Noname manuscript No.
(will be inserted by the editor)

A Model-based Approach for Planning Blockchain Service Provisioning

Carlos Melo, Jean Araujo, Jamilson Dantas, Paulo Pereira and Paulo Maciel

Received: date / Accepted: date

Abstract Recently, the blockchain-as-a-service paradigm arose, and many works have evaluated the performance issues related to it. However, not as much has been done regarding Dependability attributes, which have ever been a crucial topic on service provisioning, let it be either public or private infrastructures. This paper presents the Blockchain Provisioning Planning Tool (BPPT), a framework to evaluate the availability, deployment, and maintenance costs of Hyperledger Fabric-based applications over private computational infrastructures. The BPPT uses Continuous Time Markov Chain (CTMC) and Reliability Block Diagram (RBD) models as an evaluation method of Hyperledger Fabric's environments and determines distributed applications' deployment feasibility and endorsement policies related to the platform. We also present case studies that may help those interested in paradigm changes to decide whether they should migrate from old to new technology. Some of the obtained results pointed-out that the AND endorsement, which requires that all nodes sign the authenticity of a transaction, has the highest deployment and maintenance costs, as well as the lowest availability values due to operational requirements, already an OR endorsement, which needs that at least one available node signs the transaction, provides the best relationship between the evaluated metrics. The KooN endorsement (that requires that K out of N nodes signs a transaction authenticity) is a more general model that supports analyzing midterm configurations, besides the two extreme configurations, that is, to AND and OR arrangements.

Keywords Framework · Availability · Costs · Blockchain · Fabric

Carlos Melo, Jamilson Dantas, Paulo Pereira, and Paulo Maciel
Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil
Tel.: +55-87-999265912
E-mail: casm3,jrd,prps,prrm@cin.ufpe.br

Jean Araujo
Universidade Federal do Agreste de Pernambuco, Garanhuns, Brazil
E-mail: jean.teixeira@ufape.edu.br

1 Introduction

Blockchain is the technology behind the Bitcoin [10], which showed up at the end of the 2000s, but is not as famous among users as cryptocurrency is. This may have been due to the simple fact that it is easier to put a value over an application than assigning it over an entire technology. Nevertheless, this trending technology emerged from the distributed computing paradigm and can be (and already is) used as a safer way to transfer, host, and share information between users, companies, and entities. However, as we keep learning and improving our definition of this technology, it is possible to foresight that there are still many to be done to determine the feasibility of adopting this paradigm to deploy and host distributed applications.

The Hyperledger consortium [7], which is managed by Linux Foundation and hosts a set of projects to help users and companies to migrate from traditional and (too) old tools and mechanisms to this (not so) new (but emerging) technology. One of these projects is the Hyperledger Fabric, which has become a standard for providing blockchain-based services on major cloud computing platforms such as Amazon, Google, IBM, and Microsoft and is a kick-off for Blockchain-as-a-Service provisioning.

As with any cloud computing service, the users requires it and its data to be available 24 hours a day and seven days a week, and depending on the application hosted by the platform such as a medical history of a patient [4] or any other critical system, a failure on service provisioning may lead from financial losses to deaths. From the cloud service provider perspective, a Service Level Agreements (SLA) must be signed to guarantees performance and availability indexes that should be accomplished by those who own the infrastructure.

From the academic perspective, some works had evaluated the Hyperledger Fabric attributes, most of them focused on the platform performance, such as Sukhwani et al. 2017 [24], Pongnumkul et al. 2017 [21], and Thakkar et al. 2018 [27], while other ones such as Melo et al. 2019 [18] focused only in the system's availability issues. To the best of our knowledge, neither of the works had proposed a blockchain-specific platform that helps planners evaluate alternative configurations to enhance availability and estimate maintenance expenses. This paper presents an approach for blockchain-based technology availability modeling, and a framework evaluates proposed models to support planning blockchain services based on Hyperledger Fabric's platform.

The remainder of this paper is organized as follows. Section 2 presents the works that underlie this research. Section 3 shows the background explaining the most important topics to fully understanding this paper. Section 4 presents the support methodology that enabled the construction of the proposed models and their evaluation, which we present in Section 5. Section 6 presents an overview of our framework, mainly aiming at what it can do. Already Section 7 provides the scenarios used as a case study to demonstrate how feasible is the proposed framework. Finally, Section 8 presents the final remarks and future directions.

2 Related Works

Over the last few years, authors have devoted their efforts to study and evaluate performance and dependability attributes related to blockchain technologies. This section presents some of these remarkable works.

Pongnumkul et al. 2017 [21] proposed a performance evaluation methodology for blockchain-based environments and evaluated both Ethereum and Hyperledger Fabric's platforms. The authors concluded that the Hyperledger Fabric has a large throughput and lower latency than Ethereum, which is much like to be a result of the consensus protocol adopted. While Ethereum uses a Proof-of-Work (PoW) protocol, the Hyperledger Fabric worked with the Practical Byzantine Fault Tolerance (PBFT) protocol. This paper differs from ours in both evaluated methods and metrics and the evaluation of an Ethereum-based platform.

In Thakkar et al. 2018 [27], the authors evaluated the performance of Hyperledger Fabric and identified some of its performance bottlenecks. The paper proposed improvements on platform infrastructure based on parameter variation, such as block size, endorsement policy, and so on, over the transaction throughput and general latency. As a significant result that we may cite, the authors could improve the overall throughput by 16 times. Once again, the authors focused only on the system's performance issues, putting aside dependability attributes, differing from our paper in both evaluation method and metrics.

Already in Sukhwani et al. 2017 [24], the authors investigated whether a consensus process using the PBFT mechanism could be a performance bottleneck. The authors evaluated the system through experimentation and Stochastic Reward Nets (SRN) modeling, which was later used to compute critical performance metrics, such as the mean time to complete the consensus process. The PBFT protocol is one of the most used by blockchain platforms, nearly as important as the PoW protocol.

Later, in Sukhwani et al. 2018 [25], the authors evaluated the performance of the Hyperledger Fabric through SRN models, which means that this works dealt with the performance evaluation of the entire platform, improving their previous work that focused only on the consensus protocol. Once again, the proposed work differs from ours in the evaluated metrics, modeling formalism, and most importantly, the main objectives. In both papers, the authors used the SPNP package to construct and evaluate their models.

Hao et al. 2018 [6] also had evaluated the performance of both Ethereum and Hyperledger Fabric consensus algorithms in private blockchain infrastructures. The evaluation method was measurement, and their focus was on the transaction per second (TPS) and Latency metrics.

The authors in Roy et al. 2019 [22] evaluated security issues and their impact on a Hyperledger Fabric infrastructure's performance. The metrics evaluated were also TPS and Latency, and the evaluation method was measurement.

