on top of them, which is the most fundamental primitive in the asynchronous protocol stack. Binary consensus allows each replica to have its own binary input, and agree on a single binary output. *Multivalued consensus* allows replicas to agree on a value of arbitrary length. *Vector consensus* is used to agree on a vector of values, each of which has an arbitrary length. Finally, *atomic broadcast* implements a total ordering service, which is also known as BFT SMR for building blockchains. For simplicity, we name them asynchronous BFT protocols.

**Randomized asynchronous consensus frameworks.** Multiple asynchronous BFT protocols have been proposed. Correia et al. [79] show that atomic broadcast can be built on top of multivalued consensus, which - in turn - can be built on top of binary consensus. All such protocols can be built using either reliable broadcast or echo broadcast [179, 180]. Although such a protocol might not be practical due to the long latency and low throughput, it presents that the consensus abstractions can be reduced from one to another. Specifically, it shows that all known asynchronous BFT primitives can be reduced to binary consensus.

The ACS (asynchronous common subset) framework is by far the most (and arguably the only) practical framework for asynchronous BFT. Specifically, in each epoch, each replica proposes a subset of transactions in its pending transaction pool. At the end of the epoch, correct replicas deliver the union of the transactions of the agreed-upon ones and assign them a deterministic order for the transactions. The framework was used by Ben-Or et al. [41]. Recently, it has been improved and made practical with implementation and extensive

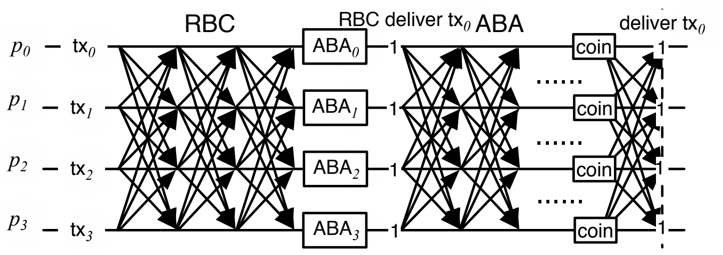


Fig. 7. The ACS workflow [41, 58, 60, 101, 120, 160, 166, 178]. The figure shows a typical workflow although different protocols use different variants. The ABA protocol might be different in different protocols.

evaluation by HoneyBadgerBFT [178] and BEAT [101]. HoneyBadgerBFT and BEAT design and implement several optimizations to reduce the communication complexity and the number of steps required in the system including a lightweight asynchronous binary agreement (ABA) protocol MMR [181]. The MMR has only two to three steps in each round which is more practical than other ABA approaches [43, 49, 50, 59, 64, 208, 214]. The MMR protocol, however, was found to have termination issues due to the underlying assumption about the common coin [2], which was solved in Cobalt by having one more step in each round [166].

The ACS framework used by HoneyBadgerBFT and BEAT is shown in Fig. 7 (with threshold encryption ignored). The protocols proceed in epochs. Each epoch consists of two phases: a reliable broadcast (RBC) phase and an ABA phase. In the RBC phase, each replica selects a subset of pending transactions that form a proposal. Each replica then uses RBC to broadcast the transactions. In the ABA phase, replicas run $n$ parallel ABA instances, the $i$-th of which is used to agree on whether the proposal from replica $p_i$ has been delivered in the RBC phase. For instance, as illustrated Fig. 7, each replica $p_i$ ($i \in [0..3]$) proposes a transaction $tx_i$ (or a batch of transaction) and uses RBC to broadcast the transaction. When a replica delivers value from an RBC instance $j \in [0..3]$, it inputs 1 to the $j$-th ABA instance. HoneyBadgerBFT and BEAT follow Ben-Or et al. [41] and ask each replica to abstain from proposing 0 until $n - f$ ABA instances are delivered. This is used to guarantee system throughput. EPIC [160] further enhanced the framework and presented an approach to achieve adaptive security. It has been shown that with little cost, asynchronous BFT with adaptive security can be made practical. Instead of using RBC and ABA with optimal resilience, MiB [161] (BFT with more replicas) chooses to utilize protocols requiring $n \geq 5f + 1$ or $n \geq 7f + 1$. It has been shown that, although more replicas are involved, the ACS framework can be made practical due to the optimization of the underlying components.

CKPS [58] and SINTRA [60] built a variant that relies on multi-valued broadcast, to be specific multi-valued binary agreement (MVBA). In particular, all replicas first broadcast their proposals with digital signatures. After each replica collects a quorum of signed proposals, they run MVBA. The MVBA consists of $n$ parallel consistent broadcast (CBC) instances (which can be viewed as reliable broadcast with $O(n)$ complexity) and an ABA phase. Each time, one ABA instance is triggered. The ABA instances, however, are triggered sequentially until one terminates. Dumbo (Dumbo2 to be specific) uses MVBA in a different way [120]. In Dumbo2, replicas first run a variant of RBC named PRBC, where the delivery of each request is associated with a valid proof (threshold signature). The proof and the index of delivered requests are used as input for MVBA. Dumbo shows that the scalability of asynchronous BFT (HoneyBadgerBFT) can be greatly improved using such a workflow.

Abraham, Malkhi, and Spiegelman (AMS) proposed another framework that reduces the message complexity (not communication complexity) of ACS from $O(n^3)$ to $O(n^2)$ [24]. As shown in Fig. 8, replicas run $n$ parallel instances each of which has a similar workflow with HotStuff [230]. In other words, every replica also proposes a subset of pending transactions. After a threshold of the instances has been completed, replicas collectively

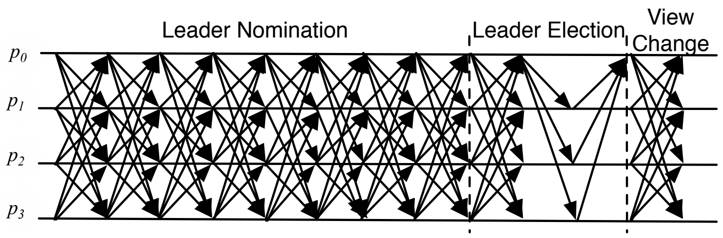


Fig. 8. The AMS workflow [24].

generate a common coin to select one of the instances, and the corresponding transaction(s) will be broadcast and delivered by the replicas. Since HotStuff has $O(n)$ message complexity, running $n$ instances in parallel makes such a framework achieve $O(n^2)$ message complexity. The framework, however, has very long latency and inefficient use of the network bandwidth. Specifically, although all replicas can propose in parallel, only the transaction(s) proposed by one replica will be delivered. The framework might therefore not be practical in real systems.

Randomized asynchronous protocols all use a common coin which can be realized using threshold cryptography in practice [48, 59, 129, 204, 205]. An important concept for threshold cryptography is whether such protocols are adaptively secure, i.e., they can handle adversaries that can adaptively choose 'which nodes' they want to corrupt. BFT protocols are therefore defined accordingly to achieve adaptive security [159, 164].

A recent work ACE shows a generic approach that can 'convert' a partially synchronous BFT protocol into an asynchronous one [211]. It uses a similar concept to AMS. Specifically, every replica can propose transactions by running parallel instances of a partially synchronous protocol. After a fraction of the instances has been completed, one instance is selected by the replicas and the corresponding transactions can be delivered. Since the approach is similar to AMS, it also has inefficient use of the network bandwidth.

Two other works are worth mentioning. Algorand [117] is a cryptocurrency that uses randomized BFT protocol as the core consensus mechanism. We ignore many details of the protocol and focus on the consensus mechanism which is the focus of this article. Algorand uses a workflow that is similar to ACS as it has two phases: reduction and binary agreement (BA). Reduction reduces the proposal of transactions to a binary decision and the BA is used to agree on whether the proposal has been received. Both the reduction phase and the BA phase are different from those used in ACS because in the reduction phase, every replica directly proposes a block of transactions by broadcasting the transactions. Other replicas can *vote* for the proposed transactions. If the number of votes for a block exceeds a certain threshold, the BA phase will terminate with value 1 and the transactions will be delivered. Otherwise, the BA will use the value of the common coin as output. Both the reduction phase and BA phase heavily rely on timers and timely messages to ensure correctness and termination. Therefore, the consensus of Algorand, despite using a randomized protocol, still assumes partially synchrony. DBFT [83] also extends the ACS framework by adding a weak coordinator which makes replicas complete the protocol *faster*. The resulting protocol assumes partially synchrony. Therefore, DBFT is not an asynchronous consensus, despite that it is randomized.

Certain asynchronous protocols are being implemented or used in industrial systems. For instance, HoneyBadgerBFT is being adopted as part of the solution in Ethereum blockchain [12]. DBFT, a randomized protocol that assumes partially synchrony, is known as the protocol for the Red Belly Blockchain [13].

▷ **Remark.** Randomized asynchronous BFT protocols are known to address the FLP impossibility at the cost of higher message complexity. The practical asynchronous protocols such as HoneyBadgerBFT and BEAT, however, have shown that such protocols can also achieve high throughput and low latency. This is because asynchronous protocols are leaderless where replicas can propose transactions in parallel. If non-overlapped transactions can be proposed by the replicas, such protocols can achieve high throughput to be used in practice, and because they are usually leaderless there is no need for a view change subprotocol. Therefore, the entire system implementation can be more lightweight than partially synchronous protocols.

## 4 BFT IN NON-PERMISSIONED BLOCKCHAINS

Besides permissioned blockchains, BFT can also be used for permissionless blockchains as well to enhance scalability and performance. These blockchains are also known as hybrid blockchains. Hybrid blockchains combine the features of permissionless and permissioned blockchains to enjoy the benefits of both types of blockchains. Most hybrid blockchains utilize PoW and its variants to build *membership* for the system and other approaches to enhance the system performance. PoW based consensus is known to prevent sybil attacks which can be used to build a permissionless blockchain. Specifically, to launch a sybil attack, a malicious party creates multiple identities in the blockchain to cause security and correctness issues. Therefore, it is unique in building an open blockchain. The drawback is that PoW is expensive. Therefore, PoW based blockchains suffer from low client scalability and performance. In contrast, the use of other approaches can be used to greatly enhance the system performance.

