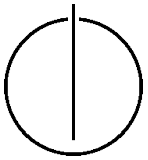
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Exploring Latency Boundaries of Blockchains in Edge Computing Networks Using Emulation

Leo Eichhorn



DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Exploring Latency Boundaries of Blockchains in Edge Computing Networks Using Emulation

Analyse der Latenzgrenzen von Blockchains in Edge Computing Netzwerken durch Emulation

Author:	Leo Eichhorn
Supervisor:	Prof. Dr.-Ing. Jörg Ott
Advisor:	Dr. Nitinder Mohan
Submission Date:	15.07.2021

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, Leo Eichhorn

Acknowledgments

First of all, I would like to thank the Chair of Connected Mobility and especially Prof. Dr.-Ing. Jörg Ott for accommodating me with the opportunity of writing this thesis. Furthermore, I would like to acknowledge the assistance of Dr. Aleksandr Zavodovski and Tanya Shreedhar for offering helpful advice and feedback throughout this work. In particular, I want to express my gratitude towards my advisor, Dr. Nitinder Mohan, for his guidance over the course of this project. His knowledge and constructive meetings were invaluable to the completion of this work. Finally, I would like to thank my friends and family for their endless support and patience throughout my studies.

Abstract

Edge computing describes the process of moving computational resources and services closer to their consuming clients, in an attempt to reduce network traffic and increase location awareness. The rapid growth of this paradigm as an enabler for IoT and latency-critical applications has posed a new challenge for building decentralized edge systems. Namely, such systems would benefit from distributed coordination, agreement, and transaction handling services without relying on any third-party authority. In this thesis, we investigate the suitability of distributed ledger technologies for addressing those needs, focusing mainly on the capabilities of existing blockchain systems to operate as a distributed marketplace for the exchange of edge computing tasks and resources. By matching requests for service deployments with (crowdsourced) edge computing resources using a decentralized blockchain auctioneer, a self-sufficient and autonomous system can be envisioned. Here, we are especially interested in potential boundaries, trade-offs or bottlenecks imposed by blockchain when applied in combination with performance optimized edge. For our purposes, we develop a generic emulation framework for distributed ledgers, enabling the measurement of parameters that are of high relevance for edge computing platforms, such as latency, consistency and fault tolerance of transactions. By supporting a total of 26 configuration variables, allowing the emulation of arbitrary peer-to-peer networks and modular consensus protocols, we evaluate the suitability of two state-of-the-art blockchain platforms for their use in edge computing. Therein, the validation of our emulator against the major blockchain networks has shown its accuracy being above 85%. Using four simple empirical models, we identify severe performance trade-offs of blockchain technology, leading to reduced efficiency, consistency and decentralization of transactions and higher level applications, when optimizing for lower latency. To improve the latency of transactions efficiently, we propose a sharding solution of fragmented edge marketplaces backed by local blockchain networks, i.e. as part of a smart city environment. The resulting individual markets are synchronized using a globally shared interledger. Our results make us cautiously optimistic about the suitability of sharded blockchain technology for edge computing applications with medium to relaxed latency requirements. However, we emphasize the need for a permissioned and edge-restricted deployment of the ledger, and propose the use of more centralized approaches to improve efficiency and performance of marketplace operations, albeit at the cost of trust and privacy.

Zusammenfassung

Im Rahmen von Edge Computing werden aus Cloud Computing bekannte Rechenressourcen näher zu deren Nutzern verlegt, um Netzwerkauslastungen zu verringern und die Lokalität der Ressourcen zu erhöhen. Aufgrund der Schlüsselrolle dieser Technologie in der Gewährleistung von IoT und anderen Applikationen mit stringenten Latenzvoraussetzungen, wächst die Nachfrage an derartige Ressourcen unaufhaltsam. Dies stellt eine neue Herausforderung für die Bereitstellung dezentraler Edge-Systeme dar. Aufgrund ihrer natürlichen, flächenmäßigen Verteilung, würden solche Systeme von Diensten profitieren, welche eine dezentrale Koordination und Verständigung über Transaktionen gewährleisteten. In dieser Arbeit untersuchen wir die Eignung von Distributed Ledgern (Blockchains) für diese Rolle. Dabei fokussieren wir uns auf deren Möglichkeit als eine verteilte Handelsplattform von Edge Ressourcen zu fungieren. Durch das dezentrale Zuordnen von Bereitstellungsanfragen eines Services und den dazu passenden Rechenressourcen, könnte eine autonome und autarke Edge Computing Umgebung vergegenwärtigt werden. In solch einer Architektur sind wir besonders an möglichen Trade-offs und Performanzgrenzen interessiert, die aus einer Verwendung von Blockchain in Edge Computing Netzwerken resultieren. Dazu entwickeln wir eine generische Plattform, welche die Emulation von Blockchains erlaubt, und damit das Messen von mehreren relevanten Parametern wie Latenz, Einheitlichkeit und Fehlertoleranz von Transaktionen ermöglicht. Durch das Bereitstellen von 26 konfigurierbaren Variablen können modulare Blockchain-Protokolle in beliebigen Netzwerken evaluiert werden. Dabei weist unser Emulator eine Genauigkeit von über 85% gegenüber realen Blockchain-Netzwerken auf. Mithilfe von vier einfachen, empirischen Modellen identifizieren wir signifikante Kapazitätsengpässe von Blockchains mit niedriger Transaktionslatenz, welche zu einer reduzierten Effizienz, Korrektheit und Dezentralität des Systems führt. Um die Latenz der Transaktionen effizient zu verbessern, schlagen wir eine Lösung mittels *Sharding* vor. Darin wird der globale Markt für Rechenressourcen in mehrere kleinere Märkte, etwa auf Smart-City Ebene, unterteilt. Diese Märkte werden von fragmentierten Blockchains bereitgestellt, welche sich über einen gemeinsamen, globalen *Interledger* synchronisieren. Unsere Ergebnisse stimmen uns zuversichtlich gegenüber einer potentiellen Umsetzung unserer Lösung in weniger Latenz-stringenten Edge-Umgebungen. Allerdings betonen wir die Notwendigkeit die Teilnahme an derartigen Edge-Marktplätzen geographisch zu begrenzen und regen zur Umsetzung zentralisierterer Ansätze an, auch wenn dies Einschränkungen in Vertrauen und Privatsphäre impliziert.

Contents

Acknowledgments **iii**

Abstract **v**

Zusammenfassung **vii**

1. Introduction **1**
1.1. Context 2
1.2. Problem Statement 2
1.3. Thesis Approach and Outline 3

2. Background **5**
2.1. Edge Computing 5
2.1.1. Edge Offloading 6
2.1.2. Applications 6
2.1.3. Challenges 9
2.2. Crowdsourcing in Edge 9
2.3. Distributed Ledger Technology 10
2.3.1. Proof-Based Blockchains 11
2.3.2. BFT Blockchains 17
2.3.3. Committee-Based Blockchains 18
2.3.4. DAG-Based DLT 19
2.3.5. Trade-offs in DLT 20

3. Related Work **23**
3.1. DLT in Edge Computing 23
3.2. Edge Computing Marketplaces and Provisioning 24
3.3. DLT Evaluation 26
3.4. Contribution 27

4. Edge Computing Marketplace **29**
4.1. Auction Protocol 29
4.2. Matching Algorithm 31
4.3. Resource Provision 31
4.4. Result Verification and Reputation 32
4.5. Dispute Resolution and Payment 32

5. Emulator **33**
5.1. Architecture . . . 34
5.1.1. Peer-to-Peer Network . . . 34
5.1.2. Emulator Components . . . 36
5.2. Configuration and Execution . . . 41
5.3. Scalability and Limitations . . . 45
5.4. Validation . . . 47

6. Evaluation **49**
6.1. Design . . . 49
6.2. Results . . . 51
6.2.1. PoX . . . 51
6.2.2. DPoS-BFT . . . 54
6.2.3. Network . . . 57
6.3. Discussion . . . 58
6.4. Limitations . . . 60

7. Conclusion **63**
7.1. Summary . . . 63
7.1.1. Realized Goals . . . 64
7.1.2. Open Goals . . . 65
7.2. Implications . . . 65
7.3. Future Work . . . 66

A. Protobuf Message Definitions **67**

B. Reproducibility **71**

List of Figures **75**

List of Tables **77**

Bibliography **79**

APSP All Pairs Shortest Path

AR Augmented Reality

ASN Autonomous System Number

BFT Byzantine Fault Tolerance

BP Block Producer

BTC The Bitcoin currency

CDN Content Delivery Network

DAG Directed Acyclic Graph

DLT Distributed Ledger Technology

DPoS Delegated Proof-of-Stake

FaaS Function-as-a-Service

GHOST Greedy Heaviest Observed Sub-Tree

LCR Longest Chain Rule

MEC Multi-Access Edge Computing

P2P Peer-to-Peer

PoX Proof-of-X, Generic proof-based consensus

QoS Quality of Service

RAN Radio Access Network

SLA Service Level Agreement

TCP Transmission Control Protocol

TSM Travelling Sales Man

Tx Transaction

IoT Internet-of-Things

VR Virtual Reality

zk Zero-Knowledge

1. Introduction

Recently, and especially with the advent of 5G technology, endeavors of next-generation applications such as IoT, autonomous vehicles, AR/VR or smart grids have received an ever increasing amount of attention in both industry and academia [1], [2]. Commonly, these applications experience stringent latency requirements of data and their processing, in order to maximize QoS and usability (i.e. reflecting user input within 20ms to avoid dizziness in VR [3], or controlling energy grids within 20ms of a lightning strike [4]). As a key enabler of low latency computation, *edge computing* describes the idea of utilizing compute resources in close proximity to end-users, as opposed to remote datacenters found in traditional cloud computing. Consequently, edge computing implies the need for vast, and geographically scattered provisioning of computing resources.

Amplified by the latest reignition of cryptocurrencies [5], Distributed Ledger Technology (DLT) represents a similarly attention-gathering development and proposed key enabler of decentralized applications [6], [7]. By replicating an irreversible ledger of transactions, DLT aims to provide a trustful environment without third-party authority and perform computation securely by the means of smart contracts [8].

Unsurprisingly, the idea of merging both technologies has gathered traction in recent years [9]. Edge computing and its applications tend to be distributed and localized by definition. In many regards, enabling a trustful and decentral environment as provided by DLT, for the use in edge computing, yields attractive benefits for a variety of edge applications [6]. To this end, highly distributed clients, such as IoT or autonomous vehicles, could take advantage of DLT in order to share data on a secure and decentralized communication medium [10]. Figure 1.1 describes the increase in research publications of DLT and edge computing, in an attempt to capture the recent interest in both technologies, as well as their amalgamation.

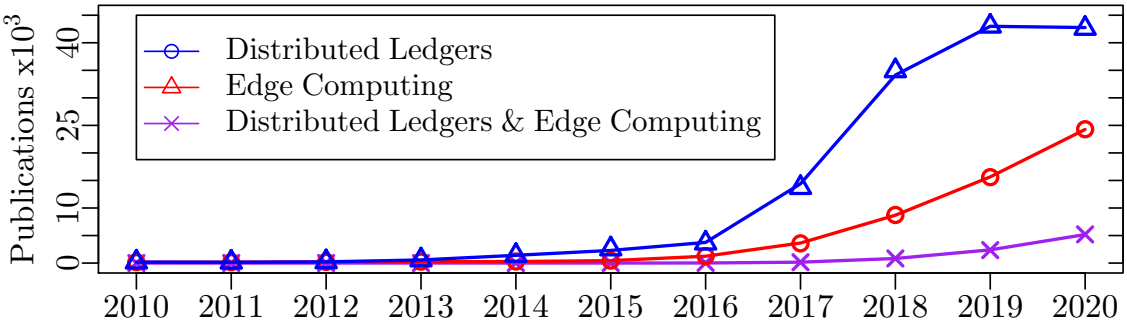


Figure 1.1.: Google Scholar publications mentioning distributed ledgers and edge computing in title or abstract.

1.1. Context

While DLT (and its subclass *blockchain*) has matured in the past decade with several commercial-grade systems such as Ethereum [11] or Hyperledger [12] offering platforms for cryptocurrencies, smart contracts and decentralized consensus, its application in the edge computing domain is still in early stages [9]. As a result, research endeavors are plentiful and explore the use of DLT in edge computing from many different directions.

A major challenge of edge computing can be seen in the installment, management and provisioning of geographically scattered, heterogeneous resources and services [13]. Here, a popular approach is the proposal of a (crowdsourced [14]) edge computing marketplace, using DLT as a decentralized auctioneer between resource bids [15], [16]. By matching under-utilized computing resources that are already part of the edge network, i.e. home desktops, mobile devices or dedicated servers, with resource requests of localized client applications and developers, an autonomous and self-sufficient edge computing ecosystem could be envisioned. Due to the latency-stringent nature of many edge applications, these marketplace operations are required to be highly performant. In order to enable on-demand edge resource provisioning, or hand-offs between different providers in case of crashes or client mobility, latency boundaries of resource bidding, matching and deployment should be reduced to maximize quality of service [17].

1.2. Problem Statement

In total, research interest in edge computing and distributed ledger technology is as high as never before, and attempts at merging both technologies seem promising [9]. Therein however, we notice a popular, and potentially *hype-driven*, tendency of treating DLT as a black-box solution for crowdsourcing, security or decentralized trust. Meanwhile, potential performance bottlenecks relevant to edge are tactfully ignored, or left open for future work (Section 3.2). Moreover, at the first glance, traditional blockchain architectures appear to deliver unsatisfying latency and throughput results towards their potential use in edge (i.e. Bitcoin blockchain creating one block every 10 minutes, leading to 7 Tx/s) [18]. Making a similar observation, authors of [19] pose the research question of finding a suitable distributed ledger for edge, and identify further requirements on DLT. As a result, we ask ourselves, how does an edge computing marketplace backed by blockchain look like in detail, and how applicable is a deployment of edge applications using DLT in reality? Our objective for this thesis is therefore threefold:

1. Investigate state-of-the-art distributed ledger technology and existing attempts at applying it in the realm of edge computing, in order to propose a decentralized and performance-optimized compute marketplace based on blockchain.

2. Evaluate potential latency boundaries, bottlenecks and capabilities imposed by the use of DLT in edge, and specifically in the proposed marketplace architecture.

3. Taking the strict performance requirements of next generation edge applications into account, formulate a realistic outlook on the use of distributed ledger technology in edge computing.

1.3. Thesis Approach and Outline

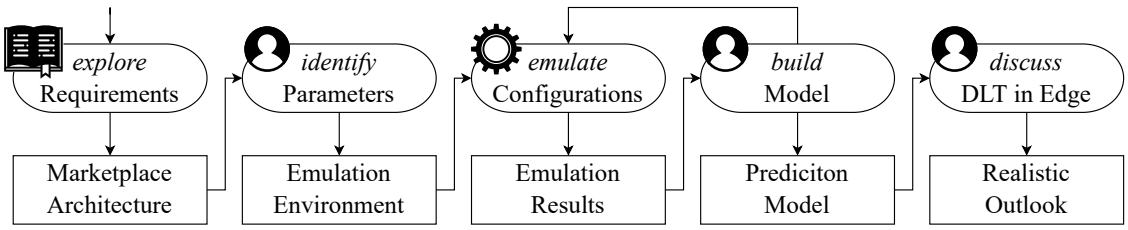


Figure 1.2.: Emulation-based approach showing literature-guided (book), manual (stick-figure) and automated (gear-wheel) activities (rounded rectangles) and corresponding artifacts (rectangles).

In general, we aim to evaluate the combination of DLT and edge computing by developing a framework allowing the emulation of generic blockchain networks. Compared to working with code of real blockchain architectures (i.e. Ethereum [11], Hyperledger [12]), an emulation-based approach is promising due to higher maintainability and versatility, while being more realistic than abstract simulations or models. Figure 1.2 depicts our approach as an activity diagram. In Chapters 2 and 3 we consult literature and related work in several directions. Firstly, we are interested in latency requirements of current and upcoming edge applications, in order to determine the performance margin available for underlying control structures. Secondly, we investigate existing DLT with the goal of devising components, parameters and performance variables to be included in our subsequent implementation. Finally, we explore previous applications and evaluations of DLT in edge, in order to discover common assumptions and goals, but also to pre-select relevant DLT based on existing results. In combination, we propose our version of the crowdsourced edge computing marketplace in Chapter 4, while focusing on optimizing interactions with the underlying blockchain. Based on our marketplace architecture and the identified DLT parameters, we develop our scalable emulation environment in Chapter 5. Subsequently, we devise an experimentation strategy encompassing the latency of common marketplace operations. Using our emulator, we conduct a variety of experiments in Chapter 6 and evaluate the performance of two configurable blockchain architectures in three different networks. Here, we are especially interested in potential trade-offs and boundaries occurring when optimizing blockchain performance for low-latency edge. We summarize our results by proposing four empirical models encompassing the most relevant parameters of blockchain performance. Finally, we discuss our results, identified problems and potential solutions, and conclude by giving a more realistic outlook on DLT in edge computing as part of Chapter 7.

2. Background

In this chapter, we provide necessary background on key concepts referred to in this work. Specifically, we introduce edge computing as an enabler of latency-stringent applications and devices by moving computation closer to the end-user. Secondly, we showcase the proposition of how crowdsourcing can be utilized to alleviate some of the challenges, especially regarding resource provisioning, that were identified in edge computing. Finally, we investigate distributed ledgers as an application of crowdsourcing and its potential in edge. To this end, we aim to highlight intricacies of state-of-the-art blockchain technology by shining a light on its promises and hindrances towards an adoption in the edge.

2.1. Edge Computing

With the ever progressing development of information technology, the number of devices, sensors and applications inhabiting the sphere of computation and networking proliferates greatly. [20] predicts 55.7 billion connected devices worldwide by 2025, 75% of which will be interacting with IoT platforms. Similarly, some sources project the global cloud computing market to reach $1 trillion by 2026 [21]. Naturally, such a rapid growth and adoption of interconnected solutions also poses complex technical challenges to the industry. Ensuring Quality of Service (QoS) in regards to latency, throughput, fault-tolerance and more, in an environment reliant on the performance of centralized server facilities, is increasingly difficult for traditional cloud-based architectures. Consequently, the idea of moving computation, storage and networking capabilities closer towards their consumers, in an attempt to reduce traffic and congestion of data, becomes ever more attractive [22], [23].

Edge Computing (here also *Fog Computing*) describes the paradigm of extending (or even co-locating) cloud computing capabilities to the network edge. Initially introduced via the proposal of *cloudlets* [24], describing compact datacenters deployed on WiFi access points or LTE base stations, edge computing aims to improve QoS of latency-stringent applications by enabling on-site aggregation, preprocessing, delivery and analysis of data. Since then, only few promising initiatives have been taken. As such, Multi-Access Edge Computing (MEC) as coined by the European Telecommunications Standards Institute (ETSI) [25], describes placing edge servers within the Radio Access Network (RAN) of mobile devices, i.e. near telecommunication masts [26]. RAN allows applications to be enriched with real-time radio network information such as location or cell load, enabling various optimizations based on radio and network conditions [27].

2.1.1. Edge Offloading

The existence of localized compute resources as found in MEC additionally fuels the idea of (on-demand) edge offloading [28], [29]. Similar to its cloud computing counter part, edge offloading describes the provisioning of services, or the execution of on-demand tasks, as in i.e. Function-as-a-Service (FaaS), on edge resources. Especially mobile or IoT devices provide comparatively low computing power while simultaneously benefiting from low energy consumption [30]. Offloading computations to the cloud alleviates both of these issues but comes with a tradeoff in latency and thus, user experience [31]. The process of provisioning edge resources in a fashion similar to current cloud practices is an ongoing research question and also part of this work (Chapters 3 and 4). In general, the problem can be divided into five steps, namely *resource discovery*, *negotiation*, *service placement*, *execution* and *verification* [32].

Therein, resource discovery is concerned with finding suitable edge providers in close proximity to the targeted clients. Ideally, an optimal match between the client's requirements in terms of computing capabilities, latency or cost and the resource offer is to be found. During the negotiation step, a service level agreement (SLA) between resource requester and provider is reached. Additionally, payments from both parties may be transferred into an escrow account, to ensure compensation or rewards in case of contractual (un-) fulfillments. After negotiation of terms, the service or task is placed onto the edge resource, to be then executed or accessed by clients. Depending on individual SLAs, results may be verified during a final step, before they are accepted by clients and funds are released from escrow.

2.1.2. Applications

Ever since its emergence, countless proposals and case studies on applications benefiting from, or even being born by edge computing have been published [26], [27]. The following aims to provide a brief overview and summarize important requirements relevant to our work. Specifically, we notice that many applications have strict latency and bandwidth requirements, and as a result, propose edge computing as an enabling technology [33]. Figure 2.1 depicts these requirements for some driving edge applications along with their expected marketshare [3]. The authors additionally identify an edge computing feasibility zone of 200 milliseconds latency or below. For applications with less stringent requirements, a traditional cloud-based deployment might be more beneficial, especially in terms of cost effectiveness [3].

Content Delivery

As a pioneer of edge computing, Content Delivery Networks (CDNs) attempt to reduce latency and bandwidth usage of data consumption by acting as localized caches. As such, they provide static HTML content and web components, but also video and other media streams. Therein, CDNs additionally offer more advanced forms of congestion control, such as reducing graphics resolution of videos streamed by a large amount of

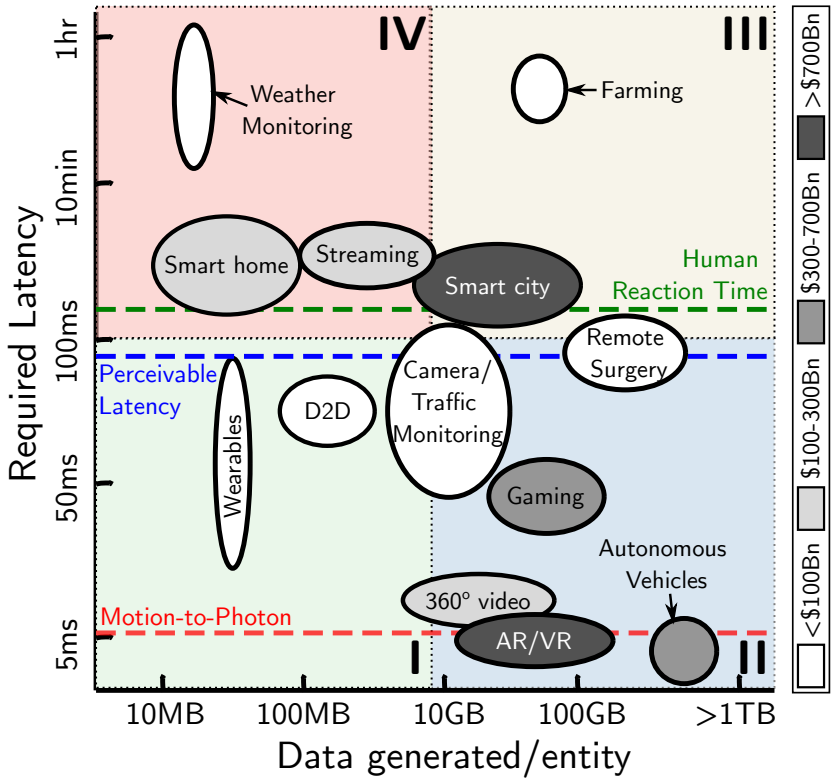


Figure 2.1.: Strictness of bandwidth and latency requirements in driving edge applications. Color denotes expected market share by 2025 [3].

users, in order to avert denial of service [26]. In 2018, North American and European regions experienced median latencies as low as 20ms to prevalent, commercial CDNs [34]. Here, especially storage and access control capabilities are moved closer to their direct consumption.

Virtual and Augmented Reality

Virtual and Augmented Reality (VR/AR) describes the concept of simulating or enriching a real environment through the use of virtual objects. To this end, the use of head-up displays, and mobile technology such as Google Goggles [35] usually requires task offloading for extensive computation, and high bandwidth in order to supply imagery with minimal latency [36]. Here, the most stringent latency boundaries and QoS requirements primarily depend on the human vestibular system requiring sensory inputs and interactions to be synchronized. In order to avoid the feeling of sickness or dizziness, these latencies should stay below 20 ms, the majority of which is however taken up by the display technology [3]. In an attempt to tackle these and related problems, [37] proposes the deployment of GPU clusters in edge, to be used at low latency by devices with limited hardware.

Smart Grid

In a smart grid network, appliances and energy resources use distributed smart meters to receive and transmit measurements of energy consumption and production. Here, edge computing is proposed to perform supervisory control and data acquisition (SCADA) within the energy network. The resulting system is intended to maintain and stabilize the power grid by balancing and scaling its load autonomously based on information provided by the smart meters [38]. To properly respond to changes in the grid, many applications have strict latency requirements in the range of 100 milliseconds to 5 seconds [39]. Critical applications dealing with, i.e. lightning strikes require even lower latencies of 20 ms or below [4].

Video Analytics and Rendering

Algorithms for object detection and classification in video streams concerned with license plate recognition, face recognition or home security surveillance have high computational complexity and are unsupported by traditional video capturing devices. However, routing video streams to the cloud for processing [40] consumes large amounts of bandwidth. Instead, video analysis could be moved closer to the capturing device in edge to avoid network congestion, and also improve the latency of results [23]. Similar to VR/AR applications, edge computing could again provide GPU clusters for rendering or machine learning tasks submitted by clients lacking the required hardware capabilities [37]. Here, the advantages of edge computing primarily manifest in bandwidth gains by processing video streams locally. Nevertheless, in use cases such as traffic monitoring, latencies the below human reaction time of 250 ms may still be required [3].

Internet-of-Things

The Internet-of-Things (IoT) envisions a wide array of physical and embedded devices to be interconnected using data centers in order to generate new magnitudes of data and value. Compute resources are again placed in the network edge to reduce latency and congestion resulting from enormous amounts of data. *Smart* devices such as phones, vehicles or general sensors and actuators are leveraged to propose a variety of new paradigms, e.g. *smart home*, *smart city* or *smart mobility* [33]. Especially in the entertainment area (i.a. TV, lights, heating), latency requirements appear to be less stringent, and usually position around human reaction time (250 ms). However, in applications concerned with traffic monitoring or autonomous vehicles, lower latencies might apply [3], [27]. Here, Vehicular Adhoc Networks (VANETs) [41] are proposed to enable communication between vehicles, mobility predictions and environmental knowledge sharing. In order to be used for road safety, latencies around 10 ms are necessary [4].