Already in Sun et al. 2019 [26], a theoretical blockchain model was proposed and evaluated through simulation. They had focused on performance as well, including both TPS and Latency metrics. This paper presented a high-level view evaluation, which is limited compared to more formal approaches, such as measuring and analytical models. As could already be seen, most papers focused only on the Hyperledger Fabric platform's performance issues, many of them tell us about how dependable is a blockchain system and so on, but nearly none had evaluated dependability attributed.

Table 1 presents a general comparison between the current state-of-art and the present paper.

Table 1: Comparison between state-of-art and present work

Authors	**Models**	**Metrics**	**Tool**
Pongnumkul et al. 2017 [21]	-	Performance	No
Thakkar et al. 2018 [27]	-	Performance	No
Sukhwani et al. 2017 [24]	SRN	Performance	No
Sukhwani et al. 2018 [25]	SRN	Performance	No
Hao et al. 2018[6]	-	Performance	No
Roy et al. 2019[22]	-	Performance	No
Sun et al. 2019[26]	Theoretical Model	Performance	No
This paper	RBD & CTMC	Availability and Costs	Yes

It is important to mention that in both Melo et al. 2019 [18,19], we had evaluated availability and costs related to the deployment of blockchain applications over cloud computing environments. These are the closest works to what we propose here and are a basis for the final models presented and used by the framework proposed in this paper. It is also important to highlight that the costs previously evaluated do not consider the values related to maintenance routines. The endorsement policies considered here were not previously evaluated, meaning that this paper combines both previous ones and adds some critical resources regarding blockchain service provisioning. Here we also proposed a tool to help companies and analysts to decide about a change in their current environments to a blockchain environment managed by Hyperledger Fabric.

3 Background

This section presents the fundamental concepts about availability evaluation, and system's modeling.

3.1 Availability Evaluation

Dependability is the computer system's ability to deliver a service that can be justifiably trusted [1]. This definition considers the users' perception and system behavior, meaning that the system is considered a black box.

Evaluating a system's dependability is usually the result of evaluating at least one of its five attributes: reliability, availability, maintainability, integrity, and safety [2]. The system's availability is the main dependability attribute evaluated by this paper and can be subdivided into two different subcategories: instantaneous and steady-state availability.

The system's instantaneous availability, is the probability that the system is operational at a time t. That is, $A(t) = P\left\{Z(t)\right\} = E\left\{Z(t)\right\}, t \geq 0$, where $Z(t) = 1$ when the system is operational, and $Z(t) = 0$.

The Steady-state availability, which is the one evaluated by this paper and is also called the long-run availability, is the limit of the availability function as time tends to infinity (Equation 1).

$$A = \lim_{t \to \infty} A(t), t \geq 0 \tag{1}$$

Also, the system's availability may be represented by a ratio between the Mean Time To Failure (MTTF) and Mean Time To Repair (MTTR) of the system (Equation 2).

$$\mathrm{A} = \frac{\mathrm{MTTF}}{\mathrm{MTTF{+}MTTR}} \tag{2}$$

The system's MTTF may be computed from Equation 3, where $\mathbf{R(t)}$ is the reliability of that system as a function of elapsed time. The Equation 4 provides a way of computing the MTTR from the values of MTTF, availability (A), and unavailability (UA $= 1 - A$).

$$\mathrm{MTTF} = \int_0^\infty R(t)\partial t \tag{3}$$

$$\mathrm{MTTR} = \mathrm{MTTF} \times (\frac{UA}{A}), \tag{4}$$

As expected, the system's unavailability is the counterpart of the system's availability and can be calculated by $1 - A$. Already the system's downtime, which corresponds to the period where the system is not available, is usually given in units of time, such as in hours per year, in a way that relates time and probability: $Unavailability \times 8760h$.

The number of nines can also represent the availability [20], as shown in Table 2. For example, a system with four 9's of availability is classified as fault-tolerant, meaning an annual downtime of nearly 1 hour.

Table 2: Service availability in number of nines.

# of 9's	**Avail. (%)**	**System Type**	**Downtime (year)**
1	90	unmanaged	5 weeks
2	99	managed	4 days
3	99.9	well managed	9 hours
4	99.99	fault tolerant	1 hour
5	99.999	high availability	5 minutes
6	99.9999	very high availability	30 seconds
7	99.99999	ultra availability	3 seconds

The system's availability of associated metrics and values can usually be evaluated through simulation, measurement, and models. The latter was chosen due to the high-level system view provided and their great flexibility in adapting parameters and achieving results faster than the other two evaluation techniques. This type of model can also be called white box or structural models and enabled us to create a mathematical representation of the system based on its relationship. Regarding modeling, we may highlight two main trends, which are:

- **Combinational or non-state-space** models capture conditions that make a system fail (or to be working) regarding structural relationships between the system's components [28];
- Already **State-space models** represent the system's behavior (failures and repair activities) by a set of states and the occurrence of events that can be expressed as rates or distribution functions. These models represent more complex relationships between the system's components than combinatorial models do [16].

Reliability Block Diagram (RBD) [14] and Fault Tree (FT) [15] are among the most prominent combinatorial modeling formalisms, whereas Petri Nets (PN), Continuous Time Markov Chain (CTMC) and Stochastic Automata Networks are well-known state-space modeling formalism [14,5,28].

RBDs were initially proposed to evaluate the system's reliability by describing its components' behavior and their relationship as a set of blocks. The main limitation of RBDs is the inability to describe concurrent and complex systems [14,12,11], which is not the case in this paper. The blocks in an RBD can be organized in parallel, serial, and as a K-out-of-N, and there are only two possible outcomes when dealing with its outputs, 0 for when the system is unavailable 1, otherwise.

For example, serially connected blocks have their steady-state reliability and availability evaluated through Equations [14].

$$R_{Series}(t) = \prod_{i=1}^{n} R_i(t) \tag{5}$$

$$A_{Series} = \prod_{i=1}^{n} A_i \tag{6}$$

Already for a set of n-parallel blocks, the Equation 7 can be used to evaluate both dependability's attributes [14].

$$R_{Parallel}(t) = 1 - \prod_{i=1}^{n}(1 - R_i(t)) \tag{7}$$

$$A_{Parallel} = 1 - \prod_{i=1}^{n}(1 - A_i) \tag{8}$$

Last but not least important is the k-out-of-n block (KooN) that describes a set of k equal components among a total of n required to perform the service provisioning. That means saying 3-out-of-4 components implies that at least three components of four available must be operational to perform an activity. The Equation 9 presents the KooN.

$$R_{KooN}(t) = \sum_{i=k}^{N} \binom{N}{i} R_c(t))^i \times (1 - R_c(t))^{N-i}, \tag{9}$$

$$A_{KooN} = \sum_{i=k}^{N} \binom{N}{i} (A_c)^i \times (1 - A_c)^{N-i}, \tag{10}$$

where $R_c(t)$ and A_c are the component´s reliability and steady-state availability, respectively. Using these basic models and CTMC, we obtained the system availability expressions that can evaluate the system's availability and associated metrics mathematically, later used by the proposed framework.

