The Hitchhiker's Guide to Blockchains: A Trust Based Taxonomy

Ramakrishna Thurimella *Yeturu Aahlad*
{ramki.thurimella,yeturu.aahlad}@wandisco.com
WANdisco, Inc.
5000 Executive Parkway, Suite 270
San Ramon CA 94583.

November 30, 2018

Abstract

A trust based taxonomy of blockchains is presented. We consider the evolution of trust and draw parallels to significant societal developments in which information technology tools played a key role. This approach permits us to understand the origins of blockchains, the excitement that currently permeates this space and the promise this technology holds for the future. Besides providing an up-to-date literature survey, we take a critical look at the architectural elements of public and private blockchains and discuss various trade-offs. In addition to demystifying the technology behind blockchains, we demonstrate that not all pieces of the blockchain puzzle are created equal. In particular, we show that consensus mechanisms play a vital role in developing highly available distributed solutions and argue that given that foundation, building distributed applications becomes extremely easy. This argument is supported by sketching the construction of a serverless gold exchange. It is also shown that the barrier to entry for consensus mechanisms is very high. After emphasizing the importance of having the correct consensus mechanism, we offer some guidelines that help in picking from a dizzying array of choices. The paper is concluded by discussing possible topics for future research.

1 Introduction

Many surveys and white papers have been written on blockchains [3, 48]. Why another? Previous surveys either compared the performance of blockchains against that of databases [49], alluded to potential new applications [50, 76], considered their use in enterprises or just simply surveyed the related work [68]. In addition to providing an *up-to-date* literature survey, the material presented here goes beyond the known results in the following ways:

1. We approach this survey through the lens of trust, drawing parallels between the distributed technology developments and the historical shifts in trust from a sociological standpoint. Approaching the taxonomy with this interdisciplinary perspective gives unique insights into how this technology might mature.
2. Most descriptions of public blockchains only explain how they work [101, 109], but do not emphasize the reasons behind design choices. We focus on the latter, with the intent of demonstrating that those early architectural decisions have irreversible implications. This knowledge allows us to compare various choices available within this space and discuss the suitability of each for a given application.
3. An equally important reason for this survey is to demystify the technology behind blockchains and to demonstrate that not all pieces of the blockchain puzzle are created equal. In particular, we show that consensus mechanisms play a vital role and the choice of which one is used can make or break a solution, and give guidelines that help navigate through the maze of many competing choices.

2 Trust

Never depend upon institutions or government to solve any problem. All social movements are founded by, guided by, motivated and seen through by the passion of individuals.

(Margaret Mead)

2.1 Societal Trust

It has been said that in recent times that there is a "crisis of trust." How did we get there? First let us define the term *trust*. For human society to function and operate smoothly, cooperation and interdependence among its members are two key prerequisites. Trust is at the core of these and other such prerequisites; no society can exist and run efficiently without it. Unfortunately, of late, trust in institutions such as banks, corporations, governments, religious organizations, and media has hit an all time low. Recent scandals have only exacerbated the problem. The decision making authority that is concentrated within the hands of a privileged few is deemed unacceptable. This does not mean that we have become a trustless society. Quite the contrary - the rising popularity of sharing economy platforms, such as AirBnb and Uber, is clear evidence that people are willing to share their house with unknown guests for a fee or jump into a stranger's car for a ride. The process of building trust and whom we deem trustworthy seems to have undergone dramatic changes.

According to Botsman [29–32], trust used to be very local - extended only to people within the immediate circle, like friends and relatives. Then it gradually became centralized, with institutions as the key trust brokers. For example, money lending for the most part moved from the neighborhood lender to big banks which in turn rely on large, national credit bureaus. This centralized trust, as argued by Botsman, is giving way to distributed trust, and the advances in digital technology are the driving force behind this revolution.

Is this the end of trust in organizations? Not really. El-Erian [51] posits that credible, effective institutions are fundamental to society, in that they help shield countries from costly shocks resulting from economic, political or social volatility. Imagine a massive sell off of a cryptocurrency based on a rumor. In the absence of institutions, there would be no deposit insurance mandated by governments, the equivalent of the Federal Deposit Insurance Act of the US, to allay the fears of panicked sellers during a bank run. A contrarian viewpoint to the trustless model is also offered by Stinchcombe [99]. The future, therefore, might hold some combination of the two viewpoints: 1) For institutions, it is no longer business as usual and they would be compelled to embrace, whether willingly or not, emerging technologies and pass on the benefits of the resulting efficiencies to their clients, and 2) Trust would take some hybrid form, neither fully centralized nor distributed. Exactly how it is going to evolve and what form it takes is anyone's guess at this point. The next section offers further evidence in support of this argument, why a more hybrid, decentralized form of trust is likely to evolve.

2.2 Subversive Technologies

Until a few decades ago, cryptography was primarily a military tool. Two developments helped bring it into the public domain: the invention of symmetric and asymmetric encryption (private-key and public-key cryptography). It was successfully used in the protection of human rights against oppressive regimes [63]. More recently, social media played an important role in the Arab Spring.

Chaum's pioneering work to protect digital privacy dates back to the early 1980s [42]. He also introduced the notion of distributed untraceable transactions in [40, 41] more than thirty years ago. A comprehensive historical perspective on commercial development of this technology that ultimately led up to the advent of current day cryptocurrencies can be found in [88].

The infamous file sharing system Napster, that came into existence in 1999, similarly challenged the entrenched power structure that existed in the content distribution aspect of the entertainment industry. While this entity has been legally shut down, file sharing has moved to mostly serverless platforms such as Bittorrent (trackers are still centralized and prone to censorship). Bittorrent and other file-sharing applications during the peak of their popularity in 2008 consumed bandwidth similar to major content providers of today like Netflix. According to one study [2], file-sharing accounted for almost 70% of all Internet traffic. Though one could argue that file-sharing applications could serve to distribute legitimate content, the overwhelming majority of the traffic, well over 90%, is illegal [43]. By 2015, this number dropped to a mere 3% [21]. This decline in its popularity could be attributed to the rise of legitimate platforms that make content available at an acceptable cost and stricter copyright laws.

Approximately a decade after the birth of Napster, when file-sharing was riding high in popularity, bitcoin was invented [87]. It came at a time when there was utter disappointment with corruption in financial institutions and the governments that propped them up. Indeed, the genesis block of the bitcoin ledger contained the headline from The Financial Times dated January 3rd, 2009 that refers to a second bailout for banks, expressing Nakamoto's disillusionment with the status quo of the time.

The motivations behind the invention of peer-to-peer file sharing platforms and that of cryptocurrencies is similar - to disrupt the status quo. Given this backdrop, it is tempting to draw parallels between these two systems: both share clever, serverless implementations for the most part, both can be used for legitimate as well as illegitimate purposes, and both drew attention from the academia, governments and regulatory agencies. The rise and fall of file-sharing platforms and the increasing dominance of legitimate, affordable content providers that are displacing them could serve as a harbinger of things to come for the financial sector. By yielding some control back to the consumer by adapting the emerging distributed technologies, traditional financial institutions could provide services that are more competitive and win back the confidence of their consumers.

2.3 What's Next?

Some notable digital trends include distributed, community driven ranking systems, such as the reputation score earned on Q&A forums like StackOverflow or personal ratings on sharing-economy platforms like AirBnb and Uber. There are discussions around maintaining sovereignty over personal data in social networking sites after a serious data breach at Facebook. Surprisingly, China is going exactly in the opposite direction, much to the dismay of privacy advocates. It is considering developing a Social Credit System, where each citizen is assigned a score that not only includes that person's history of paying bills on time but also daily activities, including shopping habits, exercise routines, social interactions, and the amount of time spent online [29].

Ledgers have undergone significant evolution starting with a paper-based double entry general ledger. During the early days of personal computers, software such as Quickbooks helped digitize the accounting process. With the advent of blockchains, the ledgers would most likely become distributed, in order to meet the increased customer expectations of auditability and transparency. Blockchain developments have captured the imagination of people from all walks of life. Ironically, significant investment to innovate in this space is coming from traditional institutions, e.g. academia, government and Big Tech, the very establishments that disruptors would like to overthrow.

It is natural to wonder, given all the benefits distributed, decentralized technology seem to offer, why wasn't it the first choice when transitioning from mainframes to the networked world? Why the circuitous path via client-server computing? Simple: *Centralized solutions are easy to implement compared to their distributed counterparts.* In fact, a good principle when designing a distributed solution is to first design and implement a centralized one, verify correct operation, and finally adapt it to the distributed model. The next section gives a more detailed explanation of these paradigm shifts in computing.

3 Parallels in the Digital Realm

In this section, we consider the local, centralized and distributed trust models and the corresponding computing paradigms with the goal of drawing out the parallels between the two.

3.1 Pre-Networked Era

Local trust corresponds to the pre-networked era of computing - the time of mainframes. These machines lacked connectivity to the outside world. The enterprise which operated the mainframe had to hire operators who sat at terminals and communicated with the enterprise's clients by telephone. The enterprise trusted the computer operators it hired. Accounting in commerce remained manual up until the mid seventies. Trade with remote vendors began with mail order catalogs and the use of the telephone with trust between the vendor and customer. The merchant placed trust in the client for payment, and the merchant is expected to deliver the correct merchandise. If either party betrays this trust, there are institutions in place to punish this kind of behavior, e.g. Better-Business Bureau for customers to report their negative experience and credit-rating agencies to protect businesses from customers who defraud them. These institutions are in active use today.

3.2 Trusted-Third Party (TTP)

The client-server paradigm resembles the trusted-third party (TTP) model. The advent of the World-Wide Web and the age of networked computing introduced the notion of electronic commerce in the 1990s. Servers enabled customers to interact directly by giving them full access to the shopping catalog and allowing them place orders, thus disintermediating the order-processing clerks. While doing so, they recognized and successfully implemented a key *security & privacy* requirement: customer isolation, i.e. insulating and concealing the activities of one user from another. This style of client-server computing roughly corresponds to the centralized trust model, in that clients fully trust the server. Even today, this model reigns supreme and the world's richest individual at the time of this writing, by a very wide margin, is the founder of Amazon. Crash resistance and high availability is achieved by heavily provisioning the server and fortifying the servers against attacks for security. If a user places an order and disavows having ever placed the order after receiving the goods, the vendor can simply request that the goods be returned; if the customer does not comply, the vendor can write it off as the cost of doing business and sever the ties with that customer, report him to credit bureaus, the authorities, or any combination thereof; a weak form of non-repudiation. In the following we show a real-life example of TTP and suggest that the third party need not always be trustworthy.
Example: Untrustworthy Third Party Consider a real-estate transaction in the US where Alice wants to buy a house from Bob. The current practice that is common in most jurisdictions is for Alice to hire buyer's agent Barry, who is familiar with the local area where the property to be purchased is located, and Bob to list the house using a seller's agent Sam, who has a good sense of the market pulse. Generally speaking, the buyer, Alice in this case, cannot come up with the money required, so she seeks the help of a lender Larry. These three intermediaries Barry, Sam and Larry become stakeholders in the transaction, in that it is in their interest to make sure that the transaction executes to completion; otherwise, none of them get paid. There are other intermediaries such as inspectors that come into play, but for the sake of simplicity let us keep them out of our discussion. Given that no one trusts anyone in this kind of a transaction, they all agree to hire a closing agent Charlie, a TTP to facilitate the transaction. Typically it is the buyer's prerogative to pick the closing agent. In reality, for practical reasons, the buyer delegates this responsibility to Barry, assuming that Barry would not betray his trust. This puts Barry in a position of power. Barry gets to pick Charlie and can even pick the inspector. It is important to note that Barry benefits most by quickly closing the deal, collecting his commission and moving on to the next buyer. Charlie and Larry would like to get repeat business from Barry, so they would not do anything to jeopardize the transaction. Bob and Sam, of course, would like get rid of the house so they also will not do anything to rock the boat. In other words, no one is really looking out for Alice. The system is stacked against a novice buyer who is not really protected from potential harm that could come from an unscrupulous buyer's agent. The real-estate industry is notorious for attracting people with questionable ethics and buyers unfortunately are taken advantage of routinely. Clearly, the trust model is broken here. Why is this practice still in use today? Because of its simplicity for a high-valued, complex transaction like a house purchase.
There are two main, orthogonal drawbacks with TTP solutions:

Drawback 1: Single-Point of Failure The notion of a central authority facilitating client transactions, like the server in the client-server model, presents itself as an inadequate solution in fault-tolerant computing because it leads to a single-point of failure, namely if the server fails, the whole systems comes to a grinding halt. Even if a select set of privileged nodes collectively take on the role of a server, while it might eliminate a single-point of failure, they nevertheless become an attractive target for attackers.