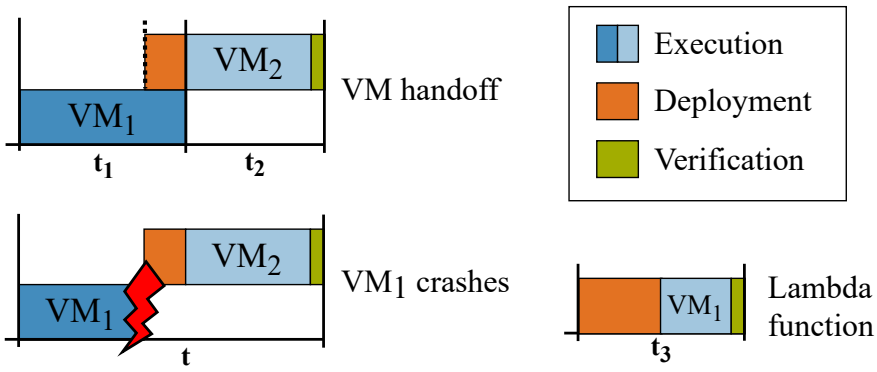


Figure 2.2.: Total time $t_1 + t_2 = t >> t_3$ until the verified completion of an offloaded task in different scenarios. Here, deployment consists of *resource discovery*, *negotiation* and *service placement*.

2.1.3. Challenges

Compared to cloud computing, edge computing violates the *economies-of-scale* paradigm in almost every regard. Centralizing compute resources within a single authority promises tremendous benefits in maintenance and deployment costs, energy deals, as well as a reduction of personnel expenses for administration and management [42]. As a result for application developers, cloud computing offers attractive pay-as-you-go deals for a wide variety of resources and deployment schemes. Furthermore, datacenter networks are highly optimized for technical performance and robustness, enabling responsive load-balancing and elasticity on demand [43]. Contrary, edge resources are decentralized and localized by design, necessitating a massive administrative overhead [44]. On top of that, requirements on elasticity, hand-offs and performance may be even more stringent in edge when compared to cloud computing. Especially when considering the potential mobility of clients, i.e. in automotive, transferring tasks in case of crashes, or offering FaaS, migrating and provisioning edge resources within the latency requirements of their applications becomes vital (Figure 2.2). The shocking realization is that not only edge resources but also its management plane needs to satisfy edge requirements in an extremely heterogeneous environment. To this end, communication technology enabled by the emerging 5G already aims for sub 10 ms latencies in the control plane [45]. Finally, we note that stringent latency and throughput requirements of applications are one of the main drivers of edge computing technology. Failure of meeting these requirements may lead to traditional cloud computing being the more attractive solution in many cases [3].

2.2. Crowdsourcing in Edge

Crowdsourcing can be defined as the practice of gathering and aggregating services, data or any other resource, by the collective efforts of multiple parties, i.e. the general

public [46]. For the area of edge resource provisioning, crowdsourcing appears to be the perfect match. Today, the computational landscape is filled with personal computers, mobile phones or intelligent cars with computational capabilities. These resources are oftentimes underutilized and placed exactly where they are needed within the edge of the network. By matching resource requests of edge applications and IoT devices to the crowd's resource offers, an autonomous and self-sufficient system could be envisioned. Overall, the crowdsourcing paradigm promises a cost effective method of pooling resources by offloading administration to individual resource owners. Furthermore, the idea of *crowd intelligence* suggests qualitative benefits in task execution, by constructing a global view of the edge computing landscape and enabling best possible matches between resource requests and offers [47]. Therein, edge computing undoubtedly satisfies all four requirements of a crowdsourced system as defined by [48], namley *diversity* in devices and tasks, *independence* by autonomous operation, *decentralization* without an arbiting governance and *aggregation* of devices and tasks to improve QoS.

Previously, *volunteer cloud computing* [14], describing the volunteered provision of computing resources for distributed scientific computations, has been very successful. Projects such as SETI@home and BOINC [49] have attracted millions of participants worldwide, sustaining a processing rate of over 28 PetaFLOPS today [50]. However, existing attempts of implementing a general crowdsourced compute market such as in EDGE [51], Sonm [52] or Dfinity [53] (Section 3.2) appear to be largely unpopular. As identified in Section 2.1.3, QoS is a vital requirement for edge computing. Especially under the influence of commodity hardware introduced by a crowdsourced edge, the blurry vision of a truly decentralized and crowdsourced platform for edge orchestration becomes even grimmer.

2.3. Distributed Ledger Technology

As unarguably one of the major drivers of crowdsourcing in recent times, DLT is influencing the industry landscape tremendously. With the rise of cryptocurrencies such as Bitcoin [18] and Ethereum [11] combined with the advent of *smart contracts*, many fields attempt to adopt DLT in order to leverage compelling promises such as decentralized trust, distributed computation or even security by design [54]. In this work, we shed a light on the roles and pitfalls of DLT in edge computing from two perspectives. Firstly, we investigate the use of DLT as an enabling technology for a variety of edge computing applications in general. Secondly, we explore DLT specifically as a distributed auctioneer of crowdsourced edge computing marketplaces, offering an open exchange and matching of edge computing tasks to requested resources. In the following, we aim to highlight some of the intricacies and parameters of distributed ledger and blockchain technology that are heavily influencing its potentiality of an adoption in edge.

Distributed ledger technology (DLT) can be described as a decentralized, ordered record of transactions. In its simplest form, DLT consists of a peer-to-peer network

of nodes[1], each maintaining a replica of the ledger. By adhering to a shared protocol and cryptographic standard, an immutable total order over all transactions is enforced. With the emergence of *smart contracts* [54], allowing the decentralized execution of code using transactions as input parameters, DLTs have entered the field of general purpose replicated state machines. To this end, DLTs have transcended their original field of cryptocurrencies (e.g. Bitcoin [18]) and aim to be an enabling technology for a variety of decentralized applications [54], such as electronic voting [55], data provenance [56], or item sharing [57].

Clients of the distributed ledger create a public and private key pair known as a *wallet*. Using the wallet's asymmetric cryptography, new transactions are signed and sent to a peer of the client's choice. The transaction is then further propagated through the network and included within the ledger. Here, a transaction could serve the purpose of sending cryptocurrency to another client, or to initiate the execution of a replicated smart contract. Replaying all transactions from the beginning constitutes the same state on all replicating nodes. This is described as the *world state*.

As a subclass of DLT, a blockchain maintains the distributed ledger by batching transactions into immutable blocks with each block referencing a previous one. Blocks are created and replicated using a consensus protocol. Such a protocol can be divided into several steps consisting of *block proposal*, *propagation*, *validation*, *finalization* and an *incentive mechanism* [8], [58]. Moreover, peers can participate in the blockchain in a variety of ways. For instance, participation can be *public* or *private* - based on blockchain's read access - as well as *permissionless* or *permissioned* - based on a peer's ability to participate in the consensus [59]. Restricting access to the ledger usually implies the need for a separate registration and certification authority. Consequently, while participating in permissionless DLTs tends to be pseudo-anonymous (Bitcoin [18]), identities of permissioned consensus nodes are revealed to enable traditional Byzantine Fault Tolerant (BFT) protocols. In permissionless DLTs, users are typically *incentivized* to participate in the consensus protocol by monetary gains enabled by *block rewards* or *transaction fees*. Consensus nodes in permissioned DLTs may instead be operated explicitly by the platform owner [8], [58].

In general, DLT can be described as a layered architecture (Figure 2.3). For our purpose, especially parameters influencing the performance of lower levels and DLT internals are of importance. To provide a summary on consensus protocols (Layer II), we further categorize DLT into four areas consisting of *proof-based*, *BFT-based* and *committee-based* blockchains, as well as *Directed Acyclic Graph (DAG) based* DLT (Table 2.1).

2.3.1. Proof-Based Blockchains

Proof-based blockchains propose new blocks by employing a distributed, pseudo-random lottery, e.g. proof-of-work (PoW [18]) or proof-of-stake (PoS [60]), and effectively simulate a random leader election. The lottery winner (i.e. the miner) includes a

[1] In this work, we use the terms *peer* and *node* interchangeably.

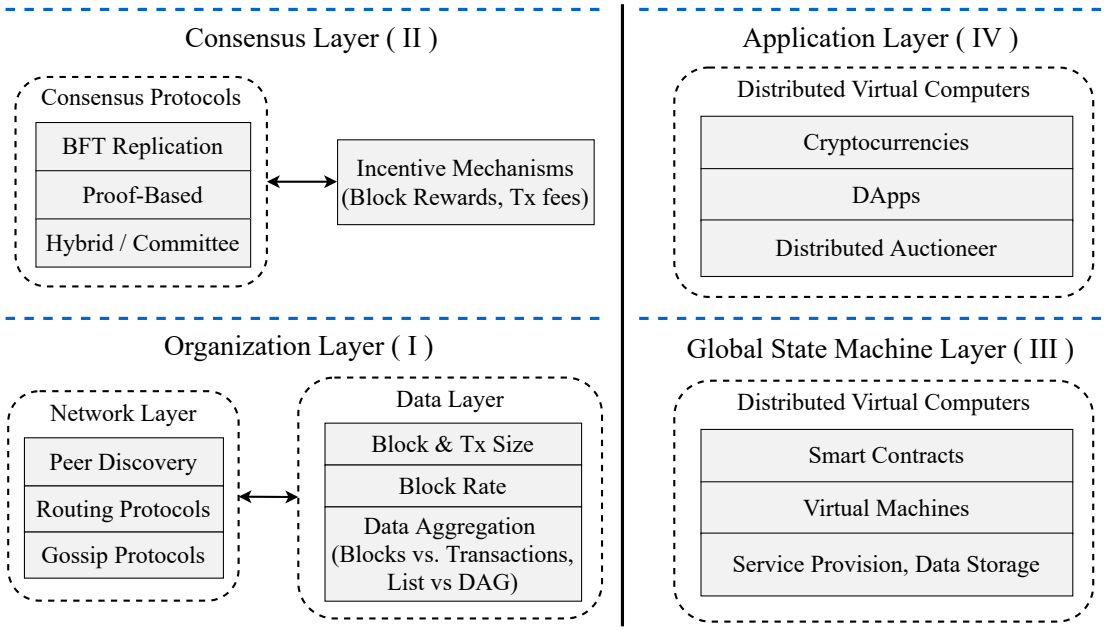


Figure 2.3.: Components of the layered DLT architecture.

zero-knowledge proof of their win within a new block, making it verifiable to other peers. Equation (2.1) describes how the probability Pr_i^{win} of a node $i \in N$ winning the consensus lottery, is directly proportional to a node's verifiable share of *lottery power* w_i, i.e. its computational power (PoW) or the amount of staked coins (PoS) [58].

$$Pr_i^{win} = \frac{w_i}{\sum_{j \in N} w_j}, \forall i \in N \tag{2.1}$$

Proof-based blockchains additionally specify a *difficulty* of the lottery, with the goal of regulating the amount of winners within a given time frame. Consequently, we can model the proof-based lottery as a Poisson process with rate λ, describing the rate of new blocks to be created by the entire network per time unit. Since the combination of N independent Poisson processes with rates λ_i still describes a Poisson process, we can further model the lotteries conducted by individual peers using their winning probabilities as shown in Equation (2.2) [58].

$$\lambda_i = \lambda Pr_i^{win}, \forall i \in N \tag{2.2}$$

Apart from the ability to propose the next block, winning the lottery traditionally entails a reward according to the employed incentive mechanism, i.e. in form of a cryptocurrency. The reward consists of transaction fees for all transactions included in the block which are paid by the submitting clients, as well as block rewards used for minting new coins (i.e. *coinbase transaction* in Bitcoin [18]). Without the existence of an incentive mechanism, consensus nodes of proof-based blockchains would operate

Table 2.1.: Consensus protocol summary [8].

Cat.	Proposal	Propagation	Validation	Finalization
PoX	PoW, PoS, PoR	Gossip to peers	zk-Check	LCR, GHOST
BFT	Round-robin, on request	Broadcast	Signature check	Mutual state agreement
Com.	Round-robin by elected delegates	Broadcast btw. delegates, gossip to peers	Proposer eligibility check	(Pipelined) BFT + LCR, GHOST
DAG	PoW, parent approval	Gossip to peers	Check zk-proof & reference to parents	DAG total order

at a massive monetary loss. By forcing participants to commit valuable resources such as computational power, memory or their staked coins, the danger of a sybill attacker controlling many nodes without overhead, in order to skew winning probabilities in their favor, is averted. In proof-based blockchains, nodes are anonymous and equal in their role of proposing and validating blocks. To this end, transactions and blocks are propagated and replicated in the peer-to-peer network using gossiping. Here, we differentiate between advertisement-based gossiping (Figure 2.4) and unsolicited block push (Figure 2.5) [8]. While in the latter strategy nodes immediately transmit new blocks to their peers in order to reduce latency, advertisement-based gossiping only propagates block headers to minimize bandwidth usage. Consequently, blocks are only transmitted after an explicit request is received. Due to the equality of peers and open participation in the random leader selection employed by lottery. Proof-based approaches are primarily adopted by public-permissionless blockchains and cryptocurrencies such as Bitcoin [18] or Ethereum [11].

Proof-of-Work

In PoW blockchains such as Bitcoin [18], the distributed lottery is represented by a cryptographic puzzle to be solved by all peers. More specifically, for a block to be accepted by peers, its *hash* is required to be smaller than a target difficulty. This is achieved by changing a nonce value within the block. Consequently, w_i (Equation (2.1)) describes the rate at which a node i is able to calculate new hashes. For a hash function $H(\cdot)$ generating bit strings of length L and a difficulty target $h < L$, the condition to be fulfilled in order to propose a new block x is therefore precisely:

$$H(x||nonce) \leq 2^{L-h} \tag{2.3}$$

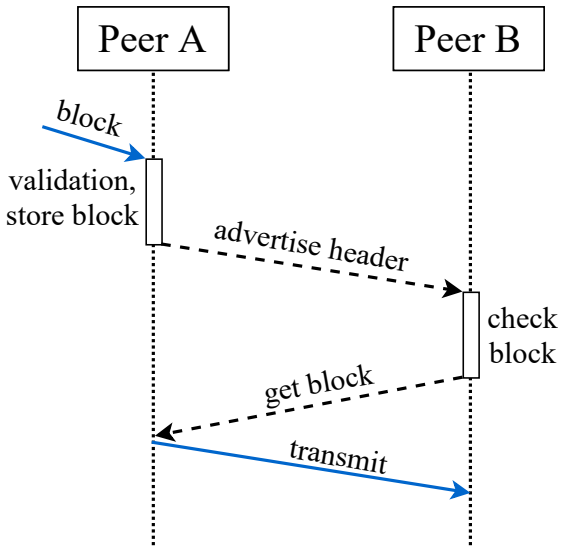


Figure 2.4.: Gossip of advertisements [8].

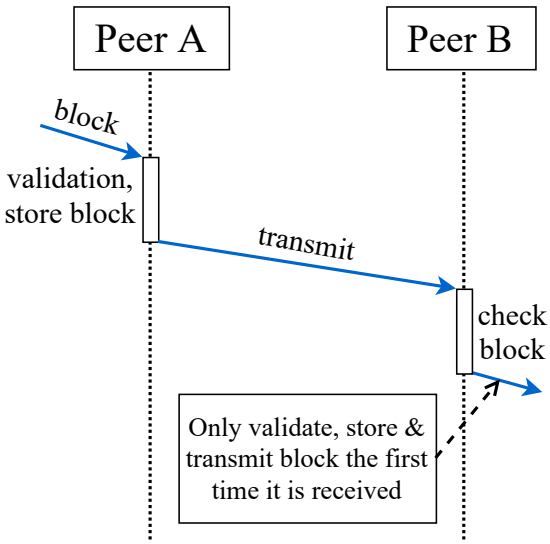


Figure 2.5.: Unsolicited block push.

Assuming $H(\cdot)$ is an irreversible cryptographic hash function, satisfying this condition cannot be more efficient than systematically trying every possible value of the nonce [58]. Therefore, the probability of finding a valid hash is a Bernoulli trial of Equation (2.3). Since a repeated Bernoulli trial in short intervals converges to a Poisson process, our model of proof-based blockchains as a Poisson process in Equation (2.2) is valid [61]. During the validation phase of a block, the proof-of-work calculated in the nonce can then easily be validated by a single call to $H(\cdot)$.

Proof-of-Stake

Initially proposed by Peercoin [62], PoS aims to reduce the energy consumption of large scale PoW mining. To this end, Peercoin introduces the notion of *coin age* describing the duration a specific amount of coins was held by a miner. The miner is then able to calculate a traditional PoW solution but reduces the difficulty target by the miner's personal coin age. As transactions are publicly stored on chain, coin age is easily verifiable by other miners. In [63] and related work, the PoW concept is replaced entirely by an algorithm described as *follow-the-satoshi*. Therein, a *satoshi* is the minimum token unit carried by the blockchain. By indexing all tokens in circulation, a complete distribution of coins and their owners is created. In a simplified protocol, the header of block $t-1$ is used to pseudo-randomly determine the winner of block t. To this end, the output of $H(header(t-1))$ is used to search the satoshi index. The owner of the chosen satoshi is then selected as the winner of the new block [58]. As a result, w_i of Equation (2.1) describes the share of coins owned by node i, the remaining equations follow analogously. In order to be the verifiable owner of the indexed satoshi, the PoS lottery winner includes their signature within the new block.

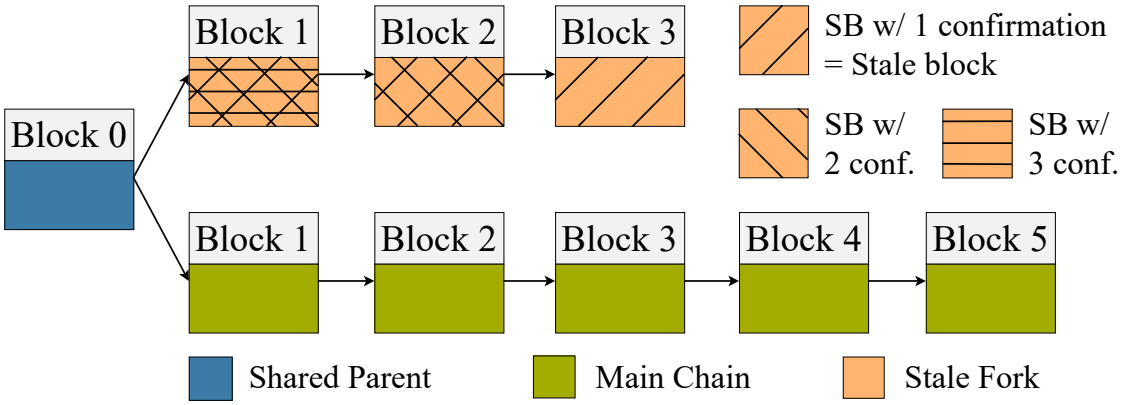


Figure 2.6.: Differentiating between stale blocks and confirmed stale blocks.

Other Proof-Based Protocols

Similar to PoW and PoS, other proof-based protocols have been proposed. To this end, Proof-of-Retrievability (PoR) [64] requires staking resources, i.a. memory, similar to PoW. Other approaches such as Proof-of-Useful-Work (PoUW) [65] aim to replace PoW with less wasteful alternatives such as calculating prime number chains. Proof-of-Elapsed-Time (PoET) as implemented in Hyperledger Sawtooth [66] or Proof-of-Luck (PoL) [67] attempt to simulate the process of PoW by using Trusted Execution Environments (TEEs) to wait for a random amount of time before a new block can be proposed by a node.

Stale Blocks and Confirmations

Revolutionary, and in some cases a drawback of proof-based consensus, is the concept of probabilistic finality. In case of multiple lottery winners in quick succession, along with network latencies, proof-based blockchains can experience *forks*, resulting in multiple blocks referencing the same parent block as shown in Figures 2.6 and 2.7. In order to restore consensus and agree on the same transaction history, fork resolution mechanisms such as the longest chain rule (LCR), GHOST [68] and additional confirmation blocks allow peers to reach eventual, probabilistic consensus finality by deciding on one of the forks (know as the *main chain*). In this work, once a transaction is included in a block, it is said to have received *one* confirmation. Every additional child block added in accordance of the applied fork resolution strategy further increments the transaction's total confirmation number. As a result, transactions with many confirmations are less likely to end up in a fork outside of the main chain (Chapter 6). Once a transaction receives enough confirmations to be deemed *irreversible* by an application, following actions relying on the acceptance of the transaction are initiated (i.e. shipping a product that was purchased using cryptocurrency). In the following, the latency of a transaction receiving enough confirmations to be accepted at the application level is termed *transaction latency* or *confirmation latency*.

Blocks outside of the main blockchain, as determined by the fork resolution, are

known as *stale blocks*. Transactions that are only present in stale blocks are automatically released back into the *memory pool*, in order to be included in future blocks of the main chain. Consequently, we additionally differentiate between general stale blocks and *confirmed* stale blocks which received enough confirmations for their transactions to be wrongfully accepted by an application, before ultimately turning out to be stale (Figure 2.6). Here, the latter potentially implies the existence of double-spends or the necessity of rollback operations after the confirmed block was detected as stale by the application [69]. In general, the creation of stale blocks represents a waste of computing power, storage and bandwidth. As a result, the percentage of stale blocks can be used as a characteristic attribute, and as an efficiency indicator of the blockchain architecture.

Longest-Chain Rule

Originally introduced by Bitcoin [18], the longest-chain rule (LCR) resolves forks by always selecting the chain with the largest amount of consecutively invested lottery power. Ties may be resolved randomly or on a first-come-first-serve basis. Here, in the case of each block having the same difficulty, LCR is simplified to selecting forks containing the highest number of consecutive blocks. To this end, a known vulnerability of proof-based blockchains is the *alternative history* or *51%* attack [18]. Therein, an attacking network secretly works on an alternative version of the blockchain, starting at a common block. Once the secret chain reaches a higher block count than the longest currently accepted chain, the secret blocks are released, effectively rewriting the transaction history (Figure 2.7). By reverting previously accepted transactions, a double-spend attack can be conducted, leading to proof-based blockchain's theoretical fault tolerance of 50% lottery power to be controlled by a benign network [18].

GHOST

Contrary to popular believe, research has shown that double-spend attacks on LCR can be conducted with less than 50% lottery power due to the existence of stale blocks [70]. Stale blocks are excluded of the longest chain and consequently do not protect against alternative history attacks, despite using up valuable lottery power during their creation. As a result, especially networks creating many stale blocks are more vulnerable to double-spend attacks [70]. To increase the resistance against attacks, the Greedy-Heaviest-Observed-Sub-Tree (GHOST) rule includes stale blocks during fork resolution. Here, instead of the longest chain, the largest sub-tree is selected at every fork. In order to avoid calculating sub-tree sizes starting from the genesis block as shown in Figure 2.7, GHOST additionally defines a *depth* parameter from which tree sizes are calculated. GHOST was initially proposed in [68] and is partially implemented in the Ethereum blockchain [11], allowing the network to create blocks at much higher rates as opposed to LCR-based blockchains such as Bitcoin [18].

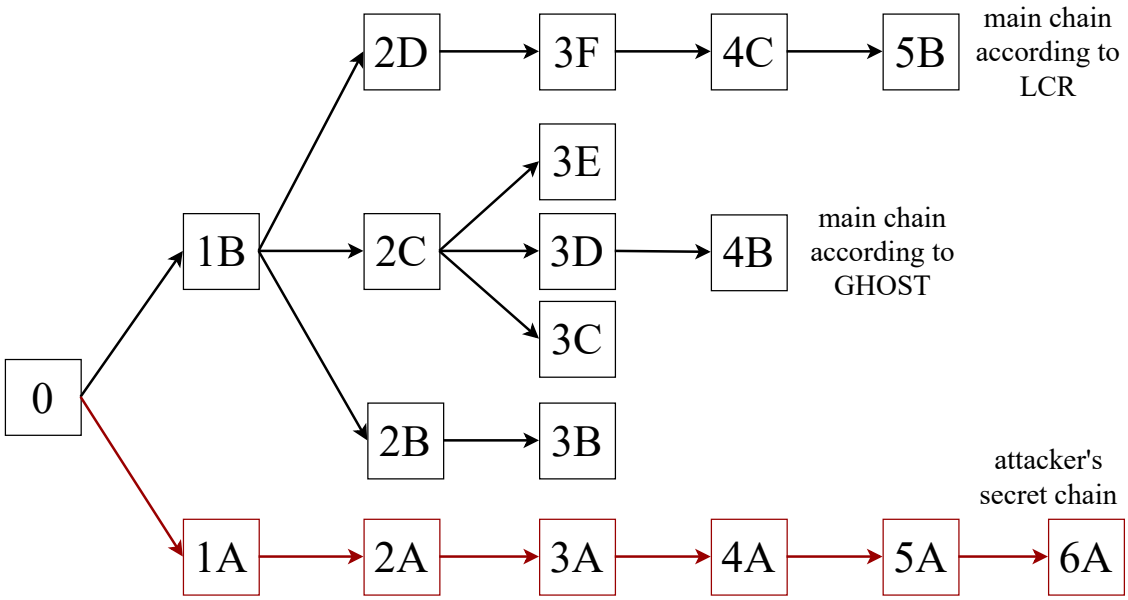


Figure 2.7.: Using GHOST to decrease vulnerability against alternative history attacks by including stale blocks during fork resolution [68].

