CHAIRE FINTECH
AMF-Finance Montréal
ESG UQÀM

CAHIER DE RECHERCHE DE LA CHAIRE FINTECH AMF – FINANCE MONTRÉAL

Towards Scalable Systems for Securities on Blockchains

Par Kaiwen Zhang
École de technologie supérieur, ÉTS.

Décembre 2021

Projet réalisé dans le cadre du 2ème appel de projets de la Chaire « Les Fintechs du Québec : du développement de l'écosystème à l'expérience, de la réglementation aux enjeux de sécurité »

École de technologie supérieure (ÉTS)
1100, rue Notre-Dame Ouest
Montréal, (QC), Canada, H3C 1K3
Tél. : (1) 514 396 8735
Département de génie logiciel et TI

Rapport final

Towards Scalable Systems for Securities on Blockchains

Prof. Kaiwen Zhang, Ph.D.

1. État final du projet

Dans l'ensemble, le projet s'est bien déroulé. Grâce aux fonds du projet, nous avons pu financer un étudiant au doctorat à l'ÉTS pour 12 mois. Il s'agit de Yahya Shahsavari.

Nous avions le mandat suivant :

Subproject 2 (Prof. Kaiwen Zhang) -
The design of performance models for blockchain networks, with Quorum as a starting point. Our performance model will allow us to optimally tune the system with respect to key parameters such as the number of nodes, block size, and block time. We will then demonstrate the applicability of our approach by evaluating the performance (throughput, confirmation time, transaction fees) of the smart contract prototype generated in (1).

Essentiellement, notre objectif était de modéliser la performance du système Quorum de JP Morgan/Consensys. Nous avons légèrement modifié cet objectif en incorporant aussi le système Diem de Meta. Les avancements du projet sont :

1. Nous avons modélisé le système Diem de Meta. Nous avons réalisé un modèle complet validé par nos simulations. Nous avons rédigé un article que nous allons soumettre à la conférence IEEE ICDCS 2022. L'article est attaché en annexe.
2. Nous avons adapté le modèle proposé pour modéliser IBFT de Quorum. Nous avons déjà écrit le chapitre qui couvre se modèle (voir l'annexe). Nous allons utiliser ce chapitre pour écrire une version étendue de l'article précédent qui va être soumise à une revue académique.

Les étapes qui restent à compléter sont :

1. Ajouter des résultats de simulation de Quorum pour valider le modèle développé.
2. Utiliser nos modèles pour analyser les applications étudiées dans le sous-projet #1, mené par le professeur Jeremy Clark de Concordia. Nous avons identifiés les « roll-ups » comme étant un bon cas d'utiisation. Les « roll-ups » sont des agrégations de transaction financières qui sont effectués hors-chaîne, ce qui permet d'alléger la charge du système. Nous allons utiliser notre modèle de performance pour étudier le potentiel d'amélioration de performance des « roll-ups » dans les systèmes Diem et Quorum. Nous soumettre un autre article de conférence avec le titre « Performance Modeling of Roll-Up Mecanisms in Blockchain Systems ».
3. Nous allons présenter nos travaux à la conférence de la chaire AMF, ainsi que déposer un rapport final dans le cahier de recherche de la chaire, avec un article pour le site web de la chaire.

   Il se peut que ces travaux dépassent la fin du projet. Nous allons utiliser d'autres fonds de recherche pour financer les études de Yahya afin d'assurer qu'il complète ses études.

Performance Modeling and Analysis of PBFT Consensus for the Diem Blockchain Network

Authors names blinded for review

***Abstract*—Facebook has recently introduced a new cryptocurrency called Diem, previously known as Libra. Contrary to popular cryptocurrencies such as Bitcoin and Ethereum, Diem operates on a permissioned network maintained by independent members of the Diem Association. Thus, Diem does not employ Proof-of-Work (PoW) and uses its own Practical Byzantine Fault-Tolerant (PBFT) consensus mechanism which is a variant of HotStuff, a well-known modern consensus algorithm. Previous studies on PBFT algorithms typically focus on the threshold for correctness (e.g., total number of peers of $3f + 1$) or the communication complexity (e.g., $\mathcal{O}(n)$ view changes). However, these works do not address the impact of network parameters on the performance of the system.**

In this paper, we seek to address this research gap by analyzing the performance of PBFT consensus, as implemented in the Diem blockchain, from a networking perspective. We present a theoretical model which accurately predicts blockchain-related metrics such as the transaction throughput and expected confirmation time using important networking parameters such as the number of validators, link latency, and packet loss. Furthermore, we validate our model through extensive simulations carried out using OMNeT++. Our results show that the QoS level of the network has a major impact on the probability of successfully achieving consensus in Diem. Furthermore, the number of faulty nodes will affect the throughput of the network, even when the number of honest nodes is greater than the established threshold bound of $3f + 1$.

***Index Terms*—Diem, performance modeling, blockchain, throughput, HotStuff, PBFT, Libra**

I. Introduction

After the successful launch of Bitcoin, many cryptocurrencies have been deployed so far with varying capabilities. While most cryptocurrencies employ a permissionless public network, Facebook has introduced a new cryptocurrency called Diem (previously known as Libra [1]) which operates on a permissioned network. The Diem network consists of a set of geographically distributed replicas referred to as validators that form the Diem association and collaborate together to process transactions and reach a consensus on the state of the distributed ledger.

A major feature of the Diem network is that it allows for the use of non-Proof-of-Work-based (PoW) consensus algorithms. Hence, Diem utilizes DiemBFT [2], which is a variant of HotStuff protocol [3]. This choice of consensus algorithm allows Diem to outperform PoW-based systems by offering superior transaction throughput and minimizes the energy consumption, at the expense of full decentralization.

DiemBFT belongs to a class of consensus algorithms called Practical Byzantine Fault-Tolerance (PBFT) which has been well-studied in the literature for several decades [4]–[8]. Previous studies on PBFT algorithms typically focus on the threshold for correctness (e.g., total number of peers of $3f + 1$) or the communication complexity (e.g., $\mathcal{O}(n)$ view changes) [9]–[14]. However, these works do not address the impact of network parameters such as network size or the number of faulty nodes on the performance of the system. Furthermore, PBFT studies predate blockchain technologies; thus, very few studies exist on the performance of PBFT consensus in the context of blockchain networks. For instance, to the best of our knowledge, there are no existing works that explain how additional honest peers beyond the minimum safety threshold of $3f+1$ total peers contribute to blockchain-related performance metrics such as transactional throughput and confirmation time.

In this paper, we seek to address the aforementioned research gap by modeling the performance of the Diem PBFT consensus algorithm. As it is the principal component of a blockchain system, it is imperative to analyze its performance in order to gain insights into the behavior and dynamics of the system, thereby facilitating the optimal configuration of the network without requiring iterative benchmarking. Contributions of our paper are as follows:

1) We present a theoretical model for the DiemBFT consensus mechanism and derive explicit mathematical equations to describe the behaviour of the system and identify the relationship between different metrics affecting the performance of Diem blockchain.
2) We validate our theoretical model through extensive simulations using the network simulator OMNeT++, which is a discrete event simulator.
3) We estimate the throughput or the number of transactions per second (TPS) in Diem. As well, we identify the most effective parameters which impact the performance of Diem (e.g., network size, number of faulty nodes, and transaction processing time).

The rest of this article is organized as follows: in Section II, we review previous works related to PBFT consensus and permissioned blockchain networks. In Section III, we present an overview of the Diem blockchain system as background material necessary to understand our paper. In Section IV, we present our proposed analytical approach for modeling the performance of Diem. In Section V, we validate our developed model through extensive simulations and study the performance of the Diem blockchain. Finally, Section VI concludes our paper.