Drawback 2: Untrustworthy Server In information security, it has long been recognized that TTP offers simple solutions that are secure, but they come at the cost of making a *big leap of trust.* But as the above example illustrates, this maybe misplaced trust. Blockchain enthusiasts rightfully reject the client-server (centralized) model on several grounds, chiefly taking issue with the concentration of power, citing that TTP is vested with too much authority, imposes unreasonable fees to mediate, and introduces avoidable delays in settlement.

The next section describes how two different research communities sought to address the aforementioned drawbacks and arrived at broadly similar, but very different distributed architectures.

3.3 Leveling the Playing Field

The fault-tolerant computing community addressed Drawback 1 from Section 3.2 by replicating the application state across several servers, thus assuring high availability and guaranteeing progress as long as a majority of the servers are functional. The replication of state is achieved by following an algorithmic process, the so-called *consensus protocol*. The challenge is in maintaining the replicated state in the face of arbitrary failures. More details on consensus are given in Section 7. Two points about this approach are noteworthy: Though all servers are equal peers, there is still a distinction between clients and servers in contrast to the *peer-to-peer* computing model where each node is running client as well as server software, and secondly to arrive at consensus, server nodes should trust each other to follow an algorithmic process.

Cryptocurrency proponents, on the other hand, arrived at a solution in dealing with Drawback 2 by eliminating trust altogether. In this, all nodes are equal - a true form of peer-to-peer computing. More importantly, because of the absence of trust, nodes cannot rely on each other to follow an algorithm. But, nevertheless, nodes must collectively agree on the state of the global data structures. This is achieved by rewarding nodes with incentives for correct behavior. This alternate method of achieving consensus can only give probabilistic guarantees. More details on this method are given in the next section.

Two very different motivations of these two communities are depicted as a fork in the evolution of computing in Figure 1.

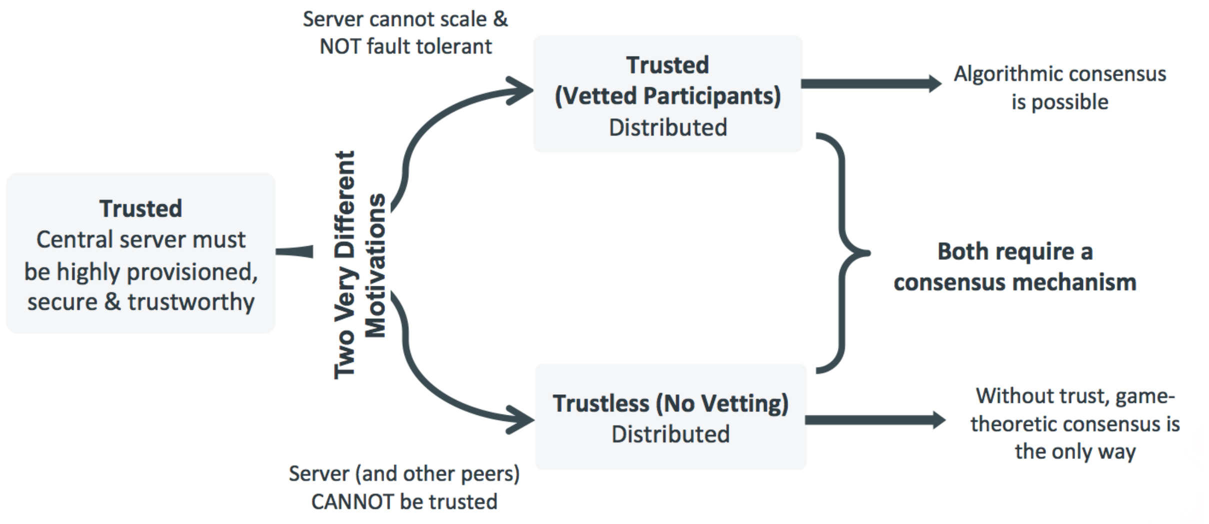

Figure 1: Two very different motivations behind the public and permissioned blockchain ledgers.

What is common to client-server and the two different paradigms of distributed computing? It is the notion of an *intent log*, a record of all clients' intentions to modify the data store. Sufficient information about the operation that will be performed is written to the log so that it can be applied at any later time, even after a failure. This, sometimes in conjunction with other techniques such as shadow-paging [26], ensures recovery without data loss or corruption after a failure. Servers use consensus mechanisms to collaborate and keep this log consistent across all nodes. The intent log for the cryptocurrency world is the familiar blockchain, a single version of the truth that all nodes strive to keep identical. The intent log in the top fork of Figure 1 is the *distributed ledger*. The top and bottom forks have come to be known as the *permissioned* and *public* blockchains respectively.

Let us briefly consider how the *double spend* problem plays out in each of these three models. Consider a scenario where Alice's account contains only one dollar, but she attempts to execute two transactions: one transferring her only dollar to Bob, and the other attempting to give the same dollar to Charlie.

In the centralized model, each transaction as it arrives, written to the log and gets executed. When the second transaction from Alice is attempted, the system recognizes that her account balance is zero during the validation phase and ignores the transaction, rendering it to a no-op.

In the permissioned model, exactly the same sequence of steps takes place because the consensus mechanism ensures that it delivers the two transactions in precisely the same order at all server nodes, first to Bob then to Charlie or the other way around. All servers process the first transaction by recording the intent followed by the successful execution of that transaction. They then update the local ledger accordingly. As with the centralized model, when the second transaction from Alice is encountered, after consulting the local copy of their ledger each node independently realizes that there is no remaining balance in her account and would treat it as a no-op.

For public blockchains, the validation process is a bit more complicated. When Alice attempts to spend same dollar two (or more) times, the validators (i.e. miners) would consider each of her transactions along with other transactions that make up a block and attempt to validate all transactions in that block. It is conceivable that validators would see Alice's dollar as unspent when they go to check the existing ledger, but once one of them adds a block containing a transaction from Alice, no one else can add any block containing other transactions from Alice to that fork of the blockchain. It is true that two blocks containing two different transactions from Alice could get added to two separate forks, but the cryptocurrency protocols make sure that as time progresses only one branch survives as the blockchain evolves. There are can be multitude of problems that can arise with this model of arriving at consensus to maintain the blockchain ledger. Next section examines some of these fundamental limitations with this approach.

4 Destination Disintermediation

Consider the difficulties inherent in operating in a totally trustless environment. Let us take cryptocurrencies as an example. Many papers have been written explaining how some popular cryptocurrencies, e.g. Bitcoin, work. For example, see [88]. Instead of reproducing the inner workings of these platforms, let us look at some key mechanisms that are at play, the rationale for their inclusion, and why they are inextricably woven into the platform design. In our discussion, public, permissionless and trustless are used interchangeably and refer to the same paradigm.

The Anathema of Proof of Work Since the very premise of cryptocurrencies is elimination of all trust, the design of the underlying network must provision for zero trust between every pair of peers. Therefore, it cannot function within the realm of the client-server paradigm, because the server cannot be trusted by any of the clients. As mentioned before, a move to the distributed model is forced in the zero-trust requirement. Stated differently, the distributed network backing a cryptocurrency must allow any user onto the network, has no idea how many there are at a given time and must assume they behave in an unpredictable, possibly malicious, manner, the *Byzantine fault tolerance (BFT) model.* It is intuitively obvious that unless a majority of nodes are honest, nothing useful can be done on this network. We will formalize this notion shortly and in fact show that even if honest nodes are present in a majority, that is not enough to prevent bad things from happening. Since we are limiting our discussion to currencies, clearly a ledger of transactions must be maintained. In a distributed setting, this ledger is the "globally accepted truth" that states the ownership of currencies at a given point in time and the transactions that have taken place thus far that led the ledger to the current state.

With this as the backdrop, let us explore the following questions:

1. **Why can't public blockchains use the well-known Byzantine consensus algorithms for distributed ledger maintenance?** These require a super-majority of honest nodes. Unfortunately, even with one malicious participant, they will not work because that one node can create several fake identities, forcing the number of honest nodes to a minority - the Sybil attack. Since public blockchains, like Bitcoin, by design do not want to know the identities of its participants for privacy reasons, the algorithmic approach is unsuitable.
2. **Why do public blockchains waste so much energy in maintaining the ledger?** In one word, it

is for immutability. In a public, distributed setting achieving immutability is a tall order. In computer science terms, the only operation that can be performed on a ledger is the Append operation. This naturally leads to the definition of blockchain. In a *blockchain*, every block includes a hash of the previous block, with the exception of the very first block - the genesis block B0, resulting in a chain of blocks. Blockchain is reminiscent of a linked list data structure with the current block at the head and the genesis block as the tail. For an excellent visual demo, see [35].

So, if a bad actor, Alice, decides to "rewrite the transaction history" and tamper with the i^{th} block, then all she would have to do is alter that block, compute the new hash, cascade this change to the remaining blocks all the way to the current block, and broadcast this new tampered ledger. It does not take much effort for Alice to mount this attack, because computing a hash does not require much computational work. While Alice is carrying out this attack, Eve could also similarly tamper with the log to suit to her needs and broadcast to the network at the same time. An honest miner witnessing multiple plausible ledgers would have no way of knowing the correct way to extend the ledger. Therefore, without further safeguards this preliminary version of blockchain design is worthless.

This is reminiscent of a problem that is familiar to everyone–spam. If there is no price to send an email, anyone can send an email to anyone and any number of times, flooding Inboxes with unwanted email. On the other hand, if the sender is forced to pay a small price analogous to a postage stamp, say in CPU cycles, before she is allowed to send an email, it would make the job of email spammers very hard, without unduly burdening a legitimate sender. All public blockchains implement a throttling mechanism to prevent "ledger spam." In a nutshell, this is the rationale for requiring Proof of Work when attempting to add a block to the chain. *Proof of Stake* is another well known throttling mechanism.