2.3.2. BFT Blockchains

In contrast to proof-based consensus, BFT blockchains are maintained by identifiable consensus nodes adapting traditional protocols known from state machine replication. To this end, a *Crash-Fault Tolerant* (CFT) protocol such as *Paxos* [71] (Figure 2.8) allows up to f simultaneous crash-failures in a network of $2f + 1$ nodes. To additionally allow for byzantine-failures introducing erroneous or malicious results, *Byantine-Fault Tolerant* (BFT) protocols such as *Practical BFT* [72] (Figure 2.9), introduce an additional synchronization step to compensate for f simultaneous failures in a network of $3f + 1$ nodes at the cost of higher message complexity [8], [58]. BFT protocols are adapted to fit DLT requirements of reaching consensus on blocks by operating in multiple rounds. In Casper FFG [73], all nodes propose blocks and transactions during individual checkpoint cycles. Out of all proposed blocks, finality is reached on one block per round. Similarly, Hyperledger Indy [12] makes use of *Redundant BFT* [74] to reach consensus on blocks proposed by a primary node. Here, blocks are signed by the proposing leader, allowing for signature checks during the consensus validation step. In a different approach, Hyperledger Fabric [75] extracts a separate ordering service, enabling modular deployment of pluggable CFT consensus protocols such as Kafka [76] or Raft [77]. The support of BFT algorithms is planned for the future. In the Cosmos Hub architecture [78], Tendermint [79] describes a variation of PBFT to be used for blockchains. Therein, *n validator nodes* propose new blocks in order, until one of the blocks is accepted by more than $2n/3$ validators in accordance to PBFT. The procedure is repeated for every new height of the blockchain.

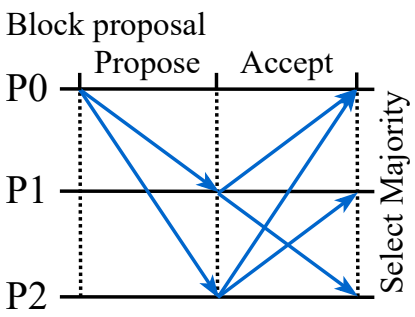


Figure 2.8.: Paxos (CFT) [8].

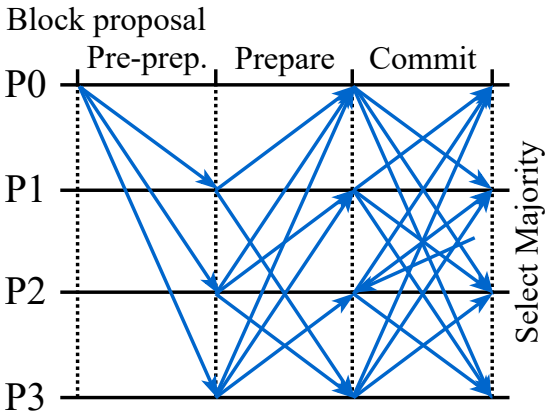


Figure 2.9.: PBFT [8].

Compared to proof-based blockchains, BFT blockchains achieve total finality without stale blocks at the cost of lower fault-tolerance and scalability, by reaching consensus immediately for every individual block. Especially the depicted quadratic message complexity in terms of participating nodes, limits their deployment to small networks [80]. Additionally, due to the nature of BFT protocols requiring nodes to be identifiable, BFT-based blockchains imply an operation in permissioned mode.

2.3.3. Committee-Based Blockchains

Hybrid or committee-based blockchains attempt to improve upon the scalability shortcomings of BFT-based blockchains by selecting a smaller subset of nodes to be used for consensus. In a public-permissionless setting, this might be enabled by allowing peers to vote for a set of delegates, which in turn propose the next set of blocks [81]–[83]. In permissioned operation mode, a committee may instead be hand-picked, or chosen by a separate *selection mechanism*. Delegates reach consensus on blocks using BFT protocols of the previous section (i.e. Tendermint in Cosmos Hub [78]) or pipelined BFT protocols with slightly relaxed finality (i.e. EOS [84]).

DPoS-BFT

In delegated proof-of-stake using pipelined BFT (DPoS-BFT) as found in EOS [84], delegates are elected in rounds by peers staking coins to vote. Subsequently, delegates each propose multiple blocks in round-robin fashion. To this end, block production intervals are deterministically assigned to consensus nodes in a way ensuring that only one node has authority over the blockchain at any given time. Here, EOS introduces the notion of a *last irreversible block*, describing the latest block on the chain which was extended by blocks of at least $^2/_3$ of remaining block producers (Figure 2.11). The block is described as irreversible under the assumption that a Block Producer (BP) building

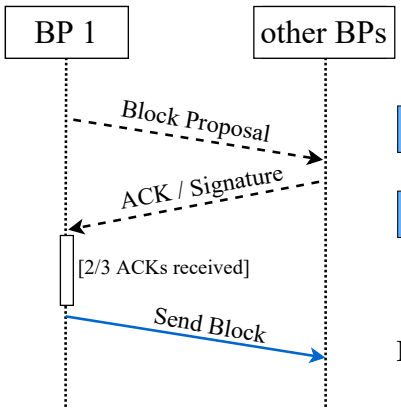


Figure 2.10.: Pipelined BFT.

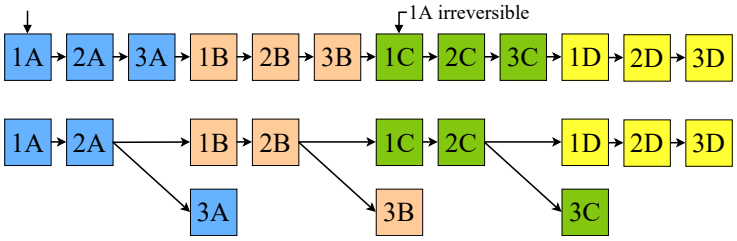


Figure 2.11.: Stale and irreversible blocks in DPoS-BFT with 4 block producers.

upon a block for the first time will continue to do so in the future. Any deviation of this assumption describes a verifiable byzantine failure, leading to an exclusion of the violating consensus node. By limiting the authority to one node at a time and allowing BPs to create multiple blocks consecutively, EOS achieves block rates of 2 blocks per second, representing a significant improvement over other permissionless approaches [84]. However, in case of blocks produced during the previous interval not arriving at the next BP in time, stale-blocks may still be created during the authority hand-off between two BPs (Figure 2.11). Therefore, an obvious optimization is ordering BPs according to their shortest Hamilton cycle, returned by solving the underlying traveling-salesman problem [85]. Since waiting for a block to be confirmed by blocks of $^2/_3$ remaining block producers increases a transaction's confirmation latency significantly, EOS additionally introduces a *pipelined* BFT protocol. Therein, BPs broadcast a block proposal to all other BPs, before sending out the new block (Figure 2.10). Only if at least $^2/_3$ consensus nodes reply with an acknowledgment, signaling their intend of extending upon the proposed block if received in time, the completed block is gossiped through the remaining network. This allows clients to skip the waiting time for a block to become irreversible and enables EOS to provide low transaction latencies with high probabilistic finality.

2.3.4. DAG-Based DLT

Finally, DLTs based on Directed Acyclic Graphs (DAGs) aim to diverge from the linearity of Bitcoin's original blockchain design, in an attempt of increasing transaction throughput. Here, the key insight is that restricting the underlying datastructure to linear growth inherently limits its expansion and scalability, especially in the presence of natural transaction concurrency. Instead, DAG-based DLT embraces concurrency by allowing blocks and transactions to reference multiple parents. To restore a consistent transaction history, the resulting partial transaction ordering imposed by the DAG is then transformed into a total order using a common procedure [8].

BlockDAGs

Similar to blockchains, blockDAGs aggregate transactions into blocks before including them in the datastructure. Here, SPECTRE [86] requires nodes to point a new block to all child-less *tips* of the current DAG by including their hashes and calculating a PoW. The nodes then initiate a recursive procedure to determine a pairwise order of blocks, eliminating all conflicting transactions [8]. As a result, the probability of a transaction being eliminated decreases proportionally to its depth in the DAG, leading to similar probabilistic finality and confirmation latency as observed in blockchains.

TxDAGs

Transaction-based DAGs (TxDAGs) further deviate from the blockchain architecture by absolving of blocks entirely, and instead creating a DAG of transactions. To this end, IOTA Tangle [87] requires new transactions to reference and approve two, preferably unapproved, transaction tips using PoW. In case of multiple conflicting tips, transactions with the highest acceptance probability are chosen using a *tip-selection scheme* [8].

For our purpose of evaluating edge computing marketplaces, purely transaction-based platforms such as IOTA are less attractive. Since the majority of edge marketplace approaches envision conducting auctions on chain (Chapter 4), they would inherently benefit from transactions being aggregated into blocks. Consequently, transaction-based platforms do not fit these marketplace requirements.

2.3.5. Trade-offs in DLT

Between the presented platforms, choosing the best suited blockchain technology for edge, or for the edge computing marketplace, requires consideration of a variety of trade-offs [59] - most notable being performance vs. security, fault-tolerance and decentralization. Table 2.2 summarizes a variety of other trade-offs to be considered when selecting between the discussed categories of existing DLT platforms. Overall, currently operating blockchains appear to be un-optimized for edge computing use cases, mainly represented by their significant confirmation latency. Specifically, cryptocurrencies employ low block rates and require many confirmations in order to remain secure and consistent (i.e. 6 confirmations at 10 minutes per block in Bitcoin [18]). While on the other hand BFT-based blockchains appear to be more performance oriented, latencies are limited by synchronization and the computational overhead resulting from serializing and deserializing a cubic amount of messages [80]. Nevertheless, state-of-the-art blockchain technology exposes many configurable parameters, allowing us to investigate their applicability in edge using reparameterization by measuring trade-offs, bottlenecks, limitations and capabilities. To quantify these trade-offs, multiple internal performance metrics of different blockchain technologies need careful investigation. In this work, we focus our attention on analyzing performance (in throughput and confirmation latency), security and consistency (in stale blocks) as well as decentralization (in the number of consensus nodes and the duration of block production intervals) [88]. As previously

Table 2.2.: Consensus trade-off overview [8].

Cat.	Participation	Complexity	Finality	Fault Tolerance	Latency
PoX	Permissionless	$O(1)$	Probabilistc	50% lottery power	minutes
BFT	Permissioned	$O(n^2) - O(n^3)$	Total	33% nodes	seconds
Com.	Either	$O(n) - O(n^3)$ delegates	Either	33% delegates	seconds
DAG	Permissionless	$O(1)$	Probabilistc	50% lottery power, participants	seconds, minutes

mentioned, we additionally differentiate between general stale blocks and *confirmed* stale blocks receiving enough confirmations to be wrongfully accepted at the application level (Figure 2.6) [69].

3. Related Work

Existing research relevant to our work can be categorized into three areas. Firstly, we are interested in previous attempts of using DLT in edge computing, in order to reveal popular strategies, but also misconceptions or assumption made about blockchain in edge. Secondly, we investigate proposals of edge computing marketplaces and provisioning using both distributed and centralized auctioneers, with the goal of extracting common components and architectures. Finally, existing evaluations and especially simulations of DLT are relevant to us. Here, we hope to find preliminary results on blockchain performance and determine the research gap to be bridged in order to make emulation a viable strategy for evaluating blockchain in edge.

3.1. DLT in Edge Computing

The following aims to provide a brief summary and selection of common frameworks attempting to integrate blockchain and edge computing systems, especially focusing on performance critical applications. For a more complete study, refer to for example [6], [7], [89], [90]. Overall, the use cases of blockchain in edge can be categorized into network, storage and computation, under the overarching goal of improving security, privacy, consistency and trust within the applied systems.

Network

In *DistBlockNet* [91], a PoW blockchain is used as a trustless mediator between IoT nodes updating flow rules of their underlying *Software Defined Network* (SDN). Here, the authors claim to detect and respond to attacks on the IoT network in real time, however, the intended blockchain architecture remains largely unmentioned and is not included in performance evaluations. The authors of [92] propose a framework of *Security Manager* (SM) nodes providing rekeying and key transfers for transportation systems entering or leaving the SM's security domain. Therein, SMs construct a blockchain network by collecting and propagating requests in blocks. While the given performance evaluation seems promising, the simulated network was only composed of 9 nodes, leaving the scalability of this approach to be questioned. [93] suggests using three different blockchain networks (monitoring, provisioning and brokering) to quickly deliver video content to end users in the public web. To this end, authors suggest the use of a permissioned blockchain *Hyperledger Fabric* [75] previously criticized for its low scalability [94]. Additionally, their approach reveals the need for five consecutive write operations, and thus - blocks on the chain, before content is finally delivered. [95],

[96] provide a thorough review on the use cases of blockchain in 5G networks. Here, the prevalent challenges are related to the performance and scalability of blockchain platforms.

Storage

Liu et al. [97] use blockchain to ensure integrity and verification of data shared between IoT producers and consumers. Despite the blockchain only being hosted by a single node in their evaluation, the results reveal significant latencies of up to several minutes, making us question the applicability of such an approach in IoT. In *BeeKeeper* [10], the privacy of IoT devices is preserved by storing data on a trustless blockchain. On request, the data is then read by external servers, processed and written back to chain without identities being revealed. The authors identify that the performance of their approach relies on the underlying blockchain. Arguably, without a suitable architecture, such an approach is infeasible. Similarly, *BlockPro* [98] defines two Ethereum smart contracts acting as a pipeline to secure provenance and integrity of data sent between IoT devices and database servers. Evaluations are conducted on two private Ethereum miners, disregarding the overhead of scaling blockchain and Ethereum in particular.

Computation

In [99], the authors propose a two-stage Stackelberg game allowing mobile devices to offload expensive PoW computation to the cloud. As a result, mobile clients are able to participate in PoW blockchains under otherwise equal conditions, without experiencing heavy computational loads. In a different approach, *Mneme* [100] defines two new consensus protocols, proof-of-concept and proof-of-equivalence, to deploy blockchains on mobile devices. Evaluation results indicate that, similar to other consensus protocols, a trade-off between performance and security has to be made. [101] proposes a *health blockchain* for recording and processing data of patients' implanted or wearable sensors. Once again, no performance analysis or description of the blockchain architecture is made. [102]–[104] and others propose the use of Ethereum smart contracts to create marketplaces for trading electricity in smart grids. As identified in Section 2.1.2, these systems could potentially benefit from latencies lower than what is offered by Ethereum. The authors of [105] propose using the blockchain black-box for trustless mediation of a crowd-intelligence ecosystem on mobile edge computing. As a result, applications of such a platform are additionally limited by the underlying blockchain architecture.

3.2. Edge Computing Marketplaces and Provisioning

Enabling crowdsourced edge marketplaces for decentralized computations has remained a topic of interest within the research community for several years. In[15], the authors propose an edge computing marketplace on top of a DPoS blockchain supported by a range of simulation results. However, results reveal that latency overheads imposed by

the enabling blockchain architecture are quite high - delivering an unsatisfying performance for edge marketplaces. DeCloud [16] defines a truthful double auction protocol and architecture designed for edge computing marketplaces reliant on blockchain. Here, bids containing resource offers and requests are first published in encrypted form. After all bids for the next auction are collected, a block header is published, prompting clients to reveal their encryption keys. Subsequently, a match is computed and the final block is added to chain. Nevertheless, especially due to performance constraints, the underlying DLT remains an open research question [19]. [106]–[108] and [109] define smart contracts enabling compute marketplaces and cloud tenant management based on Ethereum [11]. However, potential overheads resulting from an Ethereum network are not included in these studies. In BlockEdge [110], multiple blockchains are superimposed into a hierarchical structure (edge, fog and global blockchains) in order to adapt to varying network requirements, but the blockchain architectures themselves are not investigated. Focusing on a different part of the marketplace architecture, the authors of [111] propose a model to predict the probability of failure-free execution and result of decentralized resource providers, using log traces of an existing blockchain-based cloud market (iExec [112]). Sharing many research areas with compute marketplaces, [113] suggests a system for blockchain-based multi-party computation on EOS [84]. However, overheads of using the EOS platform are not investigated. In [114], IoT tasks to be executed by cloud servers are submitted on an Ethereum blockchain. Here, task submission could take up a significant portion of total provisioning latency. Conversely, in [17], [32], [115] and [116] the authors suggest more centralized solutions by making use of trusted servers as opposed to DLT. To this end, [32] proposes the use of a cloud server serving as the initial rendez-vous point, service discovery and computational fallback of edge providers. In total, many approaches suggest blockchain as a decentralized auctioneer for crowdsourced compute marketplaces. However, barely any practical solutions are provided. Additionally, existing evaluations are oftentimes lacking in the area of the underlying DLT, ignoring potential bottlenecks imposed by such an architecture.

Outside of the research community, several platforms offering the exchange of computing tasks and resources have been implemented. These consist of most notably, iExec [112], EDGE [51], Dfinity [53] and SONM [52], all of which employ blockchain and crypto tokens in parts of their operation. In EDGE, highly available master nodes maintain a PoS blockchain serving as a domain name system for gateways. Gateways in turn are used as network entry points, keeping job queues and assigning tasks to crowdsourced edge hosts. Despite only offering storage capabilities so far, support for computing tasks is planned [51]. In iExec, computing providers join working pools under a pool manager. Here, iExec describes a customizable proof-of-contribution protocol, to certify the likelihood of a result being valid, by replicating tasks on multiple workers and comparing digests of result folders. Deals between pool managers and requesters are sealed using an Ethereum on-chain component. SONM follows a similar approach by handling payments on the public Ethereum chain. Smart contracts relevant to the marketplace are executed on a private Ethereum network, creating blocks every 2

to 3 seconds. In contrast, Dfinity distances itself from traditional blockchain solutions, aiming to create the *Internet Computer* using a *Network Nervous System* as a tailored network abstraction. To summarize industrial approaches, interactions with the blockchains appear to be particularly prevalent during auctioning and sealing of deals. Especially for longer running services, or applications with relaxed latency requirements, the resulting initial overhead is manageable. However, in order to support edge computing use cases as claimed by these platforms, offering service hand-offs or FaaS by provisioning resources with negligible latency, seems to be a problem remaining to be solved.

3.3. DLT Evaluation

Concluding our results of the previous sections, we are highly interested in finding a blockchain architecture capable of supporting low latency edge computing use cases. In particular, we aim to determine how such a platform looks like, or if it even exists at all. To this end, we investigate a variety of simulators and models evaluating blockchains. A general summary over the field concerned with modeling blockchain architectures is provided by [117]. Here, we realize that existing work in this space appears to be largely unsuitable for our study due to several reasons.

BLOCKBENCH [94] describes a framework for evaluating the performance of private blockchains. In their work, the authors investigate throughput and latency of Hyperledger Fabric [75] and permissioned adaptations of Ethereum [11]. Their findings determine permissioned blockchains to be significantly slower than current database systems when comparing their capability of handling traditional data processing workloads. Although these results represent valuable insights to us, *BLOCKBENCH* itself is not suited for emulating detailed edge network topologies or permissionless blockchains. Following a similar approach to us, *SimBlock* [118] attempts to investigate ways of improving the performance of proof-based blockchains using simulation. However, the implementation does not consider other consensus protocols nor the simulation of transactions. Similarly, [119]–[122] or [123] only support proof-based blockchains and are not fine-grained enough for our purpose. As one of the few evaluations focusing on latency, [124] provides interesting results regarding the high confirmation latency of Ethereum. However, the simulation only considers parameters describing inter block time, number of confirmations and average propagation delay. Finally, [125] provides relevant comparative results on the creation of stale blocks in proof-based blockchains depending on block rate and network latency, but doesn't allow for any latency evaluation of transactions.

In contrast, DLT evaluations not using simulation are more insightful to us. [19] identifies potential shortcomings and requirements of DLT in edge. Next to low latency, the authors mention the need for privacy, accountability, juristic legislation and low cost. [126] compares the performance of blockchain platforms Hyperledger Sawtooth [66], Ethereum [11] and EOS [84]. Here, EOS appears to be the most promising in terms of CPU and RAM usage, as well as transactional performance. In [80], the authors compare

the scalability and performance of several permissioned consensus protocols, namely Tendermint [79], Paxos, PBFT and HotStuff [127], revealing severe bottlenecks at higher node counts.

In this work, we primarily focus on the latency of blockchain transactions in edge environments. However, for many applications, *transaction throughput* represents an equally important parameter. *Rainblock* [128] describes a performance bottleneck resulting from verifying many transactions per block, and proposes a solution improving underlying data structures and I/O operations of Ethereum [11]. As a result, the authors achieve throughputs of 20k transactions per second at the cost of higher latency.

3.4. Contribution

In summary, many works are tempted by the promises of DLT. As a result, strong assumptions are made, by treating blockchain as a black box solution to privacy, security or trust related issues, and no mind is paid towards potential bottlenecks or implications resulting from such an architecture. Furthermore, evaluations and implementations of proposed blockchain frameworks in edge are incomplete or not applicable for deployments at scale. While existing evaluations specifically targeting DLT give strong hints on trade-offs and limitations of blockchain in edge, previously implemented simulators are too abstract and unoptimized for the edge computing use case. However, as we do not blatantly disagree with the promises made by DLT towards a crowdsourced edge, we recognize the need for thorough investigations. Consequently, we are motivated in our contribution of evaluating blockchain performance in edge. As the optimal middleground offering better maintainability and measurability compared to evaluating real blockchain code (i.e. Hyperledger Fabric [75], Ethereum [11]) all while being more realistic and fine-grained than abstract simulations, in this work we opt for an emulation-based approach.

To this end, we develop a generic, scalable and message-based blockchain emulator (see Chapter 5) allowing the configuration and observation of several internal parameters with special influence on performance. Due to the low granularity of our approach using real world socket connections, we additionally aim to enable the potential of evaluating *hardware-in-the-loop*, by implementing a *n-to-m* mapping of emulated peers to coordinating machines. Because of the poor performance of BFT-based blockchains at scale [80], [94], we focus on implementing a generic proof-based approach, and DPoS-BFT as presented in Section 2.3. Subsequently, we will apply our emulator to evaluate performance boundaries of blockchains supporting a crowdsourced edge computing marketplace in regards to latency requirements imposed by the edge environment, and propose 4 empirical models summarizing our results.

4. Edge Computing Marketplace

In the following, we intend to investigate the internals of a crowdsourced edge computing marketplace backed by blockchain. To identify potential dependencies between both paradigms, we introduce vital components of such an architecture, as well as their interactions with the underlying DLT. Based on our results, we will deduce a strategy to evaluate the performance and latency boundaries imposed by blockchain in Chapter 6.

Figure 4.1 depicts a typical architecture of a decentralized edge resource marketplace. Such a marketplace consists of three participating entities, *resource providers* that offer their heterogeneous hardware for hosting applications (typically for an asking price), *clients* or *developers* that are looking for resources at the edge to deploy their next-generation applications, and a *blockchain* acting as a decentralized auctioneer - matching requests from clients to offers from providers. Rather than focusing on transactions and interactions between clients and providers, in this work we dissect the internals of the blockchain operations that glue the marketplace together. To furthermore highlight direct communication with the blockchain, we distinguish between off-chain and on-chain components of the marketplace. Additionally, we differentiate between devices that are required to be part of a low latency edge network (i.e. edge resource providers) and participants who may or may not be part of the abstract edge (i.e. clients and blockchain nodes) as indicated by Figure 4.1. In contrast to previous proposals of edge computing marketplaces, our approach and architecture attempts to minimize the potential of DLT becoming a bottleneck, by limiting client interactions with the blockchain to a single transaction before matching results are received.

Based on prior research in the field [15]–[17], [108], an edge marketplace relies on the blockchain to support the following operations.

4.1. Auction Protocol

The auction protocol enables bidders to participate in the matching by defining a communication channel with the auctioneer. Bids define valuation, costs and properties of the requested and offered resources, information about the submitted task and the provided hardware. To this end, requirements such as truthfulness, privacy and the message complexity of the protocol need to be considered. Using blockchain as a distributed, trustless auctioneer to record and process resource bids, promises irreversibility and decentral fairness of the resulting matches. As is typical for blockchain networks, created blocks, and thus auction results, are verified by all peers using a consensus protocol. This way, the decentralized blockchain auctioneer ensures that no bid was excluded

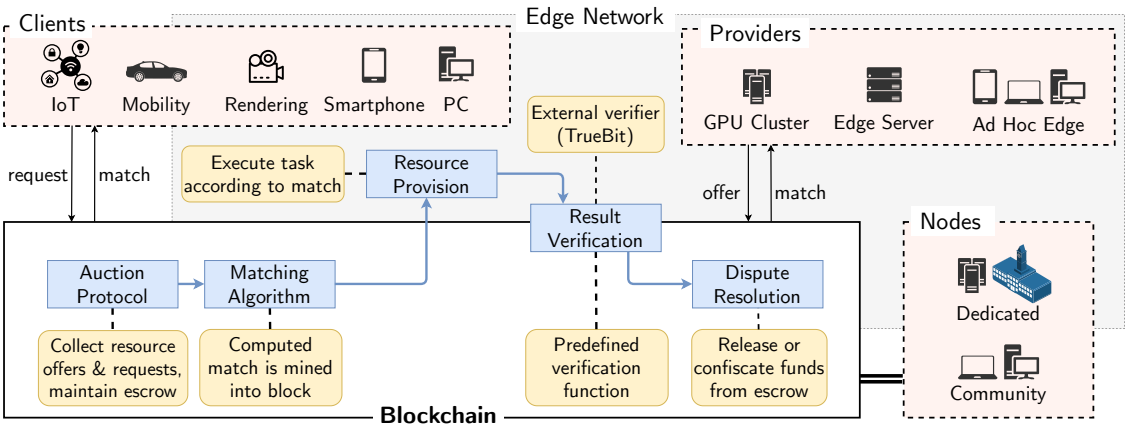


Figure 4.1.: Devices, roles and activities in a crowdsourced edge computing marketplace using blockchain.

or altered by a malicious peer, and the auction result is indeed optimal. In order to minimize latency of the blockchain auction in an edge environment, participants should be able to send their bid as a single transaction and be expected to receive a match as part of the next block containing their transaction. Several auction protocols for blockchains have been proposed in the past - most of which, however, require multiple blocks until completion to accomodate for security or privacy requirements. *Strain* [129] describes a secure auction protocol for blockchains, guaranteeing non-retractability and confidentiality of bids against fully-malicious adversaries. Yet, on top of expensive cryptographic operations taking up several seconds according to the authors' measurements, the protocol additionally requires 4 consecutive blocks on the chain. Similarly, *DeCloud* [16] sacrifices latency in favor of guaranteeing the truthfulness of bids. Here, encrypted bids are published as part of the first block. In a second block, the encryption keys are revealed and a match is computed[1]. The third and final block is then composed of the clients' confirmations. In *SmartEdge* [107], computation providers deploy smart contracts for each of their resources. Clients then send requests to the contract which are mined into a first block. Requests are subsequently accepted or rejected by the provider in the second block. [106] mirror this process by allowing clients to publish computation tasks in a first block which are then selected by *farmers* as part of a second block.