3.2 A Hyperledger Fabric Overview

The Hyperledger Fabric[1] (HLF) is a platform for the construction and deployment of solutions based on shared ledgers. This platform is an open-source standard developed and maintained by the Hyperledger Consortium[2] under the tutelage of the Linux Foundation.

The Table 3[3] shows some of the places where the Hyperledger Fabric can be used, and your applications may be hosted.

There are many ways to deploy a private or permissioned Hyperledger Fabric infrastructure. A typical HLF environment has at least two actors: the client and the service provider. In this paper, which includes the still to be presented framework, we evaluate only the service provider side, which means

[1] Hyperledger Fabric: https://www.hyperledger.org/projects/fabric

[2] Hyperledger: https://www.hyperledger.org/about

[3] Ledger Insights: https://www.ledgerinsights.com/how-to-blockchain-as-a-service-baas/

Table 3: Why should I choose the Hyperledger Fabric?

Host	**Ethereum**	**Quorum**	**Corda**	**Hyperledger Fabric**	**MultiChain**
AWS	✓	✓	✓	✓	
Azure	✓	✓	✓	✓	✓
Google	✓			✓	
HPE			✓		
IBM				✓	
Oracle				✓	
SAP				✓	✓

that the client-side does not impact either availability and both deployment and maintenance costs.

The service provider side on an HLF environment uses container technology based on smart contracts (chaincodes) management. These chaincodes must be pre-installed on the environment as it is deployed. Generally speaking, the chaincodes are responsible for managing the business rules and how the application will perform its activities. Already on the client-side, an SDK written in Node.JS or Java provides the communication between the server and client parts.

Figure 1 shows a high-level view of a server hosting the Hyperledger Fabric minimal deployment. Since any component's failure results in the service's failure, there is no real dependency between them, at least in terms of service provisioning.

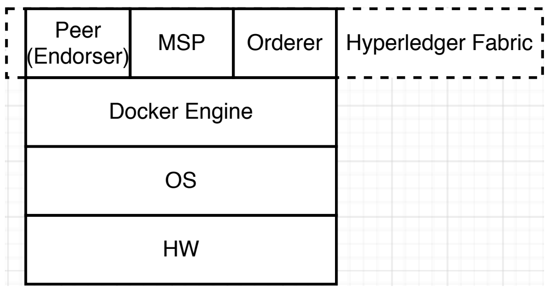


Fig. 1: Service Stack

There are four different components considered in our high-level view: the server's hardware (HW), operating system (OS), container engine (Docker Engine), and the deployed containers. The Hyperledger Fabric's environment deals with three different types of containers: (1) the peer node, which performs endorsement; (2) the MSP node, responsible for membership and access to the platform; and (3) the orderer node, that receives and assigns transactions to batches and send it back to the peer node. This high-level view is essential since our models will be provided based on these components' relationship.

3.3 A Blockchain-based Application

This paper evaluates a basic application deployed over the three Hyperledger Fabric's containers. Usually, a client uses its SDK to send a transaction to the service provider. However, many other steps are required and must be first accomplished on the other side to perform a transaction.

Figure 2 shows how the client and the service provider communicates, as well as which steps must be followed by a transaction in order to it be entirely accepted by the system [7,8].

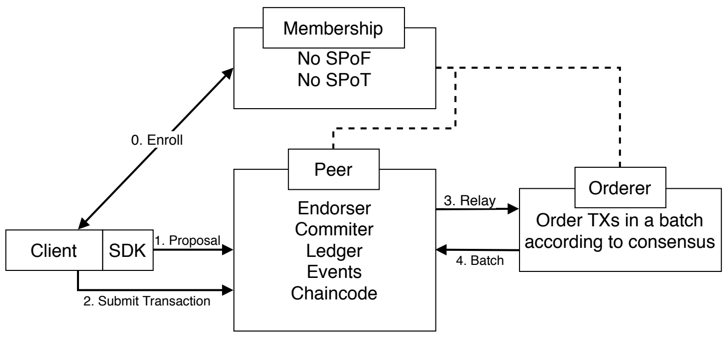


Fig. 2: Hyperledger Fabric's Overview

The client connects to the system through the **Membership** Service Provider (MSP), which should provide no Single Point of Failure (SPoF) and no Single Point of Truth (SPoT). The MSP verifies the credentials of this client and provides access to the service provider. Later, the client proposes a transaction to the node know as Peer. There are many kinds of peers. Our focus is on endorsement peers, who simulate a transaction and send it back to the client with a signature that determines if the system can perform this transaction.

As an example of a transaction, we may cite transferring assets between two clients, which needs both clients to exist and that the sender has as much balance as the value that he wants to transfer, discounting additional fees.

The transaction endorsement is based on a set of policies described by the chaincode. These policies specify how many and how these nodes agree with a transaction state. There are three central endorsement policies, which are AND, OR, and K-out-of-N (KooN). Suppose that we have three servers on the service provisioning side, each one hosting an endorsement peer. If an AND policy is being used, all three nodes must endorse the transaction, which means that all three must sign it back to the client. If we are using an OR policy, then at least one of the three nodes must sign the transaction. The same applies to the KooN policy, where we determine how many of the available peers must sign the transaction, 1-out-of-3, 2-out-of-3, or even 3-out-of-3.

After the endorsement, the transaction is submitted to the Orderer container. The orderer container joins all transactions in a batch (block) and sends it to the peers, which should commit the ledger's transactions based on the chaincode requirements. This comprises most of the Hyperledger Fabric applications.

4 Modeling and Evaluation Methodology

This section presents how we had accomplished our primary goals and how this work can be replicated. At first, the required hardware and software resources are listed, as well as their expected behavior.

Figure 3 shows an organization chart that summarizes our strategy.

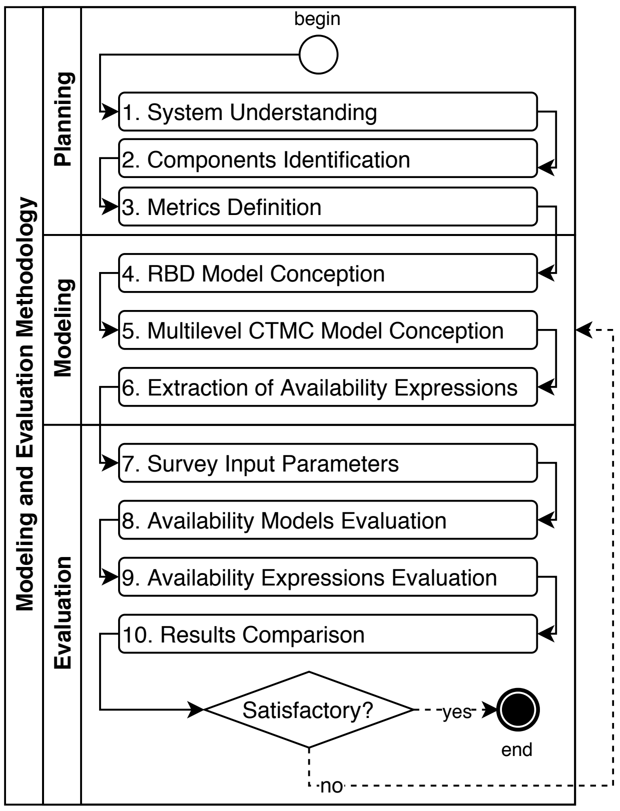

Modeling and Evaluation Methodology
Planning
Modeling
Evaluation
begin
1. System Understanding
2. Components Identification
3. Metrics Definition
4. RBD Model Conception
5. Multilevel CTMC Model Conception
6. Extraction of Availability Expressions
7. Survey Input Parameters
8. Availability Models Evaluation
9. Availability Expressions Evaluation
10. Results Comparison
Satisfactory?
yes
end
no

