University of Camerino

School of Advanced Studies

Ph.D. in Computer Science and Mathematics

(XXXV Cycle)

Performability evaluation of BFT protocols for DLT

Candidate:
Marco Marcozzi

Supervisor:
Prof. Leonardo Mostarda

Academic Year 2022-2023

Dedicated to Barbora Marcozzi

Contents

Abstract

The importance and interest about distributed systems as known an impulse, following the widespread appearance of distributed ledger technologies, in particular blockchains. Researchers and developers have determined that a main source of performance bottlenecks in the distributed system can be attributed to the consensus protocol employed. In this thesis, it is presented the multi-criteria decision-making process adopted to select a suitable consensus protocol to be used under given conditions and requirements dictated by an arbitrary use scenario for the system under development. Subsequently, two analytical models to assess availability and performability of Byzantine fault-tolerant protocols are discussed and applied. The main results include the measurement of availability when the occurrence of malicious nodes in the network follows an arbitrary probability distribution, and the estimation of performability metrics for Byzantine fault-tolerant protocols. In particular, it emerges that all the evaluated quantities are non-linearly dependent on the defined parameters, e.g. the total number of nodes and the transactions service rate. Notably, the results from the analytical evaluation correctly replicates the trend of empirical studies found in literature, even though the values from the models and from the benchmarks are not consistently matching.

Abbreviations

DLT	Distributed Ledger Technology
IoT	Internet of Things
BFT	Byzantine Fault-Tolerant
DAG	Directed Acyclic Graph
PoX	Proof of X
PoW	Proof of Work
PoS	Proof of Stake
DPoS	Delegated Proof of Stake
PoC	Proof of Capacity
PoA	Proof of Authority
PoI	Proof of Importance
PBFT	Practical Byzantine Fault-Tolerant
RCPA	Ripple Consensus Protocol/Algorithm
SCP	Stellar Consensus Protocol
ABFT	Asynchronous Byzantine Fault-Tolerant
Tx	Transaction
P2P	Peer-to-Peer
RDA	Raw Data Application
SAT	Session Application with Trace
SA	Session Application
USD/US$	United States Dollar
MCDM	Multi-Criteria Decision-Making
TPS	Transactions per second
ASIC	Application-Specific Integrated Circuit
GPU	Graphics Processing Unit
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
IaaS	Infrastructure as a Service
SRN	Stochastic Reward Net
LU	Lower–Upper (decomposition)
SVD	Singular Value Decomposition
PBFTPEM	Practical Byzantine Fault-Tolerant Performability Evaluation Model

Research goals statement

This thesis aims to answer the following research questions:

- Given certain properties characterizing a system (e.g. data throughput, latency, energy consumption, etc.), can we understand which consensus protocol is more suitable in this scenario?
- Once selected a candidate solution, can analytical modeling effectively characterize and predict the qualities of said consensus protocol?

Contribution to the field and novelty

There are some novel contributions to the field reported in this thesis, specifically:

- A MCDM framework to select the most suitable consensus protocols in various scenarios, along with the characterization of metrics associated with DLT platforms and their consensus protocols.
- Novel analytical models for the availability and performability evaluation of BFT protocols for DLT.

Published Articles, Proceedings Papers and Preprints

Articles

[Fil+22] Ernestas Filatovas et al. "A MCDM-based framework for blockchain consensus protocol selection". In: *Expert Systems with Applications* (2022), p. 117609. DOI: `10.1016/j.eswa.2022.117609`

[MMC22] Marco Marcozzi, Leonardo Mostarda, and Diletta Cacciagrano. "Off-chain trading for micro grid systems". In: *Frontiers in Blockchain* 5 (2022). DOI: `10.3389/fbloc.2022.956621`

[Mar+23a] Marco Marcozzi et al. "Availability evaluation of IoT systems with Byzantine fault-tolerance for mission-critical applications". In: *Internet of Things* 23 (2023), p. 100889. DOI: `10.1016/j.iot.2023.100889`

[MM23] Marco Marcozzi and Leonardo Mostarda. "Analytical model for performability evaluation of Practical Byzantine Fault-Tolerant systems". In: *Expert Systems with Applications* (2023), p. 121838. DOI: 10.1016/j.eswa.2023.121838

Proceedings Papers

[Bis+22] Stefano Bistarelli et al. "Blockchain and IoT Integration for Pollutant Emission Control". In: *Advanced Information Networking and Applications: Proceedings of the 36th International Conference on Advanced Information Networking and Applications (AINA-2022), Volume 3*. Springer. 2022, pp. 255–264. DOI: 10.1007/978-3-030-99619-2_25

[Mar+23b] Marco Marcozzi et al. "Availability Model for Byzantine Fault-Tolerant Systems". In: *Advanced Information Networking and Applications: Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA-2023), Volume 1*. Springer. 2023, pp. 31–43. DOI: 10.1007/978-3-031-29056-5_4

Preprints

[MM21] Marco Marcozzi and Leonardo Mostarda. "Quantum consensus: an overview". In: *arXiv preprint arXiv:2101.04192* (2021). DOI: 10.48550/arXiv.2101.04192

Introduction

Distributed computing covers a wide range of established and important applications, such as the Internet, local-area networks (e.g. Ethernet), e-mail services, replicated databases, etc. The undeniable importance of this technological paradigm alone justifies the vast amount of literature on the topic, with articles and books published since the early development of networked computers [TS17].

Blockchains - and in particular Bitcoin [Nak08] - triggered a new golden era for distributed systems, when Bitcoin emerged as a novel technology with a disruptive potential and appealing background philosophy of a currency independent from any centralized authority. However, it was with the proposal and implementation of Ethereum that the ambition of creating a completely decentralized distributed computer sparked [But14]. Since then, over a hundred of different Distributed Ledger Technology (DLT) platforms - of which blockchains are the largest subclass - have been developed and used for several scopes. Some of them have the clear intent to be an alternative currency - a cryptocurrency - to transact tokens between parties in exchange for real-world goods and services [Lee19], while other DLTs are aiming to be a platform for solving transparently and securely some compelling challenges in various sectors, such as supply chains, healthcare, Internet of Things (IoT), manufacturing, etc.

However, DLTs suffer of low performances for real-time applications and a widespread adoption of DLT solutions in most of daily situations is still not realistic. For this, the main challenge is to increase the number of operations that can be conducted in a determined interval (throughput) and the time needed for a single operation to become effectively part of the ledger (transaction latency). There are different solutions and platforms developed with the goal of increasing throughput and to reduce the transaction latency, but, up-to-date, there is not a widely recognised solution for addressing these challenges. This is because the "scalability trilemma" [AB18] poses a limit on three conflicting features of DLTs: performance, decentralization, and security. In other words, any DLT can maximize two out of three features, while the third one has to be sacrificed, e.g. if a DLT maximizes throughput and security, it has to be centralized.

The diversification in the development of DLT platforms, and the introduction of new paradigms for consensus protocols, lead to a difficult classification process in the area of DLTs. In distributed computing, broadly, a consensus protocol comprehends the algorithms and processes followed, in order to achieve agreement on the tasks to be performed by the system.

Besides the academic interest in classification and taxonomies of complex systems, the effort to compare different protocols it is actually noteworthy because it allows a better understanding of the processes underlying consensus protocols. Classification might be helpful in highlighting features and bottlenecks connected with a certain type of consensus protocol, while discriminating differences and strengths [Xia+20].

To understand peculiar features of the consensus protocol underlying a DLT solution is critical because a performance bottleneck determined by the consensus protocol induces disastrously affects on the performance of the whole system. For instance, Bitcoin's consensus protocol allows (theoretically) unlimited miners to participate in the mining of a new block, while ensuring a high security, hence the throughput has to be low. It becomes, therefore, of vital importance to determine the intended features of the consensus protocol when developing a DLT solution.

Indeed, performance and characteristics of a DLT platform are connected with the features of the underlying consensus protocol [Xia+20]. For instance, different consensus protocols have distinct levels of security, or they can allow a restricted number of participants in the consensus process to prevent performance bottlenecks: e.g., a Proof of Work (PoW) system, like Bitcoin, because of its intrinsic properties, would not be fitting an application requiring seemingly real-time response, while a Byzantine Fault-Tolerant (BFT) consensus protocol could possibly be a good fit. In this context, the national project HD3FLAB (in which my work is inserted) required the implementation of a blockchain solution to act as multipurpose platform for different stakeholders, e.g. public entities, universities, and local enterprises.

This thesis, and related published works, covers the process behind the selection and characterization of the optimal consensus protocol, which is matching the characteristics of the case study selected (see section 4.5), among the ones covered by the national project. In particular, it is here reported a Multi-Criteria Decision-Making (MCDM) framework - including a methodology and associated data -, that has be used to rank DLT platforms according to certain criteria and constrains. Among the best matches for the case-study, there are DLT platforms using a BFT consensus protocol. By any means, decision-makers opted to implement the blockchain infrastructure backed by a BFT-based consensus protocol.

After choosing a class of algorithms, it was necessary to understand the characteristics and limits of this family of protocols, notably for what concerns the performability. In this thesis, when mentioning *"performability"*, it is intended the characteristic in terms of performance (e.g., throughput and latency) and availability/reliability of the studied systems. According to the context, *"performability"* might refer to a subgroup of the total metrics of interest, because, for example, the applied methods or results lack the metrics associated with a specific quality.

To evaluate performability of a system, there are different approaches, i.e. simulations, benchmarks and analytical models. In this thesis analytical methods are explored to assess performability for BFT-based protocols. Benchmarks were also considered in order to evaluate some performability metrics, i.e. throughput and transaction latency, especially as validation for the analytical models. However, the challenging implementation of a benchmark in a geographically distributed environment precluded the continuation of this approach. It is presented in this thesis (see subsection 2.2.2) a benchmark of a blockchain built on a local machine, but the results obtainable by this method are limited and capped by the demanding execution of the process.

Although, it is clear that analytical models have their own challenges. Indeed, to evaluate the performance of consensus protocols through analytical models requires the non-trivial endeavour of developing a model for each protocol of interest. Although this conveys the impression of being utterly challenging, the study of systems performability by means of analytical models is actually a powerful method to better understand the protocol itself and to even generalize some aspect common to different protocols.

The rest of the thesis is structured as follows: chapter 1 presents information about DLT (section 1.1), and its main schemes (subsection 1.1.1 and subsection 1.1.2), along with consensus protocols (section 1.2), in particular the three most important consensus families, i.e. PoW (subsection 1.2.1), Proof of Stake (PoS) (subsection 1.2.2), and BFT (subsection 1.2.3); section 1.3 gives key concepts in the field of performability evaluation, with a focus on analytical modeling (subsection 1.3.1); in chapter 2 the literature and works related to this thesis is reviewed; chapter 3 presents the methodologies and definitions applied to carry the researches presented in this thesis; chapter 4 reports the protocol selection framework based on MCDM techniques (section 4.1), together with the criteria utilized (section 4.2), the choice of their weights (section 4.3), and the process of data collection (section 4.4); chapter 5 reports the analytical models developed to evaluate availability (section 5.1) and performability (section 5.2) of BFT systems; results are shown in chapter 6, along with a sensitivity analysis of the results coming from the performability model (subsection 6.2.1); lastly, chapter 6.2.1 summarizes the finding exposed in this thesis and possible further developments in these fields.

Chapter 1

Background knowledge

In this chapter, it is presented the background knowledge necessary to attain a basic understanding of the concepts discussed in the following chapters. The topics considered include: the characteristics of DLT, along with the features of the two types of DLT, blockchain and Directed Acyclic Graph (DAG); the consensus protocols used in DLT and their components; the methods used to assess performability and the introduction of the mathematical framework in which analytical models are developed.

1.1 Distributed Ledger Technologies

A Distributed Ledger is a consistent and immutable collection of data replicated across different networked computers, these dislocated in different geographical places [Rau+18]. Similarly to databases, DLTs store data and make it accessible to any client connected to the network. However, even replicated (or geographically distributed) databases are managed by a centralised entity, normally the owner of the database. This leads to a single point-of-failure, since if the management authority is compromised, the all system fails. For DLTs, instead, there is no need for a central authority. A DLT relies on a consensus protocol that ensures its consistency, liveness and safety. Assuming that the consensus protocol is well designed and reliable, said distributed protocol is provably difficult to compromise, since it is spread across different machines controlled and managed by different entities. Therefore the idea that, in a very simplistic way, DLTs can be assimilated to databases is conceptually incorrect, since this deep, crucial difference between DLT and database [Cho+18].

There is not a solid consensus for what concerns some important aspects regarding DLT, e.g. protocol classification, mostly because of the relevant amount of literature wrote on the topic and the fast-paced development of this technology [ØUJ17; Mau+17; Suc+18; EP18; Cho+19; Sun20; Kan+20]. Undeniably, however, there are definitions on which literature agrees, even though there is still some confusion, especially from non-specialist sources. This is the case of the definition of the security (or trust) levels and the grade of accessibility in DLTs.

A DLT is said to be *permissionless* when any entity with access to the Internet can join the network, thus operating on the DLT and participating in the management of the DLT itself.

A *permissioned* DLT, instead, allows only to a restricted pool of members to act on the ledger.

In this definition, *permissionless* and *permissioned* are not stating anything about the accessibility of data to external users, only the level of security/trust to operate on the DLT.

In fact, the accessibility to data is related to the concepts of *public*, *private*, and *consortium* DLTs. These terms intertwine with the concepts of *permissioned* and *permissionless* environments, meaning that *public*, *private*, and *consortium* can be categories for which some assumptions also on the accessibility of the network are induced.

Public DLTs are the most transparent type of DLT. While it is possible to have a *permissioned* environment, *public* DLTs are mostly *permissionless* [Pau+19]. Data in a *public* DLT can be accessed by anybody connected to the internet, its transactions can be fully followed, and they guarantee pseudo-anonymity for its users (full anonymity is not possible, since it is feasible to reconstruct the identity of the owner of a certain wallet [KL18; And+21]).

Private DLTs, on the contrary, are closed *permissioned* environments, where generally the identities and confidential information about users are handled. Since no information should leak from the network, *private* DLTs have very strict access policies and data is kept private [Pau+19].

Consortium DLTs are hybrid solutions between the previous two types. Participation in *consortium* DLT is *permissioned* (access is granted by applying to the administrators of the network), while access to data can be restricted to an authorized subgroup, to the whole network, or data can be publicly available. Indeed, the main requirement for a *consortium* DLT is to be *permissioned.* As an example, this kind of DLT can be used for a network where the same service is desired among competitors, which are willing to share data to obtain said service, but only to a selected subgroup of members in the network [Che+20].

Lastly, the boundary between whether a platform is a *consortium* or a *private* DLT is, in some cases, unclear. However, an effective criterion to discern between *private/consortium* and *public* DLT is that the latter is never built on a dedicated infrastructure, therefore if a dedicated physical network infrastructure is present, the associated DLT can not be considered public [Che+20].

A few cases to better illustrate those concepts: Bitcoin [Nak08] and Ethereum [But14] are *public permissionless* DLTs; platforms developed for companies or financial institutes are *private permissioned* DLTs; Ripple [CM18] and EOS [a] are *public permissioned* DLTs, since data is publicly accessible, but the right to write on the blockchain is not attainable freely at any time (however those are not *consortium* DLT, because the permission to be part of the consensus can be obtained through a publicly accessible process).

1.1.1 Blockchains

Undoubtedly, among all the DLTs, blockchain is the technology that entered the most the common knowledge and widespread interest [Zhe+18]. Of course, in public opinion, blockchain is identified with Bitcoin, even though there are many other platforms and more sophisticated systems, like Ethereum, just to mention one.

By any means, as depicted schematically in Figure 1.1, a very simple definition of blockchain may be *"a DLT in which transactions (or data) is organised in blocks, each linked to the previous one by a cryptographic hash".*

[a]`https://eos.io/` [Accessed November 15, 2022]

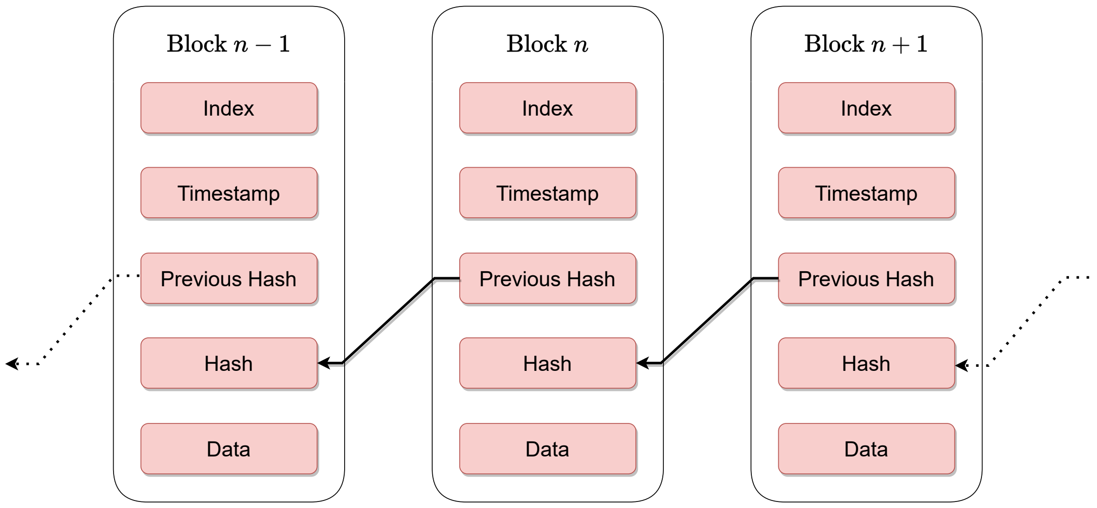


Figure 1.1: Representation of a blockchain. Each block contains an index, the timestamp, the hash of the previous block, the hash computed for the current block, and the list of transactions.

By this interpretation, we can conclude that qualifing a DLT as a blockchain is stating the data structure and some embedded cryptographic security of the DLT itself.

Indeed, the main security feature of a blockchain is that it is computationally unfeasible to modify an already added block, since an attacker would need to tamper backwards all the blocks until the one that it wants to modify. To do so, it needs the computational power (or somehow the control) of the majority of the network.

However theoretically unfeasible, some attacks on blockachains took place, as an example the emblematic attack on Ethereum leading to the hard fork that created Ethereum and Ethereum Classic [b].

1.1.2 Directed Acyclic Graphs

Beyond blockchain, there is at least one other scheme that is used to build DLT systems.

DAG [BŽ18] is a mathematical object - a graph - composed by edges and vertices, with each edge directed from one vertex to another, such that there is no

[b]`https://www.gemini.com/cryptopedia/the-dao-hack-makerdao` [Accessed November 17, 2022]

closed loop in the graph. In other words, since vertices in a DAG can be ordered, there is no edge from the latter vertex to the previous one.

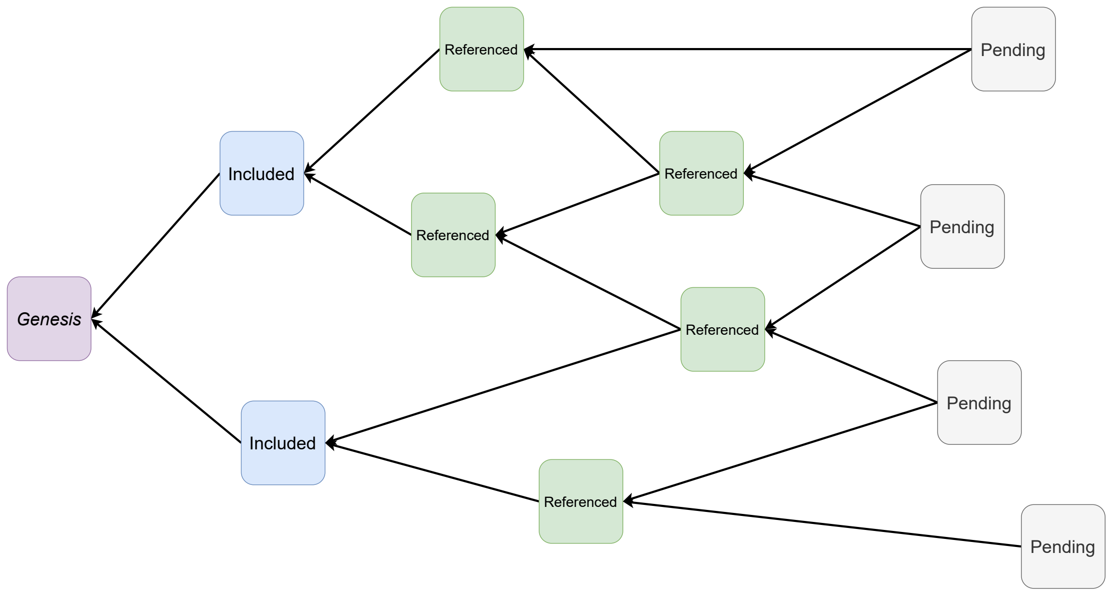


Figure 1.2: An example of DAG based on the Tangle. “Genesis” is the first generated transaction of the DAG, the one to which all the newly added ones will be point at, eventually. “Included” are transactions that have been pointed to by enough other transactions, such that the random walk from any point includes definitively the “Included” transactions. “Referenced” transactions have been tipped from other transactions, but not enough to be considered “Included”. “Pending” transactions have not been chosen by any transaction as reference.

The most visible difference between blockchain and DAG is that data in DAGs is not stored in blocks of transactions (thus a DAG is not a blockchain). In a DAG, data storage is composed of single transactions, which are incidentally the vertices of the DAG. In the action of broadcasting a new transaction to the network, the transmitting node needs to specify some previously added transactions, as reference, to attach to. This action determines the formation of a new vertex (the transaction) and new edges in the DAG, those that are connecting the submitted transaction and already existing ones. This means that, differently from a blockchain - where subsequent blocks assume the form of a line of blocks chained by cryptographic means -, DAG is evolving quite chaotically, almost resembling a tangle of vertices and edges. From this image, derives the name of one famous example of consensus protocol used in a DAG, the Tangle [Pop18].

In Tangle, as shown in Figure 1.2, to decide whether a vertex is considered “Included" (hence the associated transaction is regarded as legitimate and definitive) a rationale based on Markov Chain Monte Carlo algorithms is used. The idea is that, by assigning a weight to each vertex, if any weighted random walk is passing definitively through a certain vertex, this is considered therefore "Included". Point is that the weight of a vertex reflects the cumulative weights of the vertices referring to it, hence if enough transactions are using as a reference a certain vertex, this is included in the memory of a overwhelming portion of the network.

Although the intended scope of IOTA (the DLT implementing the Tangle) is to provide a lightweight framework to enable IoT devices to take part in the DLT economy, there are downside to this approach (which are getting addressed in Tangle 2.0 [Mül+22]), such as: a high flow of transactions is needed to ensure safety, and externally issued checkpoints (by the IOTA foundation) are used to crystallize the state of the network at regular times.

1.2 Consensus protocols for DLT

In distributed computing, when dealing with concurrent processes, state machine replication, multi-agent systems and related paradigms, the goal is to achieve an overall system reliability, despite the presence of a number of faulty processes. Roughly, there are two main properties that a reliable distributed system has to satisfy: liveness (summarised as *"something good will eventually occur"*) and safety (summarised as *"something bad will never happen"*).

Consensus protocols are fundamental parts of DLTs. The consensus protocol determines and maintains the state of the distributed ledger, while ensuring safety and liveness. Since the appearance of Bitcoin [Nak08], both academia and industry are trying to develop at a high pace more sophisticated, fast and secure consensus protocols to implement in their DLT platforms. In this context, designers and developers are facing a problem - the so called "impossible triangle" - in which a compromise between Decentralisation, Security and Scalability has to be made [AB18].
This trilemma was, in a different form, already known for general distributed data stores, but the advent of Ethereum [But14; Woo14] enhanced the interest.
The problem is so important and widely discussed that there are plenty of sur-

veys, literature reviews, taxonomies, etc. on the topic, as well as blog threads and social media posts [Cro+16; Cac17; AA19; Ban+19; Bou21; Xia+20; NL20; ZL20].

By any means, DLTs implementing a PoW consensus protocol (see subsection 1.2.1) - believed quite adequate for peer-to-peer payment and cryptocurrency transfer - is instead considered slow for actual capillary applications: retail shop payments, IoT, real-time contracting, etc.

Moreover, studies evaluating the total electric consumption to maintain the Bitcoin network found out that the network consumes as much electric power as for the needs of a small-medium sized state [ZKC20; GKS20]: this because of the intensive computation needed to solve the cryptographic puzzle in PoW systems (hence, also, the necessity of specific hardware optimized to solve such problems).

However, all the above mentioned issues opened the path to the development of different paradigms and protocols. For example, Proof of Capacity (PoC) - also know as Proof of Space - is a protocol related to PoW, but with a different perspective when solving the computational puzzle [Dzi+13]. While in a PoW protocol, miners need to compute "on the run" the hash for the block they want to propose (hence using GPU, ASIC, etc.), in PoC a miner computes ahead a list of possible hashes and stores them in its hard-drive. When it comes to submit a new block, then, the miner search in its cache a suitable hash and it submits the block. The claim of PoC developers is that their algorithm is faster and less computationally expensive compared with PoW.

