# Survey on Blockchain Networking: Context, State-of-the-Art, Challenges

MAYA DOTAN, Hebrew University, Israel
YVONNE-ANNE PIGNOLET, DFINITY, Switzerland
STEFAN SCHMID, Faculty of Computer Science, University of Vienna, Austria
SAAR TOCHNER, Hebrew University, Israel
AVIV ZOHAR, Hebrew University, Israel

Blockchains in general and cryptocurrencies such as Bitcoin in particular are realized using distributed systems and hence critically rely on the performance and security of the interconnecting network. The requirements on these networks and their usage, however can differ significantly from traditional communication networks, with implications on all layers of the protocol stack. This paper is motivated by these differences, and in particular by the observation that many fundamental design aspects of these networks are not well-understood today. In order to support the networking community to contribute to this emerging application domain, we present a structured overview of the field, from topology and neighbor discovery, over block and transaction propagation, to sharding and off-chain networks, also reviewing existing empirical results from different measurement studies. In particular, for each of these domains, we provide the context, highlighting differences and commonalities with traditional networks, review the state-of-the-art, and identify open research challenges. Our paper can hence also be seen as a call-to-arms to improve the foundation on top of which blockchains are built.

## 1 INTRODUCTION

Blockchain technologies allow mistrusting entities to cooperate in the absence of a trusted third party, by implementing a secure distributed ledger which is collectively managed by a peer-to-peer network. Blockchain technologies have recently received much attention in the context of cryptocurrencies such as Bitcoin or Ethereum, by providing means to exchange digital assets relying on strong cryptography. In contrast to centralized digital currencies and central banking systems, cryptocurrencies offer decentralized control, and accordingly, much existing research in the blockchain field focuses on the underlying cryptographic primitives and on improved distributed blockchain protocols, e.g., consensus.

The network required to connect the distributed system, however, has received relatively little attention. However, there is increasing evidence that the network can become the bottleneck and root-cause for some of the most pressing challenges cryptocurrencies specifically, and blockchains in general, face today. For example, the propagation of transactions and blocks (or other control messages for the execution of consensus algorithms), require unicast and multicast communication services. Blockchain miners which write transactions to the blockchain, are connected through dedicated miner P2P (peer-to-peer) networks [14], in addition to the public blockchain P2P network. Studies show that the cryptocurrency network layer is critical for scalability [24, 63], security [45] and privacy [44] of a blockchain, and that an efficient network layer enables higher transaction throughput and stronger resilience against malicious actors [45]. Networking issues are also not limited to overlays on the network layer. Besides node discovery and data routing, the provided network functionality

Authors' addresses: Maya Dotan, Hebrew University, Israel, MayaDotan@mail.huji.ac.il; Yvonne-Anne Pignolet, DFINITY, Switzerland, yvonneanne.pignolet@gmail.com; Stefan Schmid, Faculty of Computer Science, University of Vienna, Austria, schmiste@gmail.com; Saar Tochner, Hebrew University, Israel, saar.tochner@mail.huji.ac.il; Aviv Zohar, Hebrew University, Israel, avivz@cs.huji.ac.il.

for example includes the encoding and transmission of data, and error correction, as well as measurements of the network performance.

Interestingly, networks for blockchains and cryptocurrencies can differ significantly from traditional communication networks, also in terms of requirements and usage. For example, blockchain networks may come with different security requirements (e.g., related to anonymity), may need to serve different traffic mixes (e.g., more frequent broadcast of transactions and states of the blockchain), may need different routing mechanisms (e.g., source routing), or may be more dynamic (e.g., channels and fees in PCNs). Obviously, if in a cryptocurrency application no protocol participant can be trusted (not the users issuing transactions not the miners, and information providers may act maliciously), this requires a different design, and the incentives of all participants must be considered.

This paper is a call-to-arms to the networking community to identify the unique requirements of blockchain networks and address the open issues.

**Our Contributions.** This paper aims to provide a fast introduction to blockchain network issues with a focus on open research challenges. To this end, we provide a structured overview of blockchain networks, introducing the different aspects and their context, highlighting differences and commonalities with traditional networks, and reviewing the state-of-the-art. At the end of each section, we identify and discuss research questions. A main emphasis will be put on cryptocurrency applications specifically.

This paper hence targets junior and senior researchers with different backgrounds (e.g., in networking, algorithms, or game theory) who would like to get an overview of the state-of-the-art and start working in this area. It can also serve experts and decision-makers in the networking industry.

**Related Work.** This paper surveys blockchain network issues with a focus on research challenges. The most closely related papers in this area are surveys and systematizations of knowledge efforts on cryptocurrencies and blockchains in general [16]. Gudgeon et al. [49] provide a comprehensive survey of off-chain networks, and Neudecker et al. [81] survey the network layer of permissionless blockchains, with a focus on attacks and the design space (but less on research questions, e.g., revolving around incentives or mining). Katkuri presents a survey of data transfer and storage techniques in prevalent cryptocurrencies and suggests improvements [60]. The focus is on aspects related to the broadcast networks underlying Ethereum, Nano and IOTA. Gervais et al. [45] give a thorough security analysis of proof of work blockchain systems; their focus is on the consensus layer (i.e., block generation) whereas the network layer is abstracted. Troncoso et al. [105] show a broader perspective covering numerous systems apart from Bitcoin and Tor but also abstract from the network layer. A recent paper by Delgado-Seguara et al. [26] explores the characteristics of the P2P network established by Bitcoin, but abstracts from the design space of the network layer. Delgado et al. [26] provide an in-depth study of the Bitcoin P2P network. A preliminary and significantly shorter version of this article was presented in a conference [29]; the current version additionally covers offchain networks, networking and scalability aspects in sharding, as well as measurement studies, and includes more detailed and up-to-date discussions throughout the article.

**Organization.** The remainder of this paper is organized as follows (see Figure 1). Section 2 provides an overview of some basic aspects of blockchain networks. These include incentives, topology, communication pattern and security. Next, we discuss core aspects, namely block propagation, transaction propagation, P2P network topologies, sharding, off-chain networks, and empirical results in Sections 3–8. In each of these sections, we give some background

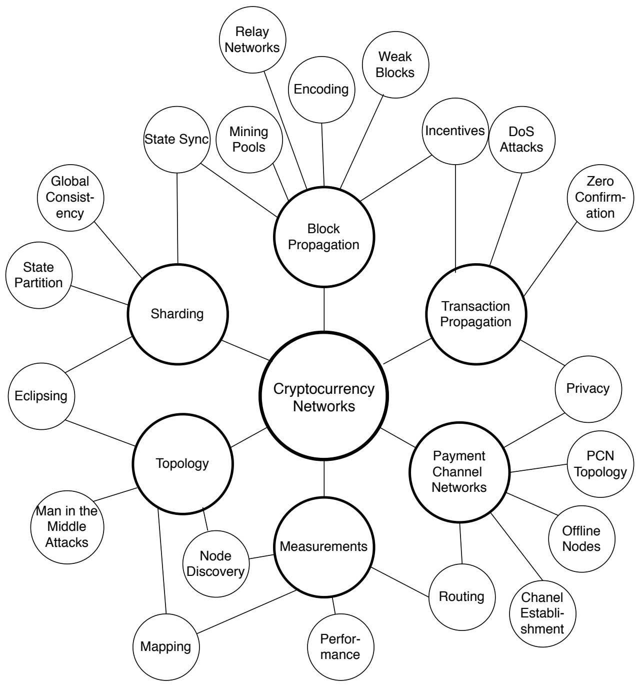

Fig. 1. Paper organization

information, present the state-of-the-art followed by open research questions that the writers of this paper find interesting. Finally, in Section 9 we conclude this survey.

## 2 BLOCKCHAIN NETWORKS IN PERSPECTIVE

### 2.1 Background

Before delving into the details, we provide some background and introduce preliminaries.

A **blockchain** is a distributed system realizing a decentralized ledger or "database", with the property that once data is added, it cannot be removed or altered.

A main application of blockchains arises in the context of **cryptocurrencies**, by permitting mutually distrusting parties to engage in financial operations securely. These systems guarantee that a transaction issued by Alice to send money to Bob reaches its destination at most once and only if Alice's balance is sufficiently high. Analogously, operations with multiple senders and receivers are typically supported. The guarantees hold despite Byzantine behavior of a fraction of the participants (nodes) maintaining the cryptocurrency service. In cryptocurrencies, the distributed ledger technology serves as a transaction database, containing´the global history with all transactions. To build this global history, typically consensus-based blockchain solutions are employed. The blockchain consists of a replicated linked list of immutable blocks comprising batches of transactions, and is maintained by a large number of nodes to tolerate malicious behavior of a small group of nodes and still reach agreement on the blocks and their content with a consensus protocol. An honest node will only propose and agree on blocks that contain valid transactions: i.e., the transaction is properly signed by the current owners of the funds, it has not been executed already, and

and the senders' balance is high enough.[1] Using blockchain technology, virtual currency can be transferred from senders to receivers in a fully distributed manner, cutting out any middle man or trusted third party. This feature has gained enormous visibility and is envisioned to transform the financial sector and potentially bring disruptive innovation to many other sectors that traditionally rely on trusted third parties.

More generally, blockchains can be used to implement a replicated state machine which tolerates Byzantine behaviors. To this end, the set of nodes maintaining the blockchain accept inputs by users to modify replicated state in an arbitrary fashion, relying on cryptographic techniques and performing repeated consensus on the input.

Blockchains in general and cryptocurrencies such as Bitcoin typically run on top of a **P2P network**. Over this network, nodes send and receive blocks and transactions, which are the basic data structures of cryptocurrencies. In permissionless protocols, such as Bitcoin and Ethereum, any machine can join the network and become a node of the P2P network. A node bootstraps its operation with a *discovery protocol* to establish connections with other nodes in the system.

In most cryptocurrencies there are two roles a node can assume: peer or miner. Peers can create and send transactions. Peers verify the correctness of received transactions and blocks and relay them if valid. Miners do anything a peer does, but they also generate blocks. Transactions and blocks are typically propagated in the network using a flooding or gossip protocol. E.g., when a node either creates or receives a transaction or block in Bitcoin, it announce that item and may request it from peers if receiving an announcement of an item it does not have yet. A node does not forward invalid items. A node keeps valid items in memory (the mempool) and answers requests for them. This way, each node in the network will eventually learn about every new item. The underlying wire protocol prescribes the data encoding and how to use which transport protocols. E.g., Bitcoin clients establish a TCP connection and perform a protocol-level three-way handshake informing each other of the height of the blockchain as they know it and the software version they use [56]. To support encryption and authentication, Ethereum defines the TCP-based DEVp2p protocol [35]. After a handshake, all exchanged messages are encrypted and authenticated via key material generated during the handshake.

In a later section of this paper, we provide a detailed discussion of block propagation and transaction propagation mechanisms and measurements. Furthermore, incentives to support reliable information forwarding and the different topologies of the virtual currency networks are presented.

Blockchains are typically managed by a P2P network of nodes which collectively create and validate new blocks. In order to improve scalability, additional payment channel networks may be implemented off-chain, offloading the blockchain. Such protocols rely on the "parent blockchain" for security. Interestingly, however, cryptocurrency networks and their applications and usage, can differ significantly from traditional communication networks. These differences influence the required performance, security, and incentives, and touch all layers of the network stack. Indeed, while early solutions relied on either a centralized issuer [94, 113] or creating inter-user credit [41], which required users to trust the original issuer; decentralized systems more critically depend on the *network* to connect their constituent parts. One main challenge is the fact that cryptocurrencies are a relatively

[1]To facilitate validity checks, many cryptocurrencies require outgoing transactions to link to a previous incoming transaction. Thus, an attempt to double-spend consists in getting the same transaction into multiple blocks that the receipients consider valid by mistake.

recent concept and many aspects are not well explored and documented, even compared to other components of blockchains and cryptocurrencies. With the exception of Bitcoin and Ethereum, many cryptocurrencies lack substantial documentation about their operational details, other than information scattered in the source code repositories. At the same time, the dependability of cryptocurrencies is becoming increasingly important, and vulnerabilities and inefficiencies are a major concern given the corresponding direct financial implications.

## 2.2 Characteristics

Before diving into the details of the state-of-the-art and research challenges, we give an overview of some of the distinguished characteristics of cryptocurrency networks.

A first important aspect concerns **incentives**. Since there is no central party or consortium paying for the resources (bandwidth, CPU, storage, ..) necessary to maintain the distributed ledger, the participating nodes must be remunerated through the protocol directly. Similarly to traditional P2P networks, nodes maintaining distributed ledgers require incentives to motivate nodes to propagate information (transactions, blocks, control information) between them. In addition, cryptocurrencies also need an incentive system to motivate nodes to verify blocks and the transactions included in them and discard invalid ones. In Bitcoin, nodes generating blocks are called miners. As a remuneration of their work the creator of block obtains a block reward and a fee for each transaction in the block. Thus, miners have an incentive to keep the knowledge of any transaction that offers a high fee to themselves instead of forwarding them, as any other node that becomes aware of the transaction will compete to include it in a block first and claim the associated fee. Additional incentive questions involve the use of lightweight nodes (know as SPV (simplified payment verification) wallets) that rely on messages from a full-node for their operation. SPV wallets do not hold a complete copy of the blockchain and so must rely on other nodes to track payments sent to them.

Another interesting characteristic is formed by the prevalent **communication pattern**. Many cryptocurrency networks are characterized by frequent broadcast operations, e.g., related to the communication of transactions and states of the blockchain. Furthermore, systems such as Bitcoin do not follow a complex multihop routing scheme but employ a simple flooding-based strategy where all peers in the network replicate the information that has been flowing through the system so far, i.e., keep a complete copy of the blockchain. Hence, there is no need to forward queries to other peers, as all information should to be available at a neighbor.

