论文中，红色荧光表示系统模型的假设；蓝色荧光表示定义；黄色荧光表示文章的创新点；绿色荧光表示文章中某些重要定理。红色下划线表示文章中写的比较好的英文；绿色下划线表示文章中仿真结果和结论。

Adapting PBFT for Use With Blockchain-Enabled IoT Systems

Jelena Mišić, Vojislav B. Mišić, Xiaolin Chang, and Haytham Qushtom

***Abstract*—This work proposes Practical Byzantine Fault Tolerance (PBFT) ordering service needed for block formation in permissioned blockchain environments. Contrary to current PBFT implementations that only provide a single point of entry to the ordering service, we allow each ordering node to act as an entry point that proposes and conducts the consensus process of including new record in the distributed ledger. To ensure atomicity of record insertion in distributed ledger, we have developed a bandwidth reservation protocol that uses a modification of CSMA/CA protocol to regulate access to the broadcast medium formed by the P2P network of TCP connections between orderers. We have modeled record insertion service time in a cluster where ordering nodes have random position within Cartesian coordinate system. We have also modeled total request access time to the ledger which includes waiting time in the orderer's queue and record insertion time. These models are used to evaluate system performance under variable request rate ordering service, variable number of nodes and variable physical cluster dimensions. Our results show the interaction between decreased request waiting time in orderer's queue and increased contention among orderers when the number of orderers increases for the given total request arrival rate. This interaction is also investigated for two different physical cluster sizes which affect record insertion time. The interplay of request rate, number of orderers and physical cluster size determines system capacity expressed in total request rate. Our model can be used to make the trade-off between the required system capacity, number of orderers, and physical cluster dimensions under constraints on Byzantine fault rate.**

***Index Terms*—Blockchain, internet of things (IoT), PBFT, performance evaluation.**

I. Introduction

RECENTLY a number of Internet-of-Things (IoT) systems based on blockchain technology has been proposed. Their purpose is long term distributed and immutable storage of *ordered* measurement and observation reports. This data is considered as IoT transactions. Important applications of this approach include logging of reports in vehicular networks, surveillance systems and industrial process monitoring systems.

Due to the different nature of data, IoT transactions can be validated outside of the replication system as they are mostly independent of each other, unlike most cryptocurrencies where the balance of individual accounts must be checked before a transaction is accepted. As the result, the IoT ordering system should focus on establishing the total order of transactions, rather than on their validation which is performed prior to request submission. Since transaction validation is performed outside of the ordering system, nodes that do the ordering of IoT transactions should have permission to do so.

One of the crucial components of such systems is the consensus protocol. Due to peculiarities of IoT systems and the reliance on permissioned architecture, the Nakamoto consensus used in Bitcoin [22] is not the best solution. Instead, most proposals [3], [24], [25] tend to rely on Practical Byzantine Fault Tolerance originally proposed in [8]. However, PBFT is focused on state machine replication which poses a somewhat different set of requirements with respect to a typical IoT application. In particular, PBFT assumes that a single ordering node acts as the proposer of transactions (i.e., the protocol leader) and, consequently, all client requests must be sent to that node. This ensures that requests are serialized and processed without concurrency problems. Failure of the leader node leads to the election of a new leader (often referred to as view change), as in PBFT [8]; alternatively, a node may nominate itself to become a leader [23]. In both cases, the leader role lasts for indefinite period of time, and election of the new leader is done as another instance of consensus. On the contrary, IoT systems that cover large geographical areas would benefit from the ability of different nodes to act as proposers for different client requests. However, this possibility leads to concurrency problems, in addition to the more common problems caused by hardware or software failures, or malicious (i.e., Byzantine) behavior of some nodes.

To address those limitations, we propose to augment the PBFT protocol with an additional step inspired by bandwidth reservation in well known CSMA/CA systems such as IEEE 802.11 [12]. The fully connected TCP overlay network that interconnects ordering servers emulates the broadcast medium in which bandwidth reservation can be implemented using Request to Send (RTS) and Clear to Send (CTS) messages. Those messages provide the necessary authorization for the proposer to continue with the PRE-PREPARE, PREPARE, and COMMIT phases of the PBFT protocol. Failure to collect the required number of CTS responses within a given time interval will lead to a backoff, i.e., waiting for a time interval of random duration, before making another proposal attempt.

Manuscript received July 15, 2020; revised October 12, 2020 and November 16, 2020; accepted December 27, 2020. Date of publication December 30, 2020; date of current version February 12, 2021. The review of this paper was coordinated by Prof. Yi Qian *(Corresponding author: Vojislav B. Mišić)*

Jelena Mišić, Vojislav B. Mišić, and Haytham Qushtom are with the Ryerson University, Toronto, ON M4J 3V4, Canada (e-mail: jmisic@ryerson.ca; vmisic@scs.ryerson.ca; hqushtom@ryerson.ca).

Xiaolin Chang is with the Beijing Jiaotong University, Beijing 100044, China (e-mail: xlchang@bjtu.edu.cn).

Digital Object Identifier 10.1109/TVT.2020.3048291

Contributions and novelty of this work are the following:

- Work presented in this paper differs from earlier works as it does not assume system saturation and round robin leader change. Instead, it allows on-demand leader assignment based on request traffic.
- The proposed protocol allows any of the ordering nodes to submit a proposal for inclusion in the system blockchain ledger where it will be made available to client applications. The leader is self-elected through successful bandwidth reservation protocol in which all nodes vote. Bandwidth reservation process is derived from RTS/CTS handshake in CSMA/CA process.
- IoT proxy servers send the data collected from IoT gateways to the nearest ordering server in terms of geographic proximity. The proposed change retains the standard fault resilience features of PBFT, namely, that the system with $n_{ord} = 3f + 1$ ordering nodes is capable of reaching consensus in the presence of up to f faulty or malicious nodes. It does include one additional step in the PBFT protocol but it brings on the ability to accommodate many proposers in a wide geographical area and thus makes possible the development of distributed IoT systems built by multiple operators that may have Byzantine behavior.
- System model contains three components. First is the derivation of maximum message propagation time among the ordering nodes which is necessary to estimate timeout values in the protocol. Second is the Discrete Time Markov Chain model for bandwidth reservation process. Third is the queuing model of the memory pool of each ordering node. These coupled models allow important insights into the interaction among parameters such as the number of orderers, request arrival rate to the system, and physical cluster size.

The paper is organized as follows. Related work is discussed in Section II. Section III describes system architecture. In Section IV, we present the medium reservation protocol which precedes the PBFT consensus protocol with multiple entry points explained in Section V. Analytical model of the ordering service is given in Section VI, the probability distribution function of one-way delay between nodes is derived in Section VII, and model of CSMA/CA contention among orderers is given in Section VIII. Performance evaluation is presented in Section IX. Finally, Section X concludes the paper.