To summarize, sending resource offers and requests to the blockchain in a single transaction, as proposed by our approach, potentially requires sacrifices in truthfulness and privacy due to i.a. abandoning the encryption step of bids as described in [16]. Additionally, we discover the need for an escrow account to be filled with funds by clients and providers before participating in any auctions. This way, payments and

[1]Note that the latency of the second block is additionally bounded by the slowest client participating in the auction. The match cannot be computed before all previously encrypted bids are revealed. Verifying consensus nodes cannot distinguish between a slow client revealing the bid too late and a malicious auction node excluding the client's bid on purpose.

penalties can be handled asynchronously without interfering with the auction from a performance perspective.

4.2. Matching Algorithm

After receiving the service bids from clients and resource offers from providers, during block creation, the matching algorithm finds optimal pairings with the goal of maximizing economic, QoS and runtime performance. Here, under the assumption of bids being hidden from other participants due to encryption, DeCloud [16] defines a welfare optimizing auction and matching mechanism. Similarly, albeit not using blockchain auctioneers, EdgeMap [17] relies on Vickrey-English-Dutch auctions [130] to maximize QoS. In general, depending on the number of participants, complexity of bids and QoS goal, calculating and verifying an optimal match represents a computationally extensive or even NP-complete task [131], [132]. Similar to PoW, a proof-based consensus protocol denoted as *Proof-of-Match* (PoM) could be envisioned, which uses the calculation of optimal matches to replace hashing of random numbers.

Unrelated to the underlying consensus protocol, our approach attempts to reduce the latency of participants waiting for the auction result by creating a new match for every new block on the blockchain and vice versa, resulting in exactly one auction per block on the chain. Consequently, the waiting time of the matching algorithm gathering bids and the underlying block rate are closely interlinked. Similarly, the block size determines the maximum number of bids to be considered for each match. Choosing a suitable block and auction rate is therefore a trade-off between latency and quality of the match. While higher block rates lead to faster auction results, a lower block rate implies waiting for more bids which naturally leads to a better matching result. Contrary to other approaches, we additionally eliminate the bidders' responsibility of confirming a computed match. Instead, we consider a rejection of the match to be an unfulfillment of the SLA made when participating in the auction, equal to delivering erroneous results.

4.3. Resource Provision

To minimize interactions with the blockchain, resource provisioning is handled off chain once both parties receive the computed match. Locations and addresses of tasks and resources that were originally included in bids are now published to both parties as part of the match. Here, distributed storage technology such as the peer-to-peer enabled *InterPlanetary file-system* (IPFS) [133], which has been successfully integrated to blockchain and IoT platforms in the past [134], [135], appears suitable for uploading results or task containers in case the storage of individual resource providers is limited. Due to the prevalence of poorly managed commodity hardware in a crowdsourced edge, longer running services are expected to experience crashes. Additionally, when offering FaaS execution of smaller tasks, resource provision potentially takes up a significant part of the total result latency. Therefore, minimizing provisioning times is critical

for short-lived lambda functions and for reducing overheads resulting from service hand-offs. To this end, technology such as *JITSU* [136] enables just-in-time summoning of Unikernel VMs in 100-150 ms.

4.4. Result Verification and Reputation

This marketplace component is responsible for verifying task results and failure-free execution as per QoS requirements in the bid. Here, we distinguish between different strategies, i.a. log verification, correctness checking and redundant computation. In *TrueBit* [137], external entities are used to move expensive verification or redundancy off-chain. Verifiers are incentivized to replicate or verify tasks by monetary rewards issued for discovering invalid results. To incentivize verifiers even in the absence of invalid results, occasional errors are introduced artificially by the auctioning process conducted on blockchain nodes. [111] proposes a model to predict the probability of failure-free execution and result of decentralized resource providers using log traces of previous tasks. Consequently, the determined probability can be reflected in a reputation score assigned to each resource provider. During bidding, clients may then specify a parameter defining the required minimum historical reliability and reputation. *BC-MDC* [15] proposes the use of predefined verification functions, possibly to be included in the client's bid. After a result is calculated, the client determines its correctness by applying its verification function. In case of invalid results, the same function is executed by a blockchain consensus node as part of a smart contract, to enable asynchronous dispute resolution. This approach essentially represents the idea of *verifiable computing* [138]. Similarly, in *Pinocchio* [139], clients are able to verify computational results based solely on cryptographic assumptions, which are defined by creating a public evaluation key describing the computation task. In order to enable task replication, the algorithm chosen in Section 4.2 may output multiple resources for a single task, depending on a requested redundancy parameter. The correct result may then be determined using majority vote, similar to the approach presented by iExec [112].

4.5. Dispute Resolution and Payment

In order to prevent free-riding, both parties transfer security deposits, including the client's payment, into escrow before partaking in the auction [15]. If a bidder's escrow account does not contain enough funds to compensate or pay their match in case of an SLA breach, they are excluded from the auction before a match is calculated. After resource provision, in case the result verification indicates a dispute, the resolution process is initiated asynchronously as a smart contract by the aggrieved party. Consequently, monetary or reputational punishments and compensations can be issued by diverting funds of both parties from escrow. By handling dispute resolution and payments asynchronously using security deposits, clients are able to access their results without having to wait until the provider is paid.

5. Emulator

The goal of *emulation* is to narrow the credibility gap of measurable differences between a simulated model and the real system by bringing the model closer to reality [140]. In our case, an emulation-based approach promises to be more configurable, maintainable and measurable compared to a real system, while being more granular than theoretical models or abstract simulations. Our goal in this chapter is therefore, firstly, to implement an extendable emulation of general peer-to-peer (P2P) networks. To this end, we will make use of socket connections and message serialization as found in real-world communication, in order to resemble natural phenomenons such as network congestion, latency or bandwidth as close as possible. At the same time however, the emulation should be scalable both vertically (i.e. emulating larger networks on a test bed of few capable machines without sacrificing realism) and horizontally (i.e. measuring performance on top of real networking hardware in a controlled environment). Moreover, the emulator's networking plane should be highly configurable and allow for custom topologies and future implementations while maintaining a high usability.

To enable the evaluation of DLT in edge we will then, secondly, extend our networking implementation with blockchain functionality. After defining a generic and pluggable blockchain abstraction layer, we will focus on implementing a general proof-based approach and DPoS-BFT as presented in Chapter 2. Here, the emulation aims to provide a snapshot of a static P2P network, replicating a shared blockchain by propagating blocks and transactions according to a defined message and consensus protocol. While remaining consistent with real-world implementations of DLT in terms of message and protocol definitions, we introduce abstraction especially in regards to cryptographic details. Specifically, we generalize the process of creating transactions or blocks and disregard encryption, authentication or signatures as commonly found in real-world counter-parts. Instead, we opt to implement artificial and configurable delays to emulate the operation of specific blockchain platforms, i.e. the verification of auction matches or the creation of blocks or transactions at configurable rates. Due to their in-applicability to our use case and low scalability as identified in Chapter 3, for now, we leave the implementation of DAG and BFT-based DLT to future work. Consequently, the emulator should allow to easily plug in future blockchain technologies with minimal overhead.

Lastly, the emulator should be measurable and repeatable especially in terms of its performance metrics. As a result, we implement latency and throughput measurements of confirmations, transactions and blocks as an integral part of the emulation. In order for our results to be repeatable, we define limitations of our approach and validate our emulator against measurements of real-world blockchains used in cryptocurrencies.

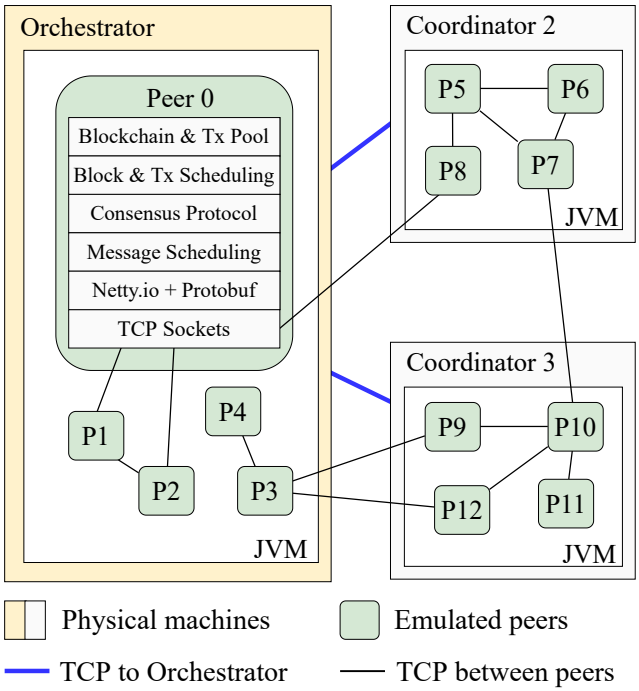


Figure 5.1.: Emulator deployment and technology stack.

5.1. Architecture

The emulator [141] is developed using `Java` and enables a *n-to-m* mapping of emulated *peers* to managing *coordinators*. Typically, each coordinator is hosted on a separate machine or VM and connected to other coordinators in order to compose a network of peers. Figure 5.1 provides an overview of the emulator's technology stack and an exemplary deployment of 13 peers to 3 coordinator machines or VMs, of which coordinator 1 assumed the *orchestrator* position. All connections between coordinators and peers are based on TCP sockets using *Netty* [142] as an asynchronous networking framework. Messages are defined using *Protobuf* [143] and strictly divided into messages sent between coordinators and messages sent between peers exclusively (Appendix A). Coordinators are responsible for instantiating, starting, connecting and stopping peers assigned to them by the orchestrator. During an emulation run, peers communicate with each other as defined by their instantiated implementation at runtime (i.e. a blockchain protocol).

5.1.1. Peer-to-Peer Network

Before a new run can be started, the orchestrator calculates a network graph defining the underlying topology of the configured P2P emulation. Using the *strategy* pattern, network topologies are easily defined and added by implementing and instantiating the `GraphStrategy` interface as shown in Figure 5.2. The interface defines methods returning

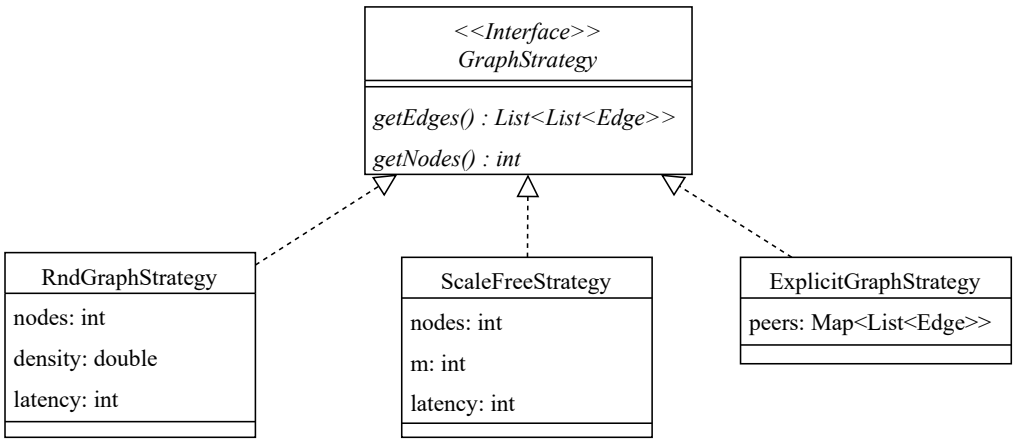


Figure 5.2.: Graph and network strategy class diagram.

the number of nodes in the network (n) and an adjacency list defining edges and latencies between nodes. The defined graph is required to be undirected and connected. During the emulation, each unique edge results in a new TCP connection between peers with added artificial latency as specified by the edge. Peers are enumerated from 0 to n referring to their index within the adjacency list. At the time of writing, three graph strategies are implemented.

Random Graph Strategy

This strategy generates a fully random graph with a given number of nodes and edges. To reduce the scale of a parameter defining the edges, a graph density parameter is introduced to calculate the number of edges. In undirected graphs of n nodes, the network density d describes the ratio of existing edges e to the maximum possible number of edges as defined in Equation (5.1). As part of the random graph strategy, networks are created by firstly generating a random spanning tree to ensure connectivity. Subsequently $e - (n-1)$ unique edges are picked and added at random from a set containing all remaining edges. As a result, the true number of edges e_t is calculated as shown in Equation (5.2).

$$d = \frac{2e}{n(n-1)} \tag{5.1}$$

$$e_t = max\left(\frac{dn(n-1)}{2}, n-1\right), \quad d \in [0,1] \tag{5.2}$$

Scale-Free Strategy

A *scale-free network* describes a graph following a power-law degree distribution, resulting in few nodes known as *hubs* with unusually high degree compared to the remaining nodes of the network. Many networks, i.a. the Ethereum network [144], are thought

to fulfill scale-free properties. Consequently, compared to fully random graphs, this strategy generates a different and potentially more realistic network topology, while still being randomized and easily configurable. As the underlying generation mechanism we choose to implement the Barabási-Albert model [145]. Starting with an initially connected component, new nodes are added iteratively to m previously included nodes until all n nodes are part of the network. To generate the aforementioned power-law distribution, when selecting m nodes to connect a new node to, node i of degree k_i is chosen with probability,

$$p_i = \frac{k_i}{\sum_j k_j}. \tag{5.3}$$

Explicit Graph Strategy

In order to be compatible with other topology generators or custom network topologies without recompilation, this strategy enables an explicit definition of a connected graph by constructing a file containing a list of edges.

Listing 5.1: Defining a network topology file using a list of edges.

```
<from_id_string> <to_id_string> <latency_in_microseconds>
<from_id_string> <to_id_string> <latency_in_microseconds>
...
```

5.1.2. Emulator Components

Throughout our implementation, we pay special attention towards the extendability, modularity and maintainability of our approach. As a result, we make use of inheritance to extract common functionality shared between components and offer easy to use interfaces to enable future extensions. Figure 5.3 presents a class diagram of the emulator architecture. Overall, our approach can be divided into three major abstract components described as *coordinator*, *peer* and the *blockchain*. While coordinators overlook the execution of emulation runs by configuring peers and their network, peers themselves share transactions and blocks to collectively extend the replicated blockchain. Concrete implementations and configurations of these components, i.e. consensus protocols, fork resolutions strategies or other parameters are then chosen at runtime according to the current emulation run. As a cross-cutting concern, we aim to encompass functionality and parameters of real world blockchain networks as closely as possible and base our implementation on literature as introduced in Section 2.3 and Chapter 3.

Coordinator

At the beginning of each emulation run, one coordinator is selected as the *orchestrator* of all coordinators. To this end, the orchestrator generates the configured network graph as described in Section 5.1.1. Subsequently, the orchestrator distributes the total number

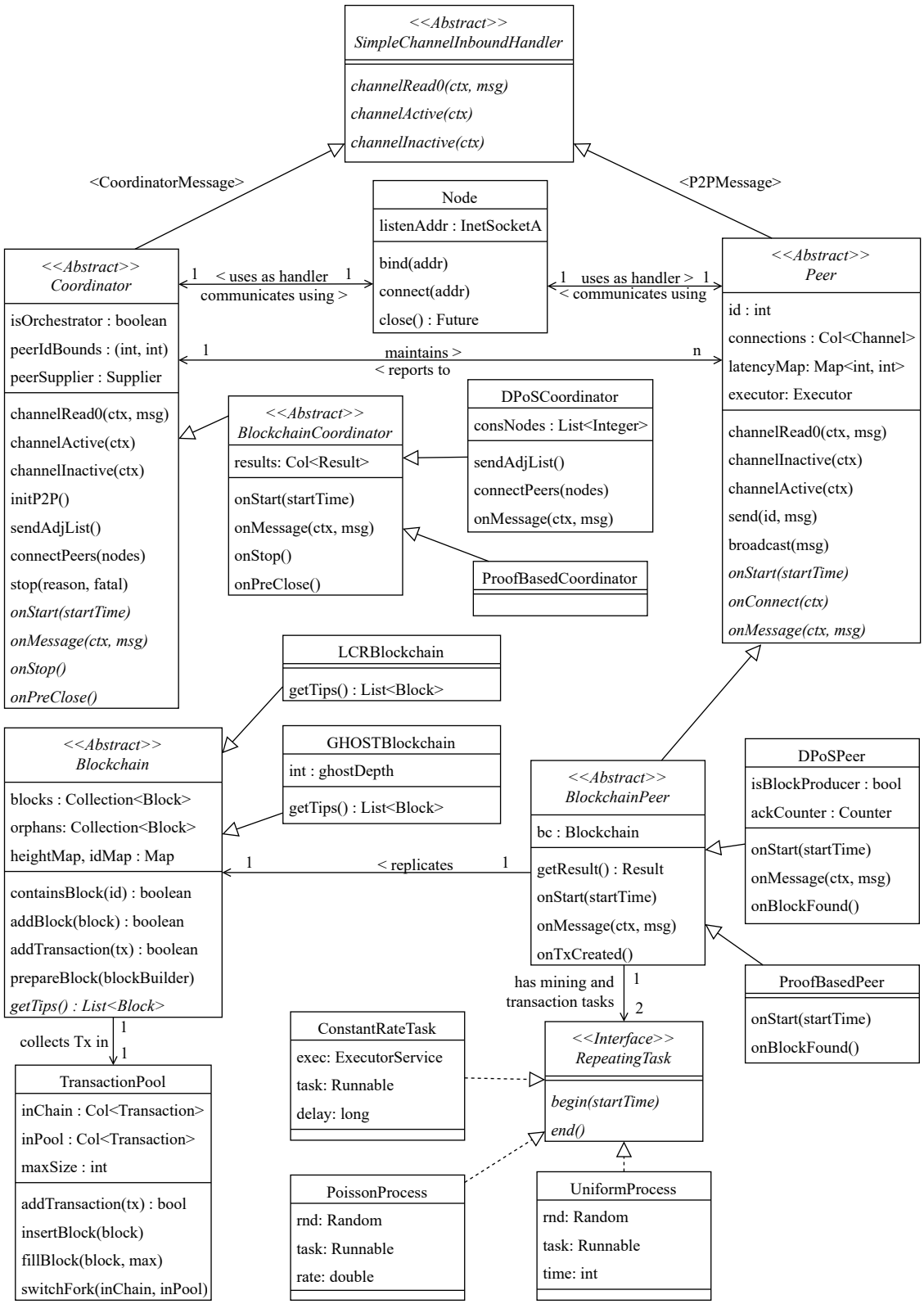


Figure 5.3.: Emulator class diagram.

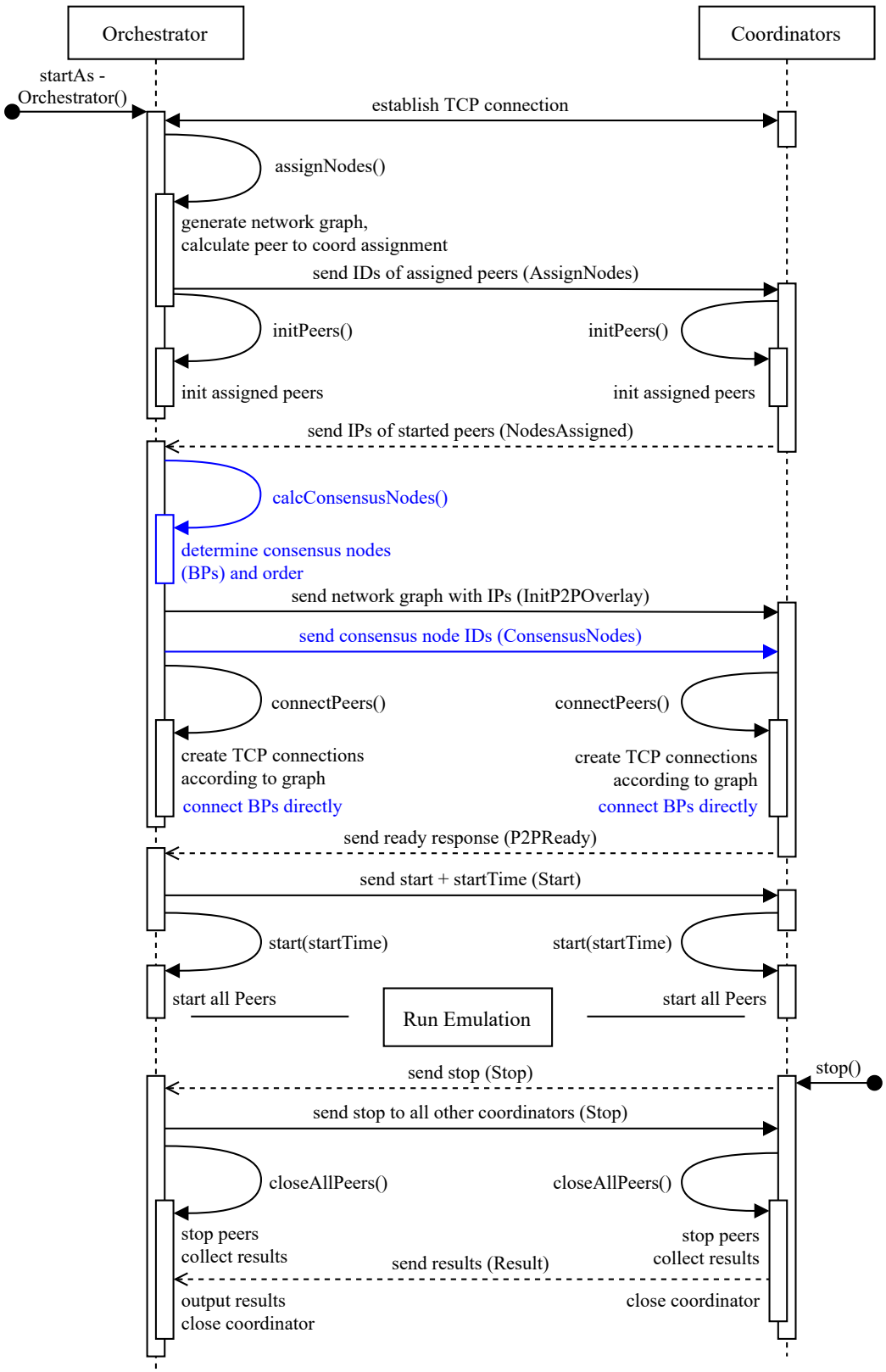


Figure 5.4.: PoX (black) and DPoS-BFT (black and blue) coordinator sequence diagrams.

of peers to be emulated between all coordinators by establishing TCP connections and initiates the creation of a P2P network. As soon as all connections between peers are established, coordinators start the emulation simultaneously. To allow for custom P2P emulations, coordinators expose abstract methods defining operations to be executed once the emulation is started and stopped (Figure 5.3). In case of a blockchain emulation, on *start*, peers begin creating transactions at the same time. Once the emulation is stopped, the *blockchain coordinator* defines the specific blockchain related results to be collected from peers. To allow for additional operations before a blockchain emulation is started, i.e. the selection of specific block producers in DPoS-BFT, the blockchain coordinator is extended further by overriding methods of the coordination protocol.

Figure 5.4 shows the communication between coordinators during a blockchain emulation as a sequence diagram. After being assigned a specific amount of peers to host, coordinators reply with the IP addresses of their instantiated peers. Subsequently, the orchestrator sends out the created network graph along with peers' IP addresses to enable the inter-connection between remote peers. Once all peers are connected according to the graph, the coordinators send a ready response, upon which the emulation start is initiated. During or after an emulation, the current run can be terminated by any coordinator sending a stop message to the orchestrator. The stop command is then propagated to all remaining coordinators. Finally, results are collected and stored by the orchestrator. The message protocol between coordinators is defined using *Protobuf* [143] as described in Appendix A.