There are also differences in **routing** itself. For example, existing network routing algorithms for data transmission experience unique challenges when applied to payment channel networks. In payment channel networks, link capacities represent payment balances, which can be highly dynamic: messages are financial transactions, which may change liquidities and introduce additional security requirements (e.g., related to privacy), and different routes may be used at different monetary costs (e.g., as intermediate nodes charge fees for forwarding). If link capacities (representing funds) or "liquidities" should be kept private, it may become difficult to design an efficient route discover process [101].

The **network topology** in cryptocurrencies can be fairly different from traditional networks. For example, while Bitcoin and Ethereum rely on flat random graph topologies, Cardano [20] uses different roles influencing how nodes connect to each other and to users (see Section 5). Also in this regard, off-chain networks are particularly interesting. Since in these networks, capacities represent financial balances which may need to be kept confidential, new threats may be introduced which are not encountered in classic P2P networks.

In terms of **security** and dependability, cryptocurrencies critically depend on a correct functioning of the consensus layer, and the knowledge of the set of information consensus is to be agreed on (e.g., blocks and transactions). Flooding or gossip protocols are used for the propagation of the required information to all peers of the network. While unstructured P2P networks have been used for decades (e.g., Gnutella) and were extensively analyzed, the considered adversarial models do not match well the threats to blockchain systems. For example, anonymity providing networks (i.e., onion routing networks [46]) have different requirements regarding information propagation than blockchain based systems. Commonly considered requirements in anonymity providing networks are high performance, low bandwidth cost, resistance to traffic analysis, and resistance to denial of service (DoS). For example, distributed denial of service attacks can be used to gain advantages in mining, voting, and other business and protocol-related activities [4, 58, 108]. To prevent malicious nodes from flooding the network with invalid blocks, nodes use a store-and-forward propagation model, where each node downloads the full block and verifies it prior to propagating it to its peers. This model allows nodes to identify any node which propagates invalid blocks as malicious, and limits the effect of such attacks to the nodes which are directly attacked.

There are also implications on **performance**. While individual nodes may support high transaction rates, the distributed propagation can introduce novel kinds of bottlenecks. Indeed, one of the main issues in blockchain systems is their scalability. Increasing the number of transactions processed by the system naturally requires more resources such as bandwidth and storage, and especially the consensus protocols underlying cryptocurrencies can slow down execution, or even effect security. For example, Nakamoto's consensus protocol which relies on the longest chain for Bitcoin is known to be susceptible to attacks by weaker attackers as transaction throughputs increase. The main underlying cause of this decline in security is the fact that blocks containing more transactions propagate more slowly through the network, which causes forks to form in the blockchain [24]. As a result, a great deal of work has been devoted to improving block propagation times (see in Section 3). In order for Bitcoin to function as a decentralized system, it must allow nodes to receive blocks at a higher rate than the rate at which blocks are being produced. Indeed, if blocks are produced at a higher rate than a node is capable of receiving them, then said node cannot keep track of balances stored in the blockchain, cannot determine whether or not transactions and blocks are valid, and is in effect excluded from the Bitcoin network. The block propagation time to reach the majority of the network does not depend solely on a receiving node's bandwidth, but also on the network topology, the bandwidth of all nodes, and the manner in which blocks propagate. In off-chain networks, as mentioned also earlier, novel issues arise which lie at the intersection between performance and security. In off-chain networks, to protect user privacy, only the total capacity of a channel is disclosed, but not how the funds are distributed among the the channel participants [67, 96, 99]. Channel transactions might therefore fail and the routing algorithms attempt different execution paths until one succeeds, which can introduce delays. Routing algorithms in payment channels networks therefore, have to account for the unique characteristic of channels to provide satisfactory path recommendations.

## 3 BLOCK PROPAGATION

### 3.1 What is it about?

Blocks in cryptocurrency protocols are used to establish common state. They form the input the consensus protocol strives to reach agreement on. Blocks order transactions, thus the state of the network can be constructed by following the ordering of transactions included in blocks in the consensus chain. Transactions once included in a block deep in the consensus chain are considered confirmed. Therefore, block propagation is an issue of utmost importance to the consensus process. How fast miners learn about new blocks, and how quickly blocks can be created and validated are crucial for the efficiency of a cryptocurrency.

A block consists of a header and a set of transactions. These transactions can be relayed by the sender together with the block, but this wastes bandwidth if they are already stored in the mempool of the receiver.

Blocks are typically re-propagated to all connected peers as soon as basic validity of the announcement has been established (e.g. after the proof of work check). In Bitcoin, propagation uses the NewBlock and NewBlockHashes messages. The NewBlock message includes the full block and is sent to a small fraction of connected peers (usually the square root of the total number of peers). All other peers are sent a NewBlockHashes message containing just the hash of the new block. Those peers may request the full block later if they fail to receive it from anyone within reasonable time.

Blocks can be relayed with a compressed encoding. Efficient propagation of blocks is critical to achieving consensus, reducing storage bloat, overcoming network firewall bottlenecks, and allowing scaling to a large number of processed transactions per second (☛**Q1**). Delayed blocks can lead to forks [24]: based on measurements of the rate of information propagation in the network, the propagation delay in the network can be the primary cause for blockchain forks. Blockchain forks should be avoided as they are symptomatic for inconsistencies among the replicas in the network. As a mitigation strategy, the authors propose pipelining the block's delivery, i.e., starting to transmit blocks before they have been fully validated.

One of the reasons for such delays is churn. Imtiaz et al. [54] report that almost all (97%) Bitcoin nodes are connected intermittently only, which results in significant numbers of unsuccessful exchanges, roughly twice the figure for continuously connected nodes. In particular, they demonstrate experimentally that this churn leads to a 135% average increase in block propagation time (i.e., 336.57 ms vs 142.62 ms), and can lead to as high as an 800-fold increase in the worst case measured.

Permissioned blockchain networks based on Byzantine Fault Tolerance (BFT) consensus algorithms are also highly affected by the propagation time. Nguyen et al. [83] demonstrate how a network delay leads to a 30 times larger offset in the consensus layer in Hyperledger.

Often, miners collaborate in mining pools to share the risks and rewards of finding blocks (☛**Q2**,**Q3**). To this end, a dedicated server is connected to a node that acts as a gateway to a cryptocurrency network. This node gathers newly transmitted transactions and newly built blocks to construct a new block template. The template header is then sent via a mining pool server to the miners which attempt to complete it to a valid block. In the simplest approach for Bitcoin, the miners try different values for the nonce field in the header. If the resulting hash has enough leading zeros for the current difficulty level, i.e., when the block is completed, it is sent back to the server, which then uses the gateway node to publish the newly formed block to the network and distributes the reward among the miners of the pool corresponding to their contributions. In 2017, Bitcoin derived at least 95% of its mining power from 10 mining pools; in the Ethereum network, 6 pools are responsible for 80% of the mining power [66].

## 3.2 State of the Art

Networking-induced bounds on throughput and latency for the protocols in Bitcoin are studied in [22]. By considering the *effective throughput* of the network, i.e., the percentage of nodes that receive a block within one block interval period, they conclude that the block size should should not exceed the block interval divided by the effective throughput. From simulating the P2P network and measuring the time that the blocks reach the nodes, they infer that for 10min or shorter intervals the block size should be below 4 MB, as they found the block propagation time to grow roughly linearly with the block size until around 80 KiB blocksize, above which the throughput dominates (around 55 Kbps or 26 transactions per second to reach 90% of all nodes within one block interval) . To improve the system's latency, the block interval can be shortened, yet the block size must be adapted in alignment with the above bound. To be able to scale beyond these limits, the authors compile a list of technical design approaches, capturing consensus and storage components in addition to networking. Many of them have been active research topics, e.g., set reconciliation protocols, relay networks, incentivation and overlay topologies, they are discussed in more detail below.
**Compressed block encoding.** Many proposals to minimize the bandwidth consumption for block propagation exist. One such proposal for Bitcoin addresses the inefficiency of broadcasting blocks with all the transactions included. By the time a new block is created, it is very likely that most peers have these same transactions stored in their mempool. As such, relaying new blocks causes inbound bandwidth spikes for receiving nodes and potentially large outbound bandwidth spikes for nodes that receive blocks before their peers, since they will flood the network with the new, raw block data.

Xtreme Thinblocks (XThin) [106], deployed in Bitcoin Unlimited (BU) clients uses Bloom filter encoding the transaction IDs in nodes' mempool, thus only missing transactions must be exchanged. In an alternative, the Compact Blocks protocol [1], deployed in the Bitcoin Core, Bitcoin ABC, and Bitcoin Unlimited clients, the block's transaction IDs are announced shortened to 6-bytes. If the receiver has missing transactions, it requests them separately. For $n$ denoting the number of transactions, the network cost is hence $6n$ bytes while, XThin's cost is in the order of $m \log f \quad 6n$, with $m$ referring to the number of transactions at the receiver and a false positive rate of $f$. Thus, if the receiver is missing many transactions, Compact Blocks have an extra roundtrip time compared to Xthin, which may cost more if enough transactions are missing. Graphene [85] combines the use of a Bloom Filter with Invertible Bloom Lookup Tables (IBLTs) [47]. The main concept of Graphene's approach consists in shrinking the size of the sender's Bloom filter by increasing its false positive rate, and correcting any false positives at the receiver with an IBLT. The summed size of the two structures is smaller than using either alone. In practice, this technique performs significantly better than Compact Blocks for all but the smallest number of transactions, and it performs better asymptotically than any approach relying on Bloom-filters only. In comparison to XThin, Graphene uses significantly lower bandwidth both when the receiver is and is not missing transactions. However, Graphene may use an additional roundtrip time to repair missing transactions. The protocols described above rely on a single peer to send the complete data of a block, opposed to using multiple peers to transmit partial data. Velocity [21] is an approach that exploits the fact that typically several peers already have (parts of the) data in a block. To this end, it applies Fountain codes, which provide a mechanism by which information can be encoded such that the resulting segments can be probabilistically re-assembled into the original data when the number of the received segments exceeds a threshold.

Upon receiving a message advertising the knowledge of one or more objects, a receiver requests any unknown blocks using a request message to all its neighboring peers. Peers which have information on the requested block(s) respond with repeated responses, each encoding one symbol. The receiver collects these symbols and reconstructs the corresponding blocks when it has received a sufficient number of symbols. If the reconstruction succeeds, it notifies its peers to stop symbol transmission. Note that this approach can be used for node bootstrapping in addition to speeding up synchronization.

**Stratum Mining Protocol.** Stratum [55, 88] is the de-facto standard mining communication protocol used by blockchain-based cryptocurrency systems. It enables miners to reliably and efficiently fetch jobs from mining pool servers.

Stratum was initially a proposal for an open source client-server overlay protocol to support lightweight clients. The Stratum mining protocol extends this proposal to a networking protocol for pooled mining services on the Bitcoin network and many other blockchain protocols. The protocol establishes client-server connections using TCP sockets between mining clients and a pool operator or server to distribute new work defined through a blockchain's proof of work protocol.

Recabarren et al. in [92] exploit Stratum's lack of encryption to develop passive and active attacks on Bitcoin's mining protocol, with important implications on the privacy, security and even safety of mining equipment owners. Active attacks can hijack shares submitted by miners, and their associated payouts, by modifying TCP packet surreptitiously without causing disconnections and session resets. To mitigate such attacks, the authors propose Bedrock, a Stratum extension that protects the privacy and security of mining participants with mining cookies. Each miner shares a secret with the pool and includes in its puzzle computations, preventing attackers from hijacking the solutions.

Another attack on mining is introduced in [75], assuming miners are ration. An attacker presents only headers, i.e. a proof that the attacker mined blocks, without the block itself. From the miner's perspective, this reduces the probability to mine a successful block and thus might make it non-profitable, in expectation after considering mining costs (e.g. electricity and wear).

**Weak blocks.** In order to speed up block propagation even further, one approach is to let miners broadcast blocks they are working on before they have finished the corresponding proof of work. More precisely, so called *weak* or *near* blocks whose proof of work is insufficient for the target difficulty, can be disseminated early. As a consequence, when the block is fully mined the corresponding payload has been received and validated by most nodes already and only the headers needs to be broadcast and processed [3].

Traditionally, the weak blocks are discarded in Bitcoin, wasting their proof of work entirely. Ideally, the chain is solified with any and all sufficient proofs of work. Weak blocks by definition have shorter interarrival times and could thus be used by miners to both receive strong confirmation signals for weak transactions (transactions in weak blocks) as well as anticipate forks sooner (since mining variance is reduced in weak blocks, conflicting blocks appear sooner). Many updates to Nakamoto Consensus have been proposed that utilize similar ideas, yet no protocol change to utilize weak blocks has made its way into the Bitcoin Core source code. Some, such as BitcoinNG [36], exploit weak blocks to store and propagate transactions. The *key* blocks serve to elect a new leader, granting that miner the right to extend the chain with *weak* blocks. The protocol splits rewards between miners of previous leader elections to incentivize against malicious behavior such as selfish mining or hidden block extension attacks. Another similar proposal termed *Flux* [116] augments the existing Bitcoin protocol with weak blocks such that chains of weak blocks or sub-chains contribute to a chain's proof of work. Using the heaviest chain rule as its consensus rule, it can provide faster transaction confirmation times by ensuring that *key* blocks that link to sub-chains contain transactions included in the sub-chain's *weak* or *sub*-blocks. This can work in practice without the buy-in from all miners as well. One can imagine that if a certain number of miners opts in for broadcasting *weak* blocks, *key* blocks (which would also serve as legacy Bitcoin blocks) could ensure that the dominant chain contains some sub-chains of *weak* blocks.