When talking about scalability for consensus protocols, the most discussed substitute of PoW is represented by the Proof-of-Stake (PoS) [Li+17]. Generally, in PoS (see subsection 1.2.2), any user of the network owning tokens are eligible to participate in consensus - therefore minting a new block of transactions - if they prove some degree of commitment to the blockchain itself (i.e. investment of wealth). Therefore, a node tends to act honestly in such an environment, since its wealth is at stake, so that it would be a damage to its own interest if the network would be compromised. Because of the little amount of computation to be performed, PoS offers an advantage respect to PoW for what concerns time and energy consumption. Nonetheless the performance advantages, researchers are attentive about the possible security vulnerabilities inherently connected with

PoS protocols, and they are concerned also about a centralisation propensity in PoS systems, especially with regard to the process of stakes delegation.

In the framework of PoS consensus protocols, a wide amount of alternatives exist when looking at the actual implementation of systems, but they rely mostly on two different approaches: randomly elected sequence of block proposers and randomly selected consensus committee/group [Che+18; Kia+17; BPS16]. In the first case, finalisation is obtained, generally, in a way similar to the longest chain rule, while, in the latter, finalisation is instantly achieved if a BFT protocol is used.

Ideas similar to PoS have been developed, where at stake is not something as “quantifiable” as tokens, but some other relevant asset. In Proof of Authority (PoA) [HSS20], actually, it is the real identity of a validator to be at stake: the idea is that the difficult process to become a validator, the investment of financial resources and of personal identity, is a deterrent to misbehave as validator in the consensus protocol.

Likewise, Proof of Importance (PoI) [c] uses a reputation system - in which transacting with other users is an incentivized behaviour that increases the node’s reputation score - alongside with a time-delayed staking process (your tokens at stake will be vested over several days) to determine the node’s status of validator.

Proposing a completely different approach, Proof of Elapsed Time [Ril18] is a protocol relying on a random selection rule: to each node participating in consensus the protocol assigns a random timer, then the node for which the timer expires first becomes the leader and, thus, it can propose a new block.

Over the years, an already established idea captured the interest of researchers and developers: because of its resilience to crush and malicious behaviour, implementations of BFT [LSP82] protocols seemed to be suitable to develop DLT systems (see subsection 1.2.3). In particular, Practical BFT (PBFT) [CL+99] entered the DLT scenario in a delegated form, like in Tendermint [Buc16; BKM18]. Moreover, with related concepts, Ripple Consensus Protocol/Algorithm (RCPA) [CM18] and Stellar Consensus Protocol (SCP) [Maz16] implements the so called Federated Byzantine Agreement. Roughly, the process is split on different phases with

[c]NEM Whitepaper. `https://whitepaper.io/document/583/nem-whitepaper` [Accessed March 30, 2023]

interaction (i.e. voting cast) between peers (sub-groups of peers in SCP) until consensus is reached. All the aforementioned protocols rely on a (at least weak) synchronicity in the system. If a protocol, instead, does not assume any synchronization between nodes, this is called an asynchronous protocol. Although it is impossible, in most cases, to achieve deterministic agreement in an asynchronous setting [FLP85], there are some proposals aiming to implement Asynchronous BFT (ABFT).

HoneyBadgerBFT [Mil+16] is the first proposed practical asynchronous BFT protocol, leveraging on randomized agreement. Indeed, it has been proved that a certain level or randomness in the system overcomes the need for synchronization in the network. Similarly, BEAT [DRZ18] was proposed as an improvement of HoneyBadgerBFT, focusing on being a flexible and versatile protocol, optimized for latency, throughput, and network scalability (in terms of the number of servers).

While both HoneyBadgerBFT and BEAT are assuming that data is stored in batches (or blocks), Hashgraph [BL20] is an ABFT protocol that is built on a DAG and it does not need for blocks of transactions. Hashgraph is optimising the throughput and latency by executing simultaneously broadcasting and voting, without further messages exchanged. As specified in subsection 1.1.2, transaction-based DAGs are a very peculiar type of consensus protocols [Pop18; Roc+19]. Indeed, systems implemented by using a Tx-based DAG approach actually can be not considered blockchains, because there are no blocks of transactions. In Tx-based DAGs, transactions are submitted to known peers creating an edge between a freshly proposed transaction and transactions already known. To finalise a transaction, a weighted random walk is performed: when the probability of passing a thought a certain path from a specific transaction to the genesis node is overwhelming, those transactions are accepted as valid by the participants.

There are known disadvantages in the implementations of a DAG. For example, in the first implementation of the Tangle [Pop18], the drawback was that, to ensure security, a very high number of transactions and participants was needed. When this was not guaranteed, some centralised intervention was needed, i.e., checkpoints issued by some super-validator users.

The new version of the Tangle, Tangle 2.0 [Mül+22], claims to implement a generalization of the Nakamoto consensus, where there longest chain rule is substitute by the heaviest DAG and PoW with stake- or reputation-based system. By *heaviest DAG*, authors mean the branch of the DAG where the weights associated with the random walk are the heaviest. Tangle 2.0 is a leaderless consensus protocol and it does not need of validators and miners. The total ordering is not necessary and it is not guaranteed, but this asynchronous setting requires the implementation of a common random coin to ensure liveness and safety for the asynchronous communication model.

1.2.1 Proof of Work

PoW is a cryptographic protocol that has gained widespread recognition for its use in blockchain technology. The protocol was initially developed to prevent denial-of-service attacks and spam in systems connected to the Internet [DN92; JJ99]. As shown in Figure 1.3, PoW requires clients to solve a computationally

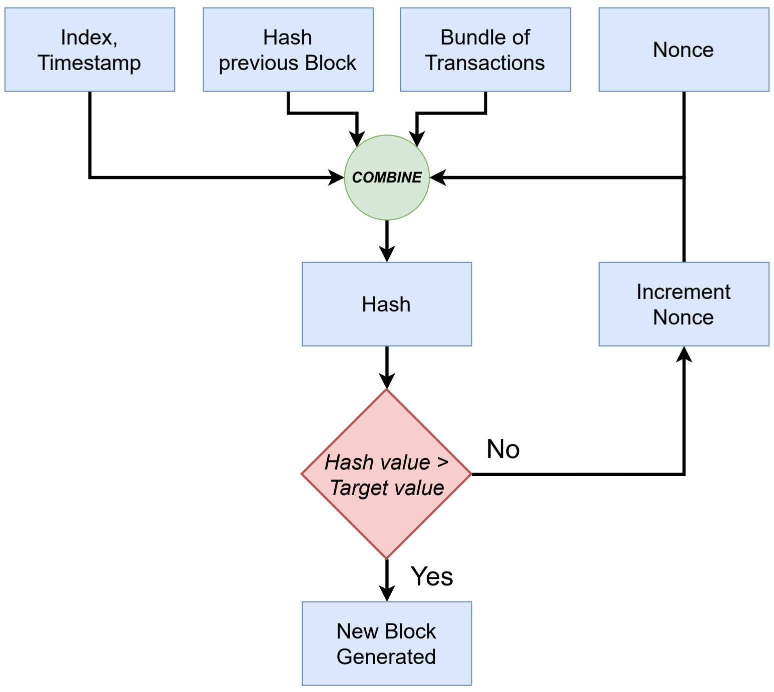


Figure 1.3: Flowchart of PoW mining process. Generated blocks, however, may be reverted or they might not be included in the blockchain. Mined blocks need a certain amount of confirmations (blocks descending from the one examined) before being considered finalized.

challenging task in order to access a service, while the service provider needs a minimal amount of computational power to verify the result. With the advent of Bitcoin, PoW has become a widely known and utilized protocol in blockchain technology. In a blockchain network using PoW, miners must solve a cryptographic puzzle, typically a partial hash-function inversion, to add a new block to the network. The finalization of a block, i.e. its definitive inclusion in the blockchain, is achieved through the application of consensus rules. In Bitcoin, for example, since the concurrent creation of blocks may lead to forks in the blockchain (parallel legit chains), the Nakamoto consensus protocol's longest-chain rule assures that, to resolve conflicting chain forks, the accepted main chain is the one in which more work (the largest number of blocks) has been created. Hence, discarding any other parallel (forked) chain generated in the same time span. This process requires a significant amount of time for a block of transactions to be considered confirmed, due to the stringent security constraints imposed by the PoW protocol.

Nonetheless, the significance of PoW in distributed systems and networking lies in its ability to secure and validate transactions in a decentralized manner. An effective way to summarize the contribution of PoW to the field of consensus protocols is to list its the benefits and drawbacks [Ger+16].

Advantages:

- *Decentralization.* PoW ensures that no single entity has control over the network, as anyone with the necessary computing power can participate in the validation process. This provides a level of transparency and fairness that is difficult to achieve in centralized systems.

- *Security.* PoW requires computational effort to validate transactions and solve complex mathematical puzzles, making it difficult for malicious actors to compromise the network. This protects against 51% attacks and other security threats, ensuring the integrity of the network.

Disadvantages:

- *Energy consumption.* PoW is computationally intensive, requiring large amounts of energy to validate transactions. This has resulted in concerns over the carbon footprint of PoW networks, as well as the potential for centralization of mining power.

- *Scalability.* As more users participate in the network, the validation process becomes increasingly complex and slower. This limits the ability of PoW networks to scale to meet growing demand, making it a challenge to support high-volume applications.

- *Cost.* Participating in PoW networks requires significant computing power and energy resources, which can be cost-prohibitive for many users. This creates barriers to entry the consensus process and it can limit the growth of the network.

In conclusion, PoW offers significant benefits for immutable records of transactions, providing secure decentralized networks. However, its high energy consumption and scalability challenges make it unsuitable for some use cases, particularly those requiring high-volume transactions or low-cost validation.

1.2.2 Proof of Stake

Along with the scalability issues, the extensive computational demands of PoW-based blockchains motivated researchers and developers to investigate alternative solutions to implement DLT systems. The most well-known and widely adopted substitutes to PoW are variations of the PoS protocol [Li+17; Ngu+19]. In PoS (see Figure 1.4), any node can possibly participate in consensus, although stakeholders are required to prove some sort of commitment to the network, notably wealth (i.e. tokens) or personal identity (i.e. PoA). The premise is that a node has a vested interest in maintaining the integrity of the blockchain, as its own wealth is at risk. PoS has the advantage of requiring less computational effort, thus reducing the time and energy consumption compared to PoW. However, there are known vulnerabilities and attacks in PoS systems, such as the nothing-at-stake problem, where a node proposes blocks on multiple competing chains, and a reduced degree of decentralization in the network, where wealth tends to be concentrated among the wealthiest participants. To address these challenges, various schemes have been developed based on the PoS concept, each with unique implementation of the consensus algorithm and methods of selecting block issuers. These variations reflect the ongoing efforts to achieve consensus in blockchain systems.

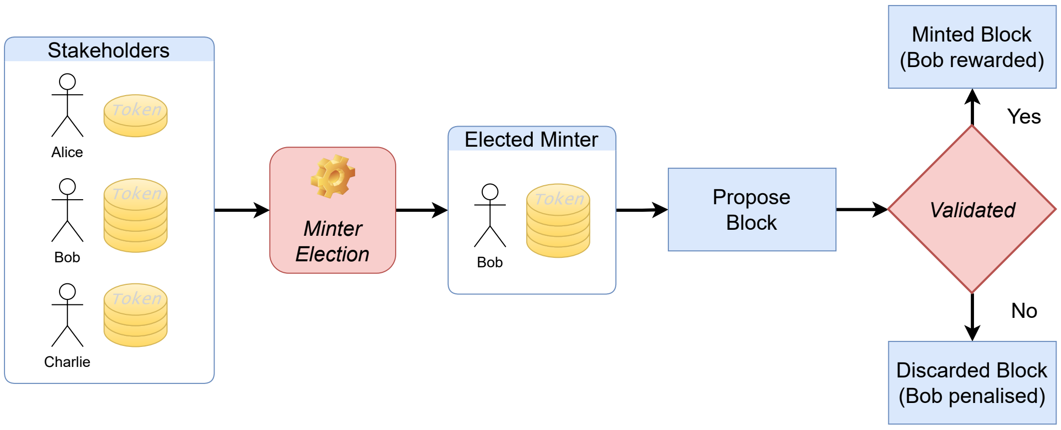


Figure 1.4: PoS flowchart. A minter is elected among the stakeholder by means of a randomized algorithm having as input the tokens at stake for each stakeholder. The winner of the draft (minter) is proposing a block, which is check by validators. If the block is validated, the minter receives tokens as a reward, otherwise its tokens at stake are slashed.

Chain-based PoS

The selection of block minters in a chain-based PoS [Xia+20] system is performed through the utilization of a pseudo-random algorithm, which assigns the right to attempt a not-so-computationally-expensive PoW, with a probability proportional to the stake held by the stakeholders. This approach incentivizes stakeholders to maintain a significant stake in the system. In this model, the target hash value for the proof-of-work process is proportional to the stake value. As a result, the higher the stake value, the lower the attempts required to find a suitable hash. The finalization of blocks in this PoS system is achieved through a process similar to the Nakamoto consensus mechanism.

Committee-based PoS

In the implementation of a committee-based PoS [Xia+20] consensus mechanism, a subset of minters is selected through a random election process, with stake serving as a weighting factor. The committee subsequently initiates a multi-round block generation protocol, in which block proposals are made by leaders in accordance with a pseudo-random sequence, thus reviewed and validated by the committee members.

BFT-based PoS

In alignment with other PoS protocols, a proposer is elected through a voting round, commonly utilizing staked tokens as voting weight. This initiates the process of generating a new block, which undergoes a multi-round BFT consensus process. During this process, a limited number of validators exchange messages to verify the proposed block, ultimately leading to its validation and immediate finalization within the blockchain.

As a case in point, the Casper FFG [BG17] consensus protocol employed in Ethereum 2.0 incorporates a BFT mechanism vote on issuing checkpoints. The consensus process is divided into epochs and during each epoch voting on ordinary blocks is structured as a chain-based PoS. However, on any occasion validators can cast votes to designate a block as a checkpoint. This operation is structured as a BFT round. All blocks on a chain falling between two established, validated checkpoints are immediately confirmed.

Delegated PoS (DPoS)

Delegated PoS (DPoS) [SR20; Zha+22] consensus mechanism endeavors to mitigate the issue of wealth centralization by enabling stakeholders to delegate their voting power (stake) to selected, wealthier delegates participating in the consensus process. By limiting the number of active validators, DPoS aims to strike a balance between mitigating centralization and ensuring scalability. While a limited consensus group surely provide a scalable platform in terms of processed transactions, the limited size of the committee appears to be in contrast with the effort of decentralize the network. Nonetheless, the possibility of moving votes between delegates is the control mechanism that delegators have in order to ensure a fair functioning of the network. For what concerns the concentration of wealth, in this model, delegation results in the redistribution of rewards generated by block production also among the small stakeholders, thereby reducing centralization.

It is important to note that the specific block proposal and validation methodology is not explicitly defined within the DPoS consensus framework. As this aspect is not central to the fundamental concept of DPoS, any of the previously described block production mechanisms may be employed as the implementation strategy.

1.2.3 Byzantine Fault-Tolerance

The BFT paradigm, represented schematically in Figure 1.5, is a well-established approach for finalizing tasks in computer systems, particularly in distributed systems, that addresses the issue of faulty and malicious participants [LSP82]. For

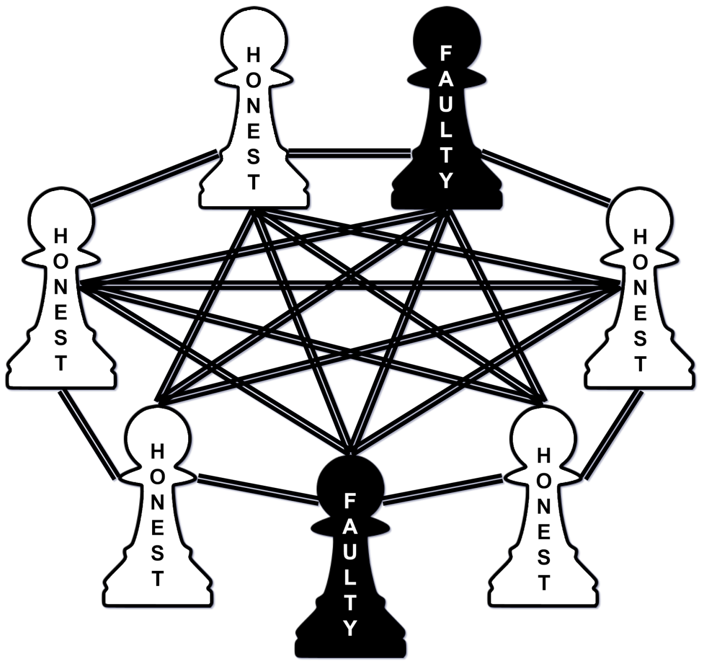


Figure 1.5: Representation of a BFT scheme with 7 nodes. The white and black nodes represent non-Byzantine and Byzantine nodes respectively.

instance, a Byzantine fault occurs when a node is acting maliciously in the network, e.g. sending contradictory messages to separate servers or being unresponsive. However, a node might also not act maliciously in the network, and yet be unresponsive due to crash or connection failures. In both cases, the system may fail to reach consensus.

In an implementation of BFT consensus protocol with $N \geq 4$ servers exchanging unsigned messages (with unsigned messages, if $N < 4$, the problem does not have a solution [LSP82]), quorum (the minimum amount of committing messages to achieve consensus) is reached when the number of honest responsive nodes h is

$$h > 2N/3, \tag{1.1}$$

therefore, a distributed system, in which unsigned messages are exchanged, can handle up to f Byzantine faults, such that

$$f < N/3. \tag{1.2}$$

Multiple implementations of the BFT paradigm have been proposed, including PBFT [CL+99], which is commonly used to implement blockchain solutions. In PBFT nodes serve as a consensus committee and allow for tolerance of up to 1/3 of faulty participants with high throughput and low latency. However, the exponential increase in complexity with the number of participants can limit the scalability of PBFT, making it suitable for only permissioned environments with a low number of nodes. Figure 1.6 represents schematically the communication scheme in PBFT: for each round, there are three phases (proposal, prepare, commit), plus an additional round to elect a new proposer/leader. Proposal is the phase where the proposer/leader sends the messagge to be committed to the other validators/nodes, then, in prepare, each node verifies the correctness of the message and eventually sends a confirmation message to all the other nodes, lastly, if enough confirmation messages have been received, nodes send a receipt of the committed message.

HotStuff [Yin+18], an improvement on PBFT, has been proposed and implemented. Like PBFT, HotStuff is a voting-based protocol that uses a leader node to collect and broadcast votes, but it utilizes a more efficient voting process and a flexible leader rotation scheme to reduce communication overhead and the risk of leader failure.

PBFT and its derived protocols operate under the assumption of eventual synchrony among nodes, which requires each node to collect messages until a timeout expires in order to reach agreement on a task. This assumption can pose a limitation in scenarios where the network is prone to frequent outages, particularly in highly overloaded networks. In contrast, Asynchronous BFT (ABFT) protocols,

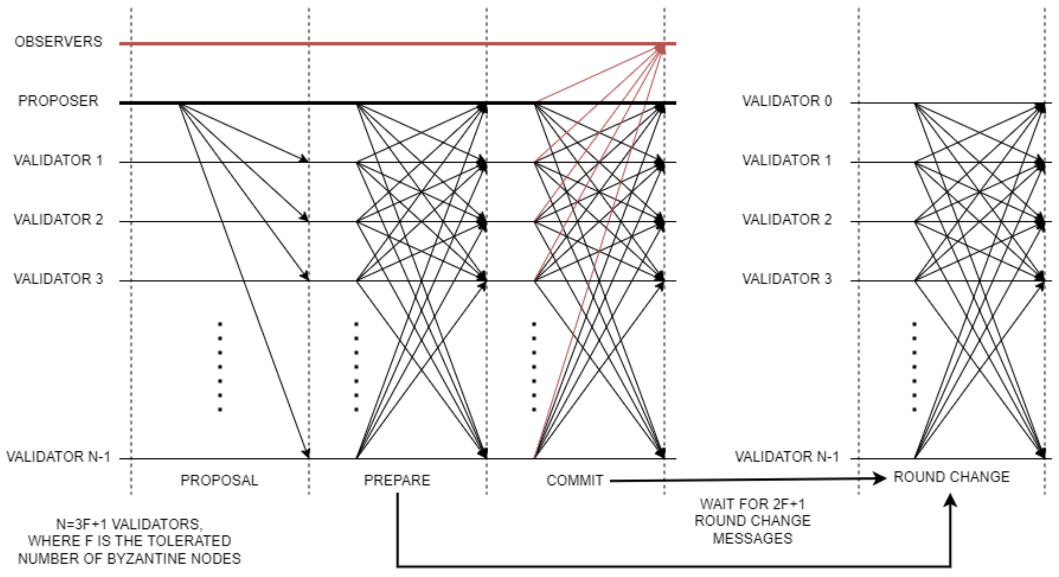


Figure 1.6: A schematic representation of the PBFT consensus protocol's workflow.

such as HoneyBadgerBFT [Mil+16], BEAT [DRZ18], and Hedera [BHM19], do not require any timing assumptions. These protocols offer alternative solutions to the Byzantine agreement problem in an asynchronous network environment, providing a potential solution for networks that are prone to frequent outages.

1.3 Performability evaluation

Performability evaluation is a crucial aspect in the study of any system, as it provides valuable insights into the system's efficiency, effectiveness, and reliability. The aim of performability evaluation is to measure and assess the behavior of a system under different conditions, environments, and workloads. This chapter provides an overview of the methodologies used for performability evaluation, including their strengths and limitations.

- *Analytical modeling.* Analytical modeling involves using mathematical models and equations to predict the performability of a system. This method is useful for systems that can be represented by well-defined mathematical models, such as queues, networks, and computer systems. The advantage of analytical modeling is that it provides a theoretical performability prediction, which can be used to compare different system configurations and to identify the bottlenecks of the system. However, this method is limited by the accuracy of the mathematical models used and may not reflect the actual performability of the system in real-world scenarios.

- *Simulation.* Simulation is a method of performability evaluation that involves creating a model of the system and running it in a simulated environment. The model is subjected to different scenarios and workloads, and the performability metrics are recorded. This method provides a more accurate representation of the system's behavior in real-world scenarios and allows for the exploration of different system configurations. The main disadvantage of simulation is that it can be time-consuming and resource-intensive, and the accuracy of the results depends on the accuracy of the model. Simulation is jointly related to analytical modeling, and in some literature it is considered an analytical method itself.

- *Experimental evaluation.* Experimental evaluation involves measuring the actual performability of the system in a controlled environment. This method involves setting up a testbed, running experiments, and collecting performability data. The advantage of experimental evaluation is that it provides accurate and precise measurements of the system's performability. However, this method can be time-consuming and resource-intensive, and may not reflect the actual performability of the system in real-world scenarios. Benchmarking belongs to this class of investigation technique.

Briefly, the aim of performability evaluation is to provide a comprehensive understanding of the system's behavior and to identify opportunities for improvement. The choice of methodology depends on the characteristics of the system being evaluated, the resources available, and the goals of the evaluation. That said, there is not a fit-for-all approach and it might occur than more than one methodology has to be adopted in order to conduct a satisfactory performability evaluation of the system, or to validate a model elaborated.

1.3.1 Analytical models

Analytical modeling is a widely-used tool for evaluating the performability of complex systems, processes, and networks. These mathematical representations of real-world systems allow for the prediction of performability under various conditions, enabling the identification of bottlenecks and the development of strategies to improve efficiency. The field of analytical modeling encompasses a variety of techniques, including queuing models [Tri08], simulation models [LKK07], and optimization models [Rao19], each with their own strengths and limitations.

Queuing models [Tri08], for example, are particularly useful in the analysis of systems that involve waiting in line, such as call centers and computer networks. Simulation models [LKK07], on the other hand, are well-suited to the study of complex systems by creating virtual representations of the system and running experiments on these models. Optimization models [Rao19], meanwhile, are particularly useful in identifying the best possible solution to a problem by finding the optimal combination of inputs.

To ensure the accuracy and relevance of an analytical model, a thorough understanding of the system or process being studied is essential. This understanding is typically obtained through the collection and analysis of data. Additionally, it is crucial to validate the model by comparing its predictions to actual data.

The choice of method or technique is also a crucial aspect of analytical modeling. For example, if the goal is to optimize the performability of a system, an optimization model may be the best choice. However, if the goal is to study the behavior of a system over time, a simulation model may be more appropriate.

In the field of performability evaluation, analytical modeling can be used to study system availability. Availability is a measure of how often a system is able to perform its intended function, and analytical models can be used to predict the likelihood of failures and the time required to repair them.

Queuing theory

Queuing theory [Tri08], a branch of operations research, has its mathematical foundation on the notions of Markov process and Poisson process. It studies the behavior of systems that involve waiting in line, also known as queuing systems.

This theoretical framework is particularly useful in the analysis of systems such as call centers, computer networks, and manufacturing systems. A queuing model can be applied to performance and availability modeling by studying the behavior of the system under different conditions, such as the number of jobs (e.g. customers), the number of servers, and the arrival rate of jobs, as shown in Figure 1.7.

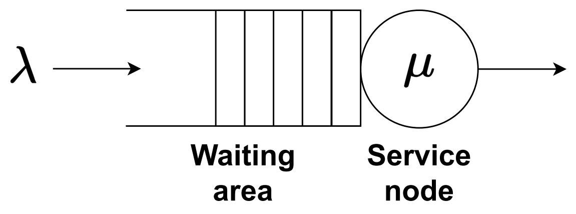


Figure 1.7: Scheme of a single queue (waiting area) with one server (service node), where λ is the arrival rate and μ the service rate.