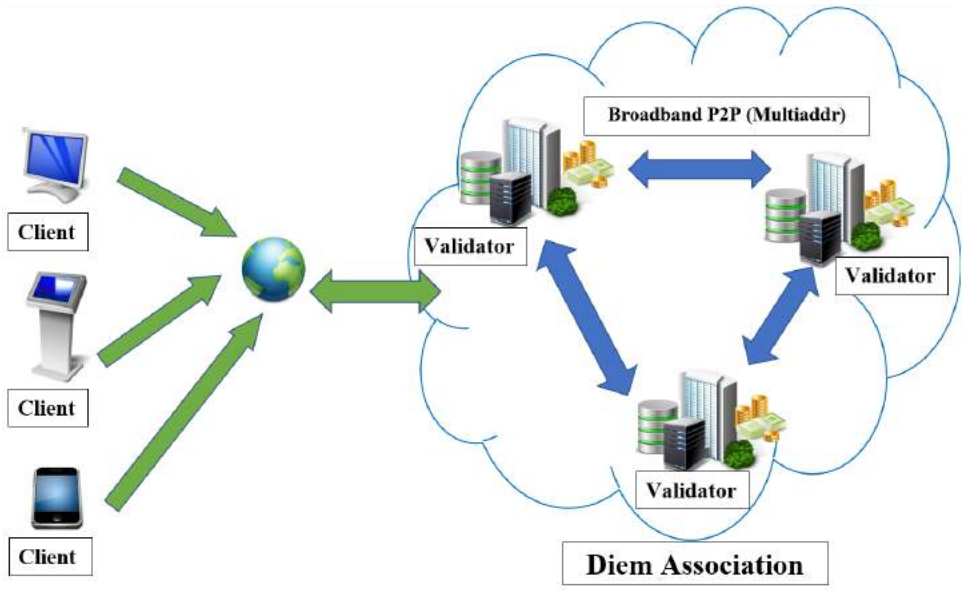


Fig. 1: Architecture of the Diem network: validators are peers belonging to members of the Diem association (e.g., central banks and multinational corporations). Each validator maintains a full copy of the blockchain distributed ledger.

II. Related Works

In this section, we survey existing works on performance modeling and analysis of Diem and related blockchains. To the best of our knowledge, there are few works on the performance of Diem as it is a relatively recent system. Nevertheless, the Practical Byzantine Fault Tolerance (PBFT) family of consensus algorithms which includes HotStuff, which is the original design of DiemBFT, has been well studied in the literature and is relevant to our work. However, no prior work proposes a theoretical performance model for the analysis of the Diem blockchain network.

In [15], a theoretical performance model for PBFT is presented. In this work, analytical formulas for estimating the performance of PBFT are extracted and validated through extensive simulations and impact of unreliable channels and the use of different transport protocols (i.e TCP and UDP) over them are discussed. However, it does not propose explicit formulas for configuration parameters such as network size or number of faulty nodes.

In [16], a comparison of PoW and PBFT is presented. According to the reported results, PBFT offers a better latency while PoW is more scalable with respect to the size of the network.

An experimental study of Diem is presented in [17]. The performance and scalability of Diem are evaluated and compared with Hyperledger Fabric, which is another popular permissioned blockchain system. The evaluation metrics are throughput (transactions per second, TPS) and execution time (i.e., the time taken to commit a block of transactions in the ledger). Parameters include the number of peers and the system workload (i.e. total number of transactions). According to the authors, the throughput of the system decreases and execution time increases when the number of peers increases for both platforms. As well, with respect to increasing the number of transactions, the throughput of both systems increases at first then decreases. A reverse trend is reported for the execution time. Unlike our work, this paper is based only on experimental results and does not study the performance and scalability of the system in the presence of faulty nodes.

A brief comparison of Bitcoin, Ethereum and Libra is presented in [18]. However, it reports the results of [17] directly and does not provide any independent performance analysis.

In [19], a classification of consensus mechanisms in blockchain-based systems is presented. In this paper, HotStuff is studied as the basis of DiemBFT. According to this work, the advantage of Diem is its low confirmation latency. However, Diem suffers from low network size scalability, high communication cost, and weak network synchronization.

In [20], a survey of empirical performance evaluation of permissioned blockchains is presented. However, this work does not introduce a systematic approach for the analysis of blockchain systems based on the selected metrics (e.g., network size scalability and the number of transactions).

[21] presents a mathematically formalized study of forensic support for PBFT-based consensus protocols in the presence of the maximum number of faulty replicas. This work claims that minor variations in the PBFT protocols can profoundly affect forensic support. Another result reported in this paper is that DiemBFT exhibits a strong forensic support capability.

In [22], an evaluation framework for evaluating the performance of pipelined Hotstuff is presented. Evaluation metrics are the rate of chain growth, quality of the chain, and latency. This framework is leveraged for evaluating some optimization schemes for Diem. According to the results reported in this paper, these optimizations make the Diem system vulnerable

to some kinds of attacks. However, this work does not present an analytical model for estimating the performance of Diem based on the network configuration parameters.

In this paper, we present an analytical model which describes the relationship between configuration and set up parameters and selected performance measures. There is a similar work for PoW-based systems such as Bitcoin [23] and proposes explicit formulas for performance measures such as block propagation speed. But in this paper we focused on BFT based consensus protocols and in particular, DiemBFT.

III. Background on Diem

In this section, we briefly introduce Diem and present the required background that is necessary to understand this paper. We study the Diem protocol and related concepts in order to develop a theoretical model for the performance analysis of the system.

Diem is a permissioned blockchain-based system that seeks to maintain a decentralized and programmable database of resources revolving around the Diem cryptocurrency. According to Diem blockchain white paper [1], these resources are owned by a variety of user accounts that are already authenticated by public key cryptographic schemes according to the set of rules defined by the original developers of the Diem. Core functionalities of the Diem are defined using its programming language: *Move*. Move is also being used for developing smart contracts, as well as defining the user memberships in the Diem ecosystem.

A. Architecture overview

Diem is a permissioned blockchain-based system that consists of a number of geographically distributed governor nodes referred to as *validators*. These validators are members of the Diem association [1]. This consortium consists of a set of founding members (e.g., well-reputed central banks and large companies) which back the Diem coin with treasuries and cash deposits. In fact, certain participation criteria such as a minimum amount of committed stake is required to join the network in bootstrapping Diem.

An overview of the Diem architecture is depicted in Figure 1. The Diem system consists of two entities: validators and *clients*. Validators are responsible for maintaining the distributed ledger of programmable resources (e.g., Diem coin). Hence, each validator keeps a full copy of the ledger, while this is optional for clients.

Validators form a fully connected network via peer-to-peer (P2P) links. Clients submit their transactions to the validators to be included in the ledger. These pending transactions are stored by the validators in a shared mempool, which is synchronized regularly. Validators use the DiemBFT protocol to decide which transactions to commit to the ledger (see the next Subsection III-B). Similar to classical PBFT, the consensus algorithm iterates through rounds. In each round, a leader is selected using a verifiable random function [24]. The leader proposes a sequence of transactions as a new block. Then, the leaders broadcast the proposed block to the rest of the validators. All of the validators should execute the transactions in the received block. Finally, they will send their vote to the next leader.

B. Diem consensus protocol

The DiemBFT consensus protocol is an implementation of the HotStuff consensus algorithm [3], with one major optimization that reduces the round synchronization overhead to one message per node during each round. The safety and liveness of the system are guaranteed assuming a partial synchrony model [5] which tolerates Byzantine failures [4]. In order to remain live, all the messages from the honest validators should be delivered to the other honest validators within a maximal network delay δ [1].