II. Related Work

Research area of Byzantine Fault Tolerance (BFT) algorithms was initiated by the finding that distributed system with nodes prone to failure cannot have simultaneous properties of liveliness and safety [10]. The first design of a protocol that tolerates Byzantine failures is Practical Byzantine Fault tolerance (PBFT) [8] using majority vote of more than 2/3 correct nodes. In this approach, $n \geq 3f + 1$ nodes act as a single correct server that can withstand a maximum f nodes that exhibit Byzantine behavior. During protocol execution one node is denoted as a leader or primary, while others are known as replicas; all nodes are connected to the clients of the service. Protocol consists of three phases:

- *pre-prepare*, where the leader sends the client's proposal (or batch of transactions) to all replicas;
- *prepare*, where replicas check the proposal and share their agreement/disagreement with each other and the leader;
- *commit*, in which replicas that receive at least $2f$ *prepare* agreements from other nodes send commit message to all other nodes.

Once a node collects $2f + 1$ commit messages, it executes the requested update proposal and informs the client accordingly. Problems arise if leader is Byzantine, in which case the leader should somehow be changed. This work has spurred a lot of off-springs in several directions.

A. Addition of Cryptographic Mechanisms

A number of cryptographic mechanisms were added to boost protocol security by authenticating clients to replicas [14], and replicas to each other. To compress replies to clients, threshold signatures are used, often based on Boneh-Lynn-Shacham [6] scheme. In this scheme there is a single public key, while each node has an individual private key. Nodes encrypt their individual messages with their respective private keys. Portions are collected into a joint message which can be decrypted with the public key, but only if it has between $2f + 1$ and $3f + 1$ components. Alternative form of threshold encryption [5] is used in [17] where replicas send their encrypted portions to the leader which verifies and combines them into a single record.

B. Addition of Algorithmic Optimizations to PBFT

Many researchers have noted relatively large communication complexity and execution time of single PBFT round, and proposed various optimizations to speed the algorithm and improve system throughput. One of the best known approaches is Zyzzyva [14] which uses speculation that all replicas are correct so that they can process the request without communicating with each other and send reply to the client. When a client receives between $2f + 1$ and $3f$ matching replies, it will create a commit certificate from $2f + 1$ replicas' replies and send it to replicas in order to achieve final consensus. This approach assumes that clients are not Byzantine and that they are connected to all validating nodes. However, in reality clients can be Byzantine and they can collude with replicas. SBFT [11] extends Zyzzyva by introducing communication and execution nodes known as collectors. It also combines protocol steps with threshold cryptography. Unfortunately collectors present single point of failure so they have to become redundant, and the algorithm has to be extended to handle their failure.

C. Addition of Mechanisms to Improve Robustness of PBFT Algorithm

Many researchers have noted vulnerability of PBFT under Byzantine attacks as well as the effect of fragile optimizations, and improved its reliability by adding more protocol phases and messages.

A major point of improvement is reliability of primary (leader) and exchange of leader in case of failure. To detect primary's misbehavior, each client is connected to all replicas so that replicas are aware of all requests. Replicas are also equipped with timers and time-out values so that a misbehaving primary can be detected and black-listed. In Prime [2], each replica maintains a vector of requests and PBFT is augmented by two pre-phases. In pre-phases, vectors are exchanged so that they can be committed using standard PBFT. In Aardvark [9] all clients are authenticated.

Furthermore, replicas monitor and slowly increase minimal throughput. If the primary does not match the throughput increase, replicas will start a round robin protocol to change the primary. In Spinning [27], primary is changed in round robin fashion after it has completed a batch of requests. In RBFT [4], reliability is achieved by resource redundancy, as each node runs $f+1$ PBFT protocol instances including one primary and f replicas. One instance is the master instance which proposed ordering that is accepted by the replicas. Other instances are backups used for throughput measurement. If measured throughput of the master instance drops below that of the best backup instance, then view should be changed.

All previous algorithms (except [17]) are designed to work in LANs with small propagation delays and under single administration authority. All systems are assumed to work in saturation regime where there is always a request to be processed. Unfortunately from the aspect of queuing theory this means that request queues at replicas and the primary are very long [13], [26]. Also performance is measured only as the throughput of requests while access delay which equals the sum of waiting delay and request processing time is not considered.

D. Deployment of BFT Algorithms in Blockchains

Recently BFT algorithms have begun to be deployed in permissioned blockchains where records are validated by a committee of nodes whose identities are known. Such systems are characterized by large geographical coverage and have to be deployed over WAN spanning many autonomous systems. Number of nodes is also large which makes the Byzantine aspect of each validator become even more important than in the LAN environment.

Honey Badger uses threshold cryptography and randomization in order to combat Byzantine primary and scheduling. SBFT [11] combines collector based optimizations, threshold signatures and speculative fast path from [14] to achieve high throughput but leaving some points of failure. In Tendermint [7], choice of primary node (proposer) follows round robin scheduling. Primary node guides the validation process of a block in rounds. Validators vote in two rounds known as pre-vote and pre-commit before the block is committed. To prevent interference with the voting process by less than one-third of nodes, this algorithm uses locks that ensure safety, i.e., atomic execution of pre-vote and pre-commit processes once a sufficient number of pre-votes is collected. In HotStuff [1], the PBFT-like steps are preceded with a step for view (i.e., leader) selection. Namely replicas submit view proposals justified by the number of votes collected in the previous view and compressed in Quorum Certificate (QC) using threshold cryptography. This step introduces additional 1.5 RTT delay for the scheme. Each step is executed atomically. However, leader change protocol was not presented.

Blockchain technology is also used for other tasks in industrial IoT networks. For example, a replicated, immutable ledger for the purpose of supporting a trading platform with a cloud mining mechanism was proposed in [31] using the tools of game theory. Also, a platform for enforcing the non-repudiation mechanism in cases of dispute between interested parties using smart contracts was proposed in [30]. Recently, it was shown that multiple entry PBFT system for IoT blockchain over wide area network outperforms single entry system [21].

III. System Architecture

System is assumed to consist of an arbitrary number of IoT domains which can be distributed over large geographic areas, as shown in Fig. 1(a). Each IoT domain, schematically shown in Fig. 1(b), contains at least one IoT proxy which collects data from IoT devices, either directly or through one or more IoT gateways, and organizes them in records (transactions).