One of the key concepts in queuing theory is the service rate, commonly indicated by μ in literature, which represents the rate at which customers are served. This value is determined by the number of servers and the service time for each customer. If the system allows the utilization of more than one server, the number of servers would determine the number of customers that can be served contemporary at any given time, with μ, the service time, describing the amount of time required to serve each customer. Another fundamental concept in queuing theory is the arrival rate, in literature denoted with λ, which represents the rate at which customers arrive at the system, i.e., the number of customers arriving in a given time span. In this context, for instance, the arrival process is described as a Poisson point process, and λ is the rate of the Poisson distribution

$$Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{1.3}$$

where X is a discrete random variable distributed according to the Poisson distribution, k the number of occurrences, and λ the arrival rate. Differently, the service process can be model as the time needed to serve a client, which is distributed exponentially

$$Pr(X = x) = \mu e^{-\mu x}, \tag{1.4}$$

where X is a continuous random variable with value x, and μ the service rate (related to the service time $1/\mu$).

Note that the inter-arrival time between two Poisson events is distributed exponentially, therefore the two formulations are strictly related.

The modelling of single queue can be described as a birth-death process, in which the arrival of a customer is a “birth” and a customer leaving the queue after service is a “death”. Indeed, given those two types of event, a continuous-time Markov chain can be constructed, where for each the number of customers k there is a correspondent state. State transitions rates are, therefore, prescribed by the arrival rate λ and service rate μ.

Finally, queuing theory can be used to study the performance of a system by predicting the average waiting time for customers, the average number of customers in the system, and the probability of a customer having to wait. Additionally, it can be applied to the study of system availability by predicting the likelihood of failures and the time required to repair them.

Kendall’s notation

In the previous section, arrival and service mechanisms have been described as Poisson or exponential processes, so that the memoryless property of Markov chains is guaranteed (see next section). Although, there are other stochastic processes, characterised by different probability distributions, that may be used to study a queue [Bol+06]. As shown in the following, Kendall’s notation [Ken53] is a simple, widely used notation, useful to describe elementary queuing systems:

$$A/S/c/K/D,$$

where A is the distribution underlying the arrival process, S the distribution pertaining the service process, c the number of processors/nodes/servers, K the upper-limit for the queue length (e.g. buffer size, allowed customers in line, etc.), and D the serving discipline. More in detail, for each parameter of the set written in Kendall’s notation, follows a description on the specific properties:

- *Arrival process A* and *Service process S*. They describe what kind of process regulate the arrival and service processes, respectively. A and S may be different from each other, with the following codes indicating their properties:

  - Markovian process (M) is a Poisson process, with exponential inter-arrival/service time;
  - Degenerate distribution (D) is a deterministic fixed inter-arrival/service time;
  - Phase-type distribution (PH) is a convolution of exponential distributions;
  - Even in case of a General distribution (G), some analytical results can be obtained. In literature it is sometimes called General Independent (GI) to specify that the processes are independent.

- *Number of servers c.*
- *Maximum size of the queue K.* When not specified, it is assumed to be $K = \infty$.
- *Service discipline D.* This is the set of rules regulating which element in the queue is being served, it includes:
  - First In First Out (FIFO), in which system is serving jobs in the order of arrival (it is the default discipline if D is not specified);
  - Last In First Out (LIFO), in which the last job arrived is served first;
  - Service In Random Order (SIRO) serves jobs in random order, regardless to arrival order;
  - Priority Queuing (PQ) sets a priority policy for serving jobs;
  - in Round Robin (RR) if the servicing of a job is not completed at the end of a time slice of specified length, the job is preempted and returns to the queue, which is served according to FIFO. This action is repeated until the job service is completed.
  - Processor Sharing (PS) corresponds to a RR with infinitesimally small time slices. It is as if all jobs are served simultaneously and the service time is increased correspondingly.

Markov Process

Markov processes, also known as Markov chains (shown in Figure 1.8 there is a Markov chain representing a simple queue, known as birth-death process), are mathematical models used to describe the behavior of systems over time. They are widely used in various fields, including physics, economics, and computer science, as a mean to predict future outcomes based on the current state of the system. Indeed, at the core of Markov processes, there is the concept of Markov (or memoryless) property, which states that the future state of a Markovian system is dependent only on its current state and not on its past history. This property allows for the modeling of systems in which the future state cannot be predicted with certainty, but instead is statistically determined by the current state. The study of Markov processes involves the analysis of transition probabilities between states, and the computation of various statistical properties, such as the stationary distribution and the expected time to reach a certain state. Before proceeding to

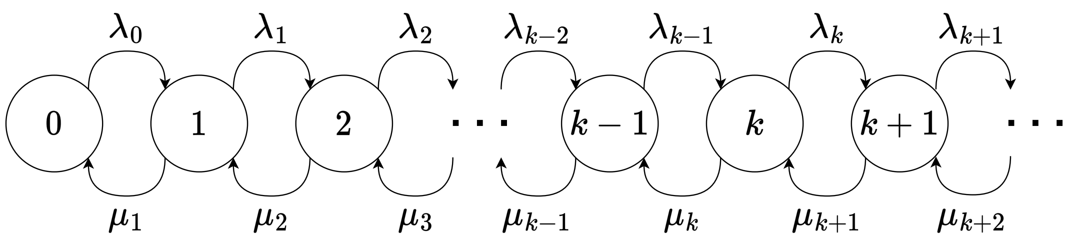


Figure 1.8: A Markov chain depicting a birth-death process. States are represented by rounds with the state number within. Transition rates ($\lambda_0, \lambda_1, \ldots$ for arrival rates and $\mu_1, \mu_2, \ldots$ for service rates) may be in principle different depending on the state, but generally an effective arrival rate λ and service rate μ can be defined.

look more in detail the mathematical description of Markov chains, assume that, unless differently stated, the possible values of the random variables X_i form a countable nonempty set S, which can be finite. There are two category of Markov chain: discrete-time Markov chains and continuous-time Markov chains (CTMC). In a discrete-time setting, time variable is, therefore, treated as discrete, with the process advancing in steps. The Markov property in this framework reads

$$P(X_{n+1} = x_{n+1} | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(X_{n+1} = x_{n+1} | X_n = x_n), \tag{1.5}$$

which states that, after a series of events for the random variables $X_1, X_2, \ldots$, the conditional probability for the random variable X_{n+1} to be x_{n+1} is conditioned exclusively by the probability of X_n to be x_n.

In continuous-time Markov chains, the transition probability from a state to another may happen at any time $t \geq 0$, with rates determined by the transition matrix and the current state of the chain. The formulation for the Markov property can be stated similarly to the discrete-time one, given that the set of states S is finite. Otherwise it can be defined as a infinitesimal time step, with probabilities computed by solving the associated first-order differential equation. In the case of a finite set, the Markov property can be written adjusting Equation 1.5, with indices n replaced by indexed times up to n, $t_0, \ldots, t_n$, where the spacing between times is not homogeneous.

An utterly significant contribution obtainable by the application of Markov chains is the assessment of the steady-state probability of the studied system. Differently from the transient analysis, which involves dealing with differential equations and yielding a time-dependant solution, the steady-state solution of a Markov chain requires to solve a set of homogeneous linear equations, resulting in a solution, the steady state probabilities, being not dependent on time. Indeed, given an irreducible recurrent, hence ergotic, CTMC, the process converges to a probability distribution P, that is not time-dependent. This probability distribution P may be found solving the eigenvalue problem

$$AP = 0, \tag{1.6}$$

subject to

$$\sum P_i = 1, \tag{1.7}$$

where P is the steady-state probability row vector (this is why, being a probability, the sum of all the elements of P must be one) and A is the transition rate matrix (i.e. the matrix of the coefficients associated with the set of linear equations). This is, indeed, an eigenvalue problem, because the row vector P is the (right) eigenvector of the matrix A corresponding to the eigenvalue zero.

Refer to [Bol+06] for a complete and exhaustive exposition on Markov chains and their application in queuing theory.

Chapter 2

Related Work

Here are reported and reviewed the works found in literature, which are underlying or are connected with the research presented in this thesis. The chapter in separated into two main branches: consensus protocol selection and performability evaluation. The latter, besides the literature on analytical modeling, includes also an attempt of benchmarking, that was developed and used to characterize the performance of a blockchain implementing two types of consensus protocols.

2.1 Consensus protocol selection

As mentioned in section 1.2, it exists a plethora of consensus protocols in the DLT domain, each of which may have variations in their implementation, leading to a vast array of options. In order to bring structure to the complex landscape of consensus protocols for DLT, several researchers have attempted to classify and categorize these protocols.

Baliga (2017) [Bal17] conducted a comparative analysis of five popular categories of consensus models using seven criteria. The authors concluded that, for permissionless platforms, achieving consensus among a large number of untrusted peers must be done through robust computational or memory complexity, thereby sacrificing transaction finality and throughput. On the other hand, for permissioned consortium blockchains, options with higher throughput, faster transaction finality, and less scalability are more desirable. The authors emphasized that when choosing the appropriate consensus model for a business, one must take into account the network scale, participant relations, and functional and non-functional aspects such as performance and confidentiality.

Mingxiao et al. (2017) [Min+17] reviewed the principles and characteristics of five consensus algorithms and analyzed their performance based on five factors. Based on this analysis, the authors provided technical guidance on the appropriate consensus protocols for three different blockchain scenarios (public, private, and permissioned). PoW, PoS, and DPoS are commonly used as consensus mechanisms in public blockchains as they are completely open and decentralized. In private blockchains, it is more crucial to handle crash faults than Byzantine faults, making PBFT and RAFT (crash tolerant) consensus mechanisms suitable choices. For permissioned blockchains, which are semi-closed networks built by multiple enterprises, PBFT may be a promising option due to the potential conflicts and possibility of malicious nodes among enterprises.

Nguyen and Kim (2018) [NK18] conducted two separate comparisons of consensus protocols. First, they compared three major categories using six criteria, and then compared vote-based and proof-based consensus protocols using eight criteria. The authors concluded that vote-based consensus protocols have more potential for use in consortium and private blockchains, while proof-based consensus protocols are commonly used in public blockchains. The authors did not compare specific protocols, but aggregated results from several specific protocols to represent each category.

Alsunaidi and Alhaidari (2019) [AA19] classified consensus protocols and compared six proof-based and two vote-based protocols based on 16 criteria. The authors noted that most proposed protocols are designed for cryptocurrency systems and smart contract transactions.

Wang et al. (2019) [Wan+19] conducted an extensive literature review on the development of decentralized consensus mechanisms in blockchain networks. The authors compared 11 Proof-of-X (PoX) schemes for permissionless blockchains and 12 virtual mining and hybrid consensus protocols based on several criteria. They analyzed the connection between permissionless and Byzantine agreement consensus protocols, the incentive compatibility in permissionless consensuses from a game-theoretic perspective, and the impact of consensus participants' strategies.

Based on their comprehensive survey, the authors provided insights into the emerging applications of blockchain networks, with a focus on the telecommunications field.

Bano et al. (2019) [Ban+19] carried out a systematic and comprehensive examination of blockchain consensus protocols was conducted, resulting in the development of a framework for evaluating their performance, security, and design properties. The analysis of three high-level design categories, including protocols based on Proof-of-Work (PoW), Proof-of-X (PoX) protocols that replace PoW with more energy-efficient alternatives, and hybrid protocols that are compositions or variations of classical consensus protocols, was based on twelve criteria and data from real blockchain systems. The authors identified research gaps and challenges to be considered for the future development of consensus protocols to promote the widespread adoption of blockchains.

Belotti et al. (2019) [Bel+19] conducted a comparative analysis of eight consensus algorithms based on eight criteria, which showed a tendency towards the implementation of safer and high-performance consensus protocols with low energy impact and latency that reach a final agreement with the guarantee that validated blocks will not be discarded.

Nijsse and Litchfield (2020) [NL20] presented a taxonomy of consensus methods applied to current blockchains and highlighted 19 consensus methods compared based on seven criteria. The authors demonstrated the extensibility of the taxonomy on four cases that were not involved in its development.

Xiao et al. (2020) [Xia+20] carried out a comprehensive survey of distributed consensus protocols for blockchain networks, where they identified five core components of blockchain consensus protocols and reviewed and compared 22 consensus protocols based on nine performance metrics. This work provided new insights into the fundamental differences of existing consensus proposals and their suitable application scenarios, assumptions, expected fault tolerance, scalability, drawbacks, and trade-offs.

Zhang and Lee (2020) [ZL20] compared five consensus protocols based on five criteria and gave guidance on selecting suitable consensus protocols depending on the blockchain type. They specified that PoW, PoS, and DPoS-type consensus algorithms are the most reasonable choices for public blockchains, while PBFT and

Ripple are more suited for permissioned blockchains to ensure higher efficiency.

Fu et al. (2021) [FWS21] carried out a detailed survey of mainstream blockchain consensus algorithms by analyzing 17 consensus protocols and providing a classification of consensus protocols based on four mode groups. The authors presented a three-dimensional evaluation framework consisting of effectiveness, decentralization, and security, and identified five common blockchain application scenario categories and provided suggestions for selecting consensus algorithms for each category.

In conclusion, the classification and analysis of consensus protocols in the field of DLT is an active area of research with numerous studies providing different methodologies and categorizations. These studies have compared different consensus protocols based on various criteria such as scalability, performance, finality, energy impact, and security. The conclusions drawn by these studies highlight the need for further research and development in the field of consensus protocols to overcome the current limitations in performance and scalability, and to ensure wide-scale adoption of blockchains. Ultimately, the choice of a suitable consensus protocol will depend on various factors such as the scale of the network, the relationship between participants, and functional and non-functional requirements.

2.2 Performability evaluation techniques

In recent years, there has been a growing interest in evaluating the performance of DLT platforms and consensus protocols, both through empirical analysis and analytical modeling. The motivation behind this interest is twofold: to gain a deeper understanding of the process through modeling, and to estimate the parameters of new systems before actually developing them, saving time and resources [RHF21; Rim+17]. Various efforts have been made to categorize methods and techniques for performance evaluation of DLT systems [Fan+20; Sme+20a].

In the following, some formal definitions used in this thesis are presented, followed by the literature related to benchmark and analytical modeling.

2.2.1 Definitions

Performability was defined by J.F. Meyer [Mey80] as “a unified performance-reliability measure” and “It is shown that performability relates directly to system effectiveness and is a proper generalization of both performance and reliability.” In [CDK05; Bol+06] there can be found the definitions of the metrics that can be computed in the context of performability.

Assuming that the distributed system under consideration exchanges messages to achieve a desired goal, there are at any given time $t \in [0, \infty)$ a defined amount of messages/jobs - $j \in [0, J]$, with $J \in [1, \infty)$ - to be handled by the nodes in the network. Note that, rigorously, this concept has to be extended to fit probability theory by introducing the random variable “number of jobs”, $\tilde{J}$, with values $j \in \{0, 1, \ldots, J\}$. However, using the formalism presented in section 1.3, the transient of the system is not studied, instead the focus is on the asymptotic behaviour. That is the reason why the measures, presented in the following, are defined as average (or mean) quantities.

Blocking probability, in this framework, is the probability that the buffer of the system, with length J, is full and it cannot accept any new message until some slot in the buffer memory is free. Formally, it is simply

$$blocking\_probability = Pr(\tilde{J} = J), \tag{2.1}$$

since it is important only to determine whether the system contains already J messages/jobs.

Mean queue length is the average number of messages that are waiting to be served by the system. The mathematical formulation descends directly from the definition of expected value, i.e.

$$mean\_queue\_length = E[\tilde{J}] = \sum_{j=0}^{J} j \, Pr(\tilde{J} = j). \tag{2.2}$$

Throughput measures the average number of messages/jobs served by the system in unit time. This can be expressed in terms of rate of incoming messages λ and the probability that there is at least one message to be served, while the system is available. This can be written as

$$throughput = \lambda \, Pr(\tilde{J} \geq 1 \cap \text{system available}) = \lambda \, Pr(\tilde{J} \geq 1) \, Pr(\text{system available}), \tag{2.3}$$

where the last equivalence holds because the two events are independent. This formula tells also that the throughput cannot be larger than the arrival rate λ, hence it is irrelevant if it is used λ or μ (the service rate) to compute the throughput, because, even in the case $\mu > \lambda$, the actual throughput cannot exceed the rate of arriving messages. In other words, the service rate μ is equal to the arrival rate λ for a queueing system in statistical equilibrium.

Transaction latency (or simply *latency*) is the average time that a message spends in the system, from its arrival until it is served. Latency can be computed using Little's Law [Lit61]:

$$latency = \frac{mean\_queue\_length}{throughput}. \tag{2.4}$$

Lastly, for what concerns the availability, it can be defined as the ability of a considered system to be in a state to perform an operation at any instant time within a given time interval. In other words, it refers to failure-free operation at a given instant of time [Tri08]. As already mentioned, however, since the transient measurements are not of interest in this analysis, it is presented the notion of limiting availability, A, that can be defined as

$$A = \frac{MTTF}{MTTF + MTTR} \tag{2.5}$$

where $MTTF$ is the *Mean Time To Failure* and $MTTR$ the *Mean Time To Repair*.

2.2.2 Benchmarking

Benchmarking is an established and variegated set of techniques used to test systems, in order to retrieve information on, for example, the performance of said systems [DJ03; AK08; MR13]. It provides a standardized way to measure the performability of systems and applications under controlled conditions, enabling accurate comparisons between different solutions and configurations [Coo+10]. Examples of benchmarking include the Google Cloud Storage Benchmark, which evaluates the performance of cloud storage systems, and the Yahoo! Cloud Serving Benchmark, which measures the performance of NoSQL databases. Another example is the Network File System benchmark, which measures the performance of NFS-based network file systems.

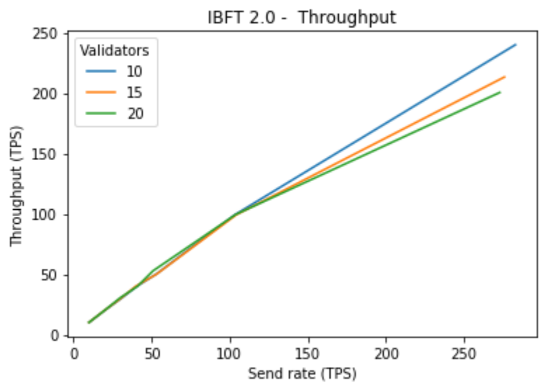
Although benchmarking cloud storage systems and distributed database is leveraging mostly on established tools and metrics, this not the case for what concerns DLT, due to a scarce standardization and a lack of solutions developed by big players (e.g. universities, multi-national companies, etc.). However, the performance evaluation of DLT systems is crucial to ensure their scalability, reliability, and suitability for various applications. In this context, benchmarking is a powerful tool, providing an objective method of measuring various performance metrics of DLT systems. In particular, when analysing distributed ledgers, important metrics to consider include latency, throughput, scalability, and robustness. The benefits of benchmarking DLT systems are numerous. First, it provides a method of evaluating the performance of different DLT systems, allowing for accurate comparisons between different solutions and configurations. Additionally, benchmarking enables the identification of performance bottlenecks, helping to improve the overall design and implementation of DLT systems. Nevertheless, as already mentioned, one of the main challenges is the lack of standardization in DLT, making it difficult to compare the results from different studies. Additionally, DLT systems are highly complex and can be difficult to evaluate accurately, making benchmarking a challenging task.

Examples of benchmarks assessing characteristics and qualities of DLT systems include [Fan+20; SS21; De 18; HR17; Tin19] On this subject, in [Bis+22], we presented a benchmark of a permissioned consortium blockchain implementing two different consensus protocols, IBFT 2.0 and Clique. The work's goal is to assess the performance in terms of throughput (see Figure 2.1) and transaction latency (see Figure 2.2) for a network of sensors recording data on a blockchain. The tested blockchain is based on Hyperledger Besu [a], an Ethereum client devoted to the development of public and private networks.

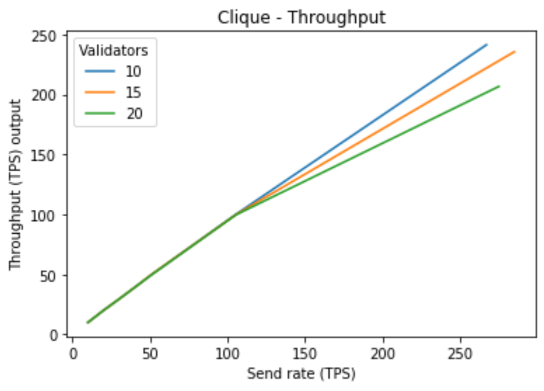
More specifically, the blockchain has been deployed and tested on a single

[a]`https://besu.hyperledger.org/` [Accessed February 26, 2023]

machine with the following characteristics: Operating System: Ubuntu 20.04.3 LTS, 2 CPU: Intel(R) Xeon(R) Gold 6256 CPU @ 3.60GHz, RAM: 128GB. The version of the Besu client used for running the nodes is the 21.7.0-RC1.

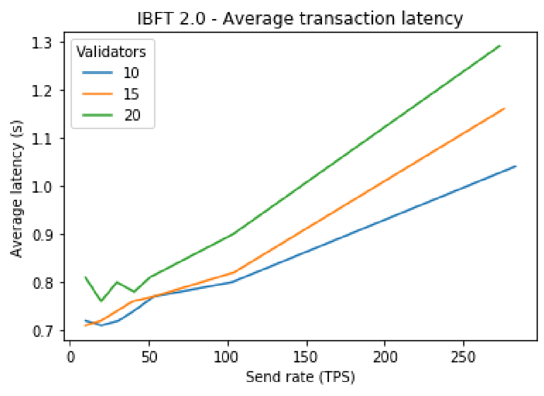


(a)

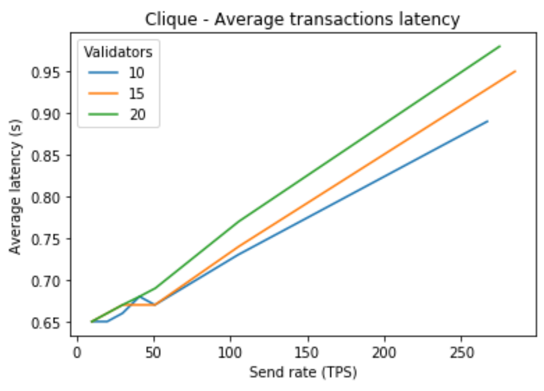


(b)

Figure 2.1: Average throughput using IBFT 2.0 and Clique.



(a)



(b)

Figure 2.2: Average transaction latency using IBFT 2.0 and Clique.

The tests are set such that blocks are produced at constant rate (one block every 1 second), while send rate, i.e. the number of transactions per second sent to the system, are varied from 10 TPS to 300 TPS. Tests are performed for 10, 15, and 20 validators present in the network.

Note that benchmarking was tested as an approach to assess the metrics of interest in the design of a blockchain system, in the context of the project behind this research. However, the technical challenges associated with the implementation of benchmarks, and the limited, not flexible environment utilized to conduct them, pushed the choice of analytical modeling as a more suitable approach to evaluate performability metrics for the system of interest.

2.2.3 Analytical modeling

There are examples of articles that have reviewed the use of analytical methods to study the performance of consensus protocols in blockchains [Sme+20b; Ma+20]. This includes the employment of techniques that can be used in the task of analytic performance evaluation, such as Stochastic Reward Net (SRN) [Suk+17], game theory [QYJ20], and hierarchical model approach [Jia+20]. In this thesis, the focus is on articles that apply Queuing Theory to the performability evaluation of DLT systems [KK17; LMC18; Li+19; Ric+19; Gei+19; Fra20; WG21; Bal+21; Bal+22; HMZ19; Men+21; MF22].

Of all the different DLT platforms available, blockchains, and in particular Bitcoin [Nak08], have received the most attention, where the goal of many related works is to develop a queuing model of a generalized PoW consensus protocol, using Bitcoin as an example [KK17; LMC18; Li+19; Ric+19; Gei+19; FM19; Fra20; WG21; Bal+21; Bal+22]. These articles focus their attention on different aspects of bitcoin-like blockchains, e.g. time delay and mining process, thus the implementation of specific models to describe those processes. Specifically, in [Ric+19], it is developed a $M/G/1$ queueing theory model to characterize the delay experienced by Bitcoin transactions. Authors of [FM19; Fra20] make use of $G/M/\infty$ and $M/G/\infty$ queues to study the synchronization in Bitcoin network. Li *et al.* [LMC18; Li+19] employ a $GI/M/1$ queue that can provide analysis both for the stationary performance measures and for the sojourn time of any transaction or block. By means of a batch Markov serving process $M/M^B/1$, in [Bal+21] the consolidation time of transactions in Bitcoin network is studied, with regard to the relation between the fee offered by a transaction and its expected consolidation time. A similar endeavour is the focus in [KK17], where $M/G^B/1$ queues are applied to study the transaction confirmation time for Bitcoin. In [WG21] it is presented a blockchain model based on a wireless infrastructure, where to determine its performance metrics a discrete-time $M/M^B/1/K$ queue is used. The work in [Gei+19] aims to investigate key performance indicators and general limits of blockchains with the aid of a discrete-time $GI/GI^B/1$ model.

