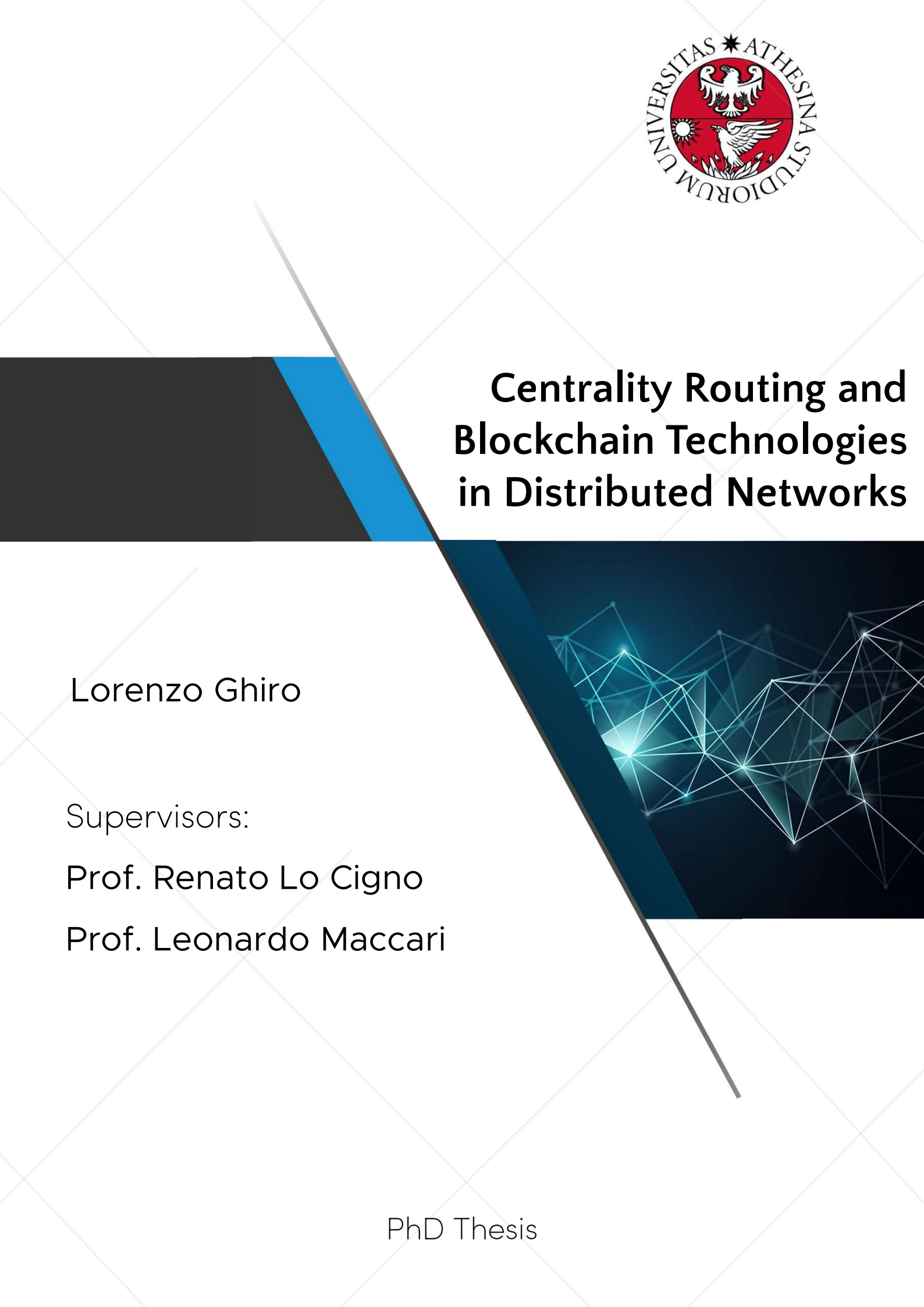


Centrality Routing and Blockchain Technologies in Distributed Networks

Lorenzo Ghiro

Supervisors:

Prof. Renato Lo Cigno

Prof. Leonardo Maccari

PhD Thesis

University of Trento
Department of Information Engineering and Computer Science
Via Sommarive 9, I-38123, Povo (TN)

PhD Advisors	Prof. Renato Lo Cigno Prof. Leonardo Maccari
Defense committee	Prof. Giuseppe Bianchi Prof. Leandro Navarro Moldes Prof. Paolo Casari
Defense date	Wednesday, May 19, 2021

"Prediction is very difficult, especially if it's about the future."
— Nils Bohr*, Nobel laureate in Physics*

"Science is essentially an anarchic enterprise: theoretical anarchism is more humanitarian and more likely to encourage progress than its law-and-order alternatives."
— Paul Karl Feyerabend*, Against Method*

"This turkey found that, on his first morning at the turkey farm, he was fed at 9 a.m. However, being a good inductivist, he did not jump to conclusions. He waited until he had collected a large number of observations of the fact that he was fed at 9 a.m., and he made these observations under a wide variety of circumstances, on Wednesdays and Thursdays, on warm days and cold days, on rainy days and dry days. Each day, he added another observation statement to his list. Finally, his inductivist conscience was satisfied and he carried out an inductive inference to conclude, "I am always fed at 9 a.m.". Alas, this conclusion was shown to be false in no uncertain manner when, on Christmas eve, instead of being fed, he had his throat cut. An inductive inference with true premises has led to a false conclusion."

— Bertrand Russell, The Problems of Philosophy

Abstract

This thesis contributes to the development of *distributed networks* proposing:

- a technique to enhance the reliability of DV routing protocols;
- a critical analysis of the integration of blockchains in distributed networks.

First, a novel algorithm for the distributed computation of the Load Centrality (LC), a graph centrality metric, is proposed and then applied for steering the optimization of the route recovery process of Distance-Vector (DV) routing protocols: this way the algorithm contributes to the enhancement of the network reliability. The algorithm convergence is proved also identifying time complexity bounds that are later confirmed by computer simulations. The proposed algorithm is designed as an extension to the Bellman-Ford one and can thus be integrated with any DV routing protocol. An implementation of the algorithm in Babel, a real world DV protocol, is provided in support of this claim. Then an application of the algorithm is presented: the LC is used to find an optimal tuning for the generation frequency of the Babel control messages. This tuning technique effectively reduces the impact of losses consequent to random node failures in the emulations of several real world wireless mesh networks, without increasing the control overhead.

A second version of the algorithm is designed to be *incrementally deployable*. This version can be deployed gradually in production networks also by uncoordinated administrators. When only a fraction of nodes is upgraded so to participate in the protocol, these upgraded nodes estimate their LC indexes approximating the theoretical ones. The approximation error is studied analytically and it is also shown that, even for low penetration ratios of upgraded nodes in the network, the algorithm accurately ranks nodes according to their theoretical centrality.

The second contribution of the thesis is the critical discussion of the integration of blockchain technologies in distributed networks. An initial analysis of the literature concerning blockchain based applications reveals an ambiguity around the term “blockchain” itself. The term is used, apparently, to identify a number of similar but different technologies proposed to empower a surprisingly broad range of applications. This thesis prompts there-

fore the need of formulating a restrictive definition for the term *blockchain*, necessary for clarifying the role of the same blockchain in distributed networks.

The proposed definition is grounded in the critical analysis of the blockchain from a distributed systems perspective: Blockchains are only those platforms that implement an open, verifiable and immutable Shared Ledger, independent of any trusted authority. Observing that the blockchain security grows with the amount of resources consumed to generate blocks, this thesis concludes that a secure blockchain is necessarily resource hungry, therefore, its integration in the constrained domain of distributed networks is not advised.

The thesis draws recommendations for a use of the blockchain not in contrast with the definition. For example, it warns about applications that require data to be kept confidential or users to be registered, because the blockchain naturally supports the openness and transparency of data together with the anonymity of users.

Finally a feasible role for the blockchain in the Internet of Things (IoT) is outlined: while most of the IoT transactions will be local and *Off-Chain*, a blockchain can still act as an external and decentralized platform supporting global transactions, offering an alternative to traditional banking services.

The enhanced reliability of DV routing protocols encourages a wider adoption of distributed networks, moreover, the distributed algorithm for the computation of centrality enables applications previously restricted to centralized networks also in distributed ones. The discussion about the blockchain increases instead the awareness about the limits and the scope of this technology, inspiring engineers and practitioners in the development of more secure applications for distributed networks. This discussion highlights, for instance, the important role of the networking protocols and communication infrastructure on the blockchain security, pointing out that large delays in the dissemination of blocks of transactions make the blockchain more vulnerable to attacks. Furthermore, it is observed that a high ability to take control over the communications in the network favors eclipse attacks and makes more profitable the so called *selfish mining* strategy, which is detrimental to the decentralization and the security of blockchains.

The two main contributions of this thesis blended together inspire the exploitation of centrality to optimize gossip protocols, minimizing block propagation delays and thus the exposure of the blockchain to attacks. Furthermore, the notion of centrality may be used by the community of miners to measure the nodes influence over the communication of blocks, so it might be used as a security index to warn against selfish mining and eclipse attack.

Acknowledgments

I would like to thank first my supervisors, prof. Renato Lo Cigno and prof. Leonardo Maccari, for their invaluable advice, continuous support and patience during my PhD study. Since this is a PhD thesis in computer science I hope this funny comparison could lead to a smile: Renato and Leonardo have been for me like the very best *servers*, I just had to knock on their *port* to find them always *available and ready to listen to me*. With Renato I always had a special relationship, characterized by his great equilibrium and contagious curiosity that always inspired and motivated me so much to literally make me *run faster*! Leonardo always inspired me as well especially with his noble attention towards making research for the good of people, an attention daily witnessed by his funny *Community Network T-shirts*. Thank you *grandi capi* for our passionate discussions which made me appreciate your good heart and immense wisdom in guiding me, making these years spent working together so special.

I would like to express my gratitude also to Michele Segata and Luca Baldesi, my *older brothers* in our beloved Advanced Networking Systems Group, to whom I could always ask help being sure to receive back a good advice and an encouragement.

I would like to add another special thanks to prof. Alberto Montresor, Christian Consonni and Alessio Guerrieri. Preparing lab exercises and fabulous assignments on Algorithms and Data Structures with you became surprisingly funny for the joy of hundreds of students! Thank you all for the constant help and assistance you gave me while I was moving my first steps as teaching assistant.

I will never forget the exciting research period spent abroad in Boston, so I am deeply grateful to prof. Stefano Basagni for having been the best tutor and *host* I could ever wish to have. Writing our paper about the blockchain mixing ancient Latin with medieval paintings has been a pleasure shared also with Francesco Restuccia, Salvatore D'Oro and prof. Tommaso Melodia, thank you all for your precious support.

I would like to thank all PhD school mates and colleagues in both the University of Trento and the Northeastern University for having shared with me the hard and the good times of the daily academic life.

My final heartfelt thanks goes to Takhmina, my family and friends for their encouragement and support all through my studies.

Contents

Chapter 1

Introduction

A reliable and performing network infrastructure is considered today not only a need but a civil right, as it is crucial for fundamental activities such as reading news, smart working, purchasing or selling goods, and also for learning online. The network infrastructure is thus strategic for the advancement of society as it empowers our democracies, education plans and economies. Especially in these recent times unfortunately populated by pandemics, fake-news and by the scandals from the big players of Internet, we learned the importance of computer networks: we want them to be more reliable, to support all our daily businesses, and more secure to protect our identities and our data across the web. But the challenges to make our networking infrastructures more reliable and secure are many. The today dominating networking paradigm based on private, for-profit Internet Service Providers (ISPs), which is almost the only option for billions of users to access the Internet, hides in fact several threats.

An ISP is a company that offers to residential and business users services related to the Internet, the main ones being access to the World Wide Web and e-mail. The users fees, together with the public funding provided by national agencies for the development of broadband connectivity, constitute the principal revenues for the ISPs, while the major costs comes from the CAPEX and OPEX[1] necessary to install and maintain the networking infrastructure. The just depicted networking paradigm, with users that pay their ISPs and completely rely on them to access the network, exposes citizens to a number of risks:

- *Privacy:* All the data of a user are transmitted through the network of its ISP. This ISP can identify the information flows belonging to the user, track its behaviors, inspect and even resell the user data.

[1] CAPEX: “Capital expenditures”, comprise major purchases that will be used in the future.
OPEX: “Operating expenditures”, represent day-to-day costs that are necessary to keep a business running.

- *Neutrality:* The ISP can restrict the access of the user to portions of the Internet (DNS blocking[2]), redirect users only to commercial partners while boycotting others, and can even censor users contents, such as private messages or commercial advertisements.

- *Digital Divide:* ISPs are for-profit companies, thus they focus their investments on the networking infrastructure in those areas with an higher expected return of investments. These are densely populated, industrialized areas, while rural territories are usually left "unconnected", deepening the digital divide problem of poor communities [1].

- *Efficiency:* Many networking applications requires the exchange of traffic between users that are geographically close. Some examples are the emails exchanged between the employees that work in the same department, the downloads of news from local newspapers, up to the exchange of information that smart-cars may perform to implement coordinated maneuvers.[3] For all these applications a local handling of the user traffic would be more efficient than sending information flows through the remote cloud-services of an ISP.

The aforementioned issues are intrinsic for the centralized networking paradigm dominating nowadays.

Characteristics of Distributed Networks

Distributed networks offer an alternative networking paradigm. The most popular and widely studied distributed networks are Wireless Mesh Networks (WMNs) [3, 4] and Ad hoc networks [5–7], both characterized by multi-hop communications among devices composing a network that is unplanned, dynamic and self-healing. A further characteristic of distributed networks is their typical mesh topology, as shown in Figure 1.1.

Ad hoc networks are small-scale local networks made of a few (tens) portable devices which may also exhibit some degree of mobility. As of today there are more than 3.5 billions of smartphones around the world, all equipped with Wi-Fi chips so that, in principle, it should be easy to configure an ad hoc network to share contents among a little group of friends, or whenever needed. However, in most of the smartphones the ad hoc networking feature is disabled, even if their Operating Systems (OSs) (Android or iOS are the most popular) would support it. It may be speculated that the ad hoc networking feature is disabled because it is a *threat to telcos*. In principle, in fact, one can easily communicate with nearby users, for free, just relying on Wi-Fi at the unlicensed ISM band of 2.4 GHz. A user allowed to do so would become less willing to pay the full price of a subscription to a cellular carrier, legitimately

[2] DNS blocking or filtering, is a strategy to filter out some domain names from the Internet Registries to make them unreachable. More on: `https://en.wikipedia.org/wiki/DNS_blocking`

[3] See for example *platooning*, i.e., a techinque to drive a group of vehicles together. More on `https://en.wikipedia.org/wiki/Platoon_(automobile)`

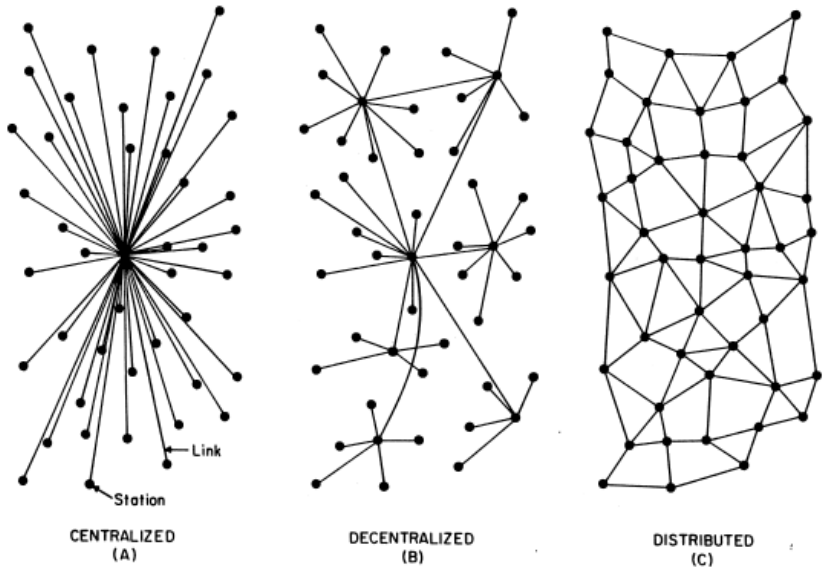


Figure 1.1 – Baran [2] introduced the concept of "Distributed Network" in 1964 illustrating the differences with centralized and decentralized networks. *Copyright © 1964, IEEE.*

asking to cut out those costs related to local communication services not performed anymore on the licensed bands of the carrier. Anyhow, there is no technical reason for restraining ad hoc networking, so it can be imagined that ad hoc networks will become popular and available to users as soon as business policies will change.

Wireless Mesh Networks (WMNs) are instead static and larger networks made of wireless devices, they can cover areas that range from a building to a whole district, or even a city. Mesh networks do not exhibit any hierarchical structure, this also means that there are no key strategic nodes that, if attacked, will bring down the whole network. The lack of critical point of failures makes mesh networks to be considered as the most robust networks, which therefore find different successful applications in the industrial and military sectors, and they are also popular to extend the WiFi coverage at home. Nonetheless, the mesh network design finds its most remarkable application in Community Networks (CNs).

Community Networks (CNs) are mesh networks built spontaneously by a community of people mainly to share the access to Internet, but also to own and manage the network infrastructure collectively, without relying on an ISP. The growth of CNs in the last years has been remarkable [8] with different research disciplines attracted by the peculiar socio-technical features of CNs, including their potential to de deployed with little costs in rural areas affected by digital divide [9]. However, the lack of a sustainable and scalable organizational model for CNs puts their existence in continuous peril [10, 11], as much as for ad hoc networks.

1.1 From Centralized to Distributed Networks

The problems inherent in centralized models of networking motivate the search of new *decentralized* paradigms able to blend together reliability, security and sustainability. The word decentralization is used to indicate the process of transferring the control of the network from central, authoritative entities to more peripheral ones, distributing management and decision making activities over the whole network. Accomplishing this process would free users from the discriminatory power of the private, for-profit ISPs, that are ultimately the root problem for both the insecurity and the unsustainability of the today evolution of networks. Distributed networks can play a role of paramount importance in this process. They can represent, for instance, the answer to the users need of having a distributed and direct control over the network infrastructure and over their own data, shared on the same network.

The paradigmatic shift from central to distributed management of networks described so far requires not only new technologies for making networks economically sustainable and independent from private, business operators, but further requires new adaptations of the traditional network-control mechanisms that should be tailored for distributed networks, for the ultimate goal of enhancing their reliability. In this regard, automatic failure recovery is a key process which needs to be re-engineered as an online and distributed function, so to address the peculiar structural properties of large and decentralized networks.

The implementation of an open, transparent and decentralized platforms able to safeguard user rights and protect the privacy of data is another relevant research direction, with the goal of improving the current technologies that empower distributed networks. The question of whether the blockchain can be successfully integrated into distributed networks to implement a technically feasible and socio-economically sustainable governance model, as originally proposed and explored in [12], is particularly central for this thesis. The blockchain is always described, in fact, as a decentralized technology for the support of economical transactions which in principle can empower an efficient market suitable for distributed networks.

The enhancement of distributed networks has been the goal that inspired my research of novel solutions to achieve greater reliability and security. In this thesis I present two main contributions:

1. A new failure recovery technique for more reliable distributed networks, further introduced in Section 1.2;

2. A critical analysis about the integration of the blockchain technology inside the network infrastructure as a measure to enhance security (Section 1.3).

1.2 Network Reliability: Failure Recovery Optimization

Failure recovery is a classic networking problem, crucial for the reliability of any network. The reliability of communication services depends, in fact, on the ability of the network to automatically detect failures and instantiate self-repair mechanisms. This means minimizing the interruption of all networking services consequent to a potential failure and, ultimately, enhance the network reliability.

An initial strategy to enhance the network's reliability could be the deployment of a powerful failure detecting systems, able to timely catch any anomaly so to trigger repair mechanisms as soon as possible. To this purpose, the network must be frequently controlled: *control messages* are periodically exchanged by all the components of a network precisely to verify, continuously over time, that all devices are working as expected. These control messages tend to be called with curious names such as *heartbeat* or *keep-alive*. These names reflect the fact that possible network failures, such as the abrupt shut-down of a node because of a loss of power or the removal of a link because a cable has been cut during road-works, appear to the network as the sudden death of one of its component. Increasing the frequency of such control messages, to faster detect these failures, would be a naive attempt for upgrading the failure-discovery process as it would lead, unfortunately, to a greater occupation of the network bandwidth, hence to a reduction of the overall performance of the network. On this observation is based one of the fundamental trade-offs of networking: the performance/control trade-off.

The simple selection of default values for those timers that regulate the generation frequency of control messages has been, for many years, the only and basic approach adopted by network administrators to balance control and performance. These timers had to be short enough to catch anomalies in reasonable time, but not so short to generate a constant storm of control messages. In general, for all popular networking protocols, such timer-values were chosen empirically many years ago and then standardized by publishing them in the official documentations of such protocols. Most of today's networks are still configured with these decades old timer-values, installed globally in all the network devices. This simple approach makes the network configuration uniform and easy, also leading to generally acceptable performance. However, advanced strategies for a fine-grained tuning of the frequency of control messages can improve the naive approach based on default timer-values deployed globally.

The first contribution of this thesis is the porting of an optimal strategy for the tuning of control messages from centralized to decentralized networks. This strategy originally called *Pop-Routing* [13, 14] introduced the pioneering idea of exploiting centrality metrics, a

graph-theoretical tool, to tune the generation frequency of control messages on a per-node basis, according to the node centrality with respect to the rest of the network.

Pop-Routing leverages on the observation that the most central nodes of a network play a strategical role and, compared to peripheral nodes, have a larger influence on the overall performance of the network. Consider for example Figure 1.2, where nodes are colored according to their Betweenness Centrality (BC) index, highlighting the strategic importance of the most central (yellowish) nodes with the respect to the darker ones. A failure of a yellow node in the core of the network would result in a greater disruption of the communications insisting over the network, while a fault in a peripheral sector would limit the damages mostly to that sector only. Starting from this observation, with Pop-Routing the generation frequency of control messages for central nodes is increased, and is instead decreased for peripheral nodes. The overall generation rate of control messages in the network is kept constant, still, it has been shown that Pop-Routing reduces the average duration and impact of network failures. This tuning strategy is called Pop-Routing since it resembles traditional equalizers for Pop-music, that typically enhance central audio frequencies while diminish lateral (low/high) ones.

The knowledge of the nodes centrality in the network is necessary to enable the Pop-Routing strategy and, in its original formulation [13], Pop-Routing was based on the notion of the *Betweenness Centrality (BC)*. The state-of-the-art algorithm for the computation of BC is the Brandes'algorithm [15], which is a centralized algorithm that requires the full knowledge of the network topology (in form of a graph), to be executed correctly. The requirement of owning a global graph knowledge limited the application of Pop-Routing to the networks configured with centralized routing protocols,[4] the only kind of protocols that collects this complete information in all the routers of the network. Distributed routing protocols were therefore excluded from the benefits of the Pop-Routing strategy, unless a new way for computing centrality could be developed on top of an only partial and local topological information.

A first contribution of this thesis is the development of such distributed algorithm that enables the computation of the Load Centrality (LC) (a centrality metric similar to the original BC) exploiting only the information contained in routing tables, which is an information locally maintained and updated by all the nodes of any network. Such distributed algorithm is a minimal extension of the Bellman-Ford algorithm —which is at the heart of all distributed routing protocols— thus it is general and simple enough to be easily embedded in all distributed networks. This thesis also reports a concrete example of how implementing the distributed computation of LC in a real-world routing protocol, namely *Babel* [16], the tuning

[4] Routing protocols are responsible for the identification of the best paths to be used to route (ergo transmit) information efficiently in a computer network.

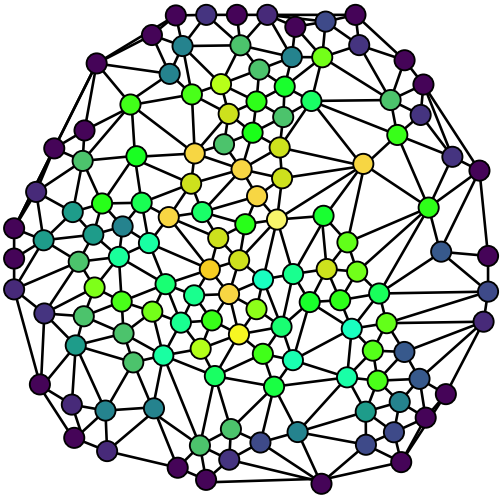


Figure 1.2 – An example of graph with the nodes colored as a function of their Betweenness Centrality (BC).

of the Babel's control messages according to the Pop-Routing methodology led to a reduction of the impact of failures in the network, effectively enhancing the network reliability [17, 18].

New proposals for the advancement of routing protocols such as the introduction of a distributed version of Pop-Routing have the tendency to be accepted with enthusiasm only in small networks under the administration of a single central administrator, while it is harder to convince different administrators of larger networks to do the same. The reason behind this common phenomenon is that the introduction of a protocol enhancement necessarily means upgrading the software run by all routers. In the worst case this upgrade must be executed at the same time on all routers, as not always the updated software is backward-compatible with the legacy one already deployed and running in the network, therefore an incomplete deployment of the upgrade would lead to continuous communication failures between upgraded and legacy routers. In practice the entire network must be temporarily shut-down to reboot all routers with the new updated software, interrupting all networking services for the time necessary for the upgrade routine. For a single administrator that control a small network it is easier to execute this maintenance procedure minimizing the service interruption. It is enough to determine a period of time in which the network is not significantly used (usually during the night) and then perform a coordinated upgrade of the few routers that compose the network. Finding an agreement among different administrators on a single day in which many more routers should go under a synchronized maintenance procedure is instead much harder. This coordination problem introduces the need of making the new software *incrementally deployable*: i.e., it should be backward-compatible and it should be possible to gradually update smaller portions of the network without causing malfunctions due to software incompatibilities. Moreover, the benefits brought by the

upgrade should become immediately effective, even if only a small subset of routers has been updated.

A second contribution presented in this thesis is the development of a more sophisticated version of the original algorithm for LC that is incrementally deployable [18]. With the new incrementally deployable version of the algorithm it is possible to show that, even at low penetration ratio, the upgraded nodes are able to accurately rank nodes according to their centrality. This result enhances the flexibility of the Pop-Routing methodology, and constitute a step forward toward its wider adoption in distributed networks.

1.3 Network Security: Blockchain to secure Distributed Networks

In our daily use of Internet we continuously need to trust many private companies for the collection and management of our data. For example, when we register an account on a social network we provide sensitive data like our name, surname and date of birth to the company that own the social network. We expect these data to be used only for legit purposes such as friendly display our name to let our friends recognize us in the network, but not to post fake-news on our behalf or to use our account to spread spam messages. Essentially, everyday the data of many users end up in big data-bases under the operational and exclusive control of private companies: we need to trust these companies, hoping that no unlawful use of our identity will be done within the closed walls of their data-centers. Not only our digital identity is in the hands of private companies, but also our money. For instance, when we purchase some product online, we need to trust the banking institution that issued our credit/debit card for correctly transferring the proper amount of money from our account to the one of the online retailer. Again, in this process we implicitly need to trust the payment circuit selected to handle our transaction, and many users probably do not know exactly which security measures are put in place to safeguard the online transfers of money. If, by any chance, our transaction will be handled on the popular Visa payment network, as users we may be surprised to learn that our transaction is just one among the hundreds of billions of transactions yearly processed by Visa, and all these transactions that happens worldwide are processed actually in just 4 big data-centers,[5] protected by the most advanced military technologies.[6] However, despite the strong military defense, it also means

[5] The map of the Visanet data-centers available at `https://baxtel.com/data-centers/visanet`

[6] The security measures include hydraulic bollards that can stop a vehicle traveling up to 50 mph, beyond a traditional biometric scan to be admitted. More on `https://www.networkcomputing.com/networking/inside-visas-data-center`

that it is enough to break into only few data-centers, for example with a software virus or corrupting an employee, to potentially steal the money of billions of persons.

Private companies centralize in their own networking infrastructure most of our data, and we hold them responsible for the safekeeping and for the legit use of them. We trust these private companies because we believe that most of the time such companies are really trustworthy and that, after all, violations are rare and our rights could be defended in the courts if necessary. However, unfortunately, Internet scandals that involve the big tech giants in massive violations of users data are daily on the news, with public institutions and authorities not able to take an action against these violations, establishing a disconcerting picture of the dominant paradigm constituted by private, opaque service providers. The people awareness about this concerning model motivates the research for new paradigms where users can always transparently verify where their data are stored and who can access them.

The blockchain is a novel technology that came into the limelight with the advent of cryptocurrencies, promising to be an open, transparent, secure and decentralized solution for the management of data. It came to the forefront as constitutional technology of the Bitcoin, the most popular cryptocurrency that starting from March 2021 exceeds the trillion of US dollars in market capitalization.[7] The features of the blockchain, i.e., its decentralization together with the ability to tolerate extremely powerful cyberattacks, attracted the interest of many research communities, including the networking one that was hoping to find in the blockchain a solution to securely decentralize classic networking functions. The hype around the blockchain led to an extremely large number of disparate proposals of using the blockchain for many applications comprising supply chain management [19–22], e-voting [23–30], the Smart Grid [31–38], healthcare [39–42], banking [43,44], smart cities [45–47] and vehicular and aerial networks [48–66]. Many surveys collect together the various efforts of applying the blockchain to different sectors, including to the networking domain [67–75]. This vast application range leads to think to the blockchain as to an almost *universal* technology, not limited anymore to cryptocurrencies but able to empower new networking applications as well.

This apparent universality of the blockchain may be surprising. The original Bitcoin blockchain, in fact, became well known not only for its high security and decentralization but also for the extreme energetic costs needed to run it together with the high latency of Bitcoin transactions, which lead to very poor performance in terms of Transactions per Second (TPS): less than 10 TPS on average [76]. Such inefficiencies are hardly tolerable by any networking applications.

[7] Statistics about the market capitalization of Bitcoin are available at `https://www.statista.com/statistics/377382/bitcoin-market-capitalization`

A deeper inspection of applications different from cryptocurrencies reveals how the characteristics of the original Bitcoin blockchain have been diluted, if not completely transformed, when moving to new application domains. For example in the original and iconic cryptocurrencies (e.g., Bitcoin and Ethereum) the blockchain plays the role of a shared ledger that transparently record all transactions to enable a distributed validation of them without involving trusted intermediaries, so much that the resulting ledger is open to anonymous users. For this reason the original blockchains are said to be *permissionless*, i.e., users do not need to be first registered and then authorized by any authority to start performing transactions trough the blockchain. This also means that an extremely powerful cryptographic security measure becomes necessary to protect the blockchain from cyber-attacks. For example, the same Bitcoin and Ethereum adopt an extremely secure but power-hungry mechanism called Proof of Work (PoW), wich is also the root cause for their well known poor performance.

On the contrary, newly proposed blockchains are completely different so much to be called *permissioned*. In permissioned blockchains the resulting ledger is not open to anyone, the access is rather controlled by a discriminatory authority (or consortium) that grant the permission to perform transactions only to registered users. This coming back to a centralized paradigm is not without consequences: doubts about the security of permissioned blockchains are legit as much for centralized platforms controlled by private companies since, after all, their degree of centralization is the same.

The doubts raised so far about the true efficiency and security of the blockchain are the ground for a further contribution of this thesis, i.e., an examination of the recent literature related to the blockchain technologies and applications, which culminates in a novel, critical analysis about the integration of the same blockchain in the networking domain. The main thesis conveyed by the critical analysis is the following. The blockchain is not a universal technology, rather, the blockchain is characterized by a set of specific features —precisely identified in the analysis— advantageous for a limited number of applications, above all for cryptocurrencies. For the greatest majority of the different applications proposed in the last years, including those belonging to the networking domain, the blockchain results instead to be inappropriate and poor-performing.

1.4 Thesis Organization

The first contribution concerning the optimization of distributed routing protocols for the achievement of a greater reliability is discussed in the rest of the thesis as follows.

- Chapter 2 provides the fundamental notions about distributed routing protocols and centrality metrics;

- Chapter 3 presents the distributed algorithm for the computation of Load Centrality (LC) and the results of its application to improve the resilience properties of distributed routing protocols.

Then the thesis moves to the critical analysis about the integration of the blockchain in distributed networks. This analysis is provided as the base for the formulation of a restrictive definition of the term "blockchain" [77, 78]. This definition is necessary to serve one main purpose of this thesis, namely, clarifying the possible role of the blockchain in the context of distributed networks. Albeit pivotal for this thesis, the author acknowledges a limit of the proposed blockchain definition, i.e., its partiality, since the arguments supporting the proposed definition are grounded in a selection of recent works that does not cover systematically the whole literature.

Despite the declared partiality the critical analysis offered by this thesis blends together technical and historical arguments that evolve around the transverse *Distributed Consensus Problem*, a key problem which lies at the origin of the blockchain technology. The critique to the blockchain technology is therefore structured as follows:

- The fundamentals on the blockchain and on the Distributed Consensus Problem are covered in Chapter 4. A brief survey on consensus protocols is included in Section 4.4, preceded by a discussion about the fundamental theorems that highlight the inescapable trade-offs for any such protocol (Section 4.3);

- Chapter 5 conveys the critique to the integration of the blockchain in the domain of distributed networks.

Chapter 2

Distributed Routing Protocols and Centrality

A computer network is usually modeled by a graph $G(\mathcal{V}, \mathcal{E})$ where:

- $\mathcal{V}$ is the set of vertexes (or nodes) of the graph, and models the devices that compose the network such as routers;
- $\mathcal{E}$ is the set of edges (or arches) of the graph, and models the links that interconnect all devices.

A routing protocol identify the best-paths in the network graph to be used for routing traffic from any source to any destination. At this stage the formal definition of *best* paths is not necessary but these could be, for example, the fastest, the shortest or those with greater capacity. Identifying such paths is of the utmost importance for the overall performance of the network, since all the messages exchanged in the network will flow through these paths.

The identification of the best-paths can be seen from a purely operational perspective as the process of updating the *Routing Tables*, i.e., those tables installed in each router that match reachable destinations with the address of the node towards which the traffic should be forwarded, called *next-hop*. In fact, when a router chooses a new next-hop for a given destination d it means it has learnt a new best-path towards d, and thereafter it will use the new next-hop (updated in the routing table) to forward the traffic destined to d. Once a router has learnt the best-paths, then the forwarding process becomes as simple as this:

- Received packets of information are inspected looking for the destination tag;
- A lookup in the routing table reveals the address of the favorite next-hop;
- The router forward the packet to that next-hop.

This way packets are progressively sent along the best route up to the final destination. Summing up, a router runs a routing protocol to identify best-paths and the selection of a best-path towards a given destination is reflected by the update, in the routing table, of the preferred next-hop for that destination.

The challenges for routing protocols

Although apparently simple, keeping routing tables updated hides different challenges that routing protocols have to address. First of all routing protocols must be aware of the network status to identify the best-paths, however, the network is dynamic and evolves over time. Secondly, given a network state represented by a graph, the pure algorithmic problem of searching the best paths in the graph arises as well. For this latter problem the space left for further research towards a more efficient and optimized search of best-paths in a graph is little. Shortest-paths algorithms are in fact well-known in the literature so that the Dijkstra and the Bellman-Ford algorithms already efficiently solve the best-path search problem and are always used, respectively, in centralized (Link-State) and in distributed (Distance/Path-Vector) protocols.

Network Dynamism

The challenges deriving from the network dynamism open instead a greater space for research. The network dynamism is evident in circumstances of failures such as the abrupt shut-down of a router or the drop of a communication link that may happen, for example, because of a loss of power or of the cut of a cable during roadworks. These kind of events modify the topology of the network and to take them into account the failed component should be removed from one of the two sets of nodes or edges. Also the installation of a new router or link results trivially in a new node or edge to be added to the network graph. Beyond simple additions/removals, the quality alterations of the various components are also an important source of dynamism and are relevant for the selection of the best-paths as well. In other terms the link-quality is a dynamic property fundamental for the formal definition of best-paths. Possible link quality metrics could be, e.g., the latency or the capacity, and accordingly a path could be defined to be the best one for connecting a pair of end-points if the sum of the latencies of the links that compose this path is the minimum, or if the sum of the link capacities is the maximum. Beside the problem of formally defining the link-quality and path metrics, it is important to notice that such metrics can change over time for many reasons such as channel interference or network congestion.

Control messages

Routing protocols cannot ignore the network dynamism: it turns out that all routing protocols dictate to exchange periodic messages to constantly monitor the network status, thus to discover new neighbors, detect failures and to measure link-quality metrics. These periodic messages called *control messages* collect the necessary information to steer the routing process, which is instead responsible for processing this information so to identify the best-paths. Two fundamental processes have been described: i) the periodic exchange of messages to monitor the network status and ii) the routing process. The way these processes are implemented define two broad classes of routing protocols, namely Link-State (LS) and Distance-Vector (DV) protocols.

Link-State (LS) protocols in a nutshell

The LS protocols are characterized by the adoption of the Dijkstra algorithm for finding the shortest paths in the network graph. To run the Djikstra algorithm a node of the graph must be chosen as "source", then an efficient exploration of links starting from the source node produces a shortest-path tree that covers all other nodes reaching them following only shortest (best) paths. A router that owns the full graph knowledge can run the Dijkstra algorithm precisely to identify all shortest-paths from itself (set as source node) toward any destination. This way it also identifies the next-hops to be recorded in its routing table. LS protocols are said to be centralized since the routers need to accumulate the full graph knowledge to run Dijkstra. To this purpose whenever the network topology changes (a failure occurred or the quality of a link changed) then a *Link-State-Advertisement (LSA)*, which is a message that reports the new topological information, must be broadcast in the whole network to let all routers update their graph first, then run Dijkstra again, and finally update their routing table if new best-paths have emerged. LS protocols are more often deployed in small networks under a single and centralized administration as long as the broadcast of any topological change can become too frequent and inefficient over large networks.

DV routing protocols are instead distributed: more details on the distributed routing paradigms are covered in Sections 2.1 and 2.2

2.1 Distance-Vector Protocols

Routing protocols can be divided according to the need of routers to know or not the complete network state in form of a graph. On one hand there are LS protocols, where all routers need to acquire such full graph knowledge thanks to LSA messages enabling this way the Dijkstra algorithm. On the other hand there are DV routing protocols, where routers do

not need the same complete graph knowledge to discover all the reachable destinations in the network. LS protocols are also said to be "centralized" because with them routers need to accumulate (in other terms "centralize") also the topological information that comes from remote nodes, e.g., the state of all network links. DV protocols are instead said to be "distributed", as nodes just need to exchange information with their direct neighbors. This section focuses on distributed routing protocols, especially on DV ones.

To enable DV routing routers must only perform two actions:

1. Measure the distance from their neighbors;
2. Keep sharing with neighbors the information they learned about reachable destinations and the distance from them;

Neighbor discovery and distance

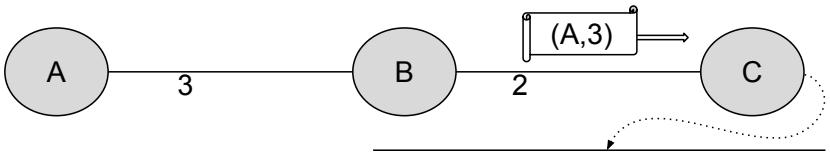
A DV router periodically broadcast HELLO messages to discover neighbors that, if present, reply with another HELLO. The reception of a reply to an HELLO means that a bidirectional link between the two neighbors has been established. The subsequent frequent exchange of HELLO messages on the link is useful to estimate the quality of this link. The link-quality can be expressed in terms of many metrics like, for example, the link-delay or the probability of successfully sending a packet over the channel. In general, especially in wireless networks, the quality of a link degrades with the growth of the distance separating two neighbors: it turns out the most popular link-quality metrics take low values for good quality links and high values for bad ones, thus they may be thought of as values of *distance*. The link-quality metrics are also called link *costs* or, in a more graph-theoretic perspective, edge *weights*.

Update messages to send vectors of distances

DV routers keep updated a vector with distance values associated to all known destination in the network. This *distance-vector* gives the name to the DV family of routing protocols.

At boot a node does not know any destination except himself, so it initializes his distance vector only with a 0 to indicate the "zero distance" of this node from itself.[8] Right after booting a node starts exchanging HELLO messages, this ways it discovers its neighbors and also measures the quality of the link (distance) with them. The node can hence extends its distance vector with the just learned distances from its neighbors. A node does not send only HELLO messages, but also UPDATE (a.k.a. "advertisement" or "announcement") messages. An UPDATE message is a pair of value (d, m) that a node v periodically broadcast to any neighbors u to inform u that v is at distance m from destination d. Figure 2.1 illustrates an

[8] Configuring a node some administrators may choose a value of distance different from zero for the destinations exported by this node to implement advanced routing strategies.



Destination	Distance	Next-Hop
A	2 + 3 = 5	B
B	2	B
C	0	//

Figure 2.1 – An illustration of an UPDATE message in a DV protocol and of the routing table of a node, C in the present case. In this example the nodes A and B are separated by a link of quality equal to 3, while C is at distance 2 from B. B already learned the existence of A thanks to an UPDATE message previously sent from A to B. Consequently B can send an UPDATE over the link of length 2 to inform C that A is reachable via B itself, thanks to a route whose length is equal to 3. C has no other shorter routes available for reaching A, so C will install in its routing a table a route towards A whose length is equal to $2+3=5$ choosing B as next-hop.

update sent from node B to C in order to advertise the availability of a route for reaching destination A. The illustration shows also the routing table of node C: the destination, the distance and the next-hop are recorded in this table for each route available to node C.

Routing Table notation

The following notation will be used in the rest of this thesis for operations with the Routing Table (RT). Let v be the node that manipulate its routing table, its routing table RT will be considered as a dictionary mapping destinations d to next-hops and distances expressed in some given metric m. The square brackets operator is used to retrieve the information recorded in the RT associated to a particular destination d, e.g., $RT[d] = (m, NH)$. The tuple (m, NH) reports the distance of v from d as m, while NH is the next-hop chosen by v to forward traffic towards destination d. The dot notation will be used to access single fields of a tuple, for example: $RT[d].NH$. The '()' will be the dictionary operator to be invoked to get the list of the dictionary keys. For example, it can be used to get the set of known destinations recorded in the routing table as in $RT() = [d_0, d_1 \dots d_N]$. The '[]' and '{}' symbols refer to the empty list and dictionary, respectively. The NH symbol does not indicate a necessarily single next-hop, but can represents a list of them, therefore, NH can capture routing protocols that implement multipath routing.

The Bellman-Ford criterion

Let s be a neighboring node that is sending an UPDATE, and let $c(s)$ be the cost of the link with s. On reception of an advertisement (d, m) sent from s the receiving node processes the advertisement and updates its routing table according to the Bellman-Ford criterion:

$$\begin{aligned} &\text{if } c(s) + m < RT[d].m \\ &\Rightarrow RT[d] = (c(s) + m,\ s) \end{aligned}$$

Essentially, a node that discovers a shorter path for some destination updates its distance from that destination and records the neighbor that announced the shortest path as favorite next-hop. Paths identified according to the Bellman-Ford criterion are minimal (shortest) by induction. The $<$ operator ensures, in fact, that paths are built as minimal combination of subpaths that are minimum by hypothesis provided that routers advertise only minimum routes. The Bellman-Ford criterion also ensures that shortest-paths are loop-free, i.e., packets going from any source to any destination will not traverse the same node twice. Loop-freedom is a fundamental property for routing because, if a packet were to enter in a loop, it would remain stuck there without being ever delivered successfully.

The Limits of DV routing

A router sending an UPDATE to notify a topological change triggers a recursive message-passing process. During this process the sent UPDATE may need to traverse the whole network, so the worst case convergence time for DV routing protocols is linear w.r.t. the network diameter.[9] If routers send UPDATES periodically instead of sending such messages as soon as they acquire new routing information, then the routing phase may take a large amount of time to converge, especially in large-scale networks. Moreover, DV protocols empower only simple minimum-distance routing strategies, while network administrators often need more advanced routing policies to implement their business strategies.

Advanced routing strategies require flexible and expressive policies to evaluate and rank candidate paths for communication. Path-Vector (PV) protocols, described in Section 2.2, support these kind of policies empowering, for example, the advanced routing strategies of the Border Gateway Protocol (BGP), the routing protocol of Internet.

[9] The diameter of a graph (or network) is the maximum distance between any pair of vertices.

2.2 Path-Vector Protocols

The Border Gateway Protocol (BGP) [79] is de facto the only routing protocol of the Internet. The Internet is essentially the network of all Autonomous Systems (ASes), where an AS is a collection of routers under a single organization's control identified by a unique Autonomous System Number (ASN). In a network diagram like the one shown in Figure 2.2 ASes are represented by clouds, and the depicted network devices residing at the margins of ASes are called *border routers*. The company or institution that governs a given AS can choose BGP

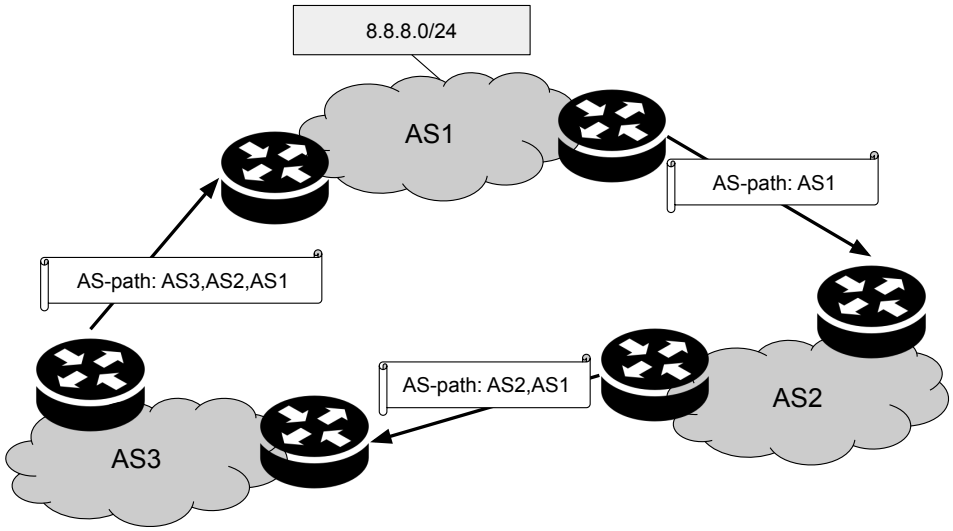


Figure 2.2 – An illustrative example of the Internet as a network connecting different Autonomous Systems (ASs). Each cloud represent an AS, which is a network itself potentially made of many internal routers. At the border of each AS a BGP router must be installed to establish a link with neighbouring ASs.

as protocol to route internal traffic. When used for this purpose BGP is called *internal*-BGP (iBGP). With iBGP all routers must be connected to each other forming a full mesh and to alleviate scalability problems Route Reflectors [80] are commonly deployed. Anyhow, the main purpose of BGP is to establish relationships between border routers that belong to different ASes. In this context the protocols is called *external*-BGP (eBGP). The acronym BGP will be used in place of eBGP, since the main duties of the BGP are precisely to establish external relationships between different ASes and to support inter-ASes routing.

Like any other routing protocol BGP defines the rules and format of messages that routers must exchange to learn about reachable networks. For instance, a BGP node advertises known routes to neighbors by sending UPDATE messages. An UPDATE message includes a section to describe the attributes of paths used to route information. Some path attributes are mandatory and must always be included in an UPDATE while others are optional. BGP is known for accepting optional *transitive* attributes, i.e., attributes that may not be recognized by all

BGP routers but, if received, should be forwarded even if not parsed. The most important attribute among the mandatory ones is the *AS-path*, where an AS-path is a sequence of ASNs indicating the list of ASes that a packet will be routed through to reach the advertised network. BGP is said to be a *path-vector* protocol because of the continuous exchange of AS-paths vectors among routers.

Path-selection and policies in BGP

Sharing AS-paths is crucial to ensure loop-freedom. A path that contains two occurrences of the same ASN cannot be selected to route traffic. BGP does not mandate any further strict rule apart from this fundamental one, network administrators are thus free to define local policies for path selection. These policies are kept secret and not shared with neighbors, import and export filters are defined as well to limit the the amount of information shared with other BGP speakers. The flexibility of BGP is fundamental for Internet operators to implement their commercial strategies, for example, to decide to route traffic to partner ASes while avoiding competitors. This same flexibility is also the origin of the notorious complexity of BGP, with operators that defined over time complex (if not bizarre) rules to tweak the protocol in the effort of increasing their revenues.

Hierarchical topology of Internet

The commercial relationships between ASes heavily influence the Internet topology as noted in [81], giving rise to a hierarchic topology that is far from a random graph. In this hierarchic topology BGP nodes can be classified into four categories: T, M, CP and C nodes. "*Tier-1*" (T) nodes are at the top of the hierarchy, they are known to be only few [82] and fully connected in a clique. T nodes establish peering agreements to exchange traffic with each others, while they are connectivity providers for mid-level (M) nodes. The Customers (C) and Content Providers (CP) are at the bottom of the hierarchy with the difference that C nodes always pay their connectivity providers, while CP nodes can also establish peering relationships with other CP or with M nodes.

Nodes and links classification

Nodes (aka ASes, but sometimes this classification holds also for single routers) can be further distinguished into *stub* and *transit*. Stub nodes belong to the C or CP classes and are those nodes that export network prefixes. This means they represent some reachable network and thus they can be sources/destinations of traffic. Conversely transit routers are never senders or recipients of traffic but are part of the infrastructure as connectivity providers. Relationships between ASes can be classified as well. A link is of type *customer-provider*

when a BGP node pays its neighbor for the provided connectivity, otherwise is a *peering* link when the two neighbors agree to exchange traffic with each others for free.

2.3 The Signalling/Performance Trade Off

In the overview of routing protocols reported so far (Sections 2.1 and 2.2) it can be noticed how these protocols generate a considerable amount of control traffic without actually conveying any valuable information. Control messages are, in fact, a technical need only, while computer networks are meant to support the traffic of other contents such as emails, files, pictures or videos. The traffic in a network can thus be divided in two categories:

- *Payload traffic*, which is the aggregation of all those messages/packets that flow in the network carrying the really essential data;
- *Control traffic*, refers instead to those messages like HELLOS and UPDATES of DV protocols that are necessary for the correct functioning of the network but that do not carry any truly valuable data.

The control traffic is also known as *signalling* traffic provided that control messages are substantially signals exchanged by routers to timely notify events relevant for the routing process, e.g., the change of a link cost. The signalling traffic turns out to be at the same time necessary for the network functioning but not properly useful to convey true data across the network. It is thus considered as an inevitable cost to be minimized as much as possible. However, an excessive reduction of control messages can lead to the malfunction of the network while an opposite excessive increase of signals would represent an inefficient use of the network bandwidth. The control/performance trade off described so far is of fundamental importance for the efficient implementation of *failure recovery* techniques: these techniques are adopted in case of failures to timely re-route traffic around the failed network components.

For example, whenever a link between nodes n and m stops working, all the routes insisting over that link fail as well. The traffic sent along these routes will get lost until n and m will detect the failure (because of not received HELLO messages) and then initiate a re-routing routine. In a LS network this re-routing process would consist in the broadcast of LSA messages that will trigger a repetition of the Dijkstra algorithm on the graph $G' = (\mathcal{V}, \mathcal{E} \setminus \{(m,n)\})$, i.e., the network graph once updated with the removal of the (m,n) link. Re-routing should be as fast as possible to minimize the losses of traffic consequent to the failure.

To achieve fast-rerouting one can precompute alternative routing paths and switch to one of them as soon as the currently selected route stops working as expected. However, a

router incurs in an overhead for precomputing alternative routes, moreover, a failure that afflicts the preferred route may compromise also the alternative routing paths, leading the node to a "route starvation" condition [83].