Fig. 3: Modeling and Evaluation Methodology

The rounded rectangles represent each step of the modeling and evaluation methodology, while the arrows connecting the rectangles define an order to this process. The evaluator only advances to the next step after the completion of the current one. Already the diamond represents a step that can lead to two different paths, that is, if the comparison of the model and expressions results are considered satisfactory, then the evaluation proceeds; otherwise, it returns to the previous modeling phase, where adjustments will be made until it shows compatible results between both formalisms.

4.1 Planning

The planning phase covers the first three steps of the modeling and evaluation methodology: (1) system understanding; (2) components identifications; and (3) metrics definition.

System understanding: this step consists of understanding the Hyperledger Fabric platform and the relationship between the Docker Engine and the container images (Peer, MSP, and Orderer nodes).

The so-called main requirements include most computer systems, including cloud platforms and their respective services, and corresponds to one of the first steps of the system's understanding process [9].

Components identification: survey all components required to accomplish service provisioning.

The prerequisites to create a blockchain environment based on the Hyperledger Fabric private or permissioned environment includes any modern hardware running Linux, MacOSX, or Microsoft Windows 10 Operating System with support to Docker Engine 18+. The Docker Engine is responsible for creating and managing the Hyperledger Fabric containers and making it easier for the deployment process.

After listing our relevant requirements and understand how they interact with each other (as already seen in Figure 1), we may determine a way to evaluate the system, which usually includes a path to obtain the needed information about it [9].

Metrics definition: the third step corresponds to the establishment of the system metrics that will be evaluated. Choosing metrics and evaluation parameters that have a low impact on the user and administrator perception will lead to a waste of time and resources [9], but availability related issues are trending among service providers.

4.2 Modeling

The modeling phase presents the other three steps related to modeling conception, including extracting availability expressions from the proposed models to use them on the proposed framework.

RBD Model Conception: in a hierarchical modeling strategy, we proposed an RBD model to present the tree base components presents in Figure 1, which are respectively Hardware, Operating System and Docker Engine. As already mentioned, there is no dependency regarding service provisioning availability, which means that the entire system could be represented through a single RBD. To generalize the model to just a set of expressions that could later be used in the framework, we also propose a CTMC.

Multilevel CTMC Model Conception: this model represents the combination of the previously proposed model (the RBD) with the Hyperledger Fabric containers, hierarchically representing the entire system.

Extraction of Availability Expressions: from the CTMC model, we could extract a set of availability expressions related to the three central endorsement policies used by Hyperledger Fabric, which are AND, OR, and K-out-of-N.

4.3 Evaluation

The evaluation phase presents the last four steps of modeling evaluation, including evaluating the availability expressions and models through a set of input values obtained from the literature review.

Survey Input Parameters: based on the planning and modeling phase and as a result of previous activities, we may obtain the input values that will be used by the proposed models by consulting previously published works, related works, and manufacturers' specifications.

Availability Models Evaluation: the proposed models were evaluated through the Mercury tool [13], right after it be fed with the set of input mean time to failure (MTTF) and mean time to repair (MTTR) values previously obtained; as an output, this step provides results that will be latter compared against the values obtained through the evaluation of the expressions.

Availability Expressions Evaluation: same as the previous step, and based on the input parameters, we performed an evaluation and obtained a set of results to be compared.

Results Comparison: the last step of the evaluation process consists of comparing both obtained results to prove that both models and expressions represent the same system. A result is considered a satisfactory result when we can obtain the same or similar availability value from both the high-level model and its expression. Then, the expression can finally be used by the framework.

5 Availability Models

This section presents the proposed availability models representing environments that can host Hyperledger Fabric's nodes and blockchain-based applications.

To represent a Hyperledger Fabric-based architecture and all components must be operational to perform service provisioning, we considered a two-stage hierarchically modeling. In the first stage, we adopted an RBD model representing the primary system's components: hardware, operating system, and container engine -Docker Engine. The system's components are presented in a serial RBD, meaning that if at least one of its components fails, then the whole system will fail as well. Assuming the failure rates are constant; hence we estimate the mean time to failure of system primary components by using Expressions 11 and 12.

$$R_{spc}(t) = R_{hw}(t) \times R_{os}(t) \times R_{de}(t), \tag{11}$$

Which is equivalent to

$$R_{spc}(t) = e^{-\lambda_{hw} t} \times e^{-\lambda_{os} t} \times e^{-\lambda_{de} t}, \tag{12}$$

where $R_i(t) = e^{-\lambda_i t}, \quad i \in \{hw, os, de\}$. Also, through Expressions 13 and 14 we show how the MTTF was calculated.

$$MTTF = \int_0^\infty e^{-(\lambda_{hw}+\lambda_{os}+\lambda de)\, t}\, dt, \tag{13}$$

which leads to,

$$MTTF = \frac{1}{\lambda_{hw} + \lambda_{os} + \lambda_{de}} \tag{14}$$

After evaluating the primary component's RBD, we had obtained their respective MTTF and MTTR. The next step in the system's modeling process deals with Hyperledger Fabric containers. Figure 4 presents the CTMC for the whole system, representing our second hierarchically modeling stage.