These studies have opened the way to the study of other consensus protocols, such as Raft and Raft-based protocols for private [HMZ19] or consortium blockchains [Men+21]. For instance, Huang *et al.* [HMZ19] model Raft using a simple, yet effective, $M/M/1$ queue. The proposed model can predict the network split time and probability this may happen. Hyperledger Fabric has been modeled in [Men+21], where a $PH/PH/1$ queue is used to analyse the consistency properties of consortium blockchain protocols.

In this work, the focus is on BFT-based protocols. These algorithms have been studied in the context of performance evaluation of blockchains using queuing theory models, with Ma and Fan [MF22] proposing a $M/PH/1$ model to evaluate the performance of the Improved PBFT protocol and Chang et al. [Cha+22] presenting a $M \oplus M^b/M^b/1$ model to describe dynamic PBFT systems.

The works here mentioned are both applying the matrix-geometric solution to analyze the PBFT blockchain system. This is a standard approach to solve the problem of the increasing complexity (and dimensionality) of the problem, exploiting the repeating structure of the matrix representing the balance equations.

For what concerns the assessment of system availability, books and surveys, e.g. [TB17], are helpful tools to have an overview of the state-of-the-art about the topic. Along with other examples, availability models relying on Markov chains found applications in several areas, including healthcare [SKU18; TX21], IoT systems [Eve+19; Per+21], wireless sensor networks [MG11; Sil+12; MAG15; Arj+17; Par19], Infrastructure-as-a-Service clouds [Ata+17; Lon+11; Gho+14], distributed storage systems [For+10], and blockchains [Mel+21].

In [SKU18], a novel approach is presented that considers two-dimensional continuous-time Markov chains for functional states of the healthcare IoT infrastructure and end nodes. The study also includes a case study with a Markov model that considers attacks on the vulnerabilities of the healthcare IoT system, along with a state diagram for these attacks. The availability of the system is presented as a function of the intensities of service requests flow, with the main emphasis being on safety and security-related issues. Furthermore, in [TX21], the availability of healthcare IoT systems is also studied. The study describes two groups of structures, which are the components of the IoT system, using separate Markov state-space models. A two-dimensional state space representation is established and the system balance equations are solved, similar to the approach presented in [SKU18]. The study also presents availability-related performance metrics such as the probabilities of full service, degraded service, and system unavailability.

The availability of IoT systems is considered in various works, such as [Per+21] and [Eve+19]. These studies examine the availability of IoT systems, with a focus on evaluating the physical edge and fog nodes that run applications. In particular, in [Per+21], authors present analytical availability models, compute the $MTTF$ and $MTTR$ values for the systems under study, and develop a two-dimensional Markov model to account for both failures and repairs. Similarly, work in [Eve+19] evaluates the performance and energy consumption-related measures of clustered IoT systems. Authors use two-dimensional models to calculate the steady-state probabilities, which are then used to compute various availability and performance metrics, such as the probability of being in a fully operational state and the mean energy consumption. In addition to the availability of IoT systems, the impact of failures on facilitating infrastructures has also been studied. For instance, in [Kir+15], researchers model the presence of failures in facilitating infrastructures.

The scalability of cloud systems, particularly Infrastructure as a Service (IaaS) based ones, is a limiting factor for modeling attempts. To address this challenge, various approaches have been proposed in the literature. In [Ata+17], the authors tackle scalability-related problems in large cloud systems using approximate Stochastic Reward Net (SRN) models combined with folding and fixed-point iteration techniques. The proposed approach is capable of capturing the failure/repair behavior of physical machines and analyzing the percentage of available physical machines for different failure and repair rates. In [Lon+11], the authors focus on the high availability of IaaS cloud systems. To reduce complexity and solution time, they employ an interacting Markov chain based approach and use SRNs to solve the Markov chains. The availability models presented in the study are used to perform trade-off analysis of longer $MTTF$ versus faster $MTTR$ on system availability, as well as the effect of having multiple concurrent repair facilities. In [Eve17], the authors introduce a novel approximate solution approach that allows the consideration of large numbers of servers for cloud-based systems. The analytical models and solutions are monolithic, but still capable of considering a large number of facility nodes, typically up to hundreds or thousands. The study considers the quality of service for cloud centers together with server availabilities, and obtains performability measures in the presence of server failures and repairs.

In recent years, the use of blockchain technology has gained significant attention as a means of supporting secure and decentralized service provisioning over cloud infrastructures. The article in [Mel+21] presents novel models for evaluating the availability and capacity-oriented availability of cloud computing infrastructures running blockchain-based distributed applications. The authors focus on the Ethereum blockchain platform, which has emerged as a prominent platform for decentralized applications. The models presented in the study use the traditional approach to represent the system's availability as the ratio between the $MTTF$ and the $MTTR$. The authors present availability results as functions of $MTTF$ and $MTTR$ for the components of the system, including the server, miner node, and bootnodes.

In summary, related works on analytical modeling for performability evaluation of DLT systems are widespread in literature, with various approaches applied to different type of environments and settings. In the reviewed literature, both for availability evaluation and for performability analysis, the main focus in on blockchain platforms and related consensus mechanisms, especially applied to model bitcoin-like systems and block mining process in PoW. For instance, the literature on the modeling of BFT-based protocols is limited in quantity, even though variegated in methods. There are some examples of Markov chain-based models applied to BFT protocols, but none of them takes into consideration explicitly the presence of Byzantine nodes in the network, rather they consider stop-fail errors, even though they effectively describe and model the dynamic variation in number of nodes during the consensus process. The analytical models reported in this thesis (as presented in [Mar+23a; MM23]), instead, are based on CTMC to assess the availability and performability of protocols related to PBFT, including the Improved PBFT protocol and other variations. The main contribution in this context is the definition of a multi-dimensional state diagram, able to give availability and performability metrics as exact solutions of the balance equations describing the system model, while accounting explicitly for the presence of Byzantine nodes.

Chapter 3

Methodology

The main focus of the research presented in this thesis is the development and implementation of techniques aimed to assess the performability of BFT-based consensus protocols for DLT systems. In general, consensus protocols have a vital role in the correct and secure functioning of DLT solutions [Sin+22; Bao+23], therefore the need for analytical modeling for performability evaluation is central, when creating a DLT platform. In this chapter, the methodology employed to design and characterize benchmarks and analytical models for performability evaluation is presented.

However, before outlining the methodology used in performability evaluation, the process behind the selection of BFT protocols, as elaborated in chapter 4, is hereby reported:

1. **Problem definition**.
   The requirements and constrains for the desired use-case are defined, stating the protocol selection as a MCDM problem. For instance, alternatives and criteria are selected, then weights are assigned to each criteria. The expected result is the ranking of the alternatives, optimized according to the MCDM techniques.

2. **Selection of alternatives**.
   Alternatives (DLT platforms) are selected using two rationales, i.e. they are the most used/valuable platform in the DLT application/commercial

sector, or they belong to and they are the most prominent representative of a consensus protocol family, as reported in literature.

3. **Selection of criteria**.
   There are numerous publications on the analysis, classification, and selection of DLT platforms in different contexts. By reviewing them, it is possible to summarize and aggregate the most suitable criteria to fully describe a consensus protocol by means of the metrics associated with its implementation.

4. **Data acquisition and processing**.
   Data for each selected alternative and criterion is gathered from reputable sources (e.g., academic literature and network analytics). Because of the heterogeneous nature of data, these have to processed and standardized.

5. **Criteria weighting**.
   MCDM experts are the primary source of knowledge to select the most appropriate weighting method and, eventually, values of these weights. Additionally, recommendations provided in the peer-reviewed scientific literature can be used as a reliable source.

6. **Ranking**.
   The selected MCDM methods are applied to obtain the ranked alternatives as result. The ranking is giving the most suitable DLT platforms for the defined scenario.

Therefore, by applying the MCDM framework described above, it is possible to select an element/family of consensus protocols to focus on. As mentioned in the Introduction, performability evaluation techniques are applied to study BFT-based consensus protocols used in DLT solutions. In the following there is the methodology applied in chapter 5:

1. **Research Design**.
   Given the nature of the research, a quantitative research design is adopted. Analytical modeling and benchmarking are employed to assess the performance of consensus protocols in DLT.

2. **Data Collection**.
   Primary data is the results generated through analytical models and benchmarks. The tools used include custom Python scripts, and the data collection procedures involve running the scripts with varying parameters.

3. **Variables and Measures**.
   Key variables include number of network nodes, amount of Byzantine nodes, failure and repair rates, transaction throughput (referred as arrival rate and service rate in analytical models). These variables are all functional within the analytical models, however in benchmarking a subset is used, i.e. input throughput and number of nodes. The measures of performability metrics include system availability, throughput, transaction latency/response time, transaction blocking/loss probability, mean number of transactions in the system; from benchmarks, a subset of these metrics can be obtained, i.e. throughput and latency/response time.

4. **Data Analysis**.
   The analysis of the data obtained from analytical models versus data from benchmarks may be used to validate the analytical model. For data analysis and visualization, custom Python scripts are used.

5. **Limitations and Assumptions**.
   Limitations include the simplifications inherent in analytical models and the assumptions made about network conditions. The impact of these limitations on the findings is discussed.

6. **Rigor and Validity**.
   Rigorous validation methods, including sensitivity analyses and comparisons with real-world data where possible, are employed to enhance the internal validity of the findings.

The methodology here presented follows a standard approach to performability evaluation, both for benchmarking and analytical modeling. The main strength of this methods is that performability evaluation techniques are flexible tools to asses important metrics of DLT systems. The main challenges are connected with the difficult abstraction of the studied protocol in order to develop an analytical model, while for benchmarks the implementation of a working system and the tuning of parameters may happen to require a large amount of resources.

Chapter 4

Protocol selection framework

Earlier, in section 2.1, it has been established that there are abounding ways to classify consensus protocols for DLT. One straightforward approach is to define whether a protocol is Proof-based either Vote-based [NK18; AA19]. Another related way is to determine what resource is fundamental in the consensus process, i.e. computational power (e.g. PoW), tokens (e.g. PoS), or votes (e.g. BFT) [NL20; Ban+19]. A different method requires to establish how much the analysed protocols suit a certain network type, i.e. public, consortium, and private networks, according to some criteria [Bal17; ZL20; Fil+22].

This chapter presents a MCDM framework for consensus protocol selection, as from [Fil+22]. This scheme has been proposed to aid decision makers in selecting a suitable consensus protocol for a certain DLT application: a decision maker distinguishes, according to the project's needs, which consensus protocols are suiting the most in those conditions, using the ranking obtained from the MCDM method as a guideline.

In the rationale of this thesis, the proposed method helped (see section 4.5) in the selection of a BFT-based consensus protocol as backbone of the system to be implemented in the context of the national project HD3FLAB. Hence, the choice to proceed and analyze the performability properties of BFT protocols as further step in the wider frame of the project.

4.1 Problem definition

The problem of ranking DLT platforms according to some criteria dictated by the application/use-case can be formalized as a multi-criteria optimization problem, which make use of MCDM techniques to achieve a solution.

The problem is set by letting $A = \{A_1, A_2, \dots, A_m\}$ $(m \geq 2)$ be the row vector of alternatives (the DLT platforms selected to represent consensus protocols), while the row vector of criteria is $C = \{C_1, C_2, \dots, C_n\}$ $(n \geq 2)$, and the vector of their weights is $W = \{w_1, w_2, \dots, w_n\}^T$, such that $\sum_{j=1}^{n} w_j = 1$ $(w_j \in [0, 1])$. The row vector of alternatives and the row vector of criteria can be used to express the decision matrix $\mathbf{X} = A \otimes C = |x_{ij}|_{m \times n}$, where $A \otimes C$ is the tensor product of the two row vectors. Practically, each element of $\mathbf{X}$, x_{ij}, is the value of the alternative A_i respect to the criterion C_j.

At this point - in the case where all the defined criteria are benefit criteria (so that the problem would turn into a maximization problem) and the weights are additive - the ranking of the alternatives would require simply to reorder the vector with the total scores for each alternative, i.e.

$$A_i^{\text{score}} = \sum_{j=1}^{n} x_{ij} w_j. \tag{4.1}$$

Indeed, this straightforward algebraic calculation (Equation 4.1), also known as Simple Additive Weighting (SAW) or Weighted Sum Model (WSM) [Mac68], is possible only under stringent conditions: it is a maximization problem and the weights are additive.

However, the protocol selection problem may require the optimization both of benefit criteria and cost criteria, plus some other constraints. For instance, there exist approaches to adapt the SAW method to discern between benefit and cost criteria under certain constrains, as well other techniques, like Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [Mac68] and VIseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR) [OT04]. Specifically, in TOPSIS it is computed the Euclidean distance between each alternative and two points: the positive ideal solution (that maximises the benefit criteria and minimizes the cost criteria) and the negative ideal solution (that maximises the cost criteria and minimizes the benefit criteria). Then, the ranking is obtained by ordering the alternatives according to how close to/far from the positive/negative ideal solutions they are, i.e. the best alternative is the one that is the closest to the positive ideal solution and farthest from the negative one. The VIKOR method is maximizing group benefits and minimizing individual regret. The method is based on creating a solution within the scope of alternatives and criteria closest to the ideal solution. VIKOR determines the compromise ranking list, the compromise solution, and the weight stability intervals for the reference stability of the compromise solution obtained with the given weights.

It is out of scope in this instance to discuss further the actual formalism and implementations of each possible MCDM techniques applicable to solve the stated problem. As a matter of fact, there are extensive and detailed works in literature explaining different methods and their construction [MA04; Gun18]. For the results reported in this thesis and in [Fil+22], it has been used an automatic tool that computes the solution to a MCDM problem using 14 different techniques, including SAW, TOPSIS, and VIKOR (used in this thesis). This tool is freely available upon request to the authors of [WR17; Wan+20].

4.2 Criteria definition

In literature, there have been studies dedicated to determine evaluation criteria for the classification of consensus protocol performance. As a basis for structuring the selection criteria, mainly two work have been used as references [BMB20; SAS20].

From the analysis of the literature, reported also in [Fil+22], it has been found that those criteria can be categorized into five groups: *throughput*, *decentralization*, *incentivization*, *sustainability*, and *security*. For each group the corresponding criteria (metrics) are taken into account. Criteria to characterize consensus protocols are specified according to the most known implementations for each protocol. This choice is based on the already discussed correlation between the consensus algorithm and DLT solution, and for data availability. Below we present our categorization and describe the corresponding criteria in more detail.

Throughput

This criteria group regards the transactions that a consensus protocol can handle and its finalization, i.e., when given transaction becomes definitively part of the ledger.

- **Transactions per second (TPS)**. This metric reports how many transactions are processed by the consensus protocol in a second. In computing this number, theoretical limits (if available) are considered, more than real-world implementations;
- **Transaction latency**. This metric measures the time needed for a transaction to become part of the ledger, from submission to finalization/validation;
- **Finalization**. A criteria defining the finalization process of the consensus protocol, either *probabilistic*, either *deterministic*. It establishes the presence or absence of forks in the ledger.

Decentralization

An important aspect in evaluating a consensus protocol, and a DLT in general, is the measure of the decentralization of the distributed system. In this group we identified two criteria.

- **Number of consensus nodes**. This metric indicates the number of actors participating in the consensus process itself;
- **Number of network nodes**. The number of nodes that are actually keeping a copy of the ledger in their own memory.

Incentivization

In order to properly work, a DLT, especially in a public implementation, needs to incentivize the nodes to participate in the consensus, e.g., a type of reward awarded to nodes.

- **Transaction fees**. This metric gives the average price (in US$) for a client to submit one transaction to the network;

- **Reward**. The average daily monetary reward (in US$) for all the consensus nodes (as a whole) is calculated in this metric. Those values have been computed in different ways - according to the reward system: the average (or fixed) reward per block times the average number of blocks, and expected daily return of investment times the number of recipients.

Sustainability

This criteria group determines the sustainability of participating to the network in terms of electrical power consumption and specific hardware requirements.

- **Power consumption**. This criteria establishes a level of electric power usage to maintain and run the network. It is an important metric to take into account. Just to give an example for mining Bitcoin, there is a total electric consumption comparable to the need of a small-medium state;
- **Hardware dependency**. A criteria stating if there is any advantage given by specialized hardware (e.g., ASICs, GPUs, etc.) to participate into consensus.

Security

A very critical criteria group is the one taking into account security issues, because security level requirements are inherently different in public-consortium-private and permissionless-permissioned DLTs.

- **Fault-tolerance**. This is the maximum percentage of faulty (or misleading) nodes allowed in the network in order to securely run the consensus protocol;
- **51% attack**. It measures the level of vulnerability of a consensus protocol to the attack conduct by a powerful (in terms of resource needed to participate into consensus) adversary;
- **Double spending**. In case of forks or other scenarios, a dishonest node may attempt to spend its own currency/tokens several times by acting maliciously in consensus process. In this metrics, it is reported if a protocol is vulnerable to such a threat.

4.3 Criteria weights

There are different approaches to set the weights of criteria in a multi-objective optimization, broadly objective weight calculation, subjective weight calculation, and a combined approach [Odu19]. In this thesis, a subjective weight calculation method is employed. This method relies on some decision makers (i.e. experts in the field) to assess the pairwise importance of the selected criteria, according to a predefined scale.

The process to determine the weights to be used in the MCDM framework is composed of the following parts: (I) an Analytic Hierarchy Process (AHP) [SP08] shows the relation among the goal of the decision, the criteria groups and their metrics; (II) some preference scale has to be determined (III) to perform a pairwise comparison of the criteria; (IV) the consistency of the choices is estimated and, lastly, (V) weights are computed, both for criteria groups and for criteria.

(I) AHP is practically a formalization of how the subdivision of criteria in groups is functional in lowering the number of pairwise comparisons to be performed. For instance, among n criteria, there are

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2} \tag{4.2}$$

unique distinct pairs, leading to 66 pairs to be evaluated, if the number of criteria is 12. Instead, if the criteria are grouped in criteria groups, and pairwise comparison of criteria is performed only among criteria belonging to the same group, the total number of comparisons decreases: in this framework, it is reduced to 19.

(II) the scale in pairwise comparison is set to determine whether a decision maker identify a criteria group/criterion to be more or equally important than another. The scale goes from 1, when two criteria contribute equally to the objective, to 9, when one criterion favors extremely over another, with unitary increments.

(III) there are 19 pairwise comparisons to be performed to evaluate the weights, first among criteria groups (10 comparisons) and then among criteria in the same group (9 pairs).

(IV) since the high amount of criteria weights to be determined, there is a high degree of uncertainty associated. To quantify this uncertainty, a consistency ratio [SP08] is computed. Simply, consistency ratio is used as measure against a threshold value of 10%, hence if the consistency ratio is larger than 0.1, then the subjective evaluation of the pairwise preferences has to be revised.

(V) finally, weights are computed using the AHP structure: the preference comparison among criteria groups gives the weight of each group, then the comparison among criteria in the same group multiplied by the weight of the group is the resulting weight of a criterion. An example to understand better the procedure: suppose that the criteria group *throughput* as weight 0.306, when compared to the other criteria groups; the three criteria in *throughput*, *TPS*, *latency*, and *finalization*, have weights 0.428, 0.428, and 0.144; therefore, the final weights for each criterion contributing to the objective are 0.131, 0.131, and 0.044, respectively.

Note that the process to determine subjective weights requires the intervention of experts in the field. For this thesis, as reported in [Fil+22], the final weights are the arithmetic mean of the evaluations coming from 6 experts.

4.4 Data acquisition

Data acquisition is a multi-step process that involves gathering information from various online sources such as whitepapers, DLT explorers, and academic studies [Pau+19; Xia+20; NL20]. The sources were selected based on their relevance and reliability, and the data was then compared and analyzed to ensure consistency. In some cases, where data was not available from the primary sources, we had to make estimations based on indirect information or by giving an educated guess. The collected data was condensed and tabulated (see Table 4.3) to provide a comprehensive understanding of DLT platforms according to their performance in terms of throughput, decentralization, incentivization, sustainability, and security.

4.5 Consensus family selection

The case study selected in the context of the HD3FLAB was related to a DLT-based bike renting system. While it is not relevant in this circumstances to analyze deeper the process leading to the selection of these values for the criteria to be used in the MCDM problem, the constrains and requirements necessary for the proper functioning of the analysed case are listed in Table 4.1. The table of requirements

Criteria group	Criteria	Required value
Throughput	Transactions per second	500 TPS
	Transaction latency	10 s
	Finalization	Deterministic
Decentralization	Number of consensus nodes	10
	Number of network nodes	10
Incentivization	Transaction fees	0 $
	Reward	0 $
Sustainability	Power consumption	Low
	Hardware dependency	No
Security	Fault-tolerance	50%
	51% attack	Not specified
	Double spending	Not specified

Table 4.1: Requirements for each criteria for the problem posed by the case study.

sets some peculiar features for the selected platform. About the throughput, the transactions per second need to be 500 TPS, the transaction latency 10 seconds, and a deterministic finalization. Decentralization depicts a small-sized blockchain platform, with 10 consensus and network nodes. There is no incentivization, since the stakeholders are assumed to participate in the network for their own interest. Sustainability requires a low power consumption and no hardware dependency. Lastly, fault-tolerance is 50%, with no specific requirement on other security features.

To obtain weights to use in the MCDM process, the pairwise weights analysis is submitted to 6 experts in the field and the taken values are the arithmetic means of the presented evaluations, as showed in Table 4.2.

Now, by using the TOPSIS method, results can be obtained in this scenario. The following is the ranking of consensus protocol family, with the associated DLT platforms in parentheses, for the considered case study:

Criteria group	Criteria	Final weight
Throughput **0.449**	Transactions per second	0.150
	Transaction latency	0.150
	Finalization	0.150
Decentralization **0.084**	Number of consensus nodes	0.042
	Number of network nodes	0.042
Incentivization **0.028**	Transaction fees	0.014
	Reward	0.014
Sustainability **0.233**	Power consumption	0.117
	Hardware dependency	0.117
Security **0.206**	Fault-tolerance	0.123
	51% attack	0.041
	Double spending	0.041

Table 4.2: Obtained weights combined by arithmetic mean from different experts.

1. DPoS (EOSIO)
2. PoS + BFT (Avalanche)
3. BFT (Cosmos)
4. PoI (NEM)
5. FBA (Ripple)
6. FBA (Stellar)
7. PoS (Algorand)
8. dBFT (NEO)
9. PoS (Ouroboros)
10. Tangle (IOTA)
11. PoA deterministic (POA)
12. PoS (Nxt)

13. PoET deterministic (Sawthoot)
14. PoA probabilistic (POA)
15. PoET probabilistic (Sawthoot)
16. PoC (Burstcoin)
17. PoW (Ethereum)
18. PoW (Bitcoin)

These results partially justifies the choice of implementing a BFT consensus protocol in the development of the infrastructure related to the national project HD3FLAB. Indeed, the among top 3 choices there are two platforms using a BFT-based algorithms as underlying consensus protocol.

In summary, the MCDM framework here presented was used to help the decision makers to determine what consensus protocol is the most suitable in the scenario defined by the case study of the national project HD3FLAB. The methodology elaborated to develop this MCDM framework includes the criteria used to characterize the qualities of consensus protocols employed in the DLT industry. This methodology also includes a AHP applied to evaluate the weights to be utilized utilized in order to solve the multi-objective optimization problem. Finally, the ranking of the consensus protocols for the intended scenario is obtained, revealing that the choice of BFT-based consensus protocols is supported by the application of the proposed MCDM framework, under the requirements and conditions specified by the case study.

Table 4.3: The collected data set. The first three columns show the name of the protocol, its consensus family according to [NL20], and the platforms considered. Then, the five criteria with their own metrics are presented: columns or specific values reported with a * have been calculated, values marked with † were guessed or extrapolated by mean of other information.