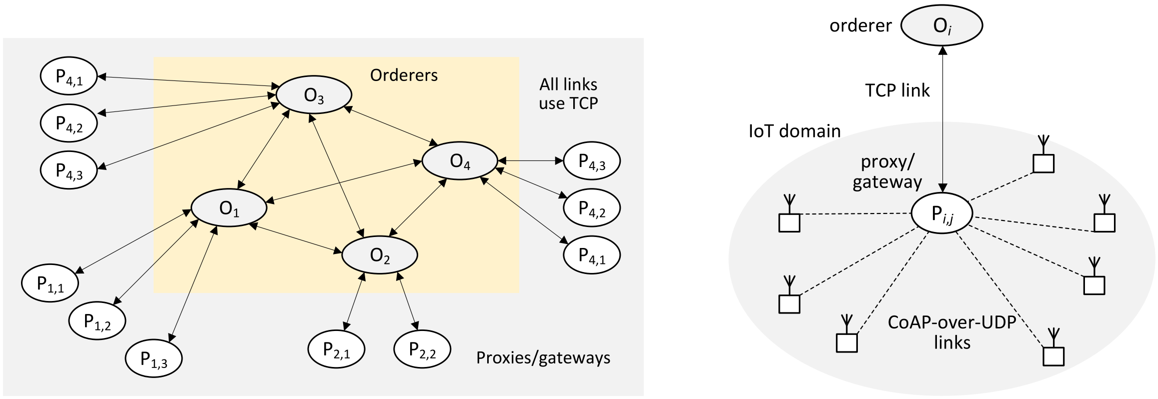
Proxies and gateways are devices with much more computational and communication power compared to ordinary IoT nodes.

Gateway can operate in forward mode where it forwards user queries to the IoT domain, receives the reply and sends the reply back with possible data caching. It can also operate in reverse mode where it proactively retrieves the measurements from the IoT nodes and replies on their behalf to the clients [18]. IoT nodes are connected using CoAP/UDP protocol suite running over IPv6, and MAC layer such as IEEE 802.15.4 or IEEE 802.11ah. Communication between the gateway and the nodes can be achieved using CoAP GET/POST, observation or multicast features.

In a financial system like Blockchain or Ethereum, transactions have to be validated against implicit or explicit user account balances. Even the next generation of blockchain systems such as Hyperledger Fabric assumes that transactions are simulated and validated in dedicated validator nodes. However, data records/transactions in IoT are of different nature and validation must proceed in a different way. It appears more reasonable to have IoT data pre-processed using some kind of machine learning algorithms in order to exclude outliers before being added to record/transaction. This kind of validation can be performed by IoT proxies, but this issue is beyond the scope of the present paper.

The system presented in Fig. 1(a) contains a number of ordering servers (orderers) which are tasked with achieving the total order of records/transactions in each orderer's memory pool.

These servers are not ordinary IoT devices, instead they have considerable computational power. Ordering servers may be geographically distributed in a manner similar, but not necessarily identical, to the IoT domains and they are interconnected using TCP connections in a fully connected graph, with each proxy connected to a number of ordering servers using TCP connections. Primary ordering server for proxy should be the one



(a) (b)

Fig. 1. Blockchain architecture across IoT domains. (a) Network topology for orderers and proxies. (b) IoT domain architecture.

with smallest round trip time (RTT); the other ordering servers connected with the proxy in question have larger RTTs and they are used as backup in case the primary ordering server is faulty, or perhaps for load balancing among the ordering servers.

Each ordering server keeps a memory pool containing records/transactions not yet included in blocks. The server memory pool is updated by record/transaction insertion service that will be explained in the next Section. Given that there are $n_{ord} = 3f + 1$ ordering servers, insertion service is capable of achieving identical total order of transactions/records in memory pools belonging to all ordering servers in the presence of up to f faulty orderers. Faults may occur due to some kind of attack such as the DoS attack on communication bandwidth towards the server or other server resources. Such faults are known as *Byzantine* faults [8], [16].

Periodically one orderer will group a number of records/transactions into a block, add the header and other metadata, and submit it to other orderers for approval. Once the orderers achieve consensus, the block is appended to each orderer's blockchain ledger. All approved ordering servers keep full copies of blockchain. Proxies and client applications can then request those blocks in order to access the data therein.

IV. Broadcast Bandwidth Reservation Protocol

Crucial part of the proposed addition to PBFT is the protocol for bandwidth reservation that aims to ensure record insertion without contention. Any orderer o_i, $i = 1 \ldots n_{ord}$ can initiate the bandwidth reservation protocol; if successful, it can then begin and complete insertion of the proposed record into the memory pool of all orderers. As noted above, this protocol is based on the broadcast communication medium established among ordering nodes, each of which must continuously listen to all of its peers at the corresponding TCP sockets.

We have adopted the principle of Carrier Sense Multiple Access with collision avoidance (CSMA/CA) protocol for collision-free access for ordering peers. This protocol was initially developed for Distributed Coordination Function (DCF) in IEEE 802.11 standard for wireless local area networks (LANs). In order to avoid packet collisions on wireless (broadcast) medium, short reservation packet known as Request to Send (RTS) is sent first. It contains a Network Allocation Vector (NAV) stating the duration of the forthcoming transmission. Receiving station replies to reservation with a Clear to Send (CTS) packet, after which transmission is conducted. In this way collisions were limited to short RTS packets instead of long data packets. We have adapted this idea to P2P broadcast medium such that requesting orderer applies for leader position (bandwidth reservation) by transmitting a RTS message. However, in this case all orderers have to reply with their CTS messages which will be received over different TCP sockets. This poses a challenge in the protocol since propagation delays among orderers are different. Therefore there is a need to compute distribution of maximum overall delay which will be needed to compute timeouts in the protocol.

A. Adapted CSMA/CA for Bandwidth Reservation

The ordering node that is about to initiate reservation has to check whether the previous insertion cycle is completed or whether the timeout has occurred. This is done by checking if at least $2f + 1$ COMMIT messages for the most recent request have been received. If this is not the case, the node needs to wait until previous record insertion (i.e., the corresponding COMMIT phase) is completed.