Bitcoin's answer to realizing a tamper-proof ledger is to require the inclusion of a random string (known as number used once, abbreviated *nonce*) in each block so that the new block to be added to the ledger along with a proper nonce when hashed produces a binary string that has a specified number of leading zeros. The onus of finding the nonce is on the network participant who would like to append a block to the ledger, a miner. Finding a nonce satisfying the leading zero property, by necessity, requires a brute-force search, known as the mining process. Miners are incentivized for their effort; in fact the very first transaction of every block is the incentive given out to the miner. To get an idea of the effort that is needs to be expended, with SHA256 as the hash algorithm and the system requiring k leading zeros, where $k = 40$, a miner would have to try on an average one trillion nonces before finding a suitable one. Currently $k = 18$. To maintain a constant transaction rate, one must progressively increase k to offset the continuously improving computing speeds. Once a solution is found, the new block B is broadcast to the network. Other miners, upon receipt of B, first verify that the transactions and the proof of work included in B are both legitimate. Other miners tacitly express their acceptance of B by working on extending that the fork to which B was was added. As can be seen from this description, the ledger could evolve along multiple paths. As per bitcoin protocol, the surviving branch is that one which is the longest, has the most number of blocks. With this safety measure, to alter a block's content, a new nonce for that block and for all subsequent blocks has to be recomputed. It should be clear from this discussion that the block that is closer to the genesis block is unlikely to be forked and "sidelined" compared to the one that is closer to the ones that have been just added.

A malicious user in a bitcoin network can try to defraud people by creating a fork by entering into a race with other miners, create an alternate ledger and potentially win the race, but at a tremendous cost in CPU cycles. On the other hand, the same work directed towards honest mining can result in possibly bigger profits, by way of incentives. Hence trying to subvert the system is generally not in one's interest. Therefore, the incentive mechanisms that are put in place rely on game-theoretic principles for the correct operation of Bitcoin and assume all miners behave in a way that is profitable to them. This is called the *Nakamoto Consensus*. As a corollary one can conclude that the Bitcoin ledger is a probabilistic ledger, i.e. as more and more blocks are added to a given block, it will get progressively harder to purge that block from the ledger. Blocks closer to the first (genesis) block have a higher probability P of being a "permanent" part of the ledger than the ones that were added most recently, $P(B_i) > P(B_{i+1})$, for all $i \geq 0$. Hence it is worth noting that the consensus (and the state of the ledger) achieved by this means is probabilistic.

One immediate negative consequence of the proof of work mechanism is the dampening effect on throughput, the number of transactions per second: Bitcoin currently can do in the range of 7 tps. Unless there are some radically new designs, it is not clear how public blockchains are going to disrupt the entrenched power structure and challenge the likes of Visa or Mastercard that operate at several orders of magnitude higher than this. Furthermore, some recent credible published sources have concluded that bitcoin mining annual electricity consumption is comparable to that of Ireland (∼24 TWh), and is projected to reach the annual consumption of the Czech Republic within a year [55]. This is simply not sustainable. The bad news for bitcoin does not end here; it is far from achieving the ideal of a total trustless environment, as the next subsection shows.

4.1 In Miners We Trust

Let us now consider the role played by miners in public blockchains. First a thought experiment: imagine a blockchain with a single miner M. If M buys a something from Alice, writes this transaction T to the ledger, Alice inspects ledger, waits until a few more blocks are added after the block containing T and releases the goods to M. The miner M later, could use the same coin and buy something else from Bob by writing a different block with this new transaction to the ledger and insert it before the block that contains T, thus forking and sidelining the branch that has the block containing T. Since there are no other miners trying to compete and extend the original branch, he can do forking after he receives goods from Alice. This is called the double-spend attack and it stems from the fact that M has too much control on the mining process.

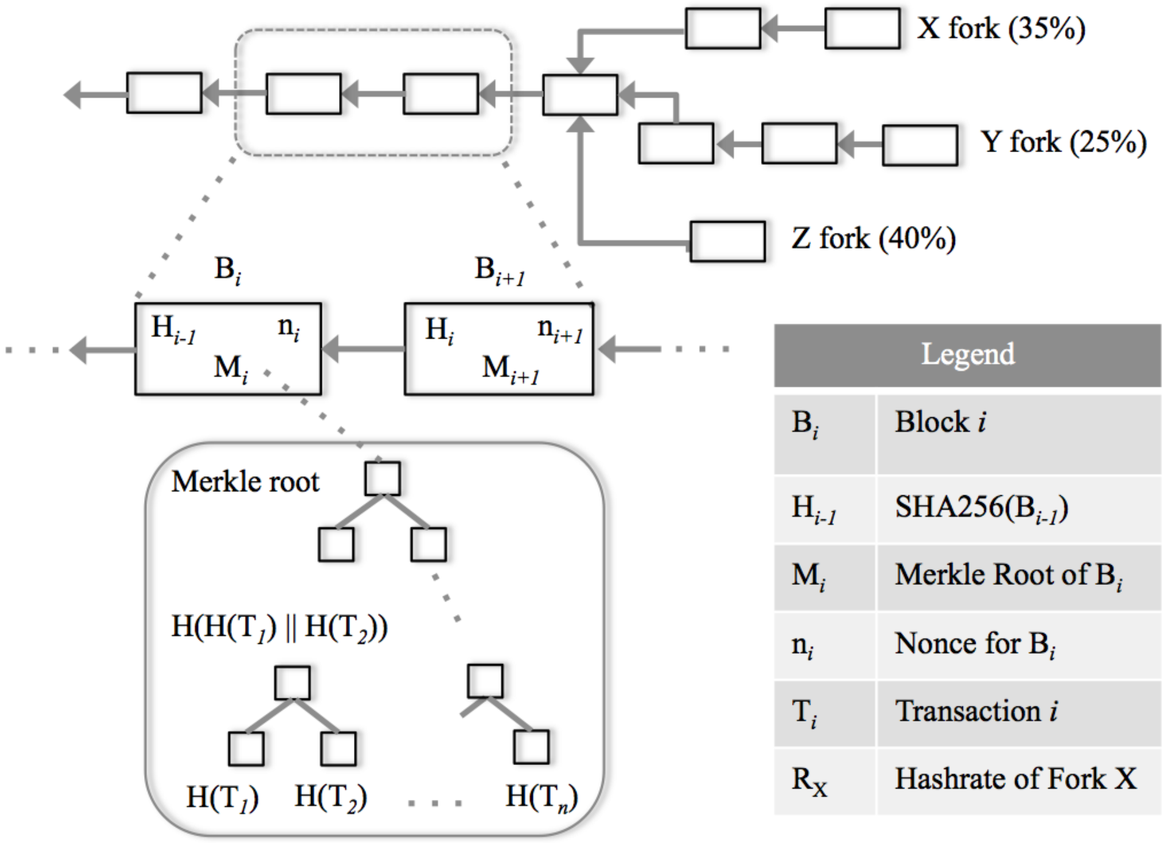

Figure 2: 51% attack can be carried out with less than 51%.

Though the previous discussion is not a proof, it should convince the reader that the more mining power a miner (or group of miners) controls, the higher influence he has in deciding which fork of the ledger grows faster. Having a majority control insures higher mathematical probability of favoring one of the forks to become the longest branch. Note that while the intent of bitcoin is to keep the currency distributed, currently if any 5 of the top 6 miners decide to collude, they would get to control over 51% mining power

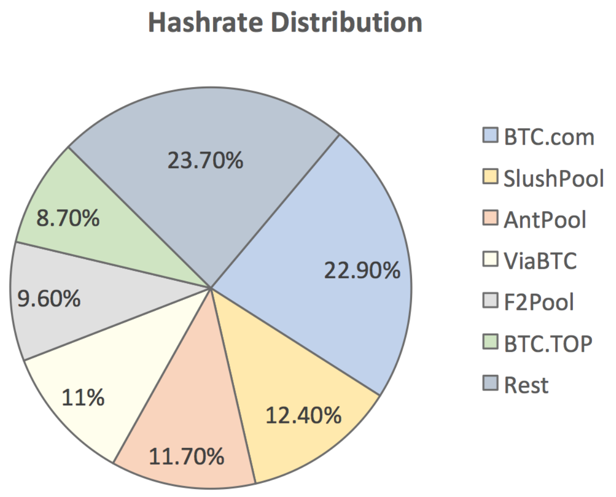

Figure 3: Hashrate distribution by miners.

which would put them in a position to selectively exclude blocks, or fork the network and mount a double spend attack. See Figure 3. What is ironic is that this advantage can be had even with only 40%, as long as the remaining 60% is split in a way that no one else has 40% control. In other words, the so-called 51% attack is a misnomer; it can potentially be carried out with even less than 51%. Stated differently, if the adversary controls 51%, then he is guaranteed to have an advantage, i.e. it is a sufficient condition but it is not necessary. Figure 2 shows three forks X, Y and Z, and their respective hashrates R_X, R_Y , and R_Z that are being expended along those forks. Assume all miners are working on one of theses three forks, i.e. $R_X + R_Y + R_Z = 100\%$. For any set of values such that $R_Z > \max(R_X, R_Y), Z$ fork, though not the longest fork at the moment, could eventually prevail. (For example take $R_X = 30\%, R_Y = 30\%$, and $R_Z = 40\%$.)

Mining power in the hands of only a few was thought of only as a theoretical threat until recently, but has proved to be very real: in May 2018, Bitcoin Gold exchange suffered from such an attack and lost over 18 million dollars [93]. This article notes "...But the rise of massive mining conglomerates that deploy specialized computer equipment has seen a growing centralization of mining in recent years." Some points are noteworthy: 1) there is nothing magical about 51%; in fact, such an attack can be carried out with a lesser percentage, but the probability of success would be smaller, on the other hand 2) it would be apparent to other users that the network is being attacked so that they could take a corrective action. A detailed discussion on the security risks posed by mining and other research directions surrounding Bitcoin can be found in [28, 54].

No discussion on public blockchains is complete without talking about the second most dominant cryptocurrency, Ethereum and the associated smart-contract ecosystem. Bitcoin can be thought of as a pure digital currency and the transactions are direct whereas Ether, the currency of Ethereum, changes hand when the previously agreed upon conditions are met. Though smart contracts were popularized by Ethereum, these were conceived way back in 1996 by Nick Szabo [100].

A simple example of a smart contract is the implementation of a sports bet, similar to an office pool, say on an event like FIFA World Cup, without the use of a trusted-third party. This program would have logic to accept bets from different individuals, wait until the championship is over and distribute the pool to the participants who guessed the winner correctly. Several business rules could be incorporated into the contract. For example, in the event that no one guesses the correct outcome, bets could be returned to the participants or awarded to those who correctly guessed the runner up.

Ethereum uses Solidity, a Turing-complete language, to write smart contracts. In comparison Bitcoin uses a fairly limited opcode based scripting language. An excerpt of a smart contract built around a coin toss written in Solidity is included in Appendix A.

Current throughput of Ethereum is not that different from that of Bitcoin. The Ethereum community considered trading proof-of-work mining in favor of proof of stake (PoS), where the network is secured by the

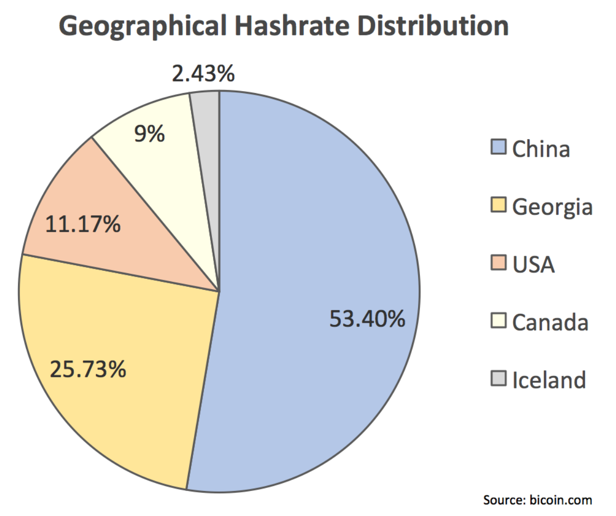

Figure 4: Hashrate distribution by country.

owners of tokens, but decided that a *pure* PoS solution brings its own set of problems. Second generation Ethereum is trying to address these following four current limitations [95, 107]:

1. **Mining**: Moving away from pure PoW to a hybrid scheme that combines PoS, in a new protocol called Casper,
2. **Privacy**: Tightening privacy measures by limiting the visibility of the ledger,
3. **Scale**: Achieving higher scale through sharding (dividing a blockchain network into several smaller component networks capable of processing transactions in parallel) and by creating child blockchain networks within the main blockchains to process information more swiftly, dubbed Plasma, similar to Bitcoin Lightning Network, and
4. **Smart contract security**: Note that any time a computer needs to run a program whose origin cannot be trusted, there is a concern. Different approaches have been proposed to deal with this concern. The second generation is exploring the use of a Python-like smart-contract programming language, "Viper", with the goal of enabling the development of safer Ethereum applications.