			Throughput			Decentralization		Incentivization		Sustainability		Security		
Family	Protocol	Platform	TPS (s)	Transaction latency (s)	Finality	Consensus nodes	Network nodes	Fees (USD/tx)	Reward* (USD/day)	Power consumption	Hardware dependency	Fault-tolerance	51% attack	Double-spending
PoW	Nakamoto	Bitcoin	7	600 - 3600	P	1000000	10000	10 - 15	51118992	High	Yes	50% comput. power	Vulnerable	Vulnerable
PoW	Nakamoto-GHOST	Ethereum	10 - 15	10 - 60	P	> 100000	8359	10 - 15	36435200	High	Yes	50% comput. power	Vulnerable	Vulnerable
PoS	ProximaX	Nxt	100	20 - 60	P	117	1371	0.03 - 0.06	16524	Low	No	50% deposited stake value	Vulnerable	Difficult
PoS	Ouroboros	Cardano	1000	20	P	368226	1955	0.2 - 0.3	3132000	Low	No	50% token wealth	Vulnerable	Difficult
PoS	Algorand	Algorand	1000	1 - 5	D	100	100	< 0.01	800000	Low	No	33% token wealth	Vulnerable	Difficult
BFT	Tendermint	Cosmos	1000 - 4000	6 - 7	D	> 10000	< 20	< 0.5	811164	Low	No	33% token wealth	Vulnerable	Difficult
DPoS	EOSIO	EOS	4000 - 6000	126	D	86371	14	0	23100	Low	No	33% delegates	Vulnerable	Vulnerable
PoC	PoC	Burstcoin	10	60 - 120	P	> 10000	977	< 0.01	76	Medium	Yes	50% storage space	Vulnerable	Vulnerable
PoI	PoI	NEM	4000	60	P	100	403	0.015 - 0.35	733	Low	No	50% importance	Safe	Safe
PoA	PoA	POA	60	5	P	12	12†	< 0.01	1730	Low	Yes	50% IDs	Safe	Vulnerable
PoA	PoA	POA	60	5	D	12	12†	< 0.01	1730	Low	Yes	33% IDs	Safe	Safe
PoET	PoET	Hyperledger Sawtooth	7	124	P	15	15	0	0	Low	Yes	50% TEEs	Safe	Safe
PoET	PoET	Hyperledger Sawtooth	7	124	D	15	15	0	0	Low	Yes	33% TEEs	Safe	Safe
dBFT	dBFT	NEO	1000	15 - 25	D	7	< 100	0	336000	Low	No	33% participants	Vulnerable	Vulnerable
FBA	RPCA	Ripple	1500	3 - 5	D	> 100	884	< 0.01	560	Low	No	20% nodes in each UNL	Safe	Safe
FBA	SCP	Stellar	1000	5	D	65	44	< 0.01	0	Low	No	variable (33% best case)	Safe	Safe
-	Tangle	IOTA	300	10 - 15	P	100000†	25	0	0	Low	No	50% comput. power	Safe	Safe
PoS + BFT	Avalanche	Avalanche	4500	< 1	P	834	834†	< 0.01	7506	Low	No	33% participants†	Safe	Safe

Chapter 5

Performability evaluation of BFT protocols

This part of the thesis is devoted to present, analyze, and discuss the proposed models employed to assess the performability of BFT-based protocols. A separate exposition of availability and performability model (even though the latter includes the earlier) is due to the actual evolution of the idea behind these two models. Indeed, the availability model was developed to effectively account for the presence of Byzantine nodes in the system, aspect that was otherwise overlooked in literature. This availability model is, then, used in tandem with a more familiar quasi-birth-death model to describe the arrival and service of messages. The union of these two model made possible to present a complete analytical model, that comprises the possibility of studying and evaluating the performability of BFT protocols.

5.1 Availability model for BFT protocols

Here it is proposed a model based on a CTMC to describe systems in which participants achieve consensus through a BFT process. The relationship between the number of participants, the breakdown and repair rates are investigated to find system configurations for the optimal availability [Mar+23b].

The advantage of the analytical approach resides in the possibility of tuning the parameters characterizing the modelled system in a straightforward and inexpensive way. An evident downside is the difficult beforehand development of an analytical model suitable to describe the network. Nonetheless, some analytical approaches, like CTMC, have been widely and successfully applied in the last few decades to evaluate availability of complex systems [GL87; Bol+06].

Consider a system with N participants entrusted to work on certain tasks by exchanging messages with each other, either in a point-to-point or in a broadcast fashion, as shown in Figure 1.5. The model is based on a CTMC in the form of a quasi-birth-death process. A quasi-birth-death process is a special case of CTMC, where the parameters ξ and η are the rates at which servers break-down ("death") and are repaired ("birth"), as represented in Figure 5.1.

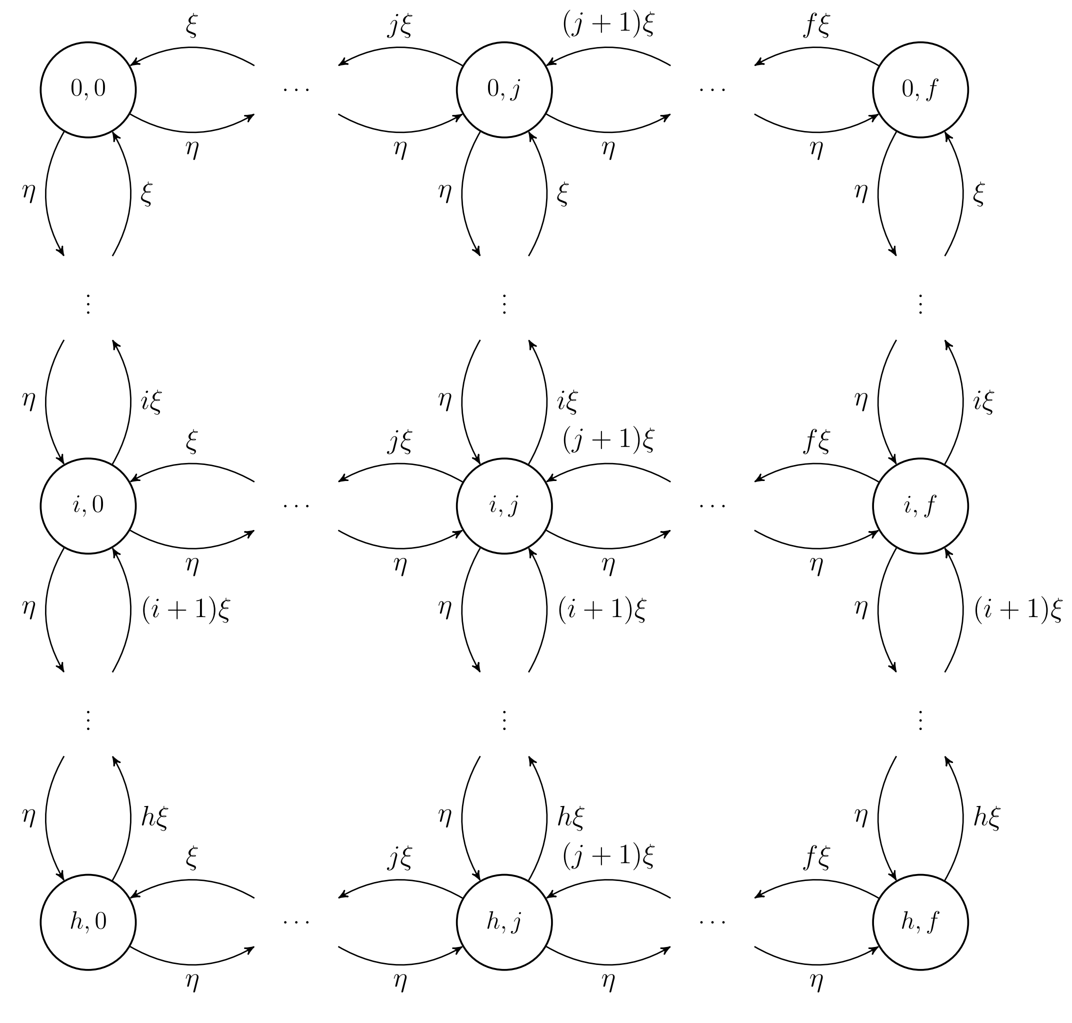


Figure 5.1: Availability model for a BFT consensus protocol.

In this model, it is assumed that servers can break-down independently, but they are repaired sequentially, one at the time. Thus, the break-down rate ξ is multiplied by a number reflecting the current number of available nodes, i.e. if there are f nodes, the break-down rate is $f\xi$, while if there is only one node available ξ is the corresponding break-down rate. This is not the case for the repair process, since repairs can occur only one at a time, with repair rate η.

The model in Figure 5.1 is proposed with the following assumptions[a]: there are N servers in the system, of which $h \leq N$ are nodes participating honestly in network operations, and $f \leq N$ are nodes acting maliciously. In this context, $H : \{h \in \mathbb{N}_0 \,|\, h \leq N]\} \rightarrow \{h \in \mathbb{N}_0 \,|\, h \leq N]\}$ and $F : \{f \in \mathbb{N}_0 \,|\, f \leq N]\} \rightarrow \{f \in \mathbb{N}_0 \,|\, f \leq N]\}$ can be treated as random variables following an arbitrary distribution. H and F are chosen such that their realizations sum up to N, i.e. $H(h) + F(f) = h + f = N$. Therefore, since N is considered to be a constant, H and F are dependent random variables and their outcomes can be written as $f = N - h$ or $h = N - f$. Without loss of generality, F is the independent discrete random variable. Hence, $h = N - f$ is a realization dependent on the value of the discrete random variable F.

The state diagram, as depicted in Figure 5.1, is composed of $(h + 1)(f + 1)$ states. All the states can be eventually visited from any starting point, thus the chain is irreducible and ergodic. These two conditions are sufficient for the chain to admit a stationary distribution.

Therefore, given a set of parameters describing a system, the system's limiting availability can be computed from the stationary probability distribution associated with its CTMC. In order to compute the stationary distribution of the chain, it is needed to establish the generating equations in the form of a system of linear equations, where the state probabilities for given transition rates can be determined. The following equation represents the whole system of linear equations:

$$\begin{aligned}
&[(2 - \delta_{i0} - \delta_{ih} - \delta_{jf})\eta + (i + j)\xi]\, P_{i,j} + \\
&\qquad - \eta\,[P_{i,j-1}(1 - \delta_{j0}) + P_{i-1,j}(1 - \delta_{i0})] + \\
&\qquad - (i + 1)\xi P_{i+1,j}(1 - \delta_{ih}) - (j + 1)\xi P_{i,j+1}(1 - \delta_{jf}) = 0, \quad (5.1)
\end{aligned}$$

where δ_{ij} indicates the Kronecker delta, i.e. $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$. In a compact form, Equation 5.1 describes all the possible equations in the system by varying the indices i and j, where $i \in [0, h]$ and $j \in [0, f]$. Thus, consistently with number of possible states, there are $(h + 1)(f + 1)$ equations to be solved simultaneously. However, because the elements of $\vec{P}$, $P_{i,j}$, are probabilities, the additional condition $\sum_i \sum_j P_{i,j} = 1$ is imposed.

In Equation 5.1, $P_{i,j}$ is the probability that the system is in state (i, j), while the coefficients of $P_{i,j}$s are the entries in a coefficient matrix, $\mathbf{Q}$. Indeed, $\mathbf{Q}$ is the stochastic transition matrix associated with the continuous-time Markov chain, in

[a]For simplicity of exposition, it is assumed that $N, h, f \in \mathbb{N}_0$. Therefore, when dealing with divisions, the *ceiling* $\lceil \cdot \rceil$ and *floor* $\lfloor \cdot \rfloor$ functions are implicitly applied accordingly.

which the transition rates from a state to another are embedded.

Note that, mirroring the lattice structure of the model, it seems natural to write the elements of $\vec{P}$ using the indices i and j, although $\dim \mathbf{Q} = (h+1)(f+1) \times (h+1)(f+1)$, because there are $(h+1)(f+1)$ states in the system. This means that is not proper to use i and j while computing the elements of $\vec{P}$. Indeed, $\vec{P}$ is having a matrix structure when expressed as $P_{i,j}$, therefore it has to be flattened into a vector with elements P_i, where $i \in [0, (h+1)(f+1)]$. This last remark is important, because it ensures that the dimensions of $\vec{P}$ and $\mathbf{Q}$ are matching, since the matrix of coefficients $\mathbf{Q}$ has indices $i, j \in [0, (h+1)(f+1)]$.

Indeed, the simultaneous equations descending from Equation 5.1 can be straightforwardly written in the form $\mathbf{Q}\vec{P} = 0$, where $\mathbf{Q}$ is the coefficient matrix, $\vec{P}$ the vector of unknowns, and 0 the vector of constants (zero). Since common methods for linear algebra, e.g. LU decomposition, can not be applied effectively (or at all) for nearly-singular matrices, the Singular Value Decomposition (SVD) method is used. As a matter of fact, matrix $\mathbf{Q}$ has at least a singular value equal to zero, therefore $\mathbf{Q}$ admits a non-trivial solution to the linear simultaneous equations.

Availability Analysis

In the study originally presenting this model [Mar+23b], a stringent premise regarding the occurrence of Byzantine nodes was adopted. Specifically, it is assumed that the threat level due to Byzantine nodes is either low, medium, or high, with a different number f for each of the three levels. It is clear that, the analysis of the availability may be influenced by the method used to describe the occurrences of Byzantine nodes. Therefore, a new paradigm was presented in [Mar+23a], in which the number of Byzantine nodes f is determined by the random variable F.

Note that the study of the stochastic properties associated with the distribution of the number of Byzantine nodes in the network is not affecting the analytical model used to describe the system. In other words, a separate layer of abstraction is added on top of the availability model, in order to analyse the system in a more general way. This additional abstraction can be considered as an experimental framework, in which the experimenter/decision-maker is testing the system. Given the system parameters (N, η, ξ) and a probability distribution $Pr(F = f) = p(f)$, the methodology to observe is composed by the steps reported in Algorithm 1.

The process described in Algorithm 1 requires a maximum number of nodes to be considered $N_{max} \geq 4$ and rates $\xi, \eta > 0$ such that $\xi/\eta \ll 1$. The process starts setting the value $f = 0$, hence imposing $h = N - f$. The matrix $\mathbf{Q}$ is determined using Equation 5.1, thus $\vec{P}$ can be computed through Singular Value Decomposition. For the pair $(h,\ f)$, compute the probabilities $P_{i,j}$ and arrange them in a matrix $\mathbf{P}$ (vector $\vec{P}$ is reshaped into $\mathbf{P}$ to reflect the two-dimensional

Algorithm 1 Pseudo-code to calculate availability with f as random variable

```
Require: $N, N_{max} \geq 4$ and $\xi, \eta > 0$ and $\xi/\eta \ll 1$
  for $N \leq N_{max}$ do
      $f \leftarrow 0$
      while $f < N/3$ do
          $h \leftarrow N - f$
          $\mathbf{Q} \leftarrow \mathbf{Q}(N, f, h, \xi, \eta)$
          $\vec{P} \leftarrow SVD(\mathbf{Q}, 0)$              ▷ compute state probabilities through SVD
          $A_{h,f} \leftarrow \sum_{i>2N/3}^{h} \sum_{j=0}^{f} P_{i,j}$                          ▷ availability
          $f \leftarrow f + 1$
      end while
      $\bar{A} \leftarrow \sum_{h,f} p(f)\, A_{h,f}$                                ▷ mean availability
  end for
```

structure as in Figure 5.1). Finally, for the resulting set (N, f, h, η, ξ), availability can be calculated. Availability is the cumulative probability that the system is working and it can commit messages. As prescribed in Equation 1.1, the system is available for all the states with $i > 2N/3$, thus the corresponding state probabilities are summed up to calculate the availability:

$$A_{h,f} = \sum_{i>\frac{2N}{3}}^{h} \sum_{j=0}^{f} P_{i,j}. \tag{5.2}$$

At each iteration, f is increased by 1. The algorithm iterates until a predefined value of N_{max} is reached. After the iterative part, there is a resulting collection of $A_{i,j}$s, one for each (N, f, h, η, ξ). Therefore, the mean value of the availability is

$$\bar{A} = \sum_{h,f} p(f)\, A_{h,f}. \tag{5.3}$$

Essentially, the procedure described above compute the mean availability for a system with N servers (subjected to break-down and repair processes at rate ξ and η), where the number of Byzantine actors in the system, f, is deriving from the realizations of a random variable F distributed according to an arbitrary probability distribution (see Table 5.1 for a concise description of the probability distributions used to determine f). This means that the procedure in Algorithm 1 can be iterated over a range of several N and different probability distributions for F. In this way, the behaviour of the system's availability, for different distributions, can be studied as a function of the number of servers. Similarly, to study the relationship between rates ξ, η and availability, the probability distribution for $Pr(F = f)$ can be fixed and then it can be computed the availability of the system at the variation of ξ and η, for different N.

A special attention should be reserved to the analysis of the Poisson distribution. The pmf of Poisson distribution is defined on the positive integers, therefore a truncated version of the pmf is needed to match the domain of definition $[0, N]$ for the occurrence of Byzantine nodes. The right-truncated Poisson distribution [JKK05] is defined as

$$p(x; \lambda, N) = \begin{cases} \frac{\lambda^x}{x!} \left(\sum_{y=0}^{N} \frac{\lambda^y}{y!} \right)^{-1}, & 0 \leq x \leq N \\ 0, & \text{otherwise} \end{cases}$$

that is derived from the definition of Poisson distribution, in which the series representation of the exponential is truncated to N. The mean can be computed from the definition of expected value $\mu = E[X] = \sum_{x=0}^{N} x\, p(x) = \lambda \frac{N\,\Gamma(N,\lambda)}{\Gamma(N+1,\lambda)}$, where $\Gamma(N, \lambda)$ is the incomplete gamma function.

Distribution	**pmf**	**Mean μ**	**Variance σ^2**
Uniform	$p(x; a, b) = \frac{1}{b-a+1}$	$\frac{b+a}{2}$	$\frac{(b-a+1)^2-1}{12}$
Right-truncated Poisson	$p(x; \lambda, n) = \frac{\lambda^x}{x!} \left(\sum_{y=0}^{n} \frac{\lambda^y}{y!} \right)^{-1}$	$\lambda \frac{n\Gamma(n,\lambda)}{\Gamma(n+1,\lambda)}$	-
Binomial	$p(x; n, q) = \binom{n}{x} q^x (1-q)^{n-x}$	nq	$nq(1-q)$
Degenerate	$p(x; x_0) = \delta_{x\, x_0}$	x_0	0

Table 5.1: A summary of the probability distributions used in this work to characterize the occurrences of Byzantine nodes, with $N \in [4, 128]$. pmf indicates probability mass function, μ the mean of each distribution, and σ^2 the variance of the distribution. Parameters for each distributions are specified in the next section, in correspondence of the two comparative results: Figure 6.1 and Figure 6.2.

Regarding the proposed methodology, note that it is vital to choose appropriately the parameters of the arbitrary probability function generating the random values f. While it is out of the scope for this study to determine whether there is an *a priori* restriction on which probability function to use in characterizing the occurrences of Byzantine nodes, it is advisable to properly select the first two moments, i.e. mean and variance, of any chosen distribution. To better explain this, consider the impact that the parameters of the probability distribution have: if the mean μ is outside the interval $[0, N/3)$ and the probability function is narrow (low variance), several zero-valued availability numbers will be sampled; same situation would occur if $\mu \in [0, N/3)$, but the variance is high; an optimal choice, instead, is represented by the distribution not spreading excessively and $\mu \in [0, N/3)$.

Lastly, by applying this methodology, it is possible to recreate the results presented in the aforementioned study [Mar+23b], where a constant number f is selected to reflect a threat level due to the ratio of Byzantine nodes in the system (see Figure 6.3). This validates the observation that, when defining some possible threat levels of the system, the investigator is, indeed, assuming a degenerate distribution for F, i.e. a constant value f representing the number of Byzantine nodes in a system of N nodes.

5.2 Performability model for BFT protocols

This section presents an analytical model, referred here as PBFTPEM (PBFT Performability Evaluation Model), which is based on CTMCs and it aims to compute the performability metrics of PBFT systems using queueing theory.

As in section 5.1, a system with $N \geq 4$ servers is considered - since, with unsigned messages, if $N < 4$ the problem has no solution. In this setting, servers can break-down independently and can be repaired in succession at rate ξ and η, respectively. Jobs, in the form of messages/transactions bundled in blocks of transactions, are handled by the system in order to agree on the validity of the submitted transactions. Hence, once the validity of each transaction is confirmed, the block is committed by all the honest nodes, each in their own memory. Transactions in blocks that can not be served immediately (because the system is busy) are stored in a finite memory buffer (with size J), until the memory buffer is not saturated, i.e. there are no memory slots free. At this point, arriving jobs would not be accepted and they will be lost. Transactions to be processed are modelled as a Poisson process with arrival rate λ, while service/processing time is exponentially distributed with rate μ.

Figure 5.2 exemplifies the state diagram for PBFTPEM. Nodes colored in red represent the area in which consensus is not reached, while nodes in green are those for which the system is available. The three indices, h, f, j, reported in each node are indicating states that the system may occupy.

Parameters of PBFTPEM are: the total number of nodes $N \geq 4$, divided in honest $h \leq H$ and Byzantine nodes $f \leq F$, where $H \in [0, N]$ is the maximum number of honest nodes and $F \in [0, N]$ the maximum amount of Byzantine nodes, such that $H + F = N$; the buffer size $J > 0$ and the number of jobs $j \in [0, J]$ in the system; break-down rate $\xi > 0$, repair rate $\eta > 0$, arrival rate $\lambda > 0$,

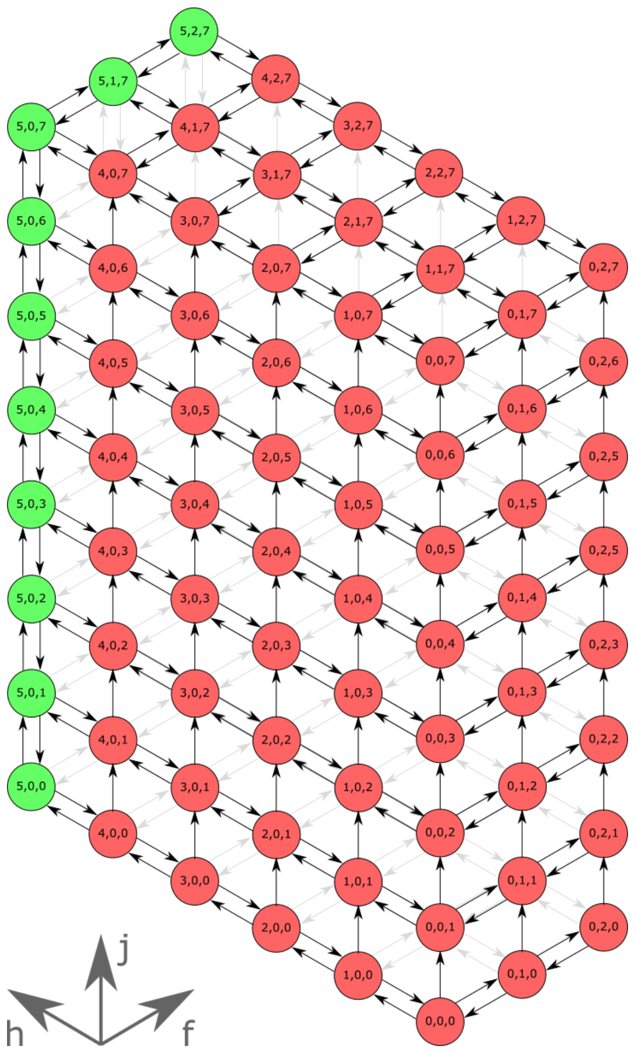


Figure 5.2: Depiction of a PBFT state diagram with $H = 5, F = 2$ (hence, $N = 7$), and $J = 7$. The states in the diagram have indices (h, f, j), where $h \in [0, H]$, $f \in [0, F]$, and $j \in [0, J]$.

service rate $\mu > 0$, and timeout rate $\mu_t > 0$. In PBFTPEM, it is assumed that N, H, h, F, f, J, j are positive integers. Therefore, for simplicity of exposition, when dealing with divisions, the *ceiling* $\lceil \cdot \rceil$ and *floor* $\lfloor \cdot \rfloor$ functions are implicitly applied, accordingly. Differently, all the rates - ξ, η, λ, μ, and μ_t - are positive real numbers. The latter, μ_t, is the rate indicating the process of losing a block of transactions, due to the internal time-out defined by the system. For instance, if a job can not be served before the time-out occurs, it is not committed by the servers that received it and it is lost. This possibility may happen when the mean service time $1/\mu < 1/\mu_t$. There are several reasons why the system might be not able to serve jobs, notably the system can commit blocks only when it is available, i.e.

$$h > 2N/3. \tag{5.4}$$

For instance, in a BFT system, there may be present servers that are not acting accordingly to the rules set by the protocol - called a Byzantine server - either maliciously, either because of malfunctioning. Hence, given a system with N servers, implementations of a BFT protocol tolerate up to $F < N/3$ Byzantine participants, that are acting deliberately in contrast with the network or are being unresponsive. In this formulation, however, unresponsive nodes are treated as broken-down servers, and not necessarily Byzantine.

Because it is assumed that servers can break-down independently, but they are repaired one at the time, the break-down rate ξ is multiplied by a number reflecting the current number of available nodes, i.e. if there are h nodes, the break-down rate is $h\xi$, while if there is only one node available ξ is the corresponding break-down rate. This is not the case for the repair process, since repairs occur only one at a time, with repair rate η.

From the state diagram in Figure 5.2 the balance equations for the system are determined. In a compact way, the balance equations can be written as

$$\begin{aligned}
[(2 - \delta_{hH} - \delta_{fF})\eta + (h+f)\xi]\, P_{h,f,j} + \\
- \eta\,[P_{h-1,f,j}(1-\delta_{h0}) + P_{h,f-1,j}(1-\delta_{f0})] + \\
- \xi\,[(h+1)P_{h+1,f,j}(1-\delta_{hH}) - (f+1)P_{h,f+1,j}(1-\delta_{fF})] + \\
- \lambda\, P_{h,f,j-1}(1-\delta_{j0}) - \mu\, P_{h,f,j+K}(1-\delta_{jJ}) = 0,
\end{aligned} \tag{5.5}$$

where δ_{ij} indicates the Kronecker delta, i.e. $\delta_{ij} = 1$ if $i = j$ else $\delta_{ij} = 0$ if $i \neq j$. K indicates the possibility of bundling jobs in batches, hence serving all at once, up to a number of K transactions. Hence, compactly, Equation 5.5 describes all the $(H+1)(F+1)(J+1)$ equations needed to describe transitions in the state diagram. Although, the condition that elements in $\vec{P}$ are probabilities imposes that

$$\sum_{j=0}^{J}\sum_{f=0}^{F}\sum_{h=0}^{H} P_{h,f,j} = 1. \tag{5.6}$$

Using the balance equations, for instance, the stationary probability distribution of the states in the system can be determined, hence the performability metrics. Therefore, because the stationary distribution of state probabilities $\vec{P}$ is to be found, the idea is to solve the matrix equation $\mathbf{Q}\vec{P} = 0$, where $\mathbf{Q}$ the coefficient matrix of the balance equations, i.e. the stochastic transition matrix associated with the CTMC. As for the same reasons as in section 5.1, SVD method is applied to compute the vector of probabilities.