**Relay Networks.** In parallel to the public P2P protocol, separate relay networks have been designed to increase network efficiency for miners. The first such system for Bitcoin, called Bitcoin relay network [78], achieves this by disseminating blocks without full block verification and retransmitting known transactions. It consists of a few nodes (supported by donations) scattered around the globe, all of which peer with each other. Another approach, Falcon [10], relies on cut-through routing for faster block propagation in addition to minimal validation and a hand-optimized topology. More recently, FIBRE [14] has been initiated to provide a similar service by combining cut-through routing with compact blocks and forward error correction over UDP (the normal Bitcoin protcol uses TCP) for registered users. Both Falcon and FIBRE can greatly reduce block propagation times and block orphan rates in the Bitcoin network, as shown in [84]. However, it is important to note that neither was designed, nor is suitable, to scale Bitcoin. Bitcoin cannot rely solely on these relay networks to achieve higher throughput, since the use of a relay network to scale, places the control over which transactions are included in the blockchain, and which miners may participate, in the hands of its operator. For example, the relay network operator may choose (or be coerced) to propagate blocks only from one group of miners, and reject all others, or to propagate blocks only to one group of miners, and not to others. Worse still, it might reject all blocks which contain transactions involving a specific address, effectively banning its owner from using it.

**State Synchronization.** For new nodes to be able to contribute to the P2P network quickly, Ethereum provides a state synchronization protocol. The first message sent by two Ethereum peers after the handshake describes their status containing information on the node's protocol version, network ID (multiple Ethereum networks exist), the hash of the genesis block, the best known block hash and the currently used difficulty. Only connections to nodes operating on the same network ID and genesis block are maintained. Based on their best block hashes the nodes will then synchronize their available information.

When a node joins the Ethereum network, it obtains a local copy of the full blockchain by first requesting block headers, which include block meta information such as parent block hash, miner address. After it has compiled a list of missing block hashes, the node then sends requests to retrieve full block contents and verify the validity of the blockchain.

In Ethereum two validation mechanisms can be distinguished: 1) block header validation and 2) blockchain state validation. Block header validation, ensures that a block's parent block hash, block number, timestamp, difficulty, gas limit, and valid proof of work hash are correct. In contrast, blockchain state validation consists of sequentially executing all transactions and thus requires significantly more computation and time. In order to reduce the time for new nodes to synchronize and validate the entire blockchain, the *fast sync mode* has been introduced. Instead of running blockchain state validation on all blocks since genesis (as necessary in Bitcoin), header validation is run until a pivot point block close to the most recent head of the blockchain is chosen (using messages to obtain meta information including gas consumption, transaction logs, and status code). At the pivot point, a fast sync node downloads a global state database at that block. From the pivot point onward, the node performs full blockchain validation.

**Incentives for Block Propagation.** For cryptocurrencies to function properly, blocks need to be propagated promptly upon their creation to all other users in the network. This is crucial both to the liveness and to the security of these protocols. To this end, there is a need to examine that miners are incentivized to follow block propagation rules, and not to vary from them (☛**Q4**). Selfish mining attacks [37, 97] aim to increase the relative fraction of blocks mined by an attacker through timing the release of blocks created by an attacker. The strategies differ based on how long the attacker waits before publishing blocks from a secret chain. These attacks show that there are cases in which miners can profit by not propagating blocks as soon as they are created.

Another case in which miners can profit by deviating from vanilla block propagation is **SPV mining.** In a similar vein, the "SPV (simplified payment verification) mining" concept can decrease the block propagation latency, by avoiding the full verification of blocks and instead relaying them partly unchecked. Originally, the approach has been developed to expedite mining: In order to build on top of the previous block and extend the chain, miners need the hash of the previous block. However, they do not need the full block with all the transaction data in order to start mining. It is in fact sufficient to only have the hash of the previous block header in order to mine a valid block. The incentive for SPV mining is a rush to mine blocks as fast as possible to increase profits. Waiting to download the full block and validate all of the transaction results in idle time which can lead to lost profits. Therefore miners may be tempted to find the next block before they have even had time to download and verify the previous block. This way, miners avoid putting any transactions in the block (apart from the coinbase transaction that rewards the miner), since they cannot know which transactions were in the previous block. Including transactions could result in double-spending (which would deem the block invalid). SPV mining is one of the reasons that empty blocks appear on the blockchain [110]. Moreover, SPV mining increases the probability of an invalid block being used to extend the chain and mine the next block linking to an invalid block (since the transactions are not validated by the following block, or even multiple blocks). This in turn results in the network being less reliable for payments as double-spends are more likely. In fact SPV mining has already caused a split in the network in the past: In 2015 there was a change implemented in the Bitcoin protocol (regarding enforcing BIP66 strict DER signatures) that was supposed to go into action after 95% of the network updated their software. The way in which this was implemented is the following: Once 950 of the last 1,000 blocks were version 3 (v3) blocks, all upgraded miners would reject version 2 (v2) blocks. On 4 July 2015, shortly after the threshold was reached, a small miner (part of the non-upgraded 5%) mined an invalid block. Unfortunately, it turned out that roughly half the network hash rate was mining without fully validating blocks, and built new blocks on top of that invalid block, causing a split.

SPV mining is also an issue in Ethereum, and in 2018 it was discovered that F2Pool (one of the largest mining pools at the time) was engaged in SPV mining which resulted in a dispropotionate number of empty blocks. [2]

Another similar strand of attack is Spy mining among competing mining pools. Spy mining occurs when attackers join the pools of others to obtain hints about new blocks appearing on the network. When a spy detects such information via the changed headers sent to it in the Stratum protocol, it can notify its other pool to avoid wasted work. Thanks to SPV, the miners can start mining a new block without seeing the old blocks content.

[2] https://medium.com/@ASvanevik/why-all-these-empty-Ethereum-blocks-666acbbf002

## 3.3 Open Questions

Latency and throughput of cryptocurrencies depend on the efficiency of block propagation. Therefore, mechanisms and incentives to spread newly minted blocks as quickly as possible while minimizing bandwidth waste are crucial. At the same time, the system design must not forego safety and the protocols in place must ensure that blocks contain valid transactions despite the hunt for speed.

---

Open Question 1. ***How to accelerate block propagation?*** *Enables scaling to higher transaction rates and lower latency. Several approaches to optimize the exchange of information in blocks between two nodes as well the dissemination in the network have been presented. Yet, both dimensions provide opportunities for empirical studies of the status quo and subsequently to devise new approaches to overcome this barrier to faster transaction rates.*

Open Question 2. ***How to design efficient networks for mining pools?*** *Pools that communicate information efficiently are better at mining, i.e., they have an advantage in mining over their competitors. Within a pool one could consider a more permissioned model with a more structured overlay topology for speed, balancing the possible velocity gains with the risk of new attacks.*

Open Question 3. ***Should mining pools be allowed?*** *Do the benefits of mining pools outweigh the security risk they introduce from a centralization aspect? If not, can they be prevented? Can pools be monitored for malicious behavior? How to design mechanism which incentivize honest pool behavior?*

Open Question 4. ***How to incentivize mining blocks with many valid transactions?*** *This will drive up transaction rates. SPV mining poses risks, thus designing a system preventing this behavior, e.g., with economic incentives is still open.*

---

## 4 TRANSACTION PROPAGATION

### 4.1 What it is about?

One of the main services provided to users in cryptocurrencies is the propagation of their transactions. Users' transactions must reach miners in order to be included in blocks, and similarly miners are interested in obtaining transactions of users in order to be able to collect their associated fees. Hence, delays in transaction propagation result in possible delays for transaction confirmation for users, and may cause losses of funds for miners.

Bitcoin's method for propagating transactions is flooding-based, using the same underlying P2P network used to relay blocks (☛**Q5**). Thus, if a user sends a transaction to a node, it is potentially sent to all other peers of that node, and then propagated onwards. Transaction relaying has been measured to be slower than block propagation [28]. While 50% of blocks were broadcast to 25% of the nodes in less than 22 seconds, 17 minutes are needed to relay 50% of the transactions to the 25% of the nodes in the sample. Similarly, Neudecker [79] reports that since 2017 transactions propagate to 50% of all nodes within around 5 seconds and to 90% of all nodes within around 15 seconds, both with a trend to increase over the horizon of the measurement campaign.

Kim et al.[62] observed that transactions dominate the network traffic, with clients using different strategies for transaction dissemination. Different protocols have been suggested, such as Geth which relays transactions to all peers, and Parity which forwards them only to a subset of size square root of number of peers in the network.

## 4.2 State of the Art

**Amplified DDoS.** One main concern with respect to transaction propagation is that attackers will try to send many transactions to nodes. If each such transaction is later propagated to the entire network unconditionally, then attackers would be able to amplify any large scale DoS attack—sending a single message causes many more messages to be generated throughout the network. Thus, an attempt is made to charge the attacker for sending messages. Unlike with blocks, that are rate-limited by the very fact that they require a proof of work in order to be valid, transactions bear no such limits. Their main cost for senders are reflected in the fees paid by the sender. These fees are not guaranteed. In fact, if a transaction is propagated to the entire network and later is not added to the blockchain, its fees are not collected. Thus, in Bitcoin, miners only convey transactions that enter their mempool – these in turn are transactions that pay a sufficient fee that is likely to lead to their inclusion in a block.

Similar concerns apply when users wish to increase the fees of their transactions (possibly after finding out the fee is insufficient to enter a block quickly enough), or to users who wish to double-spend previous unconfirmed transactions. If users are allowed to replace transactions with very modest fee increases, and the new transactions are propagated everywhere, again an attack is possible: attackers will simply send one transaction and then replace it with newer transactions with insignificant fee increases, thereby flooding the network once again. Thus, miners typically set a minimal fee increase to replace a transaction in the mempool and have it re-sent.

Similarly, if a user double-spends a transaction and redirects its funds elsewhere, miners will not typically relay the double-spend. This serves both as a countermeasure for transaction flooding and as a tool to support 0-confirmation recipient policies, which we discuss below.

**Information Eclipsing and 0-Confirmation policies.** Some users may not wish to wait until transactions are included in blocks. They then can adopt a somewhat risky policy of accepting 0-confirmation transactions, i.e., they can consider funds received if a transaction is propagated through the network (with fees that they estimate are sufficient to enter a block). Since these transactions are not yet included in a block, they are vulnerable to double-spends. Most notably, double-spends by miners that need only a single block that contains a conflicting transaction before the double spent transaction is included in a block itself (known as Finney attacks).

Due to the policy of miners mentioned above, double-spend transactions are not entered into the mempool of nodes and are not further propagated to others. This implies that some nodes are essentially "eclipsed" by their neighbors and may never find out about the existence of a double-spend, simply because these neighbors first received one transaction (and forwarded that one to the node), but did not relay a conflicting transaction that was later received, leaving the node unaware of its existence. Information eclipsing hinders nodes that would like to accept 0-confirmation policies (☛**Q6**).

Still, such attacks require mining power which can present some barrier to the attack. The question of how easy it is to attack 0-confirmation transactions without holding mining power naturally arises. Karame et al. [59] analyzed such attacks. They discuss countermeasures such as waiting a while before accepting the payment (and checking to see if conflicting transactions are propagated) and judge that these are not always sufficient to prevent attacks.

They propose modifying protocol rules so that nodes forward double-spending transactions instead of dropping them to avoid information eclipsing. They however do not analyze the effect of denial of service attacks that may be aggravated as a result.

**Privacy.** An active attacker may wish to identify the node from which transactions originated. By actively connecting to several nodes, it is possible that a curious observer will observe the transaction origin, or will be able to deduce it.

Biryukov et al. [12] present a deanonymization method for a significant fraction of Bitcoin users that correlates their pseudonyms with public IP addresses. The method explicitly targets peers behind NAT or firewalls, and can differentiate between nodes with the same public IP. They show ways to counteract the fact that nodes may use TOR to hide their IP and essentially utilize an anti DoS countermeasure in Bitcoin to cut off access to TOR. The main deanonimization technique is to identify nodes via the set of nodes it connects to (its entry nodes), transactions are mapped to a set of entry nodes, which uses a fingerprint to associate together transactions with similar sets, which suggests they belong to the same clients. The authors primarily propose to change the set of entry nodes often to avoid such correlations.

Neudecker and Hartenstein evaluate a form of deanonymization in [80]. They compare clustering approaches using transaction data (like signatures for different public keys that appear together in transactions) with clustering based on network data. Finding a correlation between the two likely means that both approaches in fact approximate the desired outcome. They show that for the majority of users no correlation between network information and the clustering performed on blockchain data could be found. A small number of participants do exhibit correlations that might make them susceptible to network based deanonymization attacks.

The Dandelion protocol [15] is a suggestion to provide better network-layer anonymity for transacting users. The main construction in Dandelion is based on having a forwarding phase for transactions before they are widely distributed, trying to mask the origin of transactions. Nodes agree on some full ordering of the network (i.e. some Hamiltonian cycle) which is changed every few minutes in order to avoid the adversary learning it. Transactions are first broadcast along this path for a random (small) number of steps. After this phase, broadcast is done through diffusion as it is done today in Bitcoin. Dandelion++ [39] extends Dandelion to defend against adversaries that are allowed to disobey the protocol. The core improvement is moving from forwarding through a long line graph to a referral 4-regular graph in the initial phase (before diffusion).