The integration of PoW and Proof-of-Stake (PoS) is an example of hybrid blockchains [54, 87, 136, 138]. In particular, the *voting power* of a node is determined in part by the amount of coins it holds. Another example is directed acyclic graph (DAG) based blockchains [36, 207]. In this type of blockchains, high forking is allowed where a block does not have to be proposed by the descendants of existing blocks. As a result, the throughput can be greatly enhanced.

Another line of hybrid blockchains utilizes PoW and BFT protocols. BFT protocols enjoy the benefits of great performance. The problem is that it can only be used in the permissioned setting where replicas have to know the identities of each other. Using PoW (or its variants) to select a group of replicas for running BFT can therefore *solve* the problem. Several hybrid blockchains combine PoW and BFT to build an open blockchain while enhancing the performance and security of permissionless blockchains [22, 89, 105, 141, 143, 165, 190, 232].

In this section, we discuss a few representative hybrid blockchains, most of which combine PoW and BFT. We discuss their generic workflow, the design motivation, with a focus on the use of BFT in such blockchains and discuss the pain points. A list of the hybrid blockchains is summarized in Table 3.

| Protocol | Membership | Rotating committee | Consensus | Fork | Sharding | Cross-Shard Tx |
|---|---|---|---|---|---|---|
| Bitcoin-NG [105] | PoW | Continuous | Single node | Yes | No | NA |
| PeerCensus [89] | PoW | Continuous | BFT | No | No | NA |
| Hybrid Consensus [190] | PoW | Periodic | BFT | No | No | NA |
| ByzCoin [141] | PoW | Continuous | BFT | No | No | NA |
| Solida [22] | PoW | Periodic | BFT | No | No | NA |
| Elastico [165] | PoW | No | BFT | No | Yes | NA |
| Omniledger [143] | PoW | Continuous | BFT | No | Yes | Yes |
| RapidChain [232] | PoW | Periodic | BFT | No | Yes | Yes |
| Thunderella [191] | PoW | Optional | BFT | Yes | No | NA |
| Algorand [117] | Public randomness | Periodic | BFT | No | No | NA |

Table 3. Comparison of hybrid blockchains.

**Hybrid blockchains.** The Bitcoin-NG [105] hybrid blockchain extends PoW based consensus by allowing each replica to propose multiple blocks of transactions. Since mining incurs high latency, allowing each replica to propose multiple blocks can greatly enhance the performance. Specifically, Bitcoin-NG has two types of blocks:

key blocks and microblocks. Microblocks are the blocks of client transactions just like those in other types of blockchains. Key blocks use PoW to select replicas and each replica becomes a *leader* which can propose a fixed number of microblocks. The major challenge Bitcoin-NG solves is that PoW (key block mining) may have forks and a faulty leader may create forks of microblocks as well. Bitcoin-NG solves the key block forks by simply asking replicas to wait until the current chain of key blocks is the longest. To solve the microblock forks, the transaction fee is split between the current leader and the next leader, making replicas have the incentive to maintain the only longest chain of microblocks. Although Bitcoin-NG does not use BFT in the design, the use of BFT in hybrid blockchain arises from it. In particular, hybrid blockchains switch between the consensus in permissionless blockchains to have a large number of permissionless members and BFT to enjoy better performance.

The majority of hybrid blockchains combine PoW and BFT [22, 89, 141, 143, 190, 191]. PeerCensus [89] was the first such protocol in this category. It uses PoW to select a number of replicas and form a *committee*. The committee members run a BFT protocol (PBFT to be specific) to propose new blocks of transactions. New members could be added to the committee through PoW and old members will leave the committee. Unfortunately, the reconfiguration of committee members was not clearly defined and a better approach has been proposed by hybrid consensus protocol [190] (which is different from the hybrid consensus category mentioned previously in Sec. 3.1). Specifically, hybrid consensus proposes to either use Fruitchain [189] (a variant of PoW) or PoW to select replicas and form a committee. The committee will be rotated periodically so that each committee runs a DailyBFT protocol (PBFT) before the next committee is selected. Solida [22] further enhanced the reconfiguration of the committee. Specifically, Solida formalizes the reconfiguration process in such protocols. Furthermore, it ensures correctness under the assumption that the adversary does not control more than one-third computational (mining) power. Thunderella provides different models where the committee members can be static or reconfigurable [191]. It combines a fast and asynchronous path with a slow synchronous path. In the fast path, a simple and almost optimistic execution is employed. When failures occur, a slow path is used, the motivation of which is similar to that in hybrid BFT.

Elastico [165] and RapidChain [232] improve scalability through the use of a sharding/parallel transaction processing. Both use the *unspent transaction output* (UTXO) model to create shards where UTXO is an approach used in Bitcoin. It allows each replica to calculate locally an Output for certain Input where the Input specifies the source of the coin in a transaction and a valid proof (digital signature). The Input range can, therefore, be used to create multiple shards that are not overlapped. Elastico [165] creates an overlay with a fixed number of committees each of which proposes transactions only for the corresponding shard. After the committees propose transactions in parallel, the proposed orders are then sent to a final committee to validate and merge the results. Elastico does not specifically address the cross-shard verification issue where the verification of a transaction involves more than one shard. RapidChain [232] provides a novel solution and an inter-committee routing protocol to solve the problem. Specifically, the cross-shard committees process the transaction in a pipeline so as to reduce inter-committee communication and enhance the system performance. Besides, RapidChain further provides several optimizations and shows experimentally that it outperforms all prior hybrid blockchains.

Note that ByzCoin [141] and Omniledger [143] are two other works that fall into the same category. Both protocols rely on a variant of PBFT that uses Cosi [213] for authentication. Cosi was found to have serious security issues and solutions have been provided accordingly [95]. Besides the BFT, ByzCoin can be viewed as a variant of PeerCensus and Omniledger is a variant of Elastico.

Algorand [117] can also be viewed as a hybrid blockchain that is also a cryptocurrency. Although it uses BFT-based consensus as the only consensus mechanism, it builds a *permissionless blockchain* where anyone can participate. To prevent sybil attacks, Algorand assigns weights to each user according to the number of coins in the wallet. In other words, Algorand is a cryptocurrency using consensus in the hybrid blockchains category.

▷ **Remark.** A fraction of hybrid blockchains are provably secure permissionless blockchains while some system do not have security proofs. It has been shown that such blockchains can greatly enhance the client scalability and throughput of permissionless blockchains. There are several major challenges. First, PoW relies on the synchronous network and the CTA model to achieve correctness. The issues brought by PoW consensus (e.g., selfish mining) cannot be solved in hybrid blockchains. Second, for a BFT-based hybrid blockchain to be secure, the committee has to have a certain size. In other words, each committee cannot have more than one-third faulty members for the BFT protocol to be correct. For instance, it is shown in Elastico that each committee has to have 600 members so that something bad (a committee has more than one-third faulty replicas) happens once every 1 million epochs. Although the requirements on committee size can be lowered, a typical BFT protocol cannot scale to such a large number of replicas. Therefore, it is still not clear how to achieve the sweet spot of client scalability and server scalability, especially in a permissionless blockchain [219]. Third, due to the design of the consensus approaches, several hybrid blockchains can only be cryptocurrencies (or the transactions have to be related to certain cryptocurrencies) instead of general-purpose blockchains due to their design.

## 5 USE CASES

Blockchains have been explored or used in different domains such as finance, biomedical, supply chain management, and governmental applications, some of which show blockchains can make a difference in the real world. We explore how it has been used to help give access to unbankable people; stop food-borne outbreaks earlier; improve the quality of health care; and make changes to how we govern. In this section, we review the use cases that are being piloted or explored globally, categorize them according to the industry. Note that extensive research efforts have been made to explore use cases for blockchains [29, 74, 121, 123, 137, 176, 201], some of which might potentially be used in the future. In this section, we only review a few representative use cases that are being piloted or deployed in real-world applications. Our purpose is not to advocate any blockchain systems or to provide an exhaustive review of the use cases, but to show the use cases and summarize how blockchains have been used in the real world. Understanding the workload and the needs of using a system (e.g., read, write, append) is a key component in the design of any systems [67, 116]. Therefore, the discussion of real-world use cases may greatly benefit researchers in the design of protocols and benefit decision makers to select the *appropriate* protocols.

### 5.1 Financial Applications

Blockchains became *famous* as cryptocurrencies, as they provide decentralized, trustless environments for people to exchange currencies with low transaction fees and relatively quick finality. Therefore, financial applications are so far still one major focus of developers and decision-makers. For instance, according to a World Bank survey, in Philippines, only 42% of Filipinos aged 15 or older have a bank account due to a combination of factors [90, 234]. One significant driver is that the rural settings of the country limit their ability to obtain one. If those banks get the infrastructure they need, then they can provide Filipinos with the ability to send and receive money, thereby reducing poverty and improving lives. The Filipino government has adopted an Ethereum-based solution for about 80 rural banks, built using Infrastructure as a Service (IaaS) from Amazon Web Service (AWS) and Microsoft Azure.

There are numerous financial use cases such as asset management, insurance claim processing, etc. For instance, in asset management, blockchains can be used to simplify the trade processes within asset management, prevent data redundancy at different roles of people in the process, and reduce the risk for trading [70].

Cross-border payments are becoming another great use case for blockchains. For instance, several major trading companies such as Ripple Labs Inc. are exploring the use of blockchains, RippleNet [14], for managing trade finance and unlock new business models. The idea is to use blockchains to enable the tokenization of existing documents, letters of credit, and more [114, 139, 156, 226]. Furthermore, general purposed blockchains might be used for cross-border payments.