However, in order to simplify the calculations and to avoid the state explosion (thus handling a tractable problem), it can be noted that PBFTPEM may be effectively divided into the product-form of two subsystems [CM83]. This means that the state diagram presented in Figure 5.2 can be decomposed and represented as two independent processes. Figure 5.1 and Figure 5.4 are the state diagrams of the two subsystems. Here, Figure 5.3 is a single layer on the plane (h, f) of Figure 5.2, while Figure 5.4 are chains parallel to the axis j of the same graphic. Notice that the scheme reported in Figure 5.3 has been already presented and analysed thoroughly in section 5.1.

Diagrams in Figure 5.4 present two distinct possibilities as serving policy: the systems may process single transactions (Figure 5.4a), one at the time, or it can bundle them in blocks of transactions (Figure 5.4b). While the former is a simple $M/M/1$ queue [Ken53], the latter is structured as a partial bulk/batch service queue, or $M/M^K/1$, where K is the maximum size of the batch (number of transactions in the block). Which model to use is up to the application under consideration and results will, in general, differ.

Performability Analysis

The process to assess the performability metrics (see Algorithm 2 for an algorithmic presentation of the process) requires to define $N \geq 4$, $J > 0$, and rates $\xi, \eta, \lambda, \mu, \mu_t > 0$. According to which one is considered as a free parameter, either H or F, the matrix $\mathbf{Q}$ is determined using Equation 5.5, and the solution $\vec{P}$ is computed through SVD. Elements in $\vec{P}$ are the stationary state probabilities of the system, and using these probabilities, important metrics associated with the system can be calculated. In this work, the following metrics can be computed: system availability, blocking probability, throughput, mean queue length, and latency. Although these metrics were introduced in subsection 2.2.1, they are reported also in this section, so that the given definitions can be adapted to fit the obtained probabilities $P_{h,f,j}$.

Similarly as for section 5.1, PBFTPEM allows to compute the availability of the system as

$$A = \sum_{j=0}^{J} \sum_{f=0}^{F} \sum_{h>2N/3}^{H} P_{h,f,j}. \tag{5.7}$$

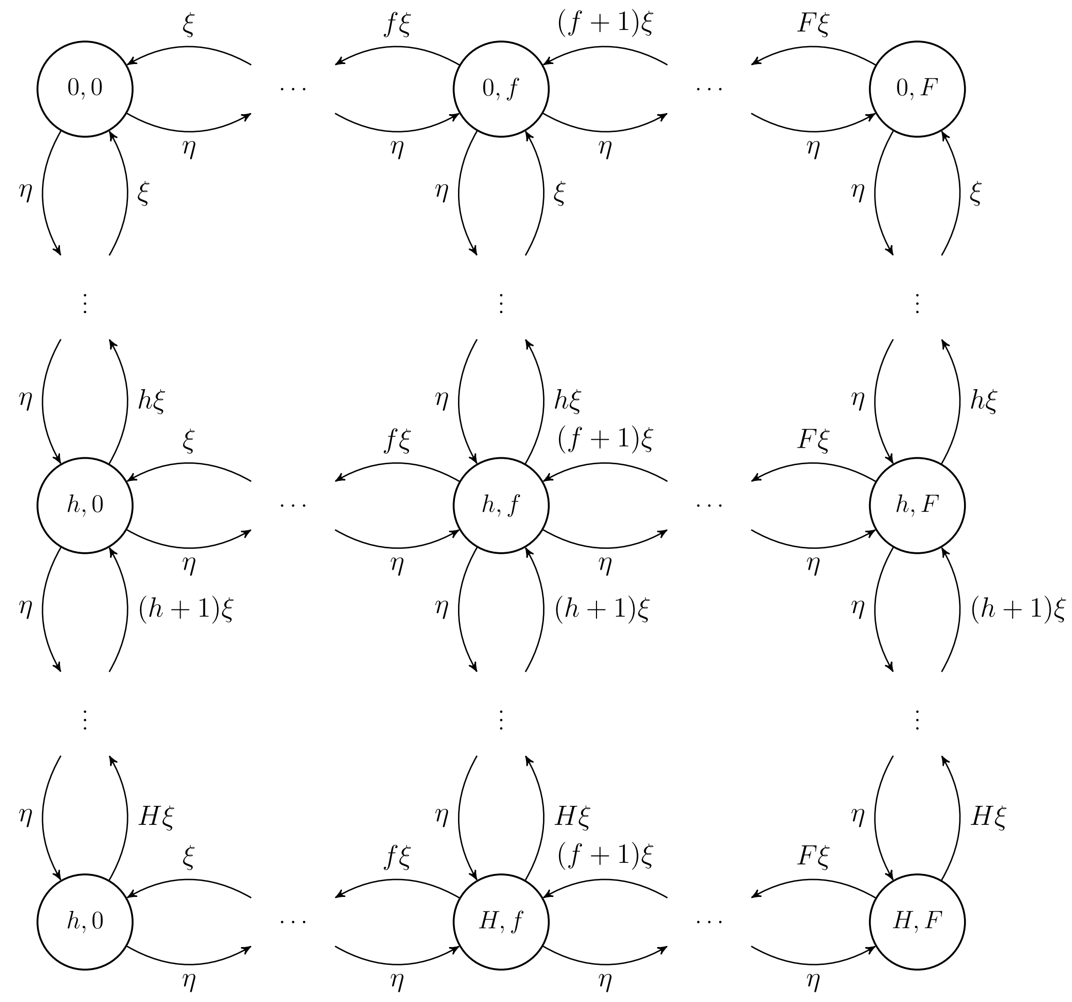


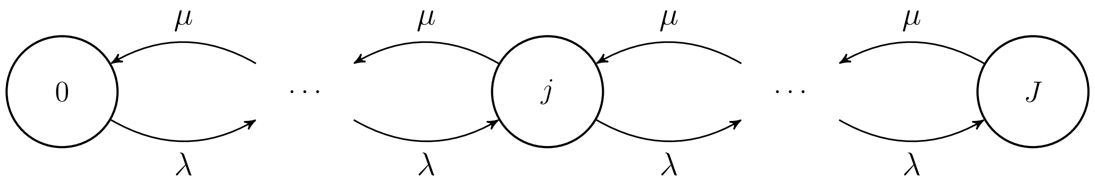
Figure 5.3: One of the two components constituting PBFTPEM for PBFT consensus protocol. This state diagram can be regarded as an availability model.

Then the blocking probability is expressed as

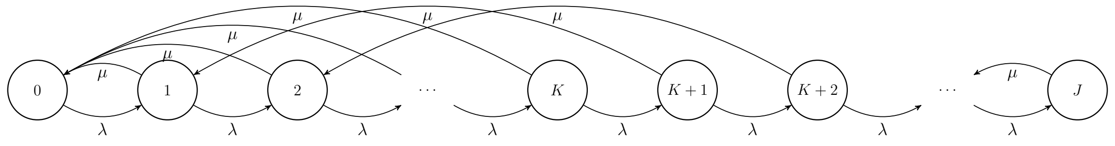
$$blocking\_probability = \sum_{f=0}^{F}\sum_{h=0}^{H} P_{h,f,J}, \tag{5.8}$$

which tells that if $j > J$, any incoming transaction is lost because of full memory buffer.

Throughput can be viewed as the amount of jobs being served by the system



(a) Queue with sequential service.



(b) Queue with bulk/batch service.

Figure 5.4: One of the two components constituting PBFTPEM for PBFT consensus protocol. These two alternative state diagrams describe the processes of jobs arrival and service.

Algorithm 2 Pseudo-code to calculate performability metrics

```
Require: $N \geq 4$ and ($H \in [0, N]$ or $F \in [0, N]$) and $J > 0$ and $\xi, \eta, \lambda, \mu, \mu_t > 0$
    $F \leftarrow N - H$                                        ▷ or $H \leftarrow N - F$
    $\mathbf{Q} \leftarrow \mathbf{Q}(H, F, J, \xi, \eta, \lambda, \mu, \mu_t)$      ▷ generate the matrix of coefficients
    $\vec{P} \leftarrow SVD(\mathbf{Q}, 0)$                     ▷ compute state probabilities through SVD
    $A \leftarrow \sum_{j=0}^{J} \sum_{f=0}^{F} \sum_{h>2N/3}^{H} P_{h,f,j}$      ▷ availability
    $bp \leftarrow \sum_{f=0}^{F} \sum_{h=0}^{H} P_{h,f,J}$                    ▷ blocking probability
    $thr \leftarrow \mu \sum_{j=1}^{J} \sum_{f=0}^{F} \sum_{h>2N/3}^{H} P_{h,f,j}$  ▷ thoughput
    $mql \leftarrow \sum_{j=0}^{J} \sum_{f=0}^{F} \sum_{h=0}^{H} j\, P_{h,f,j}$       ▷ mean queue length
    $lat \leftarrow mql/thr$                                   ▷ latency
```

in the unit time, and computed using

$$throughput = \mu \sum_{j=1}^{J} \sum_{f=0}^{F} \sum_{h>2N/3}^{H} P_{h,f,j}, \tag{5.9}$$

where $h > 2N/3$ indicates that the system can serve jobs only if there are enough available machines, i.e. their number h is greater or equal than the quorum.

The mean queue length can be determined by enumerating

$$mean\_queue\_length = \sum_{j=0}^{J} \sum_{f=0}^{F} \sum_{h=0}^{H} j\, P_{h,f,j}, \tag{5.10}$$

Lastly, the latency of the system is

$$latency = \frac{mean\_queue\_length}{throughput}. \tag{5.11}$$

As a last remark, note that the procedure described above (and summarised in Algorithm 2) can be iterated over a range of Ns, such that the relation between performability metrics and number of nodes N is explored. Similarly, this methodology allows the investigators to study the connection between different aspects of the system under examination, simply by variating the parameters of interest.

Chapter 6

Results

This chapter shows the results obtained by applying the described methodologies and models to evaluate the availability and performability of BFT-based protocols. As stated in chapter 5, there is a separate presentation for the availability and performability models, leading to a separate showcase of the results. Note that results pertaining the performability model include a sensitivity analysis to evaluate the impact of variations in the input parameters onto the resulting metrics.

6.1 Evaluation of availability for BFT protocols

In this section, results are provided to show the effects of various distributions of the Byzantine faults on availability.

Figure 6.1 shows the effects of four different probability distributions for the random variable F (as from Table 5.1) on mean system availability for $N \in [4, 128]$ and $\xi/\eta = 0.015$. Different lines represent a separate choice of a probability distribution for the value of f. Parameters of the uniform distribution are $a = 0$ and $b = N$. $\lambda = N/6$ is used for the right-truncated Poisson distribution, in which $\Gamma(n, \lambda)$ is the incomplete gamma function. For the binomial distribution $n = N$ and $q = 1/6$, where $\binom{n}{x}$ is the binomial coefficient. Lastly, the degenerate distribution uses the Kronecker delta $\delta_{x\,x_0}$, with $x_0 = N/6$. In the figure, the uniform distribution has the mean $\mu = N/2$, while all the other distributions have the mean $\mu = N/6$, the center of the interval $[0, N/3)$. The figure shows that, for different choices of the probability distribution of F, there is a distinctive behaviour of the mean availability. This behaviour varies between the worst-case scenario, which can be observed when the random variable F is drawn from a uniform distribution to the best case, where a degenerate distribution with constant value $f = N/6$ is used. However, this configuration for the uniform distribution is expected to give the worst-case scenario, since the mean of the distribution is centered around the middle of the interval for the values of N, while the other distributions are centered around $N/6$.

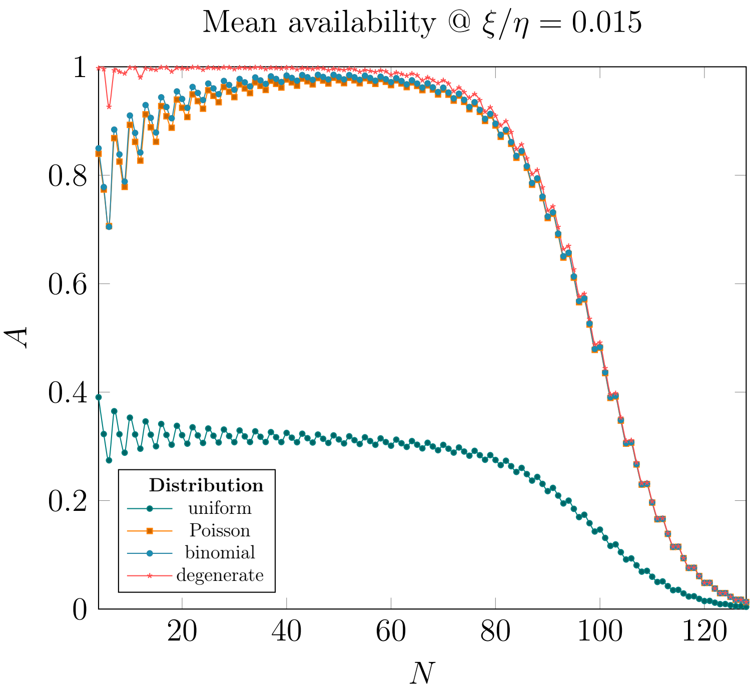


Figure 6.1: Availability as a function of the number of servers and fixed ratio ξ/η.

Figure 6.2 presents the behaviour of the mean system availability for $N \in [4, 128]$ and $\xi/\eta = 0.015$, when the mean of each probability distribution is $\mu = N/2$. In this figure, different lines represent a separate choice of probability distribution for the value of f. Parameters of the uniform distribution are $a = 0$ and $b = N$. For the right-truncated Poisson distribution, in which $\Gamma(n, \lambda)$ is the incomplete gamma function, $\lambda = N/2$ is used. The binomial distribution has $n = N$ and $q = 1/2$, where $\binom{n}{x}$ is the binomial coefficient. As expected, the degenerate distribution, when $f = N/2$, gives availability that is constantly zero, therefore it is not reported. In this graph, the best case is the one in which the uniform distribution is employed, while the worst case occurs when the binomial distribution describes the occurrence of Byzantine nodes in the system. Differently from Figure 6.1, with this configuration, the uniform distribution is clearly the distribution giving the best result in Figure 6.2. This is because the probability to get a value $f < N/3$, such that the quorum is reached, is higher for the uniform distribution than for the other distributions. This is simply because, while the mean is the same for the selected distributions, the variance of the possible values of f is larger for the uniform distribution, hence there is a higher probability to select a value f satisfying the quorum.

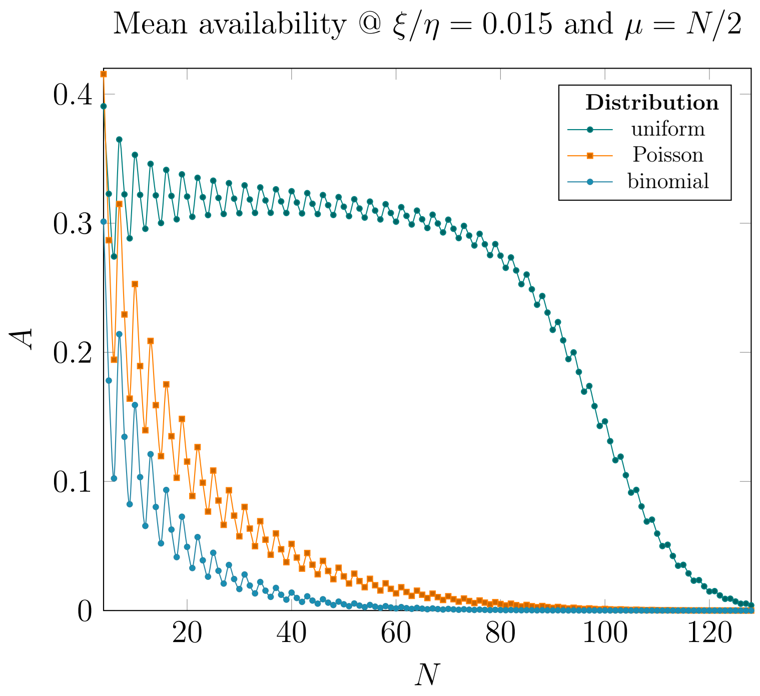


Figure 6.2: The variation in system availability as a function of the number of servers and fixed ratio ξ/η.

Figure 6.3 presents an example of system availability trend for different ratios of ξ/η. Here F is distributed according to the degenerate distribution centered around the value $f = N/6$. The plot shows how the availability of the system is degrading when the ratio ξ/η is increasing.

Please note that the values of availability are not represented by a smooth line because some numbers for N correspond to optimal configurations of BFT systems. For instance, any N satisfying the equation $(N \mod 3) = 1$, $N \geq 4$, produces a system with better availability than the ones generated by $N-1$ and $N-2$, e.g., the value of availability when $N = 16$ is higher than when $N = 15$ or $N = 14$.

In summary, this study proposes an analytical availability model which is critical for the evaluation of fault-tolerant multi-server systems. A model based on continuous-time Markov chains is used to analyse the availability of BFT systems due to break-downs and repairs, when malicious nodes are present. The total number of nodes considered changes between 4 and 128, while the proportion of malicious nodes is considered to be a random variable distributed according to a set

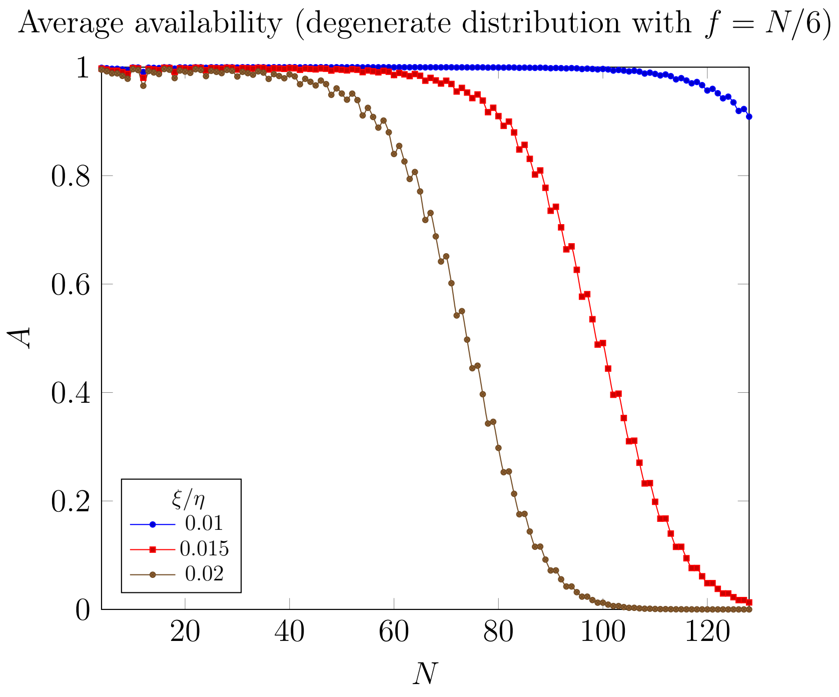


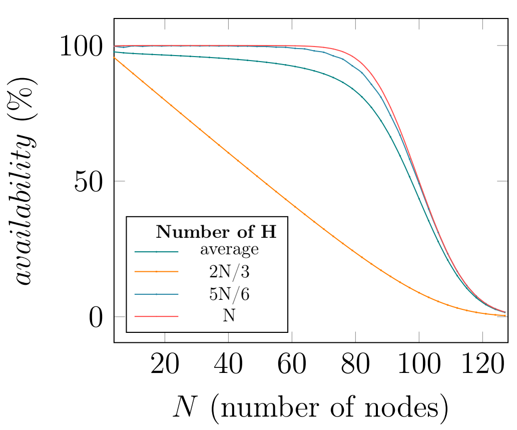
Figure 6.3: System availability as a function of the number of servers with ratio $\xi/\eta = 0.01, 0.015, 0.02$.

of probability distributions. Numerical results are presented reporting availability as a function of the number of participants and a relative number of honest actors in the system for different selected probability distributions. The contribution of this work is to extend the work in [Mar+23b] on availability calculation in order to take into account the presence of malicious nodes in a non-deterministic fashion. From the model, it can be concluded that there is a non-linear relationship between the number of servers and availability, it is inversely proportional to the number of nodes in the system, similarly for each distribution of f tested. This relationship is further strengthened as the ratio of break-down rate to the repair rate increases.

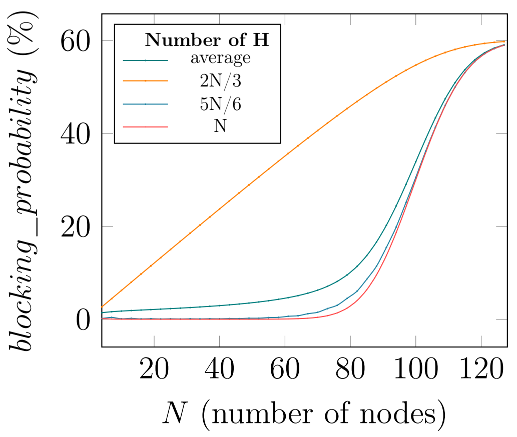
6.2 Evaluation of performability for BFT protocols

In this section, results obtained by using the presented PBFTPEM method are shown. These results evaluate the effects of distinct parameters, e.g. number of servers and Byzantine nodes, on the performability metrics of PBFT systems.

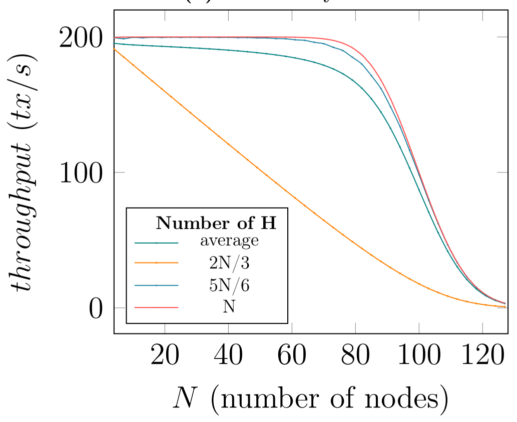
Graphs in Figure 6.4 show how the number of servers, $N \in [4, 127]$, and the ratio of Byzantine nodes in the system are effecting the performances and avail-



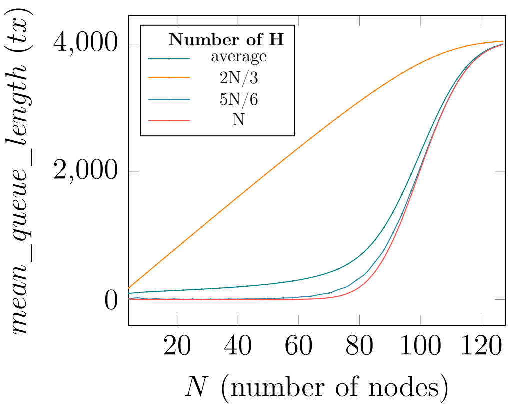
(a) Availability.



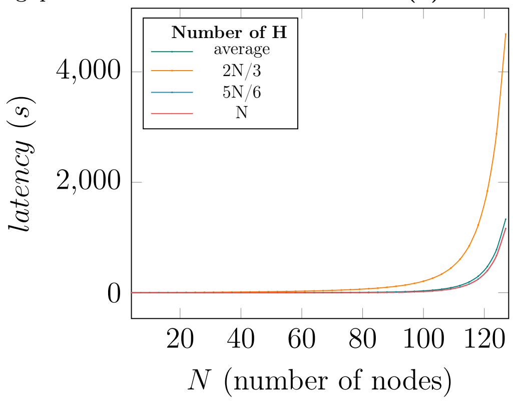
(b) Blocking probability.



(c) Throughput.



(d) Mean queue length.



(e) Transaction latency.

Figure 6.4: Performability metrics as a function of the number of servers, where $J = 4096$, $K = 100$, $\xi = 5.02 \cdot 10^{-7}$, $\eta = 3.47 \cdot 10^{-5}$, $\lambda = 250$, $\mu = 1000$, and $\mu_t = 1$.

ability of a PBFT system. In this context, Byzantine nodes are determined by counting the amount of honest nodes, $H = N, 5N/6, 2N/3$ and the average value. Parameters used for the computation of these metrics are: $J = 4096$, $K = 100$, $\xi = 5.02 \cdot 10^{-7}$, $\eta = 3.47 \cdot 10^{-5}$, $\lambda = 250$, $\mu = 1000$, and $\mu_t = 1$. This collection of results reported in Figure 6.4 displays a characteristic behavior in the performability of the system. There is a marked reduction in performance as the number of servers increases, with a seemingly threshold at $N \approx 60$, and performance worsen when the number of Byzantine nodes increment. As it might be expected, metrics are related and they show similar behaviour in pairs: Figure 6.4a and Figure 6.4c; Figure 6.4b and Figure 6.4d; Figure 6.4e is not coupled.