Regardless of whether backup paths are available or not a router should detect failures as fast as possible to timely initiate the recovery process and, similarly, should propagate correcting routing information as quick as possible. The timers that govern the generation of HELLOS and LSA should be very small, however, small timer values also mean an increase of the control overhead and the risk of routing oscillations in case of temporary failures. Once again the signalling/performance trade-off emerges. An optimized fine-tuning of the generation frequency of control messages is therefore of the utmost importance to balance the need of speeding-up the failure recover process, reducing timers, with the equally important need of lowering the control overhead as much as possible by setting large timer values.

2.3.1 The Pop-Routing Tuning Technique

For many years, despite decades of research and experience with routing protocols, there has not been any automatic nor optimal technique to address this optimization problem concerning the generation frequency of control messages. In 2016 such a technique has been proposed with the name of *Pop-Routing* [13, 14] as already introduced in Section 1.2. Pop-Routing formalizes the problem of minimizing the impact of network failures as the problem of optimally tuning the generation frequency of control messages on a per-node basis according to the centrality of the same nodes. The Pop-Routing strategy has been originally limited to centralized (LS) networks since the centrality metric chosen to drive the optimization problem was the Betweenness Centrality (BC), a centrality metric that is computed with the centralized Brandes algorithm [15] and can thus be executed only in centralized networks.

The first major contribution of this thesis is the adaptation of Pop-Routing for distributed networks, which required the development of a new algorithm able to compute centrality in a distributed fashion enabling Pop-Routing also for DV protocols. Before discussing the works related to centrality metrics applied to networking (Section 2.5), the necessary background on graph centrality indexes is introduced in Section 2.4.

2.4 Graph Centrality Indexes

Freeman introduced the Betweenness Centrality (BC) index for the analysis of social networks providing the following definition:

Def. 2.1: ***Betweenness Centrality (BC)*** The normalized *betweenness centrality* of a vertex v is given by:

$$\overline{BC}(v) = \frac{2}{N(N-1)} \sum_{\forall s,d \in \mathcal{V} | s \neq v \neq d} \frac{\sigma_{sd}(v)}{\sigma_{sd}} \tag{2.1}$$

With:

- σ_{sd} as the number of minimum weight paths between vertex s and vertex d;
- $\sigma_{sd}(v)$ as the number of those paths that pass through vertex v.

In other terms, the BC of a node v is the sum of the fractions of shortest paths —excluding those with v as endpoint— that pass through v [84]. The BC captures the prominence of a node in the control of flows over the network. It is thus relevant not only for the analysis of social networks but can be a powerful analysis tool also for studying flows of information in computer networks. The BC centrality of all nodes in a graph can be computed with the well known Brandes algorithm [15], a centralized algorithm perfectly suited to be executed, offline, to study graphs that represent snapshots of a network.

Conversely the Load Centrality (LC) index, an index similar to the BC, lends itself to be computed distributedly and online, as it is suggested by Definition 2.2:

Def. 2.2: ***Load Centrality (LC)*** Assume that all nodes generate and transmit to all other nodes some commodity. Let $\theta_{s,d}$ denotes the quantity of commodity sent by a source node s to a destination node d. Assume the commodity is always routed along minimum weight paths and, in case of multiple (minimum) next hops, this commodity is equally divided among them. Let $\theta_{s,d}(v)$ be the amount of commodity generated by s and destined to d that is routed through node v. The *load centrality* of v is then given by:

$$LC(v) = \sum_{\forall s,d \in \mathcal{V} | s \neq v \neq d} \theta_{s,d}(v) \tag{2.2}$$

The LC metric better captures the intensity of the traffic (i.e., the load) insisting over nodes. For instance, an arbitrary configuration of the $\theta_{s,d}$ parameter can model any matrix of traffic or load distribution. Otherwise, setting $\theta_{s,d} = 1$ for all s, d pairs, the LC can be turned into a pure topological metric fully defined by the graph structure, irrespective of the network traffic. Unless otherwise specified in the rest of the thesis the reader can assume $\forall s, d : \ \theta_{s,d} = 1$, so that the LC will own a pure topological interpretation. Under this

hypothesis and if the graph is undirected,[10] the LC can be normalized as:

$$\overline{LC}(v) = \frac{2}{N(N-1)} \sum_{s,d \in \mathcal{V} | s \neq v \neq d} \theta_{s,d}(v) \tag{2.3}$$

Figure 2.3 illustrates the conceptual differences between LC and BC by reporting two copies of the same graph annotated, respectively, with LC and BC values as computed for a single pair of endpoints (s, d) and assuming that all edges have the same weight.

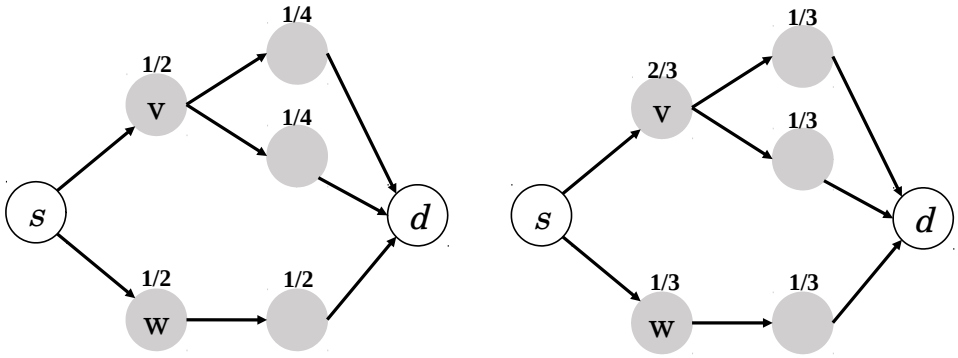


Figure 2.3 – Difference between centrality computation in the case of load (left) and betweenness (right) in the same network for the same (s, d) couple. *Copyright © 2020, IEEE.*

In the left side of the figure it can be noticed how the commodity sent from s to d is cut by two every time it crosses a branching point, in other terms, it is always split equally between equivalent next-hops. According to this mechanism the nodes v and w result to have the same LC index. The BC instead reflects the fact that, out of the the 3 minimum paths that goes from s to d (i.e., all those of length 3), 2 go through v while only 1 crosses w. This happens because v has two alternative next-hops to reach d while w has just one. Therefore, being the BC a measure of the fraction of minimum weight paths that cross a node, then $BC(v) = 2/3$ while $BC(w) = 1/3$, which makes v more central than w.

2.5 Centrality in Networks: Related Works

The Betweenness Centrality (BC) has been originally proposed by Freeman for the study of social networks [85] but found numerous applications also in networking, for example, in the enhancement of traffic monitoring [86, 87], resource allocation [88], topology control [89] and intrusion detection [90]. The Brandes algorithm [15] is the state-of-the-art, centralized

[10] In a computer network a node establishes a link with its neighbor only if this neighbor is able to reply to HELLO messages, hence only if the link is bidirectional (i.e. "undirected"). For this reason, except for particular cases of wireless networks, undirected graphs are a common choice to model computer networks.

algorithm to compute the BC indexes of all nodes in a network, with a computational complexity of $O(nm+n^2 \log n)$ for weighted graphs with n nodes and m edges. The algorithm of Brandes can be adapted to compute also other centrality indexes based on minimum-weight paths [84]. For example, Dolev et al. proposed a generalization of BC to deal with different routing policies [86]. Also these variants of BC are computed with centralized algorithms which are essentially adaptations of the fundamental algorithm of Brandes.

The problems of introducing centralized algorithms in computer networks

Two are the major problems that hamper the use of centralized algorithms in computer networks:

- *Lack of complete graph knowledge:* Only in LS network all routers accumulate a complete graph knowledge, which is a requirement for running centralized algorithms. Optimization techniques based on centrality (e.g. Pop-Routing [13, 14]) may remain for this reason limited to LS networks, without providing their benefits also to distributed networks. Moreover, in large and complex networks such as the Internet, acquiring the complete topological information is not always possible. The lack of a complete graph knowledge hampers the adoption of centralized algorithms for the computation of centrality also in these cases.

- *Scalability issues:* Centralized algorithms suffer from scalability issues. For example, despite the introduction of several heuristics [91], it has been shown that computation of BC on low-power hardware can be performed online and in real-time only on networks of up to a hundred of nodes [92], while it becomes impracticable on a larger scale.

Addressing the Scalability Challenge

The time complexity of Dijkstra, i.e. $O(n^2 \log(n))$ in graphs of the form $G(\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$, can be considered a general lower bound to the time complexity of centralized algorithms for centrality. All of them always embed, in fact, an instance of Dijkstra to initially identify all shortest-paths in the graph, a necessary knowledge to later enable the computation of the centrality metrics. The scalability issues of centralized algorithms for centrality are rooted in this complexity, so much that the computation of indexes such as the BC via the Brandes algorithm quickly becomes time-consuming and power-hungry with the growth of the network size [92]. The problem get worsened when the centrality must be recomputed continuously and online, for example, when the centrality keeps changing over an evolving networks subject to frequent topological changes. The most common approaches proposed in the literature to address the scalability issues of centralized algorithms are two:

1. *Incremental Updates of Centrality:* some algorithms are available that do not perform the recomputation of centrality from scratch for every change of the topology [93–96]. With these algorithms it becomes possible to efficiently update the precomputed indexes taking into account only the most recent modification of the network graph. Although able to take the network dynamism into account and to update efficiently the centrality indexes over evolving graphs, these algorithms cannot provide significant advantages in front of major topological modifications that affect the large majority of the indexes. For example, the removal of a link in a ring topology clearly affects the centrality indexes of all nodes, so much that the process of updating the centrality indexes is equivalent to a full recomputation from scratch. The advantages of incremental algorithms are limited therefore to those networks that exhibit a modest rate of minor topological variations over time.

2. *Random sampling:* When the size of the graph becomes too large making the centrality computation not feasible, a natural approach to reduce the size of the problem is random sampling [97–102]. Essentially, the computation of centrality is performed over a subset of nodes and edges sampled from the full graph and the so computed indexes result to be good approximations of the true centrality values. For instance, Jacob et al. [100] and Brandes and Pich [98] proposed algorithms that only consider the shortest-paths induced by a subset of vertices sampled uniformly at random to approximate the computation of BC. In both works of Riondato et al. [103] and Baglioni et al. [104], the accuracy and confidence of the approximations of BC can be adjusted tuning the size of samples. Although more efficient, algorithms based on random sampling do not provide exact results. Moreover, even a perfectly uniform sampling process induces biased sub-graphs in which the centrality of nodes included in the sample tends to be overestimated.

Albeit efficient when dealing with evolving graphs or when approximating centrality over large graphs, the general limit of centralized algorithms is the necessity of starting from the complete graph knowledge. This requirement alone hampers the introduction of centrality-based optimizations in all distributed networks, even when scalability and complexity are not an issue. For example, the Pop-Routing optimization technique [13, 14] exploits the centralized Brandes algorithm to compute the BC, and could not be introduced in distributed networks without a distributed algorithm able to compute centrality starting from local information only (such as the one that will be presented in Chapter 3 as a major contribution of this thesis). Distributed algorithms for the computation of centrality metrics are explored in Section 2.5.1.

2.5.1 Distributed Algorithms for Centrality

Distributed algorithms for the computation of centrality exist and do not require a complete topological knowledge, also exhibiting acceptable scalability properties. However, strong hypotheses on the graph structure or on the model of the distributed system are usually introduced to enable the relaxation of the full graph knowledge requirement. A compromise of the generality is therefore the cost to enable the distributed computation of centrality with such approaches. For example, the distributed algorithms proposed in [105–107] are limited to graphs in forms of DAGs or trees, this to comply with the hypotheses of working only with loop-free topologies and, in the particular case of trees, to ensure that *"every pair of nodes has at most one shortest path"* [105]. Also the works by Wang et al. [106, 107] are based on strong hypotheses on the structure of the graph, namely, they are restricted to *"undirected and unweighted tree graphs"*.

Many distributed algorithms for the computation centrality metrics such as [108, 109] proposed to compute the BC are proven to execute correctly under the popular `CONGEST` model [110]. Exemplary assumptions of the `CONGEST` model are the following:

- All links are bidirectional, so much that the network is modeled by an *undirected* graph;
- Nodes and links are reliable, i.e., failure-free: abrupt stops of nodes and drops/delays of links are not considered;
- Communication is *synchronous*, with a global clock that triggers consecutive rounds of message passing and processing;
- Node power is infinite, message processing can hence be considered instantaneous.

The `CONGEST` model, however useful for proving the theoretical correctness of distributed algorithms, sets a series of ideal conditions that cannot be met in real-world deployment: the practical feasibility of algorithms proven to work under the `CONGEST` model cannot be granted.

Distributed algorithms for centrality are designed also to overcome the scalability issues highlighted for centralized algorithms. When focusing on scalability rather than on accuracy, numerous are the distributed algorithms that can estimate different centrality metrics [111–114]. This kind of algorithms provide only approximation of the true indexes: accuracy must be necessarily sacrificed to favor a greater scalability.

Towards an exact and distributed algorithm for Load Centrality (LC)

In this thesis, namely in Chapter 3, an exact and distributed algorithm for LC is proposed to overcome the known limitations of current centralized and distributed algorithms for

centrality. This exact and distributed algorithm for LC enables centrality-based optimization techniques in distributed networks. In particular, this thesis reports the results of applying a Pop-Routing methodology, driven by LC, to reduce the impact of failures in networks managed by a DV routing protocol.

It may be also argued that the semantic of BC better captures the dynamics of flows in computer networks compared to LC. Consider, however, that BC is actually very similar to LC, so much that is often mistaken for it [84]. Moreover, the BC converges to LC under the assumption of having a single shortest-path for connecting each couple of endpoints, which is a valid assumption to model all those networks running a routing protocol that does not support multipath. For modern distributed systems that instead support multipath routing or that employ load balancing techniques —consider for insance the Stream Control Transport Protocol [115] or Multipath TCP [116]— the LC is even a better metric than BC, since it precisely capture the distribution of traffic over multiple paths. Furthermore, only the LC can capture the true paths used in computer networks with policy-based routing, a kind of knowledge that is in general not possible to acquire not even owning the full graph. For instance in the Internet, managed by the BGP Path-Vector (PV) protocol, the routing policies adopted by different ASes are kept secret to hide business strategies, and it is well known that paths chosen for routing in the Internet are not those shortest-paths that would be identified running an instance of Dijkstra on the full network graph. With the algorithm for LC the centrality is instead computed according to the propagation of UPDATE messages in the network. Therefore, although not explicitly aware of the secret routing policies, the LC still succeeds in capturing the real routing paths and the flows of traffic of a network. For this reason it can be claimed that the LC is even better than the BC to model the behavior of BGP and other policy-based routing protocols.

Chapter 3

Distributed Algorithm for Load Centrality and Application to DV Routing

- A distributed and exact algorithm for the computation of Load Centrality (LC) will be introduced in Section 3.1. The algorithm will be described in the most abstract terms to highlight its generality and scope beyond its use to steer the optimization of routing protocols.

- An incrementally deployable version of the algorithm is also provided (Section 3.2). This version specifically targets routers running a Distance-Vector (DV) routing protocol.

- A theoretical analysis of the algorithms is conducted in Section 3.3. First part of the analysis is an investigation of the algorithm convergence properties (Section 3.3.1). Then a scenario of partial deployments will be considered: this scenario describes an incremental deployment of upgraded routers (upgraded so to run the centrality algorithm) in a network of legacy routers. In Section 3.3.2 the algorithm accuracy in ranking nodes by centrality will be formalized as a function of the penetration ratio of upgraded nodes in the network.

- An application of the algorithm for the optimization of a DV protocol is described in Section 3.4. The algorithm has been implemented in `babeld`, the open source distribution of the real world DV protocol Babel (implementation notes are discussed in Section 3.4.2). Then `babeld` timers have been tuned adopting a Pop-Routing approach on the base of the nodes LC values computed experimentally. Section 3.4.3 reports the experimental performance evaluation conducted to asses the effectiveness of Pop-Routing in reducing the impact of failures in distributed networks.

3.1 Distributed Algorithm for the Computation of the Load Centrality

The distributed algorithm for the computation of LC will be firstly formalized in Section 3.1.1, then an instance of the algorithm at work in a small network will be illustrated, in Section 3.1.2, to exemplify the introduced notation.

3.1.1 Algorithm Formalization

Algorithm 1 describes the algorithm from the perspective of the node v that executes it, while Table 3.1 is the related notation table, reporting the definitions for all symbols introduced in Algorithm 1.

Algorithm 1: General distributed Protocol (executed by v)

```
1  Init:
2      load = 0;
3      loadOut[v] = contrib[v] = 0;
4      foreach d ∈ V − {v} do
5          foreach u ∈ neighbors do
6              loadIn^u[d] = 0;
7          PH[d] = [ ];
8  Repeat every δ seconds:
9      foreach d ∈ V − {v} do
10         loadOut[d] = 1 + Σ_{u∈PH[d]} loadIn^u[d];
11         contrib[d] = loadOut[d] / |NH[d]|;
12         load = load + loadOut[d];
13     send ⟨v, NH, contrib⟩ to neighbors;
14 On receive ⟨u, NH^u, contrib^u⟩ from u do
15     foreach d ∈ V − {v} do
16         if v ∈ NH^u[d] then
17             PH[d].add(u);
18             loadIn^u[d] = contrib^u[d];
19         else
20             PH[d].delete(u);
21             loadIn^u[d] = 0;
```

Algorithm 1 is a natural formalization of the load diffusion process described by Definition 2.2, with every node generating a unitary amount of load for all destinations discovered via the underlying routing protocol. Nodes periodically sends messages to their neighbors to

propagate this load in the network. This message passing process, based on both splitting and aggregation of load contributions, will let all nodes recursively accumulate their own load, i.e., their LC index. After having briefly introduced the main working principle of the algorithm it is possible to delve into the three main phases that characterize the algorithm, namely:

1. Initialization (lines from 1 to 7);
2. Sending Phase (lines from 8 to 13);
3. Receiving Phase (lines from 14 to 21);

Initialization: Node v sets initially to zero all the variables needed to keep track the evolution of the load diffusion process. The variable *load*, for example, is now set to zero but later will converge to the LC index of v. Note also the two dictionaries, namely *loadOut* and *contrib*, initially reporting a zero only for the same v: these dictionaries will support the tracking of outgoing load contributions. In particular:

- *loadOut* stores for each destination d the overall load going out from v as destined to d. The sum of all contributions stored in *loadOut* will give at the algorithm convergence the final LC index of v.
- *contrib* stores the contributions that v sends to its neighbors.

The two dictionaries are almost equivalent[11]. In fact, in general $contrib[d]$ is equal to $loadOut[d]$ for all destinations d for which v knows a single next-hop but not when the outgoing load must be splitted among multiple next-hops In this case $contrib[d] = loadOut[d]/|NH[d]|$, with $NH[d]$ used to indicate the set of next-hops known by v to reach d. NH is thus again a dictionary, reporting the set of next-hops for each destination d as updated by the underlying routing protocol. PH is the opposite dictionary of the previous-hops sets, initially empty (line 7). The last variable to be defined is *loadIn*, which is the opposite of *loadOut* as it reports the incoming load contributions received by v. It is initialized to 0 for all destinations (in lines 5-6), while waiting for contributions to come with the progress of the load diffusion process. In *loadIn* each contribution is indexed by destination d but also by the neighbor u that sent it (u appears in apex notation).

Sending phase: Each node executes the sending routine (lines 8-13) periodically every δ seconds. For every known destination d (v excluded) the node v sums up all received contributions from all its neighbors and adds to this summation the unitary amount of load generated by the same v. The "1+" at line 10 represents this unitary load contribution, while

[11] The dictionary *contrib*, albeit redundant, is introduced to ease later the analysis of the algorithm convergence properties.

the following summation is the collection of all incoming contributions (stored in *loadIn*) from any neighbor u. The so computed load defines the overall load going out from v to be forwarded to d, hence it is assigned to $loadOut[d]$. The specific contribution, possibly a fraction of the total one in case of multiple next-hops, is computed at line 11 and assigned to $contrib[d]$. Line 12 is fundamental since it updates the *load* variable within the loop that iterates over all destinations. This is fundamentally the implementation of the summation used to define the LC (recall Definition 2.2), so much that the variable *load* will converge precisely to $LC(v)$. At line 13 node v broadcast to all its neighbors:

- the just updated *contrib* dictionary;
- the dictionary of selected next-hops (NH), as always updated by the underlying routing protocol;
- the identifier of v to tag this broadcast message.

The neighbors of v will use this information to update their incoming load and their previous-hops sets.

Receiving phase: The first action performed by v upon reception of a message from a neighbor u is the scan of the NH^u dictionary (i.e., the NH dictionary of neighbor u) to check if v has been been selected by u as next-hop for some destination d (line 15-16). If so, then u is added to the $PH[d]$ set as previous-hop (line 17). Moreover, v accepts incoming contributions only from previous hops: this is why line 18, that updates the dictionary of incoming load *loadIn*, is protected by the if statement of line 16. Lines 19-21 reflect possible changes of the routing choices of a neighbor u that, for some reason, may stop choosing v as favorite next-hop for some destination d. In this case u must be removed from the PH set (line 20) and contributions received from u must be forgotten (line 21).

3.1.2 Conceptual Illustration of Algorithm 1

Figure 3.1 gives an example of Algorithm 1 at work in a small network.

Step 1 This initial step (Figure 3.1a) highlights the peripheral nodes that generate a unitary load contribution for any reachable destination, sending it to their favorite next-hop. For example, node A can reach $[B, C, D, E]$ via the next-hop C: the message above the arrow drawn from node A to C indicates that A is sending to C one unitary load contribution (the bold 1) for all these reachable destinations.

Step 2 In this second step (Figure 3.1b) load contributions are not just generated but also *aggregated*. The message sent from C to D highlights, in fact, a non unitary load contributions equal to 3: this 3 results from the aggregation of the unitary contributions

Table 3.1 – Variables used by each vertex v in Algorithm 1

Symbol	Description
$\mathcal{V}$	The set of all vertices (aka nodes)
neighbors	The neighbors set of v
NH	A dictionary indexed by destinations. For each destination $d \neq v$, the corresponding $NH[d]$ is the set of next-hops selected by the underlying routing protocol run by v to forward traffic towards d
PH	The complementary dictionary of NH. Each $PH[d]$ entry is the set of *previous hops*, i.e., those neighbors of v that selected v as next hop to reach d
loadOut	For each destination $d \neq v$, $loadOut[d]$ is the overall load routed by v towards d
contrib	Similarly to *loadOut*, *contrib* stores the outgoing load contributions, but here are splitted in case of multiple next-hops. In fact: $contrib[d] = loadOut[d]/\lvert NH[d]\rvert$
$loadIn^u$	The opposite dictionary w.r.t. *loadOut*. An entry $loadIn^u[d]$ stores the incoming load received by neighbor u for destination d
load	The variable that, at convergence, will indicate the load centrality of v

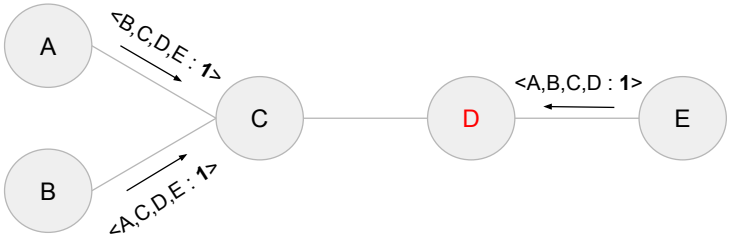
received by A and B with the one generated directly by the same C. Here the focus is on the load contributions sent by C only to D and E, i.e., the only two destinations that C reaches via D. These contributions are, in fact, the only ones among those routed by C that contribute to the LC of the node of interest D.

Step 3 At the end of the process (Figure 3.1c) D should own a table of received contributions like the one shown here. Summing up all received contributions (except those destined to D itself), node D can compute its own LC index.

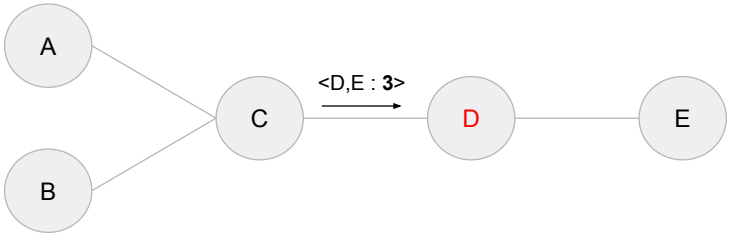
Here some illustrative parallelisms between Figure 3.1 and Algorithm 1:

- The unitary load contribution sent by nodes A and B in Figure 3.1a is the same "1+" that can be read at line 10 of Algorithm 1;
- The load contribution equal to 3 visible in Figure 3.1b would correspond to $loadOut[D] = loadOut[E] = 3$, assuming C to be the node that is running Algorithm 1;
- The summation at the end of the table reported in Figure 3.1c is the same summation that updates the variable *load* at line 12 of Algorithm 1;

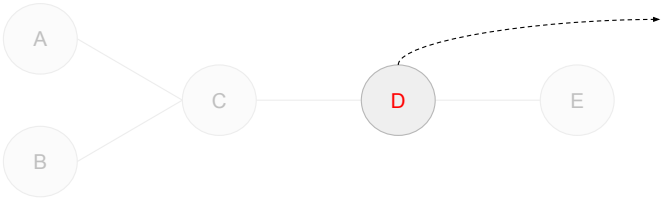
Highlights and Concluding Remarks: The protocol described so far is general enough to be executed in networks of arbitrary topology with the only assumption that an underlying routing protocol must exist to keep next-hops updated. This version of the algorithm is proposed to support any application that may exploit the LC notion. Anyhow, the main focus of this thesis is the porting of centrality-based optimizations from centralized to distributed networks. Section 3.2 presents a variation of the general Algorithm 1 that targets DV routing protocols.



(a) Step 1: Load generation



(b) Step 2: Diffusion and Aggregation of load



Destination	Contributors
A	[E, 1]
B	[E, 1]
C	[E, 1]
~~D~~	~~[C, 3] [E, 1]~~
E	[C, 3]
TOT	1 x 3 + 3 = 6

(c) Step 3: LC Computation

Figure 3.1 – Illustration of Algorithm 1 at work in a small network. The focus is on node *D* (in red), for which the LC will be computed step-by-step following the diffusion process of load contributions.

3.2 Incrementally Deployable Algorithm for DV routers

A novel incrementally deployable algorithm for LC is presented in this section. It is proposed as an extension to the fundamental Bellman-Ford algorithm and, introducing an only minimal extension to the regular distance-vector mechanism, it empowers the computation of LC in distributed networks. Furthermore, the new algorithm is designed in compliance of the *general considerations about extension mechanism for networking protocols*, reported by the IETF in RFC 6709 [117]. This means that the algorithm can be implemented extending a real-world protocol and can also be incrementally deployed in production networks.

The algorithm will be introduced describing its possible incremental deployment in a network where two kinds of nodes will coexist, namely:

- *Legacy nodes*, i.e., those nodes hypothetically running a legacy version of a generic DV protocol;
- and *Upgraded nodes*, that instead are supposed to run an updated version that integrates the LC algorithm.

Compliance with standard extension mechanism (RFC 6709): The use of type-length-value (TLV)[12] packets is one of the main recommendations about the extension of networking protocols provided by the IETF [117]. The use of TLVs simplify the extension of protocols: defining a new TLV is enough to add a new type of message. The message will be syntactically correct for any node, therefore it will be backward-compatible with legacy nodes. Only upgraded nodes, however, will be able to decode and send the new TLV, while legacy ones will simply not recognize the message type. In this case legacy nodes complying with the IETF recommendations for protocol extension will store and silently forward the information they are not able to interpret. This strategy for handling packets of unknown type is called *Silent Propagation*. The silent propagation scheme is implemented by many popular routing protocols such as OLSR [118], Babel [16] and BGP [79] that, in the present case, together represent all possible routing families (respectively, Link-State (LS), DV and Path-Vector (PV)). In particular the BGP supports the Silent Propagation scheme with its famous *Optional Transitive Attributes* [119].

Modeling Legacy and Upgraded Nodes: The silent propagation scheme has been introduced because it has been chosen to model the behavior of legacy nodes while handling *centrality* TLVs, i.e., the new TLVs that will support the distributed computation of LC in the incrementally deployable algorithm. Algorithm 2 is the full algorithmic description of the legacy nodes model. Essentially a legacy node performs a generic DV protocol based on Bellman-Ford and silently forwards the new centrality TLVs generated by upgraded nodes. The logic of upgraded nodes, comprehensive of the generation and interpretation of centrality TLVs, is instead described by Algorithm 3. The new notation used to model legacy and upgraded nodes extends the notation already introduced presenting the generic Algorithm 1; this new notation is summarized in Table 3.2.

3.2.1 Legacy Nodes

A legacy node is modeled as a generic DV router: it embeds the Bellman-Ford algorithm with the only further key aspect of owning and controlling a buffer (BUF) to temporarily store those

[12] The type-length-value (TLV) is a popular encoding scheme for data communication protocols characterized by a 3-fields format to define every type of messagge. The 3 fields are: “Type”, “Length” and “Value”. The type field reports a standardized numeric code that indicates the type of the message, for example, a specific code may indicate that a message is of type UPDATE. The byte-length of the message value is reported to let the receiver know the limits of the message to be parsed. The value field is the actual message.

Algorithm 2: Logic of a Legacy node v

```
1  Init:
2      RT[v].m = 0; RT[v].NH = [ ]; BUF[v] = [ ];
3  Repeat every δ seconds:
4      Cleaning(now);                          // Clean expired subTLVs from BUF
5      foreach u, d ∈ neighbors × RT() do
6          if u ∈ RT[d].NH then
7              fwd = BUF[d];                   // forward subTLV only on SPs
8          send ⟨d, RT[d].m, fwd⟩ to u;
9  On receive ⟨d, m, fwd⟩ from u do
       // Bellman-Ford
10     if d ∉ RT() OR m + C[u] < RT[d].m then
11         RT[d].NH = [u];
12         RT[d].m = m + C[u];
13     else if m + C[u] == RT[d].m then
14         RT[d].NH.append(u);
15     Cleaning(u, d);                         // Clear old subTLVs from u to d
16     foreach stv ∈ fwd do
17         stv.holdTime = now + ε;
18         BUF[d].append(⟨u, stv⟩);                    // Buffer subTLVs
   /* Dictionary C[·] contains the cost of the links to neighbors */
```

TLVs that must be silently forwarded. Upgraded nodes will attach the centrality information to UPDATE messages, also encoded as TLVs. The centrality information is therefore considered to be a subTLV, since it will always be a sub-element of a parent TLV (namely, a parent UPDATE). Centrality subTLVs will be recorded in the buffer mapped to the destination d that was announced in the parent UPDATE message, *BUF* is thus a dictionary indexed by destinations. At initialization the buffer is empty (line 2).

Legacy Sending phase: Lines (3-8) define the periodic routine for sending route-advertisements. UPDATE messages are sent by node v to each neighbor[13] u to announce available routes for each known destination d (line 8). Beyond the classic advertisement of the destination d with the related distance $RT[d].m$ (recall Section 2.1 for the notation introduced to manipulate Routing Tables) legacy nodes also:

- retrieve from *BUF* the subTLVs contained in recently received UPDATES for d (line 7);
- append the subTLVs to be forwarded (i.e., *fwd*) to the end of UPDATE messages.

[13] In modern protocols it is required to send different DVs to different neighbors, e.g., to implement the split-horizon technique, widely used to mitigate the well-known *count-to-infinity* problem.

This way legacy nodes implement the silent propagation scheme. Thanks to the control performed at line 6 subTLVs are forwarded only to those neighbors that are also next-hops for the announced destination. It must be noticed that, consequently, subTLVs are forwarded only along shortest paths. Lastly at line 4 a cleaning routine is invoked to discard all the expired subTLVs. SubTLVs are in fact tagged at reception and expire if not updated within a desired time interval.

Legacy Receiving phase: This phase is defined by lines 9-18 of Algorithm 2. A legacy node processes advertisements of the form $\langle d, m, fwd \rangle$: d and m form the (destination,distance) pair typical of any DV protocol, while *fwd* is the list of subTLVs that neighbor u may have silently forwarded to v. Lines 10-14 describe a classic update of the *RT* according to the Bellman-Ford criterion. Neighbors are selected as *NH* if they offer the shortest-path to reach the advertised destination. Notice that multipath is supported, with more than a single next-hop annotated in the routing table if more neighbors offer alternative equivalent paths (line 14). Legacy nodes cannot parse unknown subTLVs by assumption. Unparsed subTLVs will be therefore simply stored in *BUF* indexed by d, marked with the id of the sending neighbor u, and tagged with a time validity limit (as shown in lines 16-18). Legacy node discards previous information received by the same neighbor for the same destination before storing a new subTLV (line 15).

3.2.2 Upgraded Nodes

An upgraded node can generate and parse subTLVs. SubTLVs are the minimal amount of information required to run the incremental deployable protocol for the computation of LC and essentially encode the load contributions already seen in the general Algorithm 1. Algorithm 3 describes the logic of an upgraded node. Many fundamental mechanisms that were present already in the general Algorithm 1 are preserved in the behavior of upgraded nodes. For example, an upgraded node (Algorithm 3) can:

- generate and send load contributions (lines 5-12);
- store and aggregate received contributions (lines 14-17);
- sum up all received contributions to estimate its own LC index.

Upgraded Sending phase: This is the phase in which an upgraded node generates and sends load contributions, encoded as subTLVs. For instance, compared to a standard legacy node, at line 7 it is possible to notice the generation of the unitary load contribution, assigned to the local variable *loadOut*. In the next lines (8-9) the contributions to be sent towards the announced d are aggregated in *loadOut*. The reader should notice that the semantics of the *loadOut* slightly changed from the one used in Algorithm 1. Before it was a

Algorithm 3: Logic of an Upgraded node v

```
1  Init:
2      RT[v].m = 0; RT[v].NH = [ ]; RT[v].loadIn = { }
3  Repeat every δ seconds:
4      Cleaning(now)                                  // Remove expired contributions
5      foreach u, d ∈ neighbors × RT() do
6          if u ∈ RT[d].NH then
7              loadOut = 1                            // Generate/send contrib on SPs
8              foreach u ∈ RT[d].loadIn() do
9                  loadOut += RT[d].loadIn[u]         // Aggregate received contributions
10             contrib = loadOut/|RT[d].NH|
11             stv = (v, u, contrib)                  // subTLV with source v sent via u
12             send ⟨d, RT[d].m, stv⟩ to u
13 On receive ⟨d, m, fwd⟩ from u do
       /* (Omitted) Bellman-Ford as in Algorithm 2 */
       /* Process list of subTLVs attached to d */
14     foreach stv ∈ fwd do
15         source, rNH, contrib = stv
16         RT[d].loadIn[source, rNH] = contrib
17         RT[d].loadIn[source, rNH].holdTime = now + ε
   /* The load centrality of v is given summing up all contributions in RT().loadIn. */
```

Table 3.2 – Extended Notation for Legacy and Upgraded nodes

Field	Notation	Description
Buffer	*BUF*	dictionary mapping destinations d to the list of subTLVs contained in recent advertisements for d
SubTLV	*stv*	tuple of information, appended to a parent TLV such as an UPDATE message
Forwarded subTLVs	*fwd*	list of subTLVs
Source of load	*source*	id of the upgraded node generating a contribution
Remote NH	*rNH*	next-hop selected by a remote upgraded node to forward its subTLVs

dictionary collecting all load contributions sent out from v, now it is instead a local variable that, iterating over all destinations d, at the end of the aggregation loop converges to what was previously defined as $loadOut[d]$. Changing *loadOut* into a local variable makes the algorithm more memory-efficient. Also *contrib* is now a local variable, as usual it represents that outgoing load splitted among all possible *NH*, and is thus equal to $loadOut/|RT[d].NH|$ (line 10). A subTLV defining a load contribution is finally generated at line 11 in the form of a

3-ple $(v, u, contrib)$. Essentially node v tags with its own identifier the just generated *contrib* and also specifies that u is a favorite next-hop for v. The so formed *stv* is then appended to the UPDATE message sent at line 12.

Upgraded Receiving phase: In the upgraded receiving phase, described by lines 13-17 of Algorithm 3, the update of the Routing Table according to the Bellman-Ford criteria has been omitted, as it is the same of Algorithm 2. Unlike a legacy node an upgraded one does not ignore the subTLVs attached to received UPDATES but it parses them as shown on lines 14-17. In particular at line 15 a subTLV is unpacked into three pieces of information:

- *source*: the id of the remote node that generated this contribution;
- *rNH*: the *NH* towards which the remote node sent the contribution;
- *contrib*: the amount of forwarded load.

The reader should notice the variable-names chosen by an upgraded node when parsing a received subTLV. Consider, for instance, that a received subTLV may not come from another upgraded node, but it may arrive after a chain of silent forwardings by several legacy nodes. The *source* tag in a subTLV may therefore identify a remote upgraded node, not necessarily a neighbor of v. For the same reason *rNH* is just the next-hop that the remote *source* node chose as his favorite next-hop. Received contributions are stored in the *RT*, indexed by their *source* and *rNH* (line 16): indexing contributions also by *rNH* is needed to distinguish contributions originally splitted by the same remote *source*. Without the *rNH* information it would be impossible to sum up these kind of contributions in aggregation points reached after flowing through different paths. Figures 3.2 and 3.3 illustrate two example scenarios to highlight the important role of the *source* and *rNH* tags used by upgraded nodes to process received contributions correctly.

The time validity of contributions stored in *loadIn* is limited (line 17). The time validity of contributions is essential to properly forget those contributions that comes from paths that, because of some remote routing decision, do not cross anymore node v. Expired contributions are discarded invoking periodically a cleaning routine (line 4).

3.2.3 Conceptual Illustration of Algorithm 3

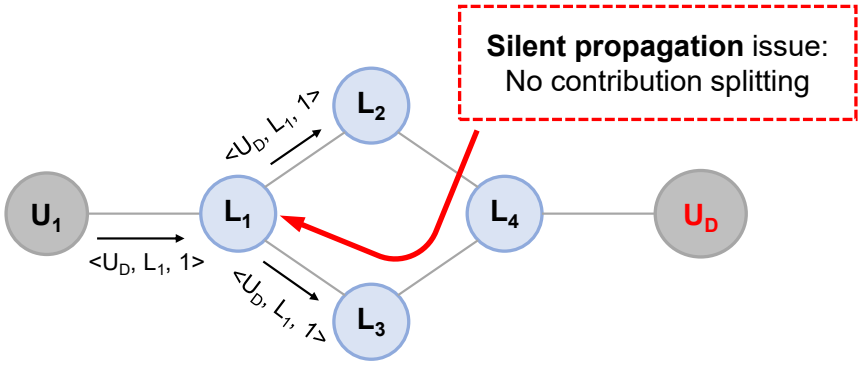
Figures 3.2 and 3.3 illustrates two main scenarios motivating the introduction of the *source* and *rNH* tag in the contributions generated by upgraded nodes. In these figures upgraded nodes are represented by gray circles and labeled with capital U, while legacy nodes are colored in light blue and are labeled with a capital L. Messages (i.e., contributions) are represented by triples of the form $\langle source, rNH, contrib \rangle$ exactly as on line 15 of Algorithm 3, with arrows to indicate the direction of the message propagation.

The main problem from an upgraded node is to understand if an incoming contribution generated by the same source is a duplicate, and must be discarded, or if it is a fraction of a contribution splitted by some remote source node to be merged with other similar fractional contributions.

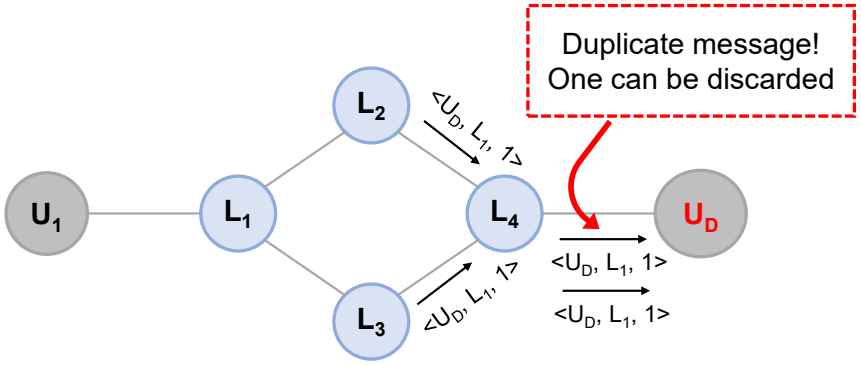
Figure 3.2 illustrates the case of a duplicate contribution correctly discarded by an upgraded node.

- For instance, in Figure 3.2a the unitary contribution generated by the upgraded node U_1 is received from the legacy node L_1 that, being unable to parse upgraded subTLVs, blindly forwards the unparsed information on both his outgoing links, erroneously duplicating the received contribution.

- Because of this error in Figure 3.2b the reader can notice how L_2 and L_3 keep blindly forwarding the duplicate contributions so that U_D receives both of them from L_4. However, U_D understands these are duplicates because they report the same $(source, rNH)$ tuple, so it is able to discard one of the two.



(a) Erroneous duplication of contributions.

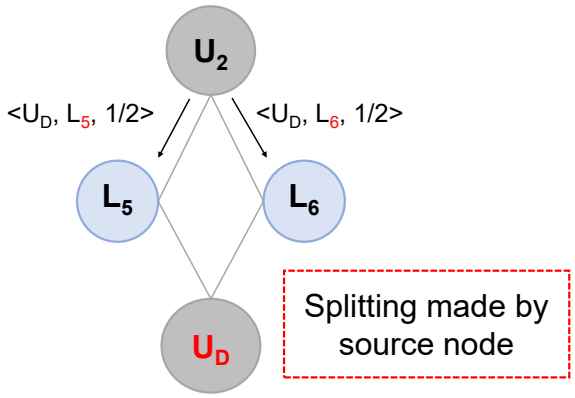


(b) Correct detection of duplicates: U_D notices that received contributions share the same $(source, rNH)$ tuple and correctly discard one of the two.

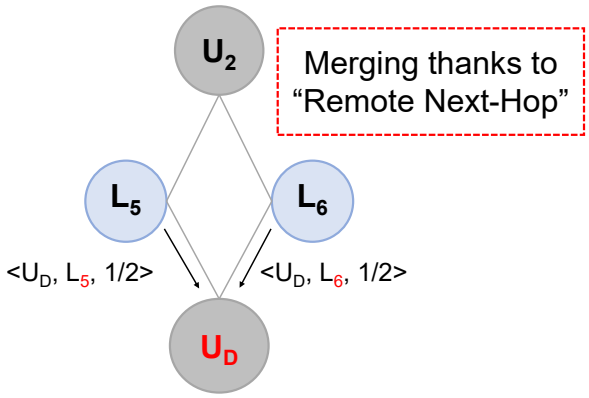
Figure 3.2 – A scenario where an upgraded node (U_D) running Algorithm 3 is able to distinguish and discard duplicate contributions erroneously forwarded by legacy nodes.

Figure 3.2 illustrates instead the case of properly splitted contributions that must be processed accordingly, without being mistakenly considered duplicates.

- In Figure 3.3a the upgraded node U_2 identifies L_5 and L_6 as equivalent next-hops to reach U_D, therefore, it halves his contribution before sending it to L_5 and L_6.
- In Figure 3.3b U_D receives two contributions reporting the same *source*, a warning symptom of duplicates. However the two contributions have been tagged by the source U_2 with different *rNH* fields: notice, in fact, the red subscripts for the *rNH* fields present in the messages that reach node U_D. Essentially, thanks to *rNH*, an upgraded node can correctly merge contributions splitted remotely in the network by some other upgraded node.



(a) Compared to Figure 3.2a, this time the contribution splitting is performed appropriately by an upgrade node (U_2) before being received by legacy nodes.



(b) Even if U_D receives two contributions tagged with the same *source*, thanks to the different *rNH* fields U_D becomes able to merge the contributions, instead of erroneously discard one of the two.

Figure 3.3 – A scenario where an upgraded node (U_D) running Algorithm 3 is able to correctly merge contributions thanks to the *rNH* tag, which enables the correct distinction between different contributions sharing the same *source*.

3.3 Theoretical Analysis

The convergence properties of Algorithm 1 are studied in Section 3.3.1:

- The proof of its convergence is elaborated in Section 3.3.1.1, together with corollaries that fix the time bounds for the algorithm execution in failure-free system epochs;
- Theorems presented in Section 3.3.1.1 are validated, via simulation, in Section 3.3.1.2.

The algorithm accuracy is further studied in Section 3.3.2, in particular:

- A sanity-check is performed in Section 3.3.2.1 to verify that the implementation of Algorithm 1 computes correct values of LC, i.e., the same values computed by a well known graph manipulation library;
- Section 3.3.2.2 studies the error in estimating the LC in partial deployments of upgraded nodes that run Algorithm 3;
- Section 3.3.2.4 exploits rank correlation coefficients to determine if the approximated LC indexes, computed in partial deployments, preserve the ranking of nodes defined by their theoretical centrality.

3.3.1 Convergence Study

3.3.1.1 Proof of Convergence

The proof that for any node v running Algorithm 1 the *load* variable eventually converges to $LC(v)$ is provided here. The proof is based on the simple observation that paths selected by routing protocol are acyclic and non-infinite. Therefore, for non-infinite sequences of failures and assuming that routing choices are stable for a long enough period, then load contributions will necessarily be sent and aggregated along the paths from any source to any destinations, which is enough to let *load* eventually converge to $LC(v)$. The following definitions support the proof:

Def. 3.1: ***Path*** Let $p(s,d) = [(s,u_0),(u_0,u_1),\ldots,(u_n,d)]$ be a (necessarily finite and acyclic) sequence of edges representing a shortest path from s to d. If appropriate for the context, $p(s,d)$ can be simplified to be a sequence of nodes $[s,u_0,u_1,\ldots,u_n,d]$.

Def. 3.2: ***Routing Graph*** Let $G_d = (\mathcal{V}, E_d)$ be the *"routing graph induced by d"*, i.e, the spanning tree rooted in d that includes all the shortest paths from any source s ending in d, as computed by the underlying routing protocol.
Notice that $E_d = \{(i,j) : (i,j) \in \bigcup_{\forall s \in \mathcal{V} \setminus \{d\}} p(s,d)\}$.

Def. 3.3: ***Collection of all Routing Graphs*** Let $\widetilde{G} = \{G_d = (\mathcal{V}, E_d) : d \in \mathcal{V}\}$ be the collection of all routing graphs induced by each node of the network.

Theorem 1:
If $\widetilde{G}$ remains stable for a time long enough then, for each node v, the '*load*' variable will eventually converge to the correct $LC(v)$.

Proof. For each destination d, the routing protocol generates a routing graph G_d that is a DAG (directed acyclic graph) made of all the (potentially multiple) shortest paths from any s

to d. Let $S = \{u_1, u_2, \ldots, u_{|\mathcal{V}|} = d\}$ be a sequence representing the reversed topological sort of G_d obtained via a breadth-first search initiated in d. Notice that the last nodes appearing in S are the closest to d, while the first ones are the furthest. It will be proven that each node v in S correctly accumulates the load contributions destined to d by induction on the sequence of nodes.

Base: By topological sort and given that G_d is a DAG, then u_0 is diametrically opposed to d, thus the set $PH[d]$ of this node is empty. The load going out from node u_0 will be set to 1 ($loadOut[d] = 1$) and, as expected, the load destined to d that *crosses* u_0 is 0. The unitary contribution sent by u_0 will be later splitted among the next-hops ($NH[d]$) of u_0.

Inductive hypothesis: Consider now a generic node u_k in S, and assume that all previous nodes in S, i.e., the nodes belonging to $Prev = \{u_0, \ldots, u_{k-1}\}$, already computed correctly their $loadOut[d]$ variable. All nodes belonging to the $PH[d]$ set of node u_k are included in *Prev* by topological sort. All the previous-hops —that are correct by hypothesis— will eventually send their load contribution to u_k, updating the corresponding entry in the *loadIn* register of u_k.

Conclusive step: Let's rename the generic node u_k as v. Nodes v correctly receives load contributions from its previous-hops because all of them are correct by inductive hypothesis. Provided that v is a generic node and that d has not been fixed, then it is possible to state that any node v accumulates properly all the incoming contributions for any d, which is necessary and sufficient to correctly update its *load* variable (recall the loop at lines 9-12 of Algorithm 1). *QED*

In this proof for Theorem 1 it is assumed that shortest-paths are stable for a long enough period of time, necessary to let the centrality computation converge according to the load diffusion process. Results are not definitive until the the load diffusion process converges, and becomes temporarily wrong in case of network dynamism. The correctness of algorithm is recovered when the routing process stabilizes again because, at that point, the periodic broadcast of load contributions will drive again towards the correct computation of the LC indexes.

Upper bounds to the convergence time: It will be shown that, without network dynamism and without failures, the load diffusion process converges in a number of steps that is in the worst case proportional to then network diameter. Let's assume that:

- nodes are synchronized;
- the time to propagate a message is small but not null;
- the re-transmission interval δ dominates over the propagation delay.

Therefore, each node v receives updates from its neighbors right after having broadcast its DV. The time to propagate an information over a k-hops long path is thus $\approx k \times \delta$ seconds. Assuming that the load diffusion process starts after the convergence of the routing protocol, then the following corollary holds:

Corollary 1:
Given a stable collection of routing graphs $\widetilde{G}$ with diameter D, the convergence time Δ_t of Algorithm 1 is in the worst case proportional to $D-1$.

Proof. The *load* of a node v converges when v has received all contributions from all the shortest-paths (towards any d) that cross the same v. Let's consider a generic path $p(s,d) = [s, u_1, u_2, \ldots, d]$ towards d that crosses v. If v is an endpoint of this path, i.e., $v = s$ or $v = d$, then $loadOut[d]$ converges immediately to zero, since no contribution for d will ever cross node v. Consider now S to be a topological sort of the graph induced by G_d, and let be $v = u_k$. For $k = 1$ then the $loadOut[d]$ variable of v converges after receiving all contributions from all paths of the form $p(s,d) = [s, u_1 = v, \ldots, d]$, and this happens in one propagation step, thus after δ seconds, thanks to the topological sort. In general, the $loadOut[d]$ variable of a node v that is the k-th element of a path $p(s,d)$ converges in k steps. In the worst case v is diametrically opposed to some destination d, i.e., $v = u_{D-1}$. In this case the $loadOut[d]$ of v converges in $\delta(D-1)$ seconds. Hence, after $D-1$ steps all nodes correctly updated their $loadOut[d]$ variable for any d, which is sufficient to let *load* converge. *QED*

When all nodes finished to compute their own LC index, they can disseminate it in the network to let all nodes learn the LC indexes of all others. This is useful, for example, if nodes need to locally compute a global ranking of nodes, or to normalize their own LC index w.r.t. to all other indexes. In general, after a first load diffusion process, a following *dissemination* process of LC indexes can take further D steps to complete. In conclusion, the time required to complete both the load diffusion process and the dissemination of the LC indexes will be, in the worst case, proportional to $2 \times \delta D$.