With these changes implemented, the Ethereum founder believes that it is on its way to millions of transactions per second. Blockchain applications seem to have found application in the financial technology (fintech) sector [8, 9]. Ethereum with its Smart Contract Eco System is considered to be well suited in this domain. Ethereum specification for enterprise deployment does not prescribe any particular consensus mechanism and leaves it up to the implementer. It has been realized with different consensus methods including Raft [92], an implementation of Castro-Liskov method [38] known as Istanbul and Proof of Authority (PoA) consensus [108]. In PoA, nodes take turns and propose. It should be noted that distributed consensus, as pointed out in Section 7.1, is too complicated to lend itself to a simple round-robin scheme. Two technologies that compete with Ethereum in the permissioned space include R3 Corda [34] and IBM Hyperledger Fabric [64]. See also [103].

4.2 Digital Meets Physical

Some trust is just unavoidable. Consider the problem of tracking newborns in a hospital ward [102]: a baby should not be mistaken for another at any cost. Given the tamper-proof, auditable nature of blockchains, it is tempting to think that they are ideally suited for this problem. We contend that "bootstrapping" the blockchain is not easy without some trust in people involved. Let us breakdown the problem from the beginning. To create a record that unmistakably identifies a baby, an immutable characteristic such as the digital hash of the baby's DNA should get recorded on the birth certificate. In addition, other pertinent information such as the time of birth, parents' names could be included as part of this digital birth certificate.

Let us examine this one level deeper. What is the process to create this digital fingerprint? Some biological information from the baby must be extracted, sent to a lab, and a privacy-preserving digital representation of this data should be created. The chain-of-custody involved in creation of this record leaves no choice but to trust the people involved in this process. The authors term this the "The Last Mile Problem" of blockchains. In other words, as Stinchcombe [99] correctly observed

> It's true that tampering with data stored on a blockchain is hard, but it's false that blockchain is a good way to create data that has integrity.

Now consider another problem that plagues most cryptocurrencies: centralized exchanges that facilitate currency transfers, between crypto and fiat currencies. These are an attractive target for attackers and a constant source of problems. This could also be the place where privacy breaches happen. Indeed, most of the attacks on cryptocurrencies we hear in the news take place exactly at this boundary [75, 86]. It has long been known that illegal activities happen at exchanges [61], spurring new regulations from governments for exchanges to monitor money-laundering and other illegal activities. Ethereum co-founder Vitalik speaking to the necessity of centralized exchanges bemoans [12]:

> ”I definitely personally hope centralized exchanges burn in hell as much as possible. In practice, particularly on the fiat to crypto side, it is very difficult to decentralize because you ultimately are interfacing with the fiat world, and the fiat world is one that only has basically centralized gateways... There are valuable services being provided there that are very hard to decentralize."

Vitalik's disdain for centralized exchanges not only stems from the fact they wield so much power in deciding which currency gets listed, but also their monopoly power in dictating the listing fee. What are the alternatives to centralized exchanges? Trading cryptos for other cryptos could be done using distributed exchange (DEX) solutions. But, they have their own host of problems [10]. With no central coordinator, the onus of making trades and transferring money is completely shifted to the user. DEXs are not regulated or insured. This level of disintermediation clearly does away with any kind of customer support; the users are on their own if there is a problem. So the holy grail of total disintermediation may after all be unattainable. But even if it can be realized partially, which would reduce the cost and time for settlement, it is still a win.

To summarize this section, operating networks in a trustless environment is challenging. In particular, one has to contend with

1. The payment mechanisms needed to modify the ledger, e.g. PoW, and its throttling effect on the transaction rate to thwart Sybil attacks,
2. Trusting miners to do the right thing,
3. Coping with eclipse attacks [62] in the event of network partitions,
4. The last-mile problem,
5. The onus of guarding the valuables is placed on the user, i.e. the *custody* problem [91] and
6. The lack of customer support

These are significant limitations. In the next section, we will see how to overcome these obstacles by relaxing the zero-trust requirement, yet keeping the most attractive part of public blockchains - a distributed, immutable ledger. This is the world of *permissioned blockchains*. Before we do that, one unintended consequence of immutability is worth mentioning - once content makes it to the blockchain, purging it is practically impossible, as hard as tampering with the legitimate history. In other words, there is no Undo button. Why would anyone want to edit the ledger, after the fact? For a couple of reasons: through human error, bad data might make it to the ledger, or an attacker inserts vile content. The current bitcoin ledger contains objectionable, in some instances illegal, content [81] in the optional fields of the block, raising the spectre of illegitimacy by mere possession for mining purposes. There may be a solution in the permissioned world, but who is authorized to edit the ledger, and how it is done would obviously vary depending on the situation.

5 It's Easier To Ask Permission Than It Is To Get Forgiveness

It's easier to ask forgiveness than it is to get permission.

(Rear Admiral Grace Murray Hopper)

We flip the quote by Hopper and demonstrate that it is better to get permission first, rather than to allow anyone onto the network and face the consequences. By restricting access to participants who have been vetted, it becomes easy to build exciting applications based on blockchains, in this *permissioned* setting. Given that blockchain innovation came from cryptocurrencies and challenged the financial industry's current practices, the banking sector first considered it for adoption. Here are three oft-mentioned use cases from this sector:

1. A government bond that is about to mature which if written and maintained with smart contracts, will disburse the maturity amount to the bearer upon maturity,
2. Shorter settlement cycles for corporate clients from syndicated (multiple lenders) loans. Estimated savings are between USD $2 billion and $7 billion annually [8], and
3. Better efficiencies in claims handling in personal motor insurance industry by automation and reduced processing overheads. Estimated savings are USD $21 billion [8].

For other financial technology (*fintech*) applications, see [52].

The successful startup company Everledger [4] is in the business of guaranteeing the provenance (the place of origin and ownership history, commonly used as a guide to the authenticity or quality) of diamonds. There are three important characteristics that are unique to this use case. First, it is a highly-valued asset, like a parcel of land, that appreciates over time. Second, due to the high stakes involved, this business attracts fraudsters, who inject fake diamonds into the supply-chain. Finally, there are multiple intermediaries and stakeholders with different motivations from the time diamonds are mined to the time they belong to a piece of jewelry worn by someone. Some example intermediaries are government authorities needing to collect export/import taxes, owners of the diamond mines, jewelry makers, jewelry buyers etc. The initial record contains a digital fingerprint made from several metadata points, the laser inscription on the girdle, color, clarity, cut, and carat weight of the diamond. Certification regarding the regulation around conflict diamonds could be made part of this record. Once this is available, the ownership as it changes hands can be maintained in a distributed ledger, and made available to participants who are privy to this information. The auditability and accountability needed to protect the supply-chain makes this a perfect use case for the permissioned blockchains [65]. But, Stinchcombe's [99] warning about the difficulty of getting correct data onto the supply-chain must be borne in mind with this application

Finally let us consider the land registration process. After watching people in Haiti fighting over land ownership when their paper ledger got destroyed in a hurricane, a blockchain solution looks very attractive. It can also facilitate the identification of assets seized on tax liens, abandoned properties, or those without 'good titles' – those that are likely targets for fraud. Let us see how the double-spend problem in this context manifests itself, and how the blockchain solution prevents the owner from selling the land to more than one person. The potential buyer could register a smart contract with the local registrar to release the funds from his bank once the ownership has been transferred. The registrar, after verifying the existence of funds in escrow, attempts to add this transaction to the blockchain. There can be two outcomes at this point. First, if the seller is the rightful owner of the land at the time of addition, everything goes as expected, the buyer gets the land and the seller receives the money. On the other hand, if the seller was never an owner of that land or is no longer an owner at the time, the smart contract voids the sale and informs the escrow to return the money to the buyer. The question is whether the registrar or the seller can purge the block containing the previous sale and defraud this buyer? That may be possible in one copy of the ledger, but if the ledger is replicated across different servers, then it is practically impossible to change all the copies.

Finally blockchain applications in the permissioned setting have also been found in copyright protection [14]. The next section walks through a detailed case study of building a gold exchange prototype in the permissioned model.

6 Consensus is the Key (Rest is Mostly Window Dressing)

In this section, we describe a simplified gold exchange, AuX. The primary motivation for its inclusion here is to illustrate the ease with which highly available, secure, fast applications can be built provided a distributed consensus engine (DConE) is available.

First let us start with the requirements for such an exchange. The trading platform AuX, should be capable of accepting trades, either asks or bids, prioritizing them primarily on the competitiveness of the price and secondarily on the time of submission when multiple trades with the same price are encountered. Each customer must be identifiable by a unique ID.

It is easy to encode business rules such as preventing buying and selling to oneself, or trades that exceed the current account balance. Another rule may be when a bid trade X_b is received that is higher than the current best ask price X_a, then much like a market order, to fulfill it immediately, but at price $(X_a + X_b)/2$, favorably beating the expectations of both the buyer and the seller. All transactions could be levied a small transaction fee that is intended to go to the platform provider as an incentive for providing the service. One could model an algorithmic trader to take the role of a market maker to provide liquidity.

In particular, we will evaluate the following three properties of different solutions:

- **Privacy**: Traders' identity and their activity should be private
- **Non-Repudiation**: Once a trade is placed, it cannot be disowned by the trader who placed it
- **Confidentiality**: All communication between the client and the platform, and within the platform must remain encrypted

Trades can be fulfilled as shown below:

```
Algorithm Settle Ask $(p, q)$        ▷ Bid$(p, q)$ is similar
Require: Trade $T$ from client $C$: Sell $q$ units of gold at price $p$ per unit
    1. Idempotence Check if $T$ was already processed. If yes, return .
    2. Business Logic  Apply common-sense rules e.g. make sure that $C$'s account currently has $q$ units
       of gold, $C$ does not have an outstanding Bid with price more than $p$. If $T$ fails these tests, return .
    3. Fulfill
       (a) Insert $T$ into Ask Min Heap $A_{min}$
       (b) if the best Ask price $p_{ask}$ as indicated by the root node of $A_{min}$ is $\leq$ the best Bid price $p_{bid}$ as
           indicated by the root node of Bid Max Heap $B_{max}$ then
             i. delete the root nodes of both $A_{min}$ and $B_{max}$, and
            ii. write this pair of trades to the ledger along with the settlement price as $(p_{ask} + p_{bid})/2$
    4. return
```

6.1 A Simple Implementation

Imagine a platform implementing the algorithm above on a client-server model. Server S operates like a typical current day brokerage firm. The registration process could simply be C generating a public/private key pair of an asymmetric cryptographic scheme, but revealing only the public key part of the pair to associate with an account owned by C. After registration, a client C places a trade T with S. The server attempts to fulfill T immediately; if it cannot, T is persisted to the collection of outstanding trades. Once a matching trade T' arrives, S would attempt to match T against T', write the transaction between T and T' to the ledger, and notify the bidder and seller of its action. Traditional database technology insures the transactional integrity against component failures during this process. Once fulfilled, S could make the relevant portion of the ledger available for inspection upon demand by C at a later time. See Figure 5.