In addition to test the relation between number of servers and Byzantine nodes, PBFTPEM can be applied to to study the behaviour of the performability metrics due to an higher break-down rate to repair rate ratio (ξ/η). The results obtained from this analysis are shown in Figure 6.5. It can be noted that the number of servers at which the performances of the system are sharply degrading is shifted to the left, at $N \approx 40$. Besides for this shift in performance, the other considerations made for the results in Figure 6.4 are, otherwise, applying also to Figure 6.5. Likewise, the results pertaining the availability metric (Figure 6.4a and Figure 6.5a) are matching the ones obtained in section 6.1 and [Mar+23b; Mar+23a].

In Figure 6.6 there are results, which are replicating - at least in their outline and approximate values - two studies found in literature. In particular, Figure 6.6a reproduce the values of throughput presented in [Tan+22]. However, the analytical result (Figure 6.6a) does not have an arched shape and a reduced decrease in the value of throughput at $N \approx 80$. Similarly, Figure 6.6b is presenting results similar to the values of transaction latency obtained in the benchmark reported in [Liu+22]. In this case, the shape of the two graphs is matching, except from irregularities in the plot found in literature. The parameters used to obtain these results are: $J = 4096$, $K = 100$, $\xi = 3.47 \cdot 10^{-7}$, $\eta = 3.47 \cdot 10^{-5}$, $\lambda = 2600$, $\mu = 10000$, and $\mu_t = 1$. Here, it is assumed that the number of Byzantine nodes is $F = N/3$, hence the honest nodes are $H = 2N/3$.

Figure 6.7 presents a comparison between the benchmark of Tendermint [BKM18] reported in [Fu+20] and the values obtained from the performability analysis. It can be noted that, while the trend of the data is reproduced by the analytical results, it fails to match consistently the values in the error interval given by the published data. The parameters used to obtain the analytical results are: $J = 4096$, $K = 3000$, $\xi = 6.593 \cdot 10^{-7}$, $\eta = 3.47 \cdot 10^{-5}$, $\mu = 0.5$, and $\mu_t = 0.2$. Here, it is assumed that the number of Byzantine nodes is $F = N/4$, hence the honest nodes are $H = 3N/4$.

In summary, from the results presented above, it can be concluded that all system's performability metrics are indeed non-linearly dependent on the number of the servers in the network. For instance, favorable metrics (availability and throughput) are decreasing at the increase of N, while the values of disadvantageous metrics (blocking probability, mean queue length, and transaction latency) are increasing. This tendency results strengthened when the rate of break-downs increases over the rate of repairs, i.e. performance decrease at the increase of ξ, or the decrease of η. Concerning the correspondence between the analytical results and data obtainable from the literature, there is indeed a certain degree of agreement in the general trend, but analytical results fail to match consistently benchmark data in the interval error provided.

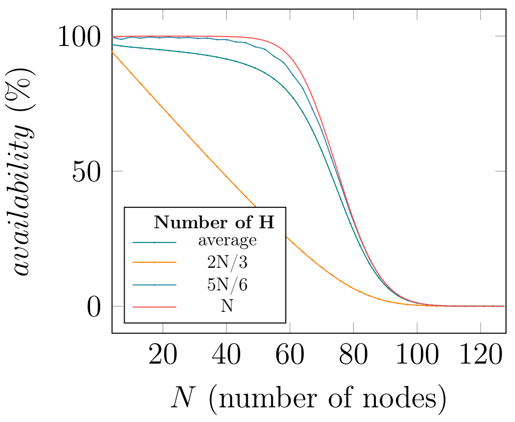
6.2.1 Sensitivity analysis

This section of the results is devoted to the presentation of a sensitivity analysis of PBFTPEM. The performed sensitivity analysis studies the percentage variation of the results from a reference value by changing one parameter at the time. In practice, for each parameter, the value from the studied case (Figure 6.7) is used as reference, then by varying the considered parameter keeping fixed the others, the output variation on the performability metrics can be analysed.

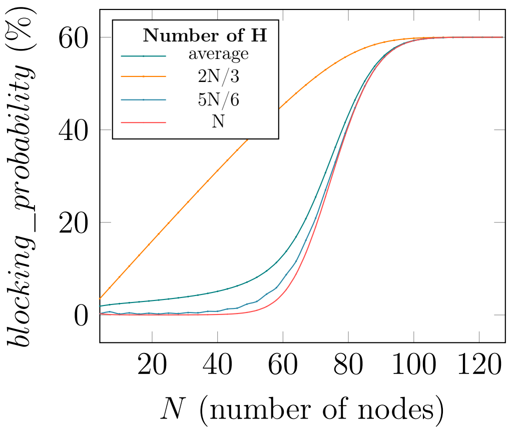
Ranges for each variable are taken such that the reference value is the median of the definition interval, i.e. $\xi \in \left[0, 2\,\xi^{\text{ref}}\right]$, $\eta \in \left[0, 2\,\eta^{\text{ref}}\right]$, $\lambda \in \left[0, 2\,\lambda^{\text{ref}}\right]$, and $\mu \in \left[0, 2\,\mu^{\text{ref}}\right]$. Note that the reference values for the parameters are chosen to be reflecting the configuration used to compare the throughput from Tendermint (Figure 6.7), since it replicates the results from a benchmark, those values for the parameters of PBFTPEM appear to be reasonable.

The results of the sensitivity analysis shown in Figure 6.8 are depicting a predictable scenario, in which (most of) the performability metrics obtainable from PBFTPEM are highly non-linear for small variations of the input parameters. Results also identify the correlation between parameters and which metrics they do influence. For instance, break-down and repair rates influence all the metrics in a highly non-linear fashion (Figure 6.8a and Figure 6.8b). Conversely, arrival rate does not influence availability and latency, while it influences the blocking probability in a seemingly-exponential way, and almost linearly determine the throughput and mean queue length (Figure 6.8c). In a similar way, the service rate is not affecting the availability of the system and its throughput (after passing the reference point), but it does respond not linearly for the other performability metrics (Figure 6.8d).

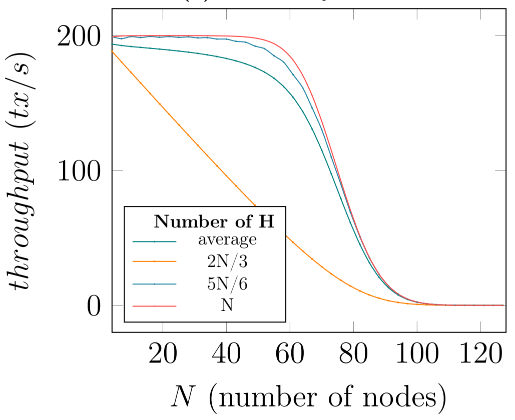
However, the behaviour of the performability metrics in the case of arrival and service rate is understandable if considering that for a system in statistical equilibrium, if $\mu \gg \lambda$, the effective throughput coincides with the arrival rate (Figure 6.8c), with no change in latency and other metrics having an “orderly” trend. Otherwise, if $\mu \leq \lambda$, the system is saturated and the throughput follows (damped) the service rate, while the other metrics (except for availability) are non-linearly decreasing, passing from higher to lower values than the reference, when $\mu \gg \lambda$.



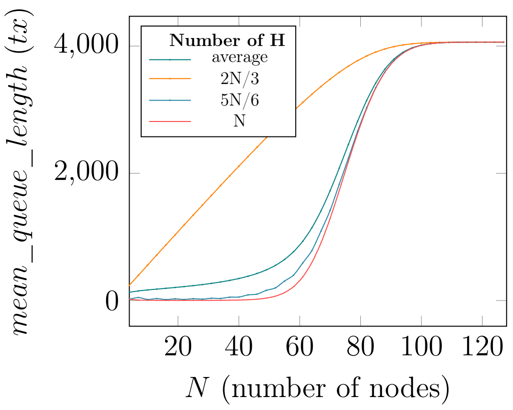
(a) Availability.



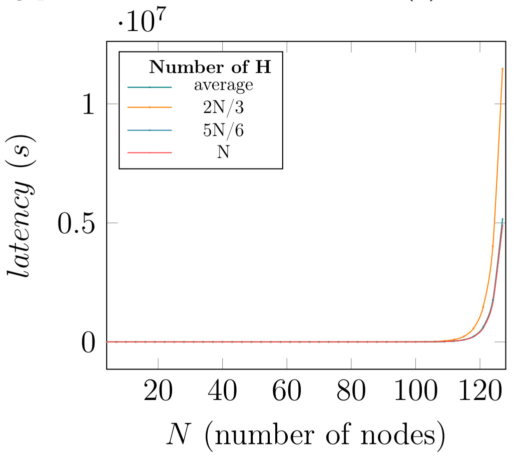
(b) Blocking probability.



(c) Throughput.

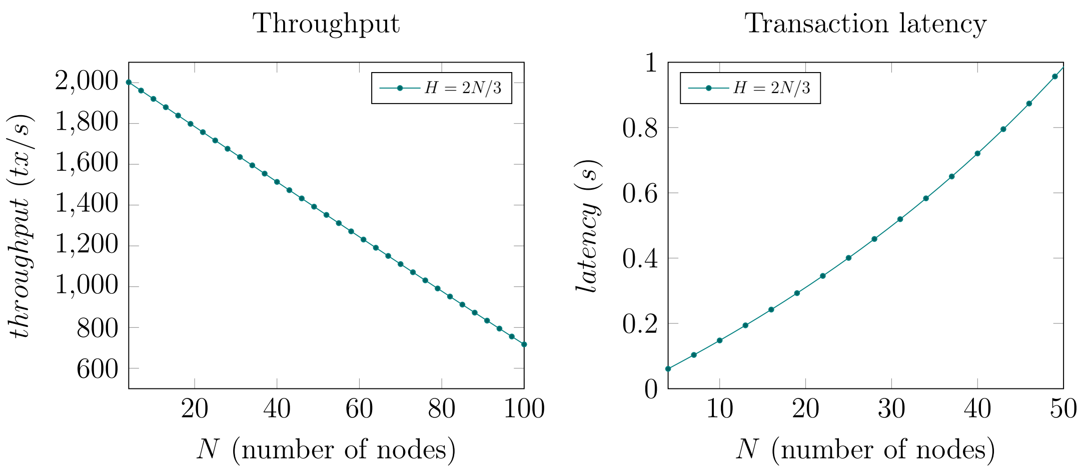


(d) Mean queue length.



(e) Transaction latency.

Figure 6.5: Performability metrics as a function of the number of servers, where $J = 4096$, $K = 100$, $\xi = 6.94 \cdot 10^{-7}$, $\eta = 3.47 \cdot 10^{-5}$, $\lambda = 250$, $\mu = 1000$, and $\mu_t = 1$.



(a) Throughput. **(b)** Transaction latency.

Figure 6.6: Throughput and latency as a function of the number of servers, where $J = 4096$, $K = 100$, $\xi = 3.47 \cdot 10^{-7}$, $\eta = 3.47 \cdot 10^{-5}$, $\lambda = 2600$, $\mu = 10000$, and $\mu_t = 1$.

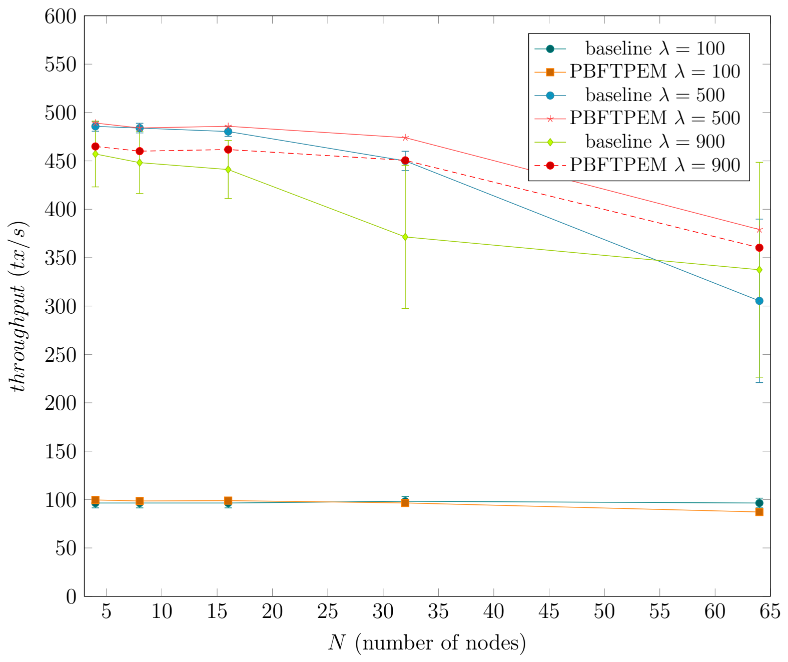
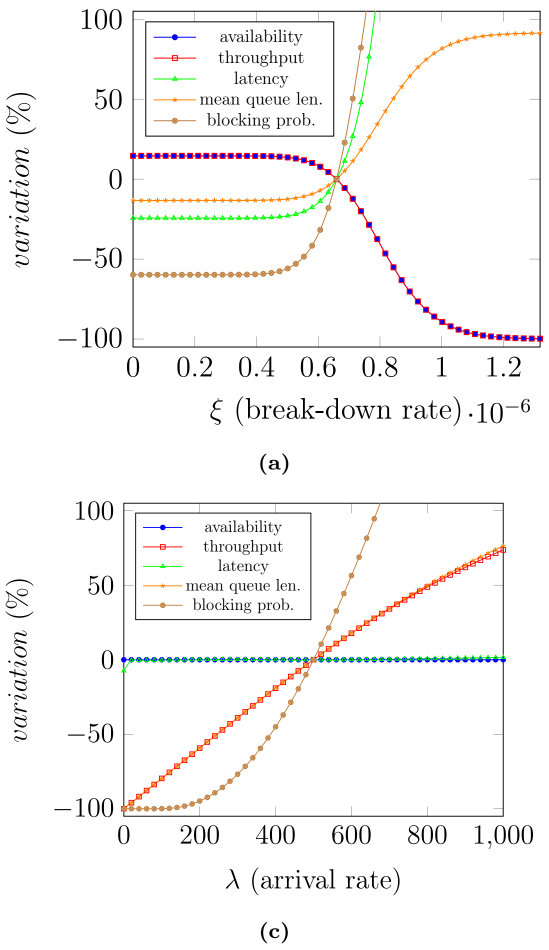


Figure 6.7: Comparison between the throughout of Tendermint from literature [Fu+20] and values of throughput obtained from the performability analysis. The parameters used in PBFT-PEM are $J = 4096$, $K = 3000$, $\xi = 6.593 \cdot 10^{-7}$, $\eta = 3.47 \cdot 10^{-5}$, $\mu = 0.5$, and $\mu_t = 0.2$.



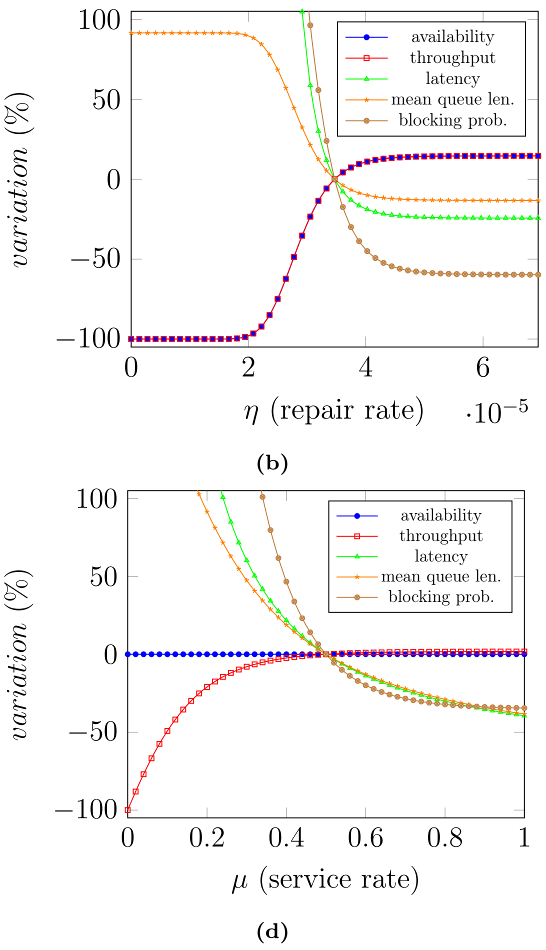


Figure 6.8: Sensitivity analysis, where fixed parameters are $J = 4096$, $N = 64$, $H = 54$, $K = 3000$, and $\mu_t = 0.2$; reference values for the parameters under investigations are $\xi^{\text{ref}} = 6.593 \cdot 10^{-7}$, $\eta^{\text{ref}} = 3.47 \cdot 10^{-5}$, $\lambda^{\text{ref}} = 500$, and $\mu^{\text{ref}} = 0.5$.

Conclusions

Consensus protocols are essential components of any distributed network, particularly DLTs. This is especially true if considering that in general DLTs are, indeed, decentralized and each node in the network is controlled by an independent entity. Hence, the correct functioning of the consensus protocol underlying a DLT solution is fundamental, since it ensures that all participants in the network will agree on the same messages/transactions to be committed in their own memory. However, there may be in the network some nodes that are acting maliciously and they could impact negatively the performance and availability of the system. For instance, DLT solutions that have a BFT consensus protocol are indeed tolerant to a certain level of malicious participants in the network.

This work is the compendium of the research done by the author during the Doctoral studies in the area of distributed systems: the thesis has a focus on the analytical evaluation of performability metrics related to DLT systems.

The two key research questions addressed are:

- Given certain properties characterizing a system (e.g. data throughput, latency, energy consumption, etc.), can we understand which consensus protocol is more suitable in this scenario?
- Once selected a candidate solution, can analytical modeling effectively characterize and predict the qualities of said consensus protocol?

The main contributions provided by this thesis to the field of performability evaluation for consensus protocols in DLT are the analytical models for availability and performability evaluation, supplemented by a MCDM framework to determine which consensus protocol was suitable for the intended scope of the research. Indeed, the work presented in this thesis encompasses the whole process from the selection of a consensus protocol, to the development of analytical models to assess the performability metrics of DLT systems using BFT-based consensus protocols to function.

For instance, the MCDM framework elaborated fills the gap in the literature concerning the ranking and selection of consensus protocols under certain conditions and requirements. It allowed the decision makers to determine that a BFT-based consensus protocol is the potentially most suitable solution for the type of application, that the DLT system under development was intended for. The MCDM framework consists in separate steps concerning the definition of the problem, the selection of the DLT platform to be used as alternatives in the selection, the study regarding the criteria to employ to fully describe any consensus protocol used in DLT, followed by the actual acquisition of the data, and the assignment of weights to each criteria. The application of this methodology lead to the definition of a ranking among the selected alternatives for the defined multi-objective optimization problem, hence the selection of a suitable solution for the first research question.

The remaining of the thesis showed how effective and adequate analytical models are in evaluating the performability of consensus protocols for distributed systems. In particular, following the results obtained from the MCDM framework, the focus of the development of such analytical models is related to consensus protocols that are based on a BFT structure. Two analytical models are proposed for the assessment of availability and performability in the analysis of BFT systems, with the latter including the former model in its blueprint.

The main relevance of the availability model derives from the stochastic analysis of the effect that Byzantine nodes have on the availability of the system. The model is based on a bi-dimensional CTMC, in the form of a quasi-birth-death process. From the analysis of availability, it is evident a non-linear correlation between number of Byzantine nodes and availability, and this behaviour is correlated with the probability distribution used to describe the occurrence of the Byzantine nodes. This leads to the conclusion that, whether it is not possible to determine a priori the distribution of malicious nodes, it is needed to carefully choose what stochastic distribution has to be used in the analysis. Another important aspect of this model is the fact that it represents one of the two components of the wider model evaluating the performability of BFT systems.

Lastly, the performability model here reported contributes to the field by providing an analytic scheme to assess comprehensively five different performability metrics: throughput, transaction latency, blocking probability, mean queue length, and availability. Also this model is based on CTMCs, but in three-dimensions and, as mentioned, it extends the availability model to include the evaluation of performance metrics, making the performability model a tool to wholly assess the performability of a BFT-based system. From the comparison between the results obtained from the performability analysis and benchmarks found in literature, it can be concludes that this analytical framework can effectively reproduce the trend of empirical studies, however the numerical values fail to fall into the error interval defined in those studies. From the sensitivity analysis performed on the parameters defined in the model, it is possible to determine that performability metrics are non-linearly dependent on all the parameters involved, with high fluctuations in the output for small variations in the input. It also confirms the expected behaviours of the system being saturated when the arrival rate is larger or approximately the service rate, along with the independence of the availability from both the arrival and service rate.

Bibliography

[AA19] Shikah J. Alsunaidi and Fahd A. Alhaidari. “A survey of consensus algorithms for blockchain technology”. In: *2019 International Conference on Computer and Information Sciences, ICCIS 2019* (2019), pp. 2–7. DOI: `10.1109/ICCISci.2019.8716424`.

[AB18] Joseph Abadi and Markus Brunnermeier. “Blockchain Economics”. In: *National Bureau of Economic Research. Available at: https://doi.org/10.3386/w25407 (Accessed October 10, 2022)* (2018).

[AK08] Gurumurthy Anand and Rambabu Kodali. “Benchmarking the benchmarking models”. In: *Benchmarking: An international journal* 15.3 (2008), pp. 257–291.

[And+21] Nitish Andola et al. “Anonymity on blockchain based e-cash protocols—A survey”. In: *Computer Science Review* 40 (2021), p. 100394.

[Arj+17] Tom Arjannikov et al. “Using Markov chains to model sensor network reliability”. In: *Proceedings of the 12th International Conference on Availability, Reliability and Security.* 2017, pp. 1–10.

[Ata+17] Ehsan Ataie et al. “Hierarchical stochastic models for performance, availability, and power consumption analysis of IaaS clouds”. In: *IEEE Transactions on Cloud Computing* 7.4 (2017), pp. 1039–1056.

[Bal+21] Simonetta Balsamo et al. “Prediction of the consolidation delay in blockchain-based applications”. In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering.* 2021, pp. 81–92.

[Bal+22] Simonetta Balsamo et al. “Transaction confirmation in proof-of-work blockchains: auctions, delays and droppings”. In: *2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*. IEEE. 2022, pp. 140–149.

[Bal17] Arati Baliga. “Understanding blockchain consensus models”. In: *Persistent* 4.1 (2017), p. 14.

[Ban+19] Shehar Bano et al. “SoK: Consensus in the Age of Blockchains”. In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. New York, NY, USA: ACM, 2019, pp. 183–198. DOI: 10.1145/3318041.3355458.

[Bao+23] Qihao Bao et al. “A survey of blockchain consensus safety and security: State-of-the-art, challenges, and future work”. In: *Journal of Systems and Software* 196 (2023), p. 111555. DOI: 10.1016/j.jss.2022.111555.

[Bel+19] Marianna Belotti et al. “A Vademecum on Blockchain Technologies: When, Which, and How”. In: *IEEE Communications Surveys & Tutorials* 21.4 (2019), pp. 3796–3838. DOI: 10.1109/COMST.2019.2928178.

[BG17] Vitalik Buterin and Virgil Griffith. “Casper the friendly finality gadget”. In: *arXiv preprint arXiv:1710.09437* (2017).

[BHM19] Leemon Baird, Mance Harmon, and Paul Madsen. “Hedera: A public hashgraph network & Governing Council”. In: *Hedera Hashgraph, LLC, White Paper* 1 (2019).

[Bis+22] Stefano Bistarelli et al. “Blockchain and IoT Integration for Pollutant Emission Control”. In: *Advanced Information Networking and Applications: Proceedings of the 36th International Conference on Advanced Information Networking and Applications (AINA-2022), Volume 3*. Springer. 2022, pp. 255–264. DOI: 10.1007/978-3-030-99619-2_25.

[BKM18] Ethan Buchman, Jae Kwon, and Zarko Milosevic. “The latest gossip on BFT consensus”. In: *arXiv preprint arXiv:1807.04938* (2018).

[BL20] Leemon Baird and Atul Luykx. "The Hashgraph protocol: Efficient asynchronous BFT for high-throughput distributed ledgers". In: *2020 International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE. 2020, pp. 1–7.

[BMB20] Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. "A survey of blockchain consensus algorithms performance evaluation criteria". In: *Expert Systems with Applications* 154 (2020), p. 113385. DOI: `10.1016/j.eswa.2020.113385`.

[Bol+06] Gunter Bolch et al. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[Bou21] Sarah Bouraga. "A taxonomy of blockchain consensus protocols: A survey and classification framework". In: *Expert Systems with Applications* 168 (2021), p. 114384.

[BPS16] Iddo Bentov, Rafael Pass, and Elaine Shi. "Snow White: Provably Secure Proofs of Stake". In: *Cryptology ePrint Archive. Available at: `https://eprint.iacr.org/2016/919` (Accessed October 10, 2022)* (2016).

[Buc16] Ethan Buchman. "Tendermint: Byzantine fault tolerance in the age of blockchains". In: *Master Thesis, University of Guelph* (2016).

[But14] Vitalik Buterin. "A next-generation smart contract and decentralized application platform". In: *Available at: `https://ethereum.org/en/whitepaper/` (Accessed October 10, 2022)* (2014).

[BŽ18] Federico Matteo Benčić and Ivana Podnar Žarko. "Distributed ledger technology: Blockchain compared to directed acyclic graph". In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2018, pp. 1569–1570.

[Cac17] Christian Cachin. "Blockchains and consensus protocols: Snake oil warning". In: *2017 13th European Dependable Computing Conference (EDCC)*. IEEE. 2017, pp. 1–2.

[CDK05] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.