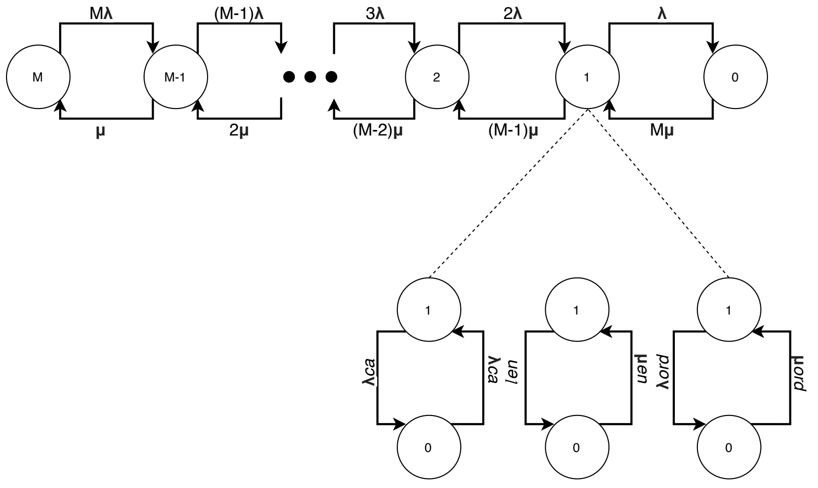


Fig. 4: Availability Model

This CTMC first level deals with the previously stated RBD, which may have up to M machines, each with its Hardware, Operating System, and Docker Engine. It is important to highlight that each machine represented in this model runs three containers (Endorser, MSP, and Orderer), represented by the CTMC's second level. The machines may enter into a failure state or be repaired following an exponential distribution-based rate λ and μ, respectively. These rates are the inverse of the MTTF and MTTR values obtained from a RBD. Already the Hyperledger Fabric's containers have their rates, which are the λ_{ca} and μ_{ca} that stands for the MSP fail and repair rates, λ_{en} and μ_{en} for the endorser container, and λ_{ord} and μ_{ord} represents the orderer rates.

From this model, by using the State Diagrams package in Wolfram Mathematica [29], we could extract a set of expressions to evaluate the availability of the system based on the aimed endorsement policy. The first expression is the server expression, that represents a model with only one physical machine and its three containers, as can be seen in Expression 15.

$$A_{\text{Server}} = \left(\frac{\mu}{\mu+\lambda}\right) \times \left(\frac{\mu_{ca}}{\mu_{ca}+\lambda_{ca}}\right) \times \left(\frac{\mu_{en}}{\mu_{en}+\lambda_{en}}\right) \times \left(\frac{\mu_{ord}}{\mu_{ord}+\lambda_{ord}}\right) \quad (15)$$

From this expression, we may calculate the availability (A) of a single server. Later, we have generalized this expression by using a binomial. The Expression 16 is the second expression; it represents the K-out-of-N endorsement policy, where someone can establish a K number of an M total of components that must be operational in order to accomplish service provisioning, that means to the system be available, and the endorsement is performed. This expression is a generalization of the previous one and can be used to calculate any servers and their associated container.

$$A_{\text{KooN}} = \sum_{i=k}^{M} \binom{M}{k} A_{\text{Server}}^{k} (1 - A_{\text{Server}})^{M-k} \quad (16)$$

It is important to mention that K stands for the number of components expected to be operational, and M is the total of resources that we have. Some specific scenarios may be extracted from the KooN policy, meaning that some other expressions can be obtained to calculate a combination of K and N values. The third Expression 17 represents the AND endorsement policy, which requires that all components in the first and second levels of the CTMC be operational, which means that it is an N out of N.

$$A_{\text{NooN}} = \left(\frac{\mu}{\mu+\lambda}\right)^{M} \times \left(\frac{\mu_{ca}}{\mu_{ca}+\lambda_{ca}}\right)^{M} \times \left(\frac{\mu_{en}}{\mu_{en}+\lambda_{en}}\right)^{M} \times \left(\frac{\mu_{ord}}{\mu_{ord}+\lambda_{ord}}\right)^{M} \quad (17)$$

The Expression 18 evaluates the system availability for a configuration when only one out of N components need to be operational. This expression presents the OR endorsement policy, where at least one of each type of component in both first and second levels of CTMC is required to be operational, meaning that the system is available and that we can perform the endorsement with only a single node and its containers from a total of N resources.

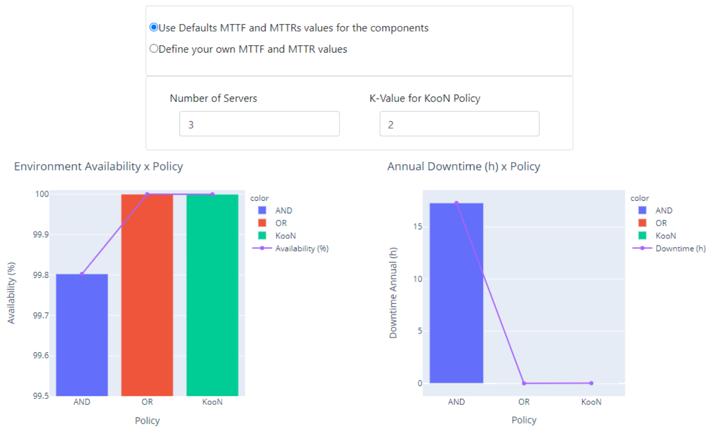
$$A_{1\text{ooN}} = \left(1 - \left(\frac{\lambda}{\lambda + \mu}\right)^{M}\right) \times \left(1 - \left(\frac{\lambda_{en}}{\lambda_{en} + \mu_{en}}\right)^{M}\right) \times \left(1 - \left(\frac{\lambda_{ord}}{\lambda_{ord} + \mu_{ord}}\right)^{M}\right) \times \left(1 - \left(\frac{\lambda_{ca}}{\lambda_{ca} + \mu_{ca}}\right)^{M}\right) \tag{18}$$

6 Framework Overview

The Blockchain Provisioning Planning Tool (BPPT), as the name says, it is a tool for planning the provisioning of a blockchain environment based on the Hyperledger Fabric Infrastructure[4]. With the BPPT, one can plan both availability-related metrics and the expenses related to maintenance and deployment of private computational infrastructures and compare it against three of the most popular public cloud service providers: Amazon AWS, Google Cloud, and Microsoft Azure.

6.1 Availability Planner

The Availability Planner uses the expressions previously extracted from the multilevel CTMC model to determine the system's availability and its respective annual downtime. The Figure 5 presents the availability planner.



[✓ Use Defaults MTTF and MTTRs values for the components]
Define your own MTTF and MTTR values

Number of Servers: 3
K-Value for KooN Policy: 2

Fig. 5: Availability Planner

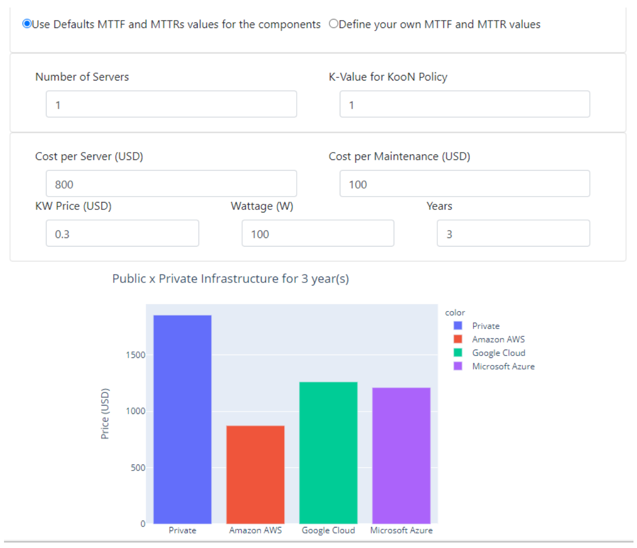
The administrator can manually provide their MTTFs and MTTRs values for each system's component or use a default set of values obtained through

[4] BPPT: `https://blockchain-bppt.herokuapp.com/apps/meu_app`

the literature review. Other parameters that can be changed are the number of available servers and the K value when the decision-maker plans to consider a K-out-of-N endorsement policy.