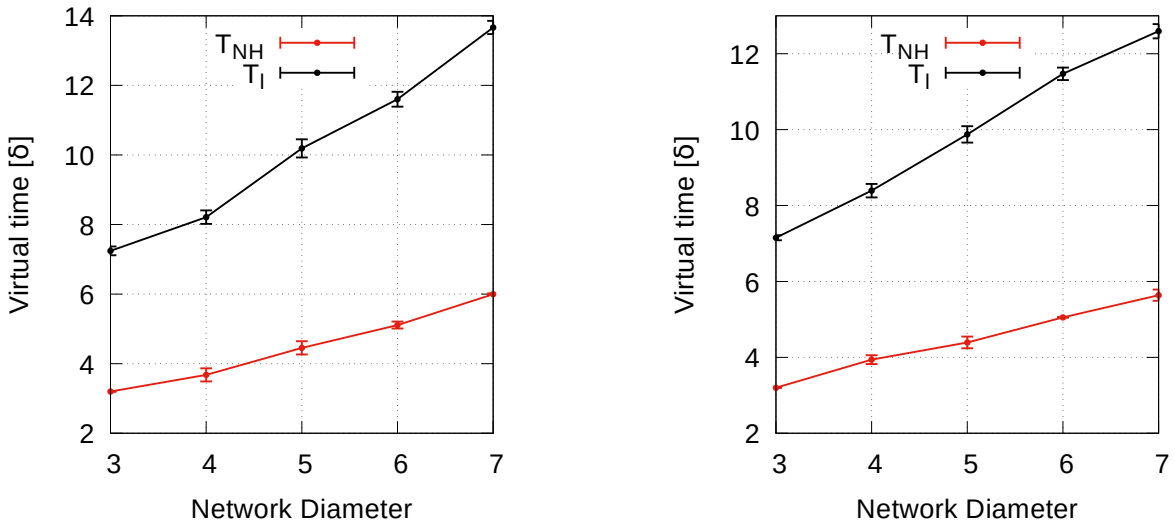
3.3.1.2 Theory validation via simulation

A Discrete Event Simulator (DES) for Algorithm 1 as executed on top of a generic DV router has been developed to measure the convergence time of both the load diffusion and the LC dissemination processes, so to validate the time upper-bounds discovered in Section 3.3.1.1. This simulator takes as input a network-graph, then uses a virtual clock to trigger in each node a *send event* periodically every δ time units. The measured convergence times will be therefore reported as multiples of δ. A small random jitter has been added to all events to avoid perfect synchronization of messages. Within a simulation it is possible to measure, for each node:

- The time needed to stabilize routing choices, i.e., to finalize the selection of next-hops included in *NH* (T_{NH});
- The "self-convergence time" (T_{sl}), i.e., the time at which each node finishes to compute its own value of centrality;
- The time T_l to learn the final LC index of all other nodes in the network.

A simulation ends when all nodes converge to steady state, i.e., when all nodes stabilized their routing tables, finished to compute their own LC index, and also learned the one of all other nodes. The slowest node to converge determines the full convergence of the simulation.

The upper-bounds on the convergence time discovered in Section 3.3.1.1 indicate a linear growth proportional to the network diameter D: for this reason, the simulator has been run over random graphs of growing diameter $D \in \{3,4,5,6,7\}$, generated according to the well-known Barabási-Albert and Erdős models. For each value of D and for both random-graph families, 40 different graphs with 1000 nodes each have been generated. These graphs have been provided as input to the simulator.



(a) Barabási-Albert networks.

(b) Erdős networks.

Figure 3.4 – T_{NH} and T_l vs. network diameter, with 99% confidence interval, for Barabási-Albert and Erdős networks. *Copyright © 2020, IEEE.*

Figures 3.4a and 3.4b show a linear growth of the average convergence time (namely of T_{NH} and T_l) with the network diameter, with the same trend visible for both Barabási-Albert and Erdős networks. This trend confirms the proportionality with the diameter stated in Corollary 1.

The time necessary to stabilize the selection of next-hops should be approximately equal to $D\ [\delta]$, as DVs must travel across the whole network to let each node discover also those

destinations that are diametrically opposed in the graph. However, in Figures 3.4a and 3.4b, T_{NH} is generally smaller than D. This result derives from the random jitter that has been added in simulation, while Corollary 1 has been elaborated under the assumption of perfect synchronization. It can thus happen that, approximately at the same time unit, a node i first receives a DV from neighbor j and then immediately propagates it to a further neighbour k, accomplishing two propagation steps in one time unit. This explains the fact that T_{NH} is in general smaller than the expected value D.

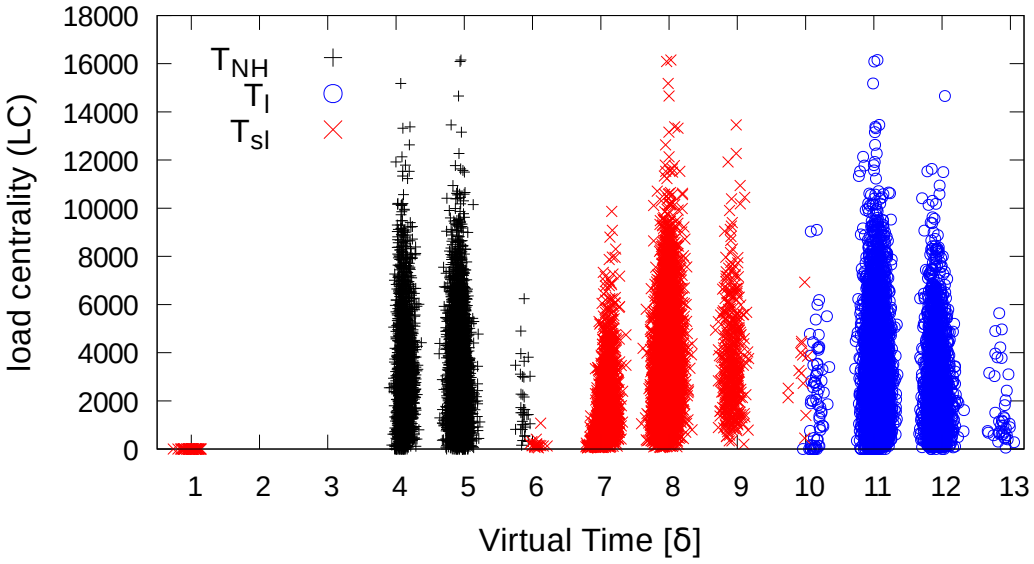


Figure 3.5 – T_{NH}, T_{sl}, and T_l for all the nodes, 40 simulations, Erdős graphs with diameter 7. *Copyright © 2020, IEEE.*

The theoretical analysis presented in Section 3.3.1.1 predicted the time to complete the LC indexes dissemination after routing convergence to be upper-bounded by $2 \times D\ [\delta]$. Simulation results indicate that T_l is in practice significantly lower, and this faster convergence is possible thanks to the parallelization of the the load-diffusion and LC-dissemination processes. This parallelization is made clear by Figure 3.5, that reports the T_{NH}, T_{sl} and T_l values for all nodes of each of the 40 Erdős networks with diameter 7 used in simulation. Networks with different parameters behave similarly and are not shown for brevity.

For instance, it is possible to notice that at time 6, when some nodes are the last to finalize their routing choices, some other nodes have already reached the self convergence. Similarly, at time 10 all nodes reached self convergence, but some others already reached full convergence. This happens because UPDATE messages carries all the information necessary for the three processes, namely for routing, to spread load contribution and to disseminate LC indexes, thus the three processes end up to be executed in parallel, reducing the expected

time to drive the network toward full convergence. At last, the reader can notice also the group of nodes that reaches self-convergence at the very first time unit: these are the leaf nodes produced by the Erdős generator whose centrality is zero and never changes since they are endpoint for all the shortest-paths they belong to.

3.3.2 Accuracy Study

- Section 3.3.2.1 reports the results of a sanity-check experiment. First, Algorithm 1 has been implemented on top of a real-world DV protocol, namely Babel, then a network graph has been chosen. At this point the LC indexes computed in emulation by the experimental implementation of the algorithm have been compared with the same values computed offline by a popular graph-manipulation library (python-`NetworkX`[14]) verifying this way that experimental results match with the expected theoretical ones. In this experiment all nodes were participating to the distributed computation of centrality, thus it can be considered as an experiment that mimes a full-deployment scenario.

- Section 3.3.2.2 reports instead the theoretical analysis of the error that afflicts upgraded nodes while estimating the LC indexes in partial deployment scenarios.

- Section 3.3.2.4 completes the accuracy study with an analysis of the *ranking-preservation* properties in partial deployments. The "ranking-preservation" will be studied in terms of the Spearman rank-correlation coefficient, used in the present case to quantify the similarity between: i) centrality rankings elaborated over LC indexes computed in simulated partial deployments with ii) the correct nodes' ranking computed in full-deployment.

3.3.2.1 Algorithm Correctness in Full Deployments

A sanity-check experiment has been conducted to validate the implementation of Algorithm 1 in Babel. Figure 3.6 reports the results by showing a comparison between empirical (black crosses) and theoretical (red dots) LC values as computed, respectively, with the Babel-implementation and with `NetworkX` for all nodes of a network. To collect the empirical LC indexes the open-source Babel routing daemon, `babeld`,[15] has been customized so to embed the distributed algorithm for load centrality. Then, a full DV network running instances of the customized `babeld` has been emulated thanks to the popular network emulator Mininet.[16] The network graph made of all the shortest-paths selected by the virtual `babeld` nodes has

[14] The `NetworkX` documentation: `https://networkx.github.io/documentation/stable/index.html`
[15] The open-source code of `babeld` is available on Github: `https://github.com/jech/babeld`
[16] More on Mininet, the Virtual Network Emulator, online: `http://mininet.org`

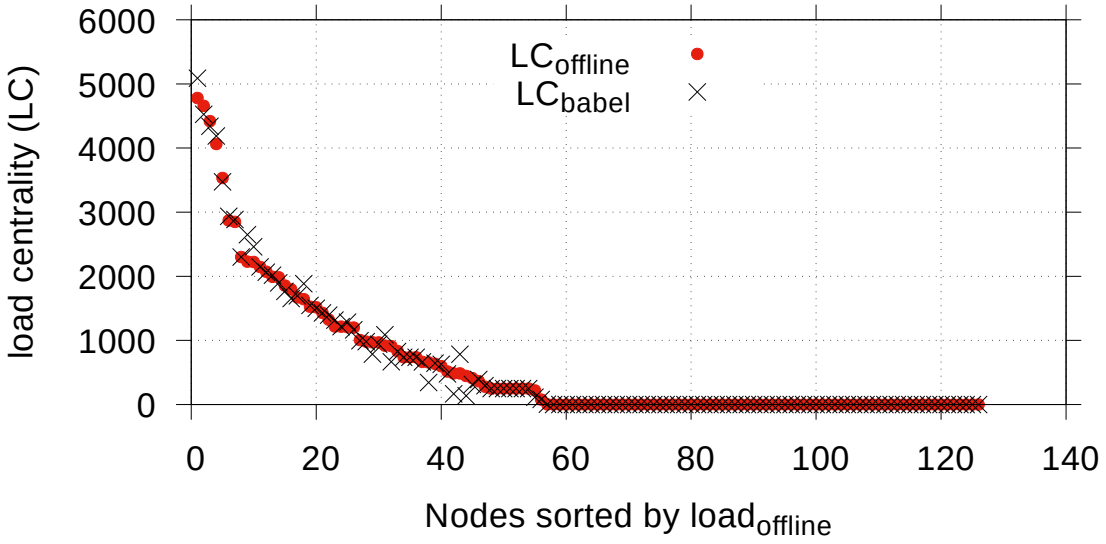


Figure 3.6 – Comparison of LC indexes computed: i) on-line with Algorithm 1 implemented on top of `babeld` and ii) off-line with `NetworkX`, for the same network-graph. *Copyright © 2020, IEEE.*

been recomputed, off-line, navigating through the routing-tables dumped by all nodes at experiment convergence. The so reconstructed network graph has been provided as input to `NetworkX`: the LC indexes computed by `NetworkX` have served therefore as ground-truth values. This sanity-check experiments has been repeated many times emulating different real-world networks for which their topology was known and made available [8]; for example, Figure 3.6 reports the results of the experiment run by emulating *ninux*,[17] the Wireless Community Network of Rome.

For most of the nodes, the empirical LC index match exactly with the expected one, while for a small fraction of nodes a little difference between empirical and expected LC can be observed. These minimal differences derive from the presence, in the network graph, of multiple equivalent shortest-paths, that are taken into account by `NetworkX` but ignored by `babeld`. This latter, in fact, selects a single next-hop for each destination. However, the sum of all LC values is the same for both empirical and theoretical values: this means that `babeld` and `NetworkX` always select shortest-paths of the same length (in number of hops), and confirms the intuition that the observed differences are only due to the different splitting of contributions performed with `NetworkX`.

3.3.2.2 Error Analysis in Partial Deployments

This section is devoted to the formulation and the analysis of the error that afflicts the upgraded nodes (running Algorithm 3) while estimating the LC in an only partial deployment.

[17] More on ninux, am Italian wireless community network: `http://ninux.org/FrontPage`

- Let $\mathcal{H} \subseteq \mathcal{V}$ be the set of upgraded nodes of a network-graph $G(\mathcal{V}, \mathcal{E})$ that, considering a partial deployment, is necessarily a subset of the overall set of nodes in the network;
- Let $\mathcal{W} = \mathcal{V} \setminus \mathcal{H}$ be instead the set of legacy nodes, complementary to $\mathcal{H}$.

A node generates load contributions destined to all other nodes $v \in \mathcal{V}$ only if it belongs to the $\mathcal{H}$ set, while a node $w \in \mathcal{W}$ does not. An upgraded node $h \in \mathcal{H}$ also i) aggregates and splits load contributions as illustrated in Algorithm 3 and ii) estimates its own centrality index and collects the estimates of all other upgraded nodes thanks to the dissemination process. Legacy nodes only perform the silent propagation of centrality subTLVs as described with Algorithm 2 just passing load contributions to their next-hops when sending UPDATE messages. However, they do not generate nor collect any load contribution so that the overall load circulating in the network will be less than the load the would be diffused in a full-deployment scenario. For this reason upgraded nodes can only underestimate their LC indexes. This underestimate computed by upgraded nodes is defined as:

Def. 3.4: ***Partial Load Centrality***

$$LC'(h \in \mathcal{H}) = \sum_{h \in \mathcal{H}} \sum_{d \in \mathcal{V}} \theta_{h,d}(h) \tag{3.1}$$

Equation (3.1) differs from the original definition of Load Centrality (Equation (2.2)) by the fact that the summation of contributions is not over *all* possible shortest-paths between any source and destination $s, d \in \mathcal{V}$, but just over those paths that go from upgraded nodes ($h \in \mathcal{H}$, i.e., the only that are sources of load) towards all other nodes. In the rest of this section the error introduced by the Partial Load Centrality (Equation (3.1)) w.r.t. the correct Load Centrality (Equation (2.2)) is formulated and studied analytically.

Error Function Definition

Let $H = |\mathcal{H}|$ be the number of upgraded nodes. Then, the definitions of Average Normalized Load (or Partial) Centrality are given:

Def. 3.5: ***Average Normalized Load Centrality***

$$\overset{\triangle}{LC} = \frac{1}{N} \sum_{v \in \mathcal{V}} \overline{LC}(v) = \frac{1}{N} \sum_{v \in \mathcal{V}} \frac{2\,LC(v)}{N(N-1)} = \frac{2 \sum_{v \in \mathcal{V}} LC(v)}{N^2(N-1)} \tag{3.2}$$

Def. 3.6: ***Average Normalized Partial Centrality***

$$\overset{\triangle}{LC'} = \frac{1}{H} \sum_{h \in \mathcal{H}} \overline{LC'}(h) = \frac{1}{H} \sum_{h \in \mathcal{H}} \frac{2\,LC'(h)}{H(N-1)} = \frac{2 \sum_{h \in \mathcal{H}} LC'(h)}{H^2(N-1)} \tag{3.3}$$

In Equations (3.2) and (3.3) averages are computed over the number of nodes that are sources of load contributions, thus over N in Equation (3.2) and over H in Equation (3.3). Moreover, the normalization coefficient that indicates the *number of pairs of endpoints* involved in the load diffusion process changes accordingly. Provided these definitions, the error is formulated as the (normalized) deviation of the partial centrality from the load centrality:

Def. 3.7: ***Average Normalized Relative Error***

$$E_{\mathcal{H}} = \frac{\overset{\triangle}{LC} - \overset{\triangle}{LC'}}{\overset{\triangle}{LC}} \tag{3.4}$$

Error Characterization

The relative error $E_{\mathcal{H}}$, defined by Equation (3.4), will be analytically characterized rewriting it so to highlight the amount of "*missing load contributions*": missing because not all nodes, but only those belonging to the $\mathcal{H}$ subset, generate and accumulate load during the diffusion process. This reformulation focused on the amount of lost contributions eases the the study of the dynamic of $E_{\mathcal{H}}$ converging to zero for H approaching N, i.e, while deploying always more upgraded nodes.

To define the amount of lost contributions it is first necessary to compute the expected overall load circulating in the ideal full deployment scenario, when all nodes are upgraded. The overall network load requires, to be well defined, the auxiliary definition of *path length* formalized as:

Def. 3.8: ***Path length*** The length of a path $p(s,d)$ is equal to $|[s,\ldots,d]|-1$

- Multiple shortest paths for the same endpoints s and d have the same path length (by assumption of being shortest);
- For s directly connected to d, i.e., when $p(s,d)=[s,d]$, the path length is 0.

This definition of length is given as it is measures well the number of nodes *crossed* in a path. This number is important in relation with the definition of LC (Definition 2.2), which states that load contributions are accumulated only by "crossed-by nodes", but not by the endpoints of a path. The overall network load can be now computed via Theorem 2:

Theorem 2: *Overall Network Load*

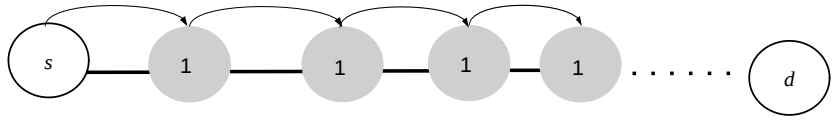
$$\sum_{v\in\mathcal{V}} LC(v) = N(N-1)\bar{l} \tag{3.5}$$

where

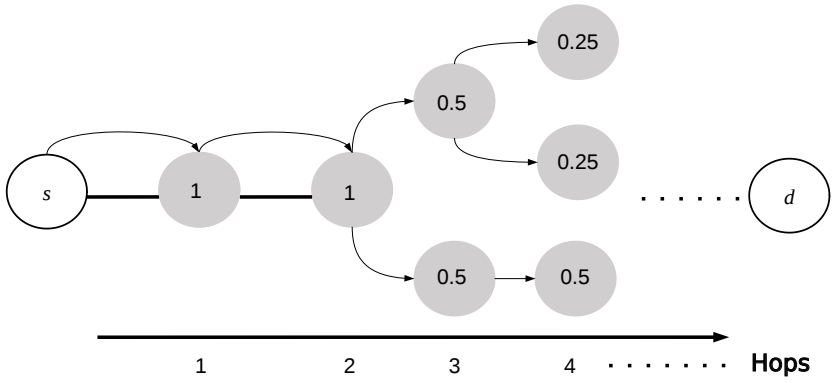
- $N(N-1)$ is the number of $(s,d) \in \mathcal{V} \times \mathcal{V}$ pairs in $G(\mathcal{V},\mathcal{E})$ with $s \neq d$;
- and $\bar{l}$ is the average shortest-paths length in the network graph $G(\mathcal{V},\mathcal{E})$.

Proof. Let's initially focus on Figure 3.7a, that shows how a load contribution is propagated over a single path from s to d. This figure demonstrates how the load contribution originated by s increases by 1 the load centrality of all nodes crossed in the path $p(s,d)$. This means that, for each pair of endpoints (s,d), the load contribution originated by s will induce an overall increment of the load centrality in the network that is equal to $|[s,\ldots,d]|-1$. By definition of path length (Definition 3.8), this increment is equal to the length of the shortest path $p(s,d)$.

This holds also if a contribution is splitted over multiple shortest paths, as illustrated by Figure 3.7b. This claim is supported by an immediate observation. Let $k = |\{u_k : u_k \in \bigcup_{s,d} p(s,d) \wedge |[s,\ldots,u_k]| = i > 0\}|$ be the number of those nodes u_k belonging to some of the multiple shortest paths from s to d that are at distance $i > 0$ from s. The sum of the fractional contributions owned by all the k nodes must add up to 1 because, by algorithm construction, at all branching points at distances $< i$ the contribution has been equally splitted. A trivial induction over i proves that this holds for any value of i. Hence, as for Figure 3.7a, the load increment in the network induced by a pair of endpoints is equal to the length of the (multiple) shortest paths also in case of splitting.



(a) A load contribution crossing grey nodes while being propagated over a unique $p(s,d)$ path.



(b) Nodes at the same distance from s are aligned in the same column: the sum over the nodes belonging to the same column is always equal to 1.

Figure 3.7 – Examples of the centrality variations induced by a load contribution sent by a source s towards a destination d. *Copyright © 2020, IEEE.*

This observation is valid for all the $N(N-1)$ possible pairs of endpoints (s,d). Introducing the notion of average path length $\bar{l}$, the sum of the lengths of all shortest paths (i.e., the *overall load in the network*) can be succinctly rewritten as reported in Theorem 2. *QED*

Substituting the overall network load (Equation (3.5)) in Equation (3.2), the Average Normalized Load Centrality can be rewritten as:

$$\overset{\triangle}{LC} = \frac{2\sum_{v\in\mathcal{V}} LC(v)}{N^2(N-1)} = \frac{2\,N(N-1)\,\bar{l}}{N^2(N-1)} = \frac{2\bar{l}}{N} \tag{3.6}$$

Theorem 2 enables the computation of the overall network load in a full deployment. A similar theorem is necessary to compute the overall load estimated in a partial deployment, i.e., to compute $\sum_{h\in\mathcal{H}} LC'(h)$. The approach to build this desired theorem will be similar to the one adopted in the proof for Theorem 2, where two key notions introduced were:

1. The average path length $\bar{l}$;
2. The number of different pairs of endpoints $s,d \in \mathcal{V}\times\mathcal{V}$ with $s\neq d$, equal to $N(N-1)$, that indicates how many load contributions are generated by source nodes.

These two notions must be updated to contemplate a partial deployment scenario in which only a subset of nodes, namely $\mathcal{H}$, generate and accumulate load contributions and, furthermore, these contributions are propagated not over all shortest paths but only over those that connect upgraded nodes to all others. For these reasons:

1. The average path length $\bar{l}$ used in Theorem 2 will be updated into $\overline{l'}$, that now indicates the average length of those paths of the form $p(h,d)$, i.e., those that connect upgraded nodes $h\in\mathcal{H}$ with any other node $d\in\mathcal{V}$.
2. Given that legacy nodes, by legacy model, do not accumulate load, then the contributions that cross nodes $w\in\mathcal{W}$ do not contribute to the overall estimated load and must be discarded. Rather, only contributions crossing the $h\in\mathcal{H}$ nodes should be summed up.

Taking into account these two considerations, the *Overall Estimated Load* is defined as:

Theorem 3: *Overall Estimated Load*

$$\sum_{h\in\mathcal{H}} LC'(h) = H(N-1)\overline{l'} - \sum_{w\in\mathcal{W}} LC'(w) \tag{3.7}$$

The Average Normalized Partial Centrality (Equation (3.3)) can now be rewritten thanks to Equation (3.7):

$$\overset{\triangle}{LC'} = 2\,\frac{H(N-1)\overline{l'} - \sum_{w\in\mathcal{W}} LC'(h)}{H^2(N-1)} \tag{3.8}$$

The conclusive reformulation of the relative error is achieved applying both Theorems 2 and 3 to Equation (3.4):

$$
\begin{aligned}
E_{\mathcal{H}} = \frac{\overset{\triangle}{LC} - \overset{\triangle}{LC'}}{\overset{\triangle}{LC}} &= \left(\frac{2\bar{l}}{N} - 2\,\frac{H(N-1)\overline{l'} - \sum_{w\in\mathcal{W}} LC'(h)}{H^2(N-1)} \right) \div \frac{2\bar{l}}{N} \\
&= 1 - \left(\cancel{2}\,\frac{H(N-1)\overline{l'} - \sum_{w\in\mathcal{W}} LC'(h)}{H^2(N-1)} \right) \times \frac{N}{\cancel{2}\bar{l}} \\
&= 1 - \frac{N}{\bar{l}} \left(\frac{\cancel{H(N-1)}\,\overline{l'}}{\cancel{H^2(N-1)}} - \frac{\sum_{w\in\mathcal{W}} LC'(h)}{H^2(N-1)} \right) \\
&= 1 - \frac{N}{\bar{l}} \left(\underbrace{\frac{\overline{l'}}{H}}_{E_1} - \underbrace{\frac{\sum_{w\in\mathcal{W}} LC'(w)}{H^2(N-1)}}_{E_2} \right)
\end{aligned}
\tag{3.9}
$$

- E_2 is the term that weights the amount of contributions not accumulated by legacy nodes since they do not recognize centrality subTLVs. This term grows (decreases) with the number of legacy (upgraded) nodes: when E_2 grows the error $E_{\mathcal{H}}$ gets larger.
- E_1 decreases deploying more upgraded nodes, but E_1 has an opposite sign compared to E_2 and, therefore, when E_1 grows the error decreases.

3.3.2.3 Error function analysis

The interpretation of the error $E_{\mathcal{H}}$ is based on the analysis of the corner cases for the error term E_2. Setting E_2 to zero models a network where legacy nodes do not generate load contributions and, furthermore, they are chosen among the graph nodes that have zero centrality. Setting $E_2 = 0$ makes the error function becomes:

$$
E_{\mathcal{H}} = 1 - \frac{N}{H}\frac{\overline{l'}}{\bar{l}} \tag{3.10}
$$

Equation (3.10) is relevant since it models the approximation error of betweenness centrality estimation techinques based on shortest-paths sampling starting from a set of selected pivots [98]. This is a popular approach, widely adopted to mitigate the scalability issues faced by centralized algorithms when the number of nodes grows up to billions of nodes [103]. The sample average paths length $\overline{l'}$ quickly converges to $\bar{l}$ if pivot-nodes are chosen at random, in this case the error is dominated by N/H. However, Equation (3.10) does not actually model any potential partial deployment of upgraded nodes protocol. In fact, in the other works all the nodes belonging to some sampled path contribute to the estimation of centrality. In the present context of a partial deployment scenario, instead, the potentially sampled

legacy nodes would not generate nor estimate their centrality. This is why the contributions of $w \in \mathcal{W}$ nodes are subtracted in Equation (3.7).

How big can be E_2 in any proper ($\mathcal{W} \neq \emptyset$) partial deployment scenario? Two are the possible further corner cases:

1. $\mathcal{W}$ includes all and only the nodes with *zero* centrality;

2. $\mathcal{W}$ includes all and only the nodes with with *non-zero* centrality.

In the first case $\sum_{w \in \mathcal{W}} LC'(w) = 0 \Rightarrow E_2 = 0$. This is the case that leads to the already discussed Equation (3.10). In this benign case, the fact that legacy nodes are not measuring their load centrality does not increase the error. In the second case, all centrality contributions propagated in the network cross only $w \in \mathcal{W}$ nodes, that together would account for the overall load circulating in the network, which is equal to $H(N-1)\overline{l'}$. If so, then $E_1 = E_2 \Rightarrow E_{\mathcal{H}} = 1$; when this happen, for any sampling of shortest-paths the relative average error will remain equal to 1. Consider, for example, a star graph: if $\mathcal{H}$ does not include the center of the star then the relative error will always be 1 irrespective of the size of $\mathcal{H}$.

Except for the first corner case, in a proper partial deployment ($\mathcal{W} \neq \emptyset$) some contributions will be necessarily lost, so in general $E_2 > 0$. The second corner case analysed the maximum value that E_2 can reach: it turned out that E_2 can grows up to be equal to E_1 thus $E_2 \leq E_1$. The combination of these two observations provides the bounds for $E_{\mathcal{H}}$:

$$1 - \frac{N}{H}\frac{\overline{l'}}{\overline{l}} < E_{\mathcal{H}} \leq 1 \tag{3.11}$$

Theorem 4:
$E_{\mathcal{H}}$ decreases upgrading more nodes.

Proof. $E_{\mathcal{H}}$ decreases when the $E_1 - E_2$ difference grows: showing that this difference is always positive and that its trend is always increasing while upgrading more nodes will prove Theorem 4. To show that $E_1 - E_2 > 0$ it is sufficient the last observation that concluded the analysis of the corner cases for E_2:

$$E_2 \leq E_1 = E_1 \geq E_2 \Rightarrow E_1 - E_2 \geq 0 \tag{3.12}$$

Remember that E_2 is maximum when $\mathcal{W}$ contains all and only the nodes crossed by contributions, that in this malign case get all lost, but these lost contributions are recovered moving $w \in \mathcal{W}$ nodes to the $\mathcal{H}$ set. This upgrading procedure necessarily reduces the E_2 term, amplifying the $E_1 - E_2$ difference, that therefore exhibits an increasing trend. *QED*

To have $E_1 \geq E_2$ is fundamental, as it means that upgrading more nodes is beneficial and reduces the average estimation error. However, the convergence to zero of the relative

normalized error is slow. In fact, even in the wishful case where $E_2 \simeq 0$ and the average sample path legth approximates well the true average ($\overline{l'} \approx \bar{l} \Rightarrow \overline{l'}/\bar{l} \approx 1$), then $E_{\mathcal{H}} \simeq 1 - \frac{N}{H}$, and for $\mathcal{H}$ covering the 80% of all nodes the error is approximately equal to $E_{\mathcal{H}} \approx 1 - 100/80 = -0.25$. Therefore, despite the high penetration ratio of upgraded nodes, the underestimate is still affected by a 25% error.

Concluding remarks

The error analysis concluded that upgrading more nodes is beneficial to reduce the estimation error but, unfortunately, the convergence to the correct computation of LC indexes is slow and requires a very high penetration ratio of upgraded nodes in the network. Anyhow, for most applications of centrality, the centrality is exploited just as an indicator of the relative importance of nodes respect to the others. The estimation error is therefore less relevant compared to centrality rankings, which are instead crucial and convey the truly desired information. The focus of Section 3.3.2.4 is thus shifted from the estimation error to the ranking preservation properties of the incremental algorithm. Through the analysis of rank correlation coefficients it will be shown that, even for very small fractions of upgraded nodes, the nodes ranking produced by the incremental algorithm are highly correlated with those produced by an ideal computation of LC in a full deployment.

3.3.2.4 Preservation of the Ranking in Partial Deployments

Rank Correlation Coefficient

The Spearman's rank correlation coefficient r_s [120] measures the correlation between two rankings. Formally, the Spearman coefficient of two variables X, Y is defined as the Pearson correlation coefficient of their rankings, where a ranking is an order relationship for the observations of a variable. The Spearman r_s ranges in $[-1, 1]$:

- $r_s = 1$ if there is a monotonic and increasing association between observations of X and those of Y;
- $r_s = -1$ if, instead, an inverse decreasing association exists.

The Spearman coefficient tends to zero if there is no clear monotonic relationship between X and Y. Table 3.3 reports the imaginary marks and ranks of several students in different disciplines, also providing a few examples of the possible r_s values.

Measuring the Rank Correlation

The legacy and upgraded nodes models (Algorithms 2 and 3) have been implemented in a Python DES. This simulator enabled the computation of rankings based on the Partial

	Math	Rank	**Science**	Rank	**Literature**	Rank	**Arts**	Rank
Alice	10	1	9	1	3	4	4	4
Bob	9	2	8	2	5	3	6	3
Charlie	5	3	6	3	8	2	8	2
Dave	4	4	4	4	10	1	9	1

$r_s(Math, Science) = 1$; $r_s(Math, Arts) = -1$.

Table 3.3 – The table reports the marks of 4 imaginary students in 4 different disciplines. The best students in Math result to still be the best also in Science, while they have worse marks compared to students that are good in Literature and Arts. These observed trends are well captured by the Spearman's rank correlation coefficient r_s.

Load Centrality Equation (3.1). These rankings, computed for different penetration ratio of upgraded nodes in different kind of networks, have been compared with the exact centrality ranking computed with the general Algorithm 1. The different network-graphs have been generated using 4 well known generators: the Waxman [121] and the Caveman [122] random-graphs have been added to the already cited Barabási-Albert and Erdős models (references in Section 3.3.1.2). The evolution of r_s while increasing the penetration ratio of upgraded nodes (from 10% to 100%) has been studied with results reported in Figure 3.8.

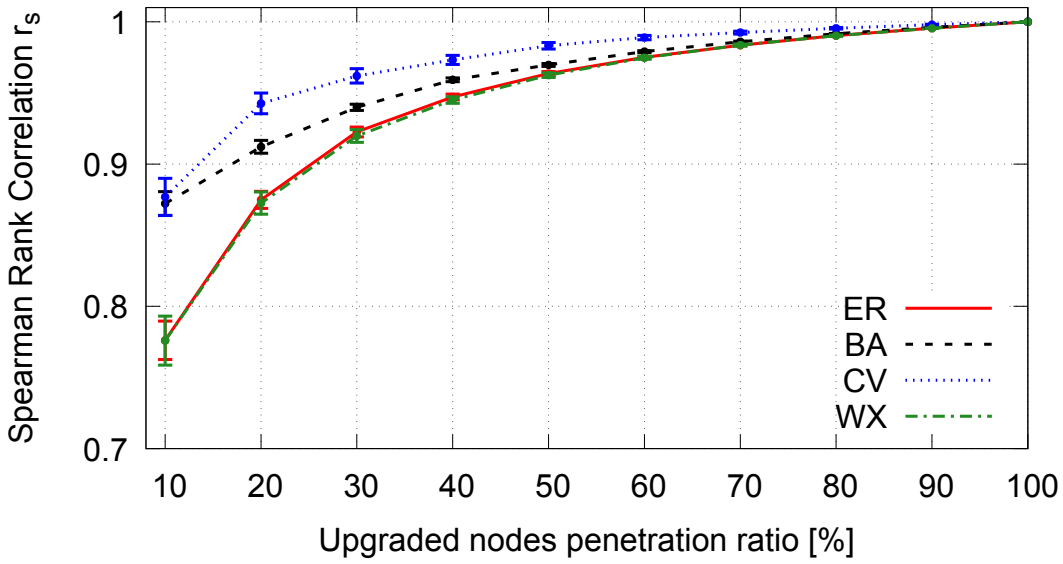


Figure 3.8 – Spearman's r_s for Barabási-Albert, Erdős, Waxman, and Caveman graphs for 1000 nodes and diameter 5 with 95% confidence intervals. *Copyright © 2020, IEEE.*

These results are the averaged outcomes of 50 repeated random experiments: 5 random repetitions for 10 similar random graphs each with 1000 nodes and diameter 5. Each experiment provided a measure of r_s according to this methodology:

- The exact centrality ranking of the nodes has been computed modeling a full deployment, thus setting $\mathcal{H} = \mathcal{V}$;
- For the given penetration ratio x, a random $\mathcal{H}$ subset covering that x% of $\mathcal{V}$ has been chosen, performing then a simulation. Every experiment provides therefore an estimation of the LC and an estimated ranking for the elements of $\mathcal{H}$ as well;
- Finally, the Spearman coefficient has been computed between the exact and the estimated rankings (limited to the nodes in $\mathcal{H}$).

Figure 3.8 reports only the results for graphs with 1000 nodes and diameter 5, but similar trends have been observed also for different diameter values and for smaller graphs with 400 nodes. It can be noticed that the correlation r_s exceeds 0.9 when the penetration ratio grows above 30%, and this results remains valid across all the studied random graph families. The correlation r_s is always above 0.75 even at the minimum coverage, i.e., when $\mathcal{H}$ covers only the 10% of the nodes. For greater coverages (above 50%), the estimated ranking is almost perfectly correlated with the exact ranking, with r_s always very close to 1.

Concluding remarks: On top of the results shown in Figure 3.8 it can be concluded that the incrementally deployable algorithm provides accurate centrality rankings even at low penetration ratio. This result makes the algorithm a good candidate tool to steer the centrality-based optimization of network protocols, and works also in partial deployment scenarios.

3.4 Application of the LC algorithm to DV protocols

The applications of centrality metrics in networking are numerous, as reported in the Background Section 2.5. Among these applications, this thesis focuses on the centrality-based optimization of DV routing protocols. This section documents the porting of the already mentioned Pop-Routing optimization technique [13, 14] from LS to DV protocols, a porting that is enabled by the distributed algorithm for the computation of Load Centrality (LC) presented in Sections 3.1 and 3.2.

3.4.1 Porting Pop-Routing to DV protocols

Pop-Routing [13, 14] is an optimization technique that minimizes the losses of traffic consequent to a potential failure in the network, thanks to to the centrality-based tuning of the timers that regulate the generation of control messages. In both LS and DV protocols, HELLO messages are periodically broadcast by routers to implement the neighbor discovery process or to detect the failure of already known neighbors. The timer that governs the sending of

HELLOS is called T_H, while the average overhead per link (if links are all configured with a default value $T_H = t_{\mathrm{H}}$), was defined to be O_H in the original Pop-Routing papers [13, 14]. The performance metric L that measures the losses in a network due to the failure of a node i with betweenness centrality b_i was defined as:

$$L(i) = V_{\mathrm{H}} t_{\mathrm{H}}(i) N(N-1) b_i \tag{3.13}$$

with V_{H} as the protocol parameter that specifies after how many consecutive lost packets a link must be considered broken. The fundamental Pop-Routing equation, the one the dictates the tuning of T_H of each node (while keeping O_H constant) and that enables the minimization of losses (Equation (3.13)), was proven to be:

$$t_{\mathrm{H}}(i) = \frac{\sqrt{d_i}}{\sqrt{b_i}} t_{\mathrm{H}} \frac{\sum_{j=1}^{N} \sqrt{b_j d_j}}{\sum_{j=1}^{N} d_j} \propto \frac{\sqrt{d_i}}{\sqrt{b_i}} \tag{3.14}$$

where d_i is the degree of node i, and t_{H} is the default T_H value that determines the constant value of overhead O_H to be kept. Equation (3.14) essentially says that a node can self-tune its $t_{\mathrm{H}}(i)$ if it knows the d_i and b_i of all other nodes. When nodes collectively perform this tuning, they achieve the optimal distribution of timer values that minimizes the average network disruption in case of failure, withtout increasing the control overhead O_H.

Originally, the Pop-Routing tuning equation was successfully applied to LS protocols, that make available to all nodes the network topological information necessary to solve Equation (3.14). This same equation cannot be applied "as-is" to DV protocols, where nodes do not own the required topological information. However, substituting in Equation (3.14) the Betweenness Centrality (BC) with the LC, and consequently adopting the distributed algorithm for LC in place of the centralized Brandes algorithm, then it is possible to explore and evaluate the effectiveness of Pop-Routing also with DV protocols.

3.4.2 Implementation of the Algortihm for LC in a real DV protocol

Before describing the methodology chosen to evaluate the performance of Pop-Routing with DV protocols and the related results, it is worthwhile to clarify some necessary modifications introduced in `babeld` w.r.t Algorithm 1.

Dissemination of nodes degree

To distribute in the networks all the topological information required to compute Equation (3.14), DV routers should disseminate not only their LC value but also their degree. To drop this requirement the tuning equation Equation (3.14) can be simplified into Equa-

tion (3.15):

$$t_{\mathrm{H}}(i) = \frac{\sqrt{d_i}}{\sqrt{b_i}} K \tag{3.15}$$

by replacing the term $\sum_{j=1}^{N} \sqrt{b_j d_j} / \sum_{j=1}^{N} d_j$ with some constant K. The network losses are minimized even when the tuning of nodes is performed according to Equation (3.15), however, in this case the control overhead O_H will not match precisely with the true degree distribution in the network. This disadvantage is compensated by the simplicity brought by Equation (3.15), in fact, now nodes can perform the self-tuning knowing only their own LC and degree. The results shown later in Section 3.4.3.4 for partial deployment scenarios are obtained adopting the simplified tuning Equation (3.15). In these cases K has been chosen to reproduce the same level of overhead O_H that there were assigning a default $t_{\mathrm{H}} = 1s$ to all T_H of all nodes.

Nodes vs. IP prefix

A graph-theoretic approach has been used so far to describe routing protocols, so that UPDATES have been modeled to announce node-ids as origin/destination of traffic. However, UPDATES of real protocols do not announce router-ids, but rather IP prefixes. Furthermore, a single router may represent more than a destination, as it actually represents all the IP prefixes that it exports in the network.

Babel dictates all UPDATES for any prefix to include a "router-id" tag that indicates which router exported this prefix in the network, and routers propagate this tag when re-announcing a learnt IP prefix. It has been therefore possible to merge the centrality contributions destined to a given node (but virtually to different IP destinations) exploiting the mapping between graph nodes (identified by their router-ids) and the IP prefixes they exported.

Protocol-specific heuristics

Pop-Routing over the customized implementation of Babel has been tested on different networks with topologies that, as physical graphs, contain multiple shortest-paths to connect some pairs of endpoints. In principle, a protocol supporting multipath could identify these multiple, available, shortest paths, and split load contributions over them. However, the chosen version of `babeld` always performs a tie-break and thus selects a single shortest path only. Moreover, `babeld` implements an heuristic to suppress route flapping. This mechanism prevents, for some time, the selection of a newly available path even if it is the minimum.

For these reasons, the experimental LC values computed with `babeld` experiments can slightly diverge from theoretical ones, as already shown and justified in Section 3.3.2.1. Anyhow, these LC indexes are the expression of the real choices that truly influence the

routing and forwarding of packets, thus they are appropriate for the purpose of tuning T_H timers.

3.4.3 Performance evaluation

3.4.3.1 Methodology

The goal of the experimental performance evaluation has been to quantify the resilience gain provided by Pop-Routing. To this end, the losses consequent to a random failure has been measured in the same network configured, initially, with default timer values, and then with values optimized via Pop-Routing. Comparing the network losses in the two scenarios it has been possible to determine whether Pop-Routing is effective in enhancing the recovery process of DV routing protocols.

Experiments have been performed in virtual networks running `babeld`, emulated with Mininet. In each experiment, at time t_f the failure of a selected node k is triggered. Then the routing tables of all nodes are dumped at high frequency, every 0.5 s, until routers finish the recovery process, i.e., until their routing tables do not stabilize again. A routing table dump is a timestamped dictionary RT_i^j: for each destination d, $RT_i^j[d]$ reports the next-hop selected at time j by node i to reach d. This kind of experiment has been repeated triggering the failure of $N_f \leq N$ different selected nodes. The $N - N_f$ excluded nodes are all those that, in case of their failure, would make impossible the re-routing of traffic around them, hampering the computation of the number of "lost-then-recovered" routes. The excluded nodes are therefore all leaf nodes (they have zero-centrality) and also cut-points, i.e., those nodes that once removed would partition the network in disconnected components. At the end of an experiment, the dumped routing tables are grouped by timestamps and recursively navigated in the effort to rebuild all the shortest-paths selected by Babel for any (s,d) pair of endpoints.

At a given instance of time $t_j > t_f$, some paths may be incomplete or still go trough the failed node k. The number of such non-working ("broken") paths, over which the traffic would got lost, is called L_j. An comparison of L_j over time computed, on one hand with plain babel (red curve) and on the other hand with centrality-optimized timers (black curve), is shown in Figure 3.9. In this figure the evolution of L_j is plotted after triggering the failure of a node around time $t = 5$ s. Babel optimized with centrality better reacts to the triggered failure: the reaction time is slightly shorter and, above all, more routes are recovered in less time w.r.t to a plain configuration of the same Babel. This is a successful example where, introducing Pop-Routing based on the distributed computation of LC, a resilience gain is achieved at almost no cost, as the complexity of the LC algorithm is minimal and the control overhead is kept constant.

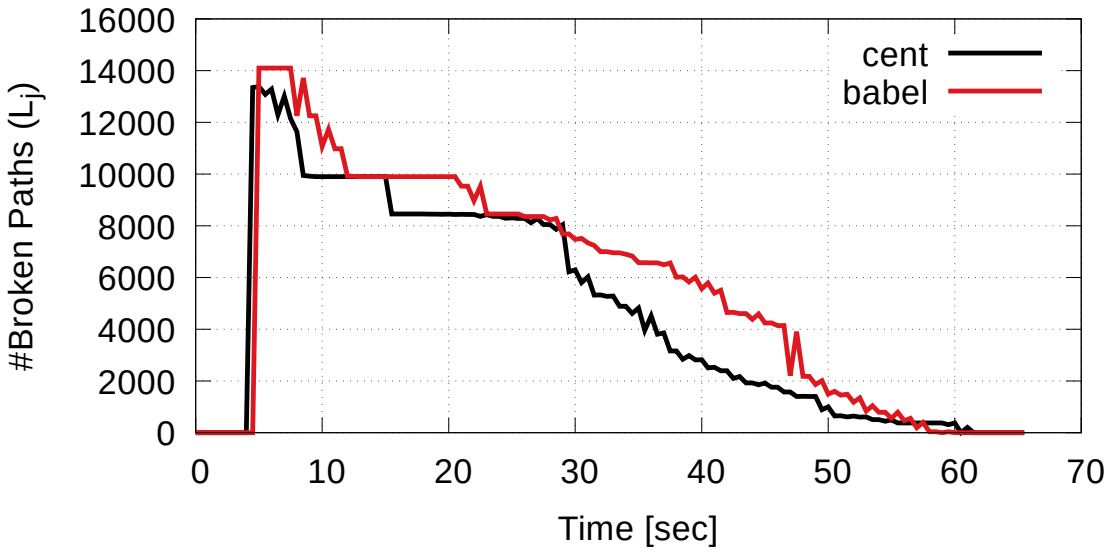


Figure 3.9 – Comparison of the network loss in ninux topology after the failure of one of the most central nodes. *Copyright © 2020, IEEE.*

Loss Reduction metric definition: The total losses over the whole duration of an experiment are defined as $L_{babel}(k) = \sum_j L_j$ when the experiment is performed with the default `babeld`; $L_{cent}(k)$ indicates the same metric but computed using the optimized timers. The *relative loss* averaged over all possible failures is defined as:

$$L = 1 - \frac{\sum_{k=1}^{N_f} L_{cent}(k)}{\sum_{k=1}^{N_f} L_{babel}(k)} \tag{3.16}$$

and indicates if the tuning of timers via Pop-Routing is advantageous or not. If $L > 0$ it means that the experiments with the optimized timers of `babeld`, averaged over N_f failures, lead to less losses w.r.t to experiments with a default timer configuration.

3.4.3.2 Dataset: real world network topologies

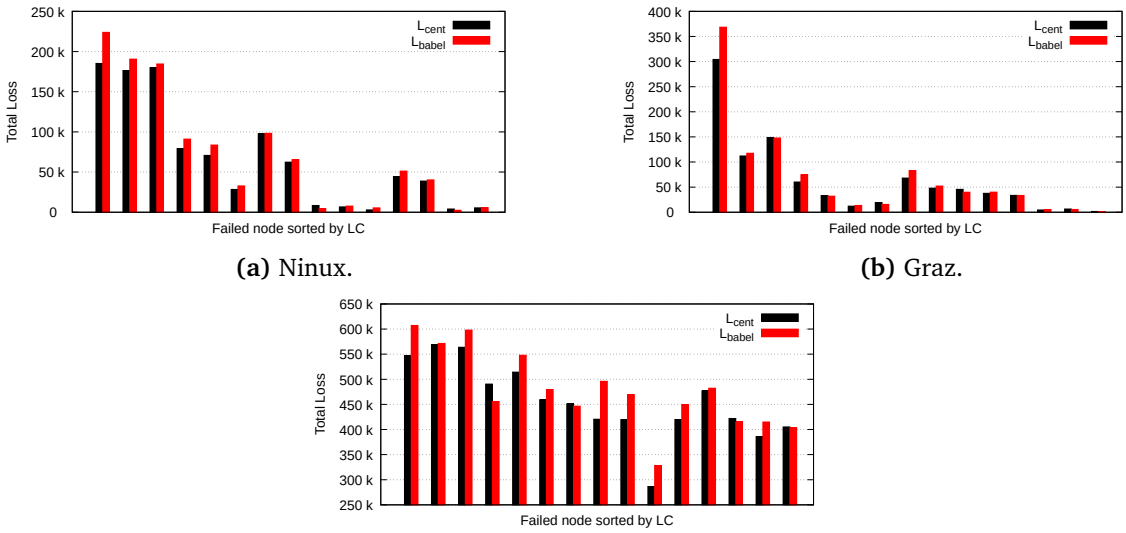
To acquire more exhaustive and realistic results, the methodology described in Section 3.4.3.1 has been applied on several different networks with real-world topologies [8]. Two of them, namely *Ninux* and *FFGraz*,[18] are two large-scale wireless mesh networks, and were included also in the original study of Pop-Routing [13]. Two more real topologies, those of the Community Networks (CNs) of *Auerbach* and *Adorf*, have been extracted from the topology repository of the German CN association "FreiFunk".[19] The FreiFunk CNs are disparate, they change in size —from few nodes up to hundreds— and composition. Some of them

[18]Hompages of the Ninux and FFGraz networks: `https://www.ninux.org`; `https://graz.funkfeuer.at`
[19]Homepage of the FreiFunk initiative `https://freifunk.net/en`

are purely wireless mesh networks while others, such as the studied *Auerbach* and *Adorf*, are heterogeneous networks that mix together wireless and wired links within their own routing domain. Four more wired networks, *Interoute, Ion, GtsCe* and *TataNld*, have been included in the experimental evaluation extracting them from *Topology Zoo* [123], a well known repository of real world networks topologies.

Table 3.4 reports a summary of the key characteristics for all the 8 considered networks, including their experimental loss reduction (Equation (3.16)).

3.4.3.3 Experimental Results in Full Deployments



(a) Ninux. **(b)** Graz.

(c) Ion.

Figure 3.10 – Comparison of the loss induced by the failure of the 15 more central nodes in ninux (a) Graz (b) and Ion (c) when standard Babel is used (L_{babel}) or the tuned version is used (L_{cent}) *Copyright © 2020, IEEE.*

Table 3.4 and Figure 3.10 summarize the experimental evaluation of the reduction of losses achieved by applying Pop-Routing in the 8 considered real networks. In particular, the histograms shown in Figure 3.10 compare the amount of losses of plain babel ($L_{babel}(k)$) with the losses obtained after tuning the control timers ($L_{cent}(k)$). This comparison is performed for the 15 most central nodes of 3 selected networks, namely Ninux (Figure 3.10a), Graz (Figure 3.10b), and Ion (Figure 3.10c), chosen because representative off all the different types of networks. Similar results can be observed also in the other evaluated networks: these results are summarized in Table 3.4 but not illustrated with more histograms for the sake of brevity.

It can be observed that L_{cent} is in general smaller than L_{babel}, even if in few cases tuning the T_H timers does not reduce the amount of losses. It can also be observed that better results are achieved in wireless and heterogeneous networks compared to wired ones. This

happens because of their topology tending to be mesh-alike, while wired ones have a more uniform structure that does not exhibit the same degree of paths redundancy. The success of Pop-Routing, in fact, strongly depends on the array of values of b_i and d_i and, ultimately, increase with a larger availability of alternative paths to route around failures. Consider, for instance, the extreme case of a ring network, or an n-regular network-graph over a torus in general: in such networks no tuning of timers is possible, because all nodes have the same centrality and degree, thus the application of Equation (3.14) would assign the same default timer value to all nodes.

Table 3.4 – Loss reductions in real networks

Network	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	N_f	Loss Reduction	Type
Interoute	110	148	63	8.37%	Wired
Ion	125	146	58	3.10%	Wired
GtsCe	149	193	98	6.05%	Wired
TataNld	145	186	68	7.34%	Wired
Ninux	126	147	17	10.65%	Wireless
FFGraz	141	200	19	13.11%	Wireless
Auerbach	123	223	70	11.29%	Heterogeneous
Adorf	123	225	65	13.27%	Heterogeneous

However, looking at Table 3.4, the reader can be assured that Pop-Routing results to be always effective in reducing the losses when averaging over all possible N_f failures. The global reduction of losses, i.e., the resilience gain, ranges for the 8 evaluated networks between 3% and 13%. Even if sometimes the gain is modest, still in all cases the performance gain is positive. These results confirm the validity of the Pop-Routing strategy also for DV protocols and encourages its wider adoption.

3.4.3.4 Analytical Results in Partial Deployments

The results shown in Section 3.4.3.3 are obtained in emulation experiments that model full deployments. This section investigates over the possible performance gain that can be achieved in partial deployments, i.e., when only a subset of upgraded nodes supports the computation of centrality and the self-tuning of control timers.