**Incentives.** The incentive of a node to participate in the dissemination of transactions is unclear. As an example, consider the case of a large transaction with high fee. In Bitcoin, the miner's incentive is to not propagate the transaction to other miners, in order to reduce competition (so it can include the transaction in one of its own blocks and claim the fee).

Although this field was studied thoroughly in the context of P2P networks (examples in [31], [30]), one of the first paper examining this for cryptocurrencies was [7]. In this paper, the authors offer to augment the protocol with a scheme that rewards information propagation, while balancing it with the incentive to decrease competition (☚**Q7**). They show that their scheme is sybil-proof (robust against creating clones) and has low overhead (a total reward that is not too high).

An improvements is proposed in [34], where additionally to awarding the propagating nodes, the authors propose "smart routing" (☚**Q8**). In this mechanism, nodes directly route the transactions to a round leader, which is known in advance (first-leader-then-block (FLTB) consensus protocols). This mechanism increases the bandwidth efficiency by reducing the propagation of redundant transactions.

### 4.3 Open Questions

If the miners do not have access to a large number of transactions to put into blocks, the throughput and latency of the cryptocurrency will be suboptimal. In addition to the technical limits of spreading transactions widely, incentives play a very important role here and must be designed carefully.

---

Open Question 5. ***How to avoid transaction floods?*** *This poses a security risk. An attacker may try to flood the network with meaningless transactions and thus cause the nodes to waste resources on them. How can one quickly identify bad transactions and discard them? How to trade off the verification cost, punishment mechanisms and velocity?*

Open Question 6. ***How to balance DOS prevention and 0-confirmation requirements?*** *On the one hand, nodes strive to avoid DoS attacks by not allowing the propagation of double-spend transactions. On the other hand, users want low-latency cryptocurrency systems and thus favor 0-confirmation policies. Mechanisms to meet these two conflicting goals would allow for a better user experience.*

Open Question 7. ***How to model and analyze cryptocurrency protocol messages?*** *The costs and benefits of propagating the different protocol messages deserves a more thorough analysis under realistic utility assumptions. This will uncover further weaknesses of current mechanisms and inform the development of superior approaches.*

Open Question 8. ***How to implement a suitable reward system for propagation of transactions?*** *This is vital for both liveness and decentralization. Currently prevalent mechanisms do not offer any incentive for nodes to propagate transactions. Moreover, miners have an incentive to keep transactions with high fees to themselves, instead of propagating them widely. This is a problem for liveness - if users can't get their transactions to all miners, transactions will be approved either slowly or not at all. On the other hand, some miners are also harmed if they can't include high paying transactions in their blocks.*

---

## 5 TOPOLOGY OF THE P2P NETWORK

### 5.1 What is it about?

The Bitcoin P2P network topology is formed by each peer connecting to 8 nodes (outbound connections) and accepting up to 125 in-coming connections. Outbound destinations are randomly selected among known identities. In other cryptocurrencies, nodes are assigned roles that influence the topology. For example, Cardano distinguishes between mutually exclusive core, relay, and edge node roles [20]. The core nodes create blocks and run the consensus protocol, in other words, the core nodes maintain the blockchain. Relay nodes protect the core nodes, serving as intermediaries between the public internet where the edge nodes reside and the core node. If relay nodes are attacked, this may lead to a service interruption, but the integrity of the core nodes (and thus the Cardano blockchain) is not compromised. Relay nodes are fully under the control of the federated committee of initial Cardano stakeholders. Edge nodes create the payload of the blocks. They can be run by anyone on their computer to create currency transactions. They cannot directly communicate with core nodes, only with relay nodes and with other edge nodes. Also Ripple distinguishes between different roles for nodes, partitioning them into superpeers and leaves [93]. A node in the leaf role does not route messages and only connects to superpeers. In the superpeer role, a peer accepts incoming connections from other leaves and superpeers up to the configured slot limit. It also routes messages.

Before being able to send and receive protocol messages a node has to find other nodes to connect to join the network. This *discovery* process typically relies on static information sources and/or Distributed Hashtable (DHT) approaches, discussed in the first part of this section. Knowledge of the network topology can give parties an advantage in the dispersal of information (blocks, transactions) which can lead to security risks. Because of this, there has been extensive research and development of tools and techniques aimed at exploring and mapping the Bitcoin P2P topology. The same holds true for many other cryptocurrencies. We present here some of these works and their main contributions.

## 5.2 State of the Art

**Discovery.** To join a cryptocurrency P2P network, a new node must find other nodes to connect with. In Bitcoin, a node first tries to connect to nodes it knows from participating previously. If no connections can be established this way, or if the node connects for the very first time, it queries a list of well known DNS seeds. As a last resource, it will try to connect to hardcoded seed nodes. The DNS seeds are maintained by Bitcoin community members: some of them provide dynamic DNS seed servers which automatically get IP addresses of active nodes by scanning the network; others provide static DNS seeds that are updated manually and are more likely to provide IP addresses for inactive nodes [13]. With the first messages exchanged between new peers, they inform each other of a random subset of locally known addresses with a timestamp of at most 3 hours ago. With this mechanism a list of addresses is maintained at each node. Each node will also accept incoming connections (up to 125 by default). The address of a new node is propagated through the network, so all peers can learn about it eventually.

Ethereum and Cardano use mechanisms inspired by Kademlia [71], a Distributed Hashtable (DHT) approach that has already been widely used for file sharing. Kademlia assigns key-value pairs to sets of peers based on the distance between the key ID and the node IDs and a routing table structure is maintained that allows to find responsible nodes recursively in a logarithmic number of steps.

While Kademlia and its derivatives have been used for many years, there is no formal proof for its performance and robustness. More theory-oriented approaches [17, 50, 114] on the other hand, have not established themselves as alternatives. [50] shows how to maintain clusters of size $O(\log n)$, each containing more than two thirds of honest nodes with high probability. Even when the system size can vary polynomially with respect to its initial size, the communication cost induced by each node arrival or departure is $O(\text{polylog } n)$. The approach guarantees robustness to a Byzantine adversary controlling a fraction 1/3 of the nodes (could be 1/2 with the application of cryptography). The proofs guarantee polylogarithmic maintenance and sampling overhead and rely on assumptions of a synchronous network. Alternatives, e.g., [114], based on the BLS threshold signature scheme, demonstrate how a DHT with quorums of logarithmic size, the time and message complexity is polylogarithmic if up to a third of nodes per quorum is malicious. To this end, any request needs quorum approval before getting answered or continued to avoid SPAM and DOS attacks and prevent wrong responses. To remain robust despite churn, routing tables are maintained according to the Cuckoo Rule. Compared to other approaches, this method requires the creation and verification of many signatures, which is demanding.

If one can accept probabilistic failures, even more efficient options are possible (☛**Q9**). Jaiyeola et al. describe in [57] how to use quorums sizes of $O(\log\log n)$, despite an adversary that controls a constant fraction of the computational resources in the network. Using their DHT approach, all but an o(1)-fraction of the machines can communicate with all but an o(1)-fraction of the machines in the network in $O(\log n \text{ poly}(\log\log n))$ steps. Instead of a DHT-based approach, Brahms [17] proposes churn and Byzantine resistant sampling. The paper presents an attack-resilient gossip-based membership protocol and shows how to extract independent uniformly random node samples from the stream of node ids gossiped by the first. It draws its power from an assumption of limited bandwidth avaialability: Byzantine nodes cannot send messages unlimitedly, if one node sends more often than expected, it is ignored. In Brahms unsynchronized gossip rounds, nodes send addresses they know to some other nodes (to reinforce the knowledge for underrepresented nodes), and request known addresses from other nodes (to spread existing knowledge). If more than the expected number of address are received in a round, Brahms does not update its view in that round to prevent malicious influence. Furthermore, locally known history also influences next views to avoid poisoning. Together with a sampling algorithm this ensures that nodes have an approximately uniform sample of the nodes in the system, even as long as every joining correct node knows some correct other node. Brahms offers a tradeoff between communication and storage and can thus be adapted to different needs. Such a sampling approach can also be run on top of other solutions.

**Eclipsing / Splitting the network.** Splitting the Bitcoin network can have severe consequences, as the shorter chain produced will not survive and as a result, many transactions are rolled back (and potentially double-spent), and the revenue of miners from these blocks is lost. Partitions affect the ability of participants to operate on transactions. This may cause exchanges to stop receiving and sending the cryptocurrency, and merchants to be unable to get paid. There are several ways to isolate nodes in the network: for example, by disrupting the routing of traffic between them, or by causing nodes to connect only to attackers (Eclipsing). We list here a few well-known eclipse attacks, and some techniques suggested in the literature to detect and avoid them.

In Bitcoin, nodes choose their peers from a list of stored IP addresses. This list is limited in size and IPs must be evicted if fresh ones are placed inside. IPs are placed in the list pseudo-randomly in a way that is based on the IP itself and the IP of the advertising node. In [52], the authors explore ways of isolating nodes in Bitcoin by affecting the way that nodes choose their peers. The main idea of the attack is to announce many IPs to the node that are either controlled by the attacker or that have no node behind them. The node eventually evicts all IPs of honest nodes from the list, and will only connect to the attacker (☛**Q10**). The idea is to cause collisions in the placement of similar IPs and of IPs advertised by the same node, thus the attack above needs some minimal number of IP addresses controlled by the attacker to succeed. This number is not high in practice. Also Ethereum is the target of eclipse attack constructions. In [112], Ethereum's P2P network is partitioned without monopolizing the connections of the victim, which is possible due to the block propagation design of Ethereum. In this attack, the attacker can potentially keep the victim from receiving a block almost indefinitely. This attack could be used as an infrastructure for a double-spend attack. The authors also present an exploit that can force a node to accept a longer chain with lower total difficulty than the main chain (also using the block sync mechanism). In this attack a node that newly connects to the network and receives a chain that is longer than the valid chain but has a lower total difficulty because the adversary advertised a higher total difficulty than honest nodes. The attacker is therefore disconnected from the network. The authors highlight a bug in Ethereum's difficulty calculation as well. This can be used in an attack that prevents the victim from synchronizing with the valid chain. The paper also outlines countermeasures.

Two attacks on Bitcoin exploiting the networking stack are presented in [5]. First, BGP hijacking that is used to disconnect parts of the network. The network is shown to be poorly distributed, so that relatively few prefixes need to be hijacked in order to partition miners from each other. Once a partition is fixed, natural churn allows nodes to connect across the partition and blocks are once again propagated. A second attack proposed in the paper utilizes the fact that Bitcoin traffic at the time was un-encrypted. Intervention in the content of announcements of new blocks and transactions as well as requests for the data of recently announced blocks was shown to severely delay block propagation. As a result nodes are left uninformed of the latest blocks in the chain for longer periods, which causes miners to waste time mining blocks that will be discarded and will yield no reward, and users to be unaware of funds they may already have received.

A game theoretic approach is used in [103] to manage the list of known peers. Consequently, attackers need to corrupt a large number of nodes to eclipse a node successfully. Similarly to the Bitcoin protocol, the paper assumes that acquiring IPs from the same prefix is cheaper than acquiring the same number of IPs from multiple prefixes, and utilizes this fact in the peer selection mechanism to increase the attack cost.

In [77] the authors investigate new "Stubborn Mining" attacks which combine eclipse attack with selfish mining [38] attacks. In this work, the authors consider the same model against users who are also eclipsed in the network and show the effect to which eclipsed users help a stubborn mining attacker. Overall, eclipse attacks empower adversarial agents with a larger strategy space to continue running attacks, and when paired with stubborn mining strategies, enable an attacker to improve their relation fraction of block rewards beyond traditional selfish mining strategies.

In [104], Tran et al. present the EREBUS attack, that partitions the Bitcoin network without any routing manipulations, which makes the attack undetectable (even against bug fixes specifically adressing partitioning attacks). Adversaries who may control large transit ISPs, are able to mount the attack. The adversary utilizes a large number of network addresses reliably over an extended period of time. A fix attempt to this attack is suggested here. This work enables Bitcoin core to prefer to connect to peers which are on different source ASNs to try to reduce the probability of any single host/path/hijack is relied on by a peer. The authors focus on the process of building the AS map, including simple filtering suggestions such as treating prefixes only reachable via a common upstream as if they were hosted directly on that upstream (by pulling routing information on diverse sources).

SABRE [4] presents a secure and scalable Bitcoin relay network resilient to routing attacks, designed to run alongside the existing P2P network and is easily deployable. The network is designed to efficiently handle high bandwidth loads, including Denial of Service attacks. The relay network provides security to Bitcoin clients by enabling them to learn the latest mined blocks and to propagate them network-wide. The authors use properties of BGP to predict where would be a good place to host relay nodes – locations that are inherently protected against routing attacks and on paths that are economically-preferred by the majority of Bitcoin clients. In addition, they provide resiliency through soft/hardware co-design through the use of caching, and offloading most operations to hardware (programmable network devices). This enables SABRE relay nodes to sustain load even when originating by DDoS attackers.

The effectiveness of mining pools can also be hampered by Distributed Denial of Service Attacks (DDoS) in order to disrupt their operations. As a consequence a competing mining pool is slowed down giving an advantage to other pools. This in turn may encourage individual miners to leave unreliable mining pools and join the attacker's pool as a result. After currency exchanges, Bitcoin mining pools are the most frequent victim of DDOS attacks [108]. Of 49 mining pools, 12 experienced (repeated) DDoS attacks. Based on a game theoretic model where pools can select between investing funds into additional mining equipment or DDOS attacks [58], larger mining pools have a slightly greater incentive to attack than smaller mining pools.