6.2 Cost Planner

The cost planner includes the previously stated input parameters plus a set of electric and maintenance expenses. Figure 6 shows the upper part of the cost planner.



[✓ Use Defaults MTTF and MTTRs values for the components] Define your own MTTF and MTTR values

Number of Servers	1
K-Value for KooN Policy	1
Cost per Server (USD)	800
Cost per Maintenance (USD)	100
KW Price (USD)	0.3
Wattage (W)	100
Years	3

Fig. 6: Cost Planner Upper Level

The user can define the cost per server in American dollars and the maintenance costs, and the city's kilowatt price or state that he lives. It is also possible to determine server wattage. As for the electric costs, we could achieve real values for the energy consumption of the selected components by using the Equation 19.

$$\mathrm{E} = \frac{\mathrm{Power\ (W)} \times \mathrm{NHD} \times \mathrm{NDY}}{1000} \tag{19}$$

where E stands for energy costs, NHD for the number of hours per day and NDY is the number of days per year that the equipment is operational.

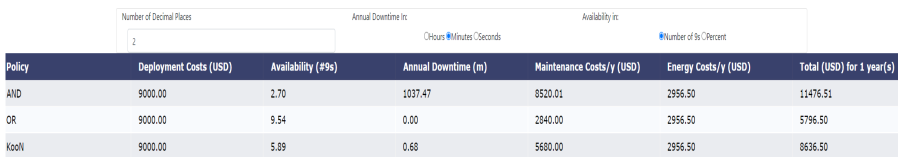
At a comparison level, we may choose public cloud computing platforms, such as Google Cloud, Amazon AWS, and Microsoft Azure. These three platforms provide environments that can be used to host Hyperledger Fabric-based applications. As expected, each platform has its characteristics. We chose the most basic container/Virtual Machine for each platform. Table 4 summarizes the selected environments.

Table 4: Annual Expenses by Instance in Public Clouds

Platform	**Instance Type**	**On-demand/y (USD)**	**Availability (%)**
Amazon	n1-standard-1	291.24	99.9
Google	m3.medium	420.48	99.9
Microsoft	D1s	403,56	99.9

The cost planner has obtained results supporting comparing the deployment and maintenance expenses of the private infrastructure with virtual machines provided by Amazon AWS, Google, and Microsoft Azure, extending the number of years to be evaluated in an on-demand model. It is essential to mention that the acquisition costs are considered only in the first year, which implies that the subsequent years will consider only electric and maintenance values.

Figure 7 shows the lower level of cost planner, which presents more detailed information about the policies being used, the 1st year deployment expenses, the expected annual downtime that can be reached, and the annual maintenance costs by considering the number of failures.



Number of Decimal Places: 2

Annual Downtime In: Hours [✓ Minutes] Seconds

Availability in: [✓ Number of 9s] Percent

Policy	Deployment Costs (USD)	Availability (#9s)	Annual Downtime (m)	Maintenance Costs/y (USD)	Energy Costs/y (USD)	Total (USD) for 1 year(s)
AND	9000.00	2.70	1037.47	8520.01	2956.50	11476.51
OR	9000.00	9.54	0.00	2840.00	2956.50	5796.50
KooN	9000.00	5.89	0.68	5680.00	2956.50	8636.50

Fig. 7: Cost Planner Lower Level

The AND endorsement policy seems to be the one with the highest number of failures since all machines need to be operational to accomplish service provisioning. So, maintenance will be called much more times than with an OR endorsement policy and usually a KooN endorsement policy.

6.3 Experiment

The experimentation screen allows the user to see the impact of a single component (simple experiment) or a set of components (general experiment) over the general system's availability, based on the input parameters' variation. The user can establish the number of sampling points to be plotted and the positive range in hours that a parameter varies. This can be considered a parametric sensitivity analysis method, so by putting everything together, one can point out the components' impacts that most affect the metrics of interest: the availability and the annual system's downtime.

7 Case Studies

This section provides a case study that demonstrates how feasible is the proposed framework. We evaluate both Availability and Cost planning.

The first step to accomplish this task is to obtain the required system's input values to feed the framework and perform the availability evaluation. Some of these values were obtained from a literature review [3,17,23], while other ones were extracted from manufacturer charts and white papers. The input values used in this paper are the same as the ones used as default in the BPPT framework and can be seen in Table 5.

Table 5: Input Parameters for Availability Evaluation

Component	MTTF (h)	MTTR (h)
Hardware (HW)	8760	1.66
Operating System (OS)	2893	0.15
Docker Engine (DE)	2516	0.15
Containers	1258	0.15

Also, it is important to mention that ten different scenarios were evaluated. These scenarios are presented in Table 6. Each one considers a possible combination of endorsement policy and several nodes from 1 up to 4 servers hosting the three containers.

Table 6: Evaluated Scenarios

Scenario	Policy	Required Servers	Total Servers
1	AND	1	1
2	AND	2	2
3	AND	3	3
4	AND	4	4
5	OR	1	2
6	OR	1	3
7	OR	1	4
8	KooN	2	3
9	KooN	2	4
10	KooN	3	4

7.1 Availability Planning

After listing the input availability values and ten different scenarios, each one following either OR, AND, or KooN endorsement policy, we have evaluated their respective availability. Table 7 presents the general availability results obtained for each proposed scenario.

Table 7: Availability Results

Scenario	Av. (%)	Av. (#9s)	A. Downtime (h)
1	99.9341	5.77	5.77
2	99.8683	2.88	11.53
3	99.8026	2.70	17.29
4	99.7369	2.58	23.05
5	99.9998	6.36	0.0038
6	99.9999	9.54	0.000003
7	99.9999	12.73	0.00000002
8	99.9987	5.89	0.011
9	99.9999	8.94	0.000009
10	99.9997	5.59	0.022

As can be seen, the availability (Av.) when evaluating an AND policy goes down as we add more resources (nodes), which is already expected since all components must be operational, and in case of failure of any of them, the transaction's endorsement can not be performed. Already when considering an OR endorsement policy, the availability grows up as more resources are added. Last but no less important, the KooN endorsement policy shows a mid-term availability compared to an AND and an OR policy; it goes down as more nodes are needed, meaning that it tends to an AND policy. The same results can be achieved by considering the Annual (A.) Downtime for each scenario and endorsement policy.