Peer

The abstract *peer* component offers methods to connect to other peers, as well as sending and receiving TCP messages. To enable the emulation of large networks on restricted hardware, peers maintain connections to other peers by additionally storing latencies as configured by the underlying network graph. Before a message is sent to a remote peer, the latency is added artificially using scheduling. Similar to coordinators, peers provide an entry point for concrete implementations by defining overridable messages called, i.e. once the emulation is started, or whenever a P2P message is received (Figure 5.3). In case of a blockchain emulation, transactions are created by all peers periodically once the emulation run is started. A *blockchain peer* additionally defines block propagation protocols such as advertisement-based gossiping or unsolicited block push (Figure 2.5) and maintains a local blockchain replica. Furthermore, measurements concerning latency or throughput of blocks and transactions are made and reported to the coordinator on termination. The blockchain peer is further extended to incorporate block consensus protocols evaluated in our work. To this end, *proof-based peers* generate blocks randomly and periodically according to a Poisson process inspired by our model of proof-based blockchains in Section 2.3.1. Therefore, each peer i is configured with an individual *mining share* which precisely represents parameter w_i of Equation (2.1). As a result, Poisson processes are instantiated with rate λ_i as defined in Equation (2.2). When emulating DPoS-BFT, a *DPoS peer* only creates new blocks if it was determined to be a

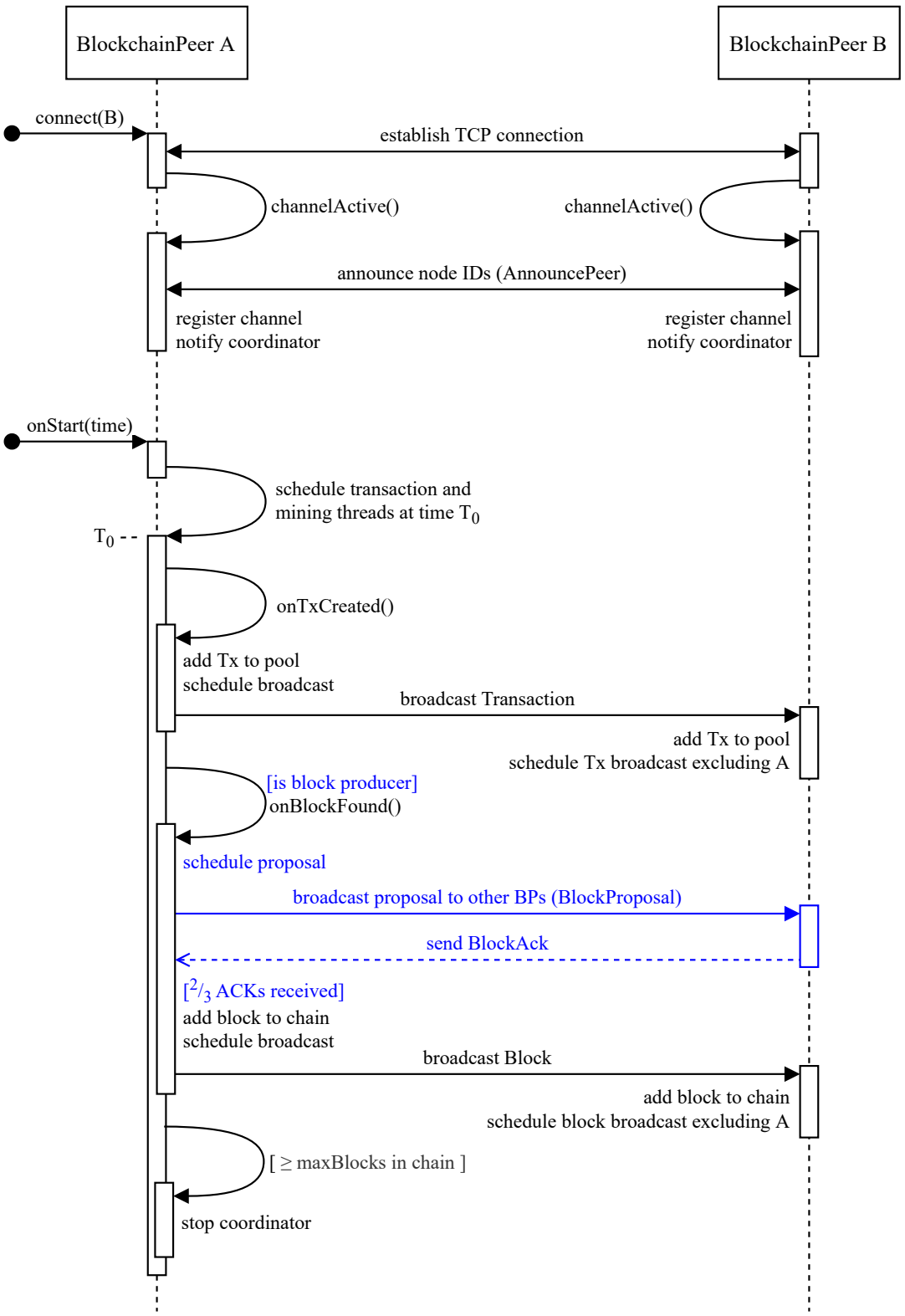


Figure 5.5.: PoX (black) and DPoS-BFT (black and blue) peer sequence diagrams. Messages in parentheses refer to definitions of Appendix A.

block producer by its coordinator. In that case, blocks are created at configurable batches in round-robin fashion as described in Section 2.3.3.

Figure 5.5 represents a sequence diagram of the communication between two blockchain peers employing a proof-based or DPoS-BFT consensus protocol depending on their concrete implementation. When establishing a connection, peers exchange their IDs in order to correctly assign latencies of future messages as defined by the network graph. After being instructed to start at a given time, blockchain peers begin to create and propagate blocks of transactions. Each peer broadcasts new and received blocks to all known peers using a block propagation protocol (here, unsolicited block push) and applies individual latencies as defined by the network graph and bandwidth. Once a maximum number of total blocks is reached, the hosting coordinator is informed to stop the current emulation run.

Blockchain

Blocks and transactions are stored in the blockchain component of each peer. The blockchain maintains multiple indices used to access blocks at specific heights of the tree or by a block's unique id. Additionally, the blockchain offers registration of *transaction listeners* to be executed once a transaction is confirmed by the configured number of blocks, in order to allow for measurements of confirmation latency. Each blockchain further consists of a *transaction pool* responsible for tracking transactions that are currently part of the main chain, in a fork or not yet included in a block. To allow for different fork resolution strategies, the blockchain framework exposes an abstract method returning all *tips* of the current chain and its forks. Here, *tips* are defined as one or more blocks without children, eligible to be selected as the parent of a new block. As defined in Section 2.3.1 and implemented in their respective sub-classes, *LCR blockchains* select tips of the longest forks, while *GHOST blockchains* choose those that are part of the heaviest sub-tree.

Figure 5.6 describes the process of adding a block to the chain. In case no parent of the new block is found, it is instead stored separately as an *orphaned block* until the respective parent is received. If the addition of the new block leads to a switch from the previous main chain to one of its forks, the transaction pool is updated to reflect the new history of transactions.

5.2. Configuration and Execution

An emulation experiment is defined by constructing a configuration file in `YAML` format and sharing it between coordinators. To enable communication of coordinators, configuration files start with a coordinator definition section, listing addresses and computing shares of coordinators participating in the next emulation run (Listing 5.3). Here, the first list entry represents the orchestrator, that initially establishes connections to all other coordinators. The computing share parameter refers to the share of total peers

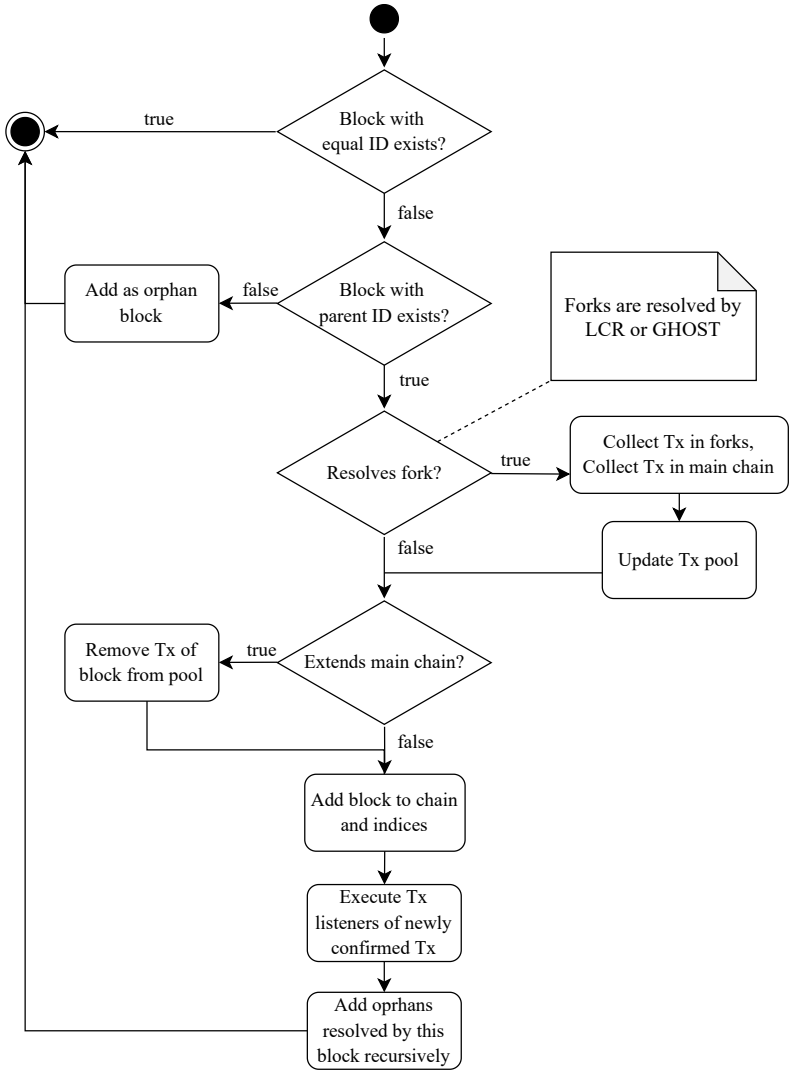


Figure 5.6.: Process of adding a block to chain.

to be hosted on each coordinator. Typically, this is achieved by specifying the number of cores on each coordinator or by using the exact number of peers to be emulated. Individual coordinators are started on their host machines or VMs by executing the compiled emulator program along with IP and port to be used for hosting the coordinator (Listing 5.2). Consequently, for each coordinator definition entry, one coordinator is started. The third argument contains a path to the (shared) folder of configuration files. After all coordinators are started, configuration files are executed in lexicographic order. Coordinators terminate automatically after the execution of all emulation runs.

Listing 5.2: Starting a coordinator.

```
$java –jar bcem.jar <ip> <port> path/to/config/folder
```

Listing 5.3: Coordinator definition.

```
coordinators:
– address: 192.168.0.124:5151
  computingShare: 40
– address: localhost:5152
  computingShare: 47
– address: vm.example.com:5153
  computingShare: 39
```

Table 5.1 gives an overview of parameters exposed by the emulator configuration. We divide between general blockchain and networking parameters, and parameters relating to our implemented consensus protocols, DPoS-BFT and PoX (proof-based). Parameters listed here define the configuration of all peers equally by i.e. specifying a mathematical distribution of lottery power between peers in PoX (P_D). Additionally, peers can be configured individually to override global settings and define specific behavior.

The following explains a selection of parameters in more detail. T_F defines a list of transaction fees and the share of their occurrence at each peer. When creating a new block, transactions with the highest fee are selected first to maximize the miner's reward. Along with T_R, T_{PD} defines the amount of transactions created at individual peers. While the total rate of transactions is determined by T_R, the distribution determines whether some peers generate more transactions than others. T_{TD} specifies the time distribution of transactions being created at each individual peer. While following a static mean derived from the global T_R and T_{PD} parameters, transactions may be created at a constant, uniform or Poisson distributed rate. T_{PS} sets the maximum number of transactions to reside in the memory pool without being included in the chain. Further transactions are ignored by the peer. The block verification time V_T is applied artificially when receiving a remote block before adding it to the local chain and propagating further. In case the fork resolution strategy B_{FR} results in a tie, it is broken either randomly or on first-come-first-serve basis as defined by B_{TR}. B_P refers to block propagation mechanisms introduced in Section 2.3.1. In case parameter B_{FB} is set to *true*, additional latency is added to block propagation according to bandwidth as if all blocks were full. D_K attempts to mitigate the creation of stale blocks in DPoS-BFT as shown in Figure 2.11. By refraining from creating the last blocks in each round, thus increasing the time between hand-offs, the number of stale-blocks is expected to decrease. With a similar goal, D_R defines whether the order of BPs should be chosen randomly or by solving the traveling-salesman problem to minimize hand-off times. In case of very large emulations of high block rates and many peers, results may be inaccurate while some mining and transaction threads are still initializing during start-up. S_B allows those first blocks to be skipped in measurements. After the orchestrator is started, the first emulation run starts automatically after N_D ms. Therefore, starting the orchestrator last is recommended.

Results of each emulation run are stored on the orchestrator's host in `JSON` format. Table 5.2 summarizes measurements relevant to this work.

Table 5.1.: Supported emulation parameters and default values.

Par.	Description	p. val.	default
B_R	Blocks generated by entire network per second	$float$	22
B_S	Maximum block size (transactions)	$[0, 2^{16}]$	5000
T_R	Transactions generated by network per second	$float$	10
T_S	Transaction payload size (byte)	$[0, 2^{16}]$	100
T_F	Transaction fee distribution	$list$	$[(1, 1)]$
T_{PD}	Transaction peer distribution (exp., const., uni.)	$enum$	$constant$
T_{TD}	Transaction time distribution (pois., const., uni.)	$enum$	$poisson$
T_{PS}	Transaction pool size (transactions)	$[0, 2^{31}]$	1000
V_T	Block verification time (μs)	$[0, 2^{31}]$	100
C	Number of required confirmation blocks	$[1, 2^{31}]$	1
B_{FR}	Fork resolution (LCR, GHOST)	$enum$	LCR
B_{TR}	Tie resolution of forks (random, first)	$enum$	$first$
B	Number of blocks to be created in this run	$[0, 2^{31}]$	5000
B_P	Use unsolicited block push	$bool$	$true$
B_{FB}	Always emulate full blocks	$bool$	$false$
B_C	Blockchain consensus (PoX, DPoS)	$enum$	PoX
Network Parameters			
N_N	Number of nodes	$[1, 2^{31}]$	500
N_B	Bandwidth (MB/s)	$[1, 2^{31}]$	1000
N_L	Mean propagation delay between peers (μs)	$[0, 2^{31}]$	$63k$
N_T	Network topology (random, scale-free, explicit)	$enum$	$scalefree$
PoX Parameters			
P_D	Peer distribution of lottery power w_i (exp., const., uni.)	$enum$	$constant$
DPoS-BFT Parameters			
D_N	Number of block producers	$[0, N_N]$	21
D_S	BP selection strategy (best, random, worst)	$enum$	$best$
D_B	Blocks created per BP each round	$[1, 2^{31}]$	50
D_K	Final blocks to be skipped per BP	$[0, D_B]$	0
D_R	Random BP order (random, best)	$bool$	$false$
Miscellaneous			
R	Number of emulation runs	$[0, 2^{31}]$	4
S_B	Number of first blocks to be discarded	$[0, B]$	0
N_D	Start-up delay (ms) to allow sync. of coords.	$[0, 2^{31}]$	5000
L	Log level, console verbosity	$enum$	$INFO$

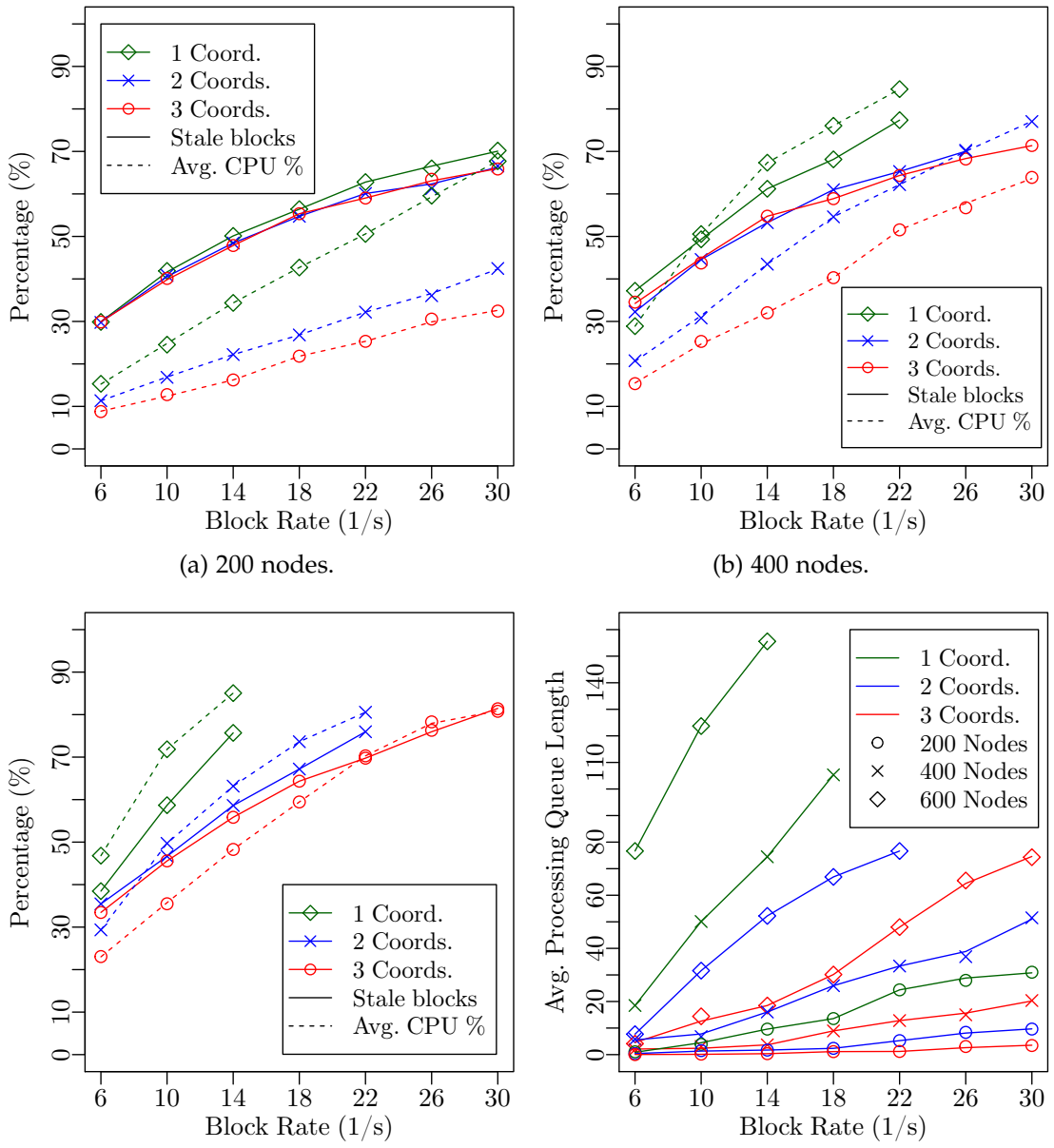
Table 5.2.: Selection of emulator results and measurements.

Par.	Description
R_{CPU}	List of maximum and average CPU loads of each coordinator
R_{BS}	Average block size
R_{FB}	Number of full blocks
R_{SB}	Number of stale blocks
R_{CSB}	Number of confirmed stale blocks
R_{FL}	List of fork lengths and the number of their occurrence
R_{BPS}	Blocks created / confirmed per peer second
R_{TPS}	Transactions created / confirmed per peer second
R_{TL}	Mean, median, minimum, maximum transaction latencies

5.3. Scalability and Limitations

In order for our results to be repeatable, experiments producing correct measurements on a single coordinator should produce the same results even when dividing the same emulated network between multiple coordinators and machines. To validate this design goal and investigate the scalability limits of our emulator, we conduct experiments on a fully switched network of up to three coordinator machines (Win10, Ryzen 5 2600X, 16GB; Ubuntu 18.04, i5-5200U, 8GB; Ubuntu 20.04, i5-2410M, 6GB). Throughout our experiments we raise the emulation complexity by increasing the rate of blocks and total number of peers. Subsequently, we repeat all experiments on one, two and three coordinators while measuring the CPU load of the first machine (Ryzen 5). To evaluate the validity of our measurements, we use the percentage of produced stale blocks as a comparative measure of correctness.

Figure 5.7 visualizes the experiment results. Specifically, Figure 5.7a shows that the percentage of stale blocks created by 200 nodes stays roughly the same regardless of the underlying hosts as per our design goal. However, we notice a slight increase of stale blocks after surpassing 50-60% CPU load when only using one machine. During the emulation of 400 nodes (Figure 5.7b) and 600 nodes (Figure 5.7c) this observation becomes even clearer. While results stay the same at lower node counts, block rates and thus, CPU usage, after passing 60% of average CPU load, additional stale blocks are introduced by the emulated peers. Due to our highly multi-threaded approach using scheduling to create and propagate blocks according to latencies, in an attempt to enable both horizontal and vertical scalability while retaining realism, the correctness of our results depends on the real-time capabilities of the underlying hosts. This observation is confirmed by Figure 5.7d showing an increasing length of queued processes when raising experiment complexity. Due to the longer processing queue and resulting CPU cycling, certain events happen later than they were scheduled. As a result, additional, artificial latencies are introduced to the network. Consequently, blocks take longer to be propagated, which in turn manifests in an increasing amount of stale blocks.



(a) 200 nodes.

(b) 400 nodes.

(c) 600 nodes.

(d) Length of processing queue.

Figure 5.7.: Additional stale blocks introduced by high CPU loads resulting from long processing queues.

Nevertheless, our results demonstrate the horizontal scalability of the emulator. Real-time capabilities of the underlying test bed can effectively be increased by adding more coordinator hosts and cores. Vertically, the scalability appears to be limited to 60% of average CPU load before result correctness starts to decrease. To compensate for this limitation we include measurements of the CPU load in the emulator's output and display warning messages in case of threshold values being surpassed.

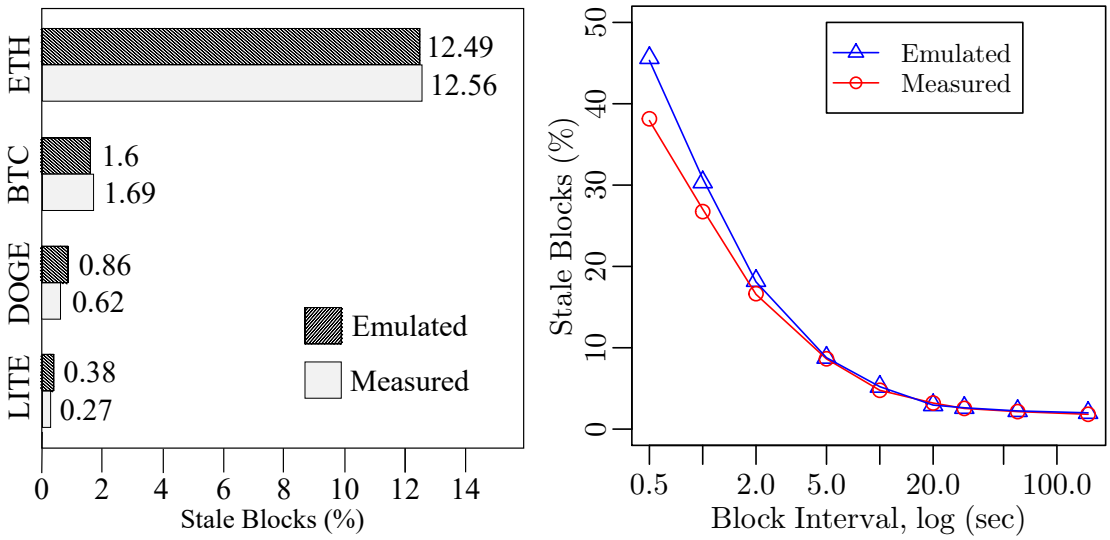
Apart from CPU load, the emulator's scalability on individual machines is additionally limited by the number of available file descriptors needed to create socket connections, as well as heap space provided by the *Java Virtual Machine* (JVM). To maximize a coordinator's capabilities, the limit of file descriptors can be increased using the `ulimit` command on Linux. Similarly, `Java` offers `-Xms` and `-Xmx` flags used to define initial and maximum heap space (Listing 5.4). Finally, our emulator focuses on the evaluation of static networks. During an emulation run, joining or leaving peers lead to a termination of the current run and thus, incomplete results.

Listing 5.4: Maximizing coordinator's resource limits.

```
$ulimit –n 100000
$java –jar –Xms4g –Xmx8g path/to/jar.jar
```

5.4. Validation

Throughout the emulator's development, exhaustive unit and integration tests resulting in 95% code coverage, and the additional use of static code analysis tools, make us confident in the functional correctness of our implementation. However, we are also highly interested in demonstrating the semantic correctness of our approach, especially in regards to the emulator's measurements and its fidelity of mimicking real-world architectures.



(a) Measured cryptocurrencies.

(b) Measured simulator [125].

Figure 5.8.: Comparing the emulator to Ethereum [121], Bitcoin [146], Dogecoin and Litecoin [125] and an existing simulator [125].

To evaluate the precision of our system, we first emulate four proof-of-work blockchains and configure their networks to have a block rate and average propagation delay

equal to real-world cryptocurrencies. For each experiment, we generate up to 10k blocks over four emulation runs and measure the number of stale blocks created by the networks. Subsequently, we compare our results to recent measurement studies of stale blocks produced by Bitcoin [146], Ethereum [121], Dogecoin and Litecoin [125]. Figure 5.8a shows that the trend of our data matches existing measurements with a mean relative error of 21.33%. Here, possible deviations may additionally be explained by missing information on parameters required by the emulator. In a second experiment, we compare our emulation to the simulator developed in [125]. To this end, we configure a network with equal latency and measure the percentage of stale blocks occurring in 10k total blocks generated at different interval times ranging from 0.5 to 150 seconds. Note that we run our emulation at 70% of the block rate used in [125], as 30% of their computational power is attributed to a selfish mining attacker. Here, our emulator is subject to a mean relative error of 8.7%.

As expected by an emulation-based approach influenced by abstraction, our results experience some inaccuracies, potentially due to incomplete configuration. Nevertheless, the general trends of our measurements match those of recent works, demonstrating our emulator's potential for use in comparative studies.

6. Evaluation

In the following, we will use our emulator to gather empirical evidence on the performance of PoX and DPoS-BFT blockchains in a variety of network archetypes. Over several experimentation runs, we will explore the effects of these consensus protocols and their parameterization on performance, consistency and fault-tolerance of transactions. In particular, we especially investigate trade-offs and boundaries arising when trying to optimize for low latency. Additionally, we propose simple mathematical models of our findings, in an attempt to describe the relationship between some of the most impactful parameters of blockchain architectures. After comparing both of our implementations directly in their potential of hosting edge applications, i.e. an edge computing marketplace, we conclude in a discussion highlighting benefits and detriments resulting from such a deployment.

6.1. Design

As our hardware configuration, we use two different setups to conduct emulation runs depending on the experiment's complexity. For large networks, we deploy a cluster of 15 coordinator VMs connected by a 1 GB/s network with a total of 58 cores and 58 GB RAM. Smaller experiments are executed on a local test bed of three machines with a total of 10 cores. Both setups are summarized in Table B.1 and referred to by their names in the following.

During emulation runs, we measure the confirmation latency of a transaction (i.e. a resource bid) starting with the time of its creation at one of the nodes. After being propagated through the network, the transaction is included in a block, representing a successful auction match with other transactions part of the same block. The block - thus, the auction - is confirmed once it is extended by the configured number of additional blocks ($C - 1$). This effectively emulates steps one and two of our marketplace setting in Chapter 4, while revealing latency margins left open for the remaining steps, i.e., off-chain provisioning and result verification. Here, the transaction represents a resource request or offer. Consequently, including the transaction in a block corresponds to a completed auction match between all transactions that are part of the block. Once a block containing the auction match is confirmed, it is considered to be irreversible by its clients. Therefore, providers begin provisioning the requested resource or service supplied by the client. Next to transaction latency, we additionally measure the amount of (confirmed) stale blocks created by the network and investigate their effects on consistency, efficiency and required fault-tolerance of auctions, or other applications,

executed on top of the chain. Throughout the emulation, transactions are disseminated equally by all peers. Unless otherwise mentioned, the emulator is configured according to the default parameters of Table 5.1. Specific overrides of these default values are listed in Appendix B for all experiment figures. For each data point, up to 30k blocks, depending on block rate, were created.