Upon confirming that the previous insertion cycle is completed, orderer i sends a Request to Send (RTS) message simultaneously over all TCP connections to its ordering peers. RTS message contains Network Allocation Vector (NAV) stating the maximum time to reach consensus which will be explained in following section. Peers respond with Clear to Send (CTS) messages containing source ID and request ID; however, they do not respond immediately, as will be explained below. CTS message sent from peer j means that the peer promises not to engage in bandwidth reservation on behalf of other clients until the current request has been processed by the orderers. In this manner, the RTS/CTS sequence enables *atomic* multicast of subsequent PRE-PREPARE, PREPARE and COMMIT messages. If the transaction batch is not successfully committed in the time stated in the NAV, the leader node has to contend for the system

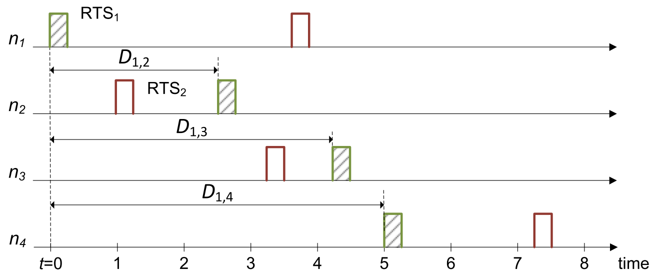


Fig. 2. Timing of RTS requests and the vulnerable period.

medium again. A node can be black-listed after a predefined number of failures to commit a batch.

This procedure is prone to *virtual collisions* of RTS requests, i.e., when two peers issue separate RTS requests within the one-way propagation time period between each other. To ensure protocol reliability in the presence of virtual collisions, it is necessary to calculate the following parameters: (a) the duration of the *vulnerable period* which starts with the receipt of first RTS and lasts as long as it is possible to receive another RTS due to finite signal propagation time among the nodes; and (b) the timeout period after which the sender of the RTS message knows that a CTS message cannot arrive.

Therefore, the sending orderer needs to listen to its TCP sockets for incoming messages during the timeout period. If it hears other RTS message(s) while it is waiting for CTS it should interpret this as virtual collision, abort its RTS attempt and enter backoff phase before attempting to send RTS again.

The receiving orderers operate as follows. Once the orderer receives the first RTS, it starts a countdown of the vulnerable period. If no other RTS is received during the vulnerable period, the orderer sends the CTS back to the initiating sender. However, if another RTS message is received during the vulnerable period (or perhaps more than one), the receiving orderer declares a collision and does not send a CTS back to any of the initiating orderers.

B. Duration of Vulnerable Period

To determine the maximum length of the vulnerable period, we assume that peers i and l send independent RTS requests at moments $t_0 = 0$ and $t_0 + T_\epsilon$, respectively. The value of T_ϵ ranges between zero and one-way propagation time between nodes i and l, i.e., $0 \leq T_\epsilon < D_{i,l}$ as can be seen in Fig. 2 for $n_{ord} = 4, i = 1, l = 2$ and sample destination node $w = 4$. (Upper limit of $T_\epsilon < D_{i,l}$ is necessary since node l would withhold its RTS upon receiving the RTS from node i.)

Peer node w will receive RTS requests at moments $D_{i,w}$ and $T_\epsilon + D_{l,w}$, respectively. Depending on the destination node, these two requests may arrive in different order to different orderers and thus lead to conflicting CTS messages if virtual collision is not detected.

Now, the vulnerable period is the maximum time period between the arrival of conflicting RTS messages. Therefore, the node that received a RTS should refrain from sending the corresponding CTS during the vulnerable period. Considering Fig. 2 where $i = 1$, $l = 2$, and $w = 4$, let us denote the time of arrival of first RTS to node w with $\min(D_{i,w}, T_\epsilon + D_{l,w})$. For simplicity, let $x \in (i, l)$ denote the index of the RTS request that arrived first. Then, the vulnerable period for node w is

$$\Delta_w = \max_{j \neq x,w} (D_{i,j} + D_{j,w} - \min(D_{i,w}, T_\epsilon + D_{l,w})) \qquad (1)$$

where index j corresponds to other orderers in the network – nodes n_2 and n_3, in this example. The last expression has two boundary values depending on T_ϵ:

$$\lim_{T_\epsilon \to D_{i,l}} \Delta_w = \max_{j \neq x,w} (D_{i,j} + D_{j,w} - \min(D_{i,w}, D_{i,l} + D_{l,w}))$$

$$\lim_{T_\epsilon \to 0} \Delta_w = \max_{j \neq x,w} (D_{i,j} + D_{j,w} - \min(D_{i,w}, D_{l,w}))$$

and the latter gives the maximum value of Δ_w.

Vulnerable period can be estimated individually at each node or for the whole network, in which case it is

$$\Delta_{\max} = 2\max(D) - \min(D) \qquad (2)$$

where $\max(D)$ and $\min(D)$ are the largest and smallest one-way delays in the network, respectively. If another RTS request, or several of them, arrive at node w within the vulnerable period from the arrival of the first RTS, node w will not return any CTS.

C. Estimation of Timeout Value for RTS-CTS Cycle

Upon receiving a RTS message from node i and in the absence of collisions, node w will return a CTS message indicating the request is now accepted by w. CTS is sent in multicast after the vulnerable period, i.e., after $D_{i,w} + \Delta_w$ from the moment when RTS was issued. Propagation of the CTS message back to the RTS issuer and also to each peer $k \in (1 \ldots n_{ord})$, $k \neq w$ will take $D_{w,k}$.

Therefore, without RTS collision, CTS will be returned from node w to the RTS issuer i within the time window of

$$\begin{aligned} T\_CTS_{i,w} &= 2D_{i,w} + \Delta_w \\ &= D_{i,w} + \max_{j \neq i,w} (D_{i,j} + D_{j,w}) \end{aligned} \qquad (3)$$

The timeout window in which node i can collect all CTS responses can be estimated as

$$\begin{aligned} TO_i &= \max_w (T\_CTS_{i,w}) \\ &= \max_w (D_{i,w} + \max_{j \neq i,w} (D_{i,j} + D_{j,w})) \end{aligned} \qquad (4)$$

The last expression can be evaluated individually per node, or maximized for the entire ordering network as $TO = 3\max(D)$, where $\max(D)$ corresponds to maximum one-way delay among the orderers. Since every node has to receive RTS and return CTS, this timeout value holds for the whole network, but it reduces network processing power.

One-way propagation delays can be measured using round trip time (via time stamps) and further estimated using exponential weighted moving average EWMA [15] or using method of kernels [20], [28]. These measurements should be done periodically – for example, for each request. As in the case of estimation of RTT in TCP, some number of standard deviations should be added to mean value when calculating the timeout value.

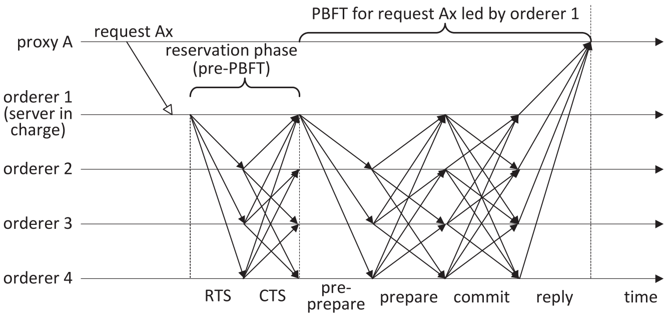


Fig. 3. Transaction ordering protocol.