DiemBFT protocol assumes that the network consists of $3f + 1$ validator nodes where f is the maximum number of tolerable Byzantine or faulty nodes. In other words, in a network of N validator nodes where $N = 3f + 1$, faulty nodes can control at most f votes, and the rest of the votes are controlled by the honest nodes.

Clients submit a proposed transaction to the validators in order to change the ledger state. Then, they start a timer and wait for the transaction to be committed to the ledger. If the timer of a client expires, it aborts the submitted transaction and starts another submission. Validators collect the submitted transactions in their shared mempool.

Processing the submitted transactions takes place within a sequence of rounds. At the beginning of each round, one of the validators gets selected as the leader of the round by the proposer of the latest block using a verifiable random mechanism that cannot be predicted in advance. This verifiable random function reduces the chance of an attacker launching a successful denial of service (DoS) attack against the leader.

The leader of the round packs a batch of transactions into a proposed block denoted by B_k and broadcasts it to the rest of the validators. Upon receiving the proposed bock, validators check their voting rules and decide whether to vote for certifying this block or not. When a validators decision is to certify the block, it speculatively executes all the transactions embedded in the block without committing it to the ledger. Then, it sends a signed vote for block B_k to the leader of the next round. The leader of the next round collects the votes of the validators. If the number of collected votes is equal to or greater than $2f + 1$, it issues a quorum certificate (QC) for the block B_k. Simultaneously, the leader of the current round proposes its own block (denoted by B_{k+1}) which is containing the transactions that are not already packed in a block or proposal and embeds the issued QC for B_k in it. Similarly, the leader broadcasts the new proposal to the rest of the validators. In the same way, validators will send their votes for this proposal to the leader of the next round. The leader of the next round collects the votes and checks the number of votes for B_k and B_{k+1} while it proposes its own block denoted by B_{k+2}. If the number of votes for QC of B_k is equal to or greater than $2f + 1$, the leader issues a quorum certificate of quorum certificate (QC-of-QC) for B_k. As well,

if the number of collected votes for B_{k+1} is equal to or greater than $2f+1$, the leader issues a QC for B_{k+1} and embed them to its own proposal denoted by B_{k+2} broadcasts it to rest of the validators. In the same way, the leader of the next round can issue a QC-of-QC-of-QC, a QC-of-QC, and a QC for B_k, B_{k+1}, and B_{k+2} respectively. When a proposed block gets three consecutive QCs, it is then committed to the ledger.

C. Networking

According to [1], the Diem network is a general-purpose network inspired by *libp2p* [25]. Moreover, the inter-validator network utilizes a P2P architecture based on *Multiaddr* [26] for addressing the validators. TCP is employed as a reliable transport protocol. The end-to-end traffic is fully encrypted using *Yamux* [27]. In the Diem network, any validator should directly connect to other validators. If any validator can not be connected directly, it will be treated like a faulty or Byzantine peer.

IV. Proposed Analytical Model

In this section, we explain our proposed theoretical model which we use to derive explicit equations for calculating two key blockchain metrics: transaction throughput and block processing time (i.e, the amount of time required from proposing a block until it gets a QC-of-QC-of-QC from one of the leaders).

A. Assumptions

Assume a network of N validators with the same processing power that should be fully connected pairwise over a P2P network (i.e., each validator is directly connected to $N-1$ validators). Any missing connection should be considered a Byzantine failure. We assume transmission delay has an exponential distribution in P2P links with a mean of $1/\mu$. Although in reality, the latency of the links does not necessarily follow an exponential distribution, nevertheless developing a theoretical model without simplification assumptions will not be a trivial job since the relationship between parameters will become intractable. The exponential distribution is fully described in [28] and has frequently been used for modeling asynchronous networks in literature (e.g. [29], [30]). As well, we assume all of the messages reach to destination with a constant probability of p (or get lost or dropped with a constant probability of $1-p$). Note that validators also set a timer in each round, and can give up a certain round if they do not receive the related messages within a predetermined interval of time. Hence, although the Diem system utilizes TCP, and thus all of the packets will eventually be delivered to the destination, the packets should also arrive at their destination within a maximum delay of δ. Otherwise, they will be considered as lost messages. If a validator's timer expires, it will send a time-out message instead of a positive/negative vote or QC and will give up the round. If a leader gathers time-out messages from a quorum of validators, it will broadcast a time-out certificate (TC) instead of QC and the round will be aborted. In this paper, we treat a timed-out validator as a faulty node.

We assume all of the validators are acting independently, as intended by the Diem association. As well, assume f out of N validators are faulty. Suppose a set of clients have submitted their transaction proposal to the shared mempool of the validators. Let us denote the leader of the K'th round with L_K. Assume L_K (i.e. V_1 in Figure 2) proposes a block containing a bunch of transactions as mentioned above for the first time as depicted in Figure 2. In order to be eligible to be appended to the blockchain, the proposed block must meet three contiguous QCs in three consecutive rounds of K, $K+1$, and $K+2$. Let us define random variables X_K and Y_K in round K as follows:

X_K= Number of nodes that receive the proposed block from the leader in round K

Y_K= Number of positive votes that leader of round $K+1$ receives from validators

B. Consensus time

In order to be committed to the distributed ledger as a valid block, the leader of round $K+3$ must accumulate at least QC positive votes from the validators for that block. The probability mass function of receiving i positive votes by L_{K+3} can be calculated as:

$$\mathbb{P}\{Y_{K+2}=i\}=\sum_{i'=i}^{N}\mathbb{P}\{Y_{K+2}=i|X_{K+2}=i'\}\mathbb{P}\{X_{K+2}=i'\} \tag{1}$$

where $\mathbb{P}\{X_{K+2}=i'\}$ is the probability that i' validators receive QC-of-QC in round $K+2$ from L_{K+2} and can be calculated as follows:

$$\mathbb{P}\{X_{K+2}=i'\}= \frac{N-f-1}{N-1}\sum_{j=i'}^{N}\mathbb{P}\{X_{K+2}=i'|Y_{K+1}=j\}\mathbb{P}\{Y_{K+1}=j\} \tag{2}$$

Note that in order to reach a consensus, all three leaders L_K, L_{K+1}, and L_{K+2} must be selected from the non-faulty nodes (also known as honest nodes). As already mentioned, the leader of each round is determined by the proposer of the latest committed block. In the future, we will see that one block will eventually be committed to the ledger at the end of each round. We assume a validator never determines itself as the leader of the next round. Thus, the factor $\frac{N-f-1}{N-1}$ in the equation (2) is the probability of selecting a honest node as L_{K+2} while L_{K+1} was already selected from honest nodes ($L_{K+2}\neq L_{K+1}$).