As the underlying network, we choose to emulate three networks of small, medium and large sizes. The small and medium networks are scale-free networks of 100 and 500 nodes, respectively. They were created using the Barabási–Albert model with parameter $m = 2$ (Section 5.1.1), resulting in a similar topology as found in Ethereum [144]. Here, the small network represents a *Smart City* network with Gaussian latency distribution and mean latency of 10 ms [45]. The medium network was configured to have an average propagation delay of 63 ms, similar to latencies of cloud data centers as measured in [147]. The large network consists of 2700 global peers extracted from the same dataset. Here, the nodes describe unique ASNs of [147], which represent ideal locations for deploying edge servers as discovered by [148]. Consequently, we use the dataset's traceroute measurements between those ASNs to create a global topology of nodes. Listings 6.1, 6.2 and 6.3 describe the emulator's configuration entries for each of these networks.

Listing 6.1: Small Network

```
networkType: scaleFree
scaleFree:
nodes: 100 # number of peers in the network
m: 2 # Barabasi-Albert parameter
latency: 10000 # microseconds
perEdge: false # latency is mean of entire network
bandwidth: 1000 # MB/s
```

Listing 6.2: Medium Network

```
networkType: scaleFree
scaleFree:
nodes: 500 # number of peers in the network
m: 2 # Barabasi-Albert parameter
latency: 63000 # microseconds
perEdge: false # latency is mean of entire network
bandwidth: 1000 # MB/s
```

Listing 6.3: Large Network

```
networkType: explicit
explicit:
bandwidth: 1000 # MB/s
fileName: cloudGraph.txt
peers: [] # replaced by file
```

6.2. Results

In the following figures, data points represent the mean of all emulation runs of the same experiment. Solid lines either refer to our proposed model functions, or fitted Weibull curves [149] in case our model is unsuitable. In total, we propose four functions describing the transaction latency and stale block of PoX and DPoS-BFT blockchains, when deployed in networks with representative latency mean, i.e. the emulated small and medium networks. To summarize the quality of these models, Table 6.1 lists their accuracy and mean squared error present in the following figures and experiments.

Table 6.1.: Coefficient of determination and relative mean squared error of model functions. nRMSE is normalized using standard deviation.

	E6.1	**E6.5**	**E6.6**	**E6.9**
$RMSE$	20.48	4.38	8.36	0.276
$nRMSE$	0.028	0.202	0.007	0.192
R^2	0.99	0.97	0.99	0.98

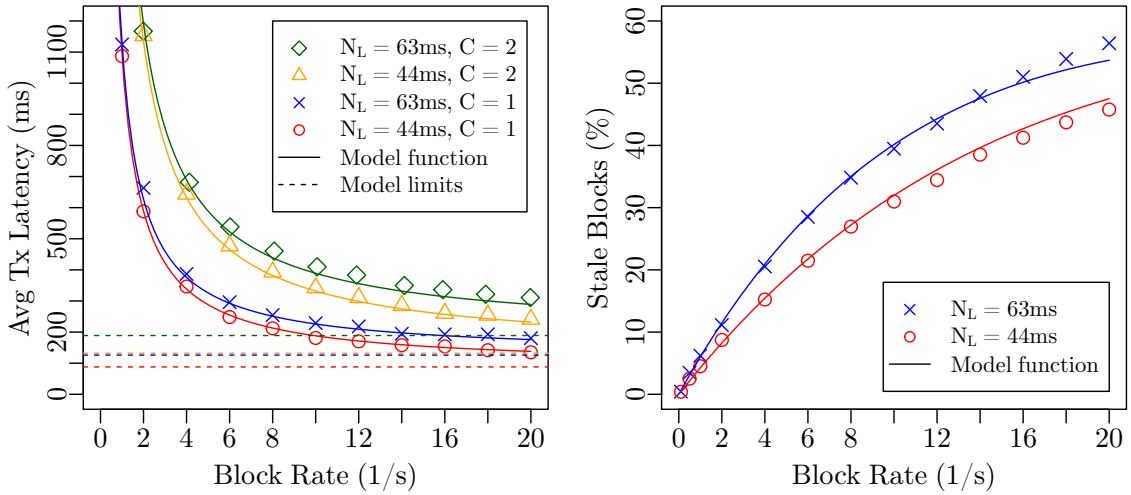
6.2.1. PoX

In our first experiment, we increase the block rate (B_R) of a generic proof-based blockchain deployed in two medium networks with 44 and 63 milliseconds mean propagation delay (N_L). The data points of Figure 6.1a describe the mean latency when waiting for one or two confirmations (C) after submitting a transaction. Overall, latencies appear to decrease for higher block rates, up to an asymptotic limit. Due to the reduced time spent by transactions waiting to be included in the next block, a higher block rate leads to faster confirmations and thus, lower transaction latencies. As more blocks are created per seconds, the returns of creating blocks even faster are diminishing, resulting in the observed lower bound. We model our results using Equation (6.1) to describe the mean transaction latency L_{Tx}.

$$L_{Tx} = N_L + (N_L + \frac{1000}{B_R}) \cdot C \tag{6.1}$$

$$\lim_{B_R \to \infty} L_{Tx} = N_L + CN_L \tag{6.2}$$

Here, the transaction latency is calculated as a sum of the average time needed to send a transaction through the blockchain network (N_L), and the latency of propagating C confirmation blocks at rate B_R influenced by the same network latency N_L. By increasing B_R, waiting times between blocks become more and more irrelevant, leading to the asymptotic limit of Equation (6.2), which precisely describes the round trip time of sending one transaction and receiving C *consecutive* confirmation blocks. In order for blocks to append upon each other, every block requires on average N_L time units in



(a) Transaction latency model at different confirmations and network latencies.

(b) Stale blocks and stochastic models at different network latencies.

Figure 6.1.: Model functions of latency and stale blocks in medium PoX network.

order to be received by other peers. Once received, peers are able to build on top of the block and create a sequence of confirmation blocks.

Blocks created outside of this sequence become *stale*. The data points of Figure 6.1b describe the increasing percentage of stale blocks generated when raising the overall rate of blocks. Naturally, as blocks are created faster by all peers and concurrency increases, the underlying network fails to propagate blocks in time, such that they cannot be built on top of one another. Consequently, more stale blocks referencing the same parent block are created. To model the creation of stale blocks, let L be the average network propagation delay between any two peers in seconds, and R the number of blocks created by the entire network per second. Then, X in Equation (6.3) represents a Poisson distributed random variable describing the number of blocks created within the time it takes for one block to be propagated on average. We can use X to calculate the probability of k blocks being created by applying the Poisson distribution's density function as shown in Equation (6.4). Observing our results of figure Figure 6.1b, the probability of a stale block being created appears to be roughly following the arithmetic mean of *at least one* block being created and *exactly one* block being created within the network's average propagation delay (Equation (6.5)).

$$X \sim Poisson(L \cdot R) \tag{6.3}$$

$$P(X = k) = \frac{e^{-(L \cdot R)}(L \cdot R)^k}{k!} \tag{6.4}$$

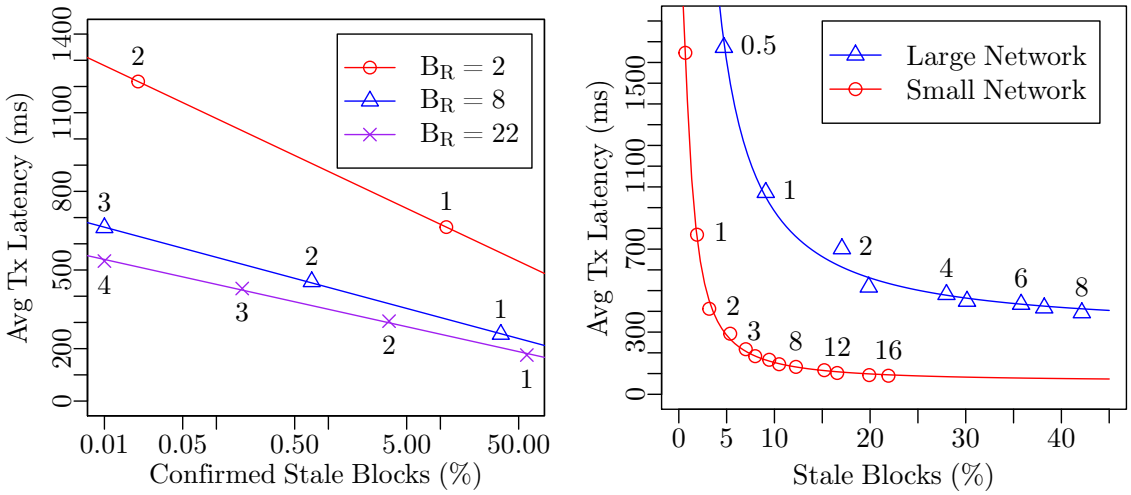
$$P[\text{Stale block}] = \frac{1}{2}(1 - P(X = 0)) + \frac{1}{2}P(X = 1) \tag{6.5}$$

Stale blocks describe a *waste* of resources, as their creation, propagation and storage does not meaningfully contribute to the progression of the distributed ledger. Furthermore, they represent a security and consistency concern. Once a client's transaction is assumed to be confirmed, subsequent actions are initiated, i.e. sending a product purchased with cryptocurrency, or provisioning resources after receiving an auction match. In case a block containing such a transaction ultimately turns out to be stale, its resulting actions (which may not be reversible) are invalid as well. In case of a purchase with cryptocurrency, this results in funds becoming double-spendable. Similarly, a confirmed auction match turning stale could entail expensive roll-back mechanisms or monetary losses. Deliberately abusing this characteristic of proof-based blockchains in the scope of a targeted attack has been investigated previously by related work [70], [125]. Consequently, waiting for additional confirmation blocks is required to ensure that a block is, in fact, part of the main chain with high probabilistic finality.

Overall, we can clearly observe a trade-off between performance (transaction latency) and efficiency, consistency, security (stale blocks) in PoX, which is further investigated in Figure 6.2. Here, Figure 6.2a shows how waiting for multiple confirmation blocks in the medium network reduces the percentage of *confirmed* stale blocks exponentially[1]. However, as indicated by Equation (6.1), it also leads to a linear increase in latency. In combination, high block rates with many confirmations appear to entail lower latencies than low block rates with fewer confirmations at an equal percentage of confirmed stale blocks. The latter observation however comes at the cost of efficiency as seen in Figure 6.2b. Here, the depicted experiment demonstrates how increasing the block rate in an attempt to reduce latencies leads to a nearly exponential increase in stale blocks and vice versa. Although the trade-off in the small, city-wide network is much cheaper compared to the global network, multiple confirmation blocks are required to ensure security and consistency.

Regarding an edge computing marketplace on top of PoX, increasing the block rate B_R also relates to raising the frequency and concurrency of auctions. Consequently, stale blocks imply invalid auctions, and potentially a single bid being included in multiple concurrent auctions. The correct auction is then determined by waiting for additional confirmation blocks. As before, reducing the block rate to minimize the number of stale blocks, directly affects the latency of auctions, and thus, the provisioning time of the requested edge resources. Although, as indicated by Figure 6.2a, waiting for one additional confirmation is enough to reduce the percentage of naturally occurring, confirmed stale blocks by over 90%, especially in a public-permissionless setting with high rate of blocks, more confirmations are required to counteract byzantine failures and malicious attacks [70], [125]. Otherwise, considering an auction block as valid too early, could lead to it becoming stale after the resource provisioning process was already initiated or even after its completion. Consequently, the provision is invalid, due to its auction match not being part of the main chain, and neither party is compensated.

[1] *Confirmed stale blocks* describe blocks turning stale despite being confirmed by the required number of additional blocks at an application level. Note that confirmations do not reduce the number of *general* stale blocks impacting the blockchain's efficiency.



(a) Log scale of confirmation trade-off at varying block rates in medium network.

(b) Fitting Weibull curve to block rate trade-off in small and large networks.

Figure 6.2.: Trading (confirmed) stale blocks for Tx latency in PoX.

Instead, the stale bids are resubmitted to be included in a future block on the main chain. Nevertheless, in a permissioned setting requiring less confirmations, PoX might be able to support low enough latencies ($< 200ms$) to enable a deployment of less stringent edge applications [3]. Here, as indicated by Equations (6.1) and (6.5), the network's propagation delay N_L appears to be a relevant parameter for both, stale blocks and transaction latency.

6.2.2. DPoS-BFT

Similar to our experiment of the previous section, the data points of Figure 6.3 describe the average latency of transactions and percentage of stale blocks when increasing the block rate (B_R) in DPoS-BFT, deployed on our medium network with $63ms$ mean propagation delay. As described in Section 2.3.3, instead of blocks being created randomly and concurrently by all peers (PoX), DPoS-BFT selects a smaller set of block producers (BPs) who then deterministically create batches of blocks in round-robin fashion. Additionally, DPoS-BFT deploys a *pipelined* BFT protocol between all BPs in order to speed up block finality (Figure 2.10). Analogous to PoX, transaction latencies decrease with higher block rates due to reduced waiting times up to an asymptotic limit. Moreover, transaction latency appears to improve even further, when fewer BPs are selected based on their connectivity. Due to the BFT protocol's broadcast between BPs, newly proposed blocks experience additional latency before they are acknowledged by all other BPs and then sent out to the remaining network. Consequently, block and transaction latency can be improved by reducing the propagation delay between BPs, as shown in Figure 6.4. Here, we raise the mean latency between BPs, and notice a near linear increase in transaction latency at factor 2-3. As the 2-stage, pipelined BFT

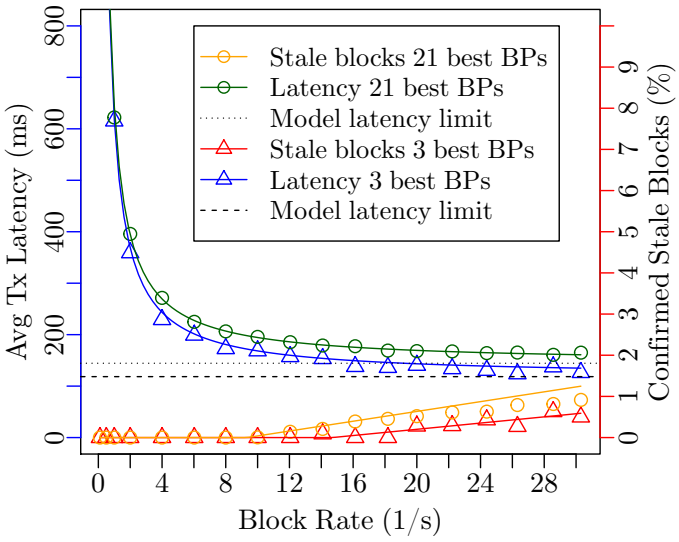


Figure 6.3.: Model functions of latency and stale blocks in medium DPoS network.

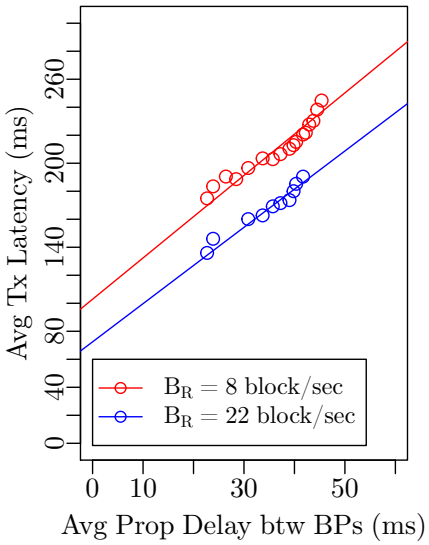
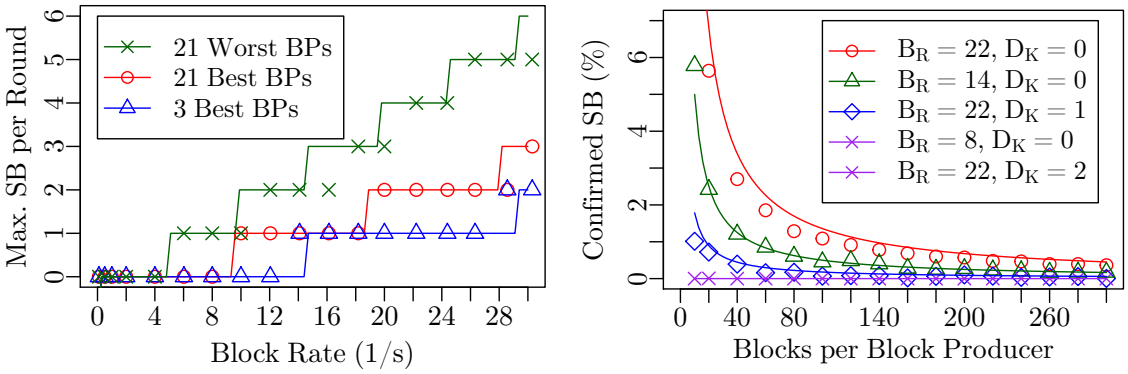


Figure 6.4.: Inc. latency between BPs.

protocol requires $^2/_3$ BPs to acknowledge the newly proposed block before it is sent out, the observed factor seems plausible. Combined, we propose Equation (6.6) as an approximation of transaction latency (L_{Tx}) in DPoS-BFT. Similar to our model of PoX, the function describes a sum of the average network latency N_L experienced by the initial transaction, the average waiting time for it to be included in a block at rate B_R, the block to be acknowledged by BPs experiencing a mean latency of L_{BP} and a constant $K = 10$.

$$L_{Tx} = N_L + 2L_{BP} + \frac{2000}{B_R} + K \tag{6.6}$$

In DPoS-BFT, stale blocks are created if the block production round of one BP starts, before all blocks of the previous BP's round were received (Figure 2.11). During the emulation, each block production *round* is divided into D_B time *intervals*. At the start of each interval, the current BP sends one block proposal to all other BPs and gathers their acknowledgments. Once enough acknowledgments are received, the block is gossiped to all peers and BPs. Consequently, in order to avoid the block becoming stale, the process of proposing, acknowledging and broadcasting a block should not take longer than its designated interval. As each BP produces multiple blocks in a row, especially the duration of the last block production interval of each round, before a BP hand-off, is relevant to the creation of stale blocks. We can increase this duration by reducing the total number of blocks produced by each BP and assigning multiple intervals to the last block created during each round. D_K describes the number of final blocks to be *skipped* this way during every round, in order to increase the length of the last production interval. We can calculate the duration of the final interval I_L using the block rate B_R as



(a) Higher latency between BPs resulting in more stale blocks $\lceil SB_R \rceil$.

(b) Reducing stale blocks SB in DPoS using B_R, D_B and D_K.

Figure 6.5.: Model function of stale blocks created by DPoS during BP hand-offs.

shown in Equation (6.7).

$$I_L = \frac{(D_K + 1) \cdot 1000}{B_R} \tag{6.7}$$

Figure 6.5a shows how increasing the rate of blocks reduces the time designated for the last block production interval, resulting in more stale blocks per round. Knowing that the propagation of a block between BPs takes 3 messages according to the pipelined BFT protocol, and is influenced by an average latency of L_{BP}, we can model the number of stale blocks created during each round as $\lceil SB_R \rceil$ of Equation (6.8). This effectively describes the number of intervals fitting within the time it takes to propose and propagate one block. To infer the percentage of produced stale blocks, we divide SB_R by the number of total blocks created during each round in Equation (6.9).

$$SB_R = Max\left(0, \frac{3L_{BP}}{I_L} - 1\right) \tag{6.8}$$

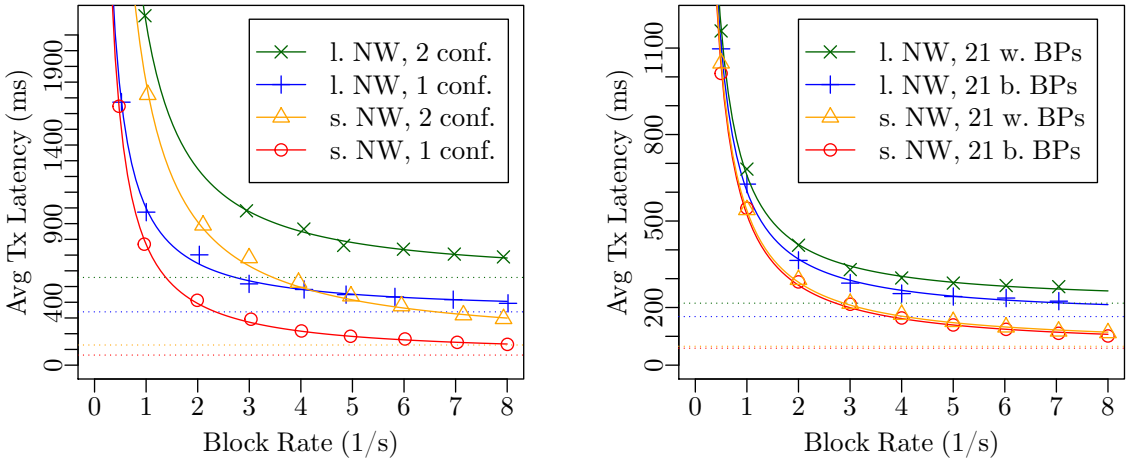
$$SB = \frac{SB_R}{D_B - D_K} \tag{6.9}$$

Figure 6.3 describes a linear increase of stale blocks, once production intervals are too short for final blocks to be propagated in time. As a result, Figure 6.5b proposes three counter measures to reduce the number of stale blocks in DPoS-BFT: 1) Reducing the rate of blocks, and thus, increasing the duration of block production intervals at the cost of latency. 2) Increasing the number of blocks created by each BP to minimize the amount of hand-offs at the cost of decentralization. 3) Increasing the duration of only the last production interval by discarding the final D_K blocks. Here, transactions handled during the prolonged interval experience slightly longer latencies.

By centralizing the consensus to a subset of BPs, DPoS-BFT produces a low percentage of stale blocks even at high block rates. In regards to our marketplace application,

DPoS-BFT ensures that only one BP proposes new blocks, and thus, auctions during any production interval. Nevertheless, stale auction blocks can still be created between hand-offs of BPs. This can be avoided by increasing the time allocated for the last block produced by each BP or by raising the total number of auctions per BP. Overall, DPoS-BFT achieves lower latencies and stale block rates compared to PoX. Instead, the performance of DPoS-BFT is influenced by the individual performance of block producers and the network between them. Consequently, QoS of our marketplace, manifesting in auction performance and provisioning time, is reliant on the quality of selected BPs. To this end, in case of a crashing or slow BP without another one immediately taking over, DPoS-BFT experiences much lower fault tolerance due to its lacking replication. This could lead to some auctions and their resource provisions taking significantly longer than others. Compared to PoX, the trade-off between consistency and performance is transformed into performance competing against fault-tolerance, decentralization and availability when opting for DPoS-BFT.

6.2.3. Network



(a) PoX at 1 or 2 confirmations.

(b) DPoS for best and worst BPs.

Figure 6.6.: Transaction latencies and asymptotic limits of fitted Weibull curves in small (s.) and large (l.) networks.

After evaluating the limits of reparameterization in PoX and DPoS, we take a closer look at the underlying network. Figure 6.6 describes PoX and DPoS-BFT in our small and large networks. While the small network represents a smart city of 100 nodes with 10*ms* average latency, the large network consists of 2700 globally distributed peers. In both figures, dashed lines represent the asymptotic limits of latency reduction induced by lower waiting times between blocks. While in PoX we additionally investigate the behavior of requiring 1 or 2 confirmations, in our DPoS-BFT experiment, we differentiate between selecting the 21 best and worst BPs in terms of their connection degree.

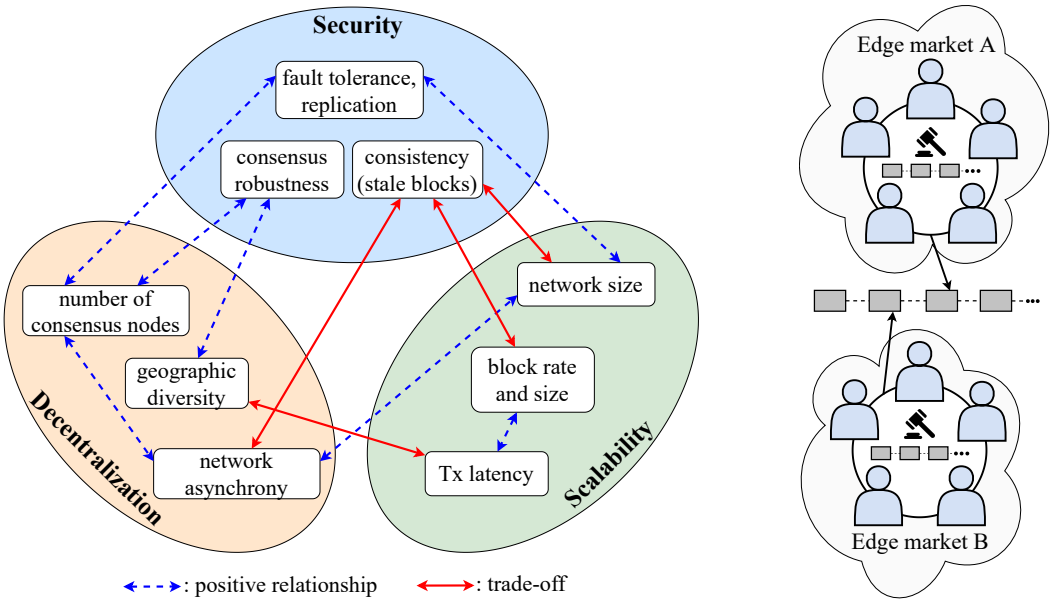
Naturally, and as previously seen in Figures 6.2b and 6.3, improving the consensus nodes' underlying network conditions allows performance to be increased altogether, avoiding some of the most costly trade-offs. As already suggested by our mathematical models, in both experiments, the asymptotic latency limits are significantly lower in the city-wide network than in the global network. Similarly, selecting well-connected nodes in the large network appears to be more critical than in the small network due to overall better conditions. For both consensus protocols, latency boundaries around 100ms can be observed if deployed entirely in the smart city environment. Depending on higher layers of the targeted architecture, this could leave a sufficient margin to support some of the edge computing use cases presented in this work, although for PoX the need for additional confirmations, depending on required consistency and security, needs to be considered.