Does this model satisfy the privacy requirement? The answer depends on the regulatory environment in which the exchange is operating. If the regulation demands that all clients be associated to some form of government issued identifier at the time of registration, then the exchange would have to comply and demand a proof of identification from every client. This is the model used by traditional banks and brokerages. On the other hand, some emerging blockchain currency platforms allow the client's account to be linked with only the public key portion of a key pair generated by the client. (Note that there have been statistical correlation attacks on the privacy proffered by this model [24].) The non-repudiation requirement could simply translate to insisting that C sign each trade using the private key corresponding to the public key that is associated with C's account. To meet the confidentiality requirement, trades can be securely transmitted from C using the HTTPS protocol. In addition, S must exercise care while executing and storing transactions, and also at the time of revealing the relevant portions of the history only to the legitimate parties by making sure that they are entitled to receive this information. These are all well-known security best practices, and already exist in most blockchain platforms.

We have described a solution *without any need for a consensus mechanism.* Where does it fall short? As mentioned before, this is the centralized-trust model: server S is assuming the role of a TTP. But, given that operating in a totally trustless environment is hopelessly difficult, what else is wrong with it? It suffers from a single-point of failure and offers no crash resistance whatsoever. A use case such as a gold exchange must promise very high availability. This is where consensus comes into play. As trades are submitted from different geographical locations of a wide-area network, they are replicated across all the participating nodes, and each node maintains a global copy of the heaps A_{min} and B_{max}. At the time of trade fulfillment, each node independently deletes the respective trades from the heaps and persists the transaction to the local ledger. This way, even when f nodes fails in a network of $2f + 1$ participants, the AuX platform can make progress and maintain a consistent distributed ledger. Implementation details can be found in Appendix B.

6.2 Making AuX Production Ready

Our description of a simplified exchange exposed several questions to ponder before it can be deployed into production:

- **Security** Though the nodes are vetted while joining a distributed network, if any one of them gets compromised, then the entire system gets doomed. A chain is as strong as its weakest link. Therefore, the security of the exchange does weaken when one server is replaced with a collection. It is clear that there is built-in Sybil resistance in a permissioned setting, but what other attacks are possible if one of the nodes goes rogue? That is, Byzantine, permissioned model requires careful study.
- **Privacy** Assume nodes running DConE are registered and vetted, all nodes must comply with the local regulations and collect identifying information from their clients. It also appears that if the exchange is operating across international borders, the most restrictive regulation is going to prevail across all nodes of the system.
- **Scale** For a platform that spans a wide-area network, what are the scaling and performance issues, particularly under failure and partition scenarios? What exactly is needed to realize low latency and high throughput? How is the ledger stored and retrieved as it grows unboundedly?
- **Disintermediation** Does the proposed exchange promise any shorter fulfillment times when operated at a global scale? Is it a cheaper alternative to the existing methods? What about the chain-of-custody: How are the assets secured and transferred between different owners? Answers to these questions requires in-depth domain knowledge and is not central to this paper.

7 The Key to Consensus

Those who cannot remember the past are condemned to repeat it. (George Santayana)

Consensus has a long, rich academic history spanning almost 40 years. In this section, we will try navigating through the maze of consensus mechanisms. There is renewed interest in this problem because of the advent of blockchains. There appear to be several surveys on consensus, at least one every decade for the last 4 decades [23, 37, 56, 98]. Consensus is at the heart of many other distributed computing problems and can be used as a primitive to solve them.

The entities taking part in the consensus algorithm can come from {processes, threads, processors, nodes, servers, clients} or more abstractly from {actors, agents, participants}. The meaning should be clear from the context. Note that there may be multiple logical nodes per physical machine.

If two separate nodes running identical software guarantee the same results, we say that the program is *deterministic.* Non-determinism could enter software in the form of program execution that depends on the value of a random number, or on local time, etc. We limit our discussion in this paper only to deterministic software. We will also assume that nodes have unique identities and that there are no anonymous nodes.

The distinction between synchronous and asynchronous is important. In the former model, it is assumed that there is a global clock, the message delay is bounded and that bound is known a priori. This leads to simplification of many algorithms. For example, from the absence of a message one may be able to infer the state of the network. There is extensive literature based on this model. See [84] and the references therein. This work may be useful for multiprocessor parallel computers, but for algorithms that need to run wide-area networks these assumptions are too restrictive. In other words, the asynchronous model assumes that the message delays are finite, but unbounded and unknown. Also, no assumption is made about the drift of local clocks.

Let us now look at the security and fault-tolerance model. In the *fail-stop model*, nodes fail by stopping. When a failed node comes back up, it would retain its identity. Fail-stopped nodes could take arbitrary time to come back alive, or they might have been permanently decommissioned. The messages sent to a node when it is down are simply lost. In this model, nodes, though unreliable, are nonetheless assumed to be not malicious .

For financial applications, such as an online gold exchange, while it may be reasonable to assume that every participant has been vetted at the time of joining, it is unreasonable to assume that they stay trustworthy forever. As mentioned before, a chain is only as strong as its weakest link. Once one of the nodes gets compromised and starts to act maliciously, the fail-stop model is completely inadequate. To account for malicious nodes, we need to define the Byzantine fault tolerance model wherein no assumption is made regarding the behavior of nodes, i.e. nodes could be malicious.

Properties of correct consensus are as follows - given a set of nodes, each with an initial value:

- **Termination** All non-faulty nodes eventually decide on a value,
- **Agreement** All non-faulty nodes that decide do so on the same value, and
- **Validity** The value that has been decided must have been proposed by some node

Agreement and validity are safety properties, while termination is a liveness property. It has been proven that time-bounded consensus, even in the asynchronous fail-stop model, is not always possible [57]. The solutions we see in the literature typically only guarantee eventual liveness or employ randomization to get around this impossibility result.

One easy way out of solving a distributed computing problem is to reduce it to a centralized solution. It can be done like this

1. The nodes elect a leader.
2. The leader does all the computation. Since it is now reduced to a sequential model, this step is simple, and
3. Finally, the leader broadcasts the results from the previous step.

There is a definite lack of aesthetic appeal to this approach. This solution is not symmetric. If the elected leader keeps crashing, the nodes are going to spend most of the time electing a leader instead of doing anything useful. From a security standpoint, the leader must be fully trusted. So, this approach might not work in the Byzantine case. Despite these limitations, leader-based approaches are used quite often in small scale replication solutions. Zookeeper's implementation seem to use this approach [83]. The following text is taken from their website [1]:

> "ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them ,which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed."

In short, this method suffers from all the shortcomings of the trusted-third party solutions.

With these definitions and background out of the way, let us see what is possible. Given the importance of consensus, why are there no production-grade, scalable, open-source implementations, similar to the likes of gcc or Apache HTTP server? This leads us to the next subsection.

7.1 Implementing Consensus Algorithms is Hard

The consensus problem statement looks deceptively simple. In fact, when Lamport originally authored a solution called Paxos (named after a fictional legislative consensus system used on the Paxos island in Greece) and submitted it to a journal, it was rejected as insignificant [60]. Lamport went on to win the Turing Award in 2014. Paxos was mentioned as one of his significant contributions to computer science [74]. To date, it remains one of the best known algorithms for this problem.

Most people looking for an open-source implementation consensus turn to Raft. This algorithm was the outcome of Ongaro's PhD thesis under the guidance of Ousterhout and was proposed as an alternative to Paxos. The authors justify their work by leveling several criticisms against Paxos. Here are some [92]:

> "... Paxos is exceptionally difficult to understand ... We struggled with Paxos ourselves ... it does not provide a good foundation for building practical implementations ... Another problem is that *Paxos uses a symmetric peer-to-peer approach at its core* (though it eventually suggests a weak form of leadership as a performance optimization). This makes sense in a simplified world ... Paxos' formulation may be a good one for proving theorems about its correctness, but real implementations are so different from Paxos that the proofs have little value ... Because of these problems, we concluded that Paxos does not provide a good foundation either for system building or for education"

Criticism in italics runs counter to the conventional wisdom in the distributed computing community and to the argument put forth in the previous section. How can a symmetric approach to distributed computing be a weakness? It should be a strength. Note that Ongaro not only got a PhD for his work from a top-tier institution, but the Raft paper received the *Best Paper Award* at one of the premier conferences in computer systems. At about the same time the Raft authors were railing against the practical relevance of Paxos, a startup company [7] was formed with Paxos as the basis and proved its practical applicability to the distributed systems field. A few years later several suggestions for implementing Paxos appeared [33, 82, 104]. It is also worth noting that Raft was never published in a refereed journal with proofs. It was tested for bugs by a neutral third-party distributed systems validator, Jepsen, and found to have issues [69].

This does not appear to be an isolated occurrence. Another published work from a different top-tier institution proposes a solution for Byzantine fault tolerance [72, 73], receives a *best paper award* from a first-rate systems conference, appears in a respectable journal only to be realized 10 years later that it violates safety. Here is a third example of an *award winning paper* with a liveness violation [78], discovered 12 years after its publication in 2005. This paper got published in an IEEE journal [79] as well! These issues and solutions were reported in [15].

From the foregoing discussion we can conclude

- Consensus is hard [47],
- There is temptation to rush headlong into concocting home brew solutions, and
- It never pays to quickly "hack" something together, underestimating the importance of formalism and thorough reasoning

With these as the guiding principles, we are now ready examine the solutions that have been proposed for blockchains; the subject of the next subsection.

7.2 Consensus Mechanisms Galore

If you can't solve a problem, then there is an easier problem (preferably related to the original!) you can solve: find it. (George Pólya)

The academic world is not alone: "The published, time-tested algorithms are hard to understand; let me invent my own. After all, how hard can it be?" bravado pervades the blockchain industry as well. A high-level survey on various recent consensus implementations is given by Colyer [45]. Also, see [19] and the Related Work section in [23, 96]. There is a rich set of academic literature on using randomization, starting with the seminal paper by Ben-Or [16, 25] in 1983. Since the publication of this paper, literally hundreds of papers have been written based on this approach. See [18, 33] and the references therein. Unfortunately, some recent algorithms, e.g. Swirlds [5, 22] on which hashgraph blockchain is based, that use randomization completely ignore this entire body of work. Then there is PARSEC (Protocol for Asynchronous, Reliable, Secure and Efficient Consensus) [44, 77] that claims to improve on Swirlds, acknowledges the prior art in randomized protocols, but in the conclusion they write " ... makes very weak synchrony assumptions..." Shouldn't that make the title PWSRSEC, with WS standing for Weak Synchronous replacing Asynchronous?

Multi-billion dollar financial blockchain company, Ripple built a hybrid consensus algorithm that is neither centralized nor completely distributed, where the number of servers is limited and all owned by Ripple. Furthermore, the original consensus in Ripple was controversial [67].

When Byzantine tolerance is required, one algorithm that stood the test of time is the one from Castro and Liskov [38] on practical Byzantine fault tolerance (PBFT). The blockchain product from IBM, Hyperledger 0.6, included PBFT as the default consensus mechanism, but surprisingly, and without any justification, replaced it with Kafka + Zab (Zookeeper Atomic Broadcast) ensemble [89] in Hyperledger Fabric [64]. Version 1.4 is slated to be released by the end of the 4th quarter and will offer Raft consensus engine for ordering [13]. One possible explanation for these changes could be lack of performance on a wide-area network[106]. (Refer to the discussion in Section 6 Zab's inability to scale to a wide-area network.) Hyperledger claims a pluggable consensus engine. In Sawtooth, consensus is obtained using a trusted computing platform base and it is based on Proof of Elapsed Time (PoET). In this mechanism, each node starts a timer set to a random number. Every node waits to submit a proposal until the timer is done. The trusted computing base implemented in hardware insures that the creation of the timer and the submission of the proposal is tamper-proof. This idea has yet to prove itself in the marketplace.