No CTS will be returned to initiating orderers in case of a collision. After their respective timeout periods expire, all requesting orderers will enter binary backoff and repeat their requests when value of individual backoff counters reach zero. Unit backoff slot has to be also derived as small fraction of maximum vulnerable period.

V. Record/transaction Ordering Protocol

When the initiating orderer receives $2f$ CTS replies within the RTS-CTS timeout window, it can begin atomic execution of PBFT. Note that orderers which did not return CTS reply (for any reason) may attempt to transmit RTS for the new request during processing of current request. Since its/their ID is known, other orderers will not reply with CTS until the current request is processed.

Protocol which combines bandwidth reservation and PBFT protocol is presented in Fig. 3 and is described as follows:

1) Proxy which has collected a batch of l records/transactions submits this batch to the one of directly connected orderers, preferably the one with the shortest RTT. Another orderer will be chosen only if the first one did not reply within some time period. Without loss of generality, we can label this orderer with ID $1 \leq i \leq n$ and refer to it as the ordering server in charge for this batch.
2) Ordering server i adds this request to its request queue which is served using FIFO principle. When the request reaches the head of the queue, the orderer will assign tentative sequence numbers to proposed records/transactions according to the sequence numbers of transactions residing in its memory pool. It will also add the client's ID and timestamp, and sign the request using its private signing key.
3) The ordering server i checks if it has received $2f + 1$ COMMIT messages for the most recent request. If this is the case, the server initiates reservation for upcoming atomic broadcast as discussed above, otherwise it needs to wait until the previous record insertion (i.e., its COMMIT phase) is completed.
4) When the requesting orderer receives $2f$ CTS replies for bandwidth reservation (within timeout window) it can start atomic broadcast of the PRE-PREPARE message containing processed client request from the head of the orderer's queue. Note that orderers which did not return CTS reply (for any reason) might attempt a new bandwidth reservation of their own during the processing of current request. However, orderers that have accepted the reservation of node i will not accept any new reservations until the current request is processed.
5) PRE-PREPARE message should arrive within local timeout which is equal to maximum one-way delay. When an orderer receives the PRE-PREPARE message, it will check its memory pool to verify that the proposed record/transaction sequence numbers comply with existing sequence numbers. If this is the case, it will send PREPARE message to all other orderers, including the orderer i which initiated the current round. PREPARE message from orderer j informs its peers that the proposed record/transaction sequence numbers are in order with respect to its memory pool. Since the requesting orderer i does not send a PREPARE message, the maximum number of PREPARE messages that an orderer can receive is $3f$, and quorum is reached with only $2f$ received PREPARE messages.
6) Every orderer needs to receive $2f$ PREPARE messages from its peers agreeing with proposed sequence numbers, requesting orderer ID, and message hash sent in PRE-PREPARE message. To be able to proceed, $2f$ PREPARE messages should arrive within local timeout value after PRE-PREPARE was sent. After that, orderer j transmits a COMMIT message acknowledging proposed sequence numbers of records in its memory pool simultaneously over all its TCP connections. When at least $2f + 1$ COMMIT messages, including its own, have been collected, peer j inserts proposed records/transactions in its memory pool and replies to the proxy informing it that the proposal has been accepted. Receipt of $2f + 1$ COMMIT messages releases the bandwidth reserved with RTS/CTS message exchange and allows the orderer j to start bandwidth reservation for new proposals, if any.

A. Block Formation

When the number of free records/transactions in memory exceeds some threshold, or the time period after the creation of the previous block has been exceeded, one of the ordering peers should initiate new block creation. This can be the peer where the last record insertion created a sufficient number of transactions which can fit into block, or the peer with ID of $K \bmod n_{ord}$, where K denotes the number of blocks in the system and n_{ord} represents the number of orderers. In any case, the peer in charge begins bandwidth reservation using the RTS/CTS handshake among orderers. When $2f$ RTS messages have been received, the peer broadcasts PRE-PREPARE message containing sequence numbers of records/transactions in the block and block header to its peers. Peers then validate the block and send PREPARE messages, followed by COMMIT messages, in the usual way. After that, the block is linked at all ordering peers and is distributed towards proxy peers. Therefore, from the system aspect, block creation presents the same sequence of steps and loads as a single record insertion.

B. Implementation of CSMA/CA Request Access in Ordering Service

As explained above, each orderer achieves atomic broadcast by sending RTS after backoff over all of its TCP connections, and by subsequently listening on all of them for the CTS messages from peer orderers. If any other content is received during the vulnerable period, the orderer will declare a RTS collision and perform new backoff before the request re-transmission. During the backoff process, the orderer also listens to all TCP sockets and suspends backoff if PBFT content is being transmitted. Since the fully connected graph of TCP connections also represents a broadcast medium, receipt of $2f+1$ CTS messages means that ordering service has been reserved for target orderer so it can initiate the PBFT exchange. This reservation will be terminated by receipt of $2f+1$ COMMIT messages.

VI. Model of Multiple Entry PBFT

We denote number of orderers as $n_{ord} = 3f+1$ where f is the maximum number of faulty or Byzantine orderers. Nodes are placed in Cartesian coordinate system. For each ordering node i, horizontal and vertical coordinates in space are uniformly distributed in the range $0 \leq dx_i, dy_i \leq R$.

Probability distribution of one-way propagation delay is derived roughly from the distance between the nodes, assuming also that the number of routers between a pair of nodes is proportional to the distance between them. Consequently, we assume that one-way delays between two ordering nodes have horizontal $x \in \{0 \ldots S_c\}$ and vertical $y \in \{0 \ldots S_c\}$ components which contribute to the total delay using geometrical principles.

Probability density function and cumulative distribution functions (pdf and CDF) of one-way delay, respectively, derived in Section VII below, amount to

$$\begin{aligned} D(u) &= \frac{(6\,u^2\pi(S_c)^2 + 3\,u^4 - 16 S_c u^3)}{6(S_c)^4 s_0} \\ d(u) &= D'(u) \end{aligned} \tag{5}$$

where s_0 is constant.

To model components of PBFT, we need to model probability distribution of maximum one-way delay between issuer of the message and peers, and the timeout period after which the issuer of RTS should collect all CTS responses, as per (4).