Other financial application areas include anti-money laundering (AML), insurance, and regulatory compliance. The Financial Action Task Force (FATF) issued guidance on virtual assets, anti-money laundering, and counter-terrorist financing regulation [73, 132, 148, 195]. It has been shown that the existing cloud-based blockchain solutions of AML are effective in balancing the threats and opportunities. Another application is the insurance industry where insurance policies can be enforced through smart contracts. For example, OpenIDL [10], a network built on the IBM Blockchain Platform, is automating insurance regulatory reporting and streamlining compliance requirements. This allows insurance claims to be recorded and validated transparently, which could eliminate invalid claims. Similarly, accounting and auditioning for regulatory can be performed via blockchains as well [84, 142]. Any record updates via blockchains can be made immediately available to regulators and businesses, preventing human errors and ensuring the integrity of records.

▷ **Remark.** Most of these financial use cases utilize the cryptocurrency nature of blockchains. Financial applications, however, do not necessarily have to rely on cryptocurrencies. Instead, general-purpose blockchains can also be used as a secure storage and data processing system to increase the transparency of the data, reduce the data redundancy, and enable flexible data sharing.

## 5.2 Supply Chain Management

Supply chain management is a good use case for blockchain since the tamper-proof ledger provides a complete, unalterable, transparent history of food as it journeys from the farm or the ocean to the consumer. The E. coli outbreak of 2018 was a good motivation for using blockchain in the supply chain. The U.S. Food and Drug Administration (FDA) investigated the outbreak and concluded that a determination of the source of it was challenging because of the short shelf-life of leafy greens and their packaging, how widespread their distribution is, and the complex supply chain from the farm to the store [110, 125]. A blockchain-based solution can potentially help identify the source in real-time as the outbreak was occurring. The provenance of the outbreak and the customers affected could have been determined much more quickly. In response to the outbreak, Walmart successfully built and implemented blockchain prototypes for tracking the provenance of mangos in the U.S.A. and pork in China [125]. Their corporate office required all leafy green suppliers to use a blockchain solution [206]. In 2019 Walmart also began using blockchain to track shrimp shipped from India to the U.S. in a program that is reminiscent of the World Wildlife Foundation's (WWF) "from bait to plate" effort [111, 194]. In 2018 the WWF prototyped a blockchain solution for tracing tuna caught in the Pacific Ocean to combat illegal fishing. That solution has grown: in early 2019 it evolved into a platform named "OpenSC" that has been implemented in Australia for tracking fish to the ocean to market, and in late 2019 Nestlé Food Group announced a collaboration with OpenSC to track milk sent from New Zealand to the Middle East [184].

▷ **Remark.** Blockchain solutions in supply chain management can potentially greatly enhance the efficiency of data management and the security of data sharing. The major challenge is that such solutions cannot prevent human errors or inaccurate data from being uploaded to the system. On the other hand, blockchain can enhance the transparency of supply chain data so as to enhance the efficiency of supply chain management. Data privacy, however, becomes another major concern as many parties are involved in the supply chain.

## 5.3 Biomedical and Healthcare

Numerous research efforts have been made to explore blockchain applications in the biomedical and healthcare domain, due to the need to build a secure system for maintaining the domain data. The U.S. Health and Human Services (HHS) Department has developed an application called Accelerate for management of contract billing that utilizes blockchain, artificial intelligence, machine learning, and process automation. The purpose of Accelerate was to manage a portfolio of 100,000 contracts worth 25 billion US dollars across almost 50 systems. The blockchain within accelerating was used to capture the record of unstructured data (such as documents), rather than storing the data itself. Accelerate proved successful in getting contract information dispersed across the entire organization through replication of data, and was later to become the first blockchain-based application to be certified by the Designated Approving Authority as having the Authorization to Operate [172]. Accelerate was later expanded to acquisition management – replicating records of those contracts to researchers to make it easier for them to find materials for their investigations. HHS has projected savings at the point of purchase of up to 720M over time. Accelerate will expand to clinical areas to use blockchain for reporting sepsis disease data to enhance the efficiency of data management [199]. Besides, the Centers for Disease Control (CDC)'s Center for Surveillance Epidemiology and Laboratory Services also built proofs of concept for surveillance across state lines to track the ongoing opioid crisis [174].

Interest in blockchain for usage in biomedical applications has grown since 2008, as can be observed by searching PubMed.gov by publication year. The interest has grown exponentially in 2018 and 2019. Unfortunately, the publications on PubMed show that much of the research being done is theoretical in nature, with very few discussing deployments of blockchain in clinical settings. Notable works focus on data exchange and interoperability. For instance, efforts have been made to utilize blockchain's tamper-resistant capability and integrate blockchains with health electronic record standards such as HL7 and ISO 13606. Research and government efforts have been made to use blockchains to manage patient data, allowing patients to have control over their data.

▹ **Remark.** Many of these healthcare use cases are similar to supply chain management, e.g., tracking sepsis or opodis is similar to tracking the products in the supply chain. In other words, they all utilize the efficiency of data management of blockchains and the tracking capabilities of the transactions. Therefore, these use cases also meet the challenge of managing the inaccurate data in the system. Besides, healthcare applications also have additional challenges due to regulations and compliance.

## 5.4 Government Applications

Governments across the world have been exploring the use cases for blockchains in many different areas, as surveyed by previous work [73], such as biomedical and healthcare applications reviewed in Sec. 5.3, financial applications, asset management, and data management. For instance, governmental records such as birth and death date property transfers are typically recorded in paper form. Due to its duplicate and distortion prevention, blockchains are being used to simplify the recordkeeping and store the data securely [151]. Similarly, personal identities could be managed in the same way, making it easy to prove people's identities [126]. Examples include ID2020 [8] and Platform Identity Management Netherlands (PIMN) [11].

Blockchains could be used by non-profit organizations (NPOs) to solve the trust issues. For instance, blockchains can maintain the transparency of the data, showing donors that NPOs are using their money as intended. Blockchains can also help NPOs to tribute funds more efficiency and enhance tracking capabilities [203].

▹ **Remark.** Governments play an important role in the adoption of new solutions. The realization of transactions via blockchains requires complex integration work and a conducive regulatory environment. According to a summary report from the 2019 OECD global blockchain policy forum, governments should partner with private firms to encourage innovation and develop a flexible regulatory framework [3].

## 5.5 Critical Infrastructures

Several efforts have been made on critical infrastructures, i.e., power grid, smart city. Malaysia's Melaka Straits city, a tourist city funded by the Chinese government, has recently started the blockchain city project to track tourist visas, passengers, luggage, and booking services using blockchain [196]. The city will also exchange its own token, the DMI coin, for tourists to exchange their money into digital currencies for payment in the city via their mobile phones.

South Korea's government recently announced a pilot to set up a blockchain-enabled virtual power plant (VPP) in the city of Busan, the second-most populous city after Seoul [188]. The virtual power plant is a cloud-based distributed power plant that integrates multiple energy resources to optimize power generation. The project involves the city of Busan and several major local companies and institutions.

▷ **Remark.** Blockchain, being used in critical infrastructures, can protect the integrity of the data due to the tamper-proof nature of the system. The system is intrusion-tolerant since it tolerates arbitrary failures. Independent of these blockchain-based solutions, BFT-based solutions have been also used to protect critical infrastructures motivated by the same reason. For instance, Supervisory Control and Data Acquisition (SCADA) system, as a key component of the power grid infrastructure, is proposed to be replicated to avoid the single point of failure. BFT is used as the consensus protocol among the replicated nodes to guarantee the security of the data [33–35]. Indeed, if the features other than BFT in blockchains (e.g., smart contract) are not desirable, building an intrusion-tolerant system and implementing additional required features from scratch might result in a more efficient and performant system.

### 5.6 Social and Educational Applications

The Department of Information and Communications Technology of the Philippine government has started a blockchain-based pilot to offer services such as cost-benefit and socio-economic analysis in the Philippines [47]. The blockchain solution will serve as a system for analyzing, storing, and optimizing cloud-based data sets.

Several industries in Japan have adopted blockchain technology [177]. For instance, Fujitsu and Sony focus on blockchain within the education sector. Sony partners with IBM to use blockchain for student data management, targeting primary and higher education learners.

Blockchains for media is an area being explored. According to America News Hour [128], the global market for blockchain in media and entertainment is estimated to reach $1.54 billion by 2024. The use of blockchains can be used for original content royalties tracking, which has the potential to eliminate fraud, reduce costs, and protect Intellectual Property (IP) rights of the content, like artworks. For example, on 19 February 2021, an animated Gif of Nyan Cat, as a unique digital item authenticated on the blockchain, sold for more than $500,000 [17]. This artwork as a non-fungible token (NFT) can be thought of as certificates of ownership for virtual or physical assets [68].

▷ **Remark.** In these applications, blockchains serve as a data storage and management system for storing the data securely and providing a gateway for data analysis. In other words, blockchains can be viewed as a system that complements cloud services.

## 6 BFT RESEARCH IN THE PAST FOUR DECADES

### 6.1 Comparison of Different BFT Categories

We compare different categories of BFT protocols in Table 4. The most robust BFT consensus protocols are broadcast-based ones. Broadcast-based consensus protocols have high message complexity and communication complexity, i.e., typically $O(n^2)$. Following the broadcast based communication pattern, the message complexity can be reduced to $O(n)$ by letting one replica communicate with all other replicas at a time. Examples include SBFT and HotStuff, which preserve the three-phase workflow of PBFT but reduce the complexity to O(n) by letting one replica (or several replicas) collect the proof of agreement. The drawback is that the system state may have forks that need to be recovered after replicas detect them. Furthermore, the message complexity can be reduced via other types of message patterns such as chain-based BFT. In chain-based BFT, each replica communicates with one or few other replicas. Due to the use of pipelining workflow, the system throughput can be improved. The major drawback is also failure handling similar to other BFT protocols with $O(n)$ complexity. Trusted hardware can be used to reduce the requirements for the total number of replicas and enhance the efficiency of the protocol, making BFT has a CFT-like complexity and cost for replication. The major challenge is that the correctness of a protocol fully relies on the trusted hardware.