**Man-in-the-middle attacks.** In [33], the authors study the impact of Man-In-The-Middle Attacks on Ethereum. The paper looks closely at the feasibility of MITM and double-spending attacks on simulated network corresponding to the real Ethereum topology with real network components. They show the impact of such attacks, also gathering public information about the network, and mimicing the structure of its biggest 10 mining pools connected through 5 BGP routers, and performing BGP hijacking and ARP spoofing. The authors find that the attack is almost infeasible in the public context (because of its structure), but in the case of route hijacking (e.g. if Ethereum is deployed over a WAN in a consortium environment, and an adversary that has control on the border gate) could double-spend through either BGP hijacking or ARP spoofing with a success rate up-to 80%.

**Privacy in topology.** The last topic which we mention briefly here is privacy with respect to hiding the topology. Knowing the topology can be exploited as we have seen above. Thus discovery and overlay maintenance mechanisms must balance efficiency and privacy (☛**Q11**). We discuss methods to map the topology in Section 8.

## 5.3 Open Questions

Open Question 9. ***Can we develop better link failure models, more suitable for cryptocurrencies and their analysis?*** *Traditional failure models consider the number of faulty nodes as the main parameter when analyzing P2P networks. For a more nuanced analysis, link failures must be taken into account as well. Link failures can be modeled as random processes or in a worst-case fashion that is bounded in some way (e.g., a strongly connected union of available links when considering a time interval [51]). More granular models and analyses tailored to the cryptocurrency conditions and constraints are necessary to better understand current cryptocurrency networks and to build the basis for future designs.*

Open Question 10. ***How to ensure an honest connected component?*** *This is necessary to avoid partition attacks. Most cryptocurrencies employ a flooding-based strategy to broadcast on a topology constructed with a (pseudo) random process. Non-honest nodes may choose to drop messages or forward outdated and wrong information. For a broadcast to succeed, thus enough honest nodes must be connected to each other via at least one path, not containing bad nodes. How can we ensure that these networks\graphs contain large connected components that consist of only honest nodes?*

Open Question 11. ***How to mask the P2P network topology to prevent attacks both on privacy of users and connectivity of nodes?*** *How can the overlay topology be efficient yet make it hard for attackers to learn it and mount eclipse and hijacking attacks? Answers to this question provide discovery mechanisms that strike a balance between containing truthful information and DoS resistance, e.g., using overlay rotation and sharding mechanisms that minimize the information necessary to participate in a cryptocurrency network.*

## 6 SHARDING

### 6.1 What is it about?

An emerging approach to overcoming performance and scalability limitations of traditional blockchain protocols is sharding. As in the database field, sharding describes the mechanisms to operate on a state partition instead of the whole state whenever possible (☛**Q12**). For cryptocurrencies that implies splitting the blockchain in smaller components and have a small committee work on each of them instead of having all nodes work on the entirety of the blockchain together. This in turn reduces the processing time of transaction, as the task is split between multiple small sets of nodes working in parallel. This reduces computation and storage per node.

Particularly interesting in the context of this survey is that sharding can also reduce the communication dramatically, allowing the system to scale to large networks. Sharding overcomes the need to disseminate all transactions and blocks in the entire network. Dissemination increases the overall latency of the consensus protocol significantly, and also consumes bandwidth resources. Sharding further introduces security risks, and in particular, global consistency among different shards needs to be ensured, careful handling of cross-shard transactions. Sharded blockchains may be more prone to failures, from the consensus point of view. These problems are aggravated as new network attacks targeting sharded blockchains emerge.

### 6.2 State of the Art

Elastico [65] is a sharding based consensus protocol for public blockchains. In this protocol, the mining nodes are partitioned into randomly selected committees that process disjoint sets of transactions with the goal of maintaining the following four properties: agreement, validity, scalability, and efficiency. The protocol executes in rounds: first, each committee uses a classical byzantine consensus protocol to agree on their set of transactions. The final decision is then sent to a "final committee". This committee combines all its input into the final consensus set which is propagated to the network. The authors implemented Elastico on AWS and showed the horizontal scaling of the network. While Elastico can improve the throughput and latency of Bitcoin, it requires all parties to re-establish their identities and re-build all committees in every epoch. Which creates communication overhead and incurs latency that scales linearly with the network size.

OmniLedger is another sharded blockchain protocol that generates identities and assigns participants to committees. The protocol assumes partially-synchronous channels to achieve fast consensus. In OmniLedger the transactions are gossiped to the entire network. This in some scenarios incurs an overhead in communication.

To tackle the above-mentioned shortcomings [115] introduced RapidChain, a sharding-based public blockchain protocol that shards the communication, computation, and storage overhead of processing transactions without assuming any trusted setup. RapidChain employs an optimal intra-committee consensus algorithm and uses a novel gossiping protocol for large blocks. The protocol also avoids gossiping transactions to the entire network. RapidChain uses sublinear communication, reduces the communication overhead and latency of P2P consensus on large blocks gossiped in each committee. RapidChain's committees discover each other via an efficient routing mechanism that incurs logarithmic latency (in the number of committees).

In [76] the authors study the issue of cross-shards traffic in blockchain systems, modelling the transactions as a graph: the accounts are the nodes, and the weighted edges are the number of transactions between them. Thus, sharding the blockchain corresponds to splitting this graph into subgraphs (mapping accounts to shards); the cross-shards traffic is represented by the edges between these subgraphs. The authors propose a traffic-aware approach to reduce cross-shards communication (☛**Q13**), as well as a radix approach to reduce the memory consumption of the mapping (☛**Q14**). As a case study, the Ethereum blockchain is considered.

To achieve a high degree of reliability, it was proposed to employ trusted hardware (Intel SGX) to create a fast sharding protocol that tolerates byzantine players [23]. The trusted hardware enables fast inner-shard consensus, as well as fast shard formation and transaction propagation, thus allowing this approach to scale up to transactions-per-seconds rates comparable to the one achieves by Visa. The authors implemented their approach on the Google Cloud Platform and compared their results to other state-of-the-art systems, getting up to 3K transactions per second.

Adversarial scenarios are also studied in [69]: the authors consider DoS attacks on separate shards, where a single "busy" shard could cause the entire network to stall. The authors suggest a vertical scaling approach, in which the sharding happens in the nodes themselves. This implies a higher resource demand for each node: each node is split between "Node-Shards" (NSs), while still maintaining efficient (O(1)) data access and security. The transactions are split into NSs in an unpredictable manner, preventing a targeted DoS and allowing concurrent data processing over separate data structures. Architectonic steps are made to reduce additional coordination between the NSs (such as un-ordered transactions inside blocks). The authors implemented and deployed their approach on AWS and showed a linear growth in performance with the addition of resources.

In [18], the sharding challenge in Ethereum's Proof-of-Stake scenario is tackled. In this model, blocks are finalized only by a single, special shard, called "shard 0". In every epoch, the validators in the shards create non-finalized blocks ( "bets"). Bets that were created in the n'th epoch are propagated to "shard 0" only during the $n$ 1'th epoch (i.e. after the epoch's validators finish voting). Finally, shard-0 will accept and vote for finalizing all the bets in the $n$ 2'th epoch. This method offers fast inner-shard transaction finalizing and lets "share-0" finalize the cross-shards transactions. This method also prevents shard-focused attacks (an attack that concentrates stake on a single shard) by choosing the validators randomly.

Finally, we would like to point the reader to the recent survey [111] which compares existing sharding technologies. The paper also includes an analysis of the communication complexity of current state-of-the-art sharding solutions.

## 6.3 Open Questions

While sharding is a well-known and frequently used technique in distributed systems, its employment in the context of cryptocurrency networks is still subject to active research. There are several fundamental questions:

---

Open Question 12. ***How to efficiently ensure ACID capabilities for cross-shard transactions?*** *ACID (Atomicity, Consistency, Isolation, Durability) capabilities facilitate reasoning on cross-shard transactions and provable security guarantees.*

Open Question 13. ***Can we implement demand-aware sharding optimized toward the specific workload distribution?*** *How to achieve this and how to efficiently collect data about the demand?*

Open Question 14. ***How to design sharding protocols that balance space & communication efficiency?*** *Ideally, sharding relies on a clustering which minimizes inter-cluster traffic. However, for an efficient implementation, additional properties need to be accounted for, e.g., regarding the mapping to shards: different mappings (e.g., DHT) entail different space and communication tradeoffs. What are the a achievable tradeoffs betweeen clustering quality and space as well as communication efficiency?*

## 7 OFF-CHAIN PAYMENT CHANNELS

### 7.1 What is it about?

Scaling limitations and transaction latencies have led to a rich corpus of work exploring different blockchain scaling solutions: including alternative blockchain consensus architectures [36, 61, 64, 73, 86], sharding [43, 65] or side-chains (mechanisms that allow digital assets from one blockchain to be used in a diffferent ones) [8], to just name a few [9].

Off-chain P2P networks are emerging as a parallel effort to rendering on-chain networks more scalable. Off-chain or so-called "layer-two" protocols (built on top of the layer-one blockchains) are typically defined as protocols that do not publish every transaction on the blockchain immediately (contrary to on-chain transactions) and entirely rely on the consensus algorithm of a parent-chain. Off-chain protocols rely on channels which establish a private P2P medium, governed by pre-set rules, e.g., a smart contract, allowing the involved parties to consent to state updates unanimously by exchanging authenticated state transitions off-chain.

Channels can come in different flavors, e.g., channels which are formed between $n$ parties, or commit-chains, which serve a similar purpose as payment channels and rely on a central but untrusted intermediary. One can also distinguish between payment channels which are off-chain payment interactions, and more general state channels, which are arbitrary off-chain interactions. Side-chains [8, 9] do not classify as layer-two solutions due to having their own consensus algorithm.

Payment channels emerged to support rapid one-way payments, then transitioned towards bi-directional channel designs where both parties can issue and receive payments. State channels generalize the concept to support the execution of arbitrary state transitions.

To give an example, consider two companies which transact frequently with each other. Rather than settling all their transactions on the blockchain the companies can each transfer a balance to a unique joint account, which is recorded in the blockchain, such that each party is entitled to receive its original balance once their joint account is closed. Afterwards, the parties can privately send money to each other using their cryptographic keys and update the state of their joint account, without revealing the updated state to the rest of the P2P network.

Since updates are not always recorded in the blockchain, the balance of the state channel changes over time, based on the transactions between the companies. When any of the parties wishes to close the state channel, it can submit the most updated state to the rest of the network to be recorded in the blockchain, thus closing the state channel and passing the funds back to the companies based on their most updated balance. State channels enable the same type of interactions as the Consensus Layer, without recording them in the blockchain. However they also introduce new complexities. For example, safety measures are required to prevent a dishonest party from closing of a state channel using an earlier version of the state channel, thus omitting recent payments.

Popular payment channel networks (PCNs) include Bitcoin Lightning [89], Ethereum Raiden [91], and XRP Ripple [41], to name a few. In all these networks, each node typically represents a user and each directed, weighted edge represents funds escrowed on a blockchain; these funds can be transacted only between the endpoints of the edge. Many PCNs such as Lightning and Raiden use source routing, in which the source of a payment specifies the complete route for the payment. If the global view of all nodes is accurate, source routing is highly effective because it finds all paths between pairs of nodes.

In several respects, routing in PCNs is fairly different from routing in traditional communication networks: in traditional communication networks, routing algorithms typically aim to find short and low-load paths in a network whose links are subject to fixed capacity constraints. In a PCN, link capacities represent payment balances, which can be highly dynamic: every transaction changes the payment balance initially set up for the channel.

Existing network routing algorithms for data transmission experience unique challenges when applied to PCNs. Node links and bandwidth capacities in data networks are not considered private information. In contrast, a PCN routing algorithm changes the state of the traversed channels to secure the asset delivery from the sender to the receiver. Depending on the transaction amount, certain channels may not be suitable to route a payment, and channel balances are thus an obstacle that routing algorithms have to account for [101]. An executed channel transaction permanently alters the state of all channels along the path.

## 7.2 State of the Art

We review state-of-the-art approaches structured around the different aspects which arise in off-chain networks: from channel establishment over route discovery to supporting scalability.
**Routing and Channel Establishment.** A distinguishing feature of PCNs is that they also support transactions between participants without direct channels, using multihop routing. In a nutshell, users can efficiently transmit funds from node A to B by relaying them over a path connecting A to B, as long as each edge in the path contains enough balance (escrowed funds) to support the transaction.

However, design tradeoffs and security implications of such multi-hop routing are not well-understood today [68]. Scalability is a concern here: currently, the Lightning network comprises more than 10k nodes and 35k channels, which are updated and changed frequently. We will dedicate a separate subsection to scalability, and focus on additional aspects in the following.

It has been shown that cost-efficient routing, aiming to minimize fees, can sometimes be exploited (☛**Q15**). Tochner et al. [102] identify and analyze a novel DoS attack arising in PCNs, based on an inherent tradeoff between how efficient (and hence predictable) versus how secure routes are: for cost-effectiveness, the routing algorithm should find paths with low transaction fees. The fees of a layer-two transaction should be lower than the fees for a layer-one transaction.

To provide privacy, routing paths should be found without disclosing transaction values (i.e. value privacy) and the involved parties (i.e. sender and receiver privacy). Tochner et al.'s attack is based on route hijacking and exploits the way transactions are routed and executed along the created channels of the network. The idea is that an attacker can first create channels that increase the probability that transactions will route through it. Using an amplification attack, the attacker can increase the delay of the new channels, delaying payments for the period of the hijack attack. The authors empirically show that with just 5 channels, an attacker can hijack the majority of transactions ($\approx 60\%$, while 30 channels hijack $\approx 90\%$). The authors also discuss countermeasures, also showing a simple example where a small change in the existing weight function of the routing algorithm decreases the hijack affect of such attacks.