Chandra and Toueg proved that atomic broadcast is theoretically equivalent to distributed consensus [39]. Rodrigues and Raynal show how it is used during crash recovery using quorums [97]. Renesse et al [105] do a comparative study of Paxos, Zookeeper Atomic Broadcast and Viewstamped Replication approaches to consensus and conclude that "... compute-intensive services are better off with a passive replication strategy such as used in VSR and Zab (provided that state updates are of a reasonable size). To achieve predictable low-delay performance for short operations during both normal case execution and recovery, an active replication strategy without designated majorities, such as used in Paxos, is the best option." For financial applications that span a wide-area, Paxos appears to be the right choice.

To summarize, the importance of choosing the right consensus mechanism cannot be overstated. Don't get carried away by the outlandish marketing claims regarding transaction rates. For permissioned blockchains, two algorithms stand out for asynchronous applications that span the entire globe: for crash-resistance Paxos is suitable; if Byzantine fault tolerance is also required, then a PBFT implementation as described in [59] or HoneyBadger [85] would be a good starting point.

8 Which Way is North?

These are truly exciting times for developing distributed computing applications. We will conclude this paper by looking at some tantalizing opportunities the recent developments have created and suggesting some future directions. Consider the sharing economy. Imagine a ride hailing service matching drivers with passengers, but without a multi-billion dollar corporation as the intermediary. This could conceivably be built on a purely distributed platform, just like the gold exchange described in Section 5.2, thus achieving true disintermediation resulting in significant cost savings to the participants. This level of disintermediation naturally raises many questions. Who is in charge of vetting the participants, i.e. doing the background checks, vouching for the drivers' skill and the condition of their vehicles, checking the creditworthiness of passengers? What happens when things go wrong - who takes the responsibility? Bootstrapping distributed trust, similar to the concept of web-of-trust that was used in Pretty Good Privacy, but approaching it from a broader perspective [27, 36] requires further study. Sufficient research along these lines in the shared economy space appears to be already underway to warrant a full-day workshop. See BlockSEA [11].

There is no doubt that government regulations directly or indirectly are going to play an important role in the future. We have already seen different countries take different approaches to Bitcoin – a few have banned it, some have allowed it, and it remains controversial in many others. It appears that most countries have taken the wait-and-see stance. Some, like South Korea and China, started out with a ban but now are working on regulating it. Notwithstanding this ambiguity, the cryptocurrency has grown at a very fast pace, amassing a total aggregate market capitalization value of a few hundred billion US dollars. One problem that seems to bog down the public blockchain world is its transaction rate.

There are copious claims on the Internet when it comes to blockchain performance: Hashgraph claims a transaction per second (tps) rate of 500,000 [22, 66], Red Belly's initial claims were even higher [6, 46], and Ethereum 2.0 can supposedly hit the 1 million tps mark sometime this year [95, 107]. Note that when

Red Belly repeated the benchmarking experiment with a more realistic setup, the number dropped by a factor of 20, to 30,000 [90]. These performance numbers are impressive, but could one buy a cup of coffee at the local cafe any time soon using any of these technologies? Probably not. What do these numbers mean for blockchain applications, if anything? What is the latency of these different methods? While there have been some studies comparing the performance of a few selected blockchain implementations against traditional databases such as BLOCKBENCH [49], this field could benefit from developing a comprehensive benchmarking standard. A study guided by scientific principles would be a worthwhile endeavor to establish credible performance standards for blockchains while clearly spelling out the trade-offs. The results would most likely differ based on the application domain.

How can Ethereum 2.0 [17, 95] make such bold claims about its scale? After all, doesn't the PoW/PoS get in the way? As mentioned before, there are two architectural decisions that could pave the way to surmount the scalability obstacle: sharding and sidechains . The former is a well known technique for partitioning data. The latter however is unique to blockchains. Sidechains allow a secure transfer of tokens between two blockchains using cryptographic mechanism known as a two-way peg [20]. Lisk, a platform that enables developers to build and publish decentralized blockchain apps (DApps) using JavaScript, was the first to implement sidechains; each dapp lives in its own sidechain without interfering with the mainchain. The vision of the public blockchain community is to be able run social media platforms and Massively multiplayer online role-playing games (MMORPGs) on a totally decentralized, censorship-free community-driven networks. Konstantopoulos makes a persuasive case for running DApps on sidechains [70]. Another innovation with a similar motivation is off-chain payments using the bitcoin lightning network [58], with Bitcoin as the arbiter, which claims several orders of magnitude improvement in transaction throughput. But, then there is a counterclaim that says that the desired level of performance can only be achieved using centralized banking hubs [94]. Needless to say, how exactly these innovations improve blockchains scale is not clear.

Privacy [53, 71] on blockchains is an active area of research. It has been shown that using the public key of a key pair as a pseudonym is not quite enough. How are the privacy requirements of users reconciled against regulations? There are several open questions surrounding privacy that deserve attention from the research community.

References

[1] Apache Zookeeper. `https://zookeeper.apache.org/`. Accessed: 2018-07-21.

[2] BitTorrent: The "one third of all Internet traffic" Myth. `https://torrentfreak.com/bittorrent-the-one-third-of-all-internet-traffic-myth/`. Accessed: 2018-09-12.

[3] Blockchain library. `https://blockchainlibrary.org/`. Accessed: 2018-07-21.

[4] Everledger. `https://www.everledger.io/`. Accessed: 2018-07-21.

[5] Hedera Hashgraph whitepaper. `http://www.hederahashgraph.com/whitepaper`. Accessed: 2018-07-21.

[6] Red Belly benchmark. `http://redbellyblockchain.io/Benchmark.html`. Accessed: 2018-07-21.

[7] Distributed computing systems and system components thereof [8,364,633], patent issued to WANdisco. `https://patents.justia.com/assignee/wandisco-inc`, Jan 2013. Filed: 2006-01-11.

[8] Smart contracts in financial services: Getting from hype to reality. Capgemini Report. `https://www.capgemini.com/consulting/resources/blockchain-smart-contracts/`, Oct 2016. Accessed: 2018-07-21.

[9] A blueprint for a new RTGS service for the United Kingdom. `https://www.bankofengland.co.uk/paper/2017/a-blueprint-for-a-new-rtgs-service-for-the-uk`, 2017. Accessed: 2018-07-21.

[10] What is a decentralized exchange (DEX)? `http://cryptoincome.io/dex-decentralized-exchange/`, Oct 2017. Accessed: 2018-07-21.

[11] BlockSEA: The 1st workshop on blockchain an sharing economy applications. `http://lab-b.tech/blocksea18/`, Nov 2018. Accessed: 2018-07-21.

[12] Ethereum founder Vitalik Buterin hopes centralized exchanges 'Burn in Hell'. Ethereum News. `https://www.ccn.com/ethereum-founder-vitalik-buterin-hopes-centralized-exchanges-burn-in-hell/`, Jul 2018. Accessed: 2018-07-21.

[13] Hyperledger fabric – pluggable consensus. `https://hyperledger-fabric.readthedocs.io/en/release-1.3/whatis.html#pluggable-consensus`, Nov 2018. Accessed: 2018-11-27.

[14] Microsoft and EY launch blockchain tool for copyright. `http://fortune.com/2018/06/20/microsoft-and-ey-launch-blockchain-tool-for-copyright/`, Jun 2018. Accessed: 2018-07-21.

[15] Ittai Abraham, Guy Gueta, Dahlia Malkhi, and Jean-Philippe Martin. Revisiting fast practical Byzantine fault tolerance: Thelma, Velma, and Zelma. *CoRR*, abs/1801.10022, 2018.

[16] Marcos Aguilera and Sam Toueg. Correctness proof of Ben-Or's randomized consensus algorithm. *Distributed Computing*, 25(5):371–381, 10 2012.

[17] Ian Allison. Enterprise Ethereum Alliance is back – and it's got a roadmap to prove it. `https://www.coindesk.com/enterprise-ethereum-alliance-isnt-dead-got-roadmap-prove/`, May 2018. Accessed: 2018-07-21.

[18] James Aspnes. Randomized protocols for asynchronous consensus. *Distrib. Comput.*, 16(2-3):165–175, September 2003.

[19] Pierre-Louis Aublin, Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 bft protocols. *ACM Trans. Comput. Syst.*, 32(4):12:1–12:45, January 2015.

[20] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. `http://www.blockstream.com/sidechains.pdf`, 2014. Accessed: 2018-07-21.

[21] Jonathan Bailey. The long, slow decline of BitTorrent: The long hard road out of hell... PlagiarismToday. `https://www.plagiarismtoday.com/2017/06/01/the-long-slow-decline-of-bittorrent/`, Jun 2017. Accessed: 2018-07-21.

[22] Leemon Baird. The Swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance. Technical Report SWIRLDS-TR-2016-01, Swirlds, 1 2016. `https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf`.

[23] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchains. *ArXiv e-prints*, November 2017.

[24] Jaume Barcelo. User privacy in the public bitcoin blockchain. `http://www.dtic.upf.edu/~jbarcelo/papers/20140704_User_Privacy_in_the_Public_Bitcoin_Blockchain/paper.pdf`, Jul 2014. Accessed: 2018-07-21.

[25] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, PODC '83, pages 27–30, New York, NY, USA, 1983. ACM.

[26] Philip A Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[27] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, SP '96, pages 164–173, Washington, DC, USA, 1996. IEEE Computer Society.

[28] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for Bitcoin and cryptocurrencies. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, SP '15, pages 104–121, Washington, DC, USA, 2015. IEEE Computer Society.

[29] Rachel Botsman. Big data meets Big Brother as China moves to rate its citizens. WIRED. `https://www.wired.co.uk/article/chinese-government-social-credit-score-privacy-invasion`, Oct 2017. Accessed: 2018-07-21.

[30] Rachel Botsman. How the blockchain is redefining trust. Wired. `https://www.wired.com/story/how-the-blockchain-is-redefining-trust/`, Dec 2017. Accessed: 2018-07-21.

[31] Rachel Botsman. Trust in 2030 – from institutions to individuals. `https://www.weforum.org/agenda/2017/11/trust-score-2030-airbnb-facebook/`, Nov 2017. Accessed: 2018-07-21.

[32] Rachel Botsman. *Who Can You Trust?: How Technology Brought Us Together – and Why It Could Drive Us Apart.* Penguin Books Limited, City of Westminster, London, England, 2017.

[33] Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast Byzantine agreement. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 57–64, New York, NY, USA, 2013. ACM.

[34] Richard Gendal Brown. Introducing R3 Corda: A distributed ledger designed for financial services. Blog. `https://gendal.me/2016/04/05/`, Apr 2016. Accessed: 2018-07-21.

[35] Anders Brownworth. Blockchain 101 - a visual demo. Video. YouTube. `https://www.youtube.com/watch?v=_160oMzblY8`, Nov 2016. Accessed: 2018-07-21.

[36] Christian Cachin. Distributing trust on the internet. In *Proceedings of the 2001 International Conference on Dependable Systems and Networks (Formerly: FTCS)*, DSN '01, pages 183–192, Washington, DC, USA, 2001. IEEE Computer Society.