A. Estimation of Maximum One-Way Delay

To estimate maximum one-way delay, let us consider a peer, say, o_1, and note that

$$\begin{aligned} P(\max(D_1) < u) = P(D_{1,2} < u)P(D_{1,3} < u)\cdots \\ P(D_{1,n_{ord}} < u) \end{aligned} \tag{6}$$

By definition $D_{1,i}(u) = P(D_{1,i} < u)$, so we can calculate the CDF of maximum one-way delay as

$$\max(D_1(u)) = \prod_{i=2}^{n_{ord}} D_{1,i}(u) = D(u)^{n_{ord}-1} \tag{7}$$

Since all one-way delays in the network follow the same probability distribution, (7) also represents maximum one-way delay for any orderer in the network, i.e., $\mathcal{D}(u) = \max(D_1(u))$.

After that we can compute pdf, mean value and Laplace Stieltjes Transform (LST) of maximum one-way delay as

$$\begin{aligned} \mathscr{d}(u) &= \frac{d(\mathcal{D}(u))}{du} \\ \overline{\mathcal{D}} &= \int_{u=0}^{S_c} u\mathscr{d}(u)du \\ \mathcal{D}^*(s) &= \int_{u=0}^{\infty} e^{-su}\mathscr{d}(u)du \end{aligned} \tag{8}$$

Maximum round trip time (RTT) can be obtained as two maximum one-way delays, i.e., $\mathcal{R}^*(s) = (\mathcal{D}^*(s))^2$.

B. Estimation of Maximum Period Between RTS and CTS

Probability distribution from (4) represents the maximum time in which CTS should be returned. First we need to find maximum of the sum of two one-way delays, which has the pdf and CDF, respectively, of

$$\begin{aligned} td(x) &= \int_{u=0}^{x} d(u)d(x-u)du \\ tD(u) &= \int_{x=0}^{u} td(x)dx \end{aligned} \tag{9}$$

Maximum of this sum can be obtained by noting that, for example, the following holds for the node o_1 with respect to node $w = o_4$:

$$P(\max(tD_1) < u) = P(tD_{1,2,4} < u) \cdot P(tD_{1,3,4} < u) \tag{10}$$

By definition $tD(u) = P(tD \leq u)$, so the CDF of maximum two hop delay is

$$\max(tD(u)) = (tD(u))^{n_{ord}-2} \tag{11}$$

If we denote the maximum two-way delay with $\mathcal{T}(u) = \max(tD(u))$, we can compute probability density function and LST of maximum two-way delay as

$$\begin{aligned} \mathscr{t}(u) &= \frac{d(\mathcal{T}(u))}{du} \\ \mathcal{T}^*(s) &= \int_{u=0}^{\infty} e^{-su}\mathscr{t}(u)du \end{aligned} \tag{12}$$

Further we need to sum maximum two hop delay with one hop delay and find the distribution of the maximum. Probability density function and CDF of this sum are given by

$$\begin{aligned} twd(y) &= \int_{u=0}^{y} d(u)td(y-u)du \\ twD(u) &= \int_{x=0}^{u} twd(y)dy \end{aligned} \tag{13}$$

Probability distribution of maximum delay for one orderer can be found in a similar manner as

$$\begin{aligned} \mathcal{W}(u) &= (twD(u))^{n_{ord}-1} \\ \omega(u) &= \frac{d(\mathcal{W}(u))}{du} \\ \mathcal{W}^*(s) &= \int_{u=0}^{\infty} e^{-su}\omega(u)du \end{aligned} \tag{14}$$

Unfortunately the last LST is difficult for manipulation and we need to approximate $\omega(u)$ with a Gamma distribution obtained using first two moments of original distribution:

$$\overline{\mathscr{W}} = \int_{u=0}^{\infty} u\omega(u)du$$
$$var(\mathscr{W}) = \int_{u=0}^{\infty} u^2\omega(u)du - \overline{\mathscr{W}}^2 \qquad (15)$$

Parameters for the Gamma distribution and pdf are calculated as

$$b_g = var(\mathscr{W})/\overline{\mathscr{W}}$$
$$c_g = \overline{\mathscr{W}}/b_g$$
$$f_g = \frac{1}{\Gamma(c_g)b_g^{c_g}} u^{c_g-1}e^{-u/b_g} \qquad (16)$$

We will denote this Gamma distribution approximation LST as $D_g^*(s) = (1 + sb_g)^{-c_g}$.

The timeout value for the issuer of RTS request can be calculated as

$$TO = \overline{\mathscr{W}} + k\sqrt{var(\mathscr{W})} \qquad (17)$$

where the mean value $\overline{\mathscr{W}}$ is augmented by a small number, say, $k \in (1\dots4)$, of standard deviations of $\mathscr{W}$ (note that this distribution has coefficient of variation smaller than 0.2).

In order to proceed with modeling of PBFT we assume that transmission time of the record is l_ds. Mean one-way delay from proxy to orderer is $\overline{d_c}$ and it has LST $T_c^*(s) = \frac{\mu_c}{\mu_c+s}$ where $\mu_c = 1/\overline{d_c}$.

LST for the time needed to receive CTS after RTS is $D_g^*(s)$, while LST of times needed for PRE-PREPARE, PREPARE and COMMIT are all equal to $\mathscr{D}^*(s))$.

Then, we can compute LST for the total record insertion time without collisions with other orderers within ordering service as

$$T_{ins}^*(s) = \mathscr{W}^*(s)(\mathscr{D}^*(s))^3 \qquad (18)$$

with mean value $\overline{T_{ins}} = -T_{ins}^{*'}(0)$.

Total insertion time taking into account submission time from the proxy and reply time to the proxy has LST of

$$T_{ins,tot}^*(s) = T_{ins}^*(s)e^{-l_d s}(T_c^*(s))^2 \qquad (19)$$

VII. Probability Distribution of One-Way Delay Between Two Nodes

One-way propagation delay probability distribution is derived from the distance between the nodes assuming also that number of routers (i.e., amount of queuing delay) between the nodes is proportional to the distance. As nodes are assumed to be placed in a Cartesian coordinate system, one-way propagation delay can be computed as $D(i,j) = (\frac{1}{v} + \kappa)\sqrt{(dx_i - dx_j)^2 + (dy_i - dy_j)^2}$, where speed of electromagnetic waves is $v \approx 2 * 10^8\ m/s$, and κ is proportionality coefficient showing impact of router queuing delays. Consequently, we assume that one-way delays between two ordering nodes have horizontal $x \in \{0\dots S_c\}$ and vertical $y \in \{0\dots S_c\}$ components which contribute to the total delay using geometrical principles. Coefficient of proportionality between the two is $S_c = (\frac{1}{v} + \kappa)R$. Fig. 4 shows an ordering network with $n_{ord} = 4$, where distances are represented by one-way propagation delays expressed in ms.