Similar in spirit is the work by Tang et al. [100]. In PCNs, whenever a transaction succeeds, edge weights are updated. However, the new channel balances (i.e., edge weights) are usually not revealed to users directly for privacy reasons. This can lead to inefficiencies: when determining a route for transactions, users first have to guess a path that might be suitable, and then check if it really supports the transaction. This guess-and-check process dramatically reduces the success rate of transactions. At the other extreme, knowing full channel balances can give substantial improvements in success rate at the expense of privacy. To address this problem, Tang et al. [100] studied whether a network can reveal noisy channel balances to trade off privacy for utility. The authors show that in general, what can be achieved in this context is fairly limited. They then propose noise mechanisms and find that it is not possible to get large gains in utility by giving up a little privacy, or large gains in privacy by sacrificing a little utility. Hence, the authors argue that it is optimal to operate either in the low-privacy or low-utility regime.

The routing and hence the performance of the network typically heavily depends on the fees. Di Stasi et al. [27] suggested to change the way how nodes apply fees for forwarding payments, while trying to keep the network balanced and improve performance, also using a multipath routing payment scheme, to further reduce the fees paid by users and keep the network balanced. In particular, the authors argue that there are requirements that the fee function should satisfy, and are not currently fulfilled. Di Stasi et al. also discuss the problem of finding multiple paths between a source and a target, to improve transaction routing.

In order to ensure anonymity, onion routing is usually used, which however requires the random selection of nodes in a path. SilentWhispers [67] and SpeedyMurmurs [96] formalize and address concrete notions of privacy in this context. SpiderNetwork [99] improves the effectiveness of source routing in a dynamic PCN by favoring routes that minimize the balance difference as well as on-chain rebalancing, meaning that nodes deposit additional coins to improve the balance; their routing relies on a packet-switched network, that is, instead of routing a complete payment, the payment is split into constant-size units which are routed individually, mitigating channel capacity limitations. SpiderNetwork is therefore effective even when balances are constantly changing, at the cost of higher latencies if on-chain rebalancing is used.

**Routing in Large-Scale Topologies.** Today's routing algorithms require every node to know (and maintain) the entire topology, in order to be able to compute a route toward the transaction's target (☛**Q16**, **Q17**). This stands in contrast with the objective to support lightweight nodes (e.g., wallets) in the network, which should be able to perform transactions as fast as possible, requiring minimal disk space and control traffic (e.g., to acknowledge new nodes and channels' updates). In this sense, the underlying challenges are reminiscent of wireless and adhoc networks [87]: nodes are resource-constrained and can decide with whom to establish new connections (which comes at a cost).

One of the first and well-known scalable approaches is Flare routing [90] with beacon nodes. First, the node proactively maintain a list of channels and beacons in its close neighborhood. The route discovery process first compares the local knowledge with the target, and if a route was not found then it queries the beacons for a route between them. The source node will finally choose a route from the results using heuristic ranks. The incentive of the beacon is to increase the chances that a node will route through it, and it will earn the routing fee.

Another routing approach is introduced in [53]. They discuss adapting the Ad-hoc On-Demand Distance Vector Routing (AODV) protocol into offchain networks. This protocol operates as a hop-by-hop basis, and since it is quick adapting, it yields good routing methods with low computation and memory consumption. They examine transactions on a randomized graph, and compare their approach to the optimal routing, bounding the overhead in number of hops and reachability.

A major step ahead was then made by the Lightning developers, that relaxed this approach by assuming that nodes can store and maintain the whole network topology and state: Lightning introduces the notion of trampoline nodes [2] to which lightweight clients can outsource the route computation. In order to find trampoline nodes, lightweight clients can simply use a breadth-first search. The trampoline nodes know the entire topology (using the current gossiping methodology) and can hence provide routes. To provide incentives Lightning employs fee mechanisms.

A user that wishes to find a route can query multiple trampoline nodes, thus the trampoline's incentive is to suggest the best route comparing to his competitors, and maximize the chance that the node will route through the him, and therefore pay the trampoline's fees (☛**Q18**, **Q19**).

A work by Tochner and Schmid [101] analyzes the tradeoff triangle between Confidentiality (using a third party to discover a route), Efficiency (the user's cost to use a route) and Effectiveness (the availability of the route) of the route discovery process (☛**Q20**). The authors shows analytically that the user can not find a route discovery policy that maximize this tradeoffs, and show empirically that there are topologies in which the tradeoff is bounded.

Another approach was taken by the authors of [53], that adapted the “Ad-hoc On-Demand Distance Vector Routing" (AODV) protocol into off-chain networks. This protocol operates in a hop-by-hop basis, thus it requires low compute and memory consumption, and adapt fast to changes in the network. They examine transactions on a randomized graph, and compare their approach to the optimal routing, bounding the overhead in number of hops and reachability.

**Topology.** There are several interesting studies on the topology of transaction networks. In particular, Rohrer et al. [95] study the resilience of the Lightning network to topology-based attacks, and in particular, to isolation attacks (☛**Q21**). The authors argue that the Lightning network can be classified as a small-world and scale free network and show that in order to perform a resource-limited attack, the attacker should employ a highest ranked minimum cut strategy. However, high budgets (around 200BTC) are required to give the adversary the power to reliably disturb all payment attempts. Nisslmueller et al. [107] also showed that active and passive topology exploration can be exploited to attack confidentiality in off-chain networks.

**Offline Nodes.** Typically, intermediaries along a channel route are required to remain online and explicitly confirm all mediated transactions. Dziembowski et al. [32, 90] address these shortcomings with the introduction of virtual channels that support payment and state transitions. All intermediaries along the route can lock coins for a fixed period of time and both parties can treat the path as a new virtual channel connecting them directly. In this manner, A and B can transact without interacting with intermediaries along the path, thus reducing the transaction latency. Virtual channels are limited by the need to recursively set up a new virtual channel for every intermediary along the path. It is the intermediary's responsibility to ensure the channels close appropriately. In [72], McCorry et al. introduce the Pisa protocol which enables parties to delegate to a third party to provide security even to parties that may go off-line for an extended period of time. Zeta Avarikioti et al. followed a similar idea in [6], and suggested to use external parties that both parties of the channel agreed on, and "approve" any unilaterally action.

## 7.3 Open Questions

Off-chain networks do not only introduce promising new solutions but also several new challenges, e.g., related to how transactions are routed or information "gossiped", or related to how the topology should be designed. Indeed, routing and topology become more tightly coupled in off-chain networks, as nodes cannot only strategically choose routes but also channels: both the establishment as well as the use of payment channels is an inherently strategic decision, and subject to complex incentives and the extent to which a participant thinks she or he can benefit from different behaviors. A participant may not only try to strategically maximize her or his profit, but may also be malicious.

---

Open Question 15. ***How to incentivize nodes to contribute to efficient routing in PCNs?*** *Several state-of-the-art works have identified issues with the current fee-based routing scheme used, e.g., in Lightning. For example, by announcing low fees, nodes can launch a Denial-of-Service attack on transaction routing. Can we design a pricing scheme which on the one hand avoids these issues and on the other hand still incentivizes nodes to contribute resources?*

Open Question 16. ***How to allow PCN users to run light blockchain nodes instead of full nodes while remaining safe?*** *E.g., clients may aim to spoof channels, fees on the blockchain, etc.*

Open Question 17. ***How to design scalable payment routing strategies?*** *The main question is: Are there better alternatives to source routing? Several state-of the art approaches such as Flare and Trampoline nodes have been proposed lately, but there is still a long way to go in finding safe and scalable strategies. In particular, routing options today heavily depend on fees, which can cause congestion problems and even be used to devise attacks against the network.*

Open Question 18. ***How to avoid single points of failure in routing?*** *More robust routing algorithms and incentives are necessary. This question relates to the previous one. Since today routing depends heavily on fees (which are controlled by the nodes that own the channel), a single node can have a very large effect on the entire network. Is this a case for traffic engineering?*

Open Question 19. ***How to best route payments along multiple paths?*** *In particular, how can this be achieved in an efficient and secure manner?*

Open Question 20. ***How to balance privacy and efficiency in channels' liquidity observability?*** *Can routing be made aware of the current liquidity balance in channels? In particular, this raises the question about a possible efficiency-privacy tradeoff.*

Open Question 21. ***How to protect from isolation and hijacking attacks?*** *We suggest studying whether long liquidity lock attacks can be prevented. In particular, what strategies can the network deploy to prevent the success of such attacks? This question is relevant both in the context of channels, and the gossip network.*

## 8 MEASUREMENTS

### 8.1 What is it about?

Throughout the past years substantial efforts to measure and asses the properties of cryptocurrency networks have been made. These include both measurements of the P2P networks and measurements of payment channel networks. We will now touch on some of the methodologies used and the insights they offer. Some of these papers and their results have been mentioned in other contexts as well, in this section we study the their measurements contributions.

### 8.2 State of the Art

**Mapping the P2P network.** Donet et al. [28] presented BTCdoNET, a measurement tool for Bitcoin's P2P network based on crawling Bitcoin peers' list of known other peers. To this end, they use a Bitcoin sniffer tool The authors reports that in 2014, nodes placed in Unites States and China amounted to 37% of the discovered nodes. Germany, United Kingdom, and Russia represented a large share of the network, with 9%, 4%, and 7%, respectively. Japan, Brazil, Mexico, and China had low adoption rates, with the number of Bitcoin nodes being less than three per every 100k Internet users. In contrast, the Netherlands, Norway, Finland, and the Czech Republic have the highest adoption rates, more than 10 times higher than those of Brazil. Most of the detected nodes, remain connected for a short time only: merely 5,769 nodes (0.66% of the discovered ones) are still connected after 37 days.

In [79], Neudecker describes the methodology and results of his Bitcoin measurement campaign from 2015 to 2018, comparing his results to measurements from other sources (all showing similar general trends as well as short-term effects). He repurposes a Bitcoin client to serve as a monitoring tool which opens up as many connections to peers as possible (instead of limiting the number of active connections to 125), sends node discovery messages repeatedly. Like this, Neudecker studies churn, sybil connections (multiple established connections with the same IP address) and geographic location of peers. The total number of connections observed varied between less than 6,000 connections in late 2016 and around 14,000 connections in 2018. The number of detected Sybil peers is generally very low (less than 50 prior to July 2017, less than 200 after August 2017), with the exception of short periods in June 2017 and August 2017. According to his study, the average share of peers connected for at least one day varies between 55% and 75%, the share of peers connected for at least one week varies between 20% and 50%. This raises questions regarding the distribution of node types in the network (☛**Q22**, **Q25**).

Gencer et al. [42] investigate decentralization aspects of Bitcoin and Ethereum. With respect to mining power, the authors observe a slightly more centralized mining process in Ethereum: In Bitcoin, the weekly mining power of a single entity never exceeded 21% of the overall power. In contrast, the top Ethereum miner never had less than 17% of the mining power. Moreover, the top four Bitcoin miners have more than 53% of the average mining power. On average, 61% of the weekly power was shared by only three Ethereum miners.

Bitcoins' P2P network topology and main properties were measured and analyzed by Ben Mariem et al. in [11]. Their BTC crawler allows to track active nodes of the BTC P2P network, by exploiting the P2P mechanism to discover new nodes in the network. Specifically, the crawler retrieves the internal list of known peers of every new node that they discover. With this method, a snapshot of the topology can be collected in less than an hour, using high concurrency. According to these measurements, there are around 9.7K active nodes

(200K inactive), out of which 4% are connected via an anonymous connection. The authors gathered geo-locations of Bitcoin users through their IP addresses, and also determined the Autonomous Systems the users belong to as well as their client version. The authors also discuss a passive approach to reconstruct the topology of a blockchain P2P network, which can unveil miners. The paper further describes the structure of addresses (IPv6, IPv4, Tor hidden addresses), and presents empirical results on the geo-location of active nodes around the world. Another approach is presented in [25], uncovering the topology of the Bitcion network by processing orphaned and conflicting transactions More precisely, the authors show that by sending different double-spending transactions, one can determine if these nodes are connected. This is possible because of how a node's mempool maintains orphaned transactions and processes conflicting transactions. This technique allows to map the topology of the Bitcoin network in roughly 8 hours. The technique enables accurate topology reconstruction because of how nodes deal with both new and orphaned transactions. Nodes that receive a request and do not possess such a transaction will request it from the same peer. However, if the node already possesses the transaction it will not send a request. By sending different nodes orphaned and conflicting, double-spend transactions and timing these actions correctly, an agent can identify if two of its peers are connected themselves.

In [74], Miller et al. determine the Bitcoin topology by utilizing the *update* method of the timestamp field in the *addrMan* of nodes in the network, to learn about the topology. With this approach, the authors show that the deployed Bitcoin topology does not resemble a random graph. It is also possible to exploit transaction accumulation to map the Bitcoin network: In the two mechanisms presented in [48], (i) a node accumulates input transactions before propagating them, and then forwards them in the original order; (ii) nodes do not forward contradicting transactions, so that by propagating a contradiction to two nodes, they know which node is closer to the target (the one which got it first). By observing transaction announcement messages from the nodes, the authors argue that it is possible to map the neighbors of a node (by sending different messages to different clients). The paper also presents effectiveness and precision plots based on simulations.