[37] Christian Cachin and Marko Vukolić. Blockchain consensus protocols in the wild. *CoRR*, abs/1707.01873, 2017.

[38] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, November 2002.

[39] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, March 1996.

[40] David Chaum. Blind signatures for untraceable payments. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *Advances in Cryptology Proceedings of Crypto 82*, pages 199–203, 1983.

[41] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, October 1985.

[42] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[43] Jacquit Cheng. Only 0.3% of files on BitTorrent confirmed to be legal: Nearly all of the files available on BitTorrent are of the illegal variety. Ars Technica. `https://arstechnica.com/tech-policy/2010/07/only-03-of-files-on-bit-torrent-confirmed-to-be-legal/`, Jul 2010. Accessed: 2018-07-21.

[44] Pierre Chevalier, Bartłomiej Kamiński, Fraser Hutchison, Qi Ma, and Spandan Sharma. Protocol for asynchronous, reliable, secure and efficient consensus (PARSEC). `http://docs.maidsafe.net/Whitepapers/pdf/PARSEC.pdf`, Jun 2018. Accessed: 2018-07-21.

[45] Adrian Colyer. Can't we all just agree? The Morning Paper. A ten-part series on consensus and replication. https://blog.acolyer.org/2015/03/01/cant-we-all-just-agree/, Mar 2015. Accessed: 2018-07-21.

[46] Tyler Crain, Mikel Gramoli, Vincent Larrea, and Michel Raynal. (Leader/Randomization/Signature)-free Byzantine consensus for consortium blockchains. http://csrg.redbellyblockchain.io/doc/ConsensusRedBellyBlockchain.pdf, May 2017. Accessed: 2018-07-21.

[47] Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv.*, 36(4):372–421, December 2004.

[48] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Ji Wang. Untangling blockchain: A data processing view of blockchain systems. *IEEE Trans. Knowl. Data Eng.*, 30(7):1366–1385, 2018.

[49] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. BLOCKBENCH: a framework for analyzing private blockchains. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1085–1100, New York, NY, USA, 2017. ACM.

[50] Kevin Doubleday. Blockchain for 2018 and beyond: A (growing) list of blockchain use cases. Medium. https://medium.com/fluree/blockchain-for-2018-and-beyond-a-growing-list-of-blockchain-use-cases-37db7c19fb99, Jun 2018. Accessed: 2018-07-21.

[51] Mohamed A. El-Erian. Why retaining trust in institutions matters. World Economic Forum. https://www.weforum.org/agenda/2017/10/why-retaining-trust-in-institutions-matters, Oct 2017. Accessed: 2018-07-21.

[52] Ittay Eyal. Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities. *Computer*, 50(9):38–49, 2017.

[53] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, pages 45–59, Berkeley, CA, USA, 2016. USENIX Association.

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

[55] Peter Fairley. Blockchain world - feeding the blockchain beast if Bitcoin ever does go mainstream, the electricity needed to sustain it will be enormous. *IEEE Spectrum*, 54:36–59, 2017.

[56] Michael J. Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *Proceedings of the 1983 International FCT-Conference on Fundamentals of Computation Theory*, pages 127–140, London, UK, UK, 1983. Springer-Verlag.

[57] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985.

[58] Jonald Fyookball. Mathematical proof that the Lightning Network cannot be a decentralized Bitcoin scaling solution. Medium. https://medium.com/@jonaldfyookball/mathematical-proof-that-the-lightning-network-cannot-be-a-decentralized-bitcoin-scaling-solution-1b8147650800, Jun 2017. Accessed: 2018-07-21.

[59] Guy Golan-Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. SBFT: a scalable decentralized trust infrastructure for blockchains. *CoRR*, abs/1804.01626, 2018.

[60] Lawrence Goodman. The stuff of genius. Brandeis Magazine. http://www.brandeis.edu/magazine/2017/fall/featured-stories/lamport.html, Fall 2017. Accessed: 2018-07-21.

[61] Sarah Gruber. Trust, identity, and disclosure: Are Bitcoin exchanges the next virtual havens for money laundering and tax evasion? *Quinnipiac Law Review*, 32(1):135–208, Nov 2013.

[62] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 129–144, Berkeley, CA, USA, 2015. USENIX Association.

[63] Russell D. Hoffman. Interview with author of PGP (Pretty Good Privacy). an interview on radio show HIGH TECH TODAY. `http://www.animatedsoftware.com/hightech/philspgp.htm`, Feb 1996. Accessed: 2018-07-21.

[64] Hyperledger. Linux foundation. `https://www.hyperledger.org/`, Dec 2015. Accessed: 2018-07-21.

[65] Tyler Jenks. Top blockchain use cases for supply chain management. `https://www.verypossible.com/blog/top-blockchain-use-cases-for-supply-chain-management`, Feb 2018. Accessed: 2018-07-21.

[66] Yaoqi Jia. Demystifying hashgraph: Benefits and challenges. Hackernoon. `https://hackernoon.com/demystifying-hashgraph-benefits-and-challenges-d605e5c0cee5`, Nov 2017. Accessed: 2018-07-21.

[67] Sid Kalla. Ripple/stellar consensus system may have serious issues. Brave Newcoin. `https://bravenewcoin.com/news/ripplestellar-consensus-system-may-have-serious-issues/`, Dec 2014. Accessed: 2018-07-21.

[68] Akanksha Kaushik, Archana Choudhary, Chinmay Ektare, Deepti Thomas, and Syed Akram. Blockchain - literature survey. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 2145–2148, May 2017.

[69] Kyle Kingsbury. Jepsen: etcd and Consul. `https://aphyr.com/posts/316-jepsen-etcd-and-consul`, Jun 2014. Accessed: 2018-07-21.

[70] Georgios Konstantopoulos. DAppChains: Scaling Ethereum DApps through sidechains, Medium. `https://medium.com/loom-network/dappchains-scaling-ethereum-dapps-through-sidechains-f99e51fff447`, Feb 2018. Accessed: 2018-07-21.

[71] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 839–858, May 2016.

[72] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: Speculative Byzantine fault tolerance. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 45–58, New York, NY, USA, 2007. ACM. Best Paper Award.

[73] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: Speculative Byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27(4):7:1–7:39, January 2010.

[74] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[75] Timothy B. Lee. A brief history of Bitcoin hacks and frauds. Ars Technica. `https://arstechnica.com/tech-policy/2017/12/a-brief-history-of-bitcoin-hacks-and-frauds/`, Dec 2017. Accessed: 2018-07-21.

[76] Josep Lluis de la Rosa, Victor Torres-Padrosa, Andrés el Fakdi, Denisa Gibovic, Hornyák O., Lutz Maicher, and Francesc Miralles. A survey of blockchain technologies for open innovation. In *4th Annual World Open Innovation Conf.*, pages 14–15, 2017.

[77] Maidsafe. PARSEC: A paradigm shift for asynchronous and permissionless consensus. `https://medium.com/safenetwork/parsec-a-paradigm-shift-for-asynchronous-and-permissionless-consensus-e312d721f9d8`, May 2018. Accessed: 2018-07-21.

[78] J. P. Martin and L. Alvisi. Fast Byzantine consensus. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 402–411, June 2005. Best Paper Award.

[79] Jean-Philippe Martin and Lorenzo Alvisi. Fast Byzantine consensus. *IEEE Trans. Dependable Secur. Comput.*, 3(3):202–215, Jul 2006.

[80] Steve Marx. Flipping a coin in Ethereum, program the blockchain: a blog for developers about cryptocurrencies and smart contracts. `https://programtheblockchain.com/posts/2018/03/16/flipping-a-coin-in-ethereum/`, Mar 2018. Accessed: 2018-07-21.

[81] Roman Matzutt, Jens Hiller, Martin Henze, Jan Henrik Ziegeldorf, Dirk Müllmann, Oliver Hohlfeld, and Klaus Wehrle. A quantitative analysis of the impact of arbitrary blockchain content on Bitcoin. In *Proceedings of the 22nd International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2018.

[82] David Mazières. Paxos made practical. Unpublished manuscript, 2007.

[83] André Medeiros. Zookeeper's atomic broadcast protocol: Theory and practice. Helsinki University of Technology - Laboratory of Theoretical Computer Science. `http://www.tcs.hut.fi/Studies/T-79.5001/reports/2012-deSouzaMedeiros.pdf`, Mar 2012. Accessed: 2018-07-21.

[84] Silvio Micali. Byzantine agreement, made trivial. `https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Distributed%20Computation/BYZANTYNE%20AGREEMENT%20MADE%20TRIVIAL.pdf`, 2017. Accessed: 2018-07-21.

[85] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The Honey Badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 31–42, New York, NY, USA, 2016. ACM.

[86] Tyler Moore and Nicolas Christin. Beware the middleman: Empirical analysis of Bitcoin-exchange risk. In *Proceedings of Financial Cryptography 2013*, April 2013.

[87] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/en/bitcoin-paper`, Oct 2008. Accessed: 2018-07-21.

[88] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, Princeton, NJ, USA, 2016.

[89] Neha Narkhede and Flavio Junqueira. Distributed consensus reloaded: Apache ZooKeeper and replication in Apache Kafka. `https://www.confluent.io/blog/distributed-consensus-reloaded-apache-zookeeper-and-replication-in-kafka/`, Aug 2015. Accessed: 2018-07-21.

[90] George Nott. Fork-free, energy efficient red belly blockchain validated in global test. `https://www.cio.com.au/article/647270/fork-free-energy-efficient-red-belly-blockchain-validated-global-test/`, Sept 2018. Accessed: 2018-11-27.

[91] Ouriel Ohayon. The sad state of crypto custody. `https://techcrunch.com/2018/02/01/the-sad-state-of-crypto-custody/`, May 2018. Accessed: 2018-08-21.

[92] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In Garth Gibson and Nickolai Zeldovich, editors, *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320, Berkeley, CA, USA, 2014. USENIX Association.

[93] Charlie Osborne. Bitcoin Gold suffers double spend attacks, $17.5 million lost. ZDNet. `https://www.zdnet.com/article/bitcoin-gold-hit-with-double-spend-attacks-18-million-lost/`, May 2018. Accessed: 2018-07-21.

[94] Joseph Poon and Thaddeus Dryja. The Bitcoin Lightning Network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf, Jan 2016. Accessed: 2018-07-21.

[95] Giulio Prisco. The Ethereum Killer Is Ethereum 2.0: Vitalik Buterin's roadmap. Bitcoin Magazine. https://bitcoinmagazine.com/articles/ethereum-killer-ethereum-20-vitalik-buterins-roadmap/, Nov 2017. Accessed: 2018-07-21.

[96] Team Rocket. Snowflake to Avalanche: a novel Metastable consensus protocol family for cryptocurrencies. https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV, May 2018. Accessed: 2018-07-21.

[97] Luís Rodrigues and Michel Raynal. Atomic broadcast in asynchronous crash-recovery distributed systems and its use in quorum-based replication. *IEEE Trans. on Knowl. and Data Eng.*, 15(5):1206–1217, September 2003.

[98] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, December 1990.

[99] Kai Stinchcombe. Blockchain is not only crappy technology but a bad vision for the future. Medium. https://medium.com/@kaistinchcombe/decentralized-and-trustless-crypto-paradise-is-actually-a-medieval-hellhole-c1ca122efdec, Apr 2018. Accessed: 2018-07-21.

[100] Nick Szabo. Smart contracts: Building blocks for digital markets, Extropy #16. http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html, 1996. Accessed: 2018-07-21.

[101] Florian Tschorsch and Bjorn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, FebMar 2016.