The scalability (server scalability) of the system can be enhanced via approaches such as sharding, e.g., parallel based BFT or hierarchical BFT. Note that asynchronous consensus protocols have even higher message complexity ($O(n^3)$ for most practical protocols) and communication complexity. Such protocols, however, allow replicas to propose transactions in parallel. Therefore, in some cases, asynchronous protocols may achieve higher throughput and better server scalability than partially synchronous protocols.

| Type of BFT | $n$ | Latency | Message Complexity | Communication Complexity | Additional requirements |
|---|---|---|---|---|---|
| Broadcast-based BFT | $3f+1$ | Low | High | High | None |
| Chain-based BFT | $3f+1$ | Medium | Low-Medium | Medium | None |
| Trusted hardware based BFT | $2f+1$ | Low | Medium | High | Trusted hardware |
| Hybrid BFT | $f+1$ to $3f+1^*$ | Low | Medium | High | Failures are infrequent |
| Scalable BFT | $3f+1$ | High | High | High | Modified failure model |
| Asynchronous BFT | $3f+1^*$ | Medium | High | High | None |

Table 4. Comparison of BFT-based consensus. *Some protocols may require different $n$.

## 6.2 BFT with the Rise of Blockchains

With the rise of blockchains, the needs to optimize BFT solutions and develop new protocols have evolved rapidly. This also explains the trend in some recent development. We identify several trends of BFT research driven by the needs of blockchains.

**The choice of cryptography.** Conventional BFT protocols are built on top of digital signatures and hashes. A lot of efforts have been made to replace digital signatures with MACs for reducing the computational cost [65, 118, 144]. This is mainly because the cost of cryptographic operations may dominate the overhead. Such an approach, due to the non-transferability of the MACs, may create problems such as big MAC attacks by the clients [75]. Due to the evolution of modern computer hardware, the cost for cryptographic operations becomes much lower. With the rise of blockchains, BFT is usually deployed in the WAN network, making the cost of cryptographic operations almost *negligible*. Furthermore, the transferrable authentication of digital signatures is important in blockchains for the verification of transaction history. Accordingly, other cryptographic choices such as threshold signatures and aggregate signatures are introduced for verification. Furthermore, as asynchronous BFT becomes more practical, it introduces an interesting observation: a BFT can be built by simply assuming an authenticated channel and some global common coin protocol, which can be built from approaches such as threshold PRF [101]. To further achieve other security goals, one could also integrate BFT with approaches such as symmetric encryption to achieve confidentiality [102] or threshold encryption to prevent censorship attacks [178].

In some application areas of blockchains, the anonymity of users and confidentiality of the data become extremely important, which require the design of new BFT approaches, integration with other cryptographic primitives, or novel system architecture [26, 71, 112, 113]. For instance, approaches such as ring signatures have recently been integrated with vector consensus [55], a weaker primitive than atomic broadcast (i.e., BFT); threshold cryptography has been used in secure causal atomic broadcast [58, 100]. The practicability of the approach and the applicability to blockchains, however, are yet to be discovered.

**The choice of simplicity.** Distributed SMR protocols have been known to be difficult to understand and implement, even for CFT protocols [67]. For instance, RAFT protocol (a CFT protocol) takes *understandability* as the first-class goal in the design [187]. BFT, in contrast, is even more difficult to understand and implement. When practical BFT was first studied, a lot of efforts were made to reduce the complexity during normal-case operation. For instance, hybrid BFT switches between different BFT instances to enjoy high performance in failure-free cases and resilience under failures [37, 97, 118, 133, 144, 186]. This is usually achieved at the cost of switching between different instances or higher cost to elect a new leader.

Due to the difficulty of implementing BFT correctly, recent trends tend to accept *simpler* BFT design. For instance, HotStuff uses only one message type to manage the entire BFT protocol [230]. The Streamlet protocol also targets a simple design for BFT [66], although it was discovered that there is a gap between Streamlet and BFT SMR [1]. Furthermore, in contrast to partially synchronous BFT, the algorithms of asynchronous BFT are also simpler since asynchronous BFT does not involve subprotocols such as view changes. The practicality in the real system, however, is yet to be discovered.

**The needs of scalability.** BFT SMR was believed to be useful only in a small network, as the case for CFT. As a result, BFT was usually tested in small networks with $f$ smaller than 5 (a maximum of 16 replicas), unless the protocols were designed to enhance the server scalability. In the blockchain setting, the system is usually deployed in a WAN network with a relatively large set of replicas. Therefore, it is desirable to achieve a balance between the system performance and server scalability [220]. As a result, in recent works on BFT, server scalability has become *de facto* study when assessing the performance and most works evaluate the system throughput using at least 100 replicas in WAN [101, 119, 120, 178, 197, 212, 230].

## 6.3 When to Choose Which?

| Use Case | Permission | Security | Performance | Client Scalability | Server Scalability |
|---|---|---|---|---|---|
| Finance | Yes/No | High | High | Medium-High | Medium |
| Supply Chain Management | Yes | High | Medium-High | High | Medium-High |
| Biomedical and Healthcare | Yes | High | Medium | Medium-High | Low-Medium |
| Critical Infrastructures | Yes | High | Medium | Medium | Low-Medium |
| Social and Education | Yes | Medium-High | Low-Medium | Low-Medium | Low-Medium |

Table 5. Summary of the use cases, the need of permission to build a blockchain-based solution, and the requirements for security, performance, server scalability, and server scalability.

We summarize and compare the use cases in Table 5 according to five different criteria. First, for each use case, whether a permissioned or permissionless system is more appropriate. Second, we summarize the security requirement for each use case, i.e., how critical are the corresponding data in each use case? Third, we evaluate the performance requirements for the use cases, in particular throughput. We evaluate the requirements according to the volume of data/transactions for most use cases in the category. Last but not least, what are the requirements for client scalability and server scalability? For most use cases, there are multiple applications where the workloads and security requirements are varying. We summarize the general requirements for use cases in different categories.

Certain financial applications utilize cryptocurrencies as a blockchain solution for payments, asset management, etc. Therefore, they can be built as a permissionless system. Besides, most of these applications need permission to build a distributed system, as the management of data has to be done by authorized users. Furthermore, since the major contribution of blockchains is to provide high availability of the service and integrity of the data,

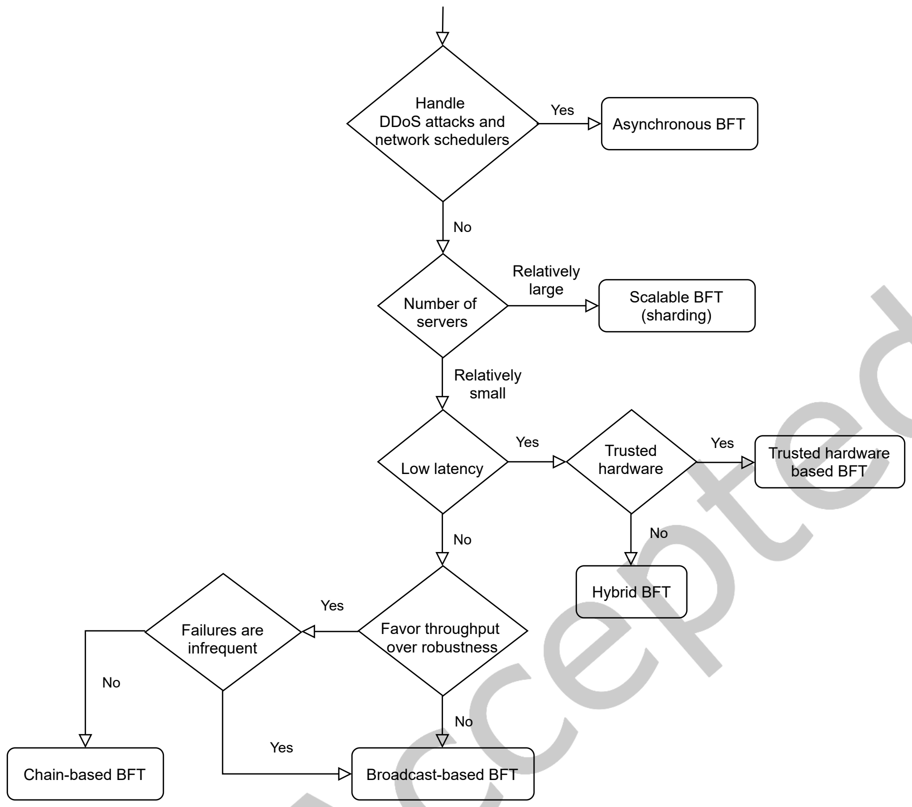


Fig. 9. Which BFT category should one choose? (Note: 'Relatively large' means over 200 replicas.)

blockchains are usually explored in mission-critical systems where there is a strong need for data security. Finally, the requirements for system performance (i.e., high throughput, low latency, and high scalability) depend on the scale of the system and the specific applications. In general, financial applications and supply chain management involve a large number of transactions, which require a system with high throughput and great scalability. In comparison, other use cases may not have a large number of concurrent transactions such as healthcare and critical infrastructures. Instead, it might be desirable to provide additional security features such as the confidentiality of user data.

It has become a common belief that there is no one-size-fits-all BFT protocol. We also illustrate in Fig. 9 a decision tree on which BFT category is the *best fit* for a given workload and system expectation. The decision tree can be viewed as a rough reference for *choosing* the right type of protocol while a lot decision details are not fully (and unfortunately impossible to be) captured. The foremost question to ask is whether the system may suffer from network scheduler attacks or unstable network conditions. If this is the case, one may choose to use asynchronous BFT. The cost is higher message complexity and relatively longer latency than BFT in other categories. If a fully asynchronous network and the resilience under extreme network conditions are not major concerns, one has to consider the number of servers that need to be supported. If server scalability is desirable, scalable BFT such as hierarchical BFT or parallel BFT might be considered. Scalable BFT can be combined with sharding to further enhance the scalability and performance by fulling utilizing parallelism. If server scalability is not needed, one may need to consider the needs for latency (under low concurrency of client requests). If low latency is required, trusted hardware based BFT or hybrid BFT can be used since protocols in these two categories optimize the workflow in the normal operation. If one favors throughput over latency, broadcast-based BFT or chain-based BFT can be considered. The difference between broadcast-based BFT and chain-based BFT is that broadcast-based BFT usually achieves a balance between latency, throughput, and robustness under failures, while chain-based BFT takes more expensive procedures to resume normal operation after failures than broadcast-based BFT.