A methodology similar to the one described in Section 3.4.3.1 has been used for experiments that model partial deployments with a growing penetration ratio of upgraded nodes $\rho \in [0.1, 0.2, \ldots, 0.9]$. Given the large number of required experiments for this investigation, the Mininet emulator has not been used anymore in favor of the lighter Python DES introduced in Section 3.3.2.4.

Given a network with N nodes and $\rho \in [0.1, 0.2, \ldots, 0.9]$, a uniformly sampled subset of $\lceil \rho N \rceil$ nodes run the Upgraded Algorithm 3, while all other nodes run instead the Legacy

Algorithm 2. At network convergence, the estimated values of centrality were collected from the upgraded nodes. The timers of the upgraded nodes have been tuned according to the collected values of partial centrality, while the remaining legacy nodes have been configured with the default t_{H} timer value. Without running emulations with the so customized timers for `babeld`, the losses produced by the failure of a node have been computed thanks to Equation (3.13). The semantic of the L_{babel} and L_{cent} terms that appear in the relative loss reduction metric L (Equation (3.16)) is therefore slightly changed. Now, in fact, the differences between L_{babel} and L_{cent} are the expression of the tuning strategy as applied only the subset of upgraded nodes. The loss metric is therefore more precisely defined by the new L' as follows:

$$L' = 1 - \frac{\sum_{k=1}^{N_f} \begin{cases} L_{cent}(k) & \text{if k is upgraded} \\ L_{babel}(k) & \text{if k is legacy} \end{cases}}{\sum_{k=1}^{N_f} L_{babel}(k)} \tag{3.17}$$

The semantic difference between L' and L hampers the direct comparison between the analytical results shown in Figure 3.11 with the experimental ones reported Table 3.4. Anyhow, the fast computation of L' thanks to the Python DES enabled the repetition, in reasonable time, of all simulations for 40 times, for ten values of ρ and for all the networks reported in Table 3.4.

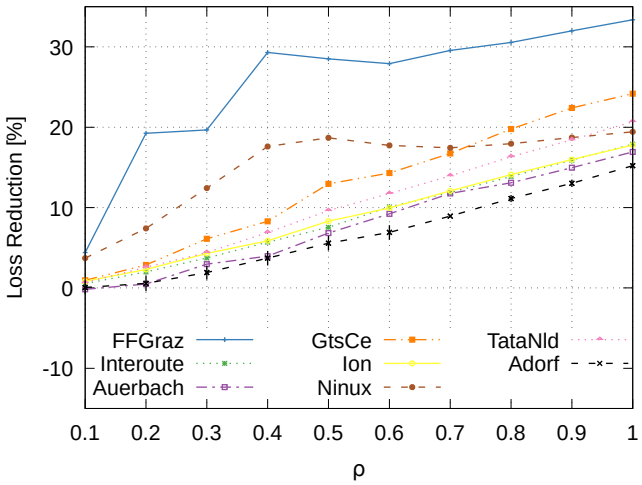


Figure 3.11 – Comparison of the loss reduction metric on various networks with only a fraction ρ of nodes supporting the improved protocol. For clarity, we report the 95% confidence interval for Adorf only, which has the largest ones in average. The intervals are barely visible. *Copyright © 2020, IEEE.*

Some of the curves shown in Figure 3.11 do not increase monotonically. Two are the justifications for this phenomenon. First, the random choice of upgraded nodes hampers

the preservation of the overhead O_H to a fixed constant level. Hence, a first choice of nodes that generate more control messages may show much better performances compared to a following and larger subset that produces a level of overhead non above O_H. This phenomenon is expected to vanish for ρ approaching 1. Second, the centrality distribution over nodes varies over the different networks, so that the random sampling of upgraded nodes produces centrality estimation that converge to the exact ones with different speeds, which depend on the specific graph-properties of each network. For example, for small ρ values, the randomly selected upgraded nodes in Ninux and FFGraz seem to generate an overestimation of centrality, this in turn produces shorter timer values and, ultimately, a higher gain because of a considerably increased control traffic.

Figure 3.11 shows also that a positive loss reduction is robustly achieved in all networks starting from ρ values larger than 0.2. This confirms that, despite the slow convergence of the partial centrality to the correct values of LC, the good ranking preservation property is sufficient to provide a consistent performance gain, even if an only minimal fraction of nodes has been upgraded. It can thus be claimed that the tuning strategy performed by upgraded nodes is truly incrementally deployable, since it is possible to observe better performances even in early deployments, way before completing the upgrade of the whole network.

Chapter 4

Blockchain in Distributed Networks

The blockchain attracted the interest of the research community as it implements a distributed and tamper-proof mechanism for storing data, making true one of the decades dream of computer science: the Shared Ledger (SL). In theory, a SL is a simple data-base cooperatively maintained by different users spread across a distributed system, however, coordinating all these users may become a daunting challenge.

The main purpose of a SL is to register *transactions*, where the word transaction is used in its most abstract meaning and indicates the generic transfer of resources between two or more parties. Usually a SL is thought as a log of economical transactions in any currency, either fiat or crypto, but in general it can record any transfer operation, from the release of a block of IP addresses from the IANA[20] to an Autonomous System (AS), up to the automatic exchange of data in an IoT network.

In general, it is hard to keep a consistent record of transactions when the distributed system scales up to count millions of users. Moreover, considering economical transactions as leading example, one can imagine that keeping a consistent SL may be further complicated by malicious users that cheat or attack the ledger in the effort of deleting those transactions where they spent considerable amounts of money. A *consensus mechanism* becomes necessary to coordinate all the users distributed over the whole network, but also to enforce consistency rules and to discourage tampering attacks.

This chapter presents the fundamental aspects of the blockchain and of consensus protocols to explain how the same blockchain solves the aforementioned problems, and can thus be considered as a Distributed Ledger Technology (DLT).

[20] The Internet Assigned Numbers Authority (IANA) is the organization that oversees the global allocation of IP addressed, autonomous system numbers, the management of root zones in the Domain Name System (DNS), and regulates also the allocation of further Internet Protocol-related symbols and numbers.

4.1 The Blockchain: a Distributed Ledger Technology

Life Cycle of a Transaction

Let's start by considering the role of the blockchain in the general life cycle of a transaction, as illustrated by Figure 4.1. Alice may decide, for instance, to issue a transaction in favour of Bob to purchase a service from him. Alice and Bob may be two persons exchanging cryptomeney, but may also represent two electronic devices exchaning any other kind of resources. For the sake of referecing to a familiar context, economic transactions in cryptocurrencies will be used as leading example most of the time. The transaction details, essentially Alice and Bob addresses plus the transaction amount, must be submitted to the distributed system. Some nodes of the system, in fact, play the role of validators and inspect the ledger to understand whether Alice really owns the amount of cryptomoney she wants to spend. Since the validators are spread across the whole network they need a consensus protocol to decide about the validity of the transaction. If the transaction is considered valid, then it is grouped together with others recently approved ones, forming this way a new *block of transactions*; this block becomes part of the ledger when it is enqueued after the previous blocks, that are chained one after the other building a *block-chain*. At the end of this process a notification is sent to Alice and Bob, informing them about the successful completion of their transaction.

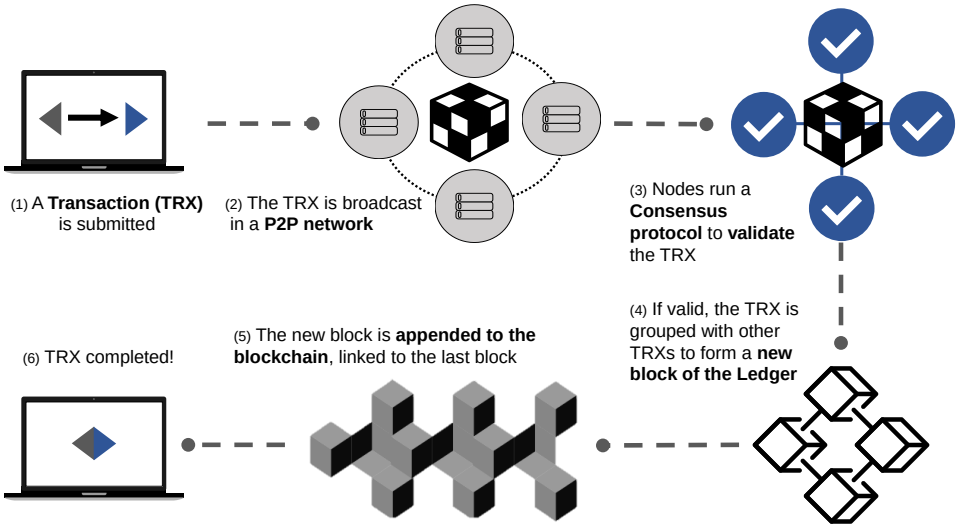


Figure 4.1 – Processing of a transaction before its inclusion in the blockchain. *Copyright © 2021, IEEE.*

Components of a Blockchain system

The blockchain turns out to be a distributed data-base for storing transactions. As a whole distributed system a blockchain includes:

- A *chain data structure*, the mere data-structure to record transactions, organized to capture their chronological order;
- A *Peer-to-Peer (P2P) Network*, i.e., the network of validator nodes. These nodes receive the transactions submitted by the system users and are responsible for their approval or refusal. Their duty is thus to maintain the ledger updated and consistent.
- A *Consensus Protocol*, namely, the rules and policies that dictate when and how new transactions can be appended to the blockchain. The consensus protocol is accepted and implemented by all validators and is essential for the consistency of the ledger.

The chronological order of transactions, captured by the blockchain, enables the decentralized validation of new ones. In fact, the distributed validators determine the validity of new transactions only by verifying their consistency w.r.t. the blockchain.

Users of a Blockchain

There exist two kinds of user (node) in a blockchain system:

1. *Simple users*: these users only *interact* with the blockchain based ledger, e.g., by submitting transactions or by controlling its state to verify the recording of some payment, but do not take part in the process of updating the ledger. Simple users are also known as *simple clients/nodes* and, in some circumstances, are also called *wallets*. Many are, in fact, the online available applications that implement a minimal interface just for sending cryptomoney (paying someone) and for checking the personal balance, recalling the same actions everyone implicitly perform using his real wallet. Still, these simple operations do not directly update the state of the ledger.
2. *Full users*: Full users (nodes) are those that run the consensus protocol to accept or reject new transactions and, accordingly, update the ledger. For this purpose full nodes own a personal, local copy of the ledger, a copy usually not owned by simple nodes that need to query full ones to learn from them the state of the ledger.

Permissionless VS Permissioned

The different ways of user participation and interaction with the ledger determine different categories of blockchains. These differences are the base for the well known dichotomy between *Permissionless* and *Permissioned* blockchains, still, also *hybrid* implementations exist.

1. *Permissionless or Public blockchains:* In a permissionless (public) blockchain there are no restrictions for both simple and full users. The consensus protocol is open to anybody so that even unregistered users, completely anonymous in the system, can become full nodes and start appending new transactions provided only that they comply with the selected consensus protocol. Simple users are not subject to any restriction as well and do not need to be authorized by some full node to start sending and receiving transactions (payments). The consensus protocol must therefore ensure security despite the absence of any form of control on users, that are not accountable as they are anonymous. It turns out that such protocols always impose stringent conditions to be met upon proposing a new transaction; conditions so severe that, if met, somehow prove the honest commitment of the proposer. For example in both Bitcoin and Etherum —the most iconic permissionless blockchains— to propose a new block of transactions the validators must provide the so called Proof of Work (PoW), which is the solution to a very hard cryptographical puzzle. Asking a PoW discourages malicious users and contribute in safeguarding the ledger, however, it also hampers the system performance. The number of Transactions per Second (TPS) processed by Bitcoin and Ethereum is, in fact, below 20 TPS on average: a very limited throughput if compared to the tens of thousands processed by Visa [124]. Moreover, in the Visa platform transactions are recorded sequentially, not in blocks. For this reason the transaction latency, i.e., the interval of time between submission and recording of a transaction, is minimal, just a few seconds, while in permissionless blockchains the same latency averages tens of minutes, sometimes growing up to few hours.

2. *Permissioned or Private blockchains:* Permissioned blockchains improve on performance and allow a fine-grained control on users. This is the kind of blockchain preferred by big enterprises to implement their internal ledger, to be shared among different departments, but also by banks to share the control of an interbank ledger over few and well known business partners. The access to these ledgers is naturally restricted only to registered (accountable) users, the only that own the permissions to update the blockchain. Full nodes are usually operated by business companies which require simple users to be registered and identified for security purposes but also to enforce the mandatory Know-Your-Customers (KYC) regulations. The security of permissioned blockchains depends therefore on classic access-lists and authentication mechanisms but not on the “hardiness” of cryptographic techniques such as the PoW. The resulting trust model, different from the one of permissionless blockchains, allows the blockchain managers to select more efficient, faster, traditional consensus mechanisms in place of the hard consensus mechanisms adopted in permissionless blockchains. The selection of such lighter consensus protocols is a necessity for permissioned blockchains because

they are critical for business operations and cannot tolerate the low transaction rates and high inefficiencies typical of permissionless ledgers.

3. *Hybrid blockchains:* If permissionless blockchains are administratively decentralized while permissioned ones, conversely, are highly centralized, some blockchains have implemented hybrid forms decentralization imposing access restrictions only to *some* users, but not to all of them. For example, in Multichain [125] full nodes are permissioned while simple users are not. Further forms of hybridization do exist. For example, Algorand [126] defines three types of nodes,[21] namely, Participation, Archival and Relay node. Participation and Archival nodes are permissionless and work both as clients and as full nodes, with the difference that Archival nodes keep in memory a longer copy of the blockchain. Relay nodes are where other nodes connect to join the Algorand network and are responsible for the propagation of transactions, blocks and all relevant information for the Algorand protocol. Relay nodes are permissioned and must be registered at the Algorand foundation. Essentially, in Algorand the consensus layer is permissionless while the network one is permissioned.

For the first time, with permissionless blockchains users are free from trusted authorities such as banks or a Public Key Infrastructure (PKI) but can still perform the distributed validation of transactions, with the added novelty that the validation process happens transaprently in public, without uncovering the identity of users. The trust required to maintain a public, permissionless ledger open to anonymous users is given by the consensus protocol only. A permissionless blockchain can thus be considered as a *trust builder in a trustless network* and also the enabler of a *disintermediated marketplace*, since it avoids the involvement of brokers. It is also an *open, privacy-preserving* technology as it allows anonymous users, that in principle do not trust each other, to inspect the history of transactions and thus perform securely new transactions.

4.1.1 History of Transactions and Double Spending

Validators need the history of transactions to determine who owns cryptomoney and how many, an indispensable knowledge to validate new transactions. However, building this history in a distributed system is not easy. Consider for instance Figure 4.2, where two transactions that spend the same coin are received in diverse order by distinct validators, making hard to sort them chronologically. Notice that this problem is general for any distributed system because of the inescapable propagation delays of messages in a network.

[21] More on the Algorand developers portal: `https://developer.algorand.org/docs/run-a-node/setup/types`

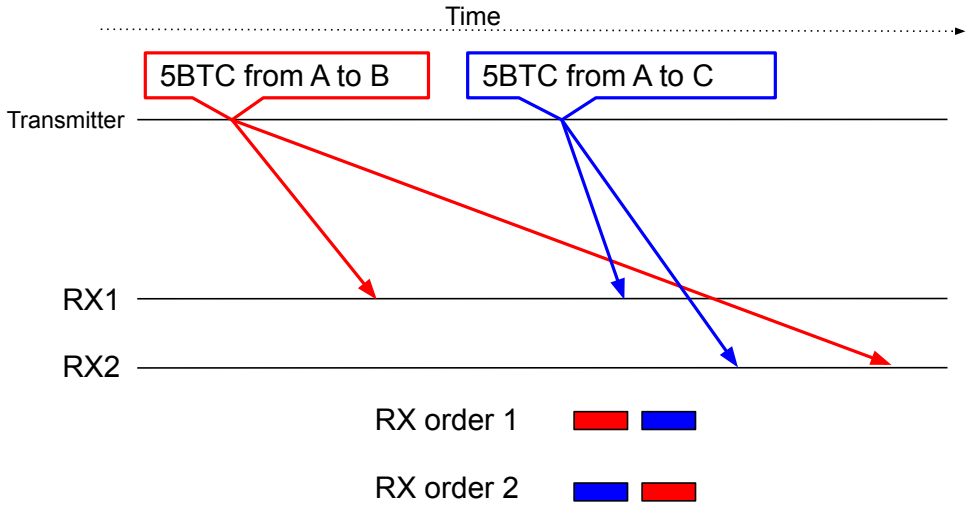


Figure 4.2 – Example of how propagation delays may lead to two different orders of reception at distinct validators. Validators need to run a consensus protocol to find an agreement on the order of transactions. In this example if A owns only 5BTC then one of the two transactions must be rejected because it would represent a double spending.

A malicious user can exploit these delays to spend two times the same money, performing the so called **Double Spending** attack. Validators need to find an agreement on the order of transactions to fend off this attack: they must decide which transaction should be considered the first and which the second, to respectively accept one of the two and reject the other.

The double spending problem raises a *Distributed Consensus Problem* on the order of transactions. The blockchain, as originally proposed for Bitcoin by a mysterious author under the pseudonym of Satoshi Nakamoto [127], is a straightforward solution to this problem: a timestamp should be assigned to each block of transactions, so that the chain of these blocks establishes a (partial but sufficient) chronological order of transactions.

The history of transactions, however, may not be enough for a correct validation, because a malicious user can alter the content of a block to repudiate an unwanted transaction, falsifying this way the validation procedure. To fend off falsification attacks a blockchain must be:

- *Tamper-proof*: i.e., made so that is easy to verify that the registered transactions have not been manipulated after their recording, and it should be likewise easy to determine if these have been actually altered in a second instance of time.

- *Immutable*: a blockchain-based ledger should adequately word off tampering attacks.

The tamper-proof property of blockchains is achieved by a clever embedding of Cryptographic Hash Functions (CHFs) into the blockchain data structure, as explained in Section 4.1.2.

4.1.2 The Blockchain Data Structure

Cryptographic Hash Functions are crucial to make the blocks of the blockchain tamper-proof. The way CHFs are embedded in the blockchain is explained starting from the illustration of the blockchain structure (Figure 4.3).

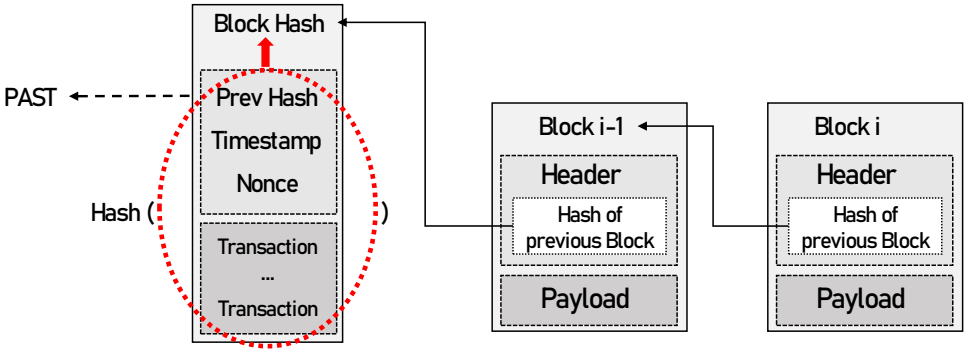


Figure 4.3 – The structure of a blockchain.

In Bitcoin, for example, a block is considered valid only if its *Block Hash*, i.e, the fixed-length digest[22] generated by applying a hash function to the content of the block, exhibits a predefined number of leading zeros. In particular, this digest must be a number lower than a given target, a target that can be changed to adjust the difficulty [128] of finding a valid Block Hash. Finding a valid Block Hash can be indeed extremely difficult considering the random nature of CHFs and the nodes strategy for generating new blocks, which is the following.

At first, a validator groups together a bunch of recent transactions and assign them a timestamp. This timestamp indicates a later point in time compared to the one of the previous block and, to further enforce the time-dependency between blocks, every new block must reference the Block Hash of the previous one. The references to previous blocks constitute the *chaining* of blocks. Having the reference to the previous block and a proper timestamp, a validator guesses a random value (the *nonce*) to be included in the new block, and finally applies the hash function to all these information. The so computed Block Hash is not valid if it is smaller than the target. If so, the node starts a brute-force search and keeps retrying with as many different random nonces as possible, until it finds a valid one.

Notice that, if someone tries to tamper a block (let's say block i) altering its content, it will invalidate its Block Hash. Not only, also block $i+1$ gets invalidated, because its reference to the previous block is changed. Essentially, because of blocks chaining, the tampering of a block invalidates all the following ones. This makes the blockchain a *tamper-proof* technology.

[22] A message digest is a fixed size numeric representation of the contents of a message, as computed by a hash function.

4.1.3 Proof of Work (PoW)

The difficulty of finding a valid nonce can be tuned changing the target value. This difficulty can be expressed as the probability of choosing a random number, smaller than the target, out of the whole range of the hash function. This same probability can be approximated to the one of finding a Block Hash with a given number of leading zeros, a number called Z. Now consider that double-SHA256 is the hash-function dictated by the Bitcoin protocol and produces digests of 256 bits, hence its range has a cardinality equal to 2^{256}. The valid digests are only those with Z zeros as prefix, so their cardinality is 2^{256-Z}. By classic probability definition, the probability $P(n)$ for a random nonce to be valid is therefore:

$$P(n) = \frac{2^{256-Z}}{2^{256}} = \frac{1}{2^Z}. \tag{4.1}$$

To make an example, with the current Bitcoin difficulty that sets $Z = 77$, then $P(n) \approx 6.62 \times 10^{-24}$.

Increasing the difficulty means making nonces always more rare, so that finding them may become an extraordinarily hard computational task which demands a great consumption of energy. If so, a node that finds a valid nonce can show it to the rest of the validators in the P2P network as a *Proof of Work (PoW)*, i.e., as the witness of having worked intensively to complete the search. In PoW-based blockchains nodes need to be encouraged in spending a great amount of computational power to provide a valid PoW, which is a requirement to propose a block. The Bitcoin protocol, for example, reserves a reward to the first node able to propose a new block of transactions that includes a valid nonce. This reward consists in the right of generating new Bitcoins and owning them. The fact that validators are constantly at work, searching for rare valid nonces that will enrich them, makes them earn the metaphorical name of "miners". Miners are asked to provide a PoW for two fundamental reasons:

1. To control the block generation frequency (Section 4.1.4);
2. To secure the blockchain from tampering attacks (Section 4.2).

4.1.4 Block Generation Frequency

The block generation frequency is an important parameter relevant, in the first place, for the consistency of the blockchain, but with important implications also on its security and on the profits of miners. The blockchain consistency is endangered by *forks*, i.e., by the different but almost contemporaneous block proposals from different miners, as illustrated in Figure 4.4.

A fork is problematic because it means that the blockchain does not define anymore a consistent order of transactions, so much that two conflicting transactions may be recorded

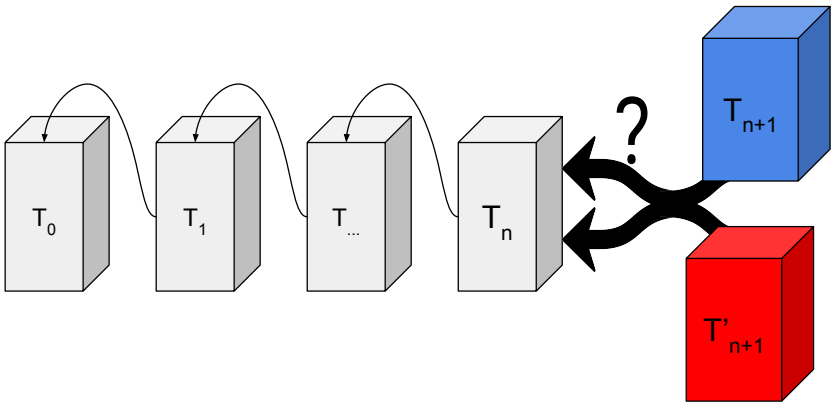


Figure 4.4 – The almost contemporaneous proposal of two distinct blocks by different miners causes a temporary divergence of the blockchain, like the one illustrated here. These temporary divergences are called *forks*.

in the two blockchain branches, re-opening the doors to a double spending attack. A fork occurs when a first block is published in the P2P network by some validator and, during the propagation time of this block announcement, a second block is generated and announced by a second miner. In general, the propagation time for any P2P overlay over an IP network is in the orders of seconds to maximum tens of seconds [129].

To avoid forks as much as possible one can slow down the generation frequency of blocks, distributing over a larger interval of time the probability that two miners independently generate a valid block. In Bitcoin, for example, the mining difficulty is tuned in order to target an average block production rate of 1 block every 10 minute. This interval of time that separates the generation of two consecutive blocks is called Block Generation Interval (B_{GI}) or also, more simply, Block Time. Bitcoin and other PoW-based blockchains target an average Block Time in the order of several minutes, as shown in Figure 4.5.

The PoW difficulty must be tuned to keep the generation frequency close to the desired target because, over time, the ability of miners in finding nonce tends to increase. For example, consider that the rise of the Bitcoin value justified considerable investments in mining-equipment, so much that today the network of Bitcoin miners as a whole is able to compute an astronomical number in the order of 10^{20} hashes per second. Figure 4.6 shows the evolution of the mining difficulty that followed closely the one of the Bitcoin network computing power, for the reason of artificially keeping a constant block production rate.

The Longest Chain rule

Forks can still occur, despite the slowing down of the block production rate, but fortunately they are transient and quickly vanish with high probability thanks to the "*longest-chain*

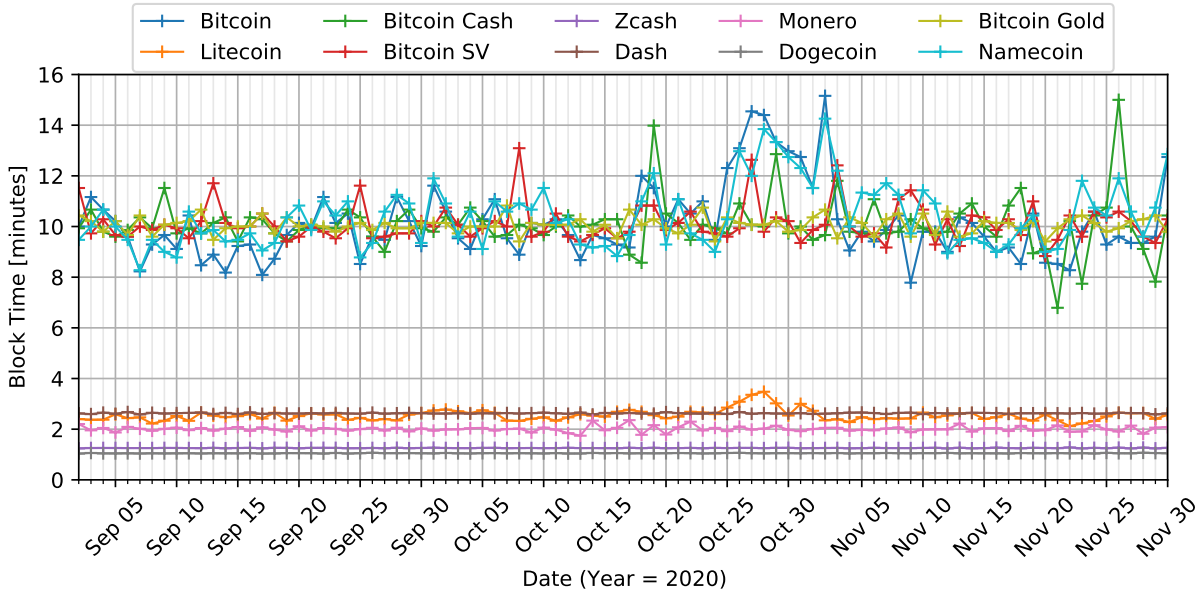


Figure 4.5 – Historical records of the daily average Block Time for many PoW-based blockchains observed over 2 months, namely, October and November 2020. Statistics are taken from `https://bitinfocharts.com`.

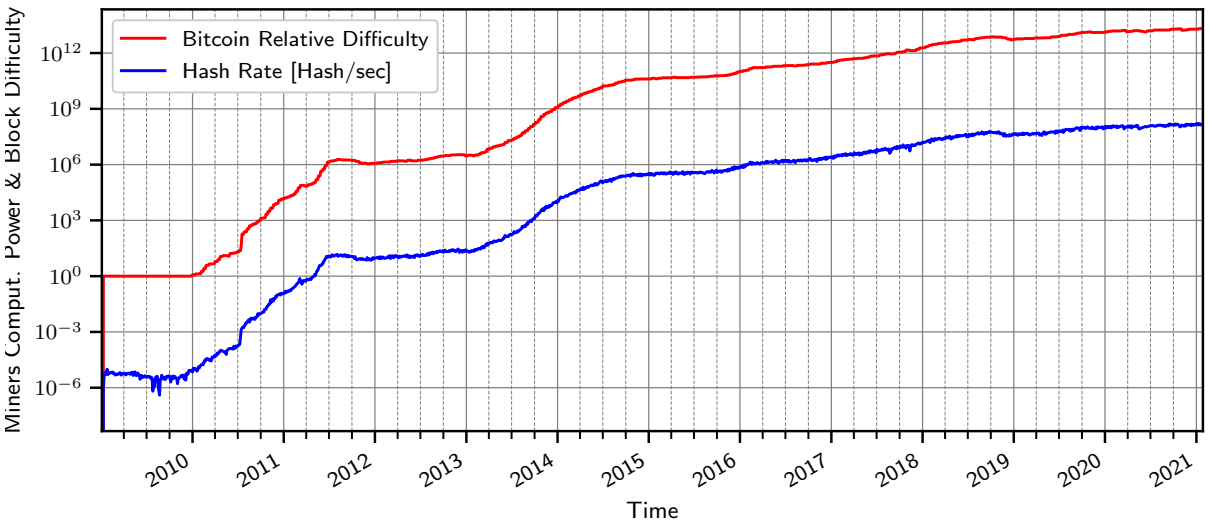


Figure 4.6 – Evolution of the Bitcoin network computing power, measured in hash per second (shown in logarithmic scale). In time the block difficulty has been adjusted to keep a constant average block production rate. Statistics are taken from `blockchain.com`. *Copyright © 2021, IEEE.*

rule" [130, 131]. This rule is inherent to the blockchain technology and states that the only valid blockchain branch is the longest one.

Thanks to the rule a fork is cleared as soon as a block is added to one of the forking branches, making it the longest one. Consider that the contemporaneous proposal of two blocks is made improbable by the hardness of the PoW, so that the probability of having

consecutive independent block proposals is even more unlikely, and this probability decays exponentially fast to zero. The rule also implies that orphan blocks that do not end up to be part of the longest chain are not valid: these blocks are aslo called *stale blocks*. The transactions included in stale blocks, reward as well, will not be considered valid. Miners do not want to lose their rewards working on branches that will not be part of the longest chain, therefore, they abandon a branch as soon as they notice the addition of a block to a longer branch. The rule thus encourages the majority of miners to always work on the longest chain, that therefore stabilizes fast and reliably, as it is grown by the majority of the network computing power.

4.2 Security and Limits of the PoW

The PoW is not only an effective mechanism to control the block generation frequency, as shown in Section 4.1.4, but it is also essential to make blockchains *immutable*, being strategic for securing the SL from falsification attacks.

The double spending attack illustrated by Figure 4.2 in Section 4.1.1 exploits the intrinsic propagation delays of distributed systems to make validators approve two conflicting transactions that spend the same resources. This first kind of double spending attack is almost completely fended off by the high difficulty of the PoW, that slows down the block generation frequency making unlikely the simultaneous submission of conflicting transactions. However, an extremely powerful attacker can leverage on the longest chain rule to perform a different kind of double spending. It can submit a double spending transaction in a new and longer branch built independently by the same attacker. This longer branch will be accepted by the miners, earning the attacker the chance to replace an unwanted transaction with a double-spending one as illustrated in Figure 4.7.

The reminder of this section reviews in chronological order the main studies that influenced the research community in the analysis of the blockchain security. These studies start from the very first approximate mathematical model provided by the same Nakamoto to calculate the success probability of double-spending attacks, a model later advanced by Rosenfeld [132], Sapirshtein et al. [133] and by Gervais et al. [134]. The review analyses also the *"selfish-mining"* strategy, a malicious strategy first described by Eyal and Sirer [135] with the potential of decreasing the amount of computing power sufficient to run a successful double-spending attack, thus a relevant strategy for the study of the blockchain security. The selfish mining strategy is reported in this thesis because it is crucial to understand the security of blockchains, however, recent economic [136] and computer-science [137–139] studies concluded that such a strategy is not economical for real world, rational investors in blockchain mining technologies, thus the theoretically devastating selfish mining strategy

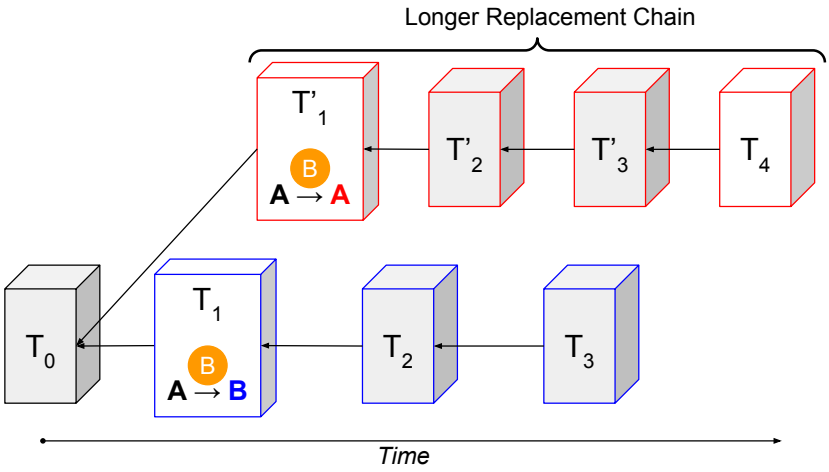


Figure 4.7 – Alice sent a considerable amount of Bitcoin to Bob and now wants to revoke her transaction. This transaction was included in the block T_1 of the blue chain, appended to T_0. If Alice is fast enough to generate a longer chain that forks from T_0, then she can replace the $A \rightarrow B$ transaction with an $A \rightarrow A$ one, or simply do not include anymore the $A \rightarrow B$ in any block, thus gaining herself the chance of spending the same money twice.

turns out to be practically irrelevant. Reviewing this studies including economic considerations is necessary to highlight the exceptional security which is the ground for claiming the *immutability* of PoW-based blockchains.

4.2.1 Nakamoto Model of Double-Spending

Nakamoto imagined the chain-replacement attack illustrated by Figure 4.7 already in the original paper that revealed Bitcoin in 2008 [127]. He considers the scenario of an attacker trying to independently build a replacement chain faster than the honest chain, and models the race between the honest and the attacker chains for becoming the longest one as a Binomial Random Walk. In such modeled stochastic process, the honest chain advances by 1 step every time the honest group of miners is able to find a block, the attacker chain grows similarly when the attacker mines a block. Being p and $q = 1 - p$ the computing power ratios respectively controlled by the honest network and by the attacker, the growth of the honest or of the attacker chain have, respectively, probability p or q to happen.

Now remind the role of confirmations in Bitcoin [140]: a merchant is supposed to wait some time before shipping or delivering the just sold product or service, more precisely, he waits a given number of blocks to be appended after the block that included the paying transaction. Consider also that the merchant generates a new payment address and sends it to the attacker just in time for setting up the transaction, preventing the attacker from precomputing a long replacement chain ahead of time. Essentially, both the honest miners and the attacker start building their alternative chain only once that the transaction to be

attacked is set up and broadcast, an event the marks the start of the race between the two competitors. During the confirmation timeframe both the honest and the attacker chains grow by some blocks; let z be the difference between the lengths of the two chains at the end of the confirmation waiting time. Nakamoto models the probability for an attacker to catch up with the honest chain once that he is left z blocks behind recalling a classic Gambler's Ruin Problem. Doing so, he models the attacker as a gambler with infinite credit that, starting at deficit z, can game the honest chain an infinite number of times while trying to reach breakeven. Calling q_z the probability for an attacker to ever catch up from a disadvantage of z blocks, q_z can be computed as function of p, q and z according to a classic solution for the Gambler's Ruin problem [141] as follows:

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases} \tag{4.2}$$

Interpretation for Equation (4.2)

When $p \leq q$ the attacker owns the majority of the computing power, meaning that the attacker block generation rate is faster than the honest one therefore, in the long run, the attacker chain will grow longer than the honest one regardless of any gambler's deficit (i.e., honest advantage) z. This part of the interpretation for Equation (4.2) is the mathematical foundation for the so called 51% attacks [142]. Notice, however, that "in the long run" translates to "with infinite time available" or, more formally, the Binomial Random Walk defined by the Gambler's Ruin Problem can me mapped to a Markov Decision Process (MDP) with infinite space where z can grow to infinite. Notice that not imposing a bound on z means that the attacker can bear the cost of attacking for infinite time, which models an attacker with infinite credit or, in equivalent terms, it models attacks that cost nothing. This modeling assumption is therefore an oversimplification to ease the mathematical solution of the Gambler's Ruin problem but, in real blockchain system, the cost of an attack is not zero and depends at least on the cost of acquiring sufficient hashrate, that should be the majority of the overall hashrate in the network, and on the duration of the attack. The non-zero but actually extremely high cost of a majority attack to the Bitcoin network will be estimated, for instance, in Section 4.2.2.1. Summing up, the attacker must have infinite time available and infinite capital for Equation (4.2) to hold. As the attacker is modeled with such supernatural powers, the q_z probabilities for running a successful attack should be interpreted as an upper bounds on the true attack probabilities.

When instead $p > q$, the probability for an attacker to "luckily" catch up with the honest (faster) chain decays exponentially with the advantage z of the honest miners. Notice also that the lower is q the faster is this decay.

In conclusion, Nakamoto provided a first quantitative model to approximately study the security of PoW-based blockchains. The main limitations of his model are the infinite time horizons and the infinite attacker credit, virtually modeling a powerful attacker ($q > p$) that attack a blockchain even if the deficit of his alternative chain wrt the honest one approaches infinite ($lim_{z\to\infty}$). In such ideal model the probability of completing a double-spending successfully converges to 1, but in reality a threshold on z should be defined, stating for instance that an attacker waives to attack if his disadvantage z is greater than a given number of blocks. For example, Gervais et al [134] advances the Nakamoto model studying a similar MDP but with finite space, imposing a cutoff value on z equal to 30.

4.2.2 Role of Confirmation

The simplified discussion by Nakamoto about the attacker probabilities of success have been more rigorously studied by Rosenfeld [132], who stresses more on the role of confirmations for securing the blockchain and also formalized more accurately the growth of the z-blocks gap between the honest and the attacker chain during the confirmation time.

The Rosenfeld model is based on these assumptions:

- The total computing power in the network is H and is constant for the whole duration of the attack.
- As for Nakamoto, the attacker owns qH of the total hashrate, while the honest network controls the remaining pH, with $p + q = 1$.
- The mining difficulty is constant such that the average Block Generation Interval B_{GI} is constant and equal to T_0.
- The honest and the attacker chain share the same blocks up to a branching block where a fork begins. The honest miners appends after this block n further blocks, while the attacker appends further m blocks. The gap between the two alternative chains is thus $z = n - m$.

The Rosenfeld model can be represented by a continuous-time Markov Chain (MC) as the one depicted in Figure 4.8, where the honest chain advantage z represents the chain states. Rosenfeld is *interested in whether z will ever become -1, and not when this will happen* [132], thus waives to treat the continuous-time MC illustrated in Figure 4.8 and, dropping the notion of time, falls back to the discrete-time case with probability p for reaching the state

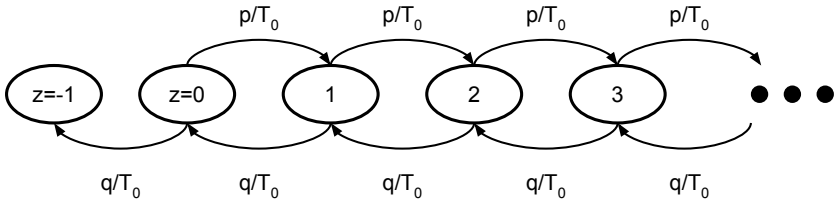


Figure 4.8 – The continuous-time Markov Chain described by Rosenfeld in [132]. The state of the chain is given by $z = n - m$, i.e., the difference between the length of the honest and the attacker chains, initially considered non-negative otherwise the attacker would start with an already longer chain, hence he would have already succeeded in his attack. The MC state increases by 1 when the honest miners finds a block and conversely decreases by 1 if the attacker finds one. According to the Rosenfeld model these two events happen with a rate of, respectively, p/T_0 and q/T_0. The state where $z = -1$ is a final one where the attacker wins the race and can reveal his longer chain. At this point the attack is concluded, this is why there is no transition back to the $z = 0$ state even if the attacker and the honest chains at this stage would be the same, hence with z turning back to 0.

$z + 1$ and q for coming back to $z - 1$, posing a problem which is almost identical to the Nakamoto model based on the Gambler's Ruin Problem.

Compared to Nakamoto, Rosenfeld calls a_z (instead of q_z as in Equation (4.2)) the probability for an attacker left z blocks behind to catch up and succeed in his attack. When $z < 0$, $a_z = 1$ as the attacker already owns a longer chain, so the case of interest is when $z \geq 0$. Under this hypothesis, a_z can be described by the following recurrence relation [132]:

$$a_z = pa_{z+1} + qa_{z-1} \tag{4.3}$$

which means that, once the advantage reached a level of z, then with probability p the honest miners find a block and push the attacker to a state where its success probability will be a_{z+1}, otherwise, with probability q the attacker finds a block reducing its probability of success to a_{z-1}. Equation (4.3) is a Second Order Homogeneous Linear Difference Equation, with $a_0 = 1$ and $p + q = 1$ as bounding conditions, whose solution is [143]:

$$a_z = min(q/p, 1)^{max(z+1,0)} \begin{cases} 1 & \text{if } z < 0 \text{ or } q > p \\ (q/p)^{z+1} & \text{if } z \geq 0 \text{ or } q \leq p \end{cases} \tag{4.4}$$

The interpretation of Equation (4.4) will be omitted as it is almost identical to the one already provided commenting Equation (4.2).

Rosenfeld extends the initial model outlined by Equation (4.4) to highlight the crucial role of confirmations in enhancing the blockchain security. In fact, Equation (4.4) informs us that once q and p are given (in the case of interest when $q \leq p$), then the attacker probability

of success a_z depends only on the initial value of z used to start the gambling game between the attacker and the honest miners, and in turn the entry-value of z depends on how m grows while the honest chain grows up to n. Under the assumption that $q \leq p$, waiting more time by increasing n should in expectation increase the honest advantage $z = n - m$ before engaging the gambling game, supporting the intuition that waiting for more confirmations is an effective strategy to enhance security. To confirm this intuition, Rosenfeld provides a model for m as a negative binomial variable, with the usual p and q as probabilities of success or failure, and computes the probability of having m failures (blocks found by the attacker) until n successes (blocks found by the honest miners) happen. He also assumes that the attacker starts with 1 pre-mined block, so that the probability of having m malicious blocks at the time when the attacked transaction has been confirmed by n subsequent blocks is the following [132]:

$$P(m) = \binom{m+n-1}{m} p^n q^m \tag{4.5}$$

Rosenfeld combines Equation (4.5) with Equation (4.4) to obtain the probability $r(n)$ of a successful attack as a function of the number of required confirmations n [132]:

$$\begin{aligned} r(n) &= \sum_{m=0}^{\infty} P(m) a_{n-m-1} \\ &= \sum_{m=0}^{n-1} \binom{m+n-1}{m} p^n q^m (min(q/p, 1))^{n-m} + \sum_{m=n}^{\infty} \binom{m+n-1}{m} p^n q^m \\ &= \begin{cases} 1 - \sum_{m=0}^{n} \binom{m+n-1}{m} (p^n q^m - p^m q^n) & \text{if } q < p \\ 1 & \text{if } q \geq p \end{cases} \end{aligned} \tag{4.6}$$

Figure 4.9 illustrates Equation (4.6) showing how the attacker probability of success approaches 1 while growing q up to 50% for different values of n. Figure 4.9 let us observe that the famous *"6-confirmations"* rule of Bitcoin implies the acceptance of a $\approx 0.1\%$ risk of being successfully attacked by a malicious user that controls the 10% of the total network hashrate.

Main Observations

1. **51% attack:** The attack probability of success is always 1 owning the hashrate majority.
2. **Probability exponential decay waiting more confirmations:** The attack probability is non-zero also for $q < 50\%$, still, it decays exponentially fast to zero growing the number of confirmations n, and the smaller is the attacker fraction of hashrate q, the faster is the decay to zero.

As noted already interpreting Equation (4.2), this probability study is *asymptotic*, as the malicious user is supposed to have infinite time (and credit) available for performing

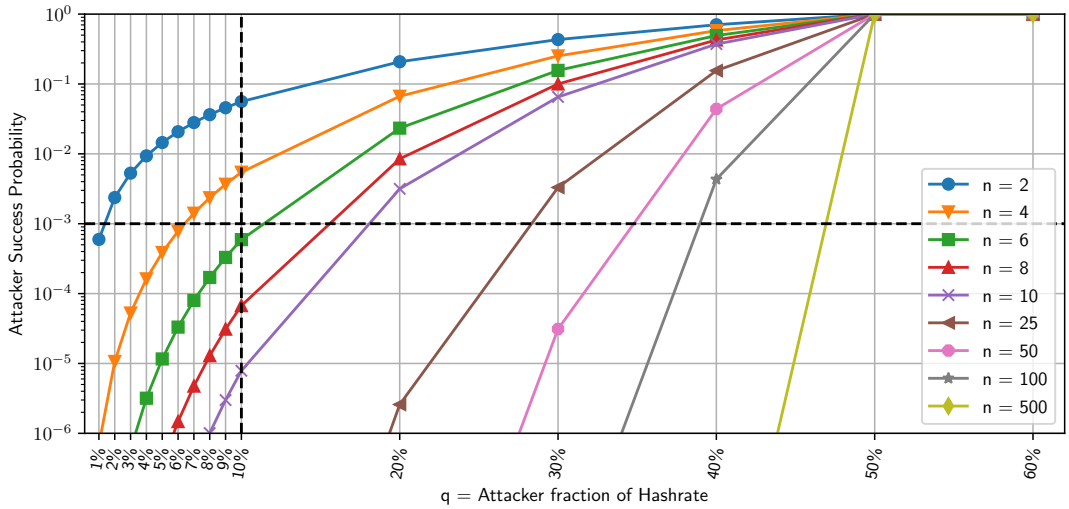


Figure 4.9 – Probability for an attacker to install a longer replacement chain as function of the attacker hashrate ratio q. This probability has been computed according to Equation (4.6) for different values of n.

the attack. Moreover, Rosenfeld dropped the notion of the Block Generation Interval B_{GI} (set to T_0 in Figure 4.8) that plays actually a fundamental role to determine the expected duration of an attack. Sapirshtein et al. further notice that while a single attack may have a non-zero probability of success even when $q \ll p$, regrettably, by the law of large numbers: *"an attacker which continuously executes double spending attempts will eventually succeed. We should therefore be more interested in the cost of an attack than in its success-probability"* [133].

Section 4.2.2.1 sketches a simple framework to:

1. Roughly estimate the cost of on attack per unit of time;
2. Roughly compute the duration of an attack;

providing the elements for a basic approximation of the cost of a complete, successful attack.

4.2.2.1 Estimate of the cost of a 51% attack

Cost of Attack per Time Unit: At the time of writing, a popular hashing power renting facility[23] provides 1 Peta Hash (PH) per Day computing power at the cost of 0.0075 Bitcoin. Converting the Bitcoin price in US dollars[24] and computing the renting cost over minutes instead of days, translates to a cost of ≈ 0.3 \$/PH/min. The total Bitcoin network hashrate[25] NET_{HR} is approximately equal to $NET_{HR} = 166 \times 10^3$PH/s. The attacker hashrate ATT_{HR} for performing a 51% attack against the aforementioned value of NET_{HR} must be equal

[23] Updated renting prices can be found on `https://www.nicehash.com/pricing`
[24] At the time of writing 1BTC is exchanged for 57916\$
[25] Statistics taken from `https://www.blockchain.com/charts/hash-rate`

to $ATT_{HR} = 51/49 \times NET_{HR} \approx 173 \times 10^3$PH/s, which converted to minutes for matching the renting cost per minute computed above gives $ATT_{HR} \approx 104 \times 10^5$PH/min[26]. Renting ATT_{HR} hashrate for a minute costs therefore $104 \times 10^5 \times 0.3 \approx 3.12 \times 10^6$\$, i.e., more than 3 millions US dollars.

Attack Duration: The honest network block generation speed is conventionally $v_h = 1/B_{GI}$, so the attacker speed is $v_a = 51/49\, v_h$. Given an attacker disadvantage z, we can compute how much time the attacker will need to catch up the honest network using simple equations from the physics of the rectilinear and uniform motion. We impose the attacker reached the same "position" (chain length) of the honest chain, but this latter starts running with a $t_0 = z \times B_{GI}$ advantage. Solving for time t we get the duration of a 51% attack as a function of B_{GI} and z:

$$\begin{aligned} x_{\text{Attacker closes z-gap}} = v_h(t + zB_{GI}) &= v_a t \\ \frac{1}{B_{GI}}(t + zB_{GI}) &= \frac{51}{49}\frac{1}{B_{GI}}t \\ t\,(1 - 51/49) &= -zB_{GI} \\ t &= \frac{zB_{GI}}{(51/49 - 1)} \end{aligned} \tag{4.7}$$

Figure 4.10 shows how the time t for performing a successful attack increases while growing the B_{GI}, and displays a different curve for each tested different value of the z-gap. Looking at Figure 4.10 where $B_{GI} = 10$ minutes as for Bitcoin, a 51% attacks takes 245, 1470 and 2450 minutes for closing a gap of, respectively, 1, 6 or 10 blocks. Multiplying this duration for the previously computed cost for renting *AR* hashrate gives us, taking $z = 1$ as example, a cost equal to $2450 \times 3.12 \times 10^6 = 7.644 \times 10^9$\$, i.e., a cost greater than 7 billions dollars. Depending on the value of Bitcoin, which is well known to be very volatile, mounting a 51% attack may become profitable. Figure 4.10 alone, not paired with a chart of the Bitcoin-to-dollars exchange price over time, cannot therefore inform us about the economic profitability of a 51% attack. Nonetheless, Figure 4.10 enables this noteworthy observation:

Observation 4.1. *Fixing a total network hashrate, the cost of an attack grows linearly with the Block Generation Interval (B_{GI}).*

In light of Section 4.2.2.1 a security expert comparing two blockchains Blk_1 and Blk_2 that differ only for their block generation interval B_{GI}, e.g., $B_{GI}(Blk_1) < B_{GI}(Blk_2)$, will conclude

[26] The 51/49 fraction is reported in this simple calculations for the sake of providing an example that immediately and visually leads to think to a majority attack while, obviously, any infinitesimal amount of hashrate greater than NET_{HR} is theoretically enough. Consider also that an attacker wishing to conclude his attack in shorter time may decide to rent even more computational power but, again, for the sake of simplicity the analysis of the cost of the attack is sketched only for $ATT_{HR} = 51/49 NET_{HR}$.