[102] Catherine Tucker and Christian Catalini. What blockchain can't do. Harvard Business Review. https://hbr.org/2018/06/what-blockchain-cant-do, Jun 2018. Accessed: 2018-07-21.

[103] Martin Valenta and Philipp Sandner. Comparison of Ethereum, Hyperledger Fabric and Corda. Frankfurt School Blockchain Center (FSBC), FSBC working paper. http://explore-ip.com/2017_Comparison-of-Ethereum-Hyperledger-Corda.pdf, Jun 2017. Accessed: 2018-07-21.

[104] Robbert Van Renesse and Deniz Altinbuken. Paxos made moderately complex. *ACM Comput. Surv.*, 47(3):42:1–42:36, February 2015.

[105] Robbert van Renesse, Nicolas Schiper, and Fred B. Schneider. Vive la différence: Paxos vs. viewstamped replication vs. zab. *IEEE Transactions on Dependable and Secure Computing*, 12(4):472–484, July 2015.

[106] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[107] Joseph Young. Scalability struggles of leading blockchain networks. Ethereum News. https://www.ccn.com/vitalik-buterin-ethereum-will-eventually-achieve-1-million-transactions-per-second/, Jun 2018. Accessed: 2018-07-21.

[108] Jim Zhang. Consensus algorithms: PoA, IBFT or Raft? https://kaleido.io/consensus-algorithms-poa-ibft-or-raft/, Feb 2018. Accessed: 2018-08-21.

[109] Aviv Zohar. Bitcoin: Under the hood. *Commun. ACM*, 58(9):104–113, August 2015.

Appendices

A Smart Contract Example in Solidity

Listing 1 shows an excerpt of a program in Solidity that implements a coin toss. Alice devises a contract consisting of a Coin Toss game. In this, she chooses heads or tails and commits her choice. Anyone, who guesses the committed value correctly wins the prize money. Bob, decides to play, and make a guess. The code needed to make sure that there are not multiple people wanting to accept the bet, Alice cannot change her original choice, and other safety measures are omitted for brevity. A completely worked out example can be found in [80].

```
contract CoinToss {
address public alice;
address public bob;
bool public bobChoice;
bytes32 public aliceCommitment;
//commitment is SHA256(toss + nonce)where toss = 0 or 1

function CoinToss(bytes32 commitment) public payable
{ //Alice sets up a coin toss game
alice = msg.sender;
aliceCommitment = commitment;
}
function takeBet(bool choice) public payable
{ //Bob takes the bet
bob = msg.sender;
bobChoice = choice;
}
//code for Alice to cancel if no player to bet
...
function reveal(bool choice, uint256 nonce) public {   //Alice reveals
//code to insure Alice has not changed the commitment
if (bobChoice == choice) {
bob.transfer(address(this).balance);
} else {
alice.transfer(address(this).balance); }
}
function claimTimeout() public {
//Bob can claim timeout if Alice refuses to reveal
require(now >= expiration);
bob.transfer(address(this).balance);
}
...
}

```

Listing 1: Coin toss example in Solidity

B AuX: Distributed Implementation of a Simple Gold Exchange

In a distributed implementation, there are several servers behind the scenes, giving the illusion of a single platform to the user. For the sake of simplicity, let us assume that none of the nodes are malicious, i.e. the non-Byzantine model. But otherwise, we would like to make it general enough, by assuming that trades can arrive from geographically dispersed servers asynchronously, contending for a spot in the global sequence to be considered for fulfillment The platform creates proposals out of these trade requests and hands them over to DConE to arrive at a consensus order, i.e. a global sequence of agreements, which then is made available in a fault-tolerant manner to all nodes. Once agreements are sequenced, the trades from them are used as input to an identical application software running deterministically, much like the description given in Section 6. Since multiple copies of deterministic software running on the same inputs, undergoes an identical set of state changes, it results in a replicated state machine producing an identical distributed ledger.

A key feature of this platform is crash resistance - progress can be made in achieving consensus about this global sequence order as long as a majority of the nodes are functioning at any given time. From the application programmer's point of view, it is sufficient to submit DConE proposals, and consume agreements from the global sequence that are output. As an aside, note that it is possible to guarantee a FIFO order for proposals originating from the same server, that is, if a proposal p arrives before q at a server, then the agreement p appears before q in the global sequence, thus promising a notion of fairness.

Note the importance of the following fault-tolerant property which follows from the crash-resistance guaranty of DConE: Under node or link failures, and potential network partition, the order in which the agreements output by DConE is identical across all nodes. Therefore, as long as the application is *deterministic*, its state can be replicated reliably. This is essential to AuX correctness, viz. the pending trades and the ledger, remain consistent across all node instances. It is important to note that at any given instant, all nodes may not return the same output when queried - just that identical results can be observed when the system reaches a *quiescent state*, a moment in time when all the submitted proposals have been agreed upon, there are no new inputs, and every node instance is aware of all the agreements. For example, if a node crashes and comes back up, for the system to come to a quiescent state, this recovering node must learn the agreements it has missed while it was down.

Fulfilled orders are entered into a ledger, in which every entry at the very minimum should show the transaction ID, buyer's ID, seller's ID, and the trading price. One way to verify that the data across all nodes is identical under failures in a quiescent state is by querying and examining data from each node using a browser. Towards that end, a REST API could be built using which the ledger from various node instances can be examined while simulating different failure scenarios. Similarly the trading history of a customer can be queried.

Figure 6 shows the architecture of AuX with 3 node instances. AuX settles its trades by dequeuing agreements from the global sequence and processing them using **Algorithm Settle Ask** (p, q).

Privacy and non-repudiation requirements can be met exactly in the same way as in the client-server model. The only change needed to meet the confidentiality requirement is to require different node instances running DConE to communicate with each other using SSL.

Each node instance maintains a copy of this ledger. DConE guarantees that AuX would continue to make progress under the failure of no more than f simultaneous node failures in an implementation involving $2f+1$ nodes. (For example, with a five-node system, even if two nodes fail at the same time, AuX continues to make progress.) As AuX maintains the customer accounts and the ledger, and since we assumed a non-Byzantine threat model, the accounts and the ledger would remain tamper-proof throughout.

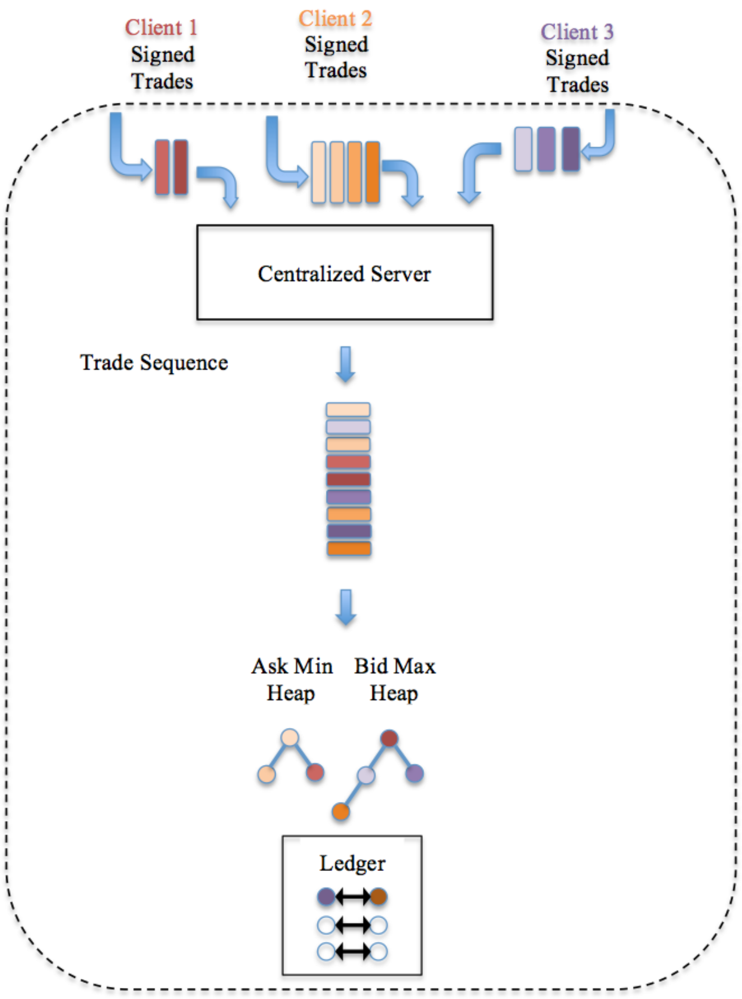

Figure 5: A simple gold exchange - client-server implementation.

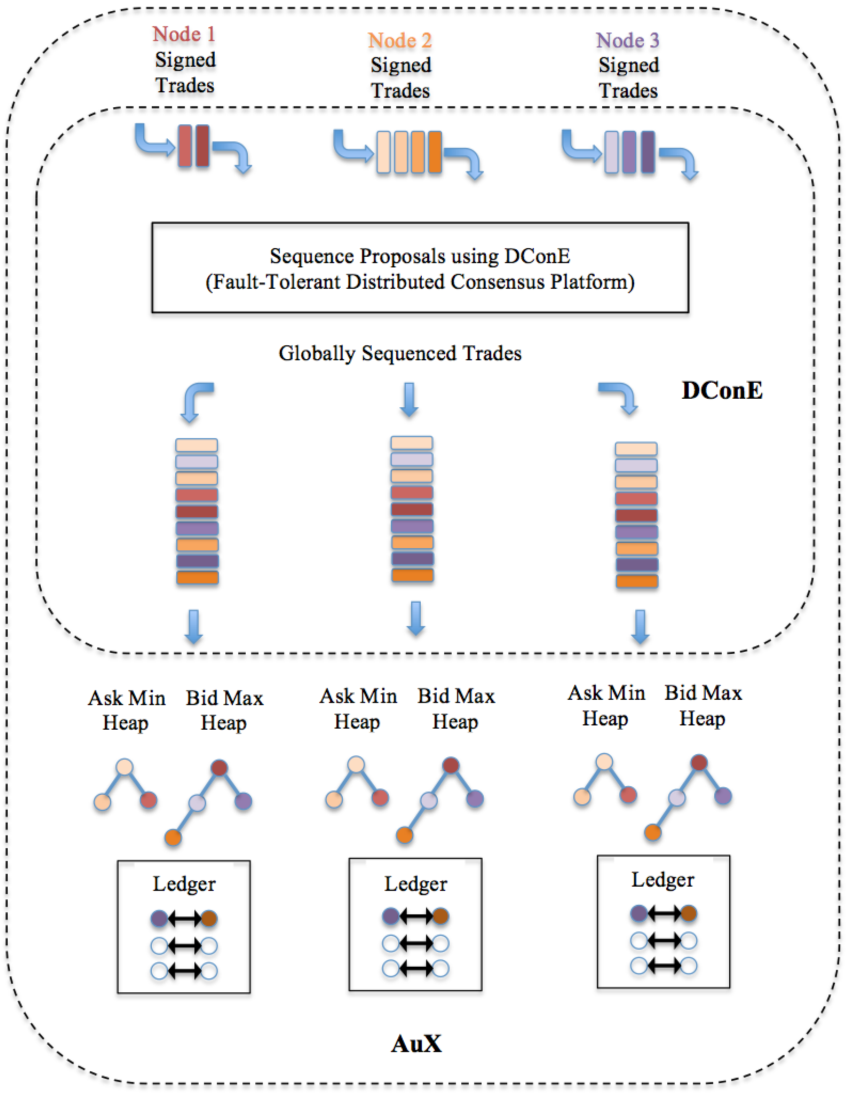

Figure 6: A simple gold exchange - distributed implementation.