Similarly, other networks have been analyzed, including Ethereum and Monero. For a measurement study of the Ethereum network peers [62], researchers developed a special node that listens to the propagation mechanism, connecting it to the main network. The study finds that 48% of all (3 million) discovered nodes do not contribute to the Ethereum Mainnet (i.e., they do not run the Ethereum sub-protocol or do not operate on the main blockchain). The authors also show that 76% use the golang implementation, 17% rust, 5% javascript (which might be cryptojacking web clients). The study also reports on the versions of the nodes (examining stable versions and up-to-date versions), the network size (activate peers), geographic distribution and node age. The authors observe that more than 40% (12%) of the nodes use US (Chinese) IP addresses. The most widely used ASes of the nodes can be assigned to cloud hosting providers, dominating residential or commercial operators. There are also tools to collect information about Monero's P2P network [19], including the network size, (geo-)distribution, and connectivity. The researchers set up 4 Monero nodes across the world and found that 87% of the nodes have a degree smaller than 8 (approx. 17% of the overall number of edges), being able to map all outgoing connections of 99% nodes in the network. The authors also succeeded to connect to 85% of the nodes that they discovered, which may enable network level attacks (eclipse, BGP hijacking, DoS).

Feld et al. [40] and Apostolokai et al. [5] pointed out a strong AS-level centralization that may impact Bitcoin network's connectivity — i.e. 10 ASes contain over 30% of peers, with 39 IP prefixes hosting half of the overall mining power (☛**Q23**).

**Mapping Payment Channel Networks** The structural properties of the Lightning Network have been studied in [98]. The authors showed that the diameter of the network is less than 3, and that low degree nodes are generally connected to high degree nodes. The authors in [70] examine network properties every month over 12 months and show the evolution of the network, from a routing perspective by studying node degree and centrality. They showed that the network becomes more centralized and, as a result, less robust to random failures.

**Attempts to mask the topology** There has also been some work that aims to fix some of the privacy concerns raised by the ability to measure the network: If the structure of the network is known, transaction messages can be more easily attributed to the originating nodes. One approach to harden the communication is to use SCION [109]. SCION's aim is to facilitate client node communication with a more direct, point-to-point infrastructure. This helps to prevent additional attacks such as network hijacks and BGP route poisoning/spoofing.

**Measuring forks** One of the earlier works measuring properties of the Bitcoin network is [24] where measurements of the number of forks, discarded blocks, and the network propagation delay are made using a "super-node" that connects to all of the discovered nodes in the BTC network.

In [82] the authors perform an empirical analysis of blockchain forks in Bitcoin. They analyze the announcement and propagation of blocks that led to forks of the Bitcoin blockchain. By analyzing the time differences in the publication of competing blocks, they compare the block propagation delay of miners to that of the average Bitcoin peer. The authors also show that the probability of a block to become part of the main chain increases roughly linearly with the headstart that block had over the competing block. In addition the authors analyze the block intervals immediately after blockchain forks in order to detect selfish-mining attacks.

**Propagation, throughput and latency.**

BTCdoNET measurements [28] also involve opening connections to a subset of the peers to listen to their announcements of transactions and blocks which they then store with timestamps in a database to infer their propagation time.

The authors found that after around 80 seconds, new blocks have reached 50% of the nodes and less than 1% of the blocks are known by 90% of the nodes in the same time period (☛**Q24**).

In addition to the nodes and their connections, Neudecker also investigated transaction and block propagation time [79], by storin their announcements of transactions and blocks. In addition, he uses Bitcoin protocol level pings as well as TCP SYN pings sent out to the known peers' Bitcoin port to measure round trip times with and without Bitcoin client processing times. He showed that 2016-2018 it took between 2 and 20 seconds until 90% of the nodes announced the reception of a new block, with a tendency for shorter propagation latency in the more recent past. This is a vast improvement considering that Donet et al. [28] reported that after 84 seconds new blocks have reached 50% of the nodes and less than 1% of the blocks are known by 90% of the nodes in the same time.

Bandwidth, latency, and decentralization metrics in the Bitcoin network and the Ethereum network were studied by Gencer et al. [42] in 2017. The study found that the observed bandwidth varies between different protocols, and the paper reports averages between 70 and 90 Mbps for Bitcoin and 55 Mbps for Ethereum nodes. The measured average latencies were between 135ms and 171ms for Bitcoin and Ethereum nodes respectively.

### 8.3 Open Questions

Open Question 22. ***Are there major differences in the connectivity of different nodes?*** *Are the differences increasing or diminishing over time?*

Open Question 23. ***Are cryptocurrency networks becoming less centralized?*** *Currently few Autonomous Systems control the majority of Bitcoin nodes. It would be interesting to measure how this changes over time.*

Open Question 24. ***How does the network behave when there is congestion of transactions, blocks and other messages?*** *When there are many transactions waiting to be propagated the propagation time (and behavior) can vary. A similar effect can happen if many block are created in a short time period. It would be interesting to measure propagation times of transactions and blocks during different congestion scenarios.*

Open Question 25. ***How many entities run full nodes? How many users rely just on light clients?*** *It would be interesting to develop a measurement methodology to explore these distributions.*

## 9 CONCLUSION

This paper presented an overview of the networking aspects of blockchains and cryptocurrencies in particular, with a focus on open research questions. Towards this goal, we also introduced the necessary context and reviewed the state-of-the-art. We believe that networking aspects of blockchains have not yet received the attention they deserve, and hope that our paper can contribute toward more research in this space.

In Table 1 we provide a summary of the questions discussed in the paper, structured around the methodologies needed to address them: from designing algorithms and network protocols, developing game theoretical models and solutions, to security analyses and network measurements. We linked each open question to the section that provides more details on its context and the existing research in its area. For example, although propagating knowledge around the network (Sections 3 and 4) is typically focused around network protocols, we argue that to address these open questions, also an understanding of cryptography and game theory is required. It is also interesting that game-theoretic aspects can be found for almost all questions, due to the lack of trust in cryptocurrencies. Thus, a game-theoretic approach is required to avoid contradicting incentives and misbehavior.

This table also highlights that there is much potential for foundational and applied future research in this area. For example, regarding scalability, it remains to explore further (and compare and combine) approaches based on sharding, off-chain network, and network design. Only with such more scalable networking solutions can blockchain networks meet their requirements under future mass adoption. Furthermore, for future adoption by larger organizations, it will be crucial to identify, understand and overcome the various novel network-level attacks on these novel networks.

**Acknowledgments.** Research in part supported by the Vienna Science and Technology Fund (WWTF) project WHATIF, grant ICT19-045, 2020-2024.

## REFERENCES

[1] [n.d.]. Bitcoin Improvement Proposals Compact Block Relay. https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki. Accessed: 2019-09-23.

| **Open Question** | **Section** | **Methodologies** |
|---|---|---|
| 1. How to accelerate block propagation? Enables scaling to higher transaction rates and lower latency. | 3 | PROT, ALG, GAME, SEC, CRYPTO |
| 2. How to design efficient networks for mining pools? Pools that communicate information efficiently are better at mining. | 3 | PROT, ALG |
| 3. Should mining pools be allowed? Do their benefits outweigh the risk they introduce from a centralization aspect? | 3 | GAME |
| 4. How to incentivize mining blocks with many valid transactions? This will drive up transaction rates. | 3 | GAME |
| 5. How to avoid transaction floods? This poses a security risk. | 4 | SEC, GAME |
| 6. How to balance DoS prevention and 0-confirmation requirements? | 4 | PROT, GAME, ALG |
| 7. How to model and analyze cryptocurrency protocol messages? | 4 | PROT, GAME |
| 8. How to implement a suitable reward system for propagation of transactions? This is vital for both liveness and decentralization. | 4 | GAME |
| 9. Can we develop better link failure models, more suitable for cryptocurrencies and their analysis? | 5 | PROT, ALG |
| 10. How to ensure an honest connected component? This is necessary to avoid partition attacks. | 5 | ALG |
| 11. How to mask the P2P network topology to prevent attacks both on privacy of users and connectivity of nodes? | 5 | SEC, CRYPT |
| 12. How to efficiently ensure ACID capabilities for cross-shard transactions? | 6 | PROT, ALG |
| 13. Can we implement demand-aware sharding optimized toward the specific workload distribution? | 6 | PROT, ALG |
| 14. How to design sharding protocols that balance space & communication efficiency? | 6 | PROT, ALG |
| 15. How to incentivize nodes to contribute to efficient routing in payment channel networks? | 7 | PROT, SEC, GAME |
| 16. How to allow PCN users to run light blockchain nodes instead of full nodes while remaining safe? | 7 | PROT, GAME, SEC |
| 17. How to design scalable payment routing strategies? | 7 | PROT, GAME |
| 18. How to avoid single points of failure in routing? More robust routing algorithms and incentives are necessary. | 7 | PROT, ALG, GAME |
| 19. How to best route payments along multiple paths? | 7 | PROT, ALG |
| 20. How to balance privacy and efficiency in channels' liquidity observability? | 7 | SEC, ALG, CRYPT |
| 21. How to protect from isolation and hijacking attacks? | 7 | PROT, ALG, GAME |
| 22. Are there major differences in the connectivity of different nodes? Are the differences increasing or diminishing over time? | 8 | MEAS, ALG |
| 23. Are cryptocurrency networks becoming less centralized? | 8 | MEAS |
| 24. How does the network behave when there is congestion of transactions, blocks, and other messages? | 8 | MEAS, PROT |
| 25. How many entities run full nodes? How many users rely just on light clients? | 8 | MEAS |

Table 1. Open networking questions and methodologies needed to address them: ALGorithms, network PROTocols, GAME theory, SECurity, CRYPTography, MEASurement.

[2] [n.d.]. Outsourcing Route Computation With Trampoline Payments. https://bitcointechweekly.com/front/outsourcing-route-computation-with-trampoline-payments/. Accessed: 2019-04-30.

[3] Gavin Andresen. 2015. *Weak Block Thoughts.* https://github.com/ethereum/devp2p/blob/master/caps/eth.md

[4] Maria Apostolaki, Gian Marti, Jan Müller, and Laurent Vanbever. 2018. SABRE: Protecting Bitcoin against Routing Attacks. *arXiv preprint arXiv:1808.06254* (2018).

[5] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2017. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 375–392.

[6] Georgia Avarikioti, Eleftherios Kokoris Kogias, and Roger Wattenhofer. 2019. Brick: Asynchronous state channels. *arXiv preprint arXiv:1905.11360* (2019).

[7] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. 2012. On bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*. 56–73.

[8] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. 2014. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains* 72 (2014).

[9] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. 2017. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936* (2017).

[10] Soumaya Basu, Ittay Eyal, and Emin Gun Sirer. 2016. *Falcon: Relay Network for Bitcoin Blocks.* https://www.falcon-net.org/

[11] Sami Ben Mariem, Pedro Casas, and Benoît Donnet. 2018. Vivisecting Blockchain P2P Networks: Unveiling the Bitcoin IP Network. In *ACM CoNEXT Student Workshop*.

[12] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. 2014. Deanonymisation of clients in Bitcoin P2P network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 15–29.

[13] Bitcoin. 2019. *P2P Guide.* https://bitcoin.org/en/p2p-network-guide

[14] BlueMatt. 2016. FIBRE. http://bitcoinfibre.org/public-network.html

[15] Shaileshh Bojja Venkatakrishnan, Giulia Fanti, and Pramod Viswanath. 2017. Dandelion: Redesigning the bitcoin network for anonymity. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 1 (2017), 22.

[16] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. 2015. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *Proc. IEEE Symposium on Security and Privacy (SP)*. IEEE, 104–121.

[17] Edward Bortnikov, Maxim Gurevich, Idit Keidar, Gabriel Kliot, and Alexander Shraer. 2009. Brahms: Byzantine resilient random membership sampling. *Computer Networks* 53, 13 (2009), 2340–2359.

[18] Vitalik Buterin. 2016. Ethereum 2.0 mauve paper. In *Ethereum developer conference*, Vol. 2.

[19] Tong Cao, Jiangshan Yu, Jérémie Decouchant, Xiapu Luo, and Paulo Veríssimo. 2019. Exploring the Monero Peer-to-Peer Network. *IACR Cryptology ePrint Archive* 2019 (2019), 411.

[20] Cardano. 2019. *P2P Topology.* https://cardanodocs.com/cardano/topology/

[21] Nakul Chawla, Hans Walter Behrens, Darren Tapp, Dragan Boscovic, and K Selçuk Candan. 2019. Velocity: Scalability improvements in block propagation through rateless erasure coding. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 447–454.

[22] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. 2016. On scaling decentralized blockchains. In *International conference on financial cryptography and data security*. Springer, 106–125.

[23] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. 2019. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 international conference on management of data*. 123–140.

[24] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the bitcoin network. In *Proc. IEEE P2P Conference*. IEEE, 1–10.

[25] Sergi Delgado-Segura, Surya Bakshi, Cristina Pérez-Solà, James Litton, Andrew Pachulski, Andrew Miller, and Bobby Bhattacharjee. 2018. TxProbe: Discovering Bitcoin's Network Topology Using Orphan Transactions. *arXiv preprint arXiv:1812.00942* (2018).

[26] Sergi Delgado-Segura, Cristina Pérez-Solà, Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, and Joan Borrell. 2018. Cryptocurrency networks: A new p2p paradigm. *Mobile Information Systems* 2018 (2018).

[27] G. Di Stasi, S. Avallone, R. Canonico, and G. Ventre. 2018. Routing Payments on the Lightning Network. In *Proc. IEEE Blockchain*. 1161–1170.

[28] Joan Antoni Donet, Cristina Pérez-Sola, and Jordi Herrera-Joancomartí. 2014. The bitcoin P2P network. In *International Conference on Financial Cryptography and Data Security*. Springer, 87–102.