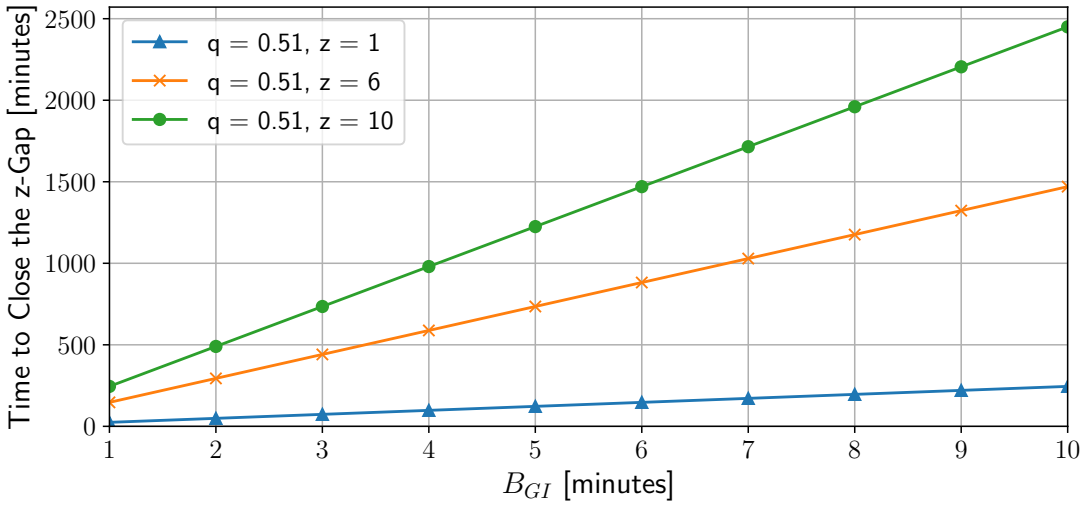


Figure 4.10 – The time necessary to complete a 51% attack is computed as a function of the B_{GI}. Each curve represents a different attack where the honest network starts with a z number of blocks of advantage. This is why the curve for $z = 10$ (the greater chosen value for z), compels the attacker for longer time compared to the case where $z = 1$.

that Blk_2 (the slowest one in the present case) is more secure. The capital required to successfully mount an attack against Blk_2 will be in fact higher than the one required to attack Blk_1, so Blk_2 is more secure as it is more protected than Blk_1 from attacks run by actors with smaller capitals.

Notice, however, that Section 4.2.2.1 cannot lead to the naive conclusion that any blockchain with a high B_{GI} is more secure than others with smaller B_{GI} periods. In fact:

Observation 4.2. *Fixing a B_{GI} and a relative mining difficulty level, the cost of an attack grows with the network total hashrate.*

According to Section 4.2.2.1 this thesis aims to confirm the intuition that mounting a 51% against an extremely powerful blockchain such as Bitcoin, backed up by an incredible amount of mining power (this amount is estimated in Section 4.2.5), is harder than attacking a novel blockchain supported by few and relatively weak miners. It is obvious, in fact, that amassing the 51% of the computing resources when the total hashrate is in the order of $HR = 166 \times 10^3$PH/s requires greater investments than acquiring the same 51% of computing resources when the total hashrate is much smaller, e.g., when it is in the order of tens of TeraHash per second like with Ethereum or tens of MegaHash per second as with Monero.[27] For this reason, 51% attacks have been never reported for Bitcoin, which is backed up by an extremely powerful mining networks, while they occur more likely on smaller blockchains.

[27] The website `2miners.com` reports the statistics about the total hashrate of many mining networks, including the one of Ethereum and Monero.
`https://2miners.com/it/eth-mining-pool` `https://2miners.com/it/xmr-mining-pool`

For example, *Bitcoin-Gold* is known to have suffered 51% attacks twice:[28] the first time in May 2018 lead to an estimated 18 millions USD overspent and the removal of the same Bitcoin Gold from the online Exchange Bittrex. The second time in January 2020 with two almost consecutive chain replacements that resulted in over 70k USD worth of BitcoinGold being double spent. It is believed that mounting this last attack had a cost similar to the amount of recouped block rewards, leading to break-even, with the attacker that gained a real profit thanks to the successful double-spending transactions included in the replacement chains. The reader should be aware that Bitcoin-Gold does not use SHA256 as Bitcoin, but relies on a different algorithm known as Equihash which is designed to be memory-hard. Even if SHA256 and Equihash are not directly comparable, Bitcoin-Gold can be assumed to be a much "smaller" network than the one of Bitcoin, with the cost of the memory necessary to mount a 51% attack that is much inferior to the same cost for renting the necessary hashrate for attacking Bitcoin, so Bitcoin-Gold is more vulnerable.

4.2.3 Selfish Mining

So far it has been implicitly assumed that the ratio of blocks produced by a given miner is in expectation equal to the ratio of computing resources owned by this same miner. The revenue of miners depends on the blocks they are able to produce so, keeping the just stated assumption, the reward assignment mechanism of Bitcoin appears to be fair, as miners share revenues in proportion to their mining power. Furthermore, Section 4.2.2.1 suggests that 51% attack are unlikely to happen at least in well-established blockchains. According to these arguments Bitcoin is believed to be truly incentive-compatible, meaning that profits for miners are positive when playing honestly and negative when deviating from the protocol.

However, Eyal and Sirer [135] revealed a malicious strategy called *"selfish-mining"* where a colluding minority of miners can obtain more than what fair incentive would reward them. In other terms, selfish miners owning the q% of the network computing power can generate more than q% of the blockchain (confirmed/final) blocks adopting the malicious strategy. If Eyal and Sirer theory were confirmed, then selfish-mining would represent a dominant strategy for rational (yet honest) miners: these latter would rapidly join the initial malicious minority forming an always growing pool of selfish miners. This process would lead to the collapse of the blockchain decentralization because the majority of the computing power would end in the hands of selfish-miners. Furthermore, a group of selfish miner may become able to mount a 51% attack even if owning only a minority of the overall computing resources, so for them a 51% attack might be profitable.

[28] Online articles report news and cost analysis about the 51% attack suffered by Bitcoin-Gold `https://cointelegraph.com/news/bitcoin-gold-blockchain-hit-by-51-attack-leading-to-70k-double-spend`

The selfish mining strategy is crucial to understand the security of blockchains even if recent economic [136] and computer-science [137–139] studies have shown that it is actually not economical, explaining why this strategy has not been observed in the real-world Bitcoin network. This section is devoted, in the first place, to the description of the selfish-mining strategy, and in the second place to its downplay explaining why despite its devastating potential it actually poses no threat to the Bitcoin protocol [136].

4.2.3.1 Selfish Mining Algorithm

The selfish mining strategy has been initially envisioned by Eyal and Sirer [135],[29] then its theoretical description has been advanced and further studied also by Sapirshtein et al. [133] and by Gervais et al. [134]. Albeit some variations have been introduced, the core strategy of selfish mining remained the same: a selfish miner secretly withholds freshly generated blocks, delaying their announcement in the network, to *selfishly* accumulate a time advantage in the mining process. This strategy is sometimes also called "secret" or "stealth" mining, as long as a selfish miner keeps secret his private chain of withheld blocks, revealing it only when the length of the public chain approaches the length of his secret chain. The core of the selfish mining strategy will be illustrated now with the help of Figure 4.11.

1. **Create a secret branch:** A selfish miner plays honestly, like all other honest miners, and mines on top of the latest block he is aware of, except when he generates a new block as shown in the left side of Figure 4.11. At this point he switches to the selfish-mining strategy, i.e., he starts withholding his blocks of advantage and also tries to secretly further extend his private chain.

2. **Exploiting advantage:** In the meantime the honest miners will keep mining on the public chain, which will sooner or later become longer as honest miners own the majority of the computing power. Essentially the selfish miner is trying to disperse the honest miners power for as long as possible, creating a time advantage that he uses to strengthen his leading position as shown in the middle section of Figure 4.11.

3. **Publish in time, successful attack:** The selfish miner keeps extending his private chain and waits the public chain to be one block behind before finally revealing his secret longer branch. According to the longest chain rule, honest miners will accept his branch as part of the main chain and all their blocks generated while under attack are pruned, becoming stale blocks. In this case the selfish miner has been able to collect all possible block rewards at the expense of the honest miners, achieving a block

[29] The original paper from Eyal and Sirer [135] has been later revisioned and re-edited by the same authors, finally being republished in 2018 [144].

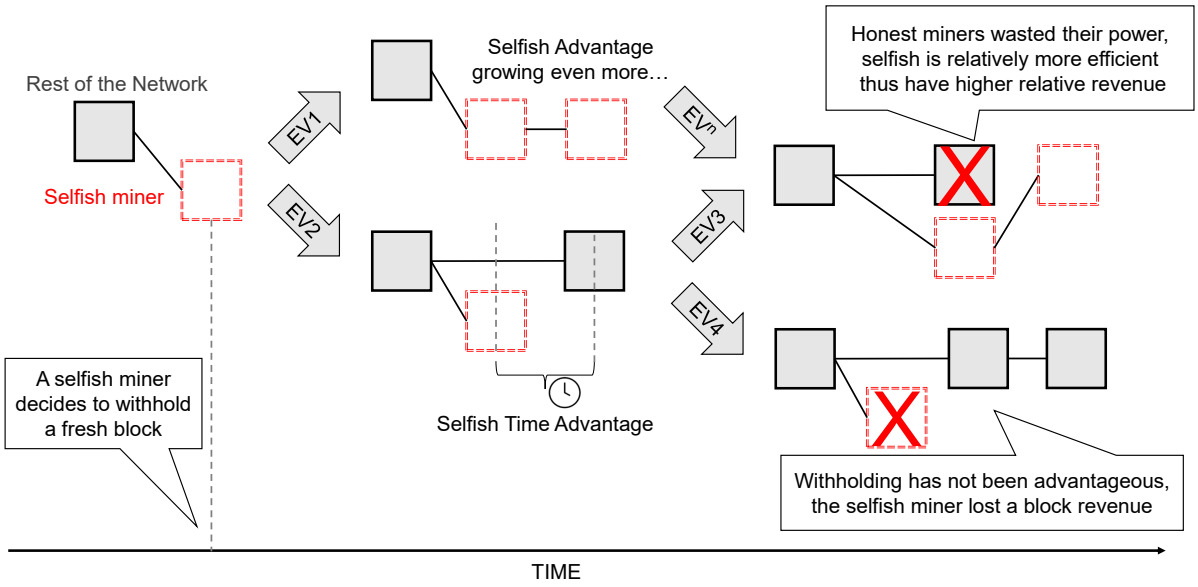


Figure 4.11 – Time sequence of the actions taken by a miner that adopts the selfish-mining strategy. In this figure time flows from the left to the right and possible events are represented by thick arrows. A miner becomes selfish when he is the first in mining a new block (dashed red block). A lucky selfish miner is able to append further blocks to his secret chain, keeping for long time an advantage over the honest chain. The case reached via event-1 (EV^1) is the general case where the selfish miner adds more than one block, EV^2 is the particular case where it adds just one block before the honest miners add a block to the public chain as well. In both cases the selfish miner, withholding the block, is trying to exploit a time advantage letting the honest miner unaware of the longest chain in the network, thus forcing them to work on potentially stale branches. The selfish miner is lucky and succeeds if is able to publish a private chain before being caught by the honest miners, as shown after EV^3 (and also after EV^n generalizing for longer chains). However, it can also be the case that the honest miners immediately close the gap and even overtake the selfish miner: in such a case (EV^4), the selfish miner strategy of withholding blocks turns out to be unprofitable.

production rate (thus revenue) that unfairly exceeds its ratio of computing power. This is the case illustrated in the top-right section of Figure 4.11.

4. **Loosing the race, unsuccessful attack:** However, the selfish miner can be unsuccessful as well. The attack can fail if a honest miner publishes a block right after the first withheld block, so that the selfish miner has no longest chain to publish and, instead, his secret chain enters a race with the public one. The selfish miner can get lucky and win the race, if so he immediately publishes his longest chain to collect all block rewards. However the honest miners can also win and the selfish miner, that owns a minority of the computing power, should give up and abandon his secret block. In this case the strategy of keeping a block secret delaying its publication backfires the selfish miner, who loses the block reward he could have gained by honestly and

timely announcing his new block. This case is illustrated in the bottom-right corner of Figure 4.11.

Main Observations on the Selfish Mining Strategy

1. The Eyal and Sirer model of selfish mining assumes the presence of just one pool of selfish miners, but the contemporary presence of more selfish pools competing with each others could compromise the expected relative revenue gains for all of them.

2. In case of race between the selfish and the honest chain, ignoring all network parameters except computing power ratios and assuming that the selfish mining pool is a minority, than the initial withholding of a fresh block is a risk with a higher probability of incurring in a loss than in a positive reward, as the selfish miner is slower and, in expectation, his chain is more likely to be overcome by the honest one.

3. In case of race between the selfish and the honest chain, the ability of the selfish miner to communicate over the network is crucial. If the selfish miner is able to spread his block to the majority of the miner faster than the block produced by the competing honest miner, then he gains higher chances to have his block accepted, because the majority of miners would contribute to the acceptance of his block.

The first two reported observations are the base for the economical critique to the selfish mining strategy [136], further developed in Section 4.2.3.2. The last one instead highlights the crucial role of the network infrastructure supporting any blockchain, discussed in more details in Section 4.2.4. In fact, an attacker able to increase his communication ability exploiting vulnerabilities of the blocks gossip protocol or of the protocol stack building the Internet can increase his probability of success even without acquiring expensive extra hashrate [145–148].

4.2.3.2 Selfish Mining Limitations

While network vulnerabilities remain a point of concern generally valid for all public blockchains, the threat posed by selfish (sometimes also called “secret”) mining has been with time better understood and considered minimal thanks to recent economic [136] and computer-science [137–139] studies that proved the strategy to be unprofitable and also poorly modeled, as it ignores the possibility of having multiple selfish miners interfering with each other.

The interference among multiple selfish miners, in fact, can be intuitively expected to lead only to losses for those selfish miners that own only a little fraction of computing power. For example, one such small selfish miner may keep trying withholding blocks every time he

believes to have been the first in discovering one, learning only later that another and more powerful secret miner was playing the same game and, armed with greater power, has been able to generate a much longer chain that overcome all those of the many smaller selfish miners. These lasts will thus abandon the selfish strategy, leading to perpetual loss of revenues for them, joining back the honest group of miners. Azimy and Ghorbani [139] started from this intuitive argument and formalize selfish mining under a competitive scenario, confirming analytically and in simulation that *with the presence of a more powerful selfish miner, selfish mining actually decreases the revenue of the weaker selfish miners and also helps the stronger selfish miner*, concluding that *first, the weaker selfish miner will suffer from a loss in almost every scenario [...] Second, the stronger selfish miner is not going to gain more revenue in every situation* so that *the wiser choice for the weaker selfish miners is to join the honest miners to be able to neutralize the effect of selfish mining for the strongest selfish miner.* Essentially, the competitiveness push all miners to immediately publish their blocks, so to immediately collect the reward, waiving to selfishly withhold blocks as another more powerful selfish miner can interfere with this strategy. The economist Paul Sztorc commented this phenomena derived by modeling competition among selfish miners with these words: *"you end up right back where you were before"* [149]. Indeed, selfish-miners rapidly understand that timely publishing their blocks —thus turning back to play honestly— is a more profitable strategy for them.

Even if a single powerful selfish miner succeeds in outpacing all competitors, achieving a personal advantage, Liu et al. [138] notice that a *mining attack is easy to be detected [...] from the variety of the stale block rate of miners*. Another symptom of selfish mining is the frequent appearance of two different forking blocks, one from an honest miner and the other published by the selfish miner after having withheld it for some time, published in reaction to the other. Heilman notes that, among the two blocks, the one from selfish miner should have an older timestamp, thus proposes a rule called "Freshness Preferred" [150] which recommends honest miners to break ties among forking blocks opting for the one with the freshest timestamp. This rule might be an effective countermeasure to selfish mining.

The unprofitability under competition and the availability of effective detection mechanisms and countermeasures are actually not the most relevant arguments for the downplay of the selfish mining threat. Wright and Savanah stress that, even in the original model of Eyal and Sirer, for a single selfish miner *"the strategy leads to lower [absolute] profits compared to profits attained with honest mining, and therefore, there is no incentive for rational miners to adopt it"* [136]. In conclusion, economists believe the selfish mining strategy to be an irrational strategy that damages the profits of miners adopting it, this way they explain why it has not been observed so far in the Bitcoin network.

4.2.4 Role of the Network

When Sirer et al. theorized the selfish mining strategy they studied the evolution of a selfish miner (SM) relative revenue as a function of the selfish miner hash power (α) but also as a function of another parameter, called γ, originally defined as *"[the] proportion of honest miners [that] mine on a SM's block during a fork in the network"* [151]. The γ parameter can be interpreted as an indicator of the miner ability to control the communications over the network, which is strategic for increasing the success of selfish mining. For example, looking back to Figure 4.11, one can realize that the expected relative revenue of a SM depends on the probability of entering in a race with the honest chain and, in turn, on the probability of winning this race. This last probability changes according to γ: in fact, if during a fork the SM is able to spread his block quickly, being the first to reach the, e.g., 90% of the other nodes in the network (i.e. $\gamma = 0.9$), than the mining power of this great majority of nodes will contribute to the confirmation of the SM's block, thus to its success. Eyal and Sirer studied the effect of γ over the miner relative revenues to understand for which critical value of γ selfish mining becomes more profitable than honest mining (see Figure 4.12). As a matter of fact they observe that in the most adversarial case with $\gamma = 1$, which models a SM with a complete control over the network, the SM wins every race, always achieving a larger profit compared to honest mining even if its computational power ratio is minimal. In the ideal and opposite case where $\gamma = 0$, modeling a SM that is always slower than the rest of the network in propagating his block, the SM strategy becomes more profitable than honest mining only when the SM accumulates a great ratio of the overall computing power.

Sapirshtein et al. notice that modeling the attacker's communication capabilities only through the γ parameter, albeit being a succinct way to capture the effect of propagation delays on the production of stale blocks it does not capture the impact of the size of such delays. Advancing the Eyal and Sirer model by explicitly model the network latency, Sapirshtein et al. are able to conclude that a blockchain *"will be more vulnerable to selfish mining if delays become more prominent, e.g., in the case of larger blocks"* [133].

Even if the Eyal and Sirer model turned out to be improper for capturing the true rational behavior of miners, also ignoring the existing competition among them and poorly modelling the impact of network delays, it had the merit of showing (in first approximation) the decisive role of the network on the blockchain security. Apostolaki et al. have been the first to analyse, for instance, the important relation between the Border Gateway Protocol (BGP) of the Internet and Bitcoin, revealing how a malicious miner colluding with a network operator can prevent an honest miner from receiving honest blocks and, instead, it will receives (and thus mine on) only blocks generated by the selfish miner. This kind of attack is called *Eclipse Attack* [152] since a node under attack is "eclipsed" and alienated from the honest network, with its communication horizon that gets restricted to a portion of the network

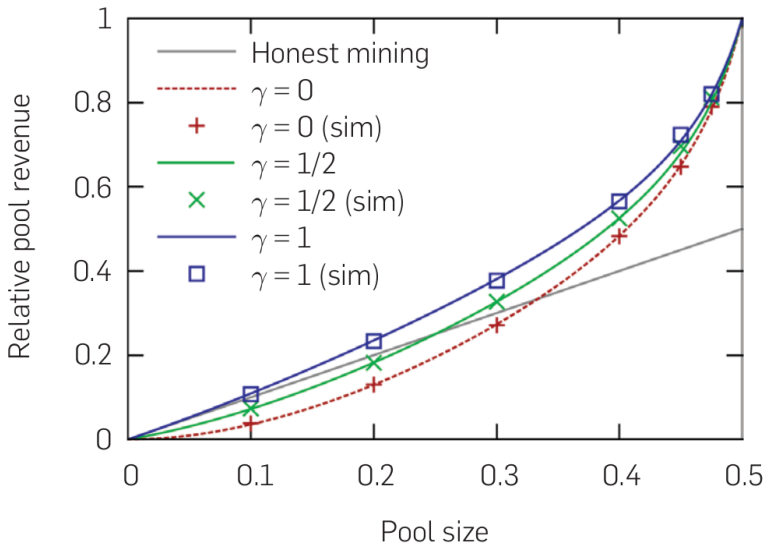


Figure 4.12 – The figure compare the revenues between Selfish and Honest mining for different values of the "network ability" γ parameter, suggesting that selfish-mining becomes profitable as soon as the malicious miner becomes able to control the communications towards a large number of honest miners. This figure was included in the original paper by Eyal et al. [144]. *Copyright © 2018, ACM.*

fully controlled by the attacker. An attacker able to run an Eclipse Attack is virtually able to capture the mining power of its victims, increasing his relative computing power up to the point it might become able to profitably run 51% attacks.

The network vulnerabilities are an important point of concern as they can affect *any* public blockchain distributed over the Internet. A general solution to mitigate these security risks is to run blockchains over permissioned networking infrastructures, thus restricting the possibility of playing the blocks gossip protocol only to *trusted* nodes over authenticated and encrypted end-to-end connections among them. This kind of solution, however, damages the decentralization degree of blockchains.

4.2.4.1 Propagation Time and Stale Rate

Gervais et al. construct a model of PoW-based blockchains which comprehensively take into account consensus and network parameters such as Block Size, Block Generation Interval and Mining difficulty to calculate the resulting Block Propagation Time, Throughput and Stale Rate. This last is proposed as main index to determine the security of a blockchain.

Gervais et al. plug their model of Stale Rate into a security model that comprehensively consider the existence of selfish miners (Section 4.2.3) or eclipse attacks (Section 4.2.4). To the best of this thesis author knowledge, the models of PoW-based blockchains and of their security described and studied by Gervais et al. (their framework is illustrated in Figure 4.13), are the most sophisticated and the only that captures so many of the relevant parameters

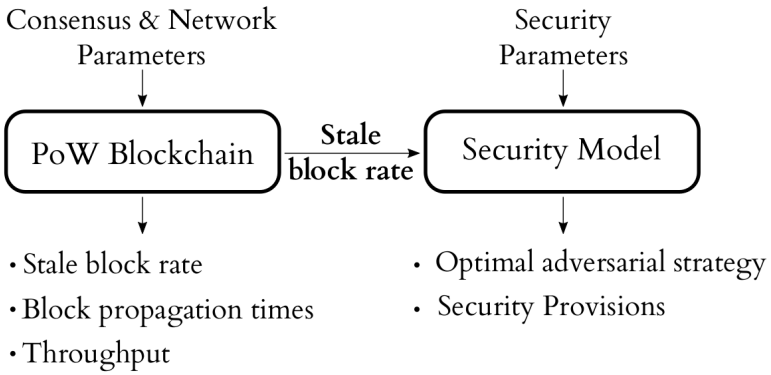


Figure 4.13 – Components of the quantitative framework provided by Gervais et al. to study the security of PoW-based blockchains. The figure is taken from [134]. *Copyright © 2016, ACM.*

for the evaluation of a complex, distributed and financial system like a blockchain. For the need of synthesis, this thesis waives to provide a complete and detailed description of the whole framework. The interested reader is directed to the original paper [134] while this thesis will only report one simplified but key observation about the Stale Rate, which is the following:

Observation 4.3.

$$\frac{\#Stale\ Blocks}{\#Stale\ +Final\ Blocks} \propto \frac{1}{B_P}$$

Observation 4.3 highlights how the Stale Rate, trivially defined as the ratio between the number of stale blocks and the total number of generated blocks, is inversely proportional to the average Block Propagation Time B_P. In fact, a larger B_P determines a larger time interval where multiple miners are not aware of freshly generated blocks: this asynchronicity favors the generation of multiple blocks that —all but one— will become stale. A high B_P contributes therefore to the growth of the network stale rate, thus to the generation of forks that breaks down the honest computing power on multiple branches, ultimately making the network more vulnerable to attacks.

4.2.5 Power consumption of the PoW

Section 4.2.2.1 highlighted how one can increase the security of a blockchain extending the Block Generation Interval (B_{GI}). An extension of the B_{GI} can be immediately achieved increasing the mining difficulty, in other terms, making harder the cryptographical puzzle necessary to produce a valid block. The effort of setting up an exceptionally secure blockchain resulted, in Bitcoin, in a PoW that has become extraordinarily power-hungry [153, 154].

It is possible to estimate the power consumption of the whole network of Bitcoin miners combining their average computing power, measured in terahash per second [TH/s], with

the efficiency of miners, measured in J/TH. The values for these two parameters are reported in Table 4.1.

A lower bound on the power consumption of the whole Bitcoin network can be estimated supposing that all mining pools deploy only the most efficient mining technology. In this case, the **Bitcoin network power consumption** is equal to:

$$1.1 \times 10^8 \frac{\text{TH}}{\text{s}} \cdot 29.5 \frac{\text{J}}{\text{TH}} = 3.245 \times 10^9 \text{J/s} = \mathbf{3.245\,GW} \tag{4.8}$$

This consumption of power is approximately 5600 times greater than the one of an electric car, or 330 times greater than the one of an high-speed train, so large to be comparable with the power consumed by a whole country such as Luxembourg. Figure 4.14 reports the data used to offer this comparison and shows the same data in form of a bar chart.

Table 4.1 – Statistics used to estimate the power consumption of the Bitcoin network.

Parameter	Value	Notes
Mean Computing Power	$\approx 1.1 \times 10^8$ TH/s	Computed over the Daily Total Hash Rates between Oct. 1, 2019 and Oct. 1, 2020. `blockchain.com/charts/hash-rate`
Miners efficiency	29.5 J/TH	At the time of writing, the most efficient rig is the Antminer S19 Pro. `shop.bitmain.com/release/AntminerS19Pro/specification`

	Power Cons. [GW]	Source
Electric Car "Tesla S"	5.80×10^{-4}	`ev-database.org`
High Speed Train *"FrecciaRossa1000" (ITA)*	9.80×10^{-3}	`ferrovie.it`
Luxembourg *elec. peak demand 2014*	1.01×10^{0}	`EU Commission`
Bitcoin	3.25×10^{0}	Equation (4.8)
Italy *elec. peak demand 2019*	5.88×10^{1}	`Terna S.p.a.`

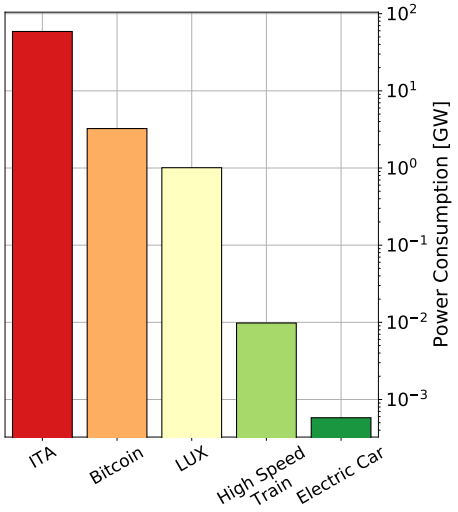


Figure 4.14 – Power consumption of the Bitcoin network compared with the one of cars, trains and even entire countries. The statistic reported for Luxembourg and Italy indicate the electricity peak demand recorded, respectively, in 2014 and 2019.

4.3 Distributed Consensus

In Sections 4.1 and 4.2, the distributed consensus problem on the order of transactions has been solved introducing the PoW. The PoW advantages are many: it is extraordinarily secure, fully distributed, and *user-agnostic*. In fact, users only have to provide some computing power to participate in the protocol, but are not asked to reveal their identity, so much that a PoW-based SL is open to everybody, even to anonymous users. Ultimately, it is thanks to the PoW that blockchains can free users from the need of trusted authorities such as banks, at the same time protecting the users privacy. The popularity of the blockchain, above all with cryptocurrencies, is most probably grounded in these two key aspects. However, the well known limits of the PoW in terms of transaction latency, throughput and power consumption are not tolerable by most networking applications.

It can be observed that all the key properties of a SL, such as consistency, security, tolerance to failures and scalability, mainly derive from the adopted consensus protocol. It is interesting to understand if it is possible to design a consensus protocol that circumvents the limits of the PoW but, nonetheless, enables the implementation of an open, decentralized, secure, blockchain-based SL that also meets the stringent performance requirements of distributed networks.

To answer this question, the rest of this section explores the rich literature on consensus protocols, a literature that dates back to the 70's but flourished again with the rise of interest around the blockchain. In particular, Section 4.3.1 revises fundamental theorems of distributed systems, elaborated decades ago, that are still valid theoretical bounds for the design of consensus protocols in general. A modern transposition of these limiting theorems culminates in the *blockchain trilemma*, discussed in Section 4.3.2, which highlights the trade-offs inherent to the blockchain technology. Consensus protocols are briefly reviewed in Section 4.4, looking for an alternative to the PoW.

4.3.1 Limiting Theorems for Consensus

Distributed systems are usually modeled as *asynchronous systems*, i.e., multi-agent systems where nodes perform some coordinated task communicating via message-passing. These systems are asynchronous because nodes do not share a perfect clock, the propagation time of messages is not fixed, and different kind of failures may occur, contributing to the a-synchronicity of the system. Two kinds of failures can occur in a distributed system, Crash and Byzantine failures. A crash-failure models the simple abrupt crash of a node, that completely stops sending and processing messages, until it recovers. Byzantine failures model any other possible failure, including those due to the attacks of malicious users able, for example, to send corrupted messages, intercept those of others to delay or destroy them,

or also to change their content. The most general formulation of the distributed consensus problem becomes, in so modeled systems, the problem of making all agents of the system reliably agree on the result of a computational process, and this agreement must be reached in bounded-time via message-passing, despite all possible failures.

Fischer, Lynch, and Paterson elaborated, in 1985, the first fundamental theorem of distributed systems, known in the literature as the *FLP impossibility proof* [155]. They concluded that it is impossible to build a consensus protocol that grants to always solve the consensus problem for any fully-asynchronous system. Their proof is based on the intuition that a failure may occur precisely all the times the group of agents is close to reach an agreement, thus preventing the termination of consensus eternally.

The CAP theorem [156] is a second, fundamental pillar of the theory of distributed systems. The CAP acronym stays for *Consistency*, *Availability* and *Partition-Tolerance*, and the theorem states that only two out these three properties can be preserved in a distributed system at the same time. Consider, for instance, the case in which a network of miners gets partitioned in two groups, and a transaction must be approved. The first group cannot contact the second one to verify the validity of the transaction and is left with two options. It can either wait the second group to come back, or it can approve and write the transaction in the ledger. In the first case, the miners sacrifice availability in favor of consistency, temporarily blocking the system. In the opposite case, they accept the risk of writing a transaction that could conflict with another one accepted by the second group. The two cases in which the system is either consistent or available (but always partition-tolerant) have been just discussed, the case left is the one of an always available and consistent system. Trivially, a partition would leads us back to the previous scenarios, this means that an available and consistent system cannot tolerate partitions, proving the theorem.

Both the FLP proof and the CAP theorem contemplate ill-behaving systems, affected by continuous failures or broken in two partitions, so they might be considered only mildly relevant given that computer networks are built to be well-functioning for most of their lifetime. However, they create the base for the definition of the compelling tradeoff between latency and consistency, as formulated by Abadi in the PACELC theorem [157], which builds on CAP (after resorting the acronym into PAC), and further extends it by adding: *"Else Latency or Consistency"*. Abadi observes that: *"Ignoring the consistency / latency tradeoff of replicated systems is a major oversight, as it is present at all times during system operation"*. It is indeed the key tradeoff that determines the security and the performance of a distributed system. In Bitcoin, for example, this tradeoff is controlled by tuning the mining difficulty, and Section 4.1.4 has shown how the 10 minutes average latency per block is fundamental to protect the blockchain from double spending, but also leads to a degradation of the system performance (Section 4.2).

Consistency problems with Distributed Transactions

It can be observed that the problem of consistency is raised only by non-commutative (conflicting) transactions [158], while most of the pairs of transactions are actually commutative, i.e., they can be written in the ledger in any order, and even in different partitions, provided that they spend different resources. A valid solution to enhance the performance of a ledger, without sacrificing its consistency, becomes therefore the execution in parallel of all commutative transactions, blocking the system only for agreeing on the order of the conflicting ones. This idea empowers high-performance permissionless blockchains such as, for example, *Chainspace* and *Omniledger* [159, 160], that define partitions of the system (called *shards*) that can process subsets of local transactions.

This same idea can be interpreted as the application of the Optimistic Concurrency Control (OCC) principle, originally proposed by Kung and Robinson for programming concurrent systems [161], that suggests to perform all transactions in parallel. Assuming that most of transactions do not interfere with each other, then only few conflicts will be recorded, and relatively rare rollback procedures will be enough to restore the ledger consistency. With OCC the transaction throughput improves significantly for low ratios of interference, but is unstable and collapses to zero for higher level of interference, as the systems will spend most of the time in rolling back transactions, instead of making progress. An analogous result, more familiar for the readers expert in wireless networking, is the performance evaluation of the CSMA/CA MAC protocol for wireless LANs [162].

One may want to implement an always consistent and available distributed system by designing atomic transactions [163], i.e., series of read and write operations that either completely occur or none at all. Atomic transactions should grants the consistency of the system by induction: they start from a consistent state, then lead either to another consistent state or leave the system unchanged. However, also atomic transactions are impossible in a general asynchronous system [164, 165], and need synchronization or locks to be implemented [166]. Nevertheless, the properties of modern distributed systems, above all those concerning their degree of decentralization and level of trust, can be accommodated rethinking the properties of atomic transactions, as preliminary explored in [167, 168].

4.3.2 Blockchain Trilemma

The *"blockchain trilemma"* formulates the conjecture that a blockchain system cannot exhibit, at the same time, maximum decentralization, security and scalability (performance). The trilemma is usually illustrated by a triangle, like the one in Figure 4.15, where one cannot draw a single point that is close to all the triangle corners, meaning that a trade-off among the three properties must be chosen [169].

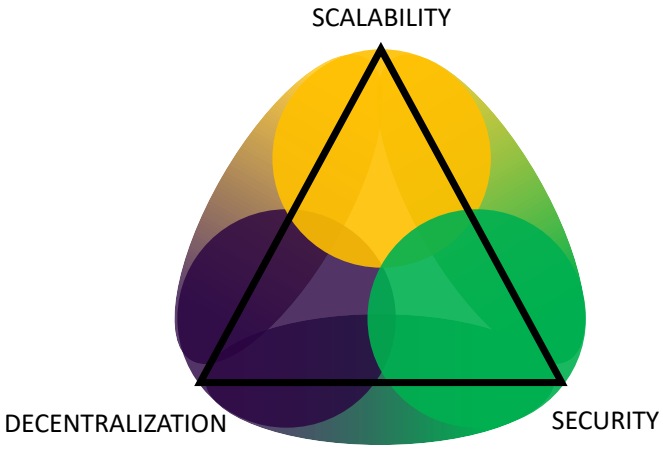


Figure 4.15 – The Blockchain Trilemma pictorially suggests that a trade off among security, decentralization and scalability must be chosen. *Copyright © 2021, IEEE.*

The trilemma can be considered as the reformulation of the PACELC theorem for the blockchain domain: it warns developers that any attempt of improving on scalability, for making a blockchain less power-hungry or more performing, implies a sacrifice in terms of security or decentralization. Consider, for instance, to improve the transaction throughput by reducing the mining difficulty, or by choosing a consensus protocol less expensive than the PoW. Although effective for increasing the block generation rate, less computing power will become enough to attack the blockchain successfully. This reasoning shows that it exists a trade-off between scalability and security. Permissioned blockchains can be seen, instead, as an example of trading decentralization for scalability. In permissioned systems, in fact, the access to the blockchain is restricted only to users that must be registered with a competent central authority. After being authenticated users share a higher level of mutual trust and can thus adopt a lighter consensus protocols, to improve on performance. This cannot be done in permissionless blockchains, that are fully decentralized and remain trustworthy only as long as they employ some cryptographically hard consensus protocol.

The crucial conclusion conveyed by the trilemma is that a trade off must always be chosen, and consensus protocols should be selected to meet the specific requirements of the target application. For example, the PoW results to be effective in supporting cryptocurrencies, where an high level of distributed trust is preferred over performance, for the sake of freeing users from centralized authorities. But the same PoW comes with high levels of power-consumption and poor transaction rates, not tolerable by most networking applications, so that developers need to find alternative consensus mechanisms. Section 4.4 briefly reviews consensus protocols to ease the selection of the most appropriate one, depending on the application requirements.

4.4 Brief Review of Consensus Protocols

There are two main categories of consensus protocols: voting and lottery based protocols, reviewed respectively in Section 4.4.1 and Section 4.4.2. A third category, presented in Section 4.4.3, welcomes those protocols that cannot be classified according to the general dichotomy. Table 4.3, reported while concluding this section, compares the surveyed consensus mechanisms in terms of scalability, security and decentralization.

4.4.1 Voting Protocols

The *Byzantine Generals problem* [170] is the renowned allegory, invented by Leslie Lamport, to describe the challenge of achieving consensus in a distributed system despite any potential failure, including malicious attacks. In this challenge, a group of generals need to take a coordinated action, either attack or retreat, and cast their votes to form a majority in favor of one action. It is hard for generals to take a decision because some generals may be traitors, as much as the messengers that communicate the votes among generals. Betrayals model this way all the possible crash/communication failures of a distributed system.

Leader-based and PBFT variants

The Byzantine Generals problem is in fact equivalent to the one of recording a transaction in a distributed system. This latter is usually solved by introducing a leader node, the "commander" of the system, that initially broadcast the transaction in the network. Each node validates the received transaction by simulating it locally. If the simulation completes successfully it means that the transaction is consistent w.r.t the local copy of the ledger, hence it is valid. The node sends then back an acknowledgment to the leader, suggesting to commit, or otherwise to abort. Acknowledgments can be interpreted as the votes of the Byzantine generals. Once that a majority of votes has been collected for either commit or abort, then the leader broadcast back its final decision to all nodes, these last must update their local copy of the ledger accordingly.

The PBFT protocol [171] is the most famous protocol that implements such a leader-based scheme to solve the consensus problem. The original illustration of the leader-based scheme empowering the PBFT protocol is shown in Figure 4.16. The PBFT protocol tolerates up to f failures in a system with $3f + 1$ nodes, this by requiring a quorum threshold of $2f + 1$ nodes, which grants to have a majority of non-faulty/malicious nodes. The protocol requires 3 rounds of message passing for an overall number of messages equal to $3f + 1$, a bound that Shostak proved to be minimal [172] and Pease proved later to be also necessary and sufficient [173]. There are many other leader-based protocols and PBFT-variants, different

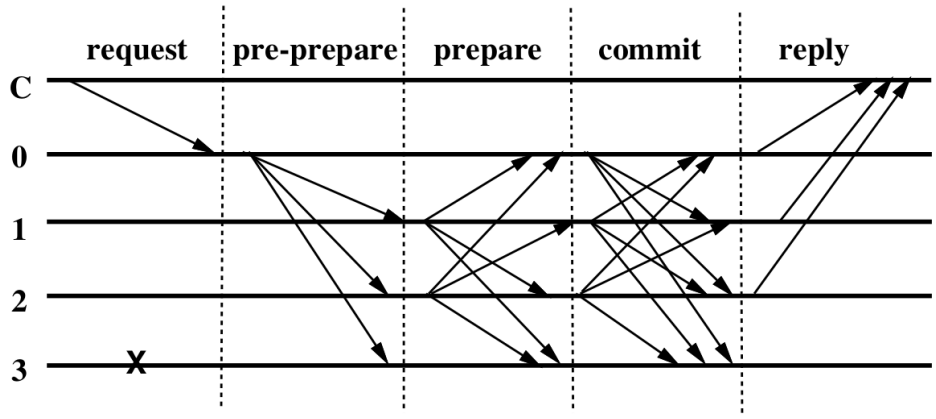


Figure 4.16 – Illustration of the normal case operation of the PBFT protocol in a distributed system, as originally drawn by M. Castro and B. Liskov in [171]. *Copyright © 1999, ACM.*

for the number of voting phases, quorum thresholds or for the mechanism to change the leader in case of failure. Another well known example of such leader-based protocol is PAXOS [174], followed by many other more recent proposals [175–180].

Limits of Leader-based protocols

Leader-based protocols are usually said to be *blocking*, meaning that they stop or abort the decision process in case of a-synchronicity. In fact, in case of communication delays or failures (due to unreliable channels, faulty or even malicious nodes), a leader can only block-and-wait for a decisive vote, or time-out and abort the protocol. Blocking is necessary to ensure safety, but also implies wastes of energies in busy-waiting and increases latency in general. One may prefer to abort instead of blocking, to favor *liveness*, but a true progress is possible only when the network recovers from its a-synchronicity and with a non-faulty leader.

In fact, blocking is a forced option when the leader fails or it is not possible to timely communicate with him. There is no way to circumvent this limit because a unique leader must be part of the design, otherwise a unique order of transactions cannot be defined, as conjectured by Gray and Lamport [181]. According to their conjecture, the distributed consensus problem becomes equivalent to the problem of unique leader election. Actually, it has been proven that the election problem is even harder than consensus in general, since *"The weakest failure detector needed to solve Election is stronger than the weakest failure detector needed to solve Consensus"* [182].

The need of a leader implies, unfortunately, severe consequences [183,184]. For example, the leader is a single point of failure, that can be attacked via Distributed Denial of Service

(DDoS) to block consensus. Moreover, the decision of the single leader is not validated by any peer, so that its decisions may diverge from the majority of received votes. Having a leader is thus usually considered a concern for the protocol fairness.

Pure Voting

To avoid introducing a single point of failure (the leader), each node may send its vote to all others, building a pure voting system. This way each node can perform the counting operations autonomously, understanding whether a majority has been reached or not, and finally committing or aborting accordingly. Given that each of the N nodes needs to send a message to all other $N-1$ nodes, in every voting phase the nodes exchange $N(N-1)$ messages. The communication complexity of a voting phase is therefore $\mathcal{O}(N^2)$, a high quadratic complexity that prevents the deployment of pure voting systems at large scales.

Federated Voting and Sharding

On one hand, the efficient leader-based protocols suffer from single-point of failures and blockades while, on the other hand, pure voting systems are impractical. Federated Voting and Sharding are hybrid approaches that mitigate these issues. With Federated Voting the consensus is first solved in subnetworks, the federated regions of the main network, then the local results are merged for the whole federation. The Stellar protocol [185] is a well-known example for this class of protocols. In a Stellar network each node first chooses a subset of trusted nodes, made of known neighbors with long established trustworthy relationships, forming a "slice". Then the inter-slice consensus is solved with any desired quorum threshold. The overall consensus is later achieved for the global network applying a principle of recursive message passing and quorum overlapping, empowered by gossip protocols [186]. Similarly to federated voting, sharding [187] means that the original system is decomposed into shards, which are processed separately, and later recombined together for building a global consensus. Protocols based on sharding are, for example, *Chainspace* and *Omniledger* [159, 160]

The advantage of these hybrid approaches is that nodes reach consensus without directly knowing all other nodes of the network. This results in considerable scaling improvements. However, most gossip protocols rely on some synchronicity assumption to practically offer a performance gain, otherwise they grants only *eventual-convergence* properties. Summing up, with these hybrid approaches the scalability of leader-based protocols is partially traded for approaching the robustness of pure-voting systems, but the original issues of the two major paradigms are only mitigated.

Virtual Voting

In virtual voting protocols, nodes do not explicitly cast their votes, they nevertheless acquire the necessary information to solve the consensus problem on the order of transactions. One such protocol is, for example, the hashgraph consensus algorithm [188]. This protocol exploits a gossiping mechanism to spread transactions epidemically, also attaching metadata to describe which transactions resulted critical for triggering new transactions. These metadata are enough to rebuild the causal relationships among transactions, allowing all nodes to sort them, even without explicitly casting their votes. A virtual voting approach improves on the performance of a pure voting system exploiting gossip protocols, that allow nodes to not directly communicate with all other nodes. Moreover, it is well known that a causal consistency order [189] is less strict than a sequential order [190]. This opens a further space for improvements on latency, in fact, for more transactions it will not be necessary to block the ledger to enforce of an only causal-order, while more strict blockades would be necessary to enforce a sequential consistency.

4.4.2 Lottery Protocols

In protocols based on voting, the nodes agree on what to add to the blockchain by either casting their votes, forming a majority in favor of a block, or by electing a leader that decides the next block for the chain. In a lottery-based protocol, instead, the consensus mechanism is a random process —a lottery— that similarly chooses the node that can add a new block to the chain. The election is transformed therefore in a lottery, with the winner that replaces the leader. For example, in Section 4.1.3 the PoW turned out to be the "winning ticket" (i.e., the proof), that a node should exhibit to claim the role of leader. A new lottery draw must take place for each new block, this to reduce the probability that a malicious node takes control over the blockchain. Lotteries do not need to follow uniform distributions, rather, each mechanism is characterized by the probability distribution chosen to bias some property of the winner, hence, each lottery protocol reflects a particular trust model. Lottery protocols are less efficient and performing than voting-based ones, but better support permissionless blockchains as they are open to anybody, welcoming anonymous validators that do not need to be registered. The rest of this section explores the lottery-based protocols in relation with their specific "proof".

Proof of Work (PoW)

The block nonce is the winning ticket for PoW-based protocols, and only the first miner that finds a valid nonce can forge a new block. Any user with some computing power can participate to a PoW-based consensus protocol, which is thus open to anybody. Although

effective for implementing a distributed, secure and open consensus mechanism, the PoW comes with severe drawbacks. The greatest concern is the high demand of power, but also the slow transaction speed together with the lack fairness. In fact, a miner is free to include in a block the transactions he prefers, deliberately ignoring some others. This will be especially true for Bitcoin when the controlled supply of new coin will terminate [191], with nodes that will process transactions ordered by fees for the miners, so that transactions without fee will be most probably never added to the chain.

Proof of Stake (PoS)

Systems based on PoS leverage on the economical rationality of their users to reach consensus and to honestly grow the blockchain while processing transactions. With PoS there is only a random process that selects the block proposers, and this process is biased towards those nodes that own more cryptocoins, making more likely the election of the richest stakeholders. The richest stakeholders have a vested interest in keeping the network well functioning, so that the overall system can be perceived trustworthy and valuable. Ultimately, it is in the rational interest of users to play honestly at the PoS-protocol in order to safeguard and enhance the value of the same system where they own a great stake. The PoS is also considered an energy-aware alternative to the PoW, as long as block miners do not need to waste power for forging new blocks.

Delegated Proof of Stake (DPoS)

The DPoS is a more scalable variant of the pure PoS that outperforms all other proof-based consensus mechanisms [192]. To improve on scalability the consensus problem is solved by a restricted number of delegates, elected in the network again by a random process biased by stake. What distinguishes a DPoS from a PoS is therefore only the election of a little committee responsible for the honest validation transactions. The fact that delegates are known (accountable) and elected according to a rational criteria ensure their honest behavior, which can be further enforced by asking them to escrow some coins that will be confiscated if they are caught misbehaving. The DPoS scheme is implemented by many popular platforms such as EOS, Tron, Steem, Bitshares, and Algorand.

Other Proofs

Essentially, all lottery-based protocols are driven by their users commitment, which is either proved demanding a sacrifice (e.g. of computing power), or enforced by biasing the users with a greater interest at stake. Other known analogous proofs are reported for the sake of comprehensiveness in Table 4.2.

Table 4.2 – List of known proofs empowering lottery-based protocols.

Proof of Elapsed Time (PoET)	The random time spent in a waiting queue is the sacrificed resource [193, 194].
Proof of Importance (PoI) & Proof of Networking (PoN)	A different mix of metrics, including network topological information, allows the computation of the node commitment in the network. The POI concept was introduced by the *NEM P2P cryptocurrency*, while the PoN has been originally proposed in [12].
Proof of Burn (PoB)	A node metaphorically "burns" coins sending them to a dead address. This way it gains the privilege of leading the consensus.
Proof of Capacity (PoC)	The well-known PoW turns into a PoC if memory is the main resource necessary to generate a valid proof, like in [195].
Proof of Deposit (PoD)	PoS-protocols suffer from the *"long range"* and the *"nothing at stake"* attacks [196]. This last attack is performed by nodes that do not own any coin, so they can keep trying to create forks, slowing down the consensus, without risking their stake. To counteract this attack, in Casper [197] nodes have to deposit some coins, an action called bonding, in order to the become validators and generate blocks.

4.4.3 Other Approaches

Round-Robin

If the network is fully synchronized, nodes can implement a fair succession procedure via Round-Robin, alternating each others as leaders. A Round-Robin scheduling is fair and can leads to high transaction rates, however, any node can maliciously deviate from the protocol and a central clock/scheduler is required.

Node-to-node Consensus

Bitcoin enacts an open, censor-ship resistant system where transactions are validated in public. However, especially to support flexible businesses, one may be willing to implement transactions via private, bilateral agreements. Two mutually trusting counterparties can thus validate their transactions between themselves, "node-to-node", without involving anybody and reducing the interactions with a public ledger as much as possible. Fast-payment channels are an example of node-to-node consensus. These are channels that two neighbors may open to perform their frequent and private transactions without incurring in the multiple fees that would derive from transacting over a public ledger. Node-to-node transactions are sometimes said to be *off-chain* [198, 199], since they are not recorded on any main blockchain. Two neighbors that opened a fast-payment channel may want to close it when the channel balance, which accounts for hundreds or even thousands off-chain transactions,

reaches a disproportionate amount in favor of one party. To clear this balance the debtor issues a closing transaction on a separated blockchain, compensating the creditor. Off-chain transactions can significantly reduce the load of a blockchain, enhancing considerably the network transaction rate.

Table 4.3 – Comparison of Consensus Techniques

	Transaction Rate	*Network Scalability*	*Consensus Participants*	*Attacks*	*Centralization*
Leader-based PBFT variants	High	High $\sim \mathcal{O}(N)$	Registered nodes	Collusion of $1/3+1$ nodes	Central coordinator
Pure Voting	Low	Low $\sim \mathcal{O}(N^2)$	Registered nodes	Collusion of $50\%+1$ nodes	Fully distributed
Sharding	Medium	Medium	Known neighbors	Collusion of a strategic minority	Federations
Virtual Voting	Medium	Medium	Known neighbors	Cheating neighbors	Fully distributed
PoW	Low	Low	Anonymous	Attacker with huge computing power	Fully distributed
PoS	Medium	Medium	Anonymous	Collusion of richest stakeholders	Fully distributed
DPoS	High	High	Elected Delegates	Collusion of delegates	Requires delegates
Round-Robin	High	High	Registered nodes	Block by any malicious user	Requires global synch
node-to-node	High	High	Known neighbors	Cheating neighbors	Fully distributed

Table 4.3 concludes this section summarizing the main properties of the surveyed consensus protocols.

4.5 Algorand: "The Trilemma Solver"

Algorand [126, 200] is the innovative blockchain-based permissionless ledger supporting the ALGO cryptocurrency. Experimental deployments of Algorand achieved a 125× Bitcoin's transactions throughput, with blocks confirmation times under a minute and without requiring any huge consumption of power as with Bitcoin. These results gained Algorand its reputation for being the *"Trilemma Solver"* [201]. This sections is devoted to the discussion about the possible collocation of Algorand within the landscape of the DLTs drawn in light of the trilemma.

4.5.1 Main Peculiarities of Algorand

Pure Proof of Stake & Secret, Random Validators

Algorand defines itself as a **Pure Proof of Stake**[30] system for stressing that, albeit relying on Stake rather than on Work, it manages to preserve a high degree of decentralization thanks to the randomized election of the delegates responsible for the approval of blocks of transactions. This fact distinguishes Algorand from a generic Delegated PoS protocol: both speed up the consensus delegating it to a relatively small number of committee members but in a generic DPoS protocol these members, once chosen, are made publicly known and remain in charge for multiple voting sessions, becoming this way clear targets of bribery from attackers, while the randomized, secret and continuous replacement of committee members designed by Algorand avoids this risk. According to the described mechanism Algorand is said to implement the **secret self-selection** of both block proposers and validators, adding **player-replaceability** to prevent an adversary from targeting publicly known validators with DDoS or bribery attacks.