[Cha+22] Yan-Xia Chang et al. "Dynamic Practical Byzantine Fault Tolerance and Its Blockchain System: A Large-Scale Markov Modeling". In: *arXiv preprint arXiv:2210.14003* (2022).

[Che+18] Jing Chen et al. "ALGORAND AGREEMENT: Super Fast and Partition Resilient Byzantine Agreement". In: *Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2018/377 (Accessed October 10, 2022)* (2018).

[Che+20] Yaoliang Chen et al. "Decentralized data access control over consortium blockchains". In: *Information Systems* 94 (2020), p. 101590.

[Cho+18] Mohammad Jabed Morshed Chowdhury et al. "Blockchain versus database: A critical analysis". In: *2018 17th IEEE International conference on trust, security and privacy in computing and communications/12th IEEE international conference on big data science and engineering (TrustCom/BigDataSE)*. IEEE. 2018, pp. 1348–1353.

[Cho+19] Mohammad Jabed Morshed Chowdhury et al. "A comparative analysis of distributed ledger technology platforms". In: *IEEE Access* 7 (2019), pp. 167930–167943.

[CL+99] Miguel Castro, Barbara Liskov, et al. "Practical Byzantine Fault Tolerance". In: *OsDI*. Vol. 99. 1999. 1999, pp. 173–186.

[CM18] Brad Chase and Ethan Macbrough. "Analysis of the XRP Ledger Consensus Protocol". In: *Available at: https://doi.org/10.48550/arXiv.1802.07242 (Accessed October 10, 2022)* (2018).

[CM83] K Mani Chandy and Alain J Martin. "A characterization of product-form queuing networks". In: *Journal of the ACM (JACM)* 30.2 (1983), pp. 286–299.

[Coo+10] Brian F Cooper et al. "Benchmarking cloud serving systems with YCSB". In: *Proceedings of the 1st ACM symposium on Cloud computing*. 2010, pp. 143–154.

[Cro+16] Kyle Croman et al. "On Scaling Decentralized Blockchains". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9604 LNCS. 2016, pp. 106–125. ISBN: 9783662533567. DOI: `10.1007/978-3-662-53357-4_8`.

[De 18] Stefano De Angelis. "Assessing security and performances of consensus algorithms for permissioned blockchains". In: *arXiv preprint arXiv:1805.03490* (2018).

[DJ03] R Dattakumar and R Jagadeesh. "A review of literature on benchmarking". In: *Benchmarking: An International Journal* 10.3 (2003), pp. 176–209.

[DN92] Cynthia Dwork and Moni Naor. "Pricing via processing or combatting junk mail". In: *Annual International Cryptology Conference*. Springer. 1992, pp. 139–147. DOI: `10.1007/3-540-48071-4_10`.

[DRZ18] Sisi Duan, Michael K Reiter, and Haibin Zhang. "BEAT: Asynchronous BFT made practical". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 2028–2041.

[Dzi+13] Stefan Dziembowski et al. "Proofs of Space". In: *Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2013/796 (Accessed October 10, 2022)* (2013).

[EP18] Nabil El Ioini and Claus Pahl. "A review of distributed ledger technologies". In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer. 2018, pp. 277–288.

[Eve+19] Enver Ever et al. "On the performance, availability and energy consumption modelling of clustered IoT systems". In: *Computing* 101.12 (2019), pp. 1935–1970.

[Eve17] Enver Ever. "Performability analysis of cloud computing centers with large numbers of servers". In: *The Journal of Supercomputing* 73.5 (2017), pp. 2130–2156.

[Fan+20] Caixiang Fan et al. "Performance Evaluation of Blockchain Systems: A Systematic Survey". In: *IEEE Access* 8 (2020), pp. 126927–126950. DOI: 10.1109/ACCESS.2020.3006078.

[Fil+22] Ernestas Filatovas et al. "A MCDM-based framework for blockchain consensus protocol selection". In: *Expert Systems with Applications* (2022), p. 117609. DOI: 10.1016/j.eswa.2022.117609.

[FLP85] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. "Impossibility of distributed consensus with one faulty process". In: *Journal of the ACM (JACM)* 32.2 (1985), pp. 374–382.

[FM19] Maria Frolkova and Michel Mandjes. "A Bitcoin-inspired infinite-server model with a random fluid limit". In: *Stochastic Models* 35.1 (2019), pp. 1–32.

[For+10] Daniel Ford et al. "Availability in globally distributed storage systems". In: *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*. 2010.

[Fra20] Brian Fralix. "On classes of Bitcoin-inspired infinite-server queueing systems". In: *Queueing Systems 2020 95:1* 95 (1 Jan. 2020), pp. 29–52. ISSN: 1572-9443. DOI: 10.1007/S11134-019-09643-W.

[Fu+20] Wei-Kang Fu et al. "Soteria: A provably compliant user right manager using a novel two-layer blockchain technology". In: *2020 IEEE Infrastructure Conference*. IEEE. 2020, pp. 1–10.

[FWS21] Xiang Fu, Huaimin Wang, and Peichang Shi. "A survey of Blockchain consensus algorithms: mechanism, design and applications". In: *Science China Information Sciences* 64.2 (2021), p. 121101. DOI: 10.1007/s11432-019-2790-1.

[Gei+19] Stefan Geissler et al. "Discrete-Time Analysis of the Blockchain Distributed Ledger Technology". In: *Proceedings of the 31st International Teletraffic Congress, ITC 2019* (Aug. 2019), pp. 130–137. DOI: 10.1109/ITC31.2019.00029.

[Ger+16] Arthur Gervais et al. “On the Security and Performance of Proof of Work Blockchains”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2016, pp. 3–16. DOI: `10.1145/2976749.2978341`.

[Gho+14] Rahul Ghosh et al. “Scalable analytics for IaaS cloud availability”. In: *IEEE Transactions on Cloud Computing* 2.1 (2014), pp. 57–70.

[GKS20] Ulrich Gallersdörfer, Lena Klaaßen, and Christian Stoll. “Energy consumption of cryptocurrencies beyond bitcoin”. In: *Joule* 4.9 (2020), pp. 1843–1846.

[GL87] Ambuj Goyal and Stephen S Lavenberg. “Modeling and analysis of computer system availability”. In: *IBM Journal of Research and Development* 31.6 (1987), pp. 651–664.

[Gun18] Nyoman Gunantara. “A review of multi-objective optimization: Methods and its applications”. In: *Cogent Engineering* 5.1 (2018), p. 1502242.

[HMZ19] Dongyan Huang, Xiaoli Ma, and Shengli Zhang. “Performance analysis of the raft consensus algorithm for private blockchains”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50.1 (2019), pp. 172–181.

[HR17] Garrick Hileman and Michel Rauchs. “2017 global blockchain benchmarking study”. In: *Available at SSRN 3040224* (2017).

[HSS20] Zhi Huang, Sam Snyder, and Gabriel Schillinger. “Aura Protocol: A Peer-to-Peer Blockchain Scaling Solution for Highly Interactive Decentralized Applications”. In: *Available at: `https://www.devgamma.com/documents/aura_technical_paper.pdf` (Accessed October 10, 2022)* (2020).

[Jia+20] Lili Jiang et al. “Performance analysis of Hyperledger Fabric platform: A hierarchical model approach”. In: *Peer-to-Peer Networking and Applications* 13 (3 May 2020), pp. 1014–1025. ISSN: 19366450. DOI: `10.1007/S12083-019-00850-Z/FIGURES/10`.

[JJ99] Markus Jakobsson and Ari Juels. "Proofs of work and bread pudding protocols". In: *Secure Information Networks: Communications and Multimedia Security IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99) September 20–21, 1999, Leuven, Belgium*. Springer. 1999, pp. 258–272.

[JKK05] Norman L Johnson, Adrienne W Kemp, and Samuel Kotz. *Univariate discrete distributions*. Vol. 444. John Wiley & Sons, 2005.

[Kan+20] Niclas Kannengießer et al. "Trade-offs between distributed ledger technology characteristics". In: *ACM Computing Surveys (CSUR)* 53.2 (2020), pp. 1–37.

[Ken53] David G Kendall. "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain". In: *The Annals of Mathematical Statistics* (1953), pp. 338–354.

[Kia+17] Aggelos Kiayias et al. "Ouroboros: A provably secure proof-of-stake blockchain protocol". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10401 LNCS (2017), pp. 357–388. ISSN: 16113349. DOI: 10.1007/978-3-319-63688-7_12.

[Kir+15] Yonal Kirsal et al. "Modelling and analysis of vertical handover in highly mobile environments". In: *The Journal of Supercomputing* 71.12 (2015), pp. 4352–4380.

[KK17] Yoshiaki Kawase and Shoji Kasahara. "Transaction-Confirmation time for Bitcoin: A Queueing Analytical Approach to Blockchain Mechanism". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10591 LNCS (2017), pp. 75–88. ISSN: 16113349. DOI: 10.1007/978-3-319-68520-5_5.

[KL18] Merve Can Kus Khalilov and Albert Levi. "A survey on anonymity and privacy in bitcoin-like digital cash systems". In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2543–2585.

[Lee19] Jei Young Lee. “A decentralized token economy: How blockchain and cryptocurrency can revolutionize business”. In: *Business Horizons* 62.6 (2019), pp. 773–784.

[Li+17] Wenting Li et al. “Securing proof-of-stake blockchain protocols”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10436 LNCS (Sept. 2017), pp. 297–315. ISSN: 16113349. DOI: `10.1007/978-3-319-67816-0_17`.

[Li+19] Quan Lin Li et al. “Markov processes in blockchain systems”. In: *Computational Social Networks* 6 (1 Dec. 2019), pp. 1–28. ISSN: 21974314. DOI: `10.1186/S40649-019-0066-1/FIGURES/4`.

[Lit61] John DC Little. “A proof for the queuing formula: L= λ W”. In: *Operations research* 9.3 (1961), pp. 383–387.

[Liu+22] Wei Liu et al. “Optimization of PBFT algorithm based on QoS-aware trust service evaluation”. In: *Sensors* 22.12 (2022), p. 4590.

[LKK07] Averill M Law, W David Kelton, and W David Kelton. *Simulation modeling and analysis*. Vol. 3. Mcgraw-hill New York, 2007.

[LMC18] Quan Lin Li, Jing Yu Ma, and Yan Xia Chang. “Blockchain Queue Theory”. In: *Lecture Notes in Computer Science* 11280 LNCS (2018), pp. 25–40. ISSN: 16113349. DOI: `10.1007/978-3-030-04648-4_3`.

[Lon+11] Francesco Longo et al. “A scalable availability model for infrastructure-as-a-service cloud”. In: *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. IEEE. 2011, pp. 335–346.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. “The Byzantine Generals Problem”. In: *ACM Transactions on Programming Languages and Systems* 4.3 (1982), pp. 382–401.

[Ma+20] Zhanyou Ma et al. “Performance Analysis of Blockchain Consensus System with Interference Factors and Sleep Stage”. In: *IEEE Access* 8 (2020), pp. 119010–119019. ISSN: 21693536. DOI: `10.1109/ACCESS.2020.3005919`.

[MA04] R Timothy Marler and Jasbir S Arora. "Survey of multi-objective optimization methods for engineering". In: *Structural and multidisciplinary optimization* 26 (2004), pp. 369–395.

[Mac68] Kenneth R MacCrimmon. *Decisionmaking among multiple-attribute alternatives: a survey and consolidated approach.* Rand Corporation Santa Monica, 1968.

[MAG15] Arslan Munir, Joseph Antoon, and Ann Gordon-Ross. "Modeling and analysis of fault detection and fault tolerance in wireless sensor networks". In: *ACM Transactions on Embedded Computing Systems (TECS)* 14.1 (2015), pp. 1–43.

[Mar+23a] Marco Marcozzi et al. "Availability evaluation of IoT systems with Byzantine fault-tolerance for mission-critical applications". In: *Internet of Things* 23 (2023), p. 100889. DOI: 10.1016/j.iot.2023.100889.

[Mar+23b] Marco Marcozzi et al. "Availability Model for Byzantine Fault-Tolerant Systems". In: *Advanced Information Networking and Applications: Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA-2023), Volume 1.* Springer. 2023, pp. 31–43. DOI: 10.1007/978-3-031-29056-5_4.

[Mau+17] Roger Maull et al. "Distributed ledger technology: Applications and implications". In: *Strategic Change* 26.5 (2017), pp. 481–489.

[Maz16] David Mazières. "The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus". In: *Available at: https://www.stellar.org/papers/stellar-consensus-protocol (Accessed October 10, 2022)* (2016).

[Mel+21] Carlos Melo et al. "Distributed application provisioning over Ethereum-based private and permissioned blockchain: availability modeling, capacity, and costs planning". In: *The Journal of Supercomputing* 77.9 (2021), pp. 9615–9641.

[Men+21] Tianhui Meng et al. “On Consortium Blockchain Consistency: A Queueing Network Model Approach”. In: *IEEE Transactions on Parallel and Distributed Systems* 32 (6 June 2021), pp. 1369–1382. ISSN: 15582183. DOI: 10.1109/TPDS.2021.3049915.

[Mey80] Meyer. “On evaluating the performability of degradable computing systems”. In: *IEEE Transactions on computers* 100.8 (1980), pp. 720–731.

[MF22] Fan Qi Ma and Rui Na Fan. “Queuing Theory of Improved Practical Byzantine Fault Tolerant Consensus”. In: *Mathematics 2022, Vol. 10, Page 182* 10 (2 Jan. 2022), p. 182. ISSN: 2227-7390. DOI: 10.3390/MATH10020182.

[MG11] Arslan Munir and Ann Gordon-Ross. “Markov modeling of fault-tolerant wireless sensor networks”. In: *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*. IEEE. 2011, pp. 1–6.

[Mil+16] Andrew Miller et al. “The honey badger of BFT protocols”. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 31–42.

[Min+17] Du Mingxiao et al. “A review on consensus algorithm of blockchain”. In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Vol. 2017-Janua. IEEE, 2017, pp. 2567–2572. DOI: 10.1109/SMC.2017.8123011.

[MM21] Marco Marcozzi and Leonardo Mostarda. “Quantum consensus: an overview”. In: *arXiv preprint arXiv:2101.04192* (2021). DOI: 10.48550/arXiv.2101.04192.

[MM23] Marco Marcozzi and Leonardo Mostarda. “Analytical model for performability evaluation of Practical Byzantine Fault-Tolerant systems”. In: *Expert Systems with Applications* (2023), p. 121838. DOI: 10.1016/j.eswa.2023.121838.

[MMC22] Marco Marcozzi, Leonardo Mostarda, and Diletta Cacciagrano. “Off-chain trading for micro grid systems”. In: *Frontiers in Blockchain* 5 (2022). DOI: `10.3389/fbloc.2022.956621`.

[MR13] Ignacio J. Martinez-Moyano and George P. Richardson. “Best practices in system dynamics modeling”. In: *System Dynamics Review* 29.2 (2013), pp. 102–123. DOI: `https://doi.org/10.1002/sdr.1495`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/sdr.1495`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/sdr.1495`.

[Mül+22] Sebastian Müller et al. “Tangle 2.0 leaderless Nakamoto consensus on the heaviest DAG”. In: *IEEE Access* 10 (2022), pp. 105807–105842.

[Nak08] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: *Decentralized Business Review* (2008).

[Ngu+19] Cong T Nguyen et al. “Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities”. In: *IEEE Access* 7 (2019), pp. 85727–85745.

[NK18] Giang Truong Nguyen and Kyungbaek Kim. “A survey about consensus algorithms used in Blockchain”. In: *Journal of Information Processing Systems* 14.1 (2018). DOI: `10.3745/JIPS.01.0024`.

[NL20] Jeff Nijsse and Alan Litchfield. “A Taxonomy of Blockchain Consensus Methods”. In: *Cryptography* 4.4 (2020), p. 32. DOI: `10.3390/cryptography4040032`.

[Odu19] GO Odu. “Weighting methods for multi-criteria decision making technique”. In: *Journal of Applied Sciences and Environmental Management* 23.8 (2019), pp. 1449–1457. DOI: `10.4314/jasem.v23i8.7`.

[OT04] Serafim Opricovic and Gwo-Hshiung Tzeng. “Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS”. In: *European journal of operational research* 156.2 (2004), pp. 445–455.

[ØUJ17] Svein Ølnes, Jolien Ubacht, and Marijn Janssen. "Blockchain in government: Benefits and implications of distributed ledger technology for information sharing". In: *Government Information Quarterly* 34.3 (2017), pp. 355–364.

[Par19] Pangun Park. "Markov chain model of fault-tolerant wireless networked control systems". In: *Wireless Networks* 25.5 (2019), pp. 2291–2303.

[Pau+19] Remigijus Paulavičius et al. "A Decade of Blockchain: Review of the Current Status, Challenges, and Future Directions". In: *Informatica* 30.4 (2019), pp. 729–748. DOI: 10.15388/Informatica.2019.227.

[Per+21] Paulo Pereira et al. "Analytical models for availability evaluation of edge and fog computing nodes". In: *The Journal of Supercomputing* 77.9 (2021), pp. 9905–9933.

[Pop18] Serguei Popov. "The Tangle". In: *Available at: https://api.semanticscholar.org/CorpusID:4958428 (Accessed October 10, 2022)* (2018).

[QYJ20] Jiaxing Qi, Jing Yu, and Shunfu Jin. "Nash Equilibrium and Social Optimization of Transactions in Blockchain System Based on Discrete-Time Queue". In: *IEEE Access* 8 (2020), pp. 73614–73622. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2986084.

[Rao19] Singiresu S Rao. *Engineering optimization: theory and practice.* John Wiley & Sons, 2019.

[Rau+18] Michel Rauchs et al. "Distributed ledger technology systems: A conceptual framework". In: *Available at SSRN 3230013* (2018). DOI: 10.2139/ssrn.3230013.

[RHF21] Mohammadreza Rasolroveicy, Wejdene Haouari, and Marios Fokaefs. "Public or private? a techno-economic analysis of blockchain". In: *Proceedings of the 31st Annual International Conference on Computer Science and Software Engineering.* 2021, pp. 83–92.

[Ric+19] Saulo Ricci et al. “Learning Blockchain Delays: A Queueing Theory Approach”. In: *ACM SIGMETRICS Performance Evaluation Review* 46 (3 Jan. 2019), pp. 122–125. ISSN: 01635999. DOI: `10.1145/3308897.3308952`.

[Ril18] Kynan Rilee. “Understanding Hyperledger Sawtooth — Proof of Elapsed Time”. In: *Available at: https://medium.com/kokster/understanding-hyperledger-sawtooth-proof-of-elapsed-time-e0c303577ec1 (Accessed October 10, 2022)* (2018).

[Rim+17] Paul Rimba et al. “Comparing blockchain and cloud services for business process execution”. In: *2017 IEEE international conference on software architecture (ICSA)*. IEEE. 2017, pp. 257–260.

[Roc+19] Team Rocket et al. “Scalable and probabilistic leaderless BFT consensus through metastability”. In: *arXiv preprint arXiv:1906.08936* (2019).

[SAS20] Rishi Kumar Srivastav, Devendra Agrawal, and Anurag Shrivastava. “A Survey on Vulnerabilities and Performance Evaluation Criteria in Blockchain Technology”. In: *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal* 9 (2 June 2020), pp. 91–105. ISSN: 2255-2863. DOI: `10.14201/adcaij20209291105`.

[Sil+12] Ivanovitch Silva et al. “Reliability and availability evaluation of wireless sensor networks for industrial applications”. In: *Sensors* 12.1 (2012), pp. 806–838.

[Sin+22] Arshdeep Singh et al. “A survey and taxonomy of consensus protocols for blockchains”. In: *Journal of Systems Architecture* 127 (2022), p. 102503. DOI: `10.1016/10.1016/j.sysarc.2022.102503`.

[SKU18] Anastasiia Strielkina, Vyacheslav Kharchenko, and Dmytro Uzun. “Availability models for healthcare IoT systems: Classification and research considering attacks on vulnerabilities”. In: *2018 IEEE 9th international conference on dependable systems, services and technologies (DESSERT)*. IEEE. 2018, pp. 58–62.

[Sme+20a] Sergey Smetanin et al. “Blockchain Evaluation Approaches: State-of-the-Art and Future Perspective”. In: *Sensors (Basel, Switzerland)* 20.12 (2020), pp. 1–20. DOI: 10.3390/s20123358.

[Sme+20b] Sergey Smetanin et al. “Modeling of Distributed Ledgers: Challenges and Future Perspectives”. In: *Proceedings - 2020 IEEE 22nd Conference on Business Informatics, CBI 2020* 1 (June 2020), pp. 162–171. DOI: 10.1109/CBI49978.2020.00025.

[SP08] T.L. Saaty and K. Peniwati. *Group Decision Making: Drawing Out and Reconciling Differences*. RWS Publications, 2008. ISBN: 9781888603088. URL: https://books.google.lt/books?id=phLWKwAACAAJ.

[SR20] Sheikh Munir Skh Saad and Raja Zahilah Raja Mohd Radzi. “Comparative review of the blockchain consensus algorithm between proof of stake (PoS) and delegated proof of stake (DPoS)”. In: *International Journal of Innovative Computing* 10.2 (2020).

[SS21] Jignasha Shah and Deepak Sharma. “Performance benchmarking frameworks for distributed ledger technologies”. In: *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE. 2021, pp. 1–5.

[Suc+18] George Suciu et al. “Comparative analysis of distributed ledger technologies”. In: *2018 Global Wireless Summit (GWS)*. IEEE. 2018, pp. 370–373.

[Suk+17] Harish Sukhwani et al. “Performance modeling of PBFT consensus process for permissioned blockchain network (Hyperledger Fabric)”. In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE. 2017, pp. 253–255.

[Sun20] Ali Sunyaev. “Distributed ledger technology”. In: *Internet Computing*. Springer, 2020, pp. 265–299.

[Tan+22] Song Tang et al. “Improved PBFT algorithm for high-frequency trading scenarios of alliance blockchain”. In: *Scientific Reports* 12.1 (2022), p. 4426.

[TB17] Kishor S Trivedi and Andrea Bobbio. *Reliability and availability engineering: modeling, analysis, and applications*. Cambridge University Press, 2017.

[Tin19] Julien Tinguely. “Benchmarking of Distributed Ledger Technology”. In: *Bachelor’s Thesis at ETH Zürich* (2019).

[Tri08] Kishor S Trivedi. *Probability & statistics with reliability, queuing and computer science applications*. John Wiley & Sons, 2008.

[TS17] A.S. Tanenbaum and M. van Steen. *Distributed Systems*. CreateSpace Independent Publishing Platform, 2017. ISBN: 9781543057386. URL: `https://books.google.lt/books?id=c77GAQAACAAJ`.

[TX21] Shensheng Tang and Yi Xie. “Availability modeling and performance improving of a healthcare internet of things (IoT) system”. In: *IoT* 2.2 (2021), pp. 310–325.

[Wan+19] Xinying Wang et al. “BlockLite: A Lightweight Emulator for Public Blockchains”. In: *arXiv e-prints*, arXiv:1905.02157 (May 2019), arXiv:1905.02157. arXiv: `1905.02157 [cs.DB]`.

[Wan+20] Zhiyuan Wang et al. “Analysis of weighting and selection methods for Pareto-optimal solutions of multiobjective optimization in chemical engineering applications”. In: *Industrial & Engineering Chemistry Research* 59.33 (2020), pp. 14850–14867.

[WG21] Francesc Wilhelmi and Lorenza Giupponi. “Discrete-Time Analysis of Wireless Blockchain Networks”. In: *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC* 2021-September (Sept. 2021), pp. 1011–1017. DOI: `10.1109/PIMRC50174.2021.9569253`.

[Woo14] Daniel Davis Wood. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Available at: https://api.semanticscholar.org/CorpusID:4836820 (Accessed October 10, 2022)* (2014).

[WR17] Zhiyuan Wang and Gade Pandu Rangaiah. “Application and analysis of methods for selecting an optimal solution from the Pareto-optimal front obtained by multiobjective optimization”. In: *Industrial & Engineering Chemistry Research* 56.2 (2017), pp. 560–574.

[Xia+20] Yang Xiao et al. “A Survey of Distributed Consensus Protocols for Blockchain Networks”. In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 1432–1465. DOI: `10.1109/COMST.2020.2969706`.

[Yin+18] Maofan Yin et al. “HotStuff: BFT consensus in the lens of blockchain”. In: *arXiv preprint arXiv:1803.05069* (2018).

[Zha+22] Kaifeng Zhang et al. “Research and Improvement of Blockchain DPoS Consensus Mechanism”. In: *International Conference on Computer Engineering and Networks*. Springer. 2022, pp. 1284–1292.

[Zhe+18] Zibin Zheng et al. “Blockchain challenges and opportunities: a survey”. In: *International Journal of Web and Grid Services* 14.4 (2018), p. 352. DOI: `10.1504/IJWGS.2018.095647`.

[ZKC20] Rong Zhang, Wai Kin, and Victor Chan. “Evaluation of Energy Consumption in Block-Chains with Proof of Work and Proof of Stake”. In: *Journal of Physics: Conference Series* 1584 (2020), p. 12023. DOI: `10.1088/1742-6596/1584/1/012023`.

[ZL20] Shijie Zhang and Jong-Hyouk Lee. “Analysis of the main consensus protocols of blockchain”. In: *ICT Express* 6.2 (2020), pp. 93–97. DOI: `10.1016/j.icte.2019.08.001`.