[29] Maya Dotan, Yvonne-Anne Pignolet, Saar Tochner, Stefan Schmid, and Aviv Zohar. 2020. Cryptocurrency Networking: Context, State-of-the-Art, Challenges. In *Proc. 15th International Conference on Availability, Reliability and Security (ARES)*.

[30] Fabio A Drucker and Lisa K Fleischer. 2012. Simpler sybil-proof mechanisms for multi-level marketing. In *Proceedings of the 13th ACM conference on Electronic commerce*. 441–458.

[31] Kirill Dyagilev, Shie Mannor, and Elad Yom-Tov. 2010. Generative models for rapid information propagation. In *Proceedings of the First Workshop on Social Media Analytics*. 35–43.

[32] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. 2017. PERUN: Virtual Payment Channels over Cryptographic Currencies. *IACR Cryptology ePrint Archive* 2017 (2017), 635.

[33] Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. 2018. Impact of man-in-the-middle attacks on ethereum. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 11–20.

[34] Oğuzhan Ersoy, Zhijie Ren, Zekeriya Erkin, and Reginald L Lagendijk. 2018. Transaction propagation on permissionless blockchains: incentive and routing mechanisms. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 20–30.

[35] Ethereum. 2020. *Ethereum Wire Protocol.* https://github.com/ethereum/devp2p/blob/master/caps/eth.md

[36] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. 2016. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 45–59.

[37] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*.

[38] Ittay Eyal and Emin Gün Sirer. 2018. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM* 61, 7 (2018), 95–102.

[39] Giulia Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller, and Pramod Viswanath. 2018. Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 2 (2018), 29.

[40] Sebastian Feld, Mirco Schönfeld, and Martin Werner. 2014. Analyzing the Deployment of Bitcoin's P2P Network under an AS-level Perspective. *Procedia Computer Science* 32 (2014), 1121–1126.

[41] Ryan Fugger. 2004. Money as IOUs in social trust networks & a proposal for a decentralized currency network protocol. *Hypertext document. Available electronically at http://ripple. sourceforge. net* 106 (2004).

[42] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert Van Renesse, and Emin Gün Sirer. 2018. Decentralization in bitcoin and ethereum networks. *arXiv preprint arXiv:1801.03998* (2018).

[43] Adem Efe Gencer, Robbert van Renesse, and Emin Gün Sirer. 2016. Service-oriented sharding with aspen. *arXiv preprint arXiv:1611.06816* (2016).

[44] Arthur Gervais, Srdjan Capkun, Ghassan O Karame, and Damian Gruber. 2014. On the privacy provisions of bloom filters in lightweight bitcoin clients. In *Proceedings of the 30th Annual Computer Security Applications Conference*. 326–335.

[45] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the security and performance of proof of work blockchains. In *Proc. of the 2016 ACM SIGSAC conference on computer and communications security*.

[46] David Goldschlag, Michael Reed, and Paul Syverson. 1999. *Onion routing for anonymous and private internet connections.* Technical Report. NAVAL RESEARCH LAB WASHINGTON DC CENTER FOR HIGH ASSURANCE COMPUTING SYSTEMS ....

[47] Michael T Goodrich and Michael Mitzenmacher. 2011. Invertible bloom lookup tables. In *Allerton*. IEEE.

[48] Matthias Grundmann, Till Neudecker, and Hannes Hartenstein. 2018. Exploiting transaction accumulation and double spends for topology inference in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 113–126.

[49] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. 2019. SoK: Off The Chain Transactions. *IACR Cryptology ePrint Archive* 2019 (2019), 360.

[50] Rachid Guerraoui, Florian Huc, and Anne-Marie Kermarrec. 2013. Highly dynamic distributed computing with byzantine failures. In *Proceedings of the 2013 ACM symposium on Principles of*

*distributed computing*. 176–183.

[51] Bernhard Haeupler. 2016. Analyzing network coding (gossip) made easy. *Journal of the ACM (JACM)* 63, 3 (2016), 1–22.

[52] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium*.

[53] Philipp Hoenisch and Ingo Weber. 2018. Aodv–based routing for payment channel networks. In *International Conference on Blockchain*. Springer, 107–124.

[54] Muhammad Anas Imtiaz, David Starobinski, Ari Trachtenberg, and Nabeel Younis. 2019. Churn in the Bitcoin Network: Characterization and Impact. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 431–439.

[55] Bitcoin Inc. 2015. *Stratum Mining Protocol*. https://en.bitcoin.it/wiki/Stratum_mining_protocol

[56] Bitcoin Inc. 2018. *Network*. https://en.bitcoin.it/wiki/Network

[57] Mercy O Jaiyeola, Kyle Patron, Jared Saia, Maxwell Young, and Qian M Zhou. 2018. Tiny Groups Tackle Byzantine Adversaries. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 1030–1039.

[58] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. 2014. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In *International Conference on Financial Cryptography and Data Security*. Springer, 72–86.

[59] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. 2012. Double-spending Fast Payments in Bitcoin. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*.

[60] Sunny Katkuri. 2018. A survey of data transfer and storage techniques in prevalent cryptocurrencies and suggested improvements. *arXiv preprint arXiv:1808.03380* (2018).

[61] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*. Springer, 357–388.

[62] Seoung Kyun Kim, Zane Ma, Siddharth Murali, Joshua Mason, Andrew Miller, and Michael Bailey. 2018. Measuring Ethereum network peers. In *Proceedings of the Internet Measurement Conference 2018*. 91–104.

[63] Uri Klarman, Soumya Basu, Aleksandar Kuzmanovic, and Emin Gün Sirer. 2018. bloxroute: A scalable trustless blockchain distribution network whitepaper. *IEEE Internet of Things Journal* (2018).

[64] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. 2016. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 279–296.

[65] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 17–30.

[66] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. 2017. Smartpool: Practical decentralized pooled mining. In *26th {USENIX} Security Symposium*.

[67] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. 2017. SilentWhispers: Enforcing security and privacy in credit networks. In *Network and Distributed System Security Symposium*.

[68] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. 2019. Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability.. In *NDSS*.

[69] Alex Manuskin, Michael Mirkin, and Ittay Eyal. 2020. Ostraka: Secure blockchain scaling by node sharding. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 397–406.

[70] Stefano Martinazzi and Andrea Flori. 2020. The evolving topology of the Lightning Network: Centralization, efficiency, robustness, synchronization, and anonymity. *Plos one* 15, 1 (2020), e0225966.

[71] Petar Maymounkov and David Mazieres. 2002. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*. Springer, 53–65.

[72] Patrick McCorry, Surya Bakshi, Iddo Bentov, Sarah Meiklejohn, and Andrew Miller. 2019. Pisa: Arbitration outsourcing for state channels. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 16–30.

[73] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. 2014. Permacoin: Repurposing bitcoin work for data preservation. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 475–490.

[74] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2015. Discovering bitcoin's public topology and influential nodes. (2015).

[75] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. 2020. BDoS: Blockchain Denial-of-Service. In *Proceedings of the 2020 ACM SIGSAC conference on Computer and Communications Security*. 601–619.

[76] A. Mizrahi and O. Rottenstreich. 2020. State Sharding with Space-aware Representations. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 1–9. https://doi.org/10.1109/ICBC48266.2020.9169402

[77] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. 2016. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 305–320.

[78] Bitcoin Relay Network. 2017. *high-speed block-relay system for miners*. http://www.bitcoinrelaynetwork.org/

[79] Till Neudecker. [n.d.]. Characterization of the Bitcoin Peer-to-Peer Network (2015-2018). ([n. d.]).

[80] Till Neudecker and Hannes Hartenstein. 2017. Could network information facilitate address clustering in Bitcoin?. In *International conference on financial cryptography and data security*. Springer, 155–169.

[81] Till Neudecker and Hannes Hartenstein. 2018. Network layer aspects of permissionless blockchains. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 838–857.

[82] Till Neudecker and Hannes Hartenstein. 2019. Short paper: An empirical analysis of blockchain forks in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 84–92.

[83] Thanh Son Lam Nguyen, Guillaume Jourjon, Maria Potop-Butucaru, and Kim Thai. 2019. Impact of network delays on Hyperledger Fabric. *arXiv preprint arXiv:1903.08856* (2019).

[84] Kai Otsuki, Yusuke Aoki, Ryohei Banno, and Kazuyuki Shudo. 2019. Effects of a Simple Relay Network on the Bitcoin Network. In *Proceedings of the Asian Internet Engineering Conference*. 41–46.

[85] A Pinar Ozisik, Gavin Andresen, Brian N Levine, Darren Tapp, George Bissias, and Sunny Katkuri. 2019. Graphene: efficient interactive set reconciliation applied to blockchain propagation. In *Proceedings of the ACM Special Interest Group on Data Communication*. 303–317.

[86] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gazi. 2018. Spacemint: A cryptocurrency based on proofs of space. *Financial Cryptography and Data Security* (2018).

[87] Charles E Perkins et al. 2001. *Ad hoc networking*. Vol. 1. Addison-wesley Reading.

[88] Slush Pool. 2019. *Stratum Mining Protocol*. https://slushpool.com/help/stratum-protocol/

[89] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments.

[90] Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. 2016. Flare: An approach to routing in lightning network. *White Paper* (2016).

[91] Raiden Homepage. 2019. The Raiden Network https://raiden.network/.

[92] Ruben Recabarren and Bogdan Carbunar. 2017. Hardening stratum, the bitcoin pool mining protocol. *Proceedings on Privacy Enhancing Technologies* 2017, 3 (2017), 57–74.

[93] Ripple. 2019. *Overlay*. https://github.com/ripple/rippled/tree/develop/src/ripple/overlay

[94] Ronald L Rivest. 1997. Electronic lottery tickets as micropayments. In *International Conference on Financial Cryptography*. Springer, 307–314.

[95] Elias Rohrer, Julian Malliaris, and Florian Tschorsch. 2019. Discharged Payment Channels: Quantifying the Lightning Network's Resilience to Topology-Based Attacks. *arXiv preprint arXiv:1904.10253* (2019).

[96] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. 2017. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *arXiv preprint arXiv:1709.05748* (2017).

[97] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*.

[98] István András Seres, László Gulyás, Dániel A Nagy, and Péter Burcsi. 2020. Topological analysis of bitcoin's lightning network. In *Mathematical Research for Blockchain Economy*. Springer, 1–12.

[99] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia Fanti, and Pramod Viswanath. 2018. Routing cryptocurrency with the spider network. *arXiv preprint arXiv:1809.05088* (2018).

[100] Weizhao Tang, Weina Wang, Giulia Fanti, and Sewoong Oh. 2019. Privacy-Utility Tradeoffs in Routing Cryptocurrency over Payment Channel Networks. *arXiv preprint arXiv:1909.02717* (2019).

[101] Saar Tochner and Stefan Schmid. 2020. On Search Friction of Route Discovery in Offchain Networks. *arXiv preprint arXiv:2005.14676* (2020).

[102] Saar Tochner, Stefan Schmid, and Aviv Zohar. 2019. Hijacking Routes in Payment Channel Networks: A Predictability Tradeoff. *arXiv preprint arXiv:1909.06890* (2019).

[103] Saar Tochner and Aviv Zohar. 2018. How to pick your friends-a game theoretic approach to p2p overlay construction. *arXiv preprint arXiv:1810.05447* (2018).

[104] Muoi Tran, Inho Choi, Gi Jun Moon, Anh V Vu, and Min Suk Kang. 2020. A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network. (2020).

[105] Carmela Troncoso, Marios Isaakidis, George Danezis, and Harry Halpin. 2017. Systematizing decentralization and privacy: Lessons from 15 years of research and deployments. *Proceedings on Privacy Enhancing Technologies* 2017, 4 (2017), 404–426.

[106] Peter Tschipper. 2016. BUIP010: Xtreme Thinblocks. In *Bitcoin Forum (1 January 2016). https://bitco.in/forum/threads/buip010-passed-xtreme-thinblocks*, Vol. 774.

[107] Stefan Schmid Utz Nisslmueller, Klaus-Tycho Foerster and Christian Decker. 2020. Toward Active and Passive Confidentiality Attacks On Cryptocurrency Off-Chain Networks. In *Proc. 6th International Conference on Information Systems Security and Privacy (ICISSP).*

[108] Marie Vasek, Micah Thornton, and Tyler Moore. 2014. Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In *International conference on financial cryptography and data security*. Springer, 57–71.

[109] Aleksandar Vorkapic. 2018. Secure Blockchain Network Communication using SCION.

[110] Canhui Wang, Xiaowen Chu, and Qin Yang. 2019. Measurement and analysis of the bitcoin networks: A view from mining pools. *arXiv preprint arXiv:1902.07549* (2019).

[111] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. 2019. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 41–61.

[112] Karl Wüst and Arthur Gervais. 2016. *Ethereum eclipse attacks*. Technical Report. ETH Zurich.

[113] Beverly Yang and Hector Garcia-Molina. 2003. PPay: micropayments for peer-to-peer systems. In *Proceedings of the 10th ACM conference on Computer and communications security*. 300–310.

[114] Maxwell Young, Aniket Kate, Ian Goldberg, and Martin Karsten. 2013. Towards practical communication in Byzantine-resistant DHTs. *IEEE/ACM Transactions on Networking (ToN)* 21, 1 (2013), 190–203.

[115] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 931–948.

[116] Alexei Zamyatin, Nicholas Stifter, Philipp Schindler, Edgar R Weippl, and William J Knottenbelt. 2018. Flux: Revisiting Near Blocks for Proof-of-Work Blockchains. *IACR Cryptology ePrint Archive* 2018 (2018), 415.