Sortition & Algorand Protocol Phases

The continuous election and replacement of blocks generators and consensus participants is performed distributedly thanks to a cryptographic primitive called *"sortition"*. This primitive exploits Verifiable Random Functions (VRFs) [202] and *ensures that a small fraction of users are selected at random, weighed by their account balance, and provides each selected user with a priority, which can be compared between users, and a proof of the chosen user's priority* [200]. Sortition is extensively exploited in the two main phases of the Algorand protocol:

1. Block Proposal
2. Block Voting

In the Block Proposal phase few nodes are selected at random, via sortition, for proposing a new block of transactions. The few blocks generated in this first phase are provided as input to the second Voting phase, where random committee members will try to reach a consensus on one given block adopting *BA⋆*, i.e., a voting-based Byzantine Agreement protocol.

The BA⋆ protocol

The Block Voting phase is driven by a voting-based consensus protocol where nodes try, in multiple consecutive steps, to form a consistent majority for a particular block and either

[30] The Algorand official blog reports a post from S. Gorbunov entitled: "Pure Proof-of-Stake Blockchains" `https://www.algorand.com/resources/blog/secure-blockchain-decentralization-via-committees`

succeeds, committing a block after a finite number of steps (thus in bounded time) or timeout and abort the *BA*⋆ protocol committing the "empty block". Sortition is used in each step of *BA*⋆ to elect new users asked to vote on blocks. The Algorand protocol fixes several parameters to regulate, e.g., the expected number of valid block proposers or the committee size τ, i.e., number of users that should participate to *BA*⋆ in the various protocol steps. A large τ mitigates the risk of a malicious user being able to collect a relatively small number of votes sufficient to reach the 2/3 quorum imposed by *BA*⋆, but slows down the voting phase. The committee size is thus a relevant parameter from a security/performance trade-off perspective. The Algorand Technical Report [203] indicates that to limit to 5×10^{-9} the probability of a safety violation in *BA*⋆, τ should be set to 2000 (quite large committee size) and the 80% of nodes in the network must be assumed to be honest. It also shows that the committee size required to keep this same probability of safety-violation grows very fast when reducing the ratio of honest users in the network.

Network Assumptions

The distributed verification of eligibility for both block proposers and committee members is made consistent mandating nodes to initialize the seed for sortition with some commonly known random information, i.e., some of the least significant bits extracted from the hash of the last approved block. The protocol round/step-number must be also included for seeding the sortition primitive. The need of this common knowledge, together with the protocol tight time constraints before firing a timeout while collecting votes in *BA*⋆, means that nodes must be highly synchronized. In fact, *To achieve liveness, Algorand makes a "strong synchrony" assumption that most honest users (e.g., 95%) can send messages that will be received by most other honest users (e.g.,95%) within a known time bound* [200], which means that Algorand makes progress only under this assumption while it will safely block (throughput = 0 TPS) when the network experiences "asynchronicity", e.g., because of longer propagation delays or attacks to the networking infrastructure. Furthermore, 2/3 of the users online must be honest in order to form voting committee that do not violate the Algorand safety conditions,

No Incentives

Algorand does not define incentives for block generators or validators: the lack of incentives separates well Algorand from the other popular cryptocurrencies whose security is usually studied under game theory, with block rewards used to drive nodes towards the adoption of a honest behavior.

4.5.2 Main Observations and Comments

1. **Latency and Honesty assumptions** Algorand includes a voting based BA protocol that, relying on the Algorand sophisticated gossip mechanism and on the Algorand relay network[31] minimizes communication latency and allow Algorand to finalize blocks in very short time, achieving this way a throughput in the order of 1,000 TPS, planned to grow soon to 46,000 TPS introducing *block pipelining* [204]. Essentially, Algorand strongly relies on the assumption that the network latency will be low to speed up as much as possible block generation and voting phases. However, this assumption on network latency turns out to be a double-edged sword for Algorand. In fact, a minimal noise or malfunction in the network can introduce delays sufficient for triggering continuous timeout in the *BA⋆* protocol, forcing the system to commit only empty blocks, thus preventing Algorand to make a true progress reducing to 0 TPS the throughput. Conti et al. [201] show that an attacker can indeed easily obstruct and stop the Algorand network exploiting this fragility. They simulate an Algorand network with only 1%-3% of malicious nodes that flood the network with bogus yet valid blocks, this flooding strategy turns out to be effective for delaying the processing capabilities of honest validator nodes, growing the execution time of the *BA⋆* protocol enough to cause timeouts. The work from Conti et al. has therefore shown that it is simple and cheap to setup a DDoS attack to Algorand based on flooding. This observations raises the awareness on the fact that Algorand is not only a PoS system, but also a voting-based one, thus inherits the main limitations of such protocols, above all the possibility of being blocked via DDoS, as discussed in Section 4.4.1.

   Furthermore, the Algorand protocol requires the 80% (4/5) of the committee members to be honest for granting a high probability of safety, introducing therefore a stronger trust assumption compared to the 2/3 requirement valid in general for BA protocols.

   In conclusion, Algorand introduces an innovative architectures that blends together advanced cryptography with sophisticated and very performing networking protocols. The design assumptions on the network latency and on the ratio of honest users in the network, albeit defensible, are strong and represents limits for the reliability and the security of Algorand.

2. **No complete Decentralization** In January 2021 a user of the Algorand community criticized the Algorand foundation for being the single maintainer of the relatively short list of relay nodes, the only involved in the spreading of protocol messages, *basically making Algorand a permissioned, centralized blockchain* [205]. The official reply states

[31] More on the Algorand network architecture that comprises Relay Nodes here: `https://algorand.foundation/algorand-protocol/network`

that Algorand is planning for the future to mix relays vetted by the foundation with permissionless ones to enhance the decentralization degree of Algorand, still, as of today the Algorand network cannot be considered fully decentralized.

3. **Lack of Incentives** The lack of an incentive mechanism may lead to the so called *"lazy validator problem"*, meaning that a validator not rewarded for his honest participation in the protocol tends to simply shutdown its node. This problem is a source of concern for Algorand as the protocol relies on the assumption that 2/3 of the nodes online are honest, an assumption that may be not fulfilled if honest nodes goes offline not contrasting malicious nodes, that are instead motivated by rewards coming from successful attacks such as double-spending. The robustness of Algorand without an incentive system is indeed still debated [206–208].

4. **Potential Errors in Security Assumptions** According to Yongge Wang [209], some of the security assumptions made by Algorand designers are wrong. In particular, the author of [209] claims that:

   - An attacker controlling even less than 1/3 of the ALGO tokens can easily originate a fork;
   - Despite the continuous replacement of committee members an attacker can contact online nodes to corrupt them before they get randomly elected;
   - The innovative $BA\star$ protocol can be actually replaced by any plain voting protocol based on majority providing the same features of $BA\star$.

The presence of a centralized networking infrastructure, such as the Relay network, and the well known security limitations of quorum based voting protocols, prevent Algorand from eluding the limits of the trilemma. Nevertheless, its unique and innovative design choices allow the depart from PoW, making Algorand energy efficient, enable a considerable growth of the Transactions Throughput when the network is tightly synchronized and grant security (by blocking) when the network experiences partitions or delays potentially originated by attackers. While communications happens on a centralized infrastructure, nodes participating to the consensus protocol are permissionless, thus Algorand is able to preserve a good degree of decentralization.

Ultimately, within the inescapable limits of distributed systems discussed in Sections 4.3.1 and 4.3.2, the trade-off chosen by Algorand seems to ensures the minimum possible disruption of Decentralization and Security to achieve the Scalability required to run modern, globally distributed applications.

4.6 Blockchain in Networking: Related Works

The blockchain-based applications documented in the literature are many and disparate. The large spectrum of blockchain-based applications, originally limited only to cryptocurrencies, seems now to be so broad to include: Supply Chain Management [19–22], E-Voting [23–30], Smart-Grid [31–38], Healthcare [39–42], Banking [43, 44], Smart Cities [45–47] and even Vehicular [48–61] and Drone [62–66] Networks. For a systematic review of blockchain-based applications the reader can refer to [210], but also to other recent review papers [67–75]. This very wide application range witnesses the "blockchain rush" of industries and academics that looked at the blockchain as to an almost limitless, universal technology, to be applied to any commercial sector and in any discipline. The doubts about this impulsive adoption of the blockchain led different authors to reflect on the true, novel advantages brought by the blockchains, with M. E. Peck and Wüst et al. that critycally raise the question: *"Do you need a blockchain?"* [211, 212].

To answer the question, Herlihy [213, 214] and De Angelis et al. [215] suggest to study the blockchain from a distributed system perspective. This is the necessary perspective to appreciate the true, original features of the blockchain, but also to uncover the major concerns of this technology and of its derivatives, such as Smart Contracts. For example, the poor support of concurrent operations [216] opens the doors to a vast number of dangerous attacks, as those surveyed by Atzei et al. [217], to be added to the other well known security risks succinctly analysed in Section 4.2, and studied in detail also in [73, 74, 134, 135, 218, 219].

While the lack of support for concurrent operations remains an important limitation of the blockchain technology, which hampers its performance and security, the other security risks can be considered only mildly relevant because, as explained in Section 4.2, the most concerning double spending attack is actually foiled adequately by the PoW. The same PoW, however, implies great consumption of power [153, 154] and contributes in slowing down the transaction rates while increasing latency [220, 221], so much that it is considered as the main problem for integrating the blockchain into the domain of distributed computer networks [222]. Not only, Gencer et al. [223] and Gervais et al. [224] criticize the PoW for not being a genuine distributed consensus mechanism, since they observe a tendency for the computing power to consolidate in the hands of few central players. Despite these disadvantages, the PoW seems to be indispensable to create trust in a blockchain, so much that financial investors focus almost exclusively on PoW-based blockchains, that together account for more than 90% of the total market capitalization of existing digital cryptocurrencies [225, 226].

As the PoW is burdensome, many have been the proposals to replace it targeting specific applications. For a survey of consensus mechanisms the reader can refer to the brief one

reported in Section 4.4 and to other available surveys such as [69]. An alternative of particular interest for distributed networks is, for example, the Proof of Networking (PoN) [12], which is essentially a formulation of the ideal properties that a consensus mechanism should blend together to boost blockchain-based applications for distributed networks. Nonetheless, the shift from ideal properties to concrete implementations remains complicated by scalability issues, as noted in the experimental work of Selimi et al. [227], that introduced a transaction platform (based on HyperLedger) in a real world wireless mesh network highlighting several potential bottlenecks that could limit the rate of economic transactions.

Both [12, 227] envision one valid use-case of the blockchain for distributed networks, i.e., the blockchain as a distributed marketplace of networking resources, that would open the way to the *Distributed Internet Service Providers (DISPs)*. Example of this vision becoming reality are two projects, namely, AMMBR and Althea,[32] that candidate themselves as wireless, blockchain-empowered DISPs, thus contributing to the implementation of a more reliable and cost-effective networking paradigm driven by cooperation. The success of projects of this kind is expected to offer a solution to the sustainability problems that afflict Community Networks (CNs), as conjectured in [228–230] and forecasted in [231, 232].

The application range of the blockchain in the domain of distributed networks goes beyond the wireless DISPs. For example, Castro et al. [233] propose the use of [sic] *"a decentralized public ledger and cryptography to provide ASes with automatic means to form, establish, and verify end-to-end connectivity agreements"*, this to solve the interdomain contractual and routing issues among different ASes. Jin et al. [234] propose the adoption of a blockchain within Named Data Networks as a measure to reliably update the hisotry of a content and to speed up the the synchronization of consumers that, while disconnected, missed a row of updates. A blockchain-based ledger can serve also as a mean to share blacklists among ASes that cooperate to counteract DDoS attacks, as proposed in [235]. Emercoin[33] is another platform that offers a portfolio of networking services, re-implemented in a distributed fashion thanks to the blockchain technology. The services include the emer-based versions of the Domain Name System (DNS), Secure SHell (SSH) and Secure Sockets Layer (SSL) protocols.

An attempt of generalizing the kind of Internet applications that one can implement on top of a blockchain has been advanced by Hari and Lakshman while formulating *"the Internet Blockchain"* [236]. To formulate this generalization, the authors of [236] first elucidate the

[32]The two projects (AMMBR and Althea) aim at incentive competition among the participants of a distributed networks setting up a blockchain-based marketplace. Leveraging on microtransactions, participants can trade networking resources playing both roles of users and service providers. While Althea relies on the *Cosmos blockchain* (`https://cosmos.network`), AMMBR is based instead on *Plasma* (`https://plasma.io/plasma.pdf`, a sidechain rooted in Ethereum). More details on the two projects are available online at `https://ammbr.com` and `https://althea.net`.

[33]Homepage of the Emercoin project: `https://emercoin.com`

peculiar features of a blockchain, then identify the problems that can be alleviated relying on these features, and finally list several use-cases belonging to the Internet domain that suffer these problems. In particular, they start by observing that a distributed, tamper-resistant shared ledger is an ideal framework for recording those digital transactions used today for the assignment of networking resources such as IP addresses, ASes numbers and domain-names. Then they focus on the main security breaches hidden under these assignments, that have in common the requirement of a central source of trust. This is considered a vulnerability because, in principle, it can be compromised by institutional actors or advanced cybersecurity attacks. All the secure assignments of IP addresses or ASes numbers happens today, in fact, through the so called Resource Public Key Infrastructure (RPKI), a critical infrastructure to prevent various attacks, including route hijacking.[34] The same RPKI is used also to sign BGPSec [237] Updates, preventing spoofing attacks. However, all these security mechanisms suffer under a single point of failure, i.e., the RPKI's root of trust.

The transaction framework elaborated in [236] is convincing because it can tolerate the typical slow transaction rates of a blockchain. Consider that, for example, the average number of ASes numbers allocated per month[35] by a Regional Internet Registry (RIR) is less than 200/month, which corresponds approximately to only 7.72×10^{-5} TPS, a tolerable transaction rate also for a public blockchain. Still, the original question of Peck and Wüst [211, 212] remain unanswered for applications that require higher transaction rates and lower power consumption, requirements of the utmost importance, e.g., for the IoT domain [238, 239]. An exhaustive list of considerations and criteria for the selection of the proper blockchain technology, according to different mix of performance requirements and trust models is instead provided by [76], which is a comprehensive vademecum for companies, academics and blockchain-practitioners in general.

However, not even the convincing *Internet Blockchain* nor the exhaustive *vademecum* completely solved the performance and power consumption issues of the blockchain. Rather, these works are limited to point out the valid use cases for the blockchain, valid for two fundamental reasons:

- First, these are the cases of applications that can tolerate the blockchain inefficiencies, where a sacrifice in terms of performance is accepted for gaining the freedom from central points of authority;

[34] By *route hijacking* it is meant the illegitimate appropriation of subnets by tampering the records in the routing tables of BGP routers. More on: `https://en.wikipedia.org/wiki/BGP_hijacking`

[35] The Allocation statistics of RIPE, the RIR that serves Europe, western Asian countries, are available online: `https://www.iana.org/numbers/allocations/ripencc/asn`

- Second, none of these applications really *integrates* the blockchain in the network, pushing it down to the protocol stack, rather, the blockchain plays the role of an external system that replaces traditional (centralized) sources of trust.

The analysis of the literature concerning the blockchain, as applied to distributed networks, suggest to reformulate the original question posed by Peck and Wüst ([211, 212]): *Can the blockchain be integrated into the network-stack? When should someone rely on a blockchain, rather than fall back to traditional, well-established ledger technologies? Can the blockchain empower a truly efficient, scalable and secure shared ledger for the IoT?*

Chapter 5 is devoted to answering these questions.

Chapter 5

Clarifying the Role of the Blockchain for Distributed Networks

The analysis of the literature concerning applications based on the blockchain, presented in Section 4.6, has raised doubts on the today adoption of the term "blockchain". The fact that this term appears in projects that exhibit extremely different levels of security, decentralization and performance creates ambiguity around the concept of blockchain, which appears to be blurred.

At this stage of the thesis one of the two is true: either the blockchain is an almost universal and virtually limitless technology, truly beneficial for the whole range of applications documented in Section 4.6 or, rather, many different technologies for some reason all named "blockchain" are actually at work under the hood. A preliminary clarification and disambiguation of the very term blockchain becomes therefore necessary before moving to the discussion of the role of the blockchain in distributed networks, main goal of this chapter.

A contribution offered in this thesis is thus the advancement of a restrictive definition of blockchain [77, 78]. This definition is elaborated to outline a set of peculiar features that can be exclusively attributed to the blockchain, distinguishing it from the rest of the Distributed Ledger Technologys (DLTs). The arguments supporting the proposed definition are grounded in the background covered in Chapter 4, which elaborates on the limited selection of academic works referenced in this thesis. The author acknowledges the lack of systematic comprehensiveness but motivates this sacrifice with the necessity of providing an unambiguous definition able to demarcate the limits of the blockchain. The definition introduces this way a novel point of reference proposed, intentionally, for discriminating those technologies that today fall all together under the common name of "blockchain".

The rest of the chapter is organized as follows:

- Section 5.1 compares the most famous platforms, considered to be emblematic blockchains, with the competing technologies for the implementation of a Shared Ledger (SL). This comparison is offered to let emerge those constitutional features that, together, are proposed to uniquely define the term “blockchain”.
- The precising definition of the term “blockchain”, of crucial importance for the later critical analysis of the blockchain integration in distributed networks, is formulated in Section 5.2.
- Recommendations on the use of the blockchain are elaborated in Section 5.3. The recommendations include a list of pitfalls that warn against those use-cases for the blockchain that, in light of the definition, turn out to be flawed.
- Section 5.4 is the *pars construens* of the critical analysis: this section will outline a possible role in the IoT for what has been restrictively defined to be a blockchain.

5.1 Blockchain VS. Traditional Technologies

Platforms considered to be emblematic blockchains will be compared with the competing technologies for the implementation of a SL. This comparison is crafted so to identify and highlight the characteristics of a blockchain, to be later condensed in a definition of blockchain.

Figure 5.1 substantiates this high level comparison and aids the discussion about the technological advantages and disadvantages of, respectively, centrally managed Data Bases (DBs), distributedly managed DBs and the still undefined “classic blockchains”. The declared intention is to capture under the name of “classic blockchains” all those platforms that preserved the revolutionary aspects proposed for the first time by the blockchain par excellence, i.e., Bitcoin.

5.1.0.1 Centrally Managed DBs

A centrally managed DB is a DB maintained by a single administrator, such as a trusted employee or, more in general, a single company in charge of keeping the DB well maintained and updated. The data contained in the DB can be shared among various clients upon request from the same clients. The DB maintainer can, at his own discretion, authorize or deny the access to the DB. According to the described paradigm, a centrally managed DB represents a possible centralized implementation of a SL.

The greatest advantage of a centrally managed SL is its high level of efficiency in terms of transaction rate, communication effort and power consumption. In fact, to implement a

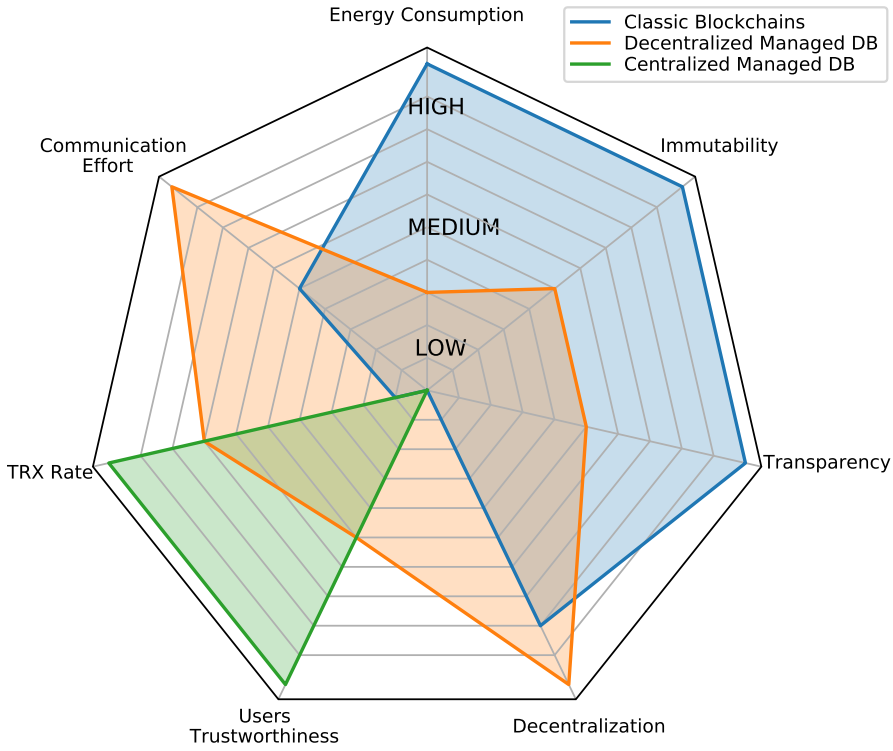


Figure 5.1 – Multidimensional comparison of the blockchain with traditional Shared Ledger technologies: i.e., centralized and decentralized managed DBs. *Copyright © 2021, IEEE.*

centrally managed DB, an administrator can exploit the well-established and decades refined technology of commercial DBs, this way achieving very high transaction rates at a low power consumption. Moreover, the DB administrator works autonomously, so it also avoids the communication efforts implied by a consensus protocol that would become necessary to coordinate the read/write operations of more DB maintainers.

Despite the many advantages described so far, centrally manged DBs also exhibit several shortcomings. For example, the SL users must completely trust the administrator, as it has full and exclusive control over the data. The administrator can, in principle, tamper or censor data, and even resell users data. Given that in a centralized system nobody controls nor validates the admin operations, a centrally managed DB is not considered transparent, and not even immutable, as the admin is free to delete data or, maybe worse, not record legitimate data.

5.1.0.2 Distributedly Managed DBs

A distributedly manged DB is a DB cooperatively maintained by a group of administrators. Each administrator may be responsible for a subset of the recorded data or for a full copy of

the DB. In the first case the reader can refer to *classic distributed DBs* [240–242],[36] while in the latter case to *replicated DBs* [246–249]. Before the rise of blockchains, distributed DBs represented the only available option to implement —distributedly— a SL.

Distributed DBs differ from centralized DB implementations for the introduction of a certain degree of redundancy, necessary to avoid single point of failures. This means that nodes must adopt a consensus protocol to coordinate write operations, enforcing this way a consistency model [190, 250]. This distributed architecture enhances the system tolerance to failures, this latter varies according to the level of redundancy dictated by the internal consensus protocol. Moreover, a distributed architecture is more secure than a centralized one, because to tamper data one need to corrupt more nodes. The collaborative approach of administrators for updating data makes distributed DBs also more transparent and, ultimately, nodes do not need anymore to place unconditional confidence in a single administrator.

The advantages of distributed DBs comes at the cost of an increased communication effort due to the running of a consensus protocol. The fact that transactions, to be approved or rejected, need to go trough this consensus protocol, deteriorates the overall transaction rate. Although the internal redundancy does not expose distributed systems to single point of failures, a distributed system is always threatened by collusive attacks led by a majority of malicious nodes.

SLs implemented on top of a distributed DB do not require to use a blockchain data-structure to record the users transactions. Any other data-structure, such as a basic linked-list or a DAG as a mean to build a dependency-graph, is acceptable as long as the DB is maintained consistent, complying at the same time with the rules of the internal consensus protocol.

5.1.0.3 Classic Blockchains

Bitcoin and Ethereum are the most emblematic, iconic blockchains. Both are based on the Proof of Work (PoW), precisely as many other popular PoW-based blockchains[37] that, together, account for more than 90% of the total market capitalization of existing digital cryptocurrencies [225, 226]. Some authors refer to these permissionless, PoW-based blockchains as to *classic blockchains* [76].

In these platforms the blockchain represents a particular case of decentralized DB, characterized by the maximum degree of transparency and immutability. By definition a classic blockchain platform leverages on a “block-chain”, i.e., a special linked list that concatenates blocks of transactions via cryptographic links, as described in Section 4.1.2. Because of the constitutional immutability of classic blockchains, in these systems it is not possible to delete

[36] Well known examples of distributed DBs are Google Spanner, MongoDB and Cassandra [243–245]

[37] Examples of other famous PoW-based cryptocurrencies are Bitcoin-Cash, Litecoin, Namecoin, Dogecoin, Primecoin, Auroracoin, Monero, Etherum-Classic and Zcash [251–258]. For a more complete list of cryptocurrencies the reader can refer to [259].

or modify transactions, which can be only appended. This feature makes classic blockchains suitable for the implementation of anti-censorship, tamper-proof registries. These registries record the complete history of transactions (remind that, in fact, transactions cannot be deleted) and are left open to the public. For these reasons, any distributed agent can check the validity of any transaction at anytime.

To participate to the PoW-based consensus, users just need to join the validator networks with their mining equipment, without the need of being registered providing their identity, so a classic blockchain is open to anonymous users. A consensus mechanism able to safeguard the ledger even from powerful anonymous users, that cannot be trusted at all, becomes an imperative necessity. However, as observed in Section 4.2, this mechanism is both a blessing and a curse: it ensures high level of security but, unfortunately, also implies high latency, low transaction rate, and massive power consumption.

The very celebrated advantage of classic blockchains is that they free users from the need of trusted authorities, thus enabling a broker-less marketplace for anonymous users, which is also a guarantee of the users privacy.

5.2 A Connotative Blockchain Definition

The analysis of the competing technologies for the implementation of a SL (Section 5.1) suggests what are the peculiar, characteristic features of a blockchain, that distinguish the same blockchain from all other DLTs. These features seems to be:

Highlight 5.2.1 Fundamental Features of a Blockchain

1. FULL DECENTRALIZATION
2. TRANSPARENCY
3. OPEN TECHNOLOGY
4. PRIVACY
5. IMMUTABILITY
6. EXTREME SECURITY

Highlight 5.2.1 is just one of the many lists of blockchain features proposed in the literature [212, 260, 261] and online.[38] The overloading of the term *blockchain*, which is motivating this thesis disambiguation purpose, most probably stems from this abundance of possible characterizations. The key problem of such characterizations is their high level of abstraction: different ledgers empowered by extremely diverse technologies can in fact claim

[38] On the world wide web the number of different blockchain characterizations are uncountable. A few resources are linked here for a quick reference: `https://tinyurl.com/deloitteBCfeatures` `https://101blockchains.com/introduction-to-blockchain-features` `https://data-flair.training/blogs/features-of-blockchain`

to grant all the properties listed in Highlight 5.2.1, advancing their "blockchain nomination", without actually supporting these features with technical elements. If it is simple to claim all the desired properties it is instead hard to determine if one particular ledger really provides the features listed, e.g., in Highlight 5.2.1. To contrast the proposal of further axiomatic characterizations that can be continuously crafted and adjusted to reflect the properties of any possible ledger, this thesis proposes the following connotative definition of the term "blockchain" embedding intentionally technical key-words:

Definition 5.2.1 Fundamental Characteristics of a Blockchain

1. OPENNESS TO ANONYMOUS USERS
2. COMPLETE & PUBLIC HISTORY OF TRANSACTIONS
3. HARD DISTRIBUTED CONSENSUS PROTOCOL

The OPENNESS TO ANONYMOUS USERS is the first, essential, constitutional element of a blockchain, supporting the FULL DECENTRALIZATION, the OPENNESS and the PRIVACY of blockchains. Without anonymity, in fact, a blockchain could not protect the users privacy more than what is done by a bank or by any other centralized implementation of a SL. The openness of the blockchain is also fundamental for making this technology distributed, fair and democratic. If users had to be registered and authenticated, a centralized trusted registrar —potentially discriminatory— would immediately become necessary, compromising all the aforementioned aspects.

The openness to anonymous users is thus a major revolution inherent to the blockchain, but introduces also a new problem when dealing with transactions, especially with disputations. In the most common case, in fact, a person that wants to dispute a transaction (e.g, a victim of a fraud), invokes the arbitrage of a court or of the used payment system (e.g., PayPal) and hopes that a fair inquiry reaches the conclusion that the money have been stolen, and that must be returned back to the legit owner. However, in a blockchain this is not possible, as no authority can prosecute an anonymous, untraceable user. The anonymity of users raises therefore the problem of the disputation of transactions, and users must accept the fact that transactions are, de facto, indisputable.

How can it be that someone agree and accept an indisputable transaction initiated by an anonymous, unknown, untrustworthy user? The blockchain offers a solution to this problem by keeping the COMPLETE & PUBLIC HISTORY OF TRANSACTIONS. In a trustless network, in fact, a user that do not trust anybody else can accept a transaction only if it is empowered to perform an independent and complete check of validity of any transaction, at any time. If the user could rely on a trusted third-party, such as a bank, this user could

confide on the bank's trustworthiness for accepting a transaction, even if the bank keeps its ledger (naturally) secret and private. But the anonymous users of a blockchain resort to a blockchain precisely for avoiding trusted intermediaries and central points of control. They therefore absolutely need a distributed, public ledger, with the full history of transactions, to enable the broker-free validation of transactions in a trustless network. It is clear that the completeness of the history of transactions is a mandatory requirement because, without it, a distributed agent could not verify if a new transaction is already included in the same history, thus would not be able to reject attempts of double spending. Once more technical keywords —COMPLETE & PUBLIC HISTORY OF TRANSACTIONS— are prompted as necessary to achieve high-level features such as DECENTRALIZATION, OPENNESS & TRANSPARENCY.

Despite the availability of a public and complete ledger, the validity check of a transaction can be still falsified by an attacker that manipulated the history of this transaction, tampering the blockchain for gaining a personal advantage. The history of transaction, namely, the blockchain, must be safeguarded therefore by a HARD DISTRIBUTED CONSENSUS PROTOCOL, i.e., a protocol which makes prohibitive the cost of attacks, otherwise nobody would trust the system. This is why a mechanism like the PoW, which introduce on purpose cryptographical hardness for raising the cost of attacks to a prohibitive threshold, is a key element that makes a blockchain a distinguished technology for its claimed IMMUTABILITY. Definition 5.2.1 binds therefore the EXTREME SECURITY of blockchain to the HARDNESS of its DISTRIBUTED CONSENSUS PROTOCOL, with this second keyword highlighted to support once more the DECENTRALIZATION of blockchains. This kind of immutability is so fundamental for the concept of blockchain that [sic] *for cryptocurrency activists and blockchain proponents even simply questioning the immutable nature of blockchain is tantamount to heresy* [262].

Highlight 5.2.2 Arguments supporting Definition 5.2.1

Users Privacy achieved by Anonymity ⇒ non-disputable Transactions;

If the History of Transactions *is* _ *then* New Transactions *are* _*:*

- Private ∨ Partial ⇒ *unverifiable*
- Public ∧ Complete ⇒ *verifiable*

A Hard Consensus defuses the falsification of the History, making it immutable

To recap: a blockchain is designed specifically to make openly available to any anonymous user the complete chronology of all transactions, this to enable the distributed validation of transactions avoiding trusted intermediaries. Moreover, a HARD DISTRIBUTED CONSENSUS PROTOCOL is essential to protect this history of transactions, i.e., the blockchain, from

tampering attacks. The mathematical hardiness of such protocols makes the blockchain trustworthy despite the untrustworthiness of users. The arguments used to justify the Blockchain-Definition 5.2.1 are concisely summarized in Highlight 5.2.2.

5.2.1 Comparison with other definitions

An initial, basic definition of blockchain could be restricted to the only hash-chain with blocks of information linked by hash-pointers, a data-structure known in computer science since the '70s [263, 264]. The advent of revolutionary platforms such as Bitcoin and Ethereum, however, enlarged the meaning of the term blockchain. As a matter of fact, Iansiti and Lakhani propose a wider definition, commonly accepted in the literature, which is the following: *"[The] blockchain is an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way"* [261]. This definition focuses on the blockchain operational purpose as distributed ledger, distinguished from other DLTs for of its Openness, Verifiability and Immutability (permanent records) properties.

The blockchain definition advanced by this thesis (Definition 5.2.1) highlights the same characteristics, but it does so embedding in the definition the constitutional elements that connote a blockchain as an open, verifiable and immutable ledger technology. Definition 5.2.1, in fact, characterizes the blockchain as a technology *open* to any anonymous user, *verifiable* thanks to the complete and public recording of all transactions, and as much *immutable* as possible by reason of a hard distributed consensus protocol. This new definition, compared to previous ones, is not axiomatic: rather, acknowledges the blockchain as an open, verifiable and immutable technology deriving these properties from the three, inseparable elements that together constitute the essence of a blockchain.

The new definition is more restrictive, as it reserves the epithet "blockchain" only to those platforms that fully reflect the blockchain specificity described by Definition 5.2.1. The various platforms usually mentioned as blockchain without actually complying with Definition 5.2.1 are criticized in Section 5.2.2.

The proposed characterization of the blockchain also serves the thesis clarification purpose, being the base for claiming that *the blockchain is not a universal technology*, but it rather has peculiar characteristics which are advantageous for the implementation of an only limited number of applications. The many other applications proposed in the recent literature and that exhibit features in contrast with Definition 5.2.1 are criticized in Section 5.3.

5.2.2 Permissioned Blockchains or Permissioned Ledgers?

Who agrees with the blockchain definition advanced by this thesis should, consequently, stop considering to be blockchains all those platforms that do not exhibit the three essential

characteristics summarized in Definition 5.2.1. A question arises: given that permissioned ledgers, by definition, are not open to anyone, and that usually rely on consensus protocols that are not mathematically hard like the PoW, shall they be considered as blockchains?

Not an Open, Decentralized, Verifiable Technology

In fully permissioned ledgers the access is restricted only to permissioned users. This implies the existence of a central registrar, acknowledged by the system players, that is the only body responsible for the identification of users and for granting read/write authorizations. Also in hybrid blockchains where there might coexist many permissionless users with few permissioned ones, still one central registrar must exist to register these lasts. One such central registrar may be discriminatory, granting authorizations according to private, opaque, internal rules inspired by business policies: the simple presence of *permissions* is a clear symptom of the need to —arbitrarily— exclude some category of users, otherwise these same permissions would have no reason to exist. A system that aims to be truly decentralized, thus not subject to the arbitrariness of some central registrar, and also aims to be truly open, cannot therefore rely on permissions and access-lists to implement its security model. Notice also that permissions cannot be defined if users are indistinguishable anonymous, this is why ANONYMITY is part of the proposed Definition 5.2.1 as a guarantee of the blockchain decentralization, in the present case intended as freedom from central registrars.

Moreover, most of the permissioned platforms are built by enterprises to implement their ledger to be shared only among the internal departments. The transactions recorded in such ledgers are business-critical information and must be kept confidential, which means that a typical permissioned ledger is not verifiable by an external agent.

Less Immutable means less Secure

Granting the access to the blockchain only to registered and accountable users represents certainly a safer trust model if compared to opening the ledger to any anonymous user. Strengthen by stronger trust assumptions permissioned ledgers usually abandon the very secure but power-hungry PoW replacing it with more traditional, efficient and more performing consensus protocols. However, dismissing the PoW makes permissioned ledgers vulnerable to traditional attacks led by the “simple” — i.e., “inexpensive”, not discouraged by any costly sacrifice— collusion of a majority of users. Permissioned ledgers are therefore less immutable and less secure than iconic blockchains such as Bitcoin or Ethereum, that instead are not affected by this vulnerability.

An Historical Perspective: Permissioned Platforms are Traditional Ledgers

So far permissioned ledgers turned out to be very distant, in terms of openness, verifiability, and immutability, from public, permissionless blockchains. Ultimately, permissioned platforms seem to be not much different from traditional ledgers that existed also before Bitcoin [265], as they are empowered by traditional consensus protocols and their trust model still depends on a central authority. Therefore, from an historical perspective, while permissionless blockchains such as Bitcoin constituted a revolutionary advancement of the state-of-the-art, freeing users for the first time from the need of a trusted authority to perform distributed transactions, permissioned platforms represents only a technological sophistication of the older and well-known DLTs.

For these reasons, this thesis suggests to consider permissioned platforms as belonging to the broader class of traditional DB-based ledgers, but not blockchain representatives. The landscape of the Shared Ledger technologies, as envisioned by this thesis, is illustrated by Figure 5.2, where the blockchain is positioned according to the fundamental Blockchain-Definition 5.2.1.

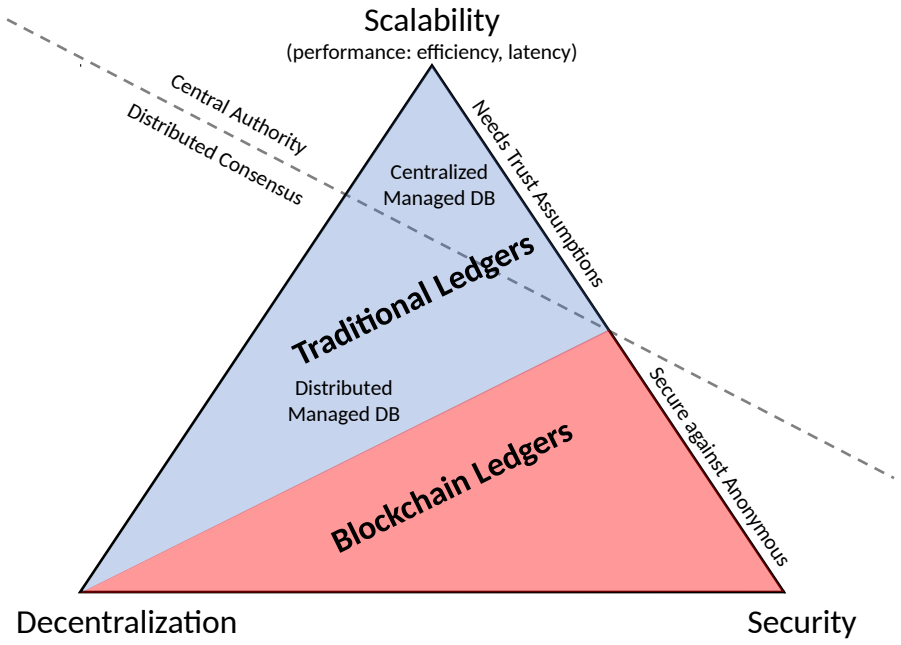


Figure 5.2 – Position of the Blockchain in the landscape of the Shared Ledger technologies.

5.2.3 Proof of Work or Proof of Stake?

The debate on what is the fairest, most decentralized yet performing consensus protocol for the blockchain is intense [266–268], with two major competing candidates: Proof of Work (PoW) and Proof of Stake (PoS).

Limits of the PoW

The limits of the PoW has been specifically investigated in Section 4.2, they can be summarized as it follows:

- Enormous power consumption (Section 4.2.5 and [153, 154]);
- PoW-based blockchains usually exhibit modest transaction rate and high latency [220, 221], at most if compared to traditional Byzantine-Fault-Tolerant alternative protocols [222];
- Tendency for the computing power to consolidate under the control of few large mining pools [223, 224]. This fact is witnessed also by Figure 5.3, that shows an estimate of the hashrate distribution among well known Bitcoin mining pools, highlighting how the 12 largest mining pools control almost the 80% of the computing power, while the top-5 pools alone control approximately the 61% of it. Nevertheless, it is worth to notice that pools are consortiums that aggregate together multiple users, so they internally implement a certain degree of decentralization.
- Lastly, some authors raised the concern that the PoW does not discourage adequately the formation of centralized cartels. For example, it has been proved that for few powerful users it may be economically advantageous to collude, behaving as "selfish miners", damaging the decentralization level of the network [135, 269].

Revisiting the limits of the PoW beyond its well-known scalability issues, it seems that it partially fails also as a tool to decentralize the control of a SL. In light of these drawbacks, in both power-efficiency and effective decentralization terms, Saleh [270] and Bentov et al. [271] encourage the shift from the PoW to the PoS to solve, at least, the problem of wasting energy, giving up with the elaboration of a truly fair and decentralized consensus fabric.

Limits of the PoS

PoS-based systems have been also investigated and criticized, with the following issues recurrently emerging as major sources of concern:

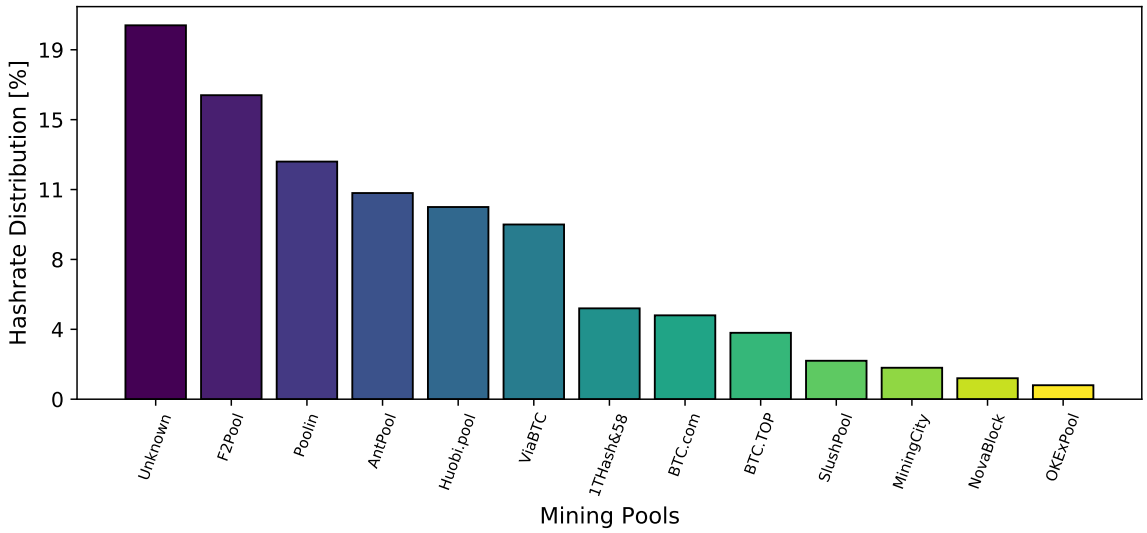


Figure 5.3 – Estimate of the hashrate distribution among well-known mining pools at the date of writing this thesis (October 21, 2020). Statistics are taken from `https://www.blockchain.com/charts/pools`.

- Protocols driven by stake are criticized for their unfairness and poor decentralization, due to their design that deliberately priorities the voting power of the richest stakeholders to the detriment of the larger community of generic shareholders. This known phenomena that contributes to the inequity of PoS systems is usually condensed in the motto *"the richer get richer"* [272].

- Moreover, while rigorous proofs of convergence and tolerance to byzantine failures exist for traditional voting protocols, and similar security analysis exist also for PoW-based blockchains [134], the studies on the economics of stake-based systems suggest that their equilibrium is not always granted [273, 274]. Some authors raise therefore the concern that, like with real markets, stake-based protocols will give rise to unstable systems subject to market failures and bubbles [275].

- The security offered by the PoS is considered inferior to those provided by the PoW. In fact, the immutability of a PoS based system is considered questionable, as a PoS-based blockchain is reversibile by means of long-range or stake-bleeding attacks [276, 277], which are attacks not discouraged by external factors such as the cost of energy or of the mining equipment.

Advantages and Disadvantage of PoW and PoS

In light of this discussion is the transition from PoW to PoS planned by many popular blockchain systems (in primis Ethereum) to stop wasting computing power still justified?

This thesis claims that both PoW and PoS essentially empower a *census suffrage* system. With both systems, in fact, the richest users acquire a greater voting power. In the case of PoW, only richer users that can afford the sophisticated mining equipment can participate in the protocol, while with PoS, the bias on voting power to favor the richest stakeholders is directly embedded in the protocol. The two systems can be therefore considered almost equivalent in terms of fairness, equity and decentralization, with the remarkable advantage offered by the PoS to save a huge amount of energy.

There is, however, a key difference of interest from the point of view of an economist. While the acquisition of computing power is subject to natural, external factors such as the price of the mining equipment and the cost of energy, the value fluctuations of a PoS-system only depends on market dynamics and on speculative mechanisms. So while with PoW it is a mathematical fact that the cost of an attack increases exponentially with the number of blocks to be changed, and this cost is bounded to a physically enormous amount of energy, the same cost for an attacker of a PoS system is unpredictable, as the cost of the "value-at-stake" for an attacker is not bounded to any external factor.

Arguing that a PoW-based system is therefore more stable and predictable than a PoS-based one, this thesis concludes that a platform that candidates itself to be the most secure one, also from a purely technical point of view, should be PoW-based.

5.3 Critique to the Abuses of the Blockchain: List of Common Pitfalls

A reader that accepts the blockchain defined as the open, verifiable, and immutable shared ledger technology par excellence, immutable by reason of a hard consensus protocol, should also acknowledges it as extremely inefficient [153, 278]. For this reason, the use of the blockchain technology is rarely recommended. One should opt for a different ledger technology whenever possible, and especially for large scale and power constrained distributed networks like the IoT.

The blockchain cornerstones clearly expressed in Definition 5.2.1 have been outlined for the precise purpose of helping engineers and practitioners in understanding when a blockchain is needed, and when is not. Application requirements that conflict with those cornerstones are an indication that another DLT should be preferred in place of a blockchain. For example, all applications that require the registration of users, or data confidentiality, or that can be considered sufficiently secure by reason of stronger trust assumptions, do not need a blockchain. The above logic considerations are illustrated in Figure 5.4, which extends the tradition started by Peck and Wüst [211, 212] to provide a guided path to clearly recognize when a blockchain is not the right choice for an application. The one provided in

this thesis distinguishes itself from previous ones[39] for its limited but precise scope, i.e., to highlight the cases when the blockchain is *not needed*.

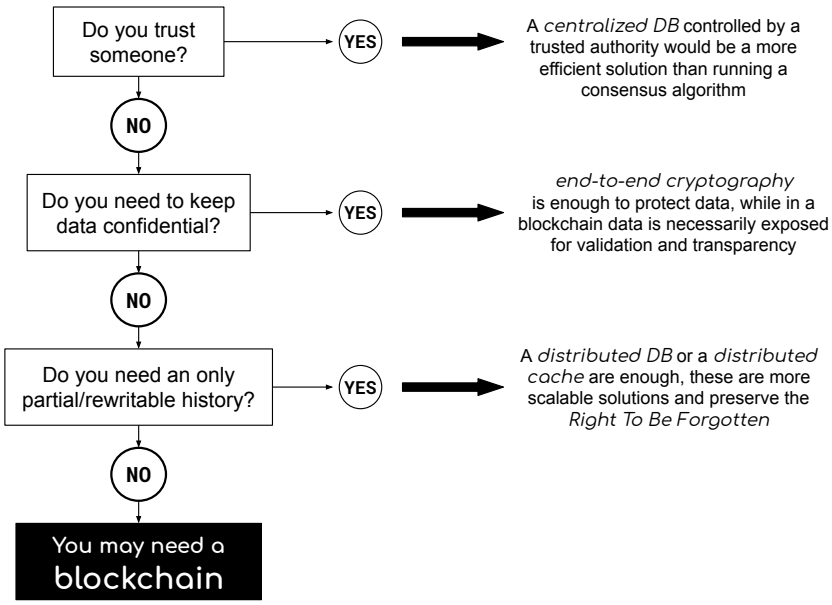


Figure 5.4 – Application requirements and ledger technology: Aid to decision. *Copyright © 2021, IEEE.*

The provided guided path also highlights, implicitly, the common abuses of the blockchain, namely, those recurrent uses of the blockchain for implementing applications whose requirements conflict with the essential blockchain characteristics. The rest of this section is a list of such recurrent pitfalls, to be interpreted as corollaries that derives logically from Definition 5.2.1: these pitfalls are the essence of the critical analysis about the use of the blockchain in general, advanced by this thesis as the ground for a later clarification focused on the role of the blockchain for distributed networks.

The List of Common Pitfalls

Pitfall 5.3.1: *Using the blockchain in presence of strong assumptions on trust.*
Morgen E. Peck first identified this pitfall by analyzing the application of the blockchain to voting [211], although there are several works that propose blockchain-based voting platforms [23–30].

A blockchain, as originally intended to implement anonymous distributed transactions, contrasts the needs of voting. In fact, while it is true that the voter's identity should be kept secret, users cannot be anonymous because their identity must be uniquely determined to ensure uniform eligibility, namely, nobody should be able to cast multiple votes (Sybil attack), *"hence an identity provider is required one way or another"* [280]. Notice that this

[39] a collection of similar decision diagrams is available online [279]

identity provider must be necessarily a centralized accredited institution unless, ad absurdum, someone accept voters providing IDs issued by unrecognised distributed agencies (i.e., fake documents!). This reasoning shows that deciding the eligibility of voters, to check if they have already voted or not, is one key problem of voting whose solution necessarily imposes the existence of an authority. Any solution to this problem cannot be decentralized not even advocating a blockchain. In the same work [280], Heiberg et al. notice also that a blockchain, which is an open, public ledger according to Definition 5.2.1, could not ensure the voters secrecy. This conflict between the blockchain technology and confidentiality is analysed later in Pitfall 5.3.4.

In general, the existence of a trusted third party or the assumption of mutual trust in a distributed network are considered strong assumptions. If a trusted third party exists, this party can play the role of the central coordinator and, by hypothesis, users can rely on it to keep a consistent DB, without going through the overhead generated by a consensus algorithm. From the perspective of the blockchain trilemma, introducing a central trusted authority is equivalent to trade decentralization for scalability. The sacrifices that come with the blockchain are, under this assumption, not necessary. Similarly, assuming mutual trust among nodes there is no need to trade performance for security. ■

Pitfall 5.3.2: *Proposing a fast blockchain.*
A large number of "fast blockchains", able to circumvent the limits of the PoW thanks to alternative consensus protocols, trust models or even data-structure, have been proposed to support various business applications. Some popular platforms advertised as secure and fast include the HyperLedger Fabric, HyperLedger Sawtooth, IOTA, Ripple, Amazon Managed Blockchain and Azure Blockchain followed by many others. All these systems are sometimes also called, collectively, Blockchain-as-a-Service (BaaS) Platforms [281].

This thesis raises the concern that, by blockchain trilemma, the enhanced performance [282] of fast ledgers can be achieved only reducing, partially, either the decentralization or the security of what this thesis defined to be a classic blockchain.

Three exemplary fast platforms, namely, HyperLedger Fabric, HyperLedger Sawtooth and IOTA, will be now studied to highlight the great differences with classic blockchains such as Bitcoin and Ethereum. These differences are highlighted to justify, once more, the restrictive definition of blockchain (Definition 5.2.1) proposed by this thesis. Not calling "blockchain" the fast ledgers is a technical choice but not a negative or positive judgment: this thesis does not discourage the adoption of fast ledgers, rather, their use is promoted whenever a blockchain is not needed (see Figure 5.4 again), especially in distributed networks.

The official documentation of **HyperLedger Fabric** [283] describes its core design. It also highlights that the essential characteristics of public permissionless blockchains, i.e., being *"public networks, open to anyone, where participants interact anonymously"* are problematic

for enterprises, especially because *"the identity of the participants is a hard requirement"* for enterprises to comply with legal obligations such as Know-Your-Customer (KYC) and Anti-Money Laundering (AML) financial regulations. This leads to the identification of a list of requirements for enterprises:

- *Participants must be identified/identifiable.*
- *Networks need to be permissioned.*
- *High transaction throughput performance.*
- *Low latency of transaction confirmation.*
- *Privacy and confidentiality of transactions and data pertaining to business transactions.*

As the HyperLedger Fabric *"has been designed for enterprise use from the outset"* these requirements are all mandatory. The following short analysis of HyperLedger Fabric (Fabric, for short) explains how it supports the listed features.

A membership service associates entities in the network with cryptographic identities. Fabric enables Privacy and Confidentiality through its Channels architecture, where Channels are defined as *"private subnet of communication between two or more specific network members, for the purpose of conducting private and confidential transactions"* [284] and through *private data* [285]. The Fabric documentation also reports this noteworthy consideration:

> *"By relying on the identities of the participants, a permissioned blockchain can use more traditional crash fault tolerant (CFT) or byzantine fault tolerant (BFT) consensus protocols that do not require costly mining."* [286]

The consensus protocols supported by Fabric are indeed traditional consensus protocols of this kind. In particular, the leader-based voting consensus protocol *Raft* [175] is the only non-deprecated protocol for deployments of Fabric. With these features Fabric can reach 20 kTPS [282], outperforming classic blockchains by 4 orders of magnitude.