6.3. Discussion

Overall, while it seems as though PoX could achieve latencies below 200ms (upper latency bound of edge computing feasibility [3]) at high block rates, in practice it is infeasible due to the large amount of stale blocks and resulting need for additional confirmations to ensure security and consistency of auctions. The only way to improve both latency and consistency (stale blocks) simultaneously, appears to be reducing the network latency, by limiting its size. In comparison, the results of DPoS-BFT are much more promising due to overall lower latency and essentially non-existing stale blocks. However, compared to PoX, it offers no consensus replication and less fault-tolerance, while benefiting equally from smaller network sizes.

In total, our results demonstrate the harsh limits and trade-offs of blockchain reparameterization [150]. Once these limits are reached, improving the underlying network conditions appears to be the only option of increasing transaction performance without sacrificing crucial security or consistency requirements of the entire architecture. This observation goes hand-in-hand with the *scalability trilemma* of blockchain, as originally proposed by Ethereum [11] co-founder Vitalik Buterin. Therein, Buterin describes the inevitability of trade-offs between the security, decentralization and scalability of DLT, and proposes *sharding* as a potential solution [151]. In general, sharding describes the idea of breaking up a larger architecture into multiple, smaller parts. The resulting *shards* may be partitions in terms of the network, its users, or the created data and transactions. Figure 6.7a summarizes some of the partially transitive relationships and trade-offs that we identified throughout our evaluation, backed by the scalability trilemma. Arguably, as security is an essential and inevitable requirement, especially the depicted relationships between decentralization and scalability are relevant to us. These specifically manifest in the latency trade-off resulting from an increased geographical diversity and total number of consensus nodes.

In relation to this observation, our results show that in order to come even close to the latency requirements of edge applications, while remaining secure, consistent and



(a) Security-decentralization-scalability trilemma in blockchain. Adaptation of [8].

(b) Using global ledger to synchronize local edge markets.

Figure 6.7.: Trade-offs in blockchain leading to sharding solutions.

available, DLT needs to reside within the edge network in its entirety. Naturally, the edge computing paradigm is very localized by design and as a result, it is only sensible for its control plane to be part of the same environment. To avoid the bottleneck of replicating data relevant to the edge in a global network, the market and the blockchain are limited to the local edge, as proposed by the sharding paradigm. The resulting small and restricted edge markets, backed by fragmented blockchains, could then eventually be synchronized by a global *interledger* as shown in Figure 6.7b, reigniting the vision of a global edge computing marketplace. Target applications aiming to work in a fully decentralized fashion are deployed in localized markets and administered by global accounts maintained on the inter ledger. Still, avoiding the global ledger becoming a bottleneck is paramount to this idea. Specifically, it should enable a communication and exchange between localized edge markets without introducing the risk of double-spending.

Similarly, while improving usability and performance tremendously, limiting the blockchain network's size to local edge markets directly inhibits its own paradigm of success, *strength-in-numbers*. By breaking the network up, especially security, privacy and trust benefits of large blockchains start to crumble, opening up potential attack vectors on the blockchain and, thus, the market. To this end, known vulnerabilities of proof-based blockchains such as alternative history or 51% attacks conducted by an outsider become even more apparent. On a similar note, DPoS blockchains like EOS [84] are already suspected of suffering under the formation of malicious cartels enabled by its public voting procedure on block producers [152]. Instead, these block

producers may need to be operated or hand-selected by a governing authority in a smart city environment. Finally, the requirement of high block rates to ensure low latency of auctions and their resulting resource provisions leads to a significant reduction of efficiency and consistency in permissionless DLT, due to probabilistic finality caused by stale blocks.

In general, limiting network size and closing aforementioned attack vectors on DLT implies a deployment in permissioned mode under a central authority, i.e., the smart city administration. As a result, the original promises making blockchain attractive in a crowdsourced edge computing marketplace, i.e., being inherently crowdsourced themselves, offering open participation, trustless consensus, privacy and security by design, become decreasingly noticeable or even disappear entirely. Instead, deploying an edge marketplace under centralized authority, to begin with, seems to yield better performance, usability and maintenance, albeit at the cost of trust and privacy. After offloading registration and certification to a trusted third party to facilitate a permissioned environment, centralizing resource auctions to a trusted auctioneer in a similar way appears to be a logical step to improve QoS.

6.4. Limitations

Our results describe networks with very high bandwidths of 1 GB/s. Additionally, unsolicited block push is used to further optimize latency at the cost of higher bandwidth consumption. The effects of lower bandwidths are expected to be similar to those of a higher latency, where a slower propagation of blocks leads to higher transaction latencies and stale block rates.

The proposed models in Equations (6.1), (6.5), (6.6) and (6.9) simplify the underlying network by reducing its parameters to solely the latency mean N_L, present in the entire network. As a result, the models' fit is significantly worse when applied to networks without a representative latency mean ($mean \approx median$), i.e. the large network created from global cloud measurements. Here, the long tail latency distribution of the cloud network, and therefore its topology, is not properly represented by its mean. Similarly, equations modeling DPoS-BFT make use of the mean latency between BPs (L_{BP}). Here, a BP with significantly worse connection than the mean will generate higher transaction latencies and stale-block rates during its production interval. However, the remaining BPs are largely unaffected during their own intervals, as only $^2/_3$ acknowledgments need to be received in order to publish a block, without having to wait for the slower BP's acknowledgment.

Due to scalability limitations of our emulator and the available test-beds (Section 5.3), experiments were conducted using comparatively low transaction rates (T_R) of 10 submitted transactions per second, to reduce CPU utilization. Therefore, throughput measurements returned by the emulator (R_{TPS}) were not representative of the maximum throughput capabilities supported by the emulated blockchain architecture. Instead, maximum throughput can be approximated using the percentage of stale blocks (R_{PSB}),

the block rate (B_R) and the block size (B_S) by calculating:

$$R_{TPS} \approx \left(1 - \frac{R_{PSB}}{100}\right) \cdot B_R \cdot B_S, \tag{6.10}$$

under the assumption that the bandwidth is high enough to support sending full blocks of size B_S without a significant increase in latency.

Driven by the exponential increase of possible experiments in the number of inputs, many parameters of our emulator remained fixed to default values throughout all evaluations and are excluded in our models. Instead, we focused on investigating the most relevant parameters of blockchain performance. Nevertheless, in the following, we summarize ancillary findings on these parameters and describe their expected effects. Transaction time and fee distributions (T_F, T_{PD}, T_{TD}) will improve the latency of some transactions, if they receive a higher fee or are created by well connected nodes. Increasing the verification time (V_T) of blocks before they are propagated is expected to have a similar effect as increasing the network's latency. While fork resolution according to GHOST promises stronger protection against alternative history attacks, it is computationally more intensive than LCR [68]. While a random tie resolution (B_{TR}) does not increase the percentage of stale blocks in general, it increases the length of created forks on average. Changing the distribution of lottery power in PoX (P_D) might improve performance if well connected peers receive the majority of lottery share.

7. Conclusion

In this thesis, we evaluated the capabilities of a crowdsourced edge computing marketplace backed by decentralized DLT using emulation and experimentation. By shining light into popular blockchain-based black boxes, we gave a more practical and realistic outlook on DLT in edge.

7.1. Summary

Chapter 1 provided an introduction to the topic by motivating the idea of using DLT in edge and specifically as a distributed auctioneer of crowdsourced edge computing marketplaces. We identified the increasing research demand resulting from blockchain being used as a black box solution in edge, and formulated our objective of using emulation to investigate common assumptions, trade-offs and capabilities of DLT in edge computing use cases.

In Chapter 2, we gave background on key concepts regarding edge computing, crowdsourcing and DLT. Here, we discovered the attractiveness of using edge computing to enable latency stringent applications, but also identified challenges resulting from managing resources in a distributed and heterogeneous environment at minimal latencies. To this end, we introduced crowdsourcing and specifically DLT as a proposed, self-sufficient solution of managing edge resources and applications in a secure and decentralized way. Therein, we divided DLT into 4 sub-categories and provided valuable ground work for our investigation of trade-offs and subsequent implementation.

As part of Chapter 3, we gave an overview of previous work relevant to us. After presenting a survey of DLT and its use in edge and specifically edge marketplaces, we discovered the temptation of adopting blockchain to leverage promises such as decentral trust and security by design, while disregarding potential performance bottlenecks and implications. As a result, we stated the pressing need for concise empirical evidence on the performance of using blockchain in edge and described our emulation-based approach in contrast to existing attempts at evaluating DLT.

Chapter 4, introduced our version of a crowdsourced edge computing marketplace backed by blockchain. After reviewing related work and its drawbacks, we devised a performance optimized marketplace architecture by aligning its components with the capabilities of blockchain. By conducting a new resource auction as part of every block on the chain, latencies are reduced and boundaries of the underlying blockchain network are easily investigated.

Chapter 5 described the architecture of our emulator framework. During development,

we paid close attention to the modularity of our approach, allowing us to support a total of 26 parameters relevant to blockchain performance. As consensus protocols, we implemented a generic proof-based solution and DPoS-BFT following our description in Chapter 2 closely. Subsequently, we were able to define our emulator's limits in terms of vertical scalability, while demonstrating its ability to scale horizontally. Finally, we validated our implementation against measurements of real cryptocurrencies at managable relative error levels, showing an accuracy of above 85%.

In Chapter 6, we applied our emulator in order to evaluate latency boundaries of blockchain transactions. Using experimentation, we discovered how limits of reparameterization in PoX and DPoS-BFT consensus protocols lead to costly trade-offs in the security and consistency of blockchain architectures. Overall, the most important results of our experiments can be summarized as follows:

- High block rates are paramount to enable low latency of auction blocks and transaction bids. However, in PoX these are practically infeasible due to the probabilistic finality of auctions and resulting need for additional confirmation blocks.
- DPoS-BFT achieves better consistency at high block rates by synchronizing and centralizing consensus to a subset of block producers, thus avoiding concurrency of several auctions. However, lower-fault tolerance implies the need for capable consensus nodes.
- The only way to achieve latencies within the edge computing feasibility zone of 200ms [3], while avoiding costly trade-offs in consistency, security and fault-tolerance, is to improve underlying network conditions and increase centralization, i.e. by limiting the blockchain to the edge network entirely and enforcing permissioned participation.

Based on these results, we proposed the use of sharding as a potential solution to the blockchain scalability trilemma. By splitting the global network into locally restricted edge marketplaces and blockchains, synchronized by a shared interledger, network conditions of local blockchains are improved sufficiently. Subsequently, we discussed potential detriments of such an approach. These especially manifested in the attractiveness and promises of blockchain in edge becoming increasingly unnoticeable, after violating the DLT paradigm of strength-in-numbers.

7.1.1. Realized Goals

In total, we successfully summarized latency requirements of next-generation edge applications and identified parameters relevant to a blockchain's performance in edge. By implementing a generic emulation framework for network and blockchain architectures, effects of these parameters were evaluated empirically. As a result, we proposed 4 simple, empirical models describing the latency and consistency of 2 consensus protocols. Lastly, we investigated the suitability of blockchain as a decentralized auctioneer for

edge computing marketplaces. Here, we proposed an optimized marketplace architecture on top of DLT and discovered its need to reside within the edge network entirely. Consequently, we proposed a sharding solution and discussed its limitations.

7.1.2. Open Goals

In this work, we especially focus on the transaction latency of blockchains as a requirement of enabling edge computing applications. However, for many use cases identified in Section 2.1.2, sufficient *transaction throughput* represents a similarly important requirement. Our results in that regard are limited by 1) our simplified bandwidth emulation not considering network congestion in case of multiple blocks being sent at once, and 2) our hardware setup not supporting the experimentation of high transaction rates while maintaining a larger network size. Instead, we present an approximate model to calculate transaction throughput and implement a parameter B_{FB}, allowing the emulation of full blocks even at lower transaction rates. As shown by our results, the necessity of low transaction latencies implies the configuration of high block rates. At equal transaction rates, a higher block rate therefore naturally results in smaller blocks. Consequently, we don't expect the individual size of blocks to be a limiting factor of blockchain in a high-bandwidth edge.

Additionally, we did not succeed in thoroughly evaluating the effects of all parameters implemented by the emulator and did not include them in our mathematical models. To the best of our knowledge, we instead focused on parameters most relevant to this work and provided a brief summary of the remaining emulation parameters' effects in Section 6.4.

7.2. Implications

Summarizing our results, a clear mismatch between blockchain and edge technologies can be observed. While DLT excels in openly large and globally distributed networks, edge computing is localized and restricted by its very own definition. Edge computing is centered around the idea of consuming data in close proximity to where it is produced, whereas in DLT, transactions are replicated and shared between every single node of a global network before they can be considered valid. Attempts at merging both technologies seem forced, and limiting the open and decentral nature of blockchain is counter-intuitive at best. Instead, these attempts may potentially be attributed to the ever-lasting *hype* surrounding DLT and edge, hidden underneath the puzzling and seemingly impenetrable blockchain black-box. For all other cases, this work offers an extensible emulation framework enabling the detailed investigation of trade-offs and capabilities of edge computing use cases built on top of DLT.

7.3. Future Work

In addition to aforementioned open goals, for future work, we plan to extend the emulator by implementing and evaluating the remaining two DLT categories identified in Section 2.3, namely DAG and BFT-based blockchains. Furthermore, we eagerly await upcoming developments of DLT to include in our evaluation. Similar to our suggested *interledger* approach, Ethereum 2.0 [153] intends to split its network into 64 *shard chains* synchronized by a central *beacon chain*, to improve transaction handling performance. *Solana* [154] proposes a novel consensus protocol, *proof-of-history*, aiming to improve the finality of existing proof-based protocols by verifying the order and passage of time between events on the ledger. In a similar approach to Ethereum 2.0, *Elrond* [155] explores adaptive state sharding by reassigning active nodes to necessitating shards. Finally, we aim to build upon our emulator by further improving its realism, especially in terms of cryptographic operations or the network bandwidth. Additionally, we plan on supporting specific use-cases, i.e. a marketplace, natively as part of the emulation.

The insights of this thesis and the developed emulation framework open up further investigations on blockchains. Here, our emulator can be used to generate empirical evidence for the suitability of blockchain in upcoming edge computing use cases, thus avoiding the pitfalls of adopting DLT as a black-box solution.

A. Protobuf Message Definitions

Listing A.1: Coordinator message wrapper

```
message CoordinatorMessage {
  optional AssignNodes assignNodes = 1;
  optional NodesAssigned nodesAssigned = 2;
  optional InitP2POverlay initP2P = 3;
  optional ConsensusNodes consensusNodes = 4;
  optional P2PReady ready = 5;
  optional Start start = 6;
  optional Stop stop = 7;
  optional Result result = 8;
}
```

Listing A.2: Coordinator messages

```
// Orchestrator signals Coordinator to initialize peers within the given (inclusive) ID range
message AssignNodes {
  required int32 from = 1;
  required int32 to = 2;
}

// Coordinator informs Orchestrator of the initialized peers and their IP addresses
message NodesAssigned {
  repeated Node node = 1;
}
// Mapping of node ID to IP address
message Node {
  required int32 nodeId = 1;
  required string address = 2;
}

// Orchestrator sends network graph as an adjacency list and a mapping of node IDs to IP addresses
message InitP2POverlay {
  repeated Node node = 1;
  repeated Edge adjacency = 2;
}
message Edge {
  required int32 node = 1; // from node
  repeated Latency edge = 2; // list of edges
}
```

```
26 message Latency {
27   required int32 to = 1; // to node
28   required int32 latency = 2; // latency to node
29 }
30
31 // In case of DPoS: Orchestrator informs other Coordinators of selected BPs
32 message ConsensusNodes {
33   repeated int32 node = 1;
34 }
35
36 // Coordinator signals Orchestrator that its side of the P2P network is ready
37 message P2PReady {
38 }
39
40 // Orchestrator signals other Coordinators to start
41 message Start {
42   required int64 time = 1; // when to start
43 }
44
45 // Coordinator tells other Coordinator to stop with the given reason, may be relayed
46 message Stop {
47   required string initiator = 1;
48   required string reason = 2;
49   required bool fatal = 3;
50 }
51
52 // Coordinator delivers results to Orchestrator
53 message Result {
54   required string coordAddress = 1;
55   required double maxCPULoad = 2;
56   required double avgCPULoad = 3;
57   // contains detailed measurements of created blocks, transactions and latencies
58   repeated ResultEntry entry = 4;
59 }
```

Listing A.3: Peer message wrapper

```
1 // Messages sent between emulated peers
2 message P2PMessage {
3   // Init:
4   optional AnnouncePeer announce = 1;
5   optional Latency latency = 2;
6   // Blockchain emulation:
7   optional Inventory inv = 3;
8   optional GetData getData = 4;
9   optional Block block = 5;
10  optional Transaction transaction = 6;
11  // DPoS only:
12  optional BlockProposal proposal = 7;
13  optional BlockAck ack = 8;
14 }
```

Listing A.4: P2P Messages

```
// Peer tells other peer who they are (node ID)
message AnnouncePeer {
  required int32 nodeId = 1;
}

// BlockchainPeer informs other BlockchainPeer
// of a new block in their inventory
message Inventory {
  required int32 sender = 1;
  required int32 id = 2;
}

// BlockchainPeer requests block
// with the given ID from other BlockchainPeer
message GetData {
  required int32 sender = 1;
  required int32 id = 2;
}

// Gossiped transaction
message Transaction {
  required int32 txId = 1;
  required int32 txFee = 2;
  required google.protobuf.Timestamp ts = 3;
  optional bytes data = 4;
}

// Gossiped block
message Block {
  required int32 blockId = 1;
  required int32 parentId = 2;
  required int32 creator = 3;
  required google.protobuf.Timestamp ts = 4;
  repeated Transaction transaction = 5;
}

// DPoS BP proposes Block to other BPs
message BlockProposal {
  required Block block = 1;
}

// BP acknowledges BlockProposal of block
// with the given ID
message BlockAck {
  required int32 blockId = 1;
}
```

B. Reproducibility

The following tables describe parameter overwrites of default values introduced in Table 5.1 for each figure presented in this work. Parameter ranges are described either as a comma-separated list or using a *dash* (-) in case of a linear increase/decrease, or if the values can be read directly from the describing figure. The emulator in [141] was configured to use either three or 15 coordinators, depending on the underlying hardware setup (Table B.1). Network parameter N refers to the investigated small, medium and large network topologies introduced in Section 6.1.

Table B.1.: Hardware configurations (Par. = H).

Name	**Cores**	**Description**
Small	10	Fully switched (100 MB/s); Win10, Ryzen 5 2600X, 16GB; Ubuntu 18.04, i5-5200U, 8GB; Ubuntu 20.04, i5-2410M, 6GB
Cluster	58	1 GB/s network; 15 VM cluster; Ubuntu 18.04; 58 GB

Table B.2.: Figure 5.7.

Par.	**Value**
H	Small
B_R	6 - 30
T_R	0
N_N	200, 400, 600
N_L	30 ms
N_T	random; d = 0.05, 0.025, 0.0167

Table B.3.: Figure 5.8a.

Par.	Value
H	Small
B_R	0.0805, 0.001667, 0.01667, 0.00667
T_R	0
B	1000, 100, 500, 250
B_P	false
N_N	1000, 600, 600, 800
N_L	766, 850, 326, 340 ms
N_T	scale-free; m = 3, 4, 6, 20
R	5

Table B.4.: Figure 5.8b.

Par.	Value
H	Small
B_R	1.4, 0.7, 0.35, 0.14, 0.07, 0.035, 0.0233, 0.011667, 0.004667
T_R	0
B	2000 - 200
B_P	false
N_N	600
N_L	273, 273, 280, 296, 333, 403, 476, 693, 1393 ms
N_T	scale-free; m = 5

Table B.5.: Figure 6.1.

Par.	Value
H	Small
B_R	0.1 - 34
C	1, 2
B	500 - 8000
N	Medium
N_L	63ms, 44ms

Table B.6.: Figure 6.2a.

Par.	Value
H	Small
B_R	2, 8, 22
N	Medium
C	1 - 4
B	4000 - 10000

Table B.7.: Figure 6.2b.

Par.	Value
H	Cluster
B_R	0.1 - 16
N	Large, Small
C	1
B	500 - 6000

Table B.8.: Figure 6.3.

Par.	Value
H	Small
B_R	0.1 - 30
N	Medium
B	500 - 10000
B_C	DPoS
D_N	21, 3
D_B	180

Table B.9.: Figure 6.4.

Par.	Value
H	Small
B_R	8, 22
N	Medium
B	8000
B_C	DPoS
D_N	3 - 65
D_B	180

Table B.10.: Figure 6.5a.

Par.	Value
H	Small
B_R	0.1 - 30
B	500 - 8000
B_C	DPoS
N	Medium
D_N	3, 21
D_S	Worst, Best
D_B	180

Table B.11.: Figure 6.5b.

Par.	Value
H	Small
B_R	8, 14, 22
B	4000 - 8000
B_C	DPoS
N	Medium
D_N	21
D_B	10 - 300
D_K	0, 1, 2

Table B.12.: Figure 6.6a.

Par.	Value
H	Cluster
B_R	0.1 - 8
N	Large, Small
B	500 - 6000
C	1, 2

Table B.13.: Figure 6.6b.

Par.	Value
H	Cluster
B_R	0.1 - 8
N	Large, Small
B	500 - 6000
B_C	DPoS

List of Figures

1.1. Google Scholar publications mentioning distributed ledgers and edge computing in title or abstract. . . . 1
1.2. Emulation-based approach showing literature-guided (book), manual (stick-figure) and automated (gear-wheel) activities (rounded rectangles) and corresponding artifacts (rectangles). . . . 3

2.1. Strictness of bandwidth and latency requirements in driving edge applications. Color denotes expected market share by 2025 [3]. . . . 7
2.2. Total time $t_1 + t_2 = t >> t_3$ until the verified completion of an offloaded task in different scenarios. Here, deployment consists of *resource discovery, negotiation* and *service placement*. . . . 9
2.3. Components of the layered DLT architecture. . . . 12
2.4. Gossip of advertisements [8]. . . . 14
2.5. Unsolicited block push. . . . 14
2.6. Differentiating between stale blocks and confirmed stale blocks. . . . 15
2.7. Using GHOST to decrease vulnerability against alternative history attacks by including stale blocks during fork resolution [68]. . . . 17
2.8. Paxos (CFT) [8]. . . . 18
2.9. PBFT [8]. . . . 18
2.10. Pipelined BFT. . . . 19
2.11. Stale and irreversible blocks in DPoS-BFT with 4 block producers. . . . 19

4.1. Devices, roles and activities in a crowdsourced edge computing marketplace using blockchain. . . . 30

5.1. Emulator deployment and technology stack. . . . 34
5.2. Graph and network strategy class diagram. . . . 35
5.3. Emulator class diagram. . . . 37
5.4. PoX (black) and DPoS-BFT (black and blue) coordinator sequence diagrams. 38
5.5. PoX (black) and DPoS-BFT (black and blue) peer sequence diagrams. Messages in parentheses refer to definitions of Appendix A. . . . 40
5.6. Process of adding a block to chain. . . . 42
5.7. Additional stale blocks introduced by high CPU loads resulting from long processing queues. . . . 46
5.8. Comparing the emulator to Ethereum [121], Bitcoin [146], Dogecoin and Litecoin [125] and an existing simulator [125]. . . . 47

6.1. Model functions of latency and stale blocks in medium PoX network. . . 52
6.2. Trading (confirmed) stale blocks for Tx latency in PoX. 54
6.3. Model functions of latency and stale blocks in medium DPoS network. . 55
6.4. Inc. latency between BPs. 55
6.5. Model function of stale blocks created by DPoS during BP hand-offs. . . . 56
6.6. Transaction latencies and asymptotic limits of fitted Weibull curves in small (s.) and large (l.) networks. 57
6.7. Trade-offs in blockchain leading to sharding solutions. 59

List of Tables

2.1. Consensus protocol summary [8]. 13
2.2. Consensus trade-off overview [8]. 21

5.1. Supported emulation parameters and default values. 44
5.2. Selection of emulator results and measurements. 45

6.1. Coefficient of determination and relative mean squared error of model functions. nRMSE is normalized using standard deviation. 51

B.1. Hardware configurations (Par. = H). 71
B.2. Figure 5.7. 71
B.3. Figure 5.8a. 72
B.4. Figure 5.8b. 72
B.5. Figure 6.1. 72
B.6. Figure 6.2a. 72
B.7. Figure 6.2b. 72
B.8. Figure 6.3. 72
B.9. Figure 6.4. 72
B.10. Figure 6.5a. 73
B.11. Figure 6.5b. 73
B.12. Figure 6.6a. 73
B.13. Figure 6.6b. 73

Bibliography

[1] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018. DOI: 10.1109/JIOT.2017.2767608.

[2] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020. DOI: 10.1109/ACCESS.2020.3001277.

[3] N. Mohan, L. Corneo, A. Zavodovski, S. Bayhan, W. Wong, and J. Kangasharju, "Pruning edge research with latency shears," in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, ser. HotNets '20, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 182–189, ISBN: 9781450381451. DOI: 10.1145/3422604.3425943.

[4] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3098–3130, Fourthquarter 2018, ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2841349.

[5] E. Mnif, A. Jarboui, and K. Mouakhar, "How the cryptocurrency market has performed during covid 19? a multifractal analysis," *Finance research letters*, vol. 36, p. 101 647, 2020.

[6] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1508–1532, 2019. DOI: 10.1109/COMST.2019.2894727.

[7] Y. Du, Z. Wang, and V. C. M. Leung, *Blockchain-enabled edge intelligence for iot: Background, emerging trends and open issues*, Feb. 2021. DOI: http://dx.doi.org/10.14288/1.0396085.

[8] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020, ISSN: 2373-745X. DOI: 10.1109/comst.2020.2969706.

[9] J. P. Queralta and T. Westerlund, "Blockchain for mobile edge computing: Consensus mechanisms and scalability," *CoRR*, vol. abs/2006.07578, 2020. arXiv: 2006.07578.

[10] L. Zhou, L. Wang, Y. Sun, and P. Lv, "Beekeeper: A blockchain-based iot system with secure storage and homomorphic computation," *IEEE Access*, vol. 6, pp. 43 472–43 488, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2847632.

[11] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[12] Linux Foundation. "Hyperledger architecture." https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf. (2017), (visited on 06/28/2021).

[13] A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane, "Resource provisioning in edge computing for latency sensitive applications," *IEEE Internet of Things Journal*, pp. 1–1, 2021, ISSN: 2327-4662. DOI: 10.1109/JIOT.2021.3052082.