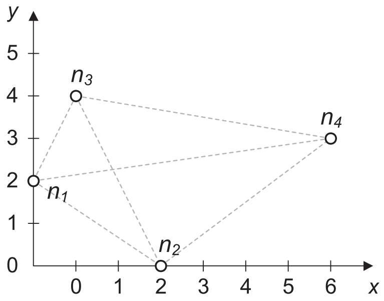


Fig. 4. Ordering network with $n_{ord} = 4$.

We now need to derive probability density function of one-way delay between the nodes given the node's Cartesian coordinates. This derivation has to be done in several steps starting from points n_1 and n_2 with coordinates (x_1, y_1) and (x_2, y_2), respectively, each uniformly distributed in the range $x_1, y_1, x_2, y_2 \in \{0\dots1\}$.

A. Distribution of $Z = |X_1 - X_2|$

Best way to derive the probability distribution function (PDF) $Z \leq z$ is to represent surface $Z = |x_1 - x_2|$. This surface consists of two equilateral triangles with total area of $S = \sqrt{3}/2$. We need to find the area of triangles below surface $Z = z$, which is $A = \frac{\sqrt{3}}{2}(1 - (1 - z)^2)$. Then PDF is equal to probability $P(Z \leq z) = A/S$. This results in PDF of

$$F_Z(z) = \left(1 - (1 - z)^2\right). \qquad (20)$$

Probability density function is then computed as $f_Z(z) = F_Z'(z) = 2(1 - z)$. If random variables are uniformly distributed in the range $0\dots S_c$ the corresponding PDF and pdf have the form:

$$F_{Z,S_c}(z) = \left(1 - (1 - z/S_c)^2\right)$$
$$f_{Z,S_c}(z) = F_{Z,S_c}'(z) = \frac{2}{S_c}\left(1 - \frac{z}{S_c}\right) \qquad (21)$$

B. Distribution of $Y = (X_1 - X_2)^2$

In this case we need to find PDF as probability $P(Z^2 \leq y) = F_Y(\sqrt{y}) = (1 - (1 - \sqrt{y})^2)$ when $y \in \{0\dots1\}$. Then pdf becomes $f_Y(y) = F'(y) = \frac{1-\sqrt{y}}{\sqrt{y}}$. In the range $y \in \{0\dots S_c\}$ corresponding functions become $F_{Y,S_c}(y) = (1 - (1 - \sqrt{y}/S_c)^2)$ and $f_{Y,S_c}(y) = \frac{1}{S_c\sqrt{y}}(1 - \frac{\sqrt{y}}{S_c})$.

C. Distribution of $W = 2Y$

Probability density function in this case can be found using convolution:

$$f_W(w) = \int_{y=0}^{w} f_Y(y)f_Y(w - y)dy = \pi + w - 4\sqrt{w} \qquad (22)$$

and this function is defined in the range $0 \leq w \leq w_0$ where w_0 is solution of equation $f_W(w) = 0$. This function also requires scaling coefficient equal to $s_0 = \int_{w=0}^{w_0} f_W(w)dw$ in order to satisfy law of total probability. Therefore final form of this pdf is $\hat{f}_W(w) = f_W(w)/s_0$. PDF is then equal to

$$\begin{aligned} F_W(w) &= \int_{x=0}^{w} \hat{f}_W(x)dx \\ &= (w\pi + 0.5\,w^2 - \frac{8}{3}w^{3/2})/s_0 \end{aligned} \tag{23}$$

If original random variables were distributed in the range $0 \ldots S_c$ convolution would be:

$$\begin{aligned} f_{W,S_c}(w) &= \int_{y=0}^{w} f_{Y,S_c}(y) f_{Y,S_c}(w-y)dy \\ &= (\pi(S_c)^2 + w - 4S_c\sqrt{w})/(S_c)^4 \end{aligned} \tag{24}$$

and this function is defined in the range $0 \leq w \leq w_{0,S_c}$ where w_{0,S_c} is solution of equation $f_{W,S_c}(w) = 0$. This function also requires scaling coefficient equal to $s_{0,S_c} = \int_{w=0}^{w_{0,S_c}} f_{W,S_c}(w)dw = s_0$ (i.e. independent of S_c) in order to satisfy law of total probability. Therefore final form of this pdf is:

$$\hat{f}_{W,S_c}(w) = f_{W,S_c}(w)/s_0 \tag{25}$$

PDF of this random variable is then equal to

$$\begin{aligned} F_{W,S_c}(w) &= \int_{x=0}^{w} \hat{f}_{W,S_c}(x)dx \\ &= \frac{(6\,w\pi(S_c)^2 + 3\,w^2 - 16S_c w^{3/2})}{6(S_c)^4 s_0} \end{aligned} \tag{26}$$

D. Distribution of $U = \sqrt{W}$

In this case we are looking into PDF as $P(\sqrt{W} \leq u) = P(W \leq u^2)$. If original variable were defined in the range $0 \ldots 1$ then this random variable is defined in the range $0 \ldots \sqrt{w_0}$. Then PDF and pdf have the form:

$$\begin{aligned} F_U(u) &= (u^2\pi + 0.5\,u^4 - \frac{8}{3}u^3)/s_0 \\ f_U(u) &= (2\pi u + 2\,u^3 - 8\,u^2)/s_0 \end{aligned} \tag{27}$$

When original random variables are distributed in the range $0 \ldots S_c$ PDF and pdf have following forms:

$$\begin{aligned} F_{U,S_c}(u) &= F_{W,S_c}(u^2) = \frac{(6\,u^2\pi(S_c^2) + 3\,u^4 - 16S_c u^3)}{6(S_c)^4 s_0} \\ f_{U,S_c}(u) &= F'_{U,S_c}(u) \end{aligned} \tag{28}$$

VIII. Modeling CSMA/CA for Record Insertion Service

We can now model CSMA/CA protocol for collision-free access for ordering peers. This is necessary in order to perform atomic bandwidth reservation for further PBFT steps. It is inspired by Distributed Coordinated Function (DCF) from IEEE 802.11. Carrier sensing is possible since each peer orderer has TCP connections towards all other orderers and each step of PBFT algorithm is implemented in simultaneous transmission over all sockets, i.e., in multicast. Both transmitting and non-transmitting peers have to always listen to all TCP sockets.

Collision avoidance is achieved using two basic mechanisms. First, reservation messages (RTS and CTS) are used to book the medium among orderers for request transmission. Therefore collisions can occur among RTS messages from different peers but subsequent request transmission will be collision free. Second, the peers use backoff procedure in case of RTS collisions or busy medium before request transmission.