Similarly, in round $K+1$ we can write:

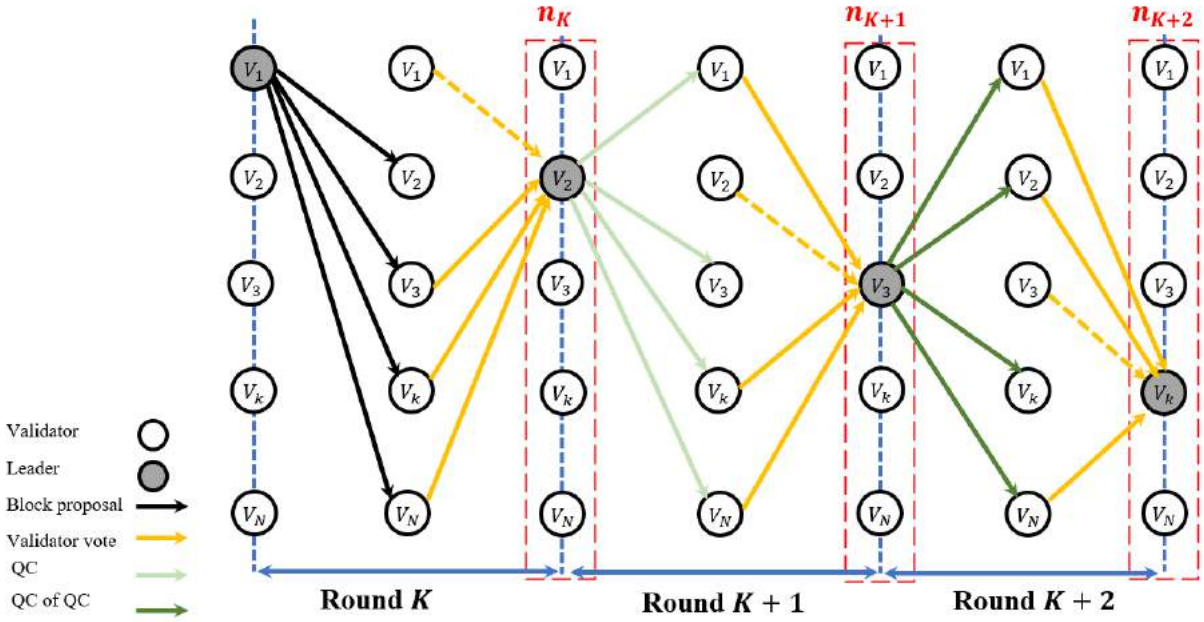


Fig. 2: Overview of the DiemBFT protocol: a proposed block gets committed to the ledger it three consecutive rounds

$$\mathbb{P}\{Y_{K+1}=j\} = \sum_{j'=j}^{N} \mathbb{P}\{Y_{K+1}=j|X_{K+1}=j'\}\mathbb{P}\{X_{K+1}=j'\} \quad (3)$$

as well,

$$\mathbb{P}\{X_{K+1}=j'\} = \frac{N-f-1}{N-1}\sum_{k=j'}^{N} \mathbb{P}\{X_{K+1}=j'|Y_K=k\}\mathbb{P}\{Y_K=k\} \quad (4)$$

in the same way,

$$\mathbb{P}\{Y_K=k\} = \sum_{k'=k}^{N} \mathbb{P}\{Y_K=k|X_K=k'\}\mathbb{P}\{X_K=k'\} \quad (5)$$

$$\mathbb{P}\{X_K=k'\} = \frac{N-f-1}{N-1}\binom{N}{k'}p^{k'}(1-p)^{N-k'} \quad (6)$$

In Equation (1), $\mathbb{P}\{Y_{K+2}=i|X_{K+2}=i'\}$ can be re-written as follows:

$$\mathbb{P}\{Y_{K+2}=i|X_{K+2}=i'\} = \binom{i'}{i}p^{i}(1-p)^{i'-i} \quad (7)$$

In the same way in Equation (3):

$$\mathbb{P}\{Y_{K+1}=j|X_{K+1}=j'\} = \binom{j'}{j}p^{j}(1-p)^{j'-j} \quad (8)$$

and in Equation (5):

$$\mathbb{P}\{Y_K=k|X_K=k'\} = \binom{k'}{k}p^{k}(1-p)^{k'-k} \quad (9)$$

So far, we are able to calculate the probability of reaching a consensus successfully (denoted as p_s) when $i = QC$. Nevertheless, consensus happens when L_{K+3} receives QC or more positive votes from the validators. Therefore,

$$P_s = \sum_{i=QC}^{N} \mathbb{P}\{Y_{K+2}=i\} \quad (10)$$

For estimating the transaction throughput, we need to calculate the mean time of reaching a consensus as follows:

$$\mathbb{E}[T_c] = \mathbb{E}[T_s] + (1-p_s)\tau_o \quad (11)$$

where $\mathbb{E}[T_c]$ and $\mathbb{E}[T_s]$ are the expected values of time for reaching a consensus (consist of all successful and unsuccessful attempts) and the expected value of the time for a successful attempt to reach a consensus, respectively. As well, τ_o is the value of the timer set by the clients as a time-out value. We assume all of the clients always set the same amount for the timer. p_s can be calculated using Algorithm 1.

To estimate $\mathbb{E}[T_s]$, we need to use a delay model for modeling the P2P delay between validators in the network. Since we assume that validators are connected together through asynchronous links, we use an exponential distribution as the delay model.

Assume n_K, n_{K+1}, and n_{K+2} are the number of validators that receive the block proposal from the leader of the round during rounds K, $K+1$, and $K+2$ respectively (see the Figure 2). The process of sending the proposal from the

leader to validators in each round can be considered as an asynchronous process since leaders broadcast the proposal to validators at almost the same time. Clearly,

$$n_k = \frac{N-f-1}{N-1} p(N-1) \tag{12}$$

the next leader needs to receive at least QC valid and positive votes. Otherwise, the consensus attempt fails. Therefore,

$$n_{k+1} = \frac{N-f-1}{N-1} p(N-1) \sum_{i=QC}^{N} \binom{n_K}{i} p^i (1-p)^{n_k - i} \tag{13}$$

Accordingly,

$$n_{k+2} = \frac{N-f-1}{N-1} p(N-1) \sum_{i=QC}^{N} \binom{n_{K+1}}{i} p^i (1-p)^{n_{k+1} - i} \tag{14}$$

Assume M nodes are sending messages to the same destination with a exponentially distributed delay. The expected time to receive i messages at the destination can be calculated as follows [28]:

$$\mathbb{E}[T_M] = \frac{1}{\mu} \sum_{j=0}^{i-1} \frac{1}{M-j} \tag{15}$$

where μ is the parameter of the exponential distribution (e.g. μ^{-1}= mean P2P delay between the nodes).

Hence,

$$\mathbb{E}[T_{n_k}] = \frac{1}{\mu} \sum_{i=0}^{QC-1} \frac{1}{n_k - i} \tag{16}$$

where $\mathbb{E}[T_{n_k}]$ is the expected time takes for L_{k+1} to receive QC valid and positive votes from the n_k validators at the end of round K. In the same way we can write:

$$\mathbb{E}[T_{n_{k+1}}] = \frac{1}{\mu} \sum_{i=0}^{QC-1} \frac{1}{n_{k+1} - i} \tag{17}$$

and,

$$\mathbb{E}[T_{n_{k+2}}] = \frac{1}{\mu} \sum_{i=0}^{QC-1} \frac{1}{n_{k+2} - i} \tag{18}$$

we are now able to calculate $\mathbb{E}[T_s]$ as follows:

$$\mathbb{E}[T_s] = \frac{3}{\mu} + \mathbb{E}[T_p] + \sum_{i=0}^{QC-1} \frac{1}{n_k - i} + \frac{1}{n_{k+1} - i} + \frac{1}{n_{k+2} - i} \tag{19}$$

where $\mathbb{E}[T_p]$ is the mean processing time. Now we can calculate $\mathbb{E}[T_c]$ from Equation (11).

C. Transaction throughput

Any leader that receives a QC or QC-of-QC from the previous leader can piggyback its own proposal [2]. Therefore, the throughput of the system γ can be calculated as follows:

$$\gamma = \frac{3}{\mathbb{E}[T_c]} \tag{20}$$

V. Model validation and analysis

In this section, we validate our theoretical model by comparing it with simulation results and explain the configuration we set up for simulation. We then discuss the results and the impact on the expected performance of the Diem network, as well as potential trade-offs.