## 7 CONCLUSION

This article reviews the Byzantine fault-tolerant protocols, with an emphasis on the application in blockchains, including permissioned and hybrid blockchains. We categorize the BFT protocols according to their types and message patterns, review their design, as well as provide insights into the BFT mechanisms. We also summarize the BFT approaches, compare and contrast their features. Finally, we review the real-world use cases and the BFT approaches adopted by real systems. By reviewing the development of BFT in blockchains and comparing the use cases, we summarize the research trend with the rise of blockchains, outlook the development of BFT, and discuss the challenges that need to be addressed.

## 8 ACKNOWLEDGMENT

Sisi Duan's work was supported in part by National Key Research and Development Program of China under grant No. 2018YFA0704701 and Shandong Key Research and Development Program under grant No. 2020ZLYS09.

## REFERENCES

[1] What they did not teach you in streamlet=. https://dahliamalkhi.github.io/posts/2020/12/what-they-didnt-teach-you-in-streamlet/.
[2] Bug in ABA protocol's use of common coin. https://github.com/amiller/HoneyBadgerBFT/issues/59, 2018.
[3] The policy environment for Blockchain innovation and adoption. 2019 OECD global Blockchain policy forum summary report. https://www.oecd.org/finance/2019-OECD-Global-Blockchain-Policy-Forum-Summary-Report.pdf, 2019.
[4] Diem. https://www.diem.com/en-us/, 2020.
[5] Enterprise-ready distributed ledger and smart contract platforms. https://github.com/jpmorganchase/quorum, 2020.
[6] Hyperledger burrow. https://github.com/hyperledger/burrow, 2020.
[7] Hyperledger iroha. https://github.com/hyperledger/iroha, 2020.
[8] Id2020. https://id2020.org/, 2020.
[9] Kadena. http://kadena.io/, 2020.
[10] Openidl. https://aaisonline.com/openidl, 2020.
[11] Platform identity management netherlands. http://www.pimn.nl/, 2020.
[12] Poa network. https://www.poa.network/, 2020.
[13] The red belly blockchain. https://gramoli.redbellyblockchain.io/web/doc/talks/facebook.pdf, 2020.
[14] Ripplenet. https://ripple.com/, 2020.
[15] Symbiont assembly. https://symbiont.io/technology, 2020.
[16] Tendermint core. https://github.com/tendermint/tendermint, 2020.
[17] Nyan cat. https://foundation.app/NyanCat/nyan-cat-219, 2021.
[18] Abd-El-Malek, M., Ganger, G. R., Goodson, G. R., Reiter, M. K., and Wylie, J. J. Fault-scalable Byzantine fault-tolerant services. *ACM SIGOPS Operating Systems Review 39*, 5 (2005), 59–74.
[19] Abraham, I., Devadas, S., Dolev, D., Nayak, K., and Ren, L. Synchronous Byzantine agreement with expected $o(1)$ rounds, expected $o(n^2)$ communication, and optimal resilience. In *International Conference on Financial Cryptography and Data Security* (2019), Springer, pp. 320–334.
[20] Abraham, I., Gueta, G., Malkhi, D., and Martin, J.-P. Revisiting fast practical byzantine fault tolerance: Thelma, velma, and zelma. *arXiv preprint arXiv:1801.10022* (2018).
[21] Abraham, I., Malkhi, D., et al. The blockchain consensus layer and BFT. *Bulletin of EATCS 3*, 123 (2017).
[22] Abraham, I., Malkhi, D., Nayak, K., Ren, L., and Spiegelman, A. Solida: A blockchain protocol based on reconfigurable Byzantine consensus. In *OPODIS* (2017).

[23] Abraham, I., Malkhi, D., Nayak, K., Ren, L., and Yin, M. Sync hotstuff: Simple and practical synchronous state machine replication. In *S&P* (2020).

[24] Abraham, I., Malkhi, D., and Spiegelman, A. Asymptotically optimal validated asynchronous Byzantine agreement. In *Proceedings of the Symposium on Principles of Distributed Computing* (2019), ACM, pp. 337–346.

[25] Alchieri, E., Dotti, F., Mendizabal, O. M., and Pedone, F. Reconfiguring parallel state machine replication. In *SRDS* (2017), IEEE, pp. 104–113.

[26] AlHaddad, N., Varia, M., and Zhang, H. High-threshold avss with optimal communication complexity. In *Financial Cryptography and Data Security* (Berlin, Heidelberg, 2021), N. Borisov and C. Diaz, Eds., Springer Berlin Heidelberg, pp. 479–498.

[27] Amir, Y., Coan, B., Kirsch, J., and Lane, J. Prime: Byzantine replication under attack. *IEEE Transactions on Dependable and Secure Computing 8*, 4 (2011), 564–577.

[28] Amir, Y., Danilov, C., Kirsch, J., Lane, J., Dolev, D., Nita-Rotaru, C., Olsen, J., and Zage, D. Scaling Byzantine fault-tolerant replication to wide area networks. In *DSN* (2006), IEEE, pp. 105–114.

[29] Amoretti, M., Brambilla, G., Medioli, F., and Zanichelli, F. Blockchain-based proof of location. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 146–153.

[30] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *EuroSys* (2018), ACM, p. 30.

[31] Androulaki, E., Cachin, C., Dobre, D., and Vukolić, M. Erasure-coded Byzantine storage with separate metadata. In *OPODIS* (2014), Springer, pp. 76–90.

[32] Aublin, P.-L., Mokhtar, S. B., and Quéma, V. RBFT: redundant byzantine fault tolerance. In *ICDCS* (2013), IEEE, pp. 297–306.

[33] Babay, A., Schultz, J., Tantillo, T., and Amir, Y. Toward an intrusion-tolerant power grid: Challenges and opportunities. In *ICDCS* (2018), IEEE, pp. 1321–1326.

[34] Babay, A., Schultz, J., Tantillo, T., Beckley, S., Jordan, E., Ruddell, K., Jordan, K., and Amir, Y. Deploying intrusion-tolerant scada for the power grid. In *DSN* (2019), IEEE, pp. 328–335.

[35] Babay, A., Tantillo, T., Aron, T., Platania, M., and Amir, Y. Network-attack-resilient intrusion-tolerant scada for the power grid. In *DSN* (2018), IEEE, pp. 255–266.

[36] Bagaria, V. K., Kannan, S., Tse, D., Fanti, G. C., and Viswanath, P. Prism: Deconstructing the blockchain to approach physical limits. In *CCS* (2019), L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds., ACM, pp. 585–602.

[37] Bahsoun, J.-P., Guerraoui, R., and Shoker, A. Making BFT protocols really adaptive. In *IPDPS* (2015), IEEE, pp. 904–913.

[38] Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., and Danezis, G. Sok: Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies* (2019), ACM, pp. 183–198.

[39] Baudet, M., Ching, A., Chursin, A., Danezis, G., Garillot, F., Li, Z., Malkhi, D., Naor, O., Perelman, D., and Sonnino, A. State machine replication in the libra blockchain. *The Libra Assn., Tech. Rep* (2019).

[40] Ben-Or, M. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *PODC* (1983), ACM, pp. 27–30.

[41] Ben-Or, M., Kelmer, B., and Rabin, T. Asynchronous secure computations with optimal resilience. In *PODC* (1994), ACM, pp. 183–192.

[42] Berger, C., Reiser, H. P., Sousa, J., and Bessani, A. Resilient wide-area byzantine consensus using adaptive weighted replication. In *SRDS* (2019), IEEE, pp. 183–192.

[43] Berman, P., and Garay, J. A. Randomized distributed agreement revisited. In *FTCS* (1993), IEEE, pp. 412–419.

[44] Bessani, A., Alchieri, E., Sousa, J., Oliveira, A., and Pedone, F. From byzantine replication to blockchain: Consensus is only the beginning. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2020), IEEE, pp. 424–436.

[45] Bessani, A. N., Santos, M., Felix, J., Neves, N. F., and Correia, M. On the efficiency of durable state machine replication. In *ATC* (2013), A. Birrell and E. G. Sirer, Eds., pp. 169–180.

[46] Bezerra, C. E., Pedone, F., and Renesse, R. V. Scalable state-machine replication. In *DSN* (2014), IEEE, pp. 331–342.

[47] Boddy, M. Philippine government tech department signs deal with blockchain firm, 2019.

[48] Boldyreva, A. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *International Workshop on Public Key Cryptography* (2003), Springer, pp. 31–46.

[49] Bracha, G. An asynchronous [(n-1)/3]-resilient consensus protocol. In *Proceedings of the third annual ACM symposium on Principles of distributed computing* (1984), pp. 154–162.

[50] Bracha, G. Asynchronous Byzantine agreement protocols. *Information and Computation 75*, 2 (1987), 130–143.

[51] Brown, R. G., Carlyle, J., Grigg, I., and Hearn, M. Corda: an introduction. *R3 CEV, August 1* (2016), 15.

[52] Buchman, E. *Tendermint: Byzantine fault tolerance in the age of blockchains.* PhD thesis, 2016.

[53] Burrows, M. The chubby lock service for loosely-coupled distributed systems. In *OSDI* (2006), USENIX Association, pp. 335–350.

[54] Buterin, V., Reijsbergen, D., Leonardos, S., and Piliouras, G. Incentives in ethereum's hybrid casper protocol. In *ICBC* (2019), IEEE, pp. 236–244.