7.1.1 Baseline Experimentation

We have conducted an experiment regarding the input parameters' impact on the overall system's annual downtime. To accomplish this task, we varied and rounded each parameter value in +50% and -50% as in the given time interval, which is presented in Table 8.

Table 8: Experiment Parameters

	Interval (h)	
Parameter	**Min.**	**Max.**
MTTF HW	4400	13200
MTTR HW	0.5	2.1
MTTF OS	1450	4340
MTTR OS	0.05	1
MTTF Docker	1260	3775
MTTR Docker	0.05	1
MTTF Container	630	1890
MTTR Container	0.05	1

The obtained results are presented in Figure 8. From this figure, we can see that the higher the MTTF, the lower the annual downtime, while the lower the MTTR, the higher the obtained downtime values, which should be the expected behavior. Among these results, we may highlight the Container MTTF and MTTR (g and h-subfigures); they seem to be the bottleneck regarding the system's availability associated metrics.

7.2 Costs Planning

As we previously did with the availability planning, we also need a set of input values to evaluate the costs of deploying and maintaining an environment. Table 9 presents the costs adopted in this case study.

Table 9: Costs Planning Input Values

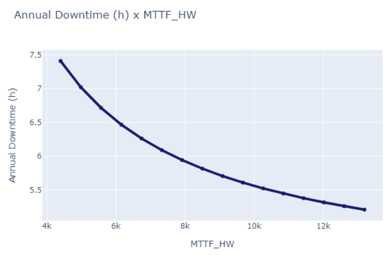
Parameter	**Value (USD)**
Cost p/ Server (USD)	1000
Cost p/ Maintenance (USD)	100
KWh Price (USD)	0.3
Server Wattage (W)	120

Table 10 presents the general costs values associated to the previous input values. We had also considered the same ten scenarios previously presented in the availability evaluation subsection, as shown in this table.

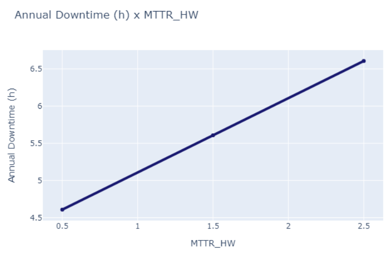
Table 10: General Costs Results

Scenario	**Costs/y (USD)**	**Costs/y (USD)**	**Total Costs (USD)**
1	2,840.00	315.36	3,155.36
2	5,680.00	1,261.44	6,941.44
3	8,520.00	2,838.24	11,358.25
4	11,360.01	5,045.76	16,405.77
5	2,840.00	1,261.44	4,101.44
6	2,840.00	2,838.24	5,678.24
7	2,840.00	5,045.76	7,885.76
8	5,680.00	2,838.24	7,572.16
9	5,680.00	5,045.76	8,202.88
10	8,520.01	5,045.76	12,304.32

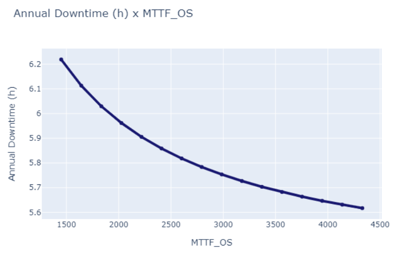
It is essential to mention that the cooling system's energy consumption is considered equal to the energy consumed with the server's operation. Simultaneously, the evaluated annual maintenance expenses were based on the occurrence of failures and a maintenance value of 100 dollars per call. This value is arbitrary and usually varies according to who is going to perform the maintenance. If the service provider already has or pays someone to perform corrective maintenance on their system, these values tend to be zero on the



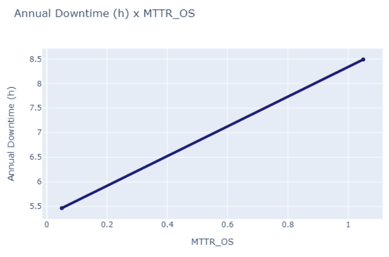
(a)



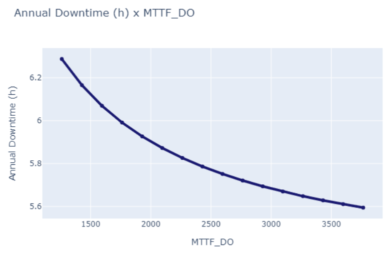
(b)



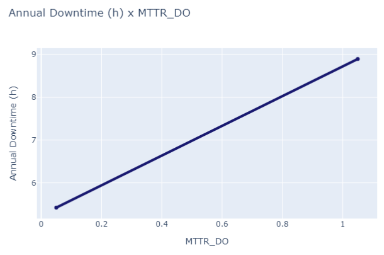
(c)



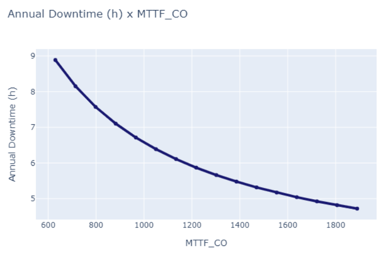
(d)



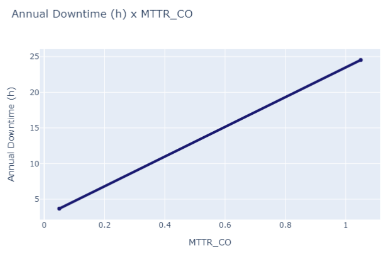
(e)



(f)



(g)



(h)

Fig. 8: Experimental Results

framework. Otherwise, if the provider wishes to hire someone for this task, which could be a freelancer or a specialized company, these costs should be included in the evaluation process.

The AND endorsement policy has higher maintenance costs since it needs more maintenance than an OR and KooN based endorsement policy. The maintenance values required for an OR policy are lower, at least compared to AND policy infrastructures, because a system's failure only happens when all components are unavailable, then we finally call the maintenance team.

A lower-cost can be obtained with a KooN policy, comparing it with an AND policy, and another factor may impact the choice, which is a limitation of the present work and framework itself; we do not consider threats to safety-related items. Hence, an OR endorsement policy leads to a higher chance that a transaction could tamper. If a single node has tampered with, the entire infrastructure can be compromised. In an AND endorsement policy, all nodes must tamper. A KooN endorsement policy requires at least K out of N nodes tamper information on the chaincode.

All these factors must be led into account when the administrator decides to migrate to a new environment, the deployment and maintenance costs, availability, annual downtime, and safety issues when he decides on a specific endorsement policy. Many have to be done; the BPPT is only the first tool to help them decide what to do. However, it still has to be improved.

8 Conclusions and Future Works

This paper presented the BPPT, a framework to evaluate availability and costs associated with the deployment and maintenance of private computational infrastructures. The BPPT focus on the Hyperledger Fabric platform and its components for hosting blockchain-based applications.