Implementation: We used $OMNeT++$ [31] as a discrete event-based simulator of the network. As well, we used $Matlab$ for the calculation and analysis of the theoretical model. We used NED language for describing the network topology and assigning values to networking parameters (e.g. p2p latency, network size, and etc). We implemented Diem-BFT mechanism in each module using $C++$. As well, we changed random number seeds using an ini file.

Settings: According to [1], the current goal of Diem is to support around 100 validators but is expected to grow to a final size of 500-1000 validators over time. In this paper, we carry out the experiments for a variety of network sizes but limit our experiments to around 100 validators. In each experiment, we considered a network with a size of N as $N = 3f^* + 1$ where f^* is the maximum tolerable number of faulty validators. In other words, we carried out the experiments for $N = 9k + 4$ where $k = \{\mathbb{Z} \in [0, 11]\}$ and $f^* = 3k + 1$.

For all experiments, we assumed each block consists of 1000 transactions and processing of each transaction in validators takes the same amount of $1ms$. Moreover, we assumed that for a certain block in each round the leader has the same processing power and consequently the same processing delay as validators. In OMNeT++, a processing delay can be simulated using a self message scheduled for a certain event. We assumed validators spend the same time validating block proposal, QC, and Qc-of-QC. These parameters can be adjusted in the simulation code for further analysis in future work.

We set a direct P2P link between any two arbitrary validators. Hence, the network is a complete graph. We set the P2P link delay with an exponential distribution with a mean amount of $1/\mu = 1ms$. Assuming an exponential distribution for link delay is common and adequate for many use cases [32]. In OMNeT++, one can create a channel between two arbitrary modules in the Ned file with an exponential distribution and assign the mean value.

In order to simulate parameter p, we implemented all the links as ideal links without any packet loss, drop, or corruption. Instead, in the receiver module, packets get deleted upon arrival with a probability of $1 - p$ or go to the next level with a probability of p.

Algorithm 1: Calculating the probability of reaching consensus in Diem

Input : The values of N, f, QC p, μ, N_{tx}, D_{tx} and τ_o

Output: Probability of reaching a consensus

$\mathcal{P}_{Y_K}$=$[0]_{1\times(N-QC+1)}$, $\mathcal{P}_{Y_{k+1}}$=$[0]_{1\times(N-QC+1)}$,
$\mathcal{P}_{Y_{k+2}}$=$[0]_{1\times(N-QC+1)}$
$\mathcal{P}_{X_K}$=$[0]_{1\times(N-QC+1)}$, $\mathcal{P}_{X_{K+1}}$=$[0]_{1\times(N-QC+1)}$,
$\mathcal{P}_{X_{K+2}}$=$[0]_{1\times(N-QC+1)}$
for $i = 1$ *to* $N - QC + 1$ **do**
    $\mathcal{P}_{X_K}[i] \leftarrow \frac{N-f-1}{N-1}\binom{N}{i}p^i(1-p)^{N-i}$ // This loop calculates Equation (6)
end
for $i = 1$ *to* $N - QC + 1$ **do**
        // This loop calculates Equation (5)
    $A \leftarrow$i+QC-1
    for $j = i$ *to* $N - QC + 1$ **do**
        $B \leftarrow$j+QC-1
        $\mathcal{P}_{Y_K}[i] \leftarrow \sum_{j=i}^{N-QC+1}\binom{B}{A}p^A(1-p)^{B-A}\mathcal{P}_{X_K}[j]$
    end
end
for $i = 1$ *to* $N - QC + 1$ **do**
        // This loop calculates Equation (4)
    $C \leftarrow$i+QC-1
    for $j = i$ *to* $N - QC + 1$ **do**
        $D \leftarrow$j+QC-1
        $\mathcal{P}_{X_{K+1}}[i] \leftarrow \frac{N-f-1}{N-1}\sum_{J=i}^{N-QC+1}\binom{D}{C}p^C(1-p)^{D-C}\mathcal{P}_{Y_K}[j]$
    end
end
for $i = 1$ *to* $N - QC + 1$ **do**
        // This loop calculates Equation (3)
    $E \leftarrow$i+QC-1
    for $j = i$ *to* $N - QC + 1$ **do**
        $F \leftarrow$j+QC-1
        $\mathcal{P}_{Y_{K+1}}[i] \leftarrow \sum_{j=i}^{N-QC+1}\binom{F}{E}p^E(1-p)^{F-E}\mathcal{P}_{X_{K+1}}[j]$
    end
end
for $i = 1$ *to* $N - QC + 1$ **do**
        // This loop calculates Equation (2)
    $G \leftarrow$i+QC-1
    for $j = i$ *to* $N - QC + 1$ **do**
        $H \leftarrow$j+QC-1
        $\mathcal{P}_{X_{K+2}}[i] \leftarrow \frac{N-f-1}{N-1}\sum_{J=i}^{N-QC+1}\binom{H}{G}p^G(1-p)^{H-G}\mathcal{P}_{Y_{K+1}}[j]$
    end
end
for $i = 1$ *to* $N - QC + 1$ **do**
    $M \leftarrow$i+QC-1
    for $j = i$ *to* $N - QC + 1$ **do**
        $N \leftarrow$j+QC-1
        $\mathcal{P}_{Y_{K+2}}[i] \leftarrow \sum_{j=i}^{N-QC+1}\binom{N}{M}p^M(1-p)^{N-M}\mathcal{P}_{X_{K+2}}[j]$
    end
end
$P_s \leftarrow \sum_{i=QC}^{N}\mathcal{P}\{Y_{K+2} = i\}$
return P_s

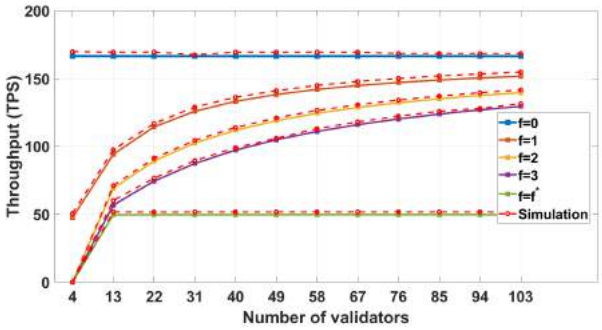


Fig. 3: Throughput: Theoretical vs. simulation results

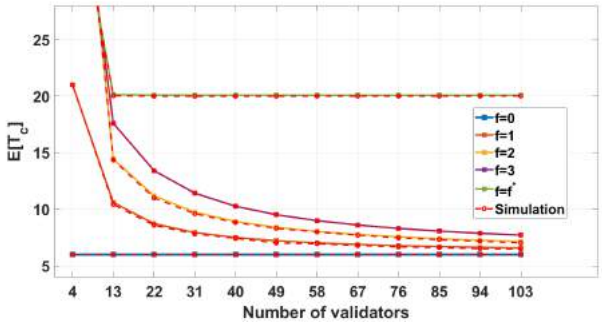
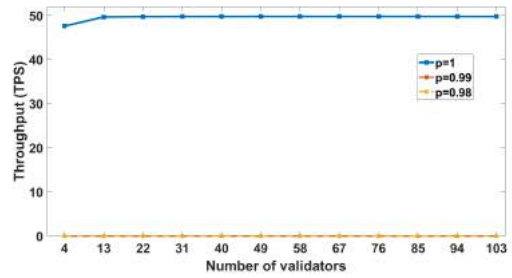


Fig. 4: Consensus time: Theoretical vs. simulation results

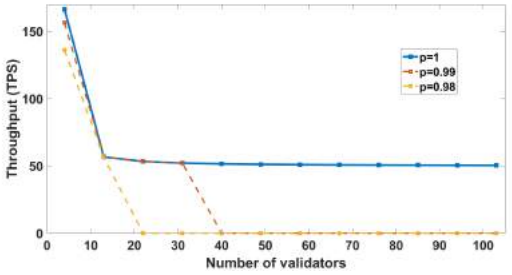
Finally, we assumed that Byzantine or faulty nodes can either send invalid responses or do not send any response to the block proposal, QC, or QC-of-QC. In the simulation code, faulty nodes always send an invalid response that gets deleted in the destination node. In addition, the validators that their valid votes get deleted due to packet loss are also treated as faulty nodes.