Observing Fabric it can be noticed how the use of more efficient and traditional consensus mechanisms results in greater performance compared to classic blockchains. Moreover, the security is implemented via traditional access control (permissions) rather than relying on strong consensus mechanisms. The resulting ledger is accessible only to permissioned parties and is vulnerable to collusive attacks lead by a majority of users not defused by any deterrent cost such as the CPU energy of a PoW mechanism. As such, Fabric represents a decentralized platform customized for the enterprise which is more similar to traditional ledgers than to classic blockchain systems such as Bitcoin [287, 288]. For this reason, this thesis proposes to include Fabric in the broader class of the traditional Shared Ledgers.

HyperLedger Sawtooth [289] is an extension of HyperLedger that explicitly requires the Intel SGX framework. Sawtooth promotes the Proof of Elapsed Time (PoET) as a *"solution to the Byzantine Generals Problem that utilizes a trusted execution environment to improve on the efficiency of present solutions such as Proof-of-Work ..."* and *"assumes the use of Intel SGX as the trusted execution environment"* [194]. The idea behind PoET is the following. A random waiting time is distributed to all nodes competing to become the next block miner. When the waiting time expires, the node proves that it waited by providing the PoET generated by its Intel chip. The first node that exhibits a valid PoET is elected as block-miner. This protocol is much more energy efficient since the Intel chip consumes much less than a Bitcoin miner to generate a PoET. However, in this protocol users must trust the server distributing random waiting times and must also trust the proprietary Intel SGX technology, (which has been already attacked successfully multiple times because of its internal architectural flaws [290]). Moreover, if Sawtooth mining would become popular and highly profitable, nothing prevents the start of a mining race with many people buying thousands or more SGX chips to increase the chances of being elected as block-miner, skyrocketing the network power consumption as much as with Bitcoin.

One more system of interest for its special focus to the IoT domain is **IOTA** [291–293], which claims to be an *"Open, Feeless Data and Value Transfer Protocol [that] has fundamentally reengineered distributed ledger technology"*.[40] IOTA, however, does not share with classic blockchains not even the use of a "chain of blocks" for storing transactions and instead adopts a directed acyclic graph (DAG) called, in the IOTA jargon, the "Tangle". The main white paper by S. Popov published to describe IOTA reports in its introduction: *"In this paper we discuss an innovative approach that does not incorporate blockchain technology. This approach is currently being implemented as a cryptocurrency called iota, which was designed specifically for the IoT industry"* [291]. Albeit the clearly announced departure from the blockchain technology, IOTA is included in this short analysis because it candidates itself as competing distributed ledger technology tailored for the IoT, thus is of great interest for this thesis.

As already introduced, IOTA stores transactions in a DAG that is updated by those users that issue new transactions in the system. Users are asked to obey a single rule, i.e., each new transaction must validate (approve) two previous transactions. Considering the typical limited CPU or bandwidth resources of an IoT network, IOTA needs to prevent the excessive generation of transactions that could congest the network or, even worst for tiny devices, could saturate their capacity up to break them. As a countermeasure IOTA mandates new transactions to carry a proof of work as anti-spam and rate controller mechanism, shifting the PoW from block mining to the generation of transactions. IOTA claims to be *Feeless and Free* [293] since it avoids the mining race for generating new blocks of transactions.

[40] A claim well visible on the IOTA website: `https://www.iota.org`

IOTA devices do not receive incentives for creating new transactions. Any kind of device can join the permissionless network of IOTA, so tiny embedded devices connected via kbit/s connections can coexist with powerful computers that access the network trough a Gbit/s connection. The IOTA network heterogeneity leads to a great variance in terms of processing capacity (TPS) and network latency experienced by users. Even if IOTA is meant for IoT devices, these last actually risk to never catch up with the evolution of the DAG and to remain essentially excluded from the IOTA network. To avoid this situation the IOTA foundation maintain a central coordinator that, periodically, issue an empty transaction confirming a subset of the most recent ones, implementing this way a synchronization mechanism and also solving potential divergences among the network participants.

IOTA has been extensively criticized for its architecture and design choices, with the following as recurring and most remarkable critics:

1. The coordinator is a central point of failure and, even more concerning, an absolute authority that can arbitrarily confirm or reject transactions. A large number of attacks[41] have been successfully launched against IOTA exploiting the vulnerability of the coordinator. For example, on the 12th of February 2020 the IOTA foundation had to stop the coordinator, essentially shutting down the entire network, to restore a number of accounts that were compromised by hackers[42].

2. Also Curl-P, the cryptographic function extensively used by IOTA for signing transactions, has been discovered to be flawed [294], making IOTA wallets vulnerable to thefts.

3. The ledger resulting from the IOTA's DAG is expected to either never stabilize into a consistent state or to exclude IoT devices from the ledger because of the high load and varying transaction capacity. This intuition comes from the missing definition of a minimum required capacity[43] to join the network. By Max-Flow Min-Cut theorem [295], in fact, the slowest node will impose its capacity as maximum transaction processing capacity of the network and, considering a modest IoT device, this capacity will be much inferior to the network load. Ultimately, albeit meant for IoT devices, any reasonable load of transactions in the IOTA network can become a barrier for IoT devices and the network must artificially slow down its Transaction rate if IOTA wants to really welcome IoT devices.

[41] A short list of attacks against IOTA is documented online: `https://en.wikipedia.org/wiki/IOTA_(technology)#Attacks`

[42] More on the IOTA shutting down consequent to an hacker attack online: `https://www.zdnet.com/article/iota-cryptocurrency-shuts-down-entire-network-after-wallet-hack`

[43] This limit exists in Bitcoin in form of maximum Block Size and target generation interval, which leads to the famous capacity limit of ≈ 7 Transaction per second.

4. Despite its claim of being *feeless*, IOTA actually requires the inclusion of a PoW in newly generated transactions: this claim is therefore wrong because it ignores energetic costs.

5. A further source of concern is the lack of an incentive mechanism, which makes unclear what kind of stakeholders will bare the costs of keeping the IOTA DAG updated and consistent.

In general, all *fast* or *low-energy* proofs facilitate attacks, so that mining-difficulty must be artificially kept high (as described for Bitcoin in Section 4.1.4) to ensure high level of security. The trilemma warns to beware of fast consensus protocols. They should only be run under strong trust assumptions, namely, in centralized, private platforms with restrictions on user access, thus in non transparent organizations. Under different trust assumption they are instead prone to be easily attacked.

Summing up, applications implemented on top of BaaS platforms cannot be as secure and transparent as if implemented on top of what this thesis defines to be a blockchain. ■

Pitfall 5.3.3: *Validating sensory data through a blockchain.*
This is a common pitfall besetting, for example, all those who choose the blockchain for supply chain management, a quintessential type of IoT application relying on sensor readings and other "things" from the physical world [19–22]. One is lured into using the blockchain for it makes handovers among intermediate dealers manifest, from the producer to the final customer. However, it cannot ensure that the traded goods have been transported correctly. As observed by Wüst et al., the problem with the supply chain lies in the *trust of sensors* or, in other terms, in the way trusted information is acquired from the real world [212]. For example, a malicious truck company that wants to cut costs of transport of refrigerated food, can claim its trustworthiness showing compliance with the laws by installing "trusted" thermometers with GPS. The company can in fact cheat by installing the thermometer in a little empty fridge traveling on the truck together with the rest of the (not refrigerated) load. In general, the "perception layer" of the IoT is the most vulnerable to attacks [296, 297], so that IoT devices (potentially deployed outdoor without supervision) must implement in-hardware mechanisms to be secured. Still no blockchain will ever be able to prevent all possible physical sabotages of sensors.

In general sensors cannot be trusted not just because they can be compromised, but also because of the inescapable *uncertainty of measurements*, independently on the source—whether benign or malicious—of the uncertainty. Consider, for example, a smart contract used to buy train tickets that embeds in its business logic the automatic refund for travelers in the case of train delays above 30 minutes. It is not hard to imagine that the rail company may cheat by reporting (false) delays that are lower than 30 minutes to avoid paying refunds. All applications that rely on measurements, taken by any kind of sensors or IoT

device, at some stage must trust either the centralized company controlling that sensor or a consortium that collected the measurement. When this happens one can refer to *oracles* [298] that must be introduced to obviate the trust problem on sensor by providing trusted information services [299]. However, these oracles are either centralized trusted authorities (see Pitfall 5.3.1), or systems that require distributed consensus, which reintroduce all the issues and trade-offs discussed in Section 4.3.2. With oracles the blockchain looses its meaning as a tool to implement distributed trust. ■

Pitfall 5.3.4: *Proposing a blockchain-based approach for confidentiality.*
Confidentiality, namely, providing data secrecy, is critical for many applications. There is no reason for publishing and recording confidential data on a blockchain, as confidentiality is in clear contrast with one of the key characteristics of the blockchain: transparency. Who accepts Definition 5.2.1 should agree that the blockchain is a transparent technology, as the transparency of a ledger is naturally given by the sum of its openness and completeness.

Works that advocate the use of the blockchain for keeping user data confidential include [40, 300–305] ; since confidentiality contrasts with the public nature of blockchains, a careful justification of design choices is necessary. ■

Pitfall 5.3.5: *Storing sensitive information on the blockchain.*
Registering user credentials and account information on a blockchain is an irreversible operation, as data cannot be deleted. Services implemented on top of a blockchain will not be able to delete user data; not even upon legitimate request. This could be an issue for a user that wants to abandon a service and also for authorities that need to enforce the "Right to be Forgotten," which is a legal provision in several countries [262, 306–308]. ■

Pitfall 5.3.6: *Verifying the authenticity of digital documents or real goods with a blockchain.*
Many proposals concern the use of the blockchain as a decentralized platform to store digitally signed documents, i.e., certificates [309–313]. One might be drawn to believe that, as it is part of a blockchain, that certificate is authentic. However, the authenticity of a certificate is guaranteed by its digital signature, which depends on a trusted authority external to the blockchain and not subject to the trust obtained via a consensus protocol. For example, let us consider the following certificate digitally signed by an institution I and recorded in a blockchain: "Alice passed exam E after attending the course provided by Institution I on date D." The fact that this certificate belongs to a blockchain does not in any way prove that Alice really attended a course and passed an exam, and legitimate doubts can also arise about the timestamped date D. Trusting the authenticity of this document only relies on unconditionally trusting the issuer, namely, institution I. Overall, the blockchain can be an evidence of a chronological series of valid steps of a process but, for non transactional data like common certificates, patents and proofs of authorship, the blockchain alone cannot provide any form of authenticity or validity.

This holds also for identity documents, as argued in Pitfall 5.3.1, and for real goods too. Somehow, in fact, many have been inspired by the immutability property of the blockchain technology to guarantee the authenticity of real goods [19–21, 314–317]. The common underlying idea is to associate any good with a digital identifier (e.g., a QR code) tracked by a blockchain so that a final customer receiving the good can use the attached identifier as a proof of authenticity. Unluckily, there is no mechanical tool to make objects of the real world unforgeable: for as much as the identifier on the blockchain is immutable (unforgeable), this is not true for the associated good, which can be physically replaced with another of lower quality. Once again, the *digital signature* is the technology for verifying the authenticity of transactions or digital documents but the blockchain alone, instead, cannot prove the authenticity of any product or certificate. ■

Pitfall 5.3.7: *Forgetting the cost of mining.*
One can think to the blockchain as a means to enable transactions without additional fees. This is true, from the user perspective, as long as an incentive mechanism encourage validators in processing transactions, but becomes false as soon as this incentive mechanism terminates. Any newly proposed application of the blockchain should describe a sound incentive mechanism for miners without forgetting the cost of transactions: neglecting this cost leads projects to failure, as no actors of the system will bear the cost of mining. ■

Pitfall 5.3.8: *Implementing a memoryless process on top of a blockchain.*
In a typical blockchain system the full chain of transactions must be recorded for validating new transactions. For those applications designed on top of Markovian assumptions, i.e., where only the knowledge of the *current state* of the system is necessary to make progress, using a blockchain will keep recording (possibly too many) unnecessary data. For example, there is no need for a smart home IoT application to memorize the full history of the temperature of a room to decide which heat source to activate or disable at the current time. Even accounting and billing does not require the entire history, but only a previous reading and recent invoicing. ■

Pitfall 5.3.9: *Claiming to jointly provide security, decentralization and high performance.*
This last pitfall is provided to conclude stressing, once more, on the relevance of the blockchain trilemma. Given the current state of science and technology, the proposals claiming joint provision of maximum security, decentralization and performance violate the limits of distributed systems, and cannot work as promised. ■

5.4 Blockchain in Support of the IoT

The critical analysis on the uses of the blockchain has been broken down in two sections: the first devoted to the definition of the very term “blockchain” (Section 5.2), while the second

one criticized the common misuses when applying a so defined technology (Section 5.3). This clarification effort, subjected to the trade-offs imposed by the blockchain trilemma (Figure 4.15), did not hint so far a viable strategy for the *integration* of the blockchain within large scale and low power distributed networks like, e.g., the Internet of Things (IoT). Focusing now on the IoT this thesis argues, however, that the blockchain can serve as an *external service* that processes and manages specific subsets of distributed transactions, this way representing an alternative or complementary paradigm to centralized cloud services.

The "bubblechart" of Figure 5.5 is introduced to support the claim that it is not advisable to *integrate* the blockchain in large scale and low power networks, while an *external* role of the blockchain for the IoT can be envisioned. Figure 5.5 draws a multidimensional overview of the consensus mechanisms surveyed in Section 4.4 according to the following four dimensions:

- The strength of the trust assumptions, which is inversely proportional to the degree of security (x-axis).
- The openness of consensus, an indicator of the degree of democracy, from one (tyranny) to all (power to the people) (y-axis).

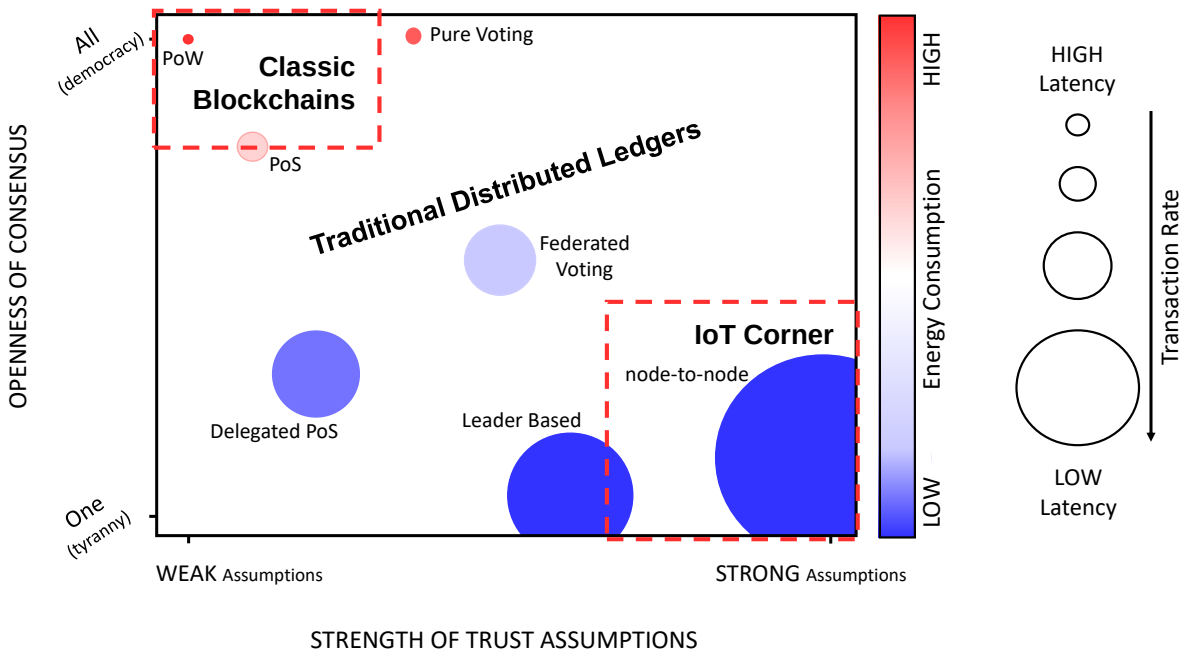


Figure 5.5 – Bubblechart of relevant building blocks for Shared Ledgers. The diagram compares solutions as a function of trustworthiness of users (x-axis) and openness of the consensus protocol (y-axis). The color and the size of each bubble offer a quick indication of energy consumption and transaction rate of each consensus protocol, respectively. *Copyright © 2021, IEEE.*

- Energy consumption (color of each bubble: red when high or blue if low).
- The transaction rate (size of each bubble: the larger the faster). The transaction rate can be considered also an indicator of transaction latency because, fixing the number of transactions registered with a single operation (block-size), a lower latency implies a higher transaction rate.

The figure enriches the trilemma by breaking down the "scalability vertex" (Figure 4.15) into two distinct dimensions, i.e., energy consumption and transaction rate. Furthermore, the decentralization and security properties are renamed to more accurately capture the fundamental trade offs arising in the landscape of the Shared Ledger technologies. The ideal "blockchain-for-IoT-bubble" would be a blue and large one, thus low power and very performing, in the top-left corner of the chart, which is the most decentralized and most secure sector.

Blockchain systems are naturally located in the top-left corner, characterized by being democratically open and secure despite weak trust assumptions, but also extremely slow and resource hungry. The opposite corner is where IoT applications reside, with their scalability requirements, tight resource constraints, and high global transaction rates. This corner also highlights that the ultimate participants are "things" rather than humans, thus affording a higher degree of trust, at least towards some other parts of the system. As such, Figure 5.5 pictorially conveys a crucial observation: classic blockchains are in contrast with IoT requirements, which keeps the two realms well separated.

Despite the lack of space for blockchain systems in the bottom-right corner, the bubblechart helps in identifying the *node-to-node consensus* (recall the Section 4.4.3 of the brief review on consensus protocols reported by this thesis) as a means to build trust among operators/systems without established relations. The key word, here, is *node-to-node*, which restricts the distributed consensus problem to few nodes, usually a couple, although extensions to small numbers is efficiently conceivable. To settle a transaction it is sufficient for the transacting parties to agree on a transaction protocol, and this agreement can be reached privately by the two or few more parties in any fashion. What makes node-to-node consensus appealing for IoT is its efficient support of *local consensus*, which is natural for many IoT applications such as those with groups of sensors or a platoon of vehicles.

The number of different node-to-node consensus protocols is limited only by imagination. Specific transitive properties, i.e., how and to what extent if node A trusts node B and node B trusts node C, then node A can trust node C, can be defined to be applied to large clusters of trusted entities, ultimately leading to a network of diverse but interoperating "channels": in some sense, to the IoT itself. The term "channel" is inherited from the world of cryptocurrencies (*Networks of Payment Channels* [318–322]) and from that of communication networks, where a network is a set of channels interconnecting nodes. Since IoT transactions

are not necessarily monetary, the generic term *Transaction Channels* will be used in the rest of the thesis.

5.4.1 Networks of Transaction Channels

Transaction Channels are all those techniques used to group off-chain transactions between the same small group of users, for the reasons of speeding them up and avoid paying multiple transaction fees. A Transaction Channel is therefore a node-to-node consensus protocol where the two transacting parties establish a fast "payment" method, and agree to postpone the clearing of the transactions' balance. Recording the status of the channel on the blockchain can be periodic or event-based. What is stored in the blockchain is a meaningful representation of the history of transactions on the channel. For example, this could be the stochastic representation of a long-term distributed measure. It could also be the amount of energy exchanged in a smart grid, or between electrical vehicles and the grid supposing that the battery of the vehicles can be used by the grid for short-term accumulation. Many other could be the further possible example, not mentioned here for brevity.

The most notable implementation of Transaction Channels is the Lightning Network [198], which scales up the technique to a full network of such channels. This allow payments to be routed between remote end-points thanks to a dedicated routing protocol for payments, with the aim of providing a globally scalable mechanism for fast transactions. The Lightning Network is "Bitcoin oriented," but the concept of a network of payment channels may become the transaction paradigm to enable a global market at the IoT scale.

It also opens the way to thrilling research challenges, such as bringing network science and expertise into the domain of transporting and routing payments within Payment Networks, as initially explored in [323]. Major open problems include addressing the depletion of channel capacity, especially for the most loaded nodes in the center of the network, developing enhanced centrality-aware routing strategies [17,18], and rebalancing techniques [324–326].

5.4.2 Role of the Blockchain in the IoT

Figure 5.6 illustrates the vision proposed by this thesis, i.e., the IoT empowered by Networks of Transaction Channels. These latter are deployed at the IoT layer, where classic blockchains play as supporting external ledgers similarly to the use of the Bitcoin blockchain in the Lightning Network.

This vision is justified by the fact that most IoT applications will have *local* relevance. For example, domotic devices will intercommunicate mostly only over a house local network. Similarly, agricultural sensors for precision farming will mostly communicate with each other and convey the local information to a gateway controlled by the farmer. Industrial

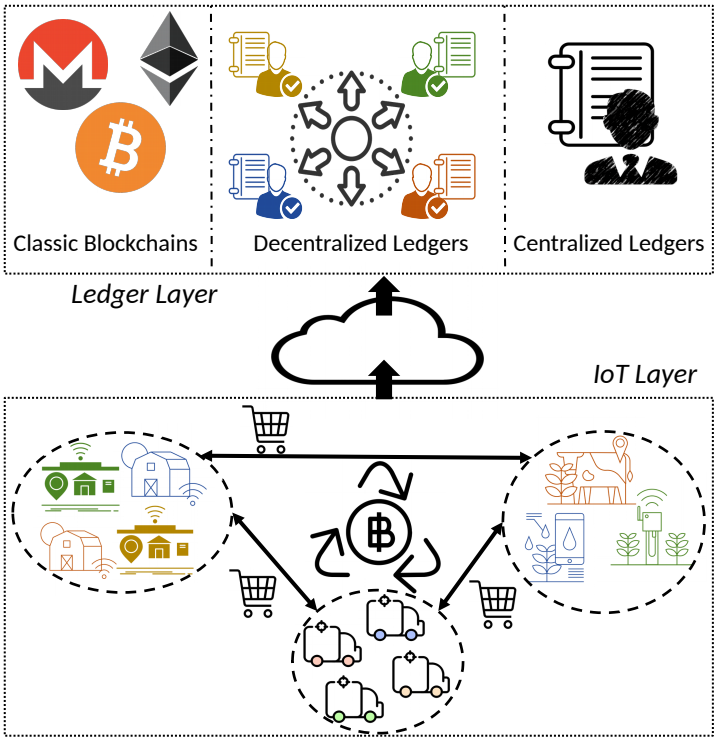


Figure 5.6 – Different IoT clusters, made of devices managed either by private (home-/enterprises) or public (institutional) entities, perform most transactions locally, in the IoT layer. The different IoT clusters access intermediary platforms for interoperability purposes. Blockchain-based platforms can provide specific services at this level, in line with their characteristic capabilities of granting security and immutability even in the absence of trust. *Copyright © 2021, IEEE.*

IoT often requires high levels of privacy and confidentiality, clearly in contrast with the open, immutable nature of a blockchain; vehicular networks and intelligent transportation systems may require transactions with latency smaller than a few milliseconds, and rates in the order of kTPS per vehicle, again in full contrast with the characteristics of blockchains. IoT transactions can be therefore assumed to be local and lightweight in nature, calling for likewise local and lightweight solutions implemented by the platform that should process them.

From time to time, separate IoT domains, platforms and applications may need to carry out and record transactions with a global, final, and immutable scope. At this level blockchains can play a relevant role, freeing IoT systems from the need to subscribe to a global, centralized, expensive, trust-based service whose security and reliability have well-known limitations. Using blockchains externally would therefore bring added value to the IoT domain, responding to its requirements of extending beyond local, context-limited applications when needed.

Chapter 6

Conclusion

This thesis proposed two contributions for the development of distributed networks. The first is a novel algorithm for the computation of the Load Centrality (LC) index that enables a strategy for tuning the frequency of control messages in Distance-Vector (DV) routing protocols. The second is the critical discussion about the integration of the blockchain in distributed networks. Concluding remarks and further reflections on the many, intriguing research questions left open are drawn in Sections 6.1 and 6.2 that focus, respectively, on the two main topics investigated in this thesis.

6.1 Centrality based Optimization of Routing Protocols

This thesis introduces a novel, distributed algorithm for the exact computation of the LC index in a generic graph [17]. The LC coincide with the Betweenness Centrality (BC) in networks that do not support multipath, which is the case for most communication networks. It can therefore replace the BC to enable the large plethora of BC-based applications already known for centralized networks also in distributed ones. It is worth to notice that, in the case of multipath, the LC captures the network dynamics better than the BC, so its use may be further encouraged.

This thesis also proves analytically and verifies with computer simulations that the algorithm converges in a number of time steps that is proportional to the diameter of the network. Chapter 3 explains how the algorithm is designed as a minimal extension of the Bellman-Ford one, thus it can be easily integrated into existing DV routing protocols. This claim is supported by reporting the details of a concrete implementation of the algorithm in Babel, a well known DV protocol. This thesis provides also an example of the algorithm potential applications in real world networks. For instance, by emulating several real networks, it is shown that the losses consequent to a node-failure can be tangibly reduced if the timers of Babel are optimized according to a Pop-Routing approach based on LC.

A second, more sophisticated version of the distributed algorithm has been developed to make the same algorithm *incrementally deployable*, i.e., applicable also in networks where only a subset of routers has been upgraded to run the new algorithm for LC [18]. Section 3.3.2 reports the mathematical study that characterizes the approximation error performed in a partial deployment scenario when not all nodes contribute to the computation of centrality. Unfortunately this study reveals that the error slowly converges to zero while growing the ratio of upgraded nodes deployed in the network. However, it turns out that the two nodes rankings produced by the estimated and by the theoretical centrality values are highly correlated even when only few upgraded nodes are deployed and estimate their centrality. In other words the incrementally deployable algorithm accurately ranks nodes by their theoretical centrality even if the ratio of upgraded nodes is small. This result encourages the application of the Pop-Routing strategy also in early deployment stages.

6.1.1 Future Work

While the Pop-Routing approach has been applied successfully, so far, to the optimal tuning of the timers that regulate the generation of control messages, the application of a similar technique to other timers has not been explored yet. The same strategy could be explored for tuning other timers influencing the performance of networking protocols, for example, on packets aggregation and expiry timers.

Expiry timers are used to implement different functions of routing protocols. They are used, e.g., to implement the garbage-collection of routing table entries that have not been refreshed on time or to control the automatic retransmission of sent messages that have not been acknowledged by the receiver. Packets aggregation timers, instead, play a fundamental role especially when communications are bursty or the communication channel is shared. A performance evaluation of the Pop-Routing approach applied to such timers, above all to those that implement the automatic-retransmission functionality of wireless MAC protocols, becomes therefore an intriguing research direction for future works.

Moving to networks at the scale of Internet there are two more timers whose regulation is debated since decades: these two are the MRAI and the RFD timers defined by the Border Gateway Protocol (BGP). MRAI stays for *Minimum Route Advertisment Interval* and it is the amount of time that a BGP speaker must wait before transmitting a new UPDATE for the same prefix, even if a new route became available in this interval of time. Different ASes administrators do not cooperate for tuning it, so that some administrators configure their routers with the default —quite high— value of 30 seconds, while many others choose different lower values. Still, configurations are always static, far from being optimal nor even dynamic. The RFD acronyms instead stays for *Route Flap Dumping* and is a controversial

method used to prevent the oscillations of routes by suppressing, for some time, repeated BGP UPDATES.

The research community still did not find an agreement on how to calibrate these timers so to achieve an optimal trade-off between route convergence speed and control overhead. The difficulty of finding this trade-off is witnessed by the historical fact that through the last 20 or more years the recommendations provided by the IETF on the use of both MRAI and RFD changed multiple times, sometimes even proposing to completely suppress their use.

Considering that BGP is a distributed Path-Vector (PV) protocol based on Bellman-Ford, sharing a working principle similar to the one of DV routing protocols, then the LC based optimization presented in this thesis for Babel seems possible for BGP as well. The exploration of the Pop-Routing approach at the Internet scale on top of BGP appears to be, therefore, an exciting direction for further research.

Many more applications beyond the tuning of different timers can benefit from the distributed computation of the LC for all nodes of a graph, or just from partial nodes ranking. Whenever the nodes position in the network is strategic researchers and engineers can consider the use of the distributed LC algorithm to implement an optimal strategy for the allocation of resources. The algorithm may enable, for example, an optimal strategy for the placement of cached contents in Content Delivery Networks but this is just one of the potential, further, unexplored applications.

This thesis makes available a distributed algorithm for centrality that, wishfully, will be exploited for the development of many new applications for distributed networks.

6.2 Blockchain for Securing Distributed Networks

This thesis argues that the blockchain is not the appropriate technology for securing large scale, power constrained networks, which is especially true for the world of the Internet of Things (IoT). For the purpose of clarifying the possible role of the blockchain in securing distributed networks this thesis started from the analysis of the blockchain from a distributed system perspective, stressing on the importance of distributed consensus protocols.

The thesis raises the concern that consensus protocols driven by stake rely completely on the rationality assumption of their users and, compared to traditional voting based protocols, lack of mathematical stability properties. Moreover, it highlights that the voting power has a tendency to consolidate in the hands of few great stakeholders under Proof of Stake (PoS) and also under Proof of Work (PoW). This thesis therefore suggests to consider both proof mechanisms as *census suffrage* systems that exhibit an almost equivalent degree of decentralization. Section 5.2.3 points out that the cost of a PoW is stabilized by the real world cost of energy and mining equipment, while the cost of a PoS only depends on the

virtual and fluctuating "value-at-stake". This thesis concludes that the PoW, being more stable than the PoS, offers a higher degree of security and its cryptographical hardness is the key technical element supporting the immutability property of blockchains.

The thesis stresses on the innovative and peculiar aspects of permissionless blockchains, the only free from any trusted authority, while downplays permissioned ledgers to traditional managed data bases. The main conclusion is that the term "blockchain" should be reserved to those platforms characterized by: *i)* Openness to anonymous users; *ii)* Public and complete history of transactions, and *iii)* Safeguarded by a hard consensus protocol.

This definition has far-reaching consequences. Above all, the constitutional hard consensus protocol necessarily brings high power consumption and degrades the rate and the latency of transactions. For this reason, the blockchain is not advised to support popular applications such as E-voting and Supply Chain Management. Furthermore, not to violate the Right to Be Forgotten, the blockchain can hardly support Identity Management applications nor work as archive of certificates and other sensitive information.

Lastly, this thesis recommends a wiser use of the blockchain in distributed networks promoting the paradigm of Transaction Channels. This paradigm enables a high-rate of inexepensive, off-chain, local transactions, while leaves to users the possibility to interact through more rare and expensive cross-platform transactions, these latter recorded in a global, blockchain-based, shared ledger. This paradigm better suits the typical IoT requirements and free users from the need of a centralized, trusted ledger service (banks), still supporting transactions across diverse applications, technologies and administrative domains.

6.2.1 Future work

Although this thesis downplayed the role of the blockchain in distributed networks it also outlined open problems and intriguing research directions about the same blockchain in networking.

A first direction has been introduced while presenting Transaction Channels as the ideal tool to support local transactions in small portions of a network, where one such portion could be an IoT cluster, an ad hoc or a community network. However, the technology that supports networks of Transaction Channels is in its infancy and only in recent times projects such as the Lightning Network [198] provided, for the first time, a concrete implementation of a payment network linked to a main blockchain. There is still space for further research towards the advancement of the technology underpinning these projects, with an important contribution from the networking research community that would be key to address the well known problems affecting the routing of payments.

A few research pioneers already started transferring the networking expertise to payment networks, designing new routing protocols for payments that embed the well established

routing mechanisms used since decades in computer networks [318–321]. These mechanisms should be reintroduced for, e.g., routing payments around those channels that are depleting their credit or for jointly reduce the end-to-end payment latency while maximizing the fairness of the path selection process. The problem of a fair selection of paths is important to balance the network traffic without insisting on the most central nodes that, otherwise, would be overloaded and would rapidly deplete their credit.

For all these purposes there is the need of disseminating the channel balances, to be used for steering the routing process. A further open research direction is therefore the development of new distributed protocols able to limit the propagation scope of the channels balance information, safeguarding the privacy of users as much as possible. Preliminar works [323–326] addressing the aforementioned challenges are opening the way to further research efforts.

Another research challenge is to increase the security of blockchains by, e.g:

1. speeding up the gossip protocols used to propagate new blocks among validators
2. understanding and fixing the vulnerabilities of the networking infrastructure that expose blockchains to eclipse attacks.

These two ideas arise looking back to Section 4.2.4 that pointed out how an attacker can easily enhances his probability of success exploiting propagation delays and influencing the communications of his victims. For example, adopting the stale rate as main indicator of the blockchain security as suggested by Gervais et al. [134], this thesis pointed out the existing proportionality between the stale rate and the Block Propagation Time (B_p). This observed proportionality (see Observation 4.3) suggests that an enhancement of the gossip protocols responsible for the propagation of blocks may be decisive to strengthen the security of blockchains. Optimization of gossip protocols based on centrality are already known in the literature, e.g., for minimizing the delays while distributing chunks of videos in real time streaming applications [327]. One novel research direction could be the investigation of similar centrality based strategies for minimizing B_p, thus to enhance the blockchain security.

Section 4.2.4 also directed the attention to the γ parameter, a parameter commonly interpreted as the ability of a selfish miner (SM) to take control of the network and influence communications. The higher is γ the higher is the exposure of a blockchain to attacks. The γ parameter can be well approximated by network centrality indexes, since their purpose is exactly to capture the nodes influence on the network communications. In the future the here introduced algorithm for the computation of centrality could be used to asses the security level of the communication networks used by blockchains, reporting warnings when extremely high centrality values are detected.

These last two lines of research are inspired by the blending of the two main contributions of this thesis.

List of Abbreviations

AS Autonomous System

ASN Autonomous System Number

BC Betweenness Centrality

BGP Border Gateway Protocol

CHF Cryptographic Hash Function

CN Community Network

DB Data Base

DDoS Distributed Denial of Service

DES Discrete Event Simulator

DISP Distributed Internet Service Provider

DLT Distributed Ledger Technology

DNS Domain Name System

DPoS Delegated Proof of Stake

DV Distance-Vector

IoT Internet of Things

ISP Internet Service Provider

KYC Know-Your-Customers

LC Load Centrality

LS Link-State

LSA Link-State-Advertisement

OS Operating System

P2P Peer-to-Peer

PBFT Practical Byzantine Fault Tolerance

PKI Public Key Infrastructure

PoN Proof of Networking

PoS Proof of Stake

PoW Proof of Work

PV Path-Vector

RIR Regional Internet Registry

RPKI Resource Public Key Infrastructure

SL Shared Ledger

SSH Secure SHell

SSL Secure Sockets Layer

TLV type-length-value

TPS Transactions per Second

VRF Verifiable Random Function

WMN Wireless Mesh Network

List of Figures

List of Tables

Bibliography

[1] M. B. Gaved, "An Investigation Into Grassroots Initiated Networked Communities As A Means Of Addressing The Digital Divide" Ph.D. dissertation, The Open University, Milton Keynes, UK, sep 2011.

[2] P. Baran, "On Distributed Communications Networks" in *IEEE Trans. Commun. Syst.*, vol. 12, no. 1, pp. 1–9, 1964.

[3] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey" in *Elsevier Computer Networks*, vol. 47, no. 4, pp. 445–487, mar 2005.

[4] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks" in *IEEE Commun. Magazine*, vol. 43, no. 9, pp. S23–S30, sep 2005.

[5] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: commodity multihop ad hoc networks" in *IEEE Commun. Magazine*, vol. 43, no. 3, pp. 123–131, mar 2005.

[6] J. Y. Yu and P. H. J. Chong, "A Survey of Clustering Schemes for Mobile Ad Hoc Networks" in *IEEE Commun. Surveys Tuts.*, vol. 7, no. 1, pp. 32–48, may 2005.

[7] H. Hartenstein and L. P. Laberteaux, "A Tutorial Survey on Vehicular Ad Hoc Networks" in *IEEE Commun. Magazine*, vol. 46, no. 6, pp. 164–171, jun 2008.

[8] L. Maccari and R. Lo Cigno, "A Week in the Life of Three Large Wireless Community Networks" in *Elsevier Ad Hoc Networks*, vol. 24, pp. 175–190, jan 2015.

[9] R. Lo Cigno and L. Maccari, "Urban Wireless Community Networks: Challenges and Solutions for Smart City Communications" in *Proc. of the ACM Int. Workshop on Wireless and Mobile Technologies for Smart Cities (WiMobCity '14)*, pp. 49–54, Philadelphia, PA, USA, aug 2014.

[10] C. Fuchs, M. Michalis, and D. Boucas, "The Multiple Aspects of Politics of Sustainability in Community Networks: Definitions, Challenges, and Countermeasures (D2.1)" https://tinyurl.com/netcommonsD21, jun 2016.

[11] D. Boucas, M. Michalis, and C. Fuchs, "The Multiple Aspects of Politics of Sustainability in Community Networks: Definitions, Challenges, and Countermeasures (D2.2)" https://tinyurl.com/netcommonsD22, jan 2017.

[12] **L. Ghiro**, L. Maccari, and R. Lo Cigno, "Proof of Networking: Can Blockchains Boost the Next Generation of Distributed Networks?" in *Proc. of the 14th IEEE Conf. on Wireless On-demand Netw. Syst. and Services (WONS'18)*, pp. 29–32, Isola 2000, France, feb 2018.

[13] L. Maccari and R. Lo Cigno, "Pop-Routing: Centrality-based Tuning of Control Messages for Faster Route Convergence" in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM'16)*, San Francisco, CA, USA, apr 2016.

[14] L. Maccari and R. Lo Cigno, "Improving Routing Convergence With Centrality: Theory and Implementation of Pop-Routing" in *IEEE/ACM Trans. on Networking*, vol. 26, no. 5, pp. 2216–2229, oct 2018.

[15] U. Brandes, "A Faster Algorithm for Betweenness Centrality" in *Taylor & Francis J. of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, jun 2001.

[16] J. Chroboczek, "The Babel Routing Protocol" RFC 6126, apr 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6126.txt

[17] L. Maccari, **L. Ghiro**, A. Guerrieri, A. Montresor, and R. Lo Cigno, "On the Distributed Computation of Load Centrality and its Application to DV Routing" in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM'18)*, Honolulu, HI, USA, apr 2018.

[18] L. Maccari, **L. Ghiro**, A. Guerrieri, A. Montresor, and R. Lo Cigno, "Exact Distributed Load Centrality Computation: Algorithms, Convergence, and Applications to Distance Vector Routing" in *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1693–1706, jul 2020.

[19] M. Montecchi, K. Plangger, and M. Etter, "It's real, trust me! Establishing supply chain provenance using blockchain" in *Elsevier Business Horizons*, vol. 62, no. 3, pp. 283–293, may 2019.

[20] D. Tse, B. Zhang, Y. Yang, C. Cheng, and H. Mu, "Blockchain Application in Food Supply Information Security" in *Proc. of the IEEE Int. Conf. on Industrial Eng. and Eng. Manag. (IEEM'17)*, pp. 1357–1361, Singapore, dec 2017.

[21] F. Tian, "An agri-food supply chain traceability system for China based on RFID and blockchain technology" in *Proc. of the 13th IEE Int. Conf. on Service Syst. and Service Manag. (ICSSSM'16)*, pp. 1–6, Kunming, China, jun 2016.

[22] D. Miller, "Blockchain and the Internet of Things in the Industrial Sector" in *IEEE IT Professional*, vol. 20, no. 3, pp. 15–18, may 2018.

[23] N. Kshetri and J. Voas, "Blockchain-Enabled E-Voting" in *IEEE Software*, vol. 35, no. 4, pp. 95–99, jul 2018.

[24] A. Ben Ayed, "A Conceptual Secure Blockchain Based Electronic Voting System" in *AIRCC Int. J. of Netw. Security and Its Appl.*, vol. 9, no. 3, pp. 01–09, may 2017.

[25] S. Bistarelli, M. Mantilacci, P. Santancini, and F. Santini, "An End-to-end Voting-system Based on Bitcoin" in *Proc. of the ACM Symp. on Applied Comput. (SAC'17)*, pp. 1836–1841, Marrakech, Morocco, apr 2017.

[26] Y. Liu and Q. Wang, "An E-voting Protocol Based on Blockchain" in *IACR Cryptology ePrint Archive*, vol. 2017, no. 1043, 2017.

[27] F. Sheer Hardwick, A. Gioulis, R. Naeem Akram, and K. Markantonakis, "E-Voting With Blockchain: An E-Voting Protocol with Decentralisation and Voter Privacy" in *Proc. of the IEEE Int. Conf. on Internet of Things (iThings), Green Comput. and Commun. (GreenCom), Cyber, Physical and Social Comput. (CPSCom) and Smart Data (SmartData)*, pp. 1561–1567, Halifax, NS, Canada, jul 2018.

[28] B. Wang, J. Sun, Y. He, D. Pang, and N. Lu, "Large-scale Election Based On Blockchain" in *Elsevier Procedia Computer Science*, vol. 129, pp. 234–237, 2018.

[29] R. Qi, C. Feng, Z. Liu, and N. Mrad, "Blockchain-Powered Internet of Things, E-Governance and E-Democracy" in *E-Democracy for Smart Cities*, T. Vinod Kumar, Ed. Springer, may 2017, pp. 509–520.

[30] P. Noizat, "Blockchain Electronic Vote" in *Elsevier Handbook of Digital Currency*, may 2015, pp. 453–461.

[31] K. Gai, Y. Wu, L. Zhu, L. Xu, and Y. Zhang, "Permissioned Blockchain and Edge Computing Empowered Privacy-Preserving Smart Grid Networks" in *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 7992–8004, oct 2019.

[32] F. Lombardi, L. Aniello, S. De Angelis, A. Margheri, and V. Sassone, "A Blockchain-based Infrastructure for Reliable and Cost-effective IoT-aided Smart Grids" in *Proc. of the IET Conf. Living in the Internet of Things: Cybersecurity of the IoT*, London, UK, mar 2018.

[33] M. Mylrea and S. N. G. Gourisetti, "Blockchain for Smart Grid Resilience: Exchanging Distributed Energy at Speed, Scale and Security" in *Proc. of the IEEE Resilience Week (RWS'17)*, pp. 18–23, Wilmington, DE, USA, sep 2017.

[34] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: towards sustainable local energy markets" in *Springer Comput. Sci. Res. Dev.*, vol. 33, no. 1-2, pp. 207–214, aug 2017.

[35] E. Munsing, J. Mather, and S. Moura, "Blockchains for Decentralized Optimization of Energy Resources in Microgrid Networks" in *Proc. of the IEEE Conf. on Control Technol. and Appl. (CCTA'17)*, pp. 2164–2171, Mauna Lani, HI, USA, aug 2017.

[36] A. Hahn, R. Singh, C.-C. Liu, and S. Chen, "Smart Contract-based Campus Demonstration of Decentralized Transactive Energy Auctions" in *Proc. of the IEEE Power and Energy Soc. Innovative Smart Grid Technol. Conf. (ISGT'17)*, Arlington, VA, USA, apr 2017.

[37] A. Laszka, A. Dubey, M. Walker, and D. Schmidt, "Providing Privacy, Safety, and Security in IoT-BasedTransactive Energy Systems using Distributed Ledgers" in *Proc. of the 7th ACM Int. Conf. on the Internet of Things (IoT'17)*, Linz, Austria, oct 2017.

[38] H. Yan, B.-B. Huang, and B.-W. Hong, "Distributed Energy Transaction Pattern and Block Chain Based Architecture Design" in *DEStech Trans. Environment, Energy Earth Sci*, feb 2018.

[39] J. Xu, K. Xue, S. Li, H. Tian, J. Hong, P. Hong, and N. Yu, "Healthchain: A Blockchain-Based Privacy Preserving Scheme for Large-Scale Health Data" in *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 8770–8781, oct 2019.

[40] A. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, "A Decentralized Privacy-Preserving Healthcare Blockchain for IoT" in *MDPI Sensors*, vol. 19, no. 2, p. 326, jan 2019.

[41] M. Mettler, "Blockchain technology in healthcare: The revolution starts here" in *Proc. of the 18th IEEE Int. Conf. on e-Health Networking, Applications and Services (HEALTHCOM'18)*, pp. 1–3, Munich, Germany, sep 2016.

[42] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control" in *Springer J. of Medical Syst.*, vol. 40, no. 10, aug 2016.

[43] L. Cocco, A. Pinna, and M. Marchesi, "Banking on Blockchain: Costs Savings Thanks to the Blockchain Technology" in *MDPI Future Internet*, vol. 9, no. 3, p. 25, jun 2017.

[44] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry" in *Springer Financial Innovation*, vol. 2, no. 1, dec 2016.

[45] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-Preserving Support Vector Machine Training Over Blockchain-Based Encrypted IoT Data in Smart Cities" in *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 7702–7712, oct 2019.

[46] S. Ibba, A. Pinna, M. Seu, and F. E. Pani, "CitySense: blockchain-oriented smart cities" in *Proc. of the XP2017 Scientific Workshops (XP'17)*, pp. 1–5, Cologne, Germany, may 2017.

[47] A. S. Patil, B. A. Tama, Y. Park, and K.-H. Rhee, "A Framework for Blockchain Based Secure Smart Green House Farming" in *Springer, Advances in Computer Science and Ubiquitous Computing*, pp. 1162–1167, dec 2017.

[48] K. Liu, W. Chen, Z. Zheng, Z. Li, and W. Liang, "A Novel Debt-Credit Mechanism for Blockchain-Based Data-Trading in Internet of Vehicles" in *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 9098–9111, oct 2019.

[49] T. Jiang, H. Fang, and H. Wang, "Blockchain-Based Internet of Vehicles: Distributed Network Architecture and Performance Analysis" in *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4640–4649, jun 2019.

[50] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-Based Decentralized Trust Management in Vehicular Networks" in *IEEE Internet of Things J.*, vol. 6, no. 2, pp. 1495–1505, apr 2019.

[51] A. Patel, N. Shah, T. Limbasiya, and D. Das, "VehicleChain: Blockchain-based Vehicular Data Transmission Scheme for Smart City" in *Proc. of the IEEE Int. Conf. on Syst. Man. and Cybern. (SMC'19)*, pp. 661–667, Bari, Italy, oct 2019.

[52] M. Cebe, E. Erdin, K. Akkaya, H. Aksu, and S. Uluagac, "Block4Forensic: An Integrated Lightweight Blockchain Framework for Forensics Applications of Connected Vehicles" in *IEEE Commun. Magazine*, vol. 56, no. 10, pp. 50–57, oct 2018.

[53] R. A. Michelin, A. Dorri, M. Steger, R. C. Lunardi, S. S. Kanhere, R. Jurdak, and A. F. Zorzo, "SpeedyChain: A framework for decoupling data from blockchain for smart cities" in *Proc. of the 15th EAI Int. Conf. on Mobile and Ubiquitous Syst.: Comput., Netw. and Services (MobiQuitous'18)*, pp. 145–154, New York, NY, USA, nov 2018.

[54] X. Zhang, R. Li, and B. Cui, "A security architecture of VANET based on blockchain and mobile edge computing" in *Proc. of the 1st IEEE Int. Conf. on Hot Information-Centric Networking (HotICN'18)*, pp. 258–259, Shenzhen, China, aug 2018.

[55] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "CreditCoin: A Privacy-Preserving Blockchain-Based Incentive Announcement Network for Communications of Smart Vehicles" in *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2204–2220, jul 2018.

[56] S. Rowan, M. Clear, M. Gerla, M. Huggard, and C. M. Goldrick, "Securing Vehicle to Vehicle Communications using Blockchain through Visible Light and Acoustic Side-Channels" in *arXiv preprint arXiv:1704.02553*, apr 2017.

[57] P. K. Sharma, S. Y. Moon, and J. H. Park, "Block-VN: A Distributed Blockchain Based Vehicular Network Architecture in Smart City" in *KIPS Journal of Information Processing Systems*, vol. 13, no. 1, pp. 184–195, feb 2017.

[58] M. Singh and S. Kim, "Intelligent Vehicle-Trust Point: Reward based Intelligent Vehicle Communication using Blockchain" in *arXiv preprint arXiv:1707.07442*, 2017.

[59] Z. Yang, K. Zheng, K. Yang, and V. C. M. Leung, "A Blockchain-basedReputation System forData Credibility Assessment in Vehicular Networks" in *Proc. of the 28th Annu. IEEE Int. Symp. on Pers., Indoor and Mobile Radio Commun. (PIMRC'17)*, pp. 1–5, Montreal, QC, Canada, oct 2017.

[60] Y. Yuan and F.-Y. Wang, "Towards Blockchain-based Intelligent Transportation Systems" in *Proc. of the 19th IEEE Int. Conf. on Intell. Transp. Syst. (ITSC'16)*, pp. 2663–2668, Rio de Janeiro, Brazil, nov 2016.

[61] B. Leiding, P. Memarmoshrefi, and D. Hogrefe, "Self-managed and Blockchain-based Vehicular Ad-hoc Networks" in *Proc. of the ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing: Adjunct (UbiComp'16)*, pp. 137–140, Heidelberg, Germany, sep 2016.

[62] A. Kapitonov, S. Lonshakov, A. Krupenkin, and I. Berman, "Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of UAVs" in *Proc. of the IEEE Workshop on Research, Educ. and Dev. of Unmanned Aerial Syst. (RED-UAS'17)*, pp. 84–89, Linkoping, Sweden, oct 2017.

[63] X. Liang, J. Zhao, S. Shetty, and D. Li, "Towards data assurance and resilience in IoT using blockchain" in *Proc. of the IEEE Military Commun. Conf. (MILCOM'17)*, pp. 261–266, Baltimore, MD, USA, oct 2017.

[64] V. Sharma, I. You, and G. Kul, "Socializing Drones for Inter-Service Operability in Ultra-Dense Wireless Networks using Blockchain" in *Proc. of the ACM Int. Workshop on Managing Insider Security Threats (MIST'17)*, pp. 81–84, Dallas, TX, USA, oct 2017.

[65] E. C. Ferrer, "The Blockchain: A New Framework for Robotic Swarm Systems" in *Springer Advances in Intell. Syst. and Comput.*, pp. 1037–1058, oct 2018.

[66] A. Kuzmin and E. Znak, "Blockchain-base structures for a secure and operate network of semi-autonomous Unmanned Aerial Vehicles" in *Proc. of the IEEE Int. Conf. on Service Operations, Logistics and Informatics (SOLI'18)*, pp. 32–37, Singapore, jul 2018.

[67] M. Wu, K. Wang, X. Cai, S. Guo, M. Guo, and C. Rong, "A Comprehensive Survey of Blockchain: From Theory to IoT Applications and Beyond" in *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 8114–8154, oct 2019.

[68] W. Viriyasitavat, L. D. Xu, Z. Bi, and D. Hoonsopon, "Blockchain Technology for Applications in Internet of Things - Mapping From System Design Perspective" in *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 8155–8168, oct 2019.

[69] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks" in *IEEE Access*, vol. 7, pp. 22 328–22 370, jan 2019.

[70] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A Taxonomy of Blockchain-Based Systems for Architecture Design" in *Proc. of the IEEE Int. Conf. on Softw. Archit (ICSA'17)*, pp. 243–252, Gothenburg, Sweden, apr 2017.

[71] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and IoT Integration: A Systematic Survey" in *MDPI Sensors*, vol. 18, no. 8, p. 2575, aug 2018.

[72] H. Wang, Z. Zheng, S. Xie, H. N. Dai, and X. Chen, "Blockchain challenges and opportunities: a survey" in *Inderscience Int. J. Web and Grid Services*, vol. 14, no. 4, pp. 352–375, oct 2018.