BPPT are two models of two different modeling formalism, an RBD and a CTMC; both represent the Hyperledger Fabric environment and are organized hierarchically. The availability expression that represents these models were extracted using Wolfram Mathematica and Mercury modeling tool.

As for the maintenance and deployment expenses, the BPPT receives input parameter values related to kilowatt, server wattage, and maintenance values. The BPPT also presents an experiment screen, where the stakeholders and decision-makers may evaluate the impact of varying the parameters on systems availability and associated annual downtime. We presented case studies that dealt with availability and costs evaluation of a set of scenarios varying the endorsement policy, which is a vital characteristic of a Hyperledger Fabric infrastructure.

We intend to update the BPPT with performance-related planning and an automatized sensitivity analysis mechanism that could help decision-makers obtain the aimed values related to the impact of a component over a metric of interest more quickly. Also, some safety information is required since it dramatically impacts the possibility of adopting the blockchain paradigm.

Is it wise to change from a common distributed database to this so-called new technology? We already know the costs and availability, but safety and performance are yet to come.

Acknowledgements The authors would like to thank the Brazilian Government for the financial support through the Fundação de Amparo a Ciência e Tecnologia de Pernambuco (FACEPE), the Modeling of Distributed and Concurrent Systems (MoDCS) group for the help on improving this research.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Avizienis, A., Laprie, J., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing **1**, 11–33 (2004)
2. Avižienis, A., Laprie, J., Randell, B., of Newcastle upon Tyne. Computing Science, U.: Fundamental Concepts of Dependability. Technical report series. University of Newcastle upon Tyne, Computing Science (2001). URL `https://books.google.com.br/books?id=cDkmGwAACAAJ`
3. Dantas, J.: Modelos para analise de dependabilidade de arquiteturas de computação em nuvem. Master's thesis, Centro de Informática - Universidade Federal de Pernambuco (Recife, Brasil) (2013)
4. Ekblaw, A., Azaria, A., Halamka, J.D., Lippman, A.: A case study for blockchain in healthcare: "medrec" prototype for electronic health records and medical research data. In: Proceedings of IEEE open & big data conference, vol. 13, p. 13 (2016)
5. Garg, S., A, P., M, T., Trivedi, K.S.: Analysis of software rejuvenation using markov regenerative stochastic petri net. In: Proc. In: Sixth International Symposium on Software Reliability Engineering, (ISSRE'95), pp. 180–187. Paderborn (1995)
6. Hao, Y., Li, Y., Dong, X., Fang, L., Chen, P.: Performance analysis of consensus algorithm in private blockchain. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 280–285. IEEE (2018)
7. Hyperledger: An introduction to hyperledger. Tech. rep. (2018)
8. Hyperledger: Introduction to hyperledger business blockchain design philosophy and consensus. Tech. rep. (2018)
9. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley Computer Publishing, John Wiley & Sons, Inc., New York (1991)
10. Kumar, M.V., Iyengar, N.C.S.N., Goar, V.: Employing blockchain in rice supply chain management. In: Advances in Information Communication Technology and Computing, pp. 451–461. Springer
11. Kuo, W., Zuo, M.: Optimal Reliability Modeling: Principles and Applications. Wiley (2003). URL `https://books.google.com.br/books?id=vdZ4Bm-LnHMC`
12. Maciel, P., Lins, R., Cunha, P.: Uma Introducao as Redes de Petri e Aplicacoes. Sociedade Brasileira de Computacao (1996)
13. Maciel, P., Matos, R., Silva, B., Figueiredo, J., Oliveira, D., Fé, I., Maciel, R., Dantas, J.: Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 50–57 (2017). DOI 10.1109/PRDC.2017.16

14. Maciel, P., Trivedi, K., Matias, R., Kim, D.: Dependability modeling. In: Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions (2011)
15. Malhotra, M., Trivedi, K.: Power-hierarchy of dependability-model types. Reliability, IEEE Transactions on **43**(3), 493–502 (1994). DOI 10.1109/24.326452
16. Matos, R., Araujo, J., Oliveira, D., Maciel, P., Trivedi, K.: Sensitivity analysis of a hierarchical model of mobile cloud computing. Simulation Modelling Practice and Theory **50**, 151 – 164 (2015). DOI https://doi.org/10.1016/j.simpat.2014.04.003. URL `http://www.sciencedirect.com/science/article/pii/S1569190X14000616`. Special Issue on Resource Management in Mobile Clouds
17. Melo, C., Dantas, J., Araujo, J., Maciel, P.: Availability models for synchronization server infrastructure. In: Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'16). Budapest, Hungary (2016)
18. Melo, C., Dantas, J., Maciel, R., Pereira, P., Quesado, E., Maciel, P.: Blockchain provisioning over private cloud computing environments: Availability modeling and cost requirements. In: 2019 IEEE 8th International Conference on Cloud Networking (CloudNet), pp. 1–3. IEEE (2019)
19. Melo, C., Dantas, J., Maciel, R., Silva, P., Maciel, P.: Models to evaluate service provisioning over cloud computing environments-a blockchain-as-a-service case study. Revista de Informática Teórica e Aplicada **26**(3), 65–74 (2019)
20. Öhmann, D., Simsek, M., Fettweis, G.P.: Achieving high availability in wireless networks by an optimal number of rayleigh-fading links. In: 2014 IEEE Globecom Workshops (GC Wkshps), pp. 1402–1407. IEEE (2014)
21. Pongnumkul, S.e.a.: Performance analysis of private blockchain platforms in varying workloads. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1–6. IEEE (2017)
22. Roy, G.G.R., Kumar, S.B.R.: A security framework for a sustainable smart ecosystem using permissioned blockchain: Performance evaluation
23. Sebastio, S., Ghosh, R., Mukherjee, T.: An availability analysis approach for deployment configurations of containers. IEEE Transactions on Services Computing (2018)
24. Sukhwani, H., Martínez, J.M., Chang, X., Trivedi, K.S., Rindos, A.: Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric). In: 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), pp. 253–255. IEEE (2017)
25. Sukhwani, H., Wang, N., Trivedi, K.S., Rindos, A.: Performance modeling of hyperledger fabric (permissioned blockchain network). In: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), pp. 1–8. IEEE (2018)
26. Sun, Y., Zhang, L., Feng, G., Yang, B., Cao, B., Imran, M.A.: Blockchain-enabled wireless internet of things: Performance analysis and optimal communication node deployment. IEEE Internet of Things Journal **6**(3), 5791–5802 (2019)
27. Thakkar, P., Nathan, S., Viswanathan, B.: Performance benchmarking and optimizing hyperledger fabric blockchain platform. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 264–276. IEEE (2018)
28. Trivedi, K.S., Hunter, S., Garg, S., Fricks, R.: Reliability analysis techniques explored through a communication network example (1996)
29. Weisstein, E.W., et al.: Mathworld–a wolfram web resource (2004)