Results and discussion: In Figure 3 and Figure 4, we depict the results of our experiments for transaction throughput and consensus time versus the number of validators for different numbers of Byzantine validators in the Diem blockchain with the aforementioned settings. First, as we can see, the simulation results closely follow the theoretical amounts. Second, as it is expected, when increasing the number of faulty validators, throughput goes down and consensus time goes up. Another interesting conclusion that can be derived is that while there is at least one faulty node and $f < f^*$, increasing the number of validator nodes can improve the performance of the systems. However, when the number of Byzantine validators is equal to the maximum number of tolerable Byzantine validators (i.e. $f = f^*$), there is no benefit in adding more honest validators. It is to be noted that when $f = f^*$ for a network with a certain number of validators when we add one or more validators, f will be no longer equal to f^* as long as all of the new validators are honest.

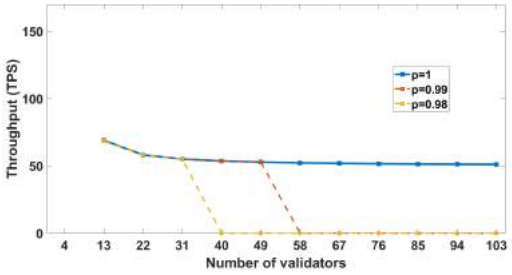
Our theoretical model can be used for performance analysis of Diem blockchain against a variety of configuration parameters. We studied the impact of communication quality on the



(a) $f = f^*$



(b) $f = f^* - 1$



(c) $f = f^* - 2$

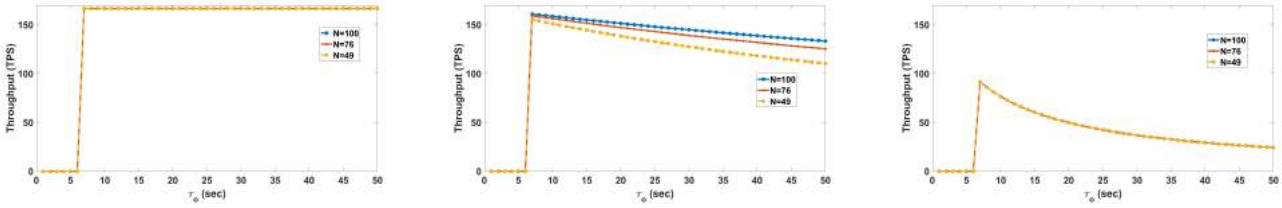
Fig. 5: Throughput versus p

performance of the Diem. We calculated system throughput for different amounts of p when the network is ideal and when there is packet loss in the network. In an ideal case or when $p = 1$, the model assumes that all the packets reach their destination within a certain time interval before the destination validator times out. Results can be seen in Figure 5. We first conducted the simulation for a network in which $f = f^*$. Results of this situation are shown in Figure 5a. one can observe that the network has a normal throughput when $p = 1$. But when the p is slightly decreased to 0.99 or 0.98, the system throughput tends to zero. This happens when the number of Byzantine validators still does not violate the maximum amount of f^*. This fact shows that the condition of $f \leq f^*$ might not be enough to guarantee a non-zero throughput for the system. Instead, the liveness of the system is highly dependent on the limited number of re-transmissions and consequently the limited amount of delay for delivering the messages to the validator nodes.

In another set of experiments, we reduced the number of faulty nodes to $f = f^* - 1$. In this configuration, the quorum size is still $2f^* + 1$. For a network with $N = 4$, when the network is ideal (i.e. $p = 1$) and there is no packet loss, we observe a high throughput compared to the throughput for the rest of the network sizes. That is because when $N = 4$, then $f^* - 1 = 0$ and hence there is no faulty node in the network. For $N = 13$, the throughput drastically goes down due to the presence of one faulty validator in the system. When $p = 0.99$, we observe the same pattern when we increase the network size until $N = 40$. At this point, the throughput goes down and tends to zero again. It happens because one of the leaders is not able to gather enough valid and positive votes for the proposal. A similar pattern happens when $p = 0.98$ but the throughput goes down faster when $N = 22$.

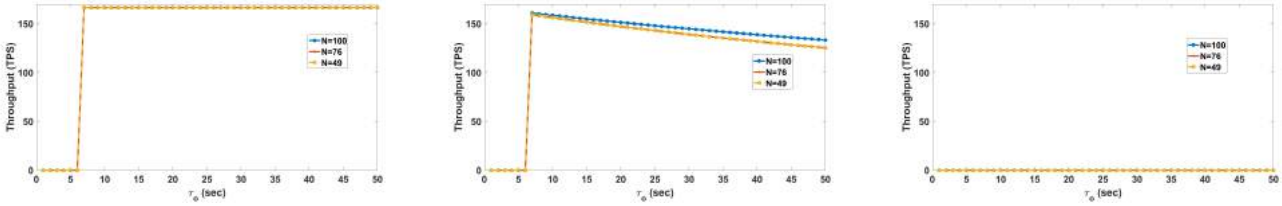
In another set of experiments, we are interested to study the impact of different amounts of τ_o on the performance of the system. We first conducted the experiments for a lossless network in which we assume $p = 1$. The results can be seen in Figure 6. We conducted the experiments for three conditions: when there is no faulty node in the network when there is only one faulty node in the network, and when there are f^* faulty nodes in the network. We repeated all experiments for the different network sizes in which $N = 100, N = 76$, and $N = 49$. Results for a lossless network with no Byzantine node are depicted in Figure 6a. We did not observe any meaningful difference in the transaction throughput for networks with different sizes. However, for $1 \leq \tau_o \leq 6$. The reason is that for this setting, $\mathbb{E}[T_s]$ is a bit more than 6 seconds (i.e $\mathbb{E}[T_s] = 6.01sec$). Hence when all the clients set τ_o to an amount between 1 to 6 seconds, they will always abort their proposed transaction and there will be no processed transaction. When τ_o is set to an amount of more than 6 seconds, the throughput reaches the maximum amount since there is no faulty node in the network. When we put a Byzantine validator in the system, we can see the effect of a higher τ_o as depicted in Figure 6b. We observe that throughput is still zero when $1 \leq \tau_o \leq 6$. The justification is the same as when $f = 0$. Nevertheless, throughput goes down for higher amounts of τ_o. As well, we observe that for the same amount of τ_o, bigger networks exhibit more throughput. That is because the ratio of honest nodes is higher for bigger networks. Hence, we can conclude that when there is a faulty node in the network, throughput can be compensated by adding extra validators to the system. In another set of experiments, we put f^* Byzantine nodes in the network. Results can be observed in Figure 6c. Once again we see a zero throughput for $1 \leq \tau_o \leq 6$. After that, throughput decreases when tau_o is increased. Since $p = 1$, all the networks follow the same pattern.