[55] Cachin, C., Collins, D., Crain, T., and Gramoli, V. Anonymity preserving byzantine vector consensus. In *European Symposium on Research in Computer Security* (2020), Springer, pp. 133–152.
[56] Cachin, C., Dobre, D., and Vukolić, M. Separating data and control: Asynchronous bft storage with 2t+ 1 data replicas. In *Symposium on Self-Stabilizing Systems* (2014), Springer, pp. 1–17.
[57] Cachin, C., Guerraoui, R., and Rodrigues, L. *Introduction to reliable and secure distributed programming*. Springer Science & Business Media, 2011.
[58] Cachin, C., Kursawe, K., Petzold, F., and Shoup, V. Secure and efficient asynchronous broadcast protocols. In *Annual International Cryptology Conference* (2001), Springer, pp. 524–541.
[59] Cachin, C., Kursawe, K., and Shoup, V. Random oracles in constantinople: Practical asynchronous Byzantine agreement using cryptography. *Journal of Cryptology 18*, 3 (2005), 219–246.
[60] Cachin, C., and Poritz, J. A. Secure intrusion-tolerant replication on the internet. In *DSN* (2002), IEEE, pp. 167–176.
[61] Cachin, C., and Tackmann, B. Asymmetric distributed trust. In *OPODIS* (2019), P. Felber, R. Friedman, S. Gilbert, and A. Miller, Eds., vol. 153, pp. 7:1–7:16.
[62] Cachin, C., and Vukolić, M. Blockchain consensus protocols in the wild. In *DISC* (2017), pp. 1:1–1:16.
[63] Cachin, C., and Zanolini, L. From symmetric to asymmetric asynchronous byzantine consensus. *arXiv preprint arXiv:2005.08795* (2020).
[64] Canetti, R., and Rabin, T. Fast asynchronous Byzantine agreement with optimal resilience. In *STOC* (1993), vol. 93, Citeseer, pp. 42–51.
[65] Castro, M., and Liskov, B. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS) 20*, 4 (2002), 398–461.
[66] Chan, B. Y., and Shi, E. Streamlet: Textbook streamlined blockchains. In *AFT* (2020), ACM, pp. 1–11.
[67] Chandra, T. D., Griesemer, R., and Redstone, J. Paxos made live: an engineering perspective. In *PODC* (2007), I. Gupta and R. Wattenhofer, Eds., ACM, pp. 398–407.
[68] Chevet, S. Blockchain technology and non-fungible tokens: Reshaping value chains in creative industries. *Available at SSRN 3212662* (2018).
[69] Chiu, G.-M., and Hsiao, C.-M. A note on total ordering multicast using propagation trees. *IEEE Transactions on Parallel and Distributed Systems 9*, 2 (1998), 217–223.
[70] Chiu, J., and Koeppl, T. V. Blockchain-based settlement for asset trading. *The Review of Financial Studies 32*, 5 (2019), 1716–1753.
[71] Chow, S. S., Zhang, H., and Zhang, T. Real hidden identity-based signatures. In *International Conference on Financial Cryptography and Data Security* (2017), Springer, pp. 21–38.
[72] Chun, B.-G., Maniatis, P., Shenker, S., and Kubiatowicz, J. Attested append-only memory: making adversaries stick to their word. In *SOSP* (2007), pp. 189–204.
[73] Clavin, J., Duan, S., Zhang, H., Janeja, V. P., Joshi, K. P., Yesha, Y., Erickson, L. C., and Li, J. D. Blockchains for government: Use cases and challenges. *Digital Government: Research and Practice 1*, 3 (2020), 1–21.
[74] Clavin, J. R., Prakash, P. M., and Duan, S. Byzgame: byzantine generals game. In *DEBS* (2020), J. Gascon-Samson, K. Zhang, K. Daudjee, and B. Kemme, Eds., ACM, pp. 137–140.
[75] Clement, A., Junqueira, F., Kate, A., and Rodrigues, R. On the (limited) power of non-equivocation. In *PODC* (2012), pp. 301–308.
[76] Clement, A., Wong, E. L., Alvisi, L., Dahlin, M., and Marchetti, M. Making Byzantine fault tolerant systems tolerate Byzantine faults. In *NSDI* (2009), vol. 9, pp. 153–168.
[77] Copeland, C., and Zhong, H. Tangaroa: a byzantine fault tolerant raft. Tech. rep., 2016.
[78] Correia, M., Neves, N. F., and Verissimo, P. How to tolerate half less one Byzantine nodes in practical distributed systems. In *SRDS* (2004), IEEE, pp. 174–183.
[79] Correia, M., Neves, N. F., and Veríssimo, P. From consensus to atomic broadcast: Time-free Byzantine-resistant protocols without signatures. *The Computer Journal 49*, 1 (2006), 82–96.
[80] Correia, M., Veronese, G. S., Neves, N. F., and Verissimo, P. Byzantine consensus in asynchronous message-passing systems: a survey. *International Journal of Critical Computer-Based Systems 2*, 2 (2011), 141–161.
[81] Correia, M., VerÃŋssimo, P., and Neves, N. F. The design of a cots real-time distributed security kernel. In *EDCC* (2001).
[82] Cowling, J., Myers, D., Liskov, B., Rodrigues, R., and Shrira, L. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance. In *OSDI* (2006), USENIX Association, pp. 177–190.
[83] Crain, T., Gramoli, V., Larrea, M., and Raynal, M. Dbft: Efficient Byzantine consensus with a weak coordinator and its application to consortium blockchains. *arXiv preprint arXiv:1702.03068* (2017).
[84] Dai, J., and Vasarhelyi, M. A. Toward blockchain-based accounting and assurance. *Journal of Information Systems 31*, 3 (2017), 5–21.
[85] Damgard, I., Desmedt, Y., Fitzi, M., and Nielsen, J. B. Secure protocols with asymmetric trust. In *ASIACRYPT* (2007).
[86] Das, A., Gupta, I., and Motivala, A. Swim: Scalable weakly-consistent infection-style process group membership protocol. In *DSN* (2002), IEEE, pp. 303–312.

[87] David, B., Gazi, P., Kiayias, A., and Russell, A. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT* (2018), J. B. Nielsen and V. Rijmen, Eds., vol. 10821, Springer, pp. 66–98.

[88] De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., and Sassone, V. Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain. In *Italian Conference on Cyber Security* (2018).

[89] Decker, C., Seidel, J., and Wattenhofer, R. Bitcoin meets strong consistency. In *ICDCN* (2016), ACM, pp. 1–10.

[90] Demirguc-Kunt, A., Klapper, L., Singer, D., Ansar, S., and Hess, J. *The Global Findex Database 2017: Measuring financial inclusion and the fintech revolution.* The World Bank, 2018.

[91] Dettoni, F., Lung, L. C., Correia, M., and Luiz, A. F. Byzantine fault-tolerant state machine replication with twin virtual machines. In *ISCC* (2013), pp. 398–403.

[92] Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., and Wang, J. Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering 30*, 7 (2018), 1366–1385.

[93] Distler, T. Byzantine fault-tolerant state-machine replication from a systems perspective. *ACM Comput. Surv. 54*, 1 (2021), 24:1–24:38.

[94] Dolev, D., and Strong, H. R. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing 12*, 4 (1983), 656–666.

[95] Drijvers, M., Edalatnejad, K., Ford, B., Kiltz, E., Loss, J., Neven, G., and Stepanovs, I. On the security of two-round multi-signatures. In *Security and Privacy* (2019), pp. 1084–1101.

[96] Duan, S., Levitt, K., Meling, H., Peisert, S., and Zhang, H. ByzID: Byzantine fault tolerance from intrusion detection. In *SRDS* (2014), IEEE, pp. 253–264.

[97] Duan, S., Li, Y., and Levitt, K. Cost sensitive moving target consensus. In *NCA* (2016), IEEE, pp. 272–281.

[98] Duan, S., Meling, H., Peisert, S., and Zhang, H. BChain: Byzantine replication with high throughput and embedded reconfiguration. In *OPODIS* (2014), pp. 91–106.

[99] Duan, S., Peisert, S., and Levitt, K. N. hbft: speculative byzantine fault tolerance with minimum cost. *IEEE Transactions on Dependable and Secure Computing 12*, 1 (2014), 58–70.

[100] Duan, S., Reiter, M. K., and Zhang, H. Secure causal atomic broadcast, revisited. In *DSN* (2017), IEEE, pp. 61–72.

[101] Duan, S., Reiter, M. K., and Zhang, H. BEAT: Asynchronous bft made practical. In *CCS* (2018), ACM, pp. 2028–2041.

[102] Duan, S., and Zhang, H. Practical state machine replication with confidentiality. In *SRDS* (2016), IEEE, pp. 187–196.

[103] Dwork, C., Lynch, N., and Stockmeyer, L. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM) 32*, 2 (1988), 288–323.

[104] Eischer, M., and Distler, T. Scalable Byzantine fault tolerance on heterogeneous servers. In *EDCC* (2017), IEEE, pp. 34–41.

[105] Eyal, I., Gencer, A. E., Sirer, E. G., and Van Renesse, R. Bitcoin-NG: A scalable blockchain protocol. In *NSDI* (2016), pp. 45–59.

[106] Feldman, P., and Micali, S. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing 26*, 4 (1997), 873–933.

[107] Fischer, M. J., and Lynch, N. A. A lower bound for the time to assure interactive consistency. Tech. rep., GEORGIA INST OF TECH ATLANTA SCHOOL OF INFORMATION AND COMPUTER SCIENCE, 1981.

[108] Fischer, M. J., Lynch, N. A., and Paterson, M. S. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM) 32*, 2 (1985), 374–382.

[109] Fitzi, M., and Garay, J. A. Efficient player-optimal protocols for strong and differential consensus. In *PODC* (2003), pp. 211–220.

[110] Food, U., and Administration, D. Outbreak investigation of E. coli O157:H7 linked to romaine lettuce grown in CA, 2019.

[111] Foundation, T. W. W. From bait to plate: Preventing illegally caught seafood from entering our food chain, 2020.