[14] M. Hosseini, C. M. Angelopoulos, W. Chai, and S. Kündig, "Crowdcloud: A crowdsourced system for cloud infrastructure," *Cluster Computing*, vol. 22, pp. 455–470, 2018.

[15] M. Wang, C. Xu, X. Chen, L. Zhong, Z. Wu, and D. O. Wu, "Bc-mobile device cloud: A blockchain-based decentralized truthful framework for mobile device cloud," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1208–1219, 2021. DOI: 10.1109/TII.2020.2983209.

[16] A. Zavodovski, S. Bayhan, N. Mohan, P. Zhou, W. Wong, and J. Kangasharju, "Decloud: Truthful decentralized double auction for edge clouds," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2019, pp. 2157–2167. DOI: 10.1109/ICDCS.2019.00212.

[17] A. G. Tasiopoulos, O. Ascigil, I. Psaras, and G. Pavlou, "Edge-map: Auction markets for edge resource provisioning," in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Jun. 2018, pp. 14–22. DOI: 10.1109/WoWMoM.2018.8449792.

[18] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[19] A. Zavodovski, N. Mohan, W. Wong, and J. Kangasharju, "Open infrastructure for edge: A distributed ledger outlook," in *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*, Renton, WA: USENIX Association, Jul. 2019.

[20] IDC. "Iot growth demands rethink of long-term storage strategies, says idc." https://www.idc.com/getdoc.jsp?containerId=prAP46737220. (2020), (visited on 06/28/2021).

[21] FNF. "Global cloud computing market projected to reach usd 1025.9 billion by 2026." https://www.fnfresearch.com/news/rise-in-adoption-of-cloud-based-solutions-will. (2020), (visited on 06/28/2021).

[22] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016, ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2579198.

[23] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2745201.

[24] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009, ISSN: 1558-2590. DOI: 10.1109/MPRV.2009.82.

[25] F. Giust, X. Costa-Perez, and A. Reznik, "Multi-access edge computing: An overview of etsi mec isg," *IEEE 5G Tech Focus*, vol. 1, no. 4, p. 4, 2017.

[26] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb. 2018, ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2750180.

[27] Y. Zhao, W. Wang, Y. Li, C. Colman Meixner, M. Tornatore, and J. Zhang, "Edge computing and networking: A survey on infrastructures and applications," *IEEE Access*, vol. 7, pp. 101 213–101 230, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2927538.

[28] V. Cozzolino, A. Y. Ding, and J. Ott, "Fades: Fine-grained edge offloading with unikernels," in *Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems*, ser. HotConNet '17, Los Angeles, CA, USA: Association for Computing Machinery, 2017, pp. 36–41, ISBN: 9781450350587. DOI: 10.1145/3094405.3094412.

[29] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, thirdquarter 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2682318.

[30] A. Mebrek, L. Merghem-Boulahia, and M. Esseghir, "Efficient green solution for a balanced energy consumption and delay in the iot-fog-cloud computing," in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, Oct. 2017, pp. 1–4. DOI: 10.1109/NCA.2017.8171359.

[31] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010, ISSN: 1558-0814. DOI: 10.1109/MC.2010.98.

[32] A. Zavodovski, N. Mohan, S. Bayhan, W. Wong, and J. Kangasharju, "Exec: Elastic extensible edge cloud," in *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking*, ser. EdgeSys '19, Dresden, Germany: Association for Computing Machinery, 2019, pp. 24–29, ISBN: 9781450362757. DOI: 10.1145/3301418.3313941.

[33] F. Holik, "Meeting smart city latency demands with sdn," in *Intelligent Information and Database Systems: Recent Developments*, M. Huk, M. Maleszka, and E. Szczerbicki, Eds. Cham: Springer International Publishing, 2020, pp. 43–54, ISBN: 978-3-030-14132-5. DOI: 10.1007/978-3-030-14132-5_4.

[34] R. Singh, A. Dunna, and P. Gill, "Characterizing the deployment and performance of multi-cdns," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18, Boston, MA, USA: Association for Computing Machinery, 2018, pp. 168–174, ISBN: 9781450356190. DOI: 10.1145/3278532.3278548.

[35] T. Olsson and M. Salo, "Online user survey on current mobile augmented reality applications," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Oct. 2011, pp. 75–84. DOI: 10.1109/ISMAR.2011.6092372.

[36] Z. Cheng, P. Li, J. Wang, and S. Guo, "Just-in-time code offloading for wearable computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 1, pp. 74–83, 2015. DOI: 10.1109/TETC.2014.2387688.

[37] J. M. Cecilia, J.-C. Cano, J. Morales-García, A. Llanes, and B. Imbernón, "Evaluation of clustering algorithms on gpu-based edge computing platforms," *Sensors*, vol. 20, no. 21, 2020, ISSN: 1424-8220. DOI: 10.3390/s20216335.

[38] R. Mahmud, R. Vallakati, A. Mukherjee, P. Ranganathan, and A. Nejadpak, "A survey on smart grid metering infrastructures: Threats and solutions," in *2015 IEEE International Conference on Electro/Information Technology (EIT)*, May 2015, pp. 386–391. DOI: 10.1109/EIT.2015.7293374.

[39] P. Kansal and A. Bose, "Bandwidth and latency requirements for smart transmission grid applications," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1344–1352, Sep. 2012, ISSN: 1949-3061. DOI: 10.1109/TSG.2012.2197229.

[40] A. Anjum, T. Abdullah, M. F. Tariq, Y. Baltaci, and N. Antonopoulos, "Video stream analysis in clouds: An object detection and classification framework for high performance video analytics," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1152–1167, Oct. 2019, ISSN: 2168-7161. DOI: 10.1109/TCC.2016.2517653.

[41] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular adhoc network with fog computing," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 1202–1207. DOI: 10.1109/INM.2015.7140467.

[42] J. Weinman, *Cloudonomics: The business value of cloud computing*. John Wiley and Sons, 2012.

[43] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in cloud computing: A survey," *annals of telecommunications-annales des télécommunications*, vol. 70, no. 7, pp. 289–309, 2015.

[44] Z. Tao, Q. Xia, Z. Hao, C. Li, L. Ma, S. Yi, and Q. Li, "A survey of virtual machine management in edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1482–1499, 2019.

[45] D. Soldani, Y. J. Guo, B. Barani, P. Mogensen, C.-L. I, and S. K. Das, "5g for ultra-reliable low-latency communications," *IEEE Network*, vol. 32, no. 2, pp. 6–7, Mar. 2018, ISSN: 1558-156X. DOI: 10.1109/MNET.2018.8329617.

[46] Howe and Jeff, "The rise of crowdsourcing," *Wired*, vol. 14, Jan. 2006.

[47] S. Kündig, C. M. Angelopoulos, S. R. Kuppannagari, J. Rolim, and V. K. Prasanna, "Crowdsourced edge: A novel networking paradigm for the collaborative community," in *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, May 2020, pp. 474–481. DOI: 10.1109/DCOSS49796.2020.00080.

[48] J. Surowiecki, *The Wisdom of Crowds*. Jan. 2005.

[49] D. Anderson, "Boinc: A system for public-resource computing and storage," in *Fifth IEEE/ACM International Workshop on Grid Computing*, Nov. 2004, pp. 4–10. DOI: 10.1109/GRID.2004.14.

[50] "Boinc." https://boinc.berkeley.edu/. (2021), (visited on 06/28/2021).

[51] "Edge network." https://edge.network/en/. (2021), (visited on 06/28/2021).

[52] "Sonm - decentralized fog computing platform." https://sonm.com/. (2021), (visited on 06/28/2021).

[53] "Dfinity - the internet computer." https://dfinity.org/. (2021), (visited on 06/28/2021).

[54] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2019.12.019.

[55] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *IACR Cryptol. ePrint Arch.*, 2017.

[56] A. Ramachandran and D. M. Kantarcioglu, *Using blockchain and smart contracts for secure data provenance management*, 2017. arXiv: 1709.10000 [cs.CR].

[57] A. Bogner, M. Chanson, and A. Meeuw, "A decentralised sharing app running a smart contract on the ethereum blockchain," in *Proceedings of the 6th International Conference on the Internet of Things*, ser. IoT'16, Stuttgart, Germany: Association for Computing Machinery, 2016, pp. 177–178, ISBN: 9781450348140. DOI: 10.1145/2991561.2998465.

[58] 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," *IEEE Access*, vol. 7, pp. 22 328–22 370, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2896108.

[59] N. Kannengießer, S. Lins, T. Dehling, and A. Sunyaev, "Trade-offs between distributed ledger technology characteristics," *ACM Comput. Surv.*, vol. 53, no. 2, May 2020, ISSN: 0360-0300. DOI: 10.1145/3379463.

[60] F. Saleh, "Blockchain without Waste: Proof-of-Stake," *The Review of Financial Studies*, vol. 34, no. 3, pp. 1156–1190, Jul. 2020, ISSN: 0893-9454. DOI: 10.1093/rfs/hhaa075. eprint: https://academic.oup.com/rfs/article-pdf/34/3/1156/36264599/hhaa075_supplementary_data.pdf.

[61] A. K. Miller and J. LaViola, "Byzantine consensus from moderately-hard puzzles : A model for bitcoin," 2014.

[62] S. King and S. Nadal, *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. peercoin whitepaper*, 2012.

[63] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 3, pp. 34–37, Dec. 2014, ISSN: 0163-5999. DOI: 10.1145/2695533.2695545.

[64] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 475–490. DOI: 10.1109/SP.2014.37.

[65] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, *Proofs of useful work*, Cryptology ePrint Archive, Report 2017/203, https://eprint.iacr.org/2017/203, 2017.

[66] K. Olson, M. Bowman, J. Mitchell, S. Amundson, D. Middleton, and C. Montgomery, "Sawtooth: An introduction," *Linux Foundation*, 2018.

[67] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," *CoRR*, vol. abs/1703.05435, 2017. arXiv: 1703.05435.

[68] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security*, R. Böhme and T. Okamoto, Eds., Springer Berlin Heidelberg, 2015, pp. 507–527, ISBN: 978-3-662-47854-7.

[69] V. Botta, D. Friolo, D. Venturi, and I. Visconti, *Shielded computations in smart contracts overcoming forks*, Cryptology ePrint Archive, Report 2019/891, https://eprint.iacr.org/2019/891, 2019.

[70] D. Marmsoler and L. Eichhorn, "On the impact of architecture design decisions on the quality of blockchain-based applications," *The Knowledge Engineering Review*, vol. 35, e24, 2020. DOI: 10.1017/S0269888920000193.

[71] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[72] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[73] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *CoRR*, vol. abs/1710.09437, 2017. arXiv: 1710.09437.

[74] P.-L. Aublin, S. B. Mokhtar, and V. Quéma, "Rbft: Redundant byzantine fault tolerance," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*, Jul. 2013, pp. 297–306. DOI: 10.1109/ICDCS.2013.53.

[75] 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 *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18, Porto, Portugal: Association for Computing Machinery, 2018, ISBN: 9781450355841. DOI: 10.1145/3190508.3190538.

[76] J. Kreps, "Kafka : A distributed messaging system for log processing," 2011.

[77] D. Huang, X. Ma, and S. Zhang, "Performance analysis of the raft consensus algorithm for private blockchains," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 172–181, 2019.

[78] J. Kwon and E. Buchman, "Cosmos whitepaper," vol. 1, 2021, https://v1.cosmos.network/resources/whitepaper.

[79] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on BFT consensus," vol. abs/1807.04938, 2018. arXiv: 1807.04938.

[80] S. Alqahtani and M. Demirbas, *Bottlenecks in blockchain consensus protocols*, 2021. arXiv: 2103.04234 [cs.DC].

[81] Q. Wang, M. Xu, X. Li, and H. Qian, "Revisiting the fairness and randomness of delegated proof of stake consensus algorithm,"

[82] S. Leonardos, D. Reijsbergen, and G. Piliouras, "Weighted voting on the blockchain: Improving consensus in proof of stake protocols," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 376–384. DOI: 10.1109/BLOC.2019.8751290.

[83] C. Tan and L. Xiong, "Dposb: Delegated proof of stake with node's behavior and borda count," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2020, pp. 1429–1434. DOI: 10.1109/ITOEC49072.2020.9141744.

[84] block.one. "Eos.io technical white paper v2." https://github.com/EOSIO/Documentation. (2018), (visited on 06/28/2021).

[85] J. Monnot and S. Toulouse, "The traveling salesman problem and its variations," *Paradigms of combinatorial optimization: problems and new approaches*, pp. 173–214, 2014.

[86] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 1159, 2016.

[87] IOTA Foundation, "The coordicide," vol. 1, 2020, https://files.iota.org/papers/20200120_Coordicide_WP.pdf.

[88] F. Gräbe, N. Kannengießer, S. Lins, and A. Sunyaev, "Do not be fooled: Toward a holistic comparison of distributed ledger technology designs," in *HICSS*, 2020.

[89] N. Tariq, M. Asim, F. Al-Obeidat, M. Z. Farooqi, T. Baker, M. Hammoudeh, and I. Ghafir, "The security of big data in fog-enabled iot applications including blockchain: A survey," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[90] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the internet of things: Research issues and challenges," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2188–2204, 2019. DOI: 10.1109/JIOT.2018.2882794.

[91] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 78–85, Sep. 2017, ISSN: 1558-1896. DOI: 10.1109/MCOM.2017.1700041.

[92] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1832–1843, Dec. 2017, ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2740569.

[93] N. Herbaut and N. Negru, "A model for collaborative blockchain-based video delivery relying on advanced network services chains," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 70–76, Sep. 2017, ISSN: 1558-1896. DOI: 10.1109/MCOM.2017.1700117.

[94] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, *Blockbench: A framework for analyzing private blockchains*, 2017. arXiv: 1703.04057 [cs.DB].

[95] M. Tahir, M. H. Habaebi, M. Dabbagh, A. Mughees, A. Ahad, and K. I. Ahmed, "A review on application of blockchain in 5g and beyond networks: Taxonomy, field-trials, challenges and opportunities," *IEEE Access*, vol. 8, pp. 115 876–115 904, 2020. DOI: 10.1109/ACCESS.2020.3003020.

[96] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5g and beyond networks: A state of the art survey," *CoRR*, vol. abs/1912.05062, 2019. arXiv: 1912.05062.

[97] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for iot data," in *2017 IEEE International Conference on Web Services (ICWS)*, Jun. 2017, pp. 468–475. DOI: 10.1109/ICWS.2017.54.

[98] U. Javaid, M. N. Aman, and B. Sikdar, "Blockpro: Blockchain based data provenance and integrity for secure iot environments," in *Proceedings of the 1st Workshop on Blockchain-Enabled Networked Sensor Systems*, ser. BlockSys'18, Shenzhen, China: Association for Computing Machinery, 2018, pp. 13–18, ISBN: 9781450360500. DOI: 10.1145/3282278.3282281.

[99] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Edge computing resource management and pricing for mobile blockchain," *ArXiv*, vol. abs/1710.01567, 2017.

[100] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui, "Mneme: A mobile distributed ledger," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul. 2020, pp. 1897–1906. DOI: 10.1109/INFOCOM41043.2020.9155497.

[101] H. Zhao, Y. Zhang, Y. Peng, and R. Xu, "Lightweight backup and efficient recovery scheme for health blockchain keys," in *2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS)*, 2017, pp. 229–234. DOI: 10.1109/ISADS.2017.22.

[102] A. Hahn, R. Singh, C.-C. Liu, and S. Chen, "Smart contract-based campus demonstration of decentralized transactive energy auctions," in *2017 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, 2017, pp. 1–5. DOI: 10.1109/ISGT.2017.8086092.

[103] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: Towards sustainable local energy markets," *Computer Science - Research and Development*, vol. 33, pp. 207–214, 2017.

[104] K. Tanaka, K. Nagakubo, and R. Abe, "Blockchain-based electricity trading with digitalgrid router," in *2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, 2017, pp. 201–202. DOI: 10.1109/ICCE-China.2017.7991065.

[105] J. Xu, S. Wang, B. K. Bhargava, and F. Yang, "A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3538–3547, 2019. DOI: 10.1109/TII.2019.2896965.

[106] M. Nardini, S. Helmer, N. El Ioini, and C. Pahl, "A blockchain-based decentralized electronic marketplace for computing resources," *SN Computer Science*, vol. 1, Aug. 2020. DOI: 10.1007/s42979-020-00243-7.

[107] K. Wright, M. Martinez, U. Chadha, and B. Krishnamachari, "Smartedge: A smart contract for edge computing," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1685–1690. DOI: 10.1109/Cybermatics_2018.2018.00281.

[108] M. Klems, J. Eberhardt, S. Tai, S. Härtlein, S. Buchholz, and A. Tidjani, "Trustless intermediation in blockchain-based decentralized service marketplaces," in *ICSOC*, 2017.

[109] S. Nayak, N. Narendra, A. Shukla, and J. Kempf, "Saranyu: Using smart contracts and blockchain for cloud tenant management," Jul. 2018, pp. 857–861. DOI: 10.1109/CLOUD.2018.00121.

[110] T. Kumar, E. Harjula, M. Ejaz, A. Manzoor, P. Porambage, I. Ahmad, M. Liyanage, A. Braeken, and M. Ylianttila, "Blockedge: Blockchain-edge framework for industrial iot networks," *IEEE Access*, vol. 8, pp. 154 166–154 185, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3017891.

[111] A. Aral, R. B. Uriarte, A. Simonet-Boulogne, and I. Brandic, "Reliability management for blockchain-based decentralized multi-cloud," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, May 2020, pp. 21–30. DOI: 10.1109/CCGrid49817.2020.00-91.

[112] "Iexec - the first decentralized marketplace for cloud resources." https://iex.ec/. (2021), (visited on 06/28/2021).

[113] H. Gao, Z. Ma, S. Luo, and Z. Wang, "Bfr-mpc: A blockchain-based fair and robust multi-party computation scheme," *IEEE Access*, vol. 7, pp. 110 439–110 450, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2934147.

[114] K. Gai, Y. Wu, L. Zhu, Z. Zhang, and M. Qiu, "Differential privacy-based blockchain for industrial internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4156–4165, Jun. 2020, ISSN: 1941-0050. DOI: 10.1109/TII.2019.2948094.

[115] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2082–2091, Dec. 2017, ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2750030.

[116] K. Chard and K. Bubendorfer, "Co-operative resource allocation: Building an open cloud market using shared infrastructure," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 183–195, Jan. 2019, ISSN: 2168-7161. DOI: 10.1109/TCC.2016.2594174.

[117] S. Smetanin, A. Ometov, N. Kannengießer, B. Sturm, M. Komarov, and A. Sunyaev, "Modeling of distributed ledgers: Challenges and future perspectives," in *2020 IEEE 22nd Conference on Business Informatics (CBI)*, vol. 1, Jun. 2020, pp. 162–171. DOI: 10.1109/CBI49978.2020.00025.

[118] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "Simblock: A blockchain network simulator," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 325–329. DOI: 10.1109/INFCOMW.2019.8845253.

[119] R. A. Memon, J. P. Li, and J. Ahmed, "Simulation model for blockchain systems using queuing theory," *Electronics*, vol. 8, no. 2, 2019, ISSN: 2079-9292. DOI: 10.3390/electronics8020234.

[120] X. Wang, A. Al-Mamun, F. Yan, M. Sadoghi, and D. Zhao, *Blocklite: A lightweight emulator for public blockchains*, 2019. arXiv: 1905.02157 [cs.DB].

[121] M. Alharby and A. van Moorsel, "Blocksim: An extensible simulation tool for blockchain systems," *Frontiers in Blockchain*, vol. 3, Jun. 2020, ISSN: 2624-7852. DOI: 10.3389/fbloc.2020.00028.

[122] L. Stoykov, K. Zhang, and H.-A. Jacobsen, "Vibes: Fast blockchain simulations for large-scale peer-to-peer networks: Demo," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*, ser. Middleware '17, Las Vegas, Nevada: Association for Computing Machinery, 2017, pp. 19–20, ISBN: 9781450352017. DOI: 10.1145/3155016.3155020.

[123] C. Faria and M. Correia, "Blocksim: Blockchain simulator," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 439–446. DOI: 10.1109/Blockchain.2019.00067.

[124] R. Yasaweerasinghelage, M. Staples, and I. Weber, "Predicting latency of blockchain-based systems using architectural modelling and simulation," in *2017 IEEE International Conference on Software Architecture (ICSA)*, Apr. 2017, pp. 253–256. DOI: 10.1109/ICSA.2017.22.

[125] 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 *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: Association for Computing Machinery, 2016, pp. 3–16, ISBN: 9781450341394. DOI: 10.1145/2976749.2978341.

[126] S. Benahmed, I. Pidikseev, R. Hussain, J. Lee, S. A. Kazmi, A. Oracevic, and F. Hussain, "A comparative analysis of distributed ledger technologies for smart contract development," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2019, pp. 1–6. DOI: 10.1109/PIMRC.2019.8904256.

[127] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, ser. PODC '19, Toronto ON, Canada: Association for Computing Machinery, 2019, pp. 347–356, ISBN: 9781450362177. DOI: 10.1145/3293611.3331591.

[128] S. Ponnapalli, A. Shah, A. Tai, S. Banerjee, V. Chidambaram, D. Malkhi, and M. Wei, *Rainblock: Faster transaction processing in public blockchains*, 2020. arXiv: 1909.11590 [cs.DC].

[129] E.-O. Blass and F. Kerschbaum, "Strain: A secure auction for blockchains," in *Computer Security*, J. Lopez, J. Zhou, and M. Soriano, Eds., Cham: Springer International Publishing, 2018, pp. 87–110, ISBN: 978-3-319-99073-6.

[130] T. Andersson and A. Erlanson, "Multi-item vickrey–english–dutch auctions," *Games and Economic Behavior*, vol. 81, pp. 116–129, 2013.

[131] E. Ronn, "Np-complete stable matching problems," *Journal of Algorithms*, vol. 11, no. 2, pp. 285–304, 1990.

[132] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction-based vm allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Transactions on Services Computing*, 2019.

[133] "Ipfs powers the distributed web." https://ipfs.io/. (2021), (visited on 06/28/2021).

[134] S. Krejci, M. Sigwart, and S. Schulte, "Blockchain- and ipfs-based data distribution for the internet of things," in *Service-Oriented and Cloud Computing*, A. Brogi, W. Zimmermann, and K. Kritikos, Eds., Cham: Springer International Publishing, 2020, pp. 177–191, ISBN: 978-3-030-44769-4.

[135] S. Muralidharan and H. Ko, "An interplanetary file system (ipfs) based iot framework," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, 2019, pp. 1–2. DOI: 10.1109/ICCE.2019.8662002.

[136] A. Madhavapeddy, T. Leonard, M. Skjegstad, T. Gazagnaire, D. Sheets, D. Scott, R. Mortier, A. Chaudhry, B. Singh, J. Ludlam, J. Crowcroft, and I. Leslie, "Jitsu: Just-in-time summoning of unikernels," May 2015.

[137] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," *CoRR*, vol. abs/1908.04756, 2019. arXiv: 1908.04756.

[138] R. Gennaro, C. Gentry, and B. Parno, *Non-interactive verifiable computing: Outsourcing computation to untrusted workers*, Cryptology ePrint Archive, Report 2009/547, https://eprint.iacr.org/2009/547, 2009.

[139] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 238–252. DOI: 10.1109/SP.2013.47.

[140] I. McGregor, "The relationship between simulation and emulation," in *Proceedings of the Winter Simulation Conference*, vol. 2, Dec. 2002, 1683–1688 vol.2. DOI: 10.1109/WSC.2002.1166451.

[141] L. Eichhorn. "NEBULA: NEtworked Blockchain emULAtor." https://gitlab.lrz.de/cm/2020-leo-masterthesis. (2021), (visited on 06/28/2021).

[142] "Netty project." https://netty.io/. (2021), (visited on 06/28/2021).

[143] "Protocol buffers." https://developers.google.com/protocol-buffers. (2021), (visited on 06/28/2021).

[144] T. Wang, C. Zhao, Q. Yang, S. Zhang, and S. C. Liew, *Ethna: Analyzing the underlying peer-to-peer network of the ethereum blockchain*, 2021. arXiv: 2010.01373 [cs.NI].

[145] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97, Jan. 2002, ISSN: 1539-0756. DOI: 10.1103/revmodphys.74.47.

[146] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*, Sep. 2013, pp. 1–10. DOI: 10.1109/P2P.2013.6688704.

[147] M. Eder, L. Corneo, N. Mohan, A. Zavodovski, S. Bayhan, W. Wong, P. Gunningberg, J. Kangasharju, and J. Ott, *Surrounded by the clouds*, en, Dataset, https://mediatum.ub.tum.de/1593899, 2021. DOI: 10.14459/2020mp1593899.

[148] L. Corneo, M. Eder, N. Mohan, A. Zavodovski, S. Bayhan, W. Wong, P. Gunningberg, J. Kangasharju, and J. Ott, "Surrounded by the clouds: A comprehensive cloud reachability study," Feb. 2021. DOI: 10.1145/3442381.3449854.

[149] C. Ritz, F. Baty, J. C. Streibig, and D. Gerhard, "Dose-response analysis using r," *PLOS ONE*, vol. 10, no. e0146021, 12 2015.

[150] 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 *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 106–125, ISBN: 978-3-662-53357-4.

[151] Ethereum Foundation. "On sharding blockchains." https://eth.wiki/sharding/Sharding-FAQs. (), (visited on 06/28/2021).

[152] B. Xu, D. Luthra, Z. Cole, and N. Blakely, "Eos: An architectural, performance, and economic analysis," 2018.

[153] Ethereum Foundation. "Upgrading ethereum to radical new heights." https://ethereum.org/en/eth2/. (2021), (visited on 06/28/2021).

[154] A. Yakovenko. "Solana: A new architecture for a high performance blockchain." https://solana.com/solana-whitepaper.pdf. (2021), (visited on 06/28/2021).

[155] Elrond Team. "Elrond: A highly scalable public blockchain via adaptive state sharding and secure proof of stake." https://elrond.com/assets/files/elrond-whitepaper.pdf. (2021), (visited on 06/28/2021).