In the next set of experiments, we assumed that the network is lossy (i.e. $p \neq 1$). Results for $p = 0.99$ and $p = 0.98$ are shown in the Figure 7 and Figure 8 respectively. When there is no Byzantine node in the network, for both cases we see almost the same pattern as the lossless network. But in the presence of a faulty node, results are a bit different (see the Figures 7b and 8b). In both cases, we observe a lower throughput for a smaller network. The justification is the same as before for the lossless network. As expected, for the same



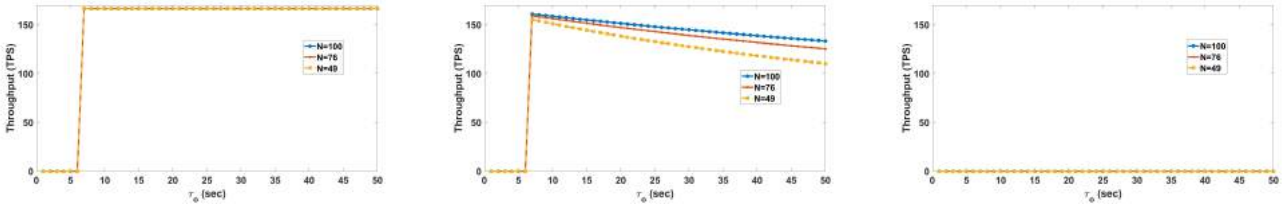
(a) $f = 0$ (b) $f = 1$ (c) $f = f^*$

Fig. 6: System throughput in different conditions when $p = 1$



(a) $f = 0$ (b) $f = 1$ (c) $f = f^*$

Fig. 7: System throughput in different conditions when $p = 0.99$



(a) $f = 0$ (b) $f = 1$ (c) $f = f^*$

Fig. 8: System throughput in different conditions when $p = 0.98$

amount of τ_o, we observe less throughput when $p = 0.98$ compared to when $p = 0.99$. Finally, we put f^* Byzantine nodes in the network as shown in the Figures 7c and 8c. The throughput is zero for all amounts of τ_o. Although $\mathbb{E}[T_s] = 6.01sec$, and for amounts of $7 \leq \tau_o$, we expect to have a non-zero throughput, nevertheless, validators fail to reach a consensus since the leader is not able to gather $2f + 1$ votes due to packet loss in the network or validators time-outs. This is an interesting and important situation since we observe that the condition of $f \leq \frac{N-1}{3}$ is not enough for the liveness of the system.

VI. Conclusion

In this paper, we proposed a theoretical performance model for the PBFT consensus algorithm of the Diem blockchain network and validated it with a set of simulations carried out using OMNeT++. As well, we presented a performance analysis and studied the system throughput for a variety of network sizes with a different number of faulty nodes. We observed that the throughput is highly dependent on the quality of service when there are faulty nodes in the network.

Setting a too low amount of time-out in clients can lead them to frequently abort their queries while setting a too high amount of time-out may slow down the network and consequently reduce the throughput. Moreover, setting an inappropriate time-out amount in validators can slow down the network in another way. Our future work is to leverage this model to dynamically determine the optimal timeout value for the clients and validators in order to maximize the throughput.

Acknowledgement

We would like to acknowledge *(name withdrawn) (affiliation withdrawn)* for their advice during the entire project presented in this article.[1]

[1]Withdrawn for the purpose of the double-blind review.

REFERENCES

[1] Z. Amsden, R. Arora, S. Bano, M. Baudet, S. Blackshear, A. Bothra, G. Cabrera, C. Catalini, K. Chalkias, E. Cheng *et al.*, "The libra blockchain," *URl: https://developers.libra.org/docs/assets/papers/the-libra-blockchain.pdf*, accessed: 20-6-2020.

[2] "State machine replication in the libra blockchain," *https://diem-developers-components.netlify.app/papers/diem-consensus-state-machine-replication-in-the-diem-blockchain/2020-05-26.pdf*, accessed: 20-6-2020.

[3] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus in the lens of blockchain," *arXiv preprint arXiv:1803.05069*, 2018.

[4] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[5] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 1988.

[6] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[7] M. Nasreen, A. Ganesh, and C. Sunitha, "A study on byzantine fault tolerance methods in distributed networks," *Procedia Computer Science*, vol. 87, pp. 50–54, 2016.

[8] M. Correia, G. S. Veronese, N. F. Neves, and P. Verissimo, "Byzantine consensus in asynchronous message-passing systems: a survey," *International Journal of Critical Computer-Based Systems*, vol. 2, no. 2, pp. 141–161, 2011.

[9] Y. Yang, "Linbft: Linear-communication byzantine fault tolerance for public blockchains," *arXiv preprint arXiv:1807.01829*, 2018.

[10] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Sok: Consensus in the age of blockchains," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 183–198.

[11] S. Agrawal and K. Daudjee, "A performance comparison of algorithms for byzantine agreement in distributed systems," in *2016 12th European Dependable Computing Conference (EDCC)*. IEEE, 2016, pp. 249–260.

[12] A. Momose and L. Ren, "Optimal communication complexity of byzantine consensus under honest majority," *arXiv e-prints*, pp. arXiv–2007, 2020.

[13] T. Distler, "Byzantine fault-tolerant state-machine replication from a systems perspective," *ACM Computing Surveys (CSUR)*, vol. 54, no. 1, pp. 1–38, 2021.

[14] C. Berger and H. P. Reiser, "Scaling byzantine consensus: A broad analysis," in *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, 2018, pp. 13–18.

[23] "Blinded for review," 2020.

[15] T. Loruenser, B. Rainer, and F. Wohner, "Towards a performance model for byzantine fault tolerant (storage) services," *arXiv preprint arXiv:2101.04489*, 2021.

[16] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *Open Problems in Network Security*, J. Camenisch and D. Kesdoğan, Eds. Cham: Springer International Publishing, 2016, pp. 112–125.

[17] J. Zhang, J. Gao, Z. Wu, W. Yan, Q. Wo, Q. Li, and Z. Chen, "Performance analysis of the libra blockchain: An experimental study," in *2019 2nd International Conference on Hot Information-Centric Networking (HotICN)*. IEEE, 2019, pp. 77–83.

[18] W. Li and M. He, "Comparative analysis of bitcoin, ethereum, and libra," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2020, pp. 545–550.

[19] X. Fu, H. Wang, and P. Shi, "A survey of blockchain consensus algorithms: mechanism, design and applications," *Science China Information Sciences*, vol. 64, no. 2, pp. 1–15, 2021.

[20] M. Dabbagh, K.-K. R. Choo, A. Beheshti, M. Tahir, and N. S. Safa, "A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities," *Computers & Security*, vol. 100, p. 102078, 2021.

[21] P. Sheng, G. Wang, K. Nayak, S. Kannan, and P. Viswanath, "Bft protocol forensics," *arXiv preprint arXiv:2010.06785*, 2020.

[22] J. Niu, F. Gai, M. M. Jalalzai, and C. Feng, "On the performance of pipelined hotstuff," *arXiv preprint arXiv:2107.04947*, 2021.

[24] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[25] "The libp2p project," *https://libp2p.io*, accessed: 20-6-2020.

[26] "The multiaddr project," *https://multiformats.io/multiaddr*, accessed: 20-6-2020.

[27] "The yamux project," *https://github.com/hashicorp/yamux*, accessed: 20-6-2020.