Ordering peer that has a request to transmit first checks if the most recent record insertion phase is completed, i.e., if it has received $2f + 1$ COMMIT messages. Then it listens to all TCP sockets for one backoff slot to check if any other peer is transmitting its RTS. If there is no transmission, the peer can start the RTS transmission; otherwise, it has to wait until the current transmission has finished. In this case, the peer will wait for one backoff slot and then generate a random backoff time before transmitting its RTS. This backoff time is uniformly chosen in the range of the backoff window (0, $W - 1$). Backoff counter will be decremented after the backoff period if the transmission medium is free, or frozen until the medium becomes free again for one backoff time slot. Node will transmit RTS when its backoff counter reaches zero. In case of unsuccessful transmission of RTS, CTS will not be received after maximum RTT time period plus one backoff slot, and new backoff with a larger window size will be performed.

Basic assumption in discrete model of backoff process is that time is slotted and slot time (or backoff period) is denoted as δ. Since ordering peers can be placed over larger geographical distances propagation times have order of tens of milliseconds. For that reason backoff slot is much smaller than one-way propagation time between two peer orderers. In our model it is more than 10 times smaller than mean one-way delay but still sufficiently large so that transmission can be detected.

Similar to IEEE 802.11 DCF, the backoff window doubles after each unsuccessful CTS/RTS attempt. The window for i-th attempt can be calculated as $W_i = \min(2^{i-1}W_{\min}, W_{\max})$ where i denotes the index of the current request transmission attempt. Choice of $W_{\min}$ and $W_{\max}$ depends on the number of ordering peers; they can be smaller compared to standard values used in IEEE 802.11 DCF. Maximum number of RTS re-transmissions which lead to increase of contention window is m_r.

A. Analytical Model for Bandwidth Reservation in PBFT Following CSMA/CA Principles

Analytical model that follows is inspired by the work [19], [29], [32] but some important differences should be noted. In classical IEEE.802.11 DCF, the network is confined to several hundreds of meters and propagation delays are smaller than several μs. Backoff slot is larger than round trip time which is not the case in our environment where backoff slot size is just a small fraction of the propagation delay. Small slot size and small backoff window size result in small differentiation times among orderers, which is appropriate for small number of orderers, say, $n_{ord} \leq 10$. Also, in IEEE 802.11 DCF there are differentiated and short interframe spaces, DIFS and SIFS, which are used to prioritize transmission of CTS and ACK frames over new frames. In this case they are not needed since CTS will be returned after expiry of vulnerable period during which all competing RTS requests should have arrived, all other nodes should have heard RTS, and there is no hidden terminal problem.

We assume that all headers and request payload are transmitted with the same data rate R (expressed in bits). Corresponding number of bits per backoff slot is $\delta_B = R\delta$. Total header size including all nested protocol data units is H_{all} (expressed in bits), or is $H_{dr} = H_{all}/\delta_B$, expressed in backoff slots. Payload of RTS and CTS messages is denoted as rc bits. Size of RTS and CTS messages in slots is $rts = cts = (H_{all} + rc)/\delta_B$. Size of the data insertion request in bytes is l_{db}. When headers are included, size of request expressed in slots is $l_d = (8l_{db} + H_{all})/\delta_B$.

LST for the successful bandwidth reservation request transmission time is

$$Sr^*(s) = e^{-s(rts+cts+3)}T^*_{ins}(s) \tag{29}$$

where separation time slots are also taken into account. Mean value of (29) is $\overline{Sr} = -Sr^{*'}(0)$.

Previous expression can be simplified and presented as Probability generating function (PGF) for the successful request transmission time which is approximately equal to

$$Sr(z) = z^{rts+cts+\overline{T_{ins}}} \tag{30}$$

with mean value $\overline{Sr} = Sr'(1) = rts + cts + \overline{T_{ins}}$.

In case of collision of RTS packets, duration of unsuccessful activity on medium has PGF equal to

$$Cr(z) = z^{rts+cts+3+TO}. \tag{31}$$

Let us denote access probability of each orderer as τ_r. Probability of successful transmission of request for each orderer is denoted as γ_r, and collision probability as $1 - \gamma_r$. It can be expressed as:

$$\gamma_r = (1 - \tau_r)^{n_{ord}-1} \tag{32}$$

Next, we need to compute probabilities that medium is busy and idle, respectively, during backoff countdown of target node. Probability that medium will be active during the backoff countdown of one node has two parts. First part is the probability that the orderer detected that the medium is busy due to successful transmission of some other among $n_o - 1$ orderers and it has the value of

$$P_{bs} = (n_{ord} - 1)\tau_r(1 - \tau_r)^{n_{ord}-2} \tag{33}$$

The other component is the probability that the orderer detects busy medium due to a collision among a subset of $n_{ord} - 1$ other orderers; it has the value of

$$P_{bc} = 1 - (1 - \tau_r)^{n_{ord}-1} - P_{bs}. \tag{34}$$

Probability that backoff count from target orderer will be suspended due to access by other orderer(s) is obtained as

$$P_b = P_{bs} + P_{bc} \tag{35}$$

PGF for the duration of time between two successive backoff countdowns executed by the target orderer is represented with

$$\beta(z) = z\gamma_r + (P_{bc}Cr(z) + P_{bs}Sr(z))\,\beta(z) \tag{36}$$

and PGF for the duration of the backoff phase is

$$B_i(z) = \sum_{k=0}^{W_i-1} \frac{1}{W_i}\beta^k(z) = \frac{(\beta(z))^{W_i} - 1}{W_i(\beta(z) - 1)} \tag{37}$$

where $W_i = 2^i W_0$, for $i \leq m_r$ and $W_i = 2^{m_r} W_0$ for $i > m_r$.

For simplicity we assume that orderer will re-transmit RTS until CTS is received which is reasonable if the system is not heavily loaded. In that case we can write PGF for the request service time as

$$\begin{aligned} T_r(z) = &\sum_{i=1}^{m_r+1}\left(\prod_{j=0}^{i-1} B_j(z)\right)(1-\gamma_r)^{i-1}Cr(z)^{i-1}\gamma_r Sr(z) \\ &+ \left(\prod_{j=0}^{m_r} B_j(z)\right)\cdot \\ &\sum_{i=m_r+2}^{\infty} B_{m_r}(z)^{i-m_r}(1-\gamma_r)^{i-1}Cr(z)^{i-1}\gamma_r Sr(z) \end{aligned} \tag{38}$$

Mean value for request service time is then $\overline{T_r} = T_r'(1)$.

We assume that total external arrival rate of requests for whole ordering system and internal block formation requests is λ_s per second. (We note that more precise model would require modeling of internal requests as vacations for queues with external requests but it would increase model complexity at the expense of clarity of main conclusions.) If requests are uniformly distributed over orderers (and this should be the built-in the load balancing feature of system architecture), individual request arrival rate per orderer becomes $\lambda_