[73] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications" in *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, oct 2017.

[74] I.-C. Lin and T.-C. Liao, "A Survey of Blockchain Security Issues and Challenges" in *National Chung Hsing University, Int. J. of Network Security*, vol. 19, no. 5, pp. 653–659, sep 2017.

[75] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT" in *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, apr 2018.

[76] M. Belotti, N. Bozic, G. Pujolle, and S. Secci, "A Vademecum on Blockchain Technologies: When, Which, and How" in *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3796–3838, jul 2019.

[77] **L. Ghiro**, F. Restuccia, S. D'Oro, S. Basagni, T. Melodia, L. Maccari, and R. Lo Cigno, "A Blockchain Definition to Clarify its Role for the Internet of Things" in *Proc. of the IEEE 19th Mediterranean Communication and Computer Networking Conf. (MedComNet'21) -To appear-*, jun 2021.

[78] L. Ghiro, F. Restuccia, S. D'Oro, S. Basagni, T. Melodia, L. Maccari, and R. Lo Cigno, "What is a Blockchain? A Definition to Clarify the Role of the Blockchain in the Internet of Things" in *arXive:2102.03750*, 2021.

[79] Y. Rekhter, S. Hares, and T. Li, "A Border Gateway Protocol 4 (BGP-4)" RFC 4271, jan 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4271.txt

[80] E. Chen, T. J. Bates, and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)" RFC 4456, apr 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4456.txt

[81] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "On the Scalability of BGP: The Role of Topology Growth" in *IEEE J. Sel. Areas Commun. (JSAC)*, vol. 28, no. 8, pp. 1250–1261, sep 2010.

[82] The Center for Applied Internet Data Analysis (CAIDA). CAIDA's ranking of Autonomous Systems (AS). https://asrank.caida.org. [Accessed: May 9, 2021].

[83] J. Chroboczek, "Solving Route Starvation in the Babel DV Routing Protocol" RFC 6126, Sec. 2.5, apr 2011. [Online]. Available: https://tools.ietf.org/html/rfc6126#section-2.5

[84] U. Brandes, "On Variants of Shortest-Path Betweenness Centrality and their Generic Computation" in *Social Networks*, vol. 30, no. 2, pp. 136–145, may 2008.

[85] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness" in *JSTOR Sociometry*, vol. 40, no. 1, pp. 35–41, mar 1977.

[86] S. Dolev, Y. Elovici, and R. Puzis, "Routing Betweenness Centrality" in *J. of the ACM (JACM)*, vol. 57, no. 4, pp. 25:1–25:27, apr 2010.

[87] P. Zilberman, R. Puzis, and Y. Elovici, "On Network Footprint of Traffic Inspection and Filtering at Global Scrubbing Centers" in *IEEE Trans. on Dependable and Secure Comput.*, vol. 14, pp. 521–534, sep 2017.

[88] M. Kas, S. Appala, C. Wang, K. M. Carley, L. R. Carley, and O. K. Tonguz, "What if Wireless Routers were Social?" in *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 36–43, dec 2012.

[89] A. Vázquez-Rodas and L. J. de la Cruz Llopis, "A centrality-based topology control protocol for wireless mesh networks" in *Elsevier Ad Hoc Networks*, vol. 24.B, pp. 34–54, jan 2015.

[90] R. Puzis, M. Tubi, Y. Elovici, C. Glezer, and S. Dolev, "A Decision Support System for Placement of Intrusion Detection and Prevention Devices in Large-Scale Networks" in *ACM Trans. Modeling Computer Simulation (TOMACS)*, vol. 22, no. 5, pp. 1–26, dec 2011.

[91] R. Puzis, P. Zilberman, Y. Elovici, S. Dolev, and U. Brandes, "Heuristics for Speeding up Betweenness CentralityComputation" in *Proc. of the ASE/IEEE Int. Conf. on Social Computing and Int. Conf. on Privacy, Security, Risk and Trust*, pp. 302–311, Amsterdam, Netherlands, sep 2012.

[92] L. Maccari, Q. Nguyen, and R. Lo Cigno, "On the Computation of Centrality Metrics for Network Security in Mesh Networks" in *Proc. of the IEEE Global Communications Conf. (GLOBECOM'16)*, pp. 1–6, Washington, DC, USA, dec 2016.

[93] E. Bergamini and H. Meyerhenke, "Fully-dynamic Approximation of Betweenness Centrality" in *Proc. of the 23rd Elsevier European Symposium on Algorithms (ESA'15)*, pp. 155–166, Patras, Greece, sep 2015.

[94] E. Bergamini, H. Meyerhenke, and C. L. Staudt, "Approximating Betweenness Centrality in Large Evolving Networks" in *Proc. of the Meeting on Algorithm Engineering and Experiments (ALENEX'15)*, pp. 133–146, San Diego, CA, USA, jan 2015.

[95] N. Kourtellis, G. De Francisci Morales, and F. Bonchi, "Scalable Online Betweenness Centrality in Evolving Graphs" in *IEEE Trans. on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2494–2506, apr 2015.

[96] Y. Yoshida, "Almost Linear-Time Algorithms for Adaptive Betweenness Centrality using Hypergraph Sketches" in *Proc. of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 1416–1425, New York, NY, USA, aug 2014.

[97] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail, "Approximating Betweenness Centrality" in *Proc. of the 5th Springer Int. Conf. on Algorithms and Models for the Web-Graph (WAW'07)*, pp. 124–137, San Diego, CA, USA, dec 2007.

[98] U. Brandes and C. Pich, "Centrality Estimation in Large Networks" in *Int. J. of Bifurcation and Chaos*, vol. 17, no. 7, pp. 2303–2318, jul 2007.

[99] R. Geisberger, P. Sanders, and D. Schultes, "Better Approximation of Betweenness Centrality" in *Proc. of the SIAM Workshop on Algorithm Engineering and Experiments (ALENEX'08)*, pp. 90–100, San Francisco, California, jan 2008.

[100] R. Jacob, D. Koschützki, K. A. Lehmann, L. Peeters, and D. Tenfelde-Podehl, "Algorithms for Centrality Indices" in *LNCS Vol. 3418: Network Analysis*, U. Brandes and T. Erlebach, Eds. Berlin, Heidelberg: Springer, 2005, pp. 62–82.

[101] Y. Lim, D. S. Menasché, B. Ribeiro, D. Towsley, and P. Basu, "Online Estimating the k Central Nodes of a Network" in *Proc. of the IEEE Network Science Workshop (NSW'11)*, pp. 118–122, West Point, NY, USA, jun 2011.

[102] A. Maiya and T. Y. Berger-Wolf, "Online Sampling of High Centrality Individuals in Social Networks" in *Proc. of the Springer Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'10)*, pp. 91–98, Hyderabad, India, jun 2010.

[103] M. Riondato and E. Upfal, "ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages" in *ACM Trans. on Knowledge Discovery from Data*, no. 5, jul 2018.

[104] M. Baglioni, F. Geraci, M. Pellegrini, and E. Lastres, "Fast Exact Computation of Betweenness Centrality in Social Networks" in *Proc. of the IEEE Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM'12)*, pp. 450–456, Istanbul, Turkey, aug 2012.

[105] K. You, R. Tempo, and L. Qiu, "Distributed Algorithms for Computation of Centrality Measures in Complex Networks" in *IEEE Trans. Autom. Control*, vol. 62, no. 5, pp. 2080–2094, may 2017.

[106] W. Wang and C. Y. Tang, "Distributed Computation of Node and Edge Betweenness on Tree Graphs" in *Proc. of the 52nd IEEE Conf. on Decision and Control (CDC'13)*, pp. 43–48, Florence, Italy, dec 2013.

[107] W. Wang and C. Y. Tang, "Distributed Computation of Classic and Exponential Closeness on Tree Graphs" in *Proc. of the IEEE American Control Conf. (ACC'14)*, pp. 2090–2095, Portland, OR, USA, jun 2014.

[108] M. Pontecorvi and V. Ramachandran, "Distributed Algorithms for Directed Betweenness Centrality and All Pairs Shortest Paths" in *arXiv preprint arXiv:1805.08124*, 2018.

[109] L. Hoang, M. Pontecorvi, R. Dathathri, G. Gill, B. You, K. Pingali, and V. Ramachandran, "A Round-Efficient Distributed Betweenness Centrality Algorithm" in *Proc. of the 24th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPoPP'19)*, pp. 272–286, Washington DC, USA, feb 2019.

[110] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, ser. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), jan 1987.

[111] Q. S. Hua, H. Fan, M. Ai, L. Qian, Y. Li, X. Shi, and H. Jin, "Nearly Optimal Distributed Algorithm for Computing Betweenness Centrality" in *Proc. of the 36th IEEE Int. Conf. on Distributed Computing Systems (ICDCS'16)*, pp. 271–280, Nara, Japan, jun 2016.

[112] F. Bonchi, G. D. F. Morales, and M. Riondato, "Centrality Measures on Big Graphs" in *Proc. of the 25th ACM Int. Conf. Companion on World Wide Web (WWW'16)*, apr 2016.

[113] K. Wehmuth and A. Ziviani, "DACCER: Distributed Assessment of the Closeness CEntrality Ranking in complex networks" in *Elsevier Computer Networks*, vol. 57, no. 13, pp. 2536–2548, sep 2013.

[114] W. Wang and C. Y. Tang, "Distributed Estimation of Closeness Centrality" in *Proc. of the 54th IEEE Conf. on Decision and Control (CDC'15)*, pp. 4860–4865, Osaka, Japan, dec 2015.

[115] R. R. Stewart, "Stream Control Transmission Protocol" RFC 4960, sep 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4960.txt

[116] A. Ford, C. Raiciu, M. J. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses" RFC 6824, jan 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6824.txt

[117] B. E. Carpenter, D. B. D. Aboba, and S. Cheshire, "Design Considerations for Protocol Extensions" RFC 6709, sep 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6709.txt

[118] T. H. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)" RFC 3626, oct 2003. [Online]. Available: https://rfc-editor.org/rfc/rfc3626.txt

[119] Y. Rekhter, S. Hares, and T. Li, "BGP Path Attributes" RFC 4271, Sec. 5, jan 2006. [Online]. Available: https://tools.ietf.org/html/rfc4271#section-5

[120] W. W. Daniel, *Applied Nonparametric Statistics*, 2nd ed. Wadsworth Pub Co., aug 1989.

[121] B. M. Waxman, "Routing of Multipoint Connections" in *IEEE J. on Selected Areas in Communications (JSAC)*, vol. 6, no. 9, pp. 1617–1622, dec 1988.

[122] D. J. Watts, "Networks, Dynamics, and the Small-world Phenomenon" in *The University of Chicago Press, American J. of Sociology*, vol. 105, no. 2, pp. 493–527, sep 1999.

[123] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo" in *IEEE J. on Selected Areas in Communications (JSAC)*, vol. 29, no. 9, pp. 1765–1775, oct 2011.

[124] D. Vujicic, D. Jagodic, and S. Randic, "Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview" in *Proc. of the 17th IEEE Int. Symp. Infoteh-Jahorina (INFOTEH'18)*, pp. 1–6, East Sarajevo, Bosnia-Herzegovina, mar 2018.

[125] G. Greenspan. (2015) Multichain private blockchain - white paper. [Online]. Available: https://www.multichain.com/download/MultiChain-White-Paper.pdf [Accessed: May 9, 2021].

[126] J. Chen and S. Micali, "Algorand: A secure and efficient distributed ledger" in *Elsevier J. of Theoretical Computer Science*, vol. 777, pp. 155–183, jul 2019.

[127] S. Nakamoto. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.

[128] Bitcoin Wiki. Bitcoin Difficulty. https://en.bitcoin.it/wiki/Difficulty. [Accessed: May 9, 2021].

[129] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network" in *Proc. of the IEEE Int. Conf. on Peer-to-Peer Computing (P2P'13)*, Trento, Italy, sep 2013.

[130] N. T. Courtois, "On The Longest Chain Rule and Programmed Self-Destruction of Crypto Currencies" in *arXiv preprint arXiv:1405.0534v11*, 2014.

[131] E. Shi, "Analysis of Deterministic Longest-Chain Protocols" in *Proc. of the 32nd IEEE Comput. Security Found. Symp. (CSF'19)*, pp. 122–12 213, Hoboken, NJ, USA, jun 2019.

[132] M. Rosenfeld, "Analysis of Hashrate-Based Double Spending" in *arXiv preprint arXiv:1402.2009*, 2014.

[133] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal Selfish Mining Strategies in Bitcoin" in *Proc. of the 20th Int. Conf. on Financial Cryptography and Data Security (FC'16)*, pp. 515–532, Christ Church, Barbados, feb 2016.

[134] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the Security and Performance of Proof of Work Blockchains" in *Proc. of the ACM SIGSAC Conf. on Computer and Communications Security (CCS'16)*, Vienna, Austria, oct 2016.

[135] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable" in *Proc. of the 18th Springer Int. Conf. on Financial Cryptography and Data Security (FC'14)*, pp. 436–454, Christ Church, Barbados, mar 2014.

[136] C. S. Wright and S. Savanah, "The Fallacy of the Selfish Miner in Bitcoin: An Economic Critique" in *Elsevier SSRN Electronic J.*, 2017.

[137] Q. Bai, X. Zhou, X. Wang, Y. Xu, X. Wang, and Q. Kong, "A Deep Dive Into Blockchain Selfish Mining" in *Proc. of the IEEE Int. Conf. on Communications (ICC'19)*, Shanghai, China, may 2019.

[138] H. Liu, N. Ruan, R. Du, and W. Jia, "On the Strategy and Behavior of Bitcoin Mining with N-attackers" in *Proc. of the ACM Asian Conf. on Computer and Communications Security (ASIACCS'18)*, Incheon, Republic of Korea, may 2018.

[139] H. Azimy and A. Ghorbani, "Competitive Selfish Mining" in *Proc. of the 17th IEEE Int. Conf. on Privacy, Security and Trust (PST'19)*, Fredericton, Canada, aug 2019.

[140] Bitcoin Wiki. Bitcoin Confirmation. https://en.bitcoin.it/wiki/Confirmation. [Accessed: May 9, 2021].

[141] W. Feller, *An introduction to probability theory and its applications, vol 2*. John Wiley & Sons, 2008.

[142] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and D. Mohaisen, "Exploring the Attack Surface of Blockchain: A Comprehensive Survey" in *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 1977–2008, mar 2020.

[143] G. Grimmett and D. Welsh, *Probability: An Introduction*. Oxford University Press, 2014.

[144] I. Eyal and E. G. Sirer, "Majority is Not Enough: Bitcoin Mining is Vulnerable (2nd edition)" in *Commun. ACM*, vol. 61, no. 7, p. 95–102, jun 2018.

[145] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies" in *Proc. of the IEEE Symp. on Security and Privacy (SP'17)*, San Jose, CA, USA, may 2017.

[146] M. Apostolaki, G. Marti, J. Müller, and L. Vanbever, "SABRE: Protecting Bitcoin against Routing Attacks" in *Proc. of the Internet Society 26th Ann. Network and Distributed System Security Symp. (NDSS'19)*, San Diego, CA, USA, feb 2019.

[147] M. Apostolaki, C. Maire, and L. Vanbever, "Perimeter: A network-layer attack on the anonymity of cryptocurrencies" in *Proc. of the 25th IFCA Int. Virtual Conf. Financial Cryptography and Data Security (FC'21)*, mar 2021.

[148] Y. Sun, M. Apostolaki, H. Birge-Lee, L. Vanbever, J. Rexford, M. Chiang, and P. Mittal, "Securing internet applications from routing attacks" in *arXiv preprint arXiv:2004.09063*, 2020.

[149] J. Frankenfield. (2021, apr) Selfish Mining. https://www.investopedia.com/terms/s/selfish-mining.asp. [Accessed: May 9, 2021].

[150] E. Heilman, "One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner" in *Proc. of the Springer Int. Conf. on Financial Cryptography and Data Security (FC'14)*, pp. 161–162, Christ Church, Barbados, mar 2014.

[151] K. A. Negy, P. R. Rizun, and E. G. Sirer, "Selfish Mining Re-Examined" in *Proc. of the Springer Int. Conf. on Financial Cryptography and Data Security (FC'20)*, pp. 61–78, Kota Kinabalu, Malaysia, feb 2020.

[152] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse Attacks on Bitcoin's Peer-to-Peer Network" in *Proc. of the 24th USENIX Security Symp. (USENIXSecurity'15)*, pp. 129–144, Washington, D.C., USA, aug 2015.

[153] P. Fairley, "Feeding the Blockchain Beast" in *IEEE Spectrum*, vol. 54, pp. 36–59, oct 2017.

[154] A. de Vries, "Bitcoin's Growing Energy Problem" in *Elsevier Joule*, vol. 2, no. 5, pp. 801–805, may 2018.

[155] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process" in *J. of the ACM*, vol. 32, no. 2, pp. 374–382, apr 1985.

[156] S. Gilbert and N. Lynch, "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services" in *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, jun 2002.

[157] D. Abadi, "Consistency Tradeoffs in Modern Distributed Database System Design" in *IEEE Computer*, vol. 45, no. 2, pp. 37–42, feb 2012.

[158] R. Friedman and K. Birman, "Trading Consistency for Availability in Distributed Systems" in *Cornell Computer Science Technical Reports*, apr 1996.

[159] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A Sharded Smart Contracts Platform" in *arXiv preprint arXiv:1708.03778*, 2017.

[160] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding" in *Proc. of the IEEE Symp. on Security and Privacy (SP'18)*, pp. 583–598, San Francisco, CA, USA, may 2018.

[161] H. T. Kung and J. T. Robinson, "On Optimistic Methods for Concurrency Control" in *ACM Trans. on Database Systems*, vol. 6, no. 2, pp. 213–226, jun 1981.

[162] G. Bianchi, L. Fratta, and M. Oliveri, "Performance Evaluation and Enhancement of the CSMA/CA MAC Protocol for 802.11 Wireless LANs" in *Proc. of the 7th Int. IEEE Symp. on Personal, Indoor, and Mobile Communications (PIMRC'96)*, vol. 2, pp. 392–396, Taipei, Taiwan, oct 1996.

[163] N. A. Lynch, M. Merritt, and R. R. Yager, *Atomic Transactions: In Concurrent and Distributed Systems*. Morgan Kaufmann Publishers Inc., aug 1993.

[164] D. Dolev and R. Strong, "Coordination Problems in Distributed Systems" in *Proc. of the 2nd IBM Symp. on Highly Availability and Horizontal Grows*, pp. 260–272, Yorktown Heights, New York, USA, oct 1988.

[165] J. Mullen, A. Elmagarmid, W. Kim, and J. Sharif-Askary, "On the Impossibility of Atomic Commitment in Multidatabase Systems System" in *Proc. of the 2nd IEEE Int. Conf. on Systems Integration*, Morristown, NJ, USA, jun 1992.

[166] B. Dwyer, *Database technology*. Morgan Kaufmann, 2016.

[167] M. Herlihy, "Atomic Cross-Chain Swaps" in *Proc. of the 2018 ACM Symp. on Principles of Distributed Computing (PODC'18)*, Egham, UK, jul 2018.

[168] M. Herlihy, B. Liskov, and L. Shrira, "Cross-chain Deals and Adversarial Commerce" in *arXiv preprint arXiv:1905.09743*, 2019.

[169] F. Gräbe, N. Kannengießer, S. Lins, and A. Sunyaev, "Do Not Be Fooled: Toward a Holistic Comparison of Distributed Ledger Technology Designs" in *Proc. of the 53rd Hawaii Int. Conf. on System Sciences (HICSS'20)*, Maui, HI, USA, jan 2020.

[170] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem" in *ACM Trans. on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, jul 1982.

[171] M. Castro, B. Liskov *et al.*, "Practical Byzantine Fault Tolerance" in *Proc. of the 3rd ACM Symp. on Operating Syst. Design and Implementation*, New Orleans, LA, USA, feb 1999.

[172] J. Wensley, L. Lamport, J. Goldberg, M. Green, K. Levitt, P. Melliar-Smith, R. Shostak, and C. Weinstock, "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control" in *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1240–1255, jan 1979.

[173] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults" in *J. of the ACM*, vol. 27, no. 2, pp. 228–234, apr 1980.

[174] L. Lamport, "Paxos Made Simple" in *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, nov 2001.

[175] D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm" in *Proc. of the USENIX Annu. Technical Conf. (USENIX-ATC'14)*, pp. 305–319, Philadelphia, PA, USA, jun 2014.

[176] P.-L. Aublin, S. B. Mokhtar, and V. Quema, "RBFT: Redundant Byzantine Fault Tolerance" in *Proc. of the 33rd IEEE Int. Conf. on Distrib. Comput. Syst. (CDCS'13)*, pp. 297–306, Philadelphia, PA, USA, jul 2013.

[177] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT Consensus in the Lens of Blockchain" in *arXiv preprint arXiv:1803.05069*, 2018.

[178] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains" in *Proc. of the 13th ACM EuroSys Conf. (EuroSys'18)*, Porto, Portugal, apr 2018.

[179] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on BFT consensus" in *arXiv preprint arXiv:1807.04938*, 2018.

[180] B. Chase and E. MacBrough, "Analysis of the XRP ledger consensus protocol" in *arXiv preprint arXiv:1802.07242*, 2018.

[181] J. Gray and L. Lamport, "Consensus on Transaction Commit" in *ACM Trans. on Database Syst.*, vol. 31, no. 1, pp. 133–160, mar 2006.

[182] L. S. Sabel and K. Marzullo, "Election Vs. Consensus in Asynchronous Systems" Cornell University, USA, Tech. Rep., feb 1995. [Online]. Available: https://ecommons.cornell.edu/handle/1813/7146

[183] N. Santoro, "Election" in *John Wiley & Sons, Inc., Design and Analysis of Distributed Algorithms*, apr 2006, ch. 3, pp. 99–224.

[184] M. Brooker. (2019) Leader election in distributed systems. https://aws.amazon.com/builders-library/leader-election-in-distributed-systems. [Accessed: May 9, 2021].

[185] M. Lokhava, G. Losa, D. Mazières, G. Hoare, N. Barry, E. Gafni, J. Jove, R. Malinowsky, and J. McCaleb, "Fast and secure global payments with Stellar" in *Proc. of the 27th ACM Symp. on Operating Syst. Principles (SOSP'19)*, Huntsville, Ontario, Canada, oct 2019.

[186] A. Montresor, "Gossip and Epidemic Protocols" in *Wiley Encyclopedia of Elect. and Electron. Eng.*, pp. 1–15, aug 2017.

[187] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "SoK: Sharding on Blockchain" in *Proc. of the 1st ACM Conf. on Advances in Financial Technologies (AFT'19)*, Zurich, Switzerland, oct 2019.

[188] L. Baird, "The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance" https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf, Swirlds, Tech. Rep. SWIRLDS-TR-2016-01, may 2016.

[189] M. Ahamad, G. Neiger, J. E. Burns, P. Kohli, and P. W. Hutto, "Causal Memory: Definitions, Implementation and Programming" in *Springer Distributed Computing*, vol. 9, no. 1, pp. 37–49, mar 1995.

[190] L. Lamport, "How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs" in *IEEE Trans. Comput.*, vol. C–28, no. 9, p. 690–691, sep 1979.

[191] Bitcoin Wiki. Bitcoin Controlled Supply. https://en.bitcoin.it/wiki/Controlled_supply. [Accessed: May 9, 2021].

[192] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative Analysis of Blockchain Consensus Algorithms" in *Proc. of the 41st IEEE Int. Conv. on Inform. and Commun. Technol., Electron. and Microelectron. (MIPRO'18)*, pp. 1545–1550, Opatija, Croatia, may 2018.

[193] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On Security Analysis of Proof-of-Elapsed-Time (PoET)" in *Proc. of the 19th Springer Int. Symp. on Stabilization, Safety, and Security of Distrib. Syst. (SSS'17)*, pp. 282–297, Boston, MA, USA, oct 2017.

[194] PoET 1.0 Specification. https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html. [Accessed: May 9, 2021].

[195] T. Larsson and R. Thorsén. (2018, jun) Cryptocurrency performance analysis of Burstcoin mining. https://tinyurl.com/BurstcoinAnalysis. [Accessed: May 9, 2021].

[196] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing Proof-of-Stake Blockchain Protocols" in *Springer Lecture Notes in Computer Science*, 2017, pp. 297–315.

[197] V. Buterin and V. Griffith, "Casper the Friendly Finality Gadget" in *arXiv preprint arXiv:1710.09437*, 2017.

[198] J. Poon and T. Dryja. (2016) The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf. [Accessed: May 9, 2021].

[199] J. Tremback, J. Kilpatrick, D. Simpier, and B. Wang. (2017) Althea whitepaper. https://althea.net/whitepaper. [Accessed: May 9, 2021].

[200] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies" in *Proc. of the 26th ACM Symp. on Operating Systems Principles (SOSP'17)*, Shanghai, China, oct 2017.

[201] M. Conti, A. Gangwal, and M. Todero, "Blockchain Trilemma Solver Algorand has Dilemma over Undecidable Messages" in *Proc. of the 14th ACM Int. Conf. on Availability, Reliability and Security (ARES'19)*, Canterbury, United Kingdom, aug 2019.

[202] S. Micali, M. Rabin, and S. Vadhan, "Verifiable Random Functions" in *Proc. of the 40th IEEE Ann. Symp. on Foundations of Computer Science*, New York, NY, USA, oct 1999.

[203] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand technical report" MIT CSAIL, Tech. Rep., 2017. [Online]. Available: https://eprint.iacr.org/2017/454.pdf

[204] S. Micali. (2021) Algorand 2021 Performance. [Online]. Available: https://www.algorand.com/120720-Algorand%202021%20Performance.pdf [Accessed: May 9, 2021].

[205] (2021) Lack of transparency post on the Algorand forum. [Online]. Available: https://forum.algorand.org/t/serious-lack-of-transparency-and-engagement-of-algorand-foundation/2385/22 [Accessed: May 9, 2021].

[206] M. Fooladgar, M. H. Manshaei, M. Jadliwala, and M. A. Rahman, "On Incentive Compatible Role-Based Reward Distribution in Algorand" in *Proc. of the 50th Ann. IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN'20)*, Valencia, Spain, jun 2020.

[207] A. Castor. (2017, apr) No Incentive? Algorand Blockchain Sparks Debate at Cryptography Event. [Online]. Available: https://www.coindesk.com/no-incentive-algorand-blockchain-sparks-debate-cryptography-event [Accessed: May 9, 2021].

[208] A. Gauba. (2018, mar) The Need for an Incentive Scheme in Algorand. [Online]. Available: https://medium.com/blockchain-at-berkeley/the-need-for-an-incentive-scheme-in-algorand-6fe9db45f2a7 [Accessed: May 9, 2021].

[209] Y. Wang, "Another Look at ALGORAND" in *arXiv preprint arXiv:1905.04463*, 2019.

[210] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues" in *Elsevier Telematics and Informatics*, vol. 36, pp. 55–81, mar 2019.

[211] M. E. Peck, "Blockchain world - Do you need a blockchain? This chart will tell you if the technology can solve your problem" in *IEEE Spectrum*, vol. 54, no. 10, pp. 38–60, oct 2017.

[212] K. Wüst and A. Gervais, "Do you need a Blockchain?" in *Proc. of the IEEE Crypto Valley Conf. on Blockchain Technol. (CVCBT'18)*, pp. 45–54, Zug, Switzerland, jun 2018.

[213] M. Herlihy, "Blockchains from a Distributed Computing Perspective" in *Communications of the ACM*, vol. 62, no. 2, pp. 78–85, jan 2019.

[214] M. Herlihy, "Blockchains and the Future of Distributed Computing" in *Proc. of the ACM Symp. on Principles of Distributed Computing (PODC'17)*, Washington DC, USA, jul 2017.

[215] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain" in *Proc. of the Italian Conf. on Cyber Security (ITASEC'18)*, Milan, Italy, feb 2018.

[216] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, "Adding Concurrency to Smart Contracts" in *Springer Distributed Computing*, vol. 33, no. 3-4, pp. 209–225, jul 2019.

[217] N. Atzei, M. Bartoletti, and T. Cimoli, "A Survey of Attacks on Ethereum Smart Contracts (SoK)" in *Proc. of the 6th Springer Int. Conf. on Principles of Security and Trust (POST'17)*, pp. 164–186, Uppsala, Sweden, apr 2017.

[218] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices" in *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 8182–8201, oct 2019.

[219] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A Survey on Ethereum Systems Security" in *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–43, jul 2020.

[220] J. Göbel and A. E. Krzesinski, "Increased block size and Bitcoin blockchain dynamics" in *Proc. of 27th IEEE Int. Telecommunication Networks and Applications Conf. (ITNAC'17)*, Melbourne, Australia, nov 2017.

[221] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, D. Song, and R. Wattenhofer, "On Scaling Decentralized Blockchains" in *Proc. of the Springer Int. Conf. on Financial Cryptography and Data Security (FC'16)*, pp. 106–125, Christ Church, Barbados, feb 2016.

[222] M. Vukolić, "The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication" in *Proc. of the Springer Int. Workshop on Open Problems in Network Security (iNetSec'15)*, pp. 112–125, Zurich, Switzerland, oct 2016.

[223] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, and E. G. Sirer, "Decentralization in Bitcoin and Ethereum Networks" in *Proc. of the Springer Int. Conf. on Financial Cryptography and Data Security (FC'18)*, pp. 439–457, Nieuwpoort, Curaçao, Netherlands, mar 2018.

[224] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, "Is Bitcoin a Decentralized Currency?" in *IEEE Security & Privacy*, vol. 12, no. 3, pp. 54–60, may 2014.

[225] F. Armknecht, J.-M. Bohli, G. O. Karame, and W. Li, "Sharding PoW-based Blockchains via Proofs ofKnowledge" in *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 1067, 2017.

[226] Cryptocurrency Prices by Market Cap. [Online]. Available: https://coinmarketcap.com [Accessed: May 9, 2021].

[227] M. Selimi, A. R. Kabbinale, A. Ali, L. Navarro, and A. Sathiaseelan, "Towards Blockchain-enabled Wireless Mesh Networks" in *Proc. of the 1st ACM Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock'18)*, Munich, Germany, jun 2018.

[228] L. Navarro, R. Baig, F. Freitag, E. Dimogerontakis, F. Treguer, M. D. de Rosnay, L. Maccari, P. Micholia, and P. Antoniadis, "Report on the existing cns and their organization (v2)–netcommons deliverable (d1.2)" https://netcommons.eu/sites/default/files/netcommons_d1.2v1.0.pdf, sep 2016.

[229] J. Martignoni, P. Antoniadis, and M. Karaliopoulos, "Economic Sustainability of CNs (v2), Community Currencies for Community Networks (D2.6)" https://netcommons.eu/sites/default/files/d.2.6-v2.3.pdf, may 2018.

[230] P. Antoniadis, J. Martignoni, L. Navarro, and P. Dini, "Complementary Networks Meet Complementary Currencies: Guifi.net Meets Sardex.net" in *The Community Network Manual: How to Build the Internet Yourself*. FGV Direito Rio, 2018.

[231] L. Maccari, M. Karaliopoulos, I. Koutsopoulos, L. Navarro, F. Freitag, and R. Lo Cigno, "5G and the Internet of EveryOne: Motivation, Enablers, and Research Agenda" in *Proc. of the IEEE European Conf. on Networks and Communications (EuCNC'18)*, Ljubljana, Slovenia, jun 2018.

[232] P. Antoniadis, J. Martignoni, R. Baig, D. Franquesa, and L. Navarro, "Community Currencies for Community Networks" in *Proc. of the 4th Int. Conf. on Social and Complementary Currencies: Money, Awareness and Values for Social Change*, Barcelona, Spain, may 2017.

[233] I. Castro, A. Panda, B. Raghavan, S. Shenker, and S. Gorinsky, "Route Bazaar: Automatic Interdomain Contract Negotiation" in *Proc. of the 15th USENIX Workshop on Hot Topics in Operating Systems (HotOS'15)*, Kartause Ittingen, Switzerland, may 2015.

[234] T. Jin, X. Zhang, Y. Liu, and K. Lei, "BlockNDN: A bitcoin blockchain decentralized system over named data networking" in *Proc. of the 9th IEEE Int. Conf. on Ubiquitous and Future Networks (ICUFN'17)*, Milan, Italy, jul 2017.

[235] B. B. Rodrigues, T. Bocek, and B. Stiller, "Multi-domain DDoS Mitigation Based on Blockchains" in *Proc. of the Springer Int. Conf. on Autonomous Infrastructure, Management, and Security (AIMS'17)*, Zurich, Switzerland, jun 2017.

[236] A. Hari and T. V. Lakshman, "The Internet Blockchain: A Distributed, Tamper-Resistant Transaction Framework for the Internet" in *Proc. of the 15th ACM Workshop on Hot Topics in Networks (HotNets'16)*, Atlanta, GA, USA, nov 2016.

[237] M. Lepinski and K. Sriram, "BGPsec Protocol Specification" RFC 8205, sep 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8205.txt

[238] S. Greengard, *The Internet of Things*, ser. Essential Knowledge. The MIT Press, mar 2015.

[239] S. Kim and G. C. Deka, *Advanced Applications of Blockchain Technology*, ser. Studies in Big Data. Singapore, Singapore: Springer Nature, 2019, vol. 60.

[240] B. G. Lindsay, P. G. Selinger, C. Galtieri, J. N. Gray, R. A. Lorie, T. G. Price, F. Putzolu, I. L. Traiger, and B. W. Wade, "Notes on Distributed Databases" IBM Research Division, Tech. Rep. RJ2571 - Computer Science, jul 1979.

[241] A. R. Bobak, *Distributed and Multi-database Systems*, ser. Artech House computer science library. Artech House, 1995.

[242] M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*, 3rd ed. Springer New York, 2011.

[243] D. F. Bacon, E. Kogan, A. Lloyd, S. Melnik, R. Rao, D. Shue, C. Taylor, M. van der Holst, D. Woodford, N. Bales, N. Bruno, B. F. Cooper, A. Dickinson, A. Fikes, C. Fraser, A. Gubarev, and M. Joshi, "Spanner: Becoming a SQL System" in *Proc. of the ACM Int. Conf. on Management of Data (SIGMOD'17)*, Chicago, IL, USA, may 2017.

[244] S. Bradshaw, E. Brazil, and K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. O'Reilly Media, 2019.

[245] A. Lakshman and P. Malik, "Cassandra - A Decentralized Structured Storage System" in *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, apr 2010.

[246] B. Kemme, "Data Replication" in *Sprigner Encyclopedia of Database Systems*, 2009, pp. 626–630.

[247] P. A. Bernstein and N. Goodman, "The Failure and Recovery Problem for Replicated Databases" in *Proc. of the 2nd ACM Symp. on Principles of Distributed Computing (PODC'83)*, aug 1983.

[248] D. Agrawal, A. E. Abbadi, and R. C. Steinke, "Epidemic Algorithms in Replicated Databases" in *Proc. of the 16th ACM Symp. on Principles of Database Systems (PODS'97)*, Tucson, Arizona, USA, may 1997.

[249] A. E. Abbadi and S. Toueg, "Maintaining Availability in Partitioned Replicated Databases" in *ACM Trans. on Database Systems*, vol. 14, no. 2, pp. 264–290, jun 1989.

[250] S. Adve and K. Gharachorloo, "Shared Memory Consistency Models: A Tutorial" in *IEEE Computer*, vol. 29, no. 12, pp. 66–76, dec 1996.

[251] Homepage of BitcoinCash. [Online]. Available: https://www.bitcoincash.org [Accessed: May 9, 2021].

[252] Homepage of Litecoin. [Online]. Available: https://litecoin.com/en [Accessed: May 9, 2021].

[253] Homepage of Namecoin. [Online]. Available: https://www.namecoin.org [Accessed: May 9, 2021].

[254] Homepage of Dogecoin. [Online]. Available: https://dogecoin.com [Accessed: May 9, 2021].

[255] Homepage of Primecoin. [Online]. Available: https://primecoin.io [Accessed: May 9, 2021].

[256] Homepage of Auroracoin. [Online]. Available: https://en.auroracoin.is [Accessed: May 9, 2021].

[257] Homepage of Ethereum Classic. [Online]. Available: https://ethereumclassic.org [Accessed: May 9, 2021].

[258] Homepage of Zcash. [Online]. Available: https://z.cash [Accessed: May 9, 2021].

[259] Wikipedia. List of cryptocurrencies. [Online]. Available: https://en.wikipedia.org/wiki/List_of_cryptocurrencies [Accessed: May 9, 2021].

[260] R. L. Karim Sultan, Umar Ruhi, "Conceptualizing blockchains: Characteristics & applications" in *Proc. of the 11th IADIS Int. Conf. on Information Systems*, Lisbon, Portugal, apr 2018.

[261] M. Iansiti and K. R. Lakhani, "The Truth About Blockchain" in *Harvard Business Review*, pp. 118–127, jan 2017.

[262] E. Politou, F. Casino, E. Alepis, and C. Patsakis, "Blockchain Mutability: Challenges and Proposed Solutions" in *IEEE Trans. Emerg. Topics Comput.*, pp. 1–1, oct 2019.

[263] D. G. Bobrow, "A Note on Hash Linking" in *Communications of the ACM*, vol. 18, no. 7, p. 413–415, jul 1975.

[264] C. Halatsis and G. Philokyprou, "Pseudochaining in Hash Tables" in *Communications of the ACM*, vol. 21, no. 7, pp. 554–557, jul 1978.

[265] D. Floyd. (2019, jun) Banks Claim They're Building Blockchains. They're Not. https://investopedia.com/news/banks-building-blockchains-distributed-ledger-permission/. [Accessed: May 9, 2021].

[266] A. Shevchenko. (2020, apr) Proof-of-Stake Vs. Proof-of-Work: Which One Is 'Fairer'? [Online]. Available: https://cointelegraph.com/news/proof-of-stake-vs-proof-of-work-which-one-is-fairer [Accessed: May 9, 2021].

[267] E. Foundation. What are the benefits of proof of stake as opposed to proof of work? [Online]. Available: https://eth.wiki/en/concepts/proof-of-stake-faqs#what-are-the-benefits-of-proof-of-stake-as-opposed-to-proof-of-work [Accessed: May 9, 2021].

[268] V. Buterin. (2016, dec) A Proof of Stake Design Philosophy. [Online]. Available: https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51 [Accessed: May 9, 2021].

[269] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the Instability of Bitcoin Without the Block Reward" in *Proc. of the ACM SIGSAC Conf. on Computer and Communications Security (CCS'16)*, Vienna, Austria, oct 2016.

[270] F. Saleh, "Blockchain Without Waste: Proof-of-Stake" in *SSRN Electronic J.*, jun 2018.

[271] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies Without Proof of Work" in *Proc. of the Springer Int. Conf. on Financial Cryptography and Data Security (FC'16)*, pp. 142–157, Christ Church, Barbados, 2016.

[272] G. Fanti, L. Kogan, S. Oh, K. Ruan, P. Viswanath, and G. Wang, "Compounding of Wealth in Proof-of-Stake Cryptocurrencies" in *Proc. of the 23rd Springer Int. Conf. on Financial Cryptography and Data Security (FC'19)*, pp. 42–61, Frigate Bay, St. Kitts and Nevis, 2019.

[273] C. Catalini, R. Jagadeesan, and S. D. Kominers, "Market Design for a Blockchain-Based Financial System" in *SSRN Electronic J.*, 2019.

[274] W. K. Härdle, C. R. Harvey, and R. C. G. Reule, "Understanding Cryptocurrencies" in *Oxford University Press, J. of Financial Econometrics*, vol. 18, no. 2, pp. 181–208, 02 2020.

[275] G. Fanti, L. Kogan, and P. Viswanath, "Economics of Proof-of-Stake Payment Systems" 2020, Preliminary and Icomplete, Latest draft: March 2020. [Online]. Available: http://fetch.econ.cam.ac.uk/papers/main_3_25_2020-Kogan.pdf

[276] E. Deirmentzoglou, G. Papakyriakopoulos, and C. Patsakis, "A Survey on Long-Range Attacks for Proof of Stake Protocols" in *IEEE Access*, vol. 7, pp. 28 712–28 725, 2019.

[277] P. Gazi, A. Kiayias, and A. Russell, "Stake-Bleeding Attacks on Proof-of-Stake Blockchains" in *Proc. of the IEEE Crypto Valley Conf. on Blockchain Technology (CVCBT'18)*, Zug, Switzerland, jun 2018.

[278] J. Li, N. Li, J. Peng, H. Cui, and Z. Wu, "Energy consumption of cryptocurrency mining: A study of electricity consumption in mining cryptocurrencies" in *Elsevier Energy*, vol. 168, pp. 160–168, feb 2019.

[279] S. Meunier. When do you need blockchain? decision models. [Online]. Available: https://medium.com/@sbmeunier/when-do-you-need-blockchain-decision-models-a5c40e7c9ba1 [Accessed: May 9, 2021].

[280] S. Heiberg, I. Kubjas, J. Siim, and J. Willemson, "On Trade-offs of Applying Block Chains forElectronic Voting Bulletin Boards" in *Proc. of the 3rd Int. Joint Conf. on Electronic Voting (E-Vote-ID'18)*, pp. 259–276, Lochau/Bregenz, Austria, oct 2018.

[281] M. M. H. Onik and M. H. Miraz, "Performance Analytical Comparison of Blockchain-as-a-Service (BaaS) Platforms" in *Proc. of the 2nd Springer Int. Conf. on Emerging Technol. in Comput. (iCETiC'19)*, pp. 3–18, London, UK, aug 2019.

[282] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second" in *Proc. of the IEEE Int. Conf. on Blockchain and Cryptocurrency (ICBC'19)*, pp. 455–463, Seoul, South Korea, may 2019.

[283] Hyperledger and its projects. https://hyperledger.org/learn. [Accessed: May 9, 2021].

[284] Hyperledger Fabric Channels. https://github.com/hyperledger/fabric/blob/master/docs/source/channels.rst. [Accessed: May 9, 2021].

[285] Hyperledger Fabric Private data. https://hyperledger-fabric.readthedocs.io/en/latest/private-data/private-data.html. [Accessed: May 9, 2021].

[286] "Permissioned vs Permissionless Blockchains" in the Fabric Documentation. https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html\#permissioned-vs-permissionless-blockchains. [Accessed: May 9, 2021].

[287] S. Popejoy. (2019, may) IBM's Hyperledger isn't a real blockchain – here's why. https://thenextweb.com/podium/2019/05/05/ibms-hyperledger-isnt-a-real-blockchain-heres-why. [Accessed: May 9, 2021].

[288] S. Popejoy. (2019, jul) Why IBM's Blockchain Isn't a Real Blockchain. https://cointelegraph.com/news/why-ibms-blockchain-isnt-a-real-blockchain. [Accessed: May 9, 2021].

[289] Hyperledger Sawtooth Homepage. https://sawtooth.hyperledger.org. [Accessed: May 9, 2021].

[290] Known Attacks to Software Guard Extensions. https://en.wikipedia.org/wiki/Software_Guard_Extensions\#Attacks. [Accessed: May 9, 2021].

[291] S. Popov. (2018) The tangle. [Online]. Available: https://tinyurl.com/IOTATanglePaper [Accessed: May 9, 2021].

[292] S. Popov, H. Moog, D. Camargo, A. Capossele, V. Dimitrov, A. Gal, A. Greve, B. Kusmierz, S. Mueller, A. Penzkofer, O. Saa, W. Sanders, L. Vigneri, W. Welz, and V. Attias. (2020) The Coordicide. [Online]. Available: https://files.iota.org/papers/20200120_Coordicide_WP.pdf [Accessed: May 9, 2021].

[293] S. Popov and Q. Lu, "IOTA: Feeless and Free" in *IEEE Blockchain Technical Briefs*, 2019. [Online]. Available: https://blockchain.ieee.org/technicalbriefs/january-2019/iota-feeless-and-free [Accessed: May 9, 2021].

[294] E. Heilman, N. Narula, G. Tanzer, J. Lovejoy, M. Colavita, M. Virza, and T. Dryja, "Cryptanalysis of Curl-P and Other Attacks on the IOTA Cryptocurrency" in *IACR Trans. on Symmetric Cryptology*, pp. 367–391, sep 2020.

[295] P. Elias, A. Feinstein, and C. Shannon, "A Note on the Maximum Flow Through a Network" in *IEEE Trans. Inform. Theory*, vol. 2, no. 4, pp. 117–119, dec 1956.

[296] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges" in *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2483–2495, aug 2018.

[297] B. Lee and J.-H. Lee, "Blockchain-based secure firmware update for embedded devices in an Internet of Things environment" in *Springer J. of Supercomput.*, vol. 73, no. 3, pp. 1152–1167, sep 2016.

[298] A. Mostefaoui, E. Mourgaya, and M. Raynal, "An introduction to oracles for asynchronous distributed systems" in *Elsevier Future Generation Comput. Syst.*, vol. 18, no. 6, pp. 757–767, may 2002.

[299] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A Decentralized Blockchain Oracle" in *Proc. of the IEEE Int. Conf. on Internet of Things (iThings), Green Comput. and Commun. (GreenCom), Cyber, Physical and Social Comput. (CPSCom) and Smart Data (SmartData)*, pp. 1145–1152, Halifax, NS, Canada, jul 2018.

[300] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home" in *Proc. of the IEEE Int. Conf. on Pervasive Comput. and Commun. Workshops (PerCom'17)*, pp. 618–623, Kona, HI, USA, mar 2017.

[301] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT" in *Proc. of the Springer Europe and MENA Cooperation Advances in Inf. and Commun. Technol. (EMENA-TSSL'16)*, pp. 523–533, Saidia, Oujda, Morocco, sep 2017.

[302] Y. Rahulamathavan, R. C.-W. Phan, M. Rajarajan, S. Misra, and A. Kondoz, "Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption" in *Proc. of the IEEE Int. Conf. on Advanced Netw. and Telecommun. Syst. (ANTS'17)*, pp. 1–6, Bhubaneswar, India, dec 2017.

[303] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "LSB: A Lightweight Scalable BlockChain for IoT Security and Privacy" in *arXiv preprint arXiv:1712.02969*, 2017.

[304] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT" in *Proc. of the 2nd IEEE/ACM Int. Conf. on Internet-of-Things Des. and Implement. (IoTDI'17)*, pp. 173–178, Pittsburgh, PA, USA, apr 2017.

[305] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "BlockChain: A Distributed Solution to Automotive Security and Privacy" in *IEEE Commun. Magazine*, vol. 55, no. 12, pp. 119–125, dec 2017.

[306] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable Blockchain – or – Rewriting History in Bitcoin and Friends" in *Proc. of the IEEE Eur. Symp. on Security and Privacy (EuroSP'17)*, pp. 111–126, Paris, France, apr 2017.

[307] M. Berberich and M. Steiner, "Practitioner's Corner - Blockchain Technology and the GDPR - How to Reconcile Privacy and Distributed Ledgers?" in *Lexxion, European Data Protection Law Review*, vol. 2, no. 3, pp. 422–426, mar 2016.

[308] G. Gabison, "Policy Considerations for the Blockchain Technology Public and Private Applications" in *SMU Sci. and Technol. Law Review*, vol. 19, no. 3, pp. 327–350, 2016.

[309] P.-Y. Chang, M.-S. Hwang, and C.-C. Yang, "A Blockchain-Based Traceable Certification System" in *Proc. of the Springer Int. Conf. on Security with Intell. Comput. and Big-data Services (SICBS'18)*, pp. 363–369, Guilin, China, mar 2018.

[310] W. Gräther, S. Kolvenbach, R. Ruland, J. Schütte, C. Torres, and F. Wendland, "Blockchain for Education: Lifelong Learning Passport" in *Proc. of the 1st EUSSET ERCIM Blockchain Workshop*, Amsterdam, Netherlands, may 2018.

[311] M. D. Fowler, "Linking the Public Benefit to the Corporation: Blockchain as a Solution for Certification in an Age of Do-Good Business" in *Vanderbilt J. Entertainment and Technol. Law*, vol. 20, no. 3, p. 881, mar 2018.

[312] R. Burstall and B. Clark, "Blockchain, IP and the Fashion Industry" in *Managing Intell. Prop.*, vol. 266, p. 9, apr 2017.

[313] J.-C. Cheng, N.-Y. Lee, C. Chi, and Y.-H. Chen, "Blockchain and Smart Contract for Digital Certificate" in *Proc. of the IEEE Int. Conf. on Applied Syst. Invention (ICASI'18)*, pp. 1046–1051, Chiba, Japan, apr 2018.

[314] W. Nowiński and M. Kozma, "How Can Blockchain Technology Disrupt the Existing Business Models?" in *CEEOL Entrepreneurial Business and Economics Review*, vol. 5, no. 3, pp. 173–188, jun 2017.

[315] J. F. Galvez, J. Mejuto, and J. Simal-Gandara, "Future challenges on the use of blockchain for food traceability analysis" in *Elsevier Trends in Analytical Chemistry (TrAC)*, vol. 107, pp. 222–232, oct 2018.

[316] A. Bahga and V. K. Madisetti, "Blockchain Platform for Industrial Internet of Things" in *SCIRP J. of Softw. Eng. and Appl.*, vol. 09, no. 10, pp. 533–546, oct 2016.

[317] V. A. J. Boehm, J. Kim, and J. W.-K. Hong, "Holistic Tracking of Products on the Blockchain Using NFC and Verified Users" in *Proc. of the Springer Int. Workshop on Inf. Security Appl. (WISA'17)*, pp. 184–195, Jeju Island, Korea, aug 2017.

[318] S. Dziembowski, S. Faust, and K. Hostáková, "General State Channel Networks" in *Proc. of the 2018 ACM SIGSAC Conf. on Computer and Comm. Security (CCS'18)*, p. 949–966, Toronto, Canada, jan 2018.

[319] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards Bitcoin Payment Networks" in *Proc. of the 21st Springer Australasian Conf. on Information Security and Privacy (ACISP'16)*, pp. 57–76, Melbourne, Australia, jun 2016.

[320] A. Ensor, S. Schefer-Wenzl, and I. Miladinovic, "Blockchains for IoT Payments: A Survey" in *Proc. of the IEEE Globecom Workshops (GC-Wkshps'18)*, pp. 1–6, Abu Dhabi, United Arab Emirates, dec 2018.

[321] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, "Sprites and State Channels: Payment Networks that Go Faster Than Lightning" in *Proc. of the Springer Int. Conf. on Financial Cryptography and Data Security (FC'19)*, pp. 508–526, Saint Kitts and Nevis, sep 2019.

[322] C. Burchert, C. Decker, and R. Wattenhofer, "Scalable funding of Bitcoin micropayment channel networks" in *Royal Society Open Science*, vol. 5, pp. 1–15, aug 2018.

[323] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High Throughput Cryptocurrency Routing in Payment Channel Networks" in *Proc. of the 17th USENIX Symp. on Networked Syst. Design and Implementation (NSDI'20)*, pp. 777–796, Santa Clara, CA, feb 2020.

[324] R. Pickhardt and M. Nowostawski, "Imbalance measure and proactive channel rebalancing algorithm for the Lightning Network" in *Proc. of the IEEE Int. Conf. on Blockchain and Cryptocurrency (ICBC'20)*, Toronto, Canada, may 2020.

[325] S. Tikhomirov, R. Pickhardt, A. Biryukov, and M. Nowostawski, "Probing Channel Balances in the Lightning Network" in *arXiv preprint arXiv:2004.00333*, 2020.

[326] L. Eckey, S. Faust, K. Hostáková, and S. Roos, "Splitting Payments Locally While Routing Interdimensionally" in *Cryptology ePrint Archive*, may 2020.

[327] L. Maccari, N. Facchi, L. Baldesi, and R. Lo Cigno, "Optimized P2P Streaming for Wireless Distributed Networks" in *Elsevier Pervasive and Mobile Computing J.*, vol. 42, pp. 335–350, dec 2017.