[28] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[29] H. Fukś, A. T. Lawniczak, and S. Volkov, "Packet delay in models of data networks," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 11, no. 3, pp. 233–250, 2001.

[30] A. M. Sukhov, M. Astrakhantseva, A. Pervitsky, S. Boldyrev, and A. Bukatov, "Generating a function for network delay," *Journal of High Speed Networks*, vol. 22, no. 4, pp. 321–333, 2016.

[31] "Omnet++ simulator," *available at http://www.omnetpp.org*.

[32] M. Hassan and R. Jain, *High performance TCP/IP networking*. Prentice Hall Upper Saddle River, NJ, 2003, vol. 29.

IBFT extension

yahya.shahsavari.1

December 2021

1 Background and IBFT system model

Although IBFT inherits the main features from the well-known PBFT, however, it slightly differs from it. The main difference is that there is no specific entity as a client in IBFT protocol. Instead, All of the validators can be seen as clients and the block proposals are proposed by validators. The mentioned proposers are being selected at the beginning of each round in a round-robin fashion.

Similar to PBFT, consensus is achieved in a set of phases known as: $pre-prepare$, $prepare$, and $commit$. At the beginning of pre-prepare phase, the proposer validator generates a block proposal from the transactions already gathered from the transaction pool and appends a `pre-prepare` message to the block and broadcasts it to rest of the validators. Upon receiving this message, validators enter pre-prepared state. Afterward, validators start the prepare phase and broadcast a prepare message. This phase is required in order to ensure that all of the validators are working at the same round and block sequence number. Once any validator gathers $2f + 1$ prepare messages, it enters the prepared state and informs the rest of the validators that it has accepted this block and is going to add it to the chain via broadcasting a `commit` message to all of the validators. Finally, validators will wait to receive at least $2f + 1$ `commit` message. Upon gathering $2f + 1$ `commit` messages, the validator will enter committed state and will add the proposed block to the ledger. It is to be noted that for the sake of preventing faulty nodes from generating invalid blocks and hence starting a separate chain, each validator appends $2f + 1$ received commit signatures to `extraData` field in the header before inserting it into the chain. Therefore forks do not happen in IBFT consensus protocol.

To find the probability of reaching a consensus in Quorum network we act as follows:

Consider a network of N validators and f faulty nodes. We denote any arbitrary validator by V_n. suppose this network is acting in round K. In each round, a successful consensus happens in three phases as already mentioned. Consensus happens if an arbitrary honest validator gathers at least $2f+1$ `commit` messages. Let random variable X_n denote the number of `commit` messages gathered by any arbitrary validator in the last phase. The probability of gathering at least $2f + 1$ `commit` messages by V_n can be computed as follows:

$$\mathbb{P}\{X_n \geq 2f+1\} = \sum_{n=2f+1}^{N} \binom{j}{n} p^n (1-p)^{j-n} \tag{1}$$

where j denotes the number of validators that already gathered at least $2f+1$ `prepare` messages at the end of last phase (i.e prepare phase). Now let us define random variable X as the number of validators that successfully gathered at least $2f+1$ `commit` messages at the end of commit phase. As well consider the random variable Y, as the number of nodes that successfully gathered at least $2f+1$ `prepare` messages at the end of prepare phase. Also, let random variable Z denote the number of nodes that received `pre-prepare` messages at the end of pre-prepare phase. We can now calculate the probability i honest validators gather at least $2f+1$ `commit` messages at the end of commit phase:

$$\mathbb{P}\{X = i\} = \sum_{j=i}^{N} \mathbb{P}\{X = i | Y = j\} \mathbb{P}\{Y = j\} \tag{2}$$

Where $\mathbb{P}\{Y = j\}$ is the probability that j honest validators already gathered `prepare` messages in previous phase and can be calculated as follows:

$$\mathbb{P}\{Y = j\} = \sum_{k=j}^{N} \mathbb{P}\{Y = j | Z = k\} \mathbb{P}\{Z = k\} \tag{3}$$

Where $\mathbb{P}\{Z = k\}$ is the probability that k honest validators already gathered `pre-prepare` messages at the end of pre-prepare phase and can be calculated as follows:

$$\mathbb{P}\{Z = k\} = \binom{N}{k} p^k (1-p)^{N-k} \tag{4}$$

It is now helpful to calculate $\mathbb{P}\{X = i | Y = j\}$:

$$\mathbb{P}\{X = i | Y = j\} = \binom{j}{i} \alpha^i (1-\alpha)^{j-i} \tag{5}$$

where $\alpha = \mathbb{P}\{X_n \geq 2f+1\}$ and was computed in Equation (1).
in the same way:

$$\mathbb{P}\{Y = j | Z = k\} = \binom{k}{j} \beta^j (1-\beta)^{k-j} \tag{6}$$

where

$$\beta = \mathbb{P}\{Y_n \geq 2f+1\} = \sum_{n=2f+1}^{N} \binom{k}{n} p^n (1-p)^{k-n} \tag{7}$$

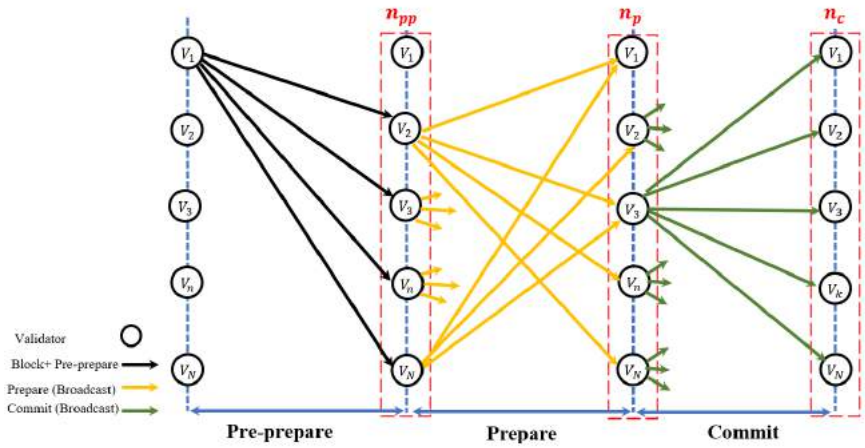


Figure 1: Overview of the IBFT protocol: a proposed block gets committed to the ledger it three consecutive phases

and Y_n is The probability of gathering at least $2f + 1$ `prepare` messages by V_n.

We assume consensus is reached and the block is inserted in the blockchain when at least one validator gathers $2f + 1$ `commit` messages. Therefore, the probability of reaching a consensus can be computed as follows:

$$P_s = \sum_{i=1}^{N} \mathbb{P}\{X = i\} \tag{8}$$

now let us to calculate the system throughput

$$\mathbb{E}[T_c] = \mathbb{E}[T_s] + (1 - p_s)\tau_o \tag{9}$$

Assume n_{pp}, n_p, and n_c respectively are the number of nodes at the end of pre-prepare, prepare and commit phases that gather at least $2f+1$ `pre-prepare`, `prepare`, and `commit` messages respectively. These parameters can be easily calculated as follows

$$n_{pp} = Np \tag{10}$$

$$n_p = n_{pp} \sum_{n=n_{pp}}^{2f+1} \binom{n_{pp}}{n} p^n (1-p)^{n_{pp}-n} \tag{11}$$

and in the same way:

$$n_c = n_p \sum_{n=n_p}^{2f+1} \binom{n_p}{n} p^n (1-p)^{n_p-n} \tag{12}$$

Rest of the calculation in order to obtain $\mathbb{E}[T_s]$ and consequently the system throughput is exactly the same as what we did in Diem BFT.