[112] Franklin, M., and Zhang, H. Unique group signatures. In *European Symposium on Research in Computer Security* (2012), Springer, pp. 643–660.

[113] Franklin, M., and Zhang, H. Unique ring signatures: A practical construction. In *International Conference on Financial Cryptography and Data Security* (2013), Springer, pp. 162–170.

[114] Fynn, E., Bessani, A., and Pedone, F. Smart contracts on the move. In *DSN* (2020).

[115] Garcia-Molina, H., and Spauster, A. Ordered and reliable multicast communication. *ACM Transactions on Computer Systems (TOCS) 9*, 3 (1991), 242–271.

[116] Ghemawat, S., Gobioff, H., and Leung, S.-T. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles* (2003), pp. 29–43.

[117] Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *SOSP* (2017), ACM, pp. 51–68.

[118] Guerraoui, R., Knežević, N., Quéma, V., and Vukolić, M. The next 700 BFT protocols. *ACM Transactions on Computer Systems 32*, 4 (2015), 12:1–12:45.

[119] Gueta, G. G., Abraham, I., Grossman, S., Malkhi, D., Pinkas, B., Reiter, M. K., Seredinschi, D.-A., Tamir, O., and Tomescu, A. SBFT: a scalable decentralized trust infrastructure for blockchains. In *DSN* (2019), pp. 568–580.

[120] Guo, B., Lu, Z., Tang, Q., Xu, J., and Zhang, Z. Dumbo: Faster asynchronous bft protocols. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020).

[121] Hasan, H. R., and Salah, K. Proof of delivery of digital assets using blockchain and smart contracts. *IEEE Access 6* (2018), 65439–65448.
[122] Hendricks, J., Sinnamohideen, S., Ganger, G. R., and Reiter, M. K. Zzyzx: Scalable fault tolerance through byzantine locking. In *DSN* (2010), IEEE, pp. 363–372.
[123] Huh, S., Cho, S., and Kim, S. Managing iot devices using blockchain platform. In *ICACT* (2017), IEEE, pp. 464–467.
[124] Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. Zookeeper: Wait-free coordination for internet-scale systems. In *USENIX annual technical conference* (2010), vol. 8, Boston, MA, USA.
[125] Hyperledger. Case study: How walmart brought unprecedented transparency to the food supply chain with hyperledger fabric, 2020.
[126] Jacobovitz, O. Blockchain for identity management. *The Lynne and William Frankel Center for Computer Science Department of Computer Science. Ben-Gurion University, Beer Sheva* (2016).
[127] Jia, X. A total ordering multicast protocol using propagation trees. *IEEE transactions on parallel and distributed systems 6*, 6 (1995), 617–627.
[128] John, S. Blockchain in media and entertainment market report 2021 recent development and trends, expected growth and its factors, cagr, industry size, business prospects and forecast 2024, says kenneth research, 2021.
[129] Joonsang Baek, and Yuliang Zheng. Simple and efficient threshold cryptosystem from the gap diffie-hellman group. In *GLOBECOM* (2003).
[130] Junqueira, F. P., Reed, B. C., and Serafini, M. Zab: High-performance broadcast for primary-backup systems. In *DSN* (2011).
[131] Kalodner, H., Goldfeder, S., Chen, X., Weinberg, S. M., and Felten, E. W. Arbitrum: Scalable, private smart contracts. In *USENIX Security* (2016).
[132] Kang, H., Kim, H. R., and Hong, S.-p. A study on the design of smart contracts mechanism based on the blockchain for anti-money laundering. *Journal of Internet Computing and Services 19*, 5 (2018), 1–11.
[133] Kapitza, R., Behl, J., Cachine, C., Distler, T., Kuhnle, S., Mohammadi, S. V., Schröder-Preikschat, W., and Stengel, K. CheapBFT: Resource-efficient Byzantine fault tolerance. In *EuroSys* (2012), pp. 295–308.
[134] Kapritsos, M., Wang, Y., Quema, V., Clement, A., Alvisi, L., and Dahlin, M. All about eve: execute-verify replication for multi-core servers. In *OSDI* (2012), pp. 237–250.
[135] Katz, J., and Koo, C.-Y. On expected constant-round protocols for byzantine agreement. In *Annual International Cryptology Conference* (2006), Springer, pp. 445–462.
[136] Kerber, T., Kiayias, A., Kohlweiss, M., and Zikas, V. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *Security and Privacy (SP)* (2019), IEEE, pp. 157–174.
[137] Khan, M. A., and Salah, K. Iot security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems 82* (2018), 395–411.
[138] Kiayias, A., Russell, A., David, B., and Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO* (2017), J. Katz and H. Shacham, Eds., vol. 10401, Springer, pp. 357–388.
[139] Kim, M., Hilton, B., Burks, Z., and Reyes, J. Integrating blockchain, smart contract-tokens, and iot to design a food traceability solution. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 335–340.
[140] King, V., and Saia, J. Breaking the $o(n^2)$ bit barrier: scalable Byzantine agreement with an adaptive adversary. *Journal of the ACM (JACM) 58*, 4 (2011), 18.
[141] Kogias, E. K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., and Ford, B. Enhancing bitcoin security and performance with strong consistency via collective signing. In *USENIX Security* (2016), pp. 279–296.
[142] Kokina, J., Mancha, R., and Pachamanova, D. Blockchain: Emergent industry adoption and implications for accounting. *Journal of Emerging Technologies in Accounting 14*, 2 (2017), 91–100.
[143] Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., and Ford, B. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 583–598.
[144] Kolta, R., Alvisi, L., Dahlin, M., Clement, A., and Wong, E. Zyzzyva: speculative byzantine fault tolerance. *ACM Transactions on Computer Systems 27*, 4 (2009), 7:1–7:39.
[145] Kotla, R., and Dahlin, M. High throughput byzantine fault tolerance. In *DSN* (2004), IEEE, pp. 575–584.
[146] Kreps, J., Narkhede, N., Rao, J., et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB* (2011), vol. 11, pp. 1–7.
[147] Krishnakumar, N., and Bernstein, A. J. Bounded ignorance: A technique for increasing concurrency in a replicated system. *ACM Transactions on Database Systems (TODS) 19*, 4 (1994), 586–625.
[148] Lai, K. Blockchain as aml tool: a work in progress. *International Financial Law Review* (2018).
[149] Lamport, L. The part-time parliament. *ACM Transactions on Computer Systems (TOCS) 16*, 2 (1998), 133–169.
[150] Lamport, L., Shostak, R., and Pease, M. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS) 4*, 3 (1982), 382–401.
[151] Lemieux, V. L. Blockchain and distributed ledgers as trusted recordkeeping systems. In *Future Technologies Conference (FTC)* (2017), vol. 2017.

[152] Lev-Ari, K., Spiegelman, A., Keidar, I., and Malkhi, D. FairLedger: A fair blockchain protocol for financial institutions. In *OPODIS* (2019).
[153] Levin, D., Douceur, J. R., Lorch, J. R., and Moscibroda, T. TrInc: Small trusted hardware for large distributed systems. In *NSDI* (2009), pp. 1–14.
[154] Li, B., Xu, W., Abid, M. Z., Distler, T., and Kapitza, R. Sarek: Optimistic parallel ordering in byzantine fault tolerance. In *EDCC* (2016), IEEE, pp. 77–88.
[155] Li, W., Sforzin, A., Fedorov, S., and Karame, G. O. Towards scalable and private industrial blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts* (2017), pp. 9–14.
[156] Li, X., Wu, X., Pei, X., and Yao, Z. Tokenization: open asset protocol on blockchain. In *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, pp. 204–209.
[157] Li, Z., Lu, Q., Chen, S., Liu, Y., and Xu, X. A landscape of cryptocurrencies. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 165–166.
[158] Li, Z., Van Roy, P., and Romano, P. Enhancing throughput of partially replicated state machines via multi-partition operation scheduling. In *NCA* (2017), IEEE, pp. 1–10.
[159] Libert, B., Joye, M., and Yung, M. Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares. *Theoretical Computer Science 645* (2016), 1–24.
[160] Liu, C., Duan, S., and Zhang, H. EPIC: efficient asynchronous BFT with adaptive security. In *DSN* (2020), IEEE, pp. 437–451.
[161] Liu, C., Duan, S., and Zhang, H. Mib: Asynchronous BFT with more replicas. *CoRR abs/2108.04488* (2021).
[162] Liu, J., Li, W., Karame, G., and Asokan, N. Scalable Byzantine consensus via hardware-assisted secret sharing. *IEEE Transactions on Computers* (2018).
[163] Liu, Z., Xiang, Y., Shi, J., Gao, P., Wang, H., Xiao, X., Wen, B., and Hu, Y.-C. Hyperservice: Interoperability and programmability across heterogeneous blockchains. In *CCS* (2019).
[164] Loss, J., and Moran, T. Combining asynchronous and synchronous Byzantine agreement: The best of both worlds. *IACR Cryptology ePrint Archive 2018* (2018), 235.
[165] Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., and Saxena, P. A secure sharding protocol for open blockchains. In *CCS* (2016), ACM, pp. 17–30.
[166] MacBrough, E. Cobalt: Bft governance in open networks. *arXiv preprint arXiv:1802.07240* (2018).
[167] Malkhi, D., and Reiter, M. Byzantine quorum systems. *Distributed computing 11*, 4 (1998), 203–213.
[168] Marandi, P. J., Bezerra, C. E., and Pedone, F. Rethinking state-machine replication for parallelism. In *ICDCS* (2014), IEEE, pp. 368–377.
[169] Marandi, P. J., and Pedone, F. Optimistic parallel state-machine replication. In *SRDS* (2014), IEEE, pp. 57–66.
[170] Marandi, P. J., Primi, M., and Pedone, F. High performance state-machine replication. In *DSN* (2011), IEEE, pp. 454–465.
[171] Martin, J.-P., and Alvisi, L. Fast byzantine consensus. *IEEE Transactions on Dependable and Secure Computing 3*, 3 (2006), 202–215.
[172] Matters, G. Hhs obtains first blockchain ATO in federal government, 2018.
[173] Mazieres, D. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation 32* (2015).
[174] Melendez, S. How ibm and the cdc are testing blockchain to track health issues like the opioid crisis. *Fast Company 4* (2018).
[175] Merideth, M. G., and Reiter, M. K. Selected results from the latest decade of quorum systems research. In *Replication*. Springer, 2010, pp. 185–206.
[176] Mettler, M. Blockchain technology in healthcare: The revolution starts here. In *Healthcom* (2016), IEEE, pp. 1–3.
[177] Miah, S. Japan is fast becoming the gold standard for blockchain adoption, 2019.
[178] Miller, A., Xia, Y., Croman, K., Shi, E., and Song, D. The honey badger of BFT protocols. In *CCS* (2016), ACM, pp. 31–42.
[179] Moniz, H., Neves, N. F., and Correia, M. Byzantine fault-tolerant consensus in wireless ad hoc networks. *IEEE Transactions on Mobile Computing 12*, 12 (2012), 2441–2454.
[180] Moniz, H., Neves, N. F., Correia, M., and Verissimo, P. Ritas: Services for randomized intrusion tolerance. *IEEE transactions on dependable and secure computing 8*, 1 (2008), 122–136.
[181] Mostefaoui, A., Hamouma, M., and Raynal, M. Signature-free asynchronous Byzantine consensus with $t < n/3$ and $o(n^2)$ messages. In *PODC* (2014), ACM, pp. 2–9.
[182] Nakamoto, S., et al. Bitcoin: A peer-to-peer electronic cash system. Tech. rep., Working Paper, 2008.
[183] Natoli, C., Yu, J., Gramoli, V., and Esteves-Verissimo, P. Deconstructing blockchains: A comprehensive survey on consensus, membership and structure. *arXiv preprint arXiv:1908.08316* (2019).
[184] Nestle. Nestlé breaks new ground with open blockchain pilot, 2019.
[185] Nguyen, G.-T., and Kim, K. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems 14*, 1 (2018).
[186] Nikolaou, S., and van Renesse, R. Turtle consensus: Moving target defense for consensus. In *Middleware* (2015), pp. 185–196.
[187] Ongaro, D., and Ousterhout, J. In search of an understandable consensus algorithm. In *ATC* (2014), pp. 305–319.
[188] Partz, H. Major South Korean city to build blockchain-enabled virtual power plant, 2018.

[189] Pass, R., and Shi, E. Fruitchains: A fair blockchain. In *the ACM Symposium* (2017).

[190] Pass, R., and Shi, E. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC* (2017).

[191] Pass, R., and Shi, E. Thunderella: blockchains with optimistic instant confirmation. In *EUROCRYPT* (2018), Springer, pp. 3–33.

[192] Pedone, F., and Schiper, A. Handling message semantics with generic broadcast protocols. *Distributed Computing 15*, 2 (2002), 97–107.

[193] Platania, M., Obenshain, D., Tantillo, T., Amir, Y., and Suri, N. On choosing server-or client-side solutions for bft. *ACM Computing Surveys 48*, 4 (2016), 61.

[194] Redmayne, J. From bait to plate: Blockchain platform platform tracks food's journey, 2019.

[195] Rennock, M. J., Cohn, A., and Butcher, J. R. Blockchain technology and regulatory investigations. *The Journal 1* (2018), 35–45.

[196] Review, A. B. Malaysia's Melaka Straits city to become world's first blockchain city, 2019.

[197] Rocket, T., Yin, M., Sekniqi, K., van Renesse, R., and Sirer, E. G. Scalable and probabilistic leaderless bft consensus through metastability. *arXiv preprint arXiv:1906.08936* (2019).

[198] Rodrigues, R., and Liskov, B. Rosebud: A scalable Byzantine-fault-tolerant storage architecture. Tech. rep., MIT, 2003.

[199] Ross, B. Us health and human services looks to blockchain to manage unstructured data, 2018.

[200] Santos, N., and Schiper, A. Achieving high-throughput state machine replication in multi-core systems. In *ICDCS* (2013), IEEE, pp. 266–275.

[201] Shan, G., Zhao, B., Clavin, J. R., Zhang, H., and Duan, S. Poligraph: Intrusion-tolerant and distributed fake news detection system. *IEEE Transactions on Information Forensics and Security* (2021).

[202] Sherman, A. T., Javani, F., Zhang, H., and Golaszewski, E. On the origins and variations of blockchain technologies. *IEEE Security Privacy 17*, 1 (2019), 72–77.

[203] Shin, E.-J., Kang, H.-G., and Bae, K. A study on the sustainable development of npos with blockchain technology. *Sustainability 12*, 15 (2020), 6158.

[204] Shoup, V. Practical threshold signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques* (2000), Springer, pp. 207–220.

[205] Shoup, V., and Gennaro, R. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptol. 15*, 2 (2002), 75–96.

[206] Smith, M. In wake of romaine e. coli scare, walmart deploys blockchain to track leafy greens, 2019.

[207] Sompolinsky, Y., and Zohar, A. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security* (2015), R. Böhme and T. Okamoto, Eds., vol. 8975, Springer, pp. 507–527.

[208] Song, Y. J., and van Renesse, R. Bosco: One-step Byzantine asynchronous consensus. In *DISC* (2008), Springer, pp. 438–450.

[209] Sousa, J., Alchieri, E., and Bessani, A. State machine replication for the masses with bft-smart. In *DSN* (2014), pp. 355–362.

[210] Sousa, J., and Bessani, A. Separating the wheat from the chaff: An empirical design for geo-replicated state machines. In *SRDS* (2016).

[211] Spiegelman, A., and Rinberg, A. Ace: Abstract consensus encapsulation for liveness boosting of state machine replication. *arXiv preprint arXiv:1911.10486* (2019).

[212] Stathakopoulou, C., David, T., and Vukolic, M. Mir-bft: High-throughput bft for blockchains. *arXiv preprint arXiv:1906.05552* (2019).

[213] Syta, E., Tamas, I., Visher, D., Wolinsky, D. I., Jovanovic, P., Gasser, L., Gailly, N., Khoffi, I., and Ford, B. Keeping authorities "honest or bust" with decentralized witness cosigning. In *Security and Privacy* (2016).

[214] Toueg, S. Randomized byzantine agreements. In *PODC* (1984), ACM, pp. 163–178.

[215] Van Renesse, R., Ho, C., and Schiper, N. Byzantine chain replication. In *International Conference On Principles Of Distributed Systems* (2012), Springer, pp. 345–359.

[216] Van Renesse, R., and Schneider, F. B. Chain replication for supporting high throughput and availability. In *OSDI* (2004), vol. 4.

[217] Veronese, G. S., Correia, M., Bessani, A. N., and Lung, L. C. Spin one's wheels? byzantine fault tolerance with a spinning primary. In *SRDS* (2009).

[218] Veronese, G. S., Correia, M., Bessani, A. N., Lung, L. C., and Verissimo, P. Efficient Byzantine fault tolerance. *IEEE Transactions on Computers 62*, 1 (2013), 16–30.

[219] Vukolic, M. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *iNetSec* (2015), pp. 112–125.

[220] Vukolić, M. Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts* (2017), pp. 3–7.

[221] Wang, W., Hoang, D. T., Xiong, Z., Niyato, D., Wang, P., Hu, P., and Wen, Y. A survey on consensus mechanisms and mining management in blockchain networks. *arXiv preprint arXiv:1805.02707* (2018), 1–33.

[222] Wang, X., Zha, X., Ni, W., Liu, R. P., Guo, Y. J., Niu, X., and Zheng, K. Survey on blockchain for internet of things. *Computer Communications* (2019).

[223] Wang, Y., Alvisi, L., and Dahlin, M. Gnothi: Separating data and metadata for efficient and available storage replication. In *USENIX Annual Technical Conference* (2012), pp. 413–424.

[224] Wilkes, J., Hoover, C., Keer, B., Mehra, P., and Veitch, A. Storage, data, and information systems. Tech. rep., HP Laboratories, 2008.

[225] Will Martino. Kadena, The first scalable, high performance private blockchain. https://d31d887a-c1e0-47c2-aa51-c69f9f998b07.filesusr.com/ugd/86a16f_aeb9004965c34efd9c48993c4e63a9bb.pdf, 2016.

[226] Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper 151* (2014), 1–32.
[227] Xu, X., Weber, I., and Staples, M. *Architecture for blockchain applications.* Springer, 2019.
[228] Yasaweerasinghelage, R., Staples, M., and Weber, I. Predicting latency of blockchain-based systems using architectural modelling and simulation. In *International Conference on Software Architecture (ICSA)* (2017), IEEE, pp. 253–256.
[229] Yin, J., Martin, J.-P., Venkataramani, A., Alvisi, L., and Dahlin, M. Separating agreement from execution for Byzantine fault tolerant services. *ACM SIGOPS Operating Systems Review 37*, 5 (2003), 253–267.
[230] Yin, M., Malkhi, D., Reiterand, M., Gueta, G. G., and Abraham, I. Hotstuff: Bft consensus with linearity and responsiveness. In *PODC* (2019).
[231] Yu, H., Nikolic, I., Hou, R., and Saxena, P. OHIE: Blockchain scaling made simple. In *Security & Privacy* (2020), IEEE, pp. 90–105.
[232] Zamani, M., Movahedi, M., and Raykova, M. Rapidchain: A fast blockchain protocol via full sharding. In *CCS* (2018), pp. 931–948.
[233] Zhang, F., Cecchetti, E., Croman, K., Juels, A., and Shi, E. Town crier: An authenticated data feed for smart contracts. In *CCS* (2016).
[234] Zottel, S., Zia, B., and Khoury, F. *Enhancing financial capability and inclusion in Sénégal: A demand-side survey.* World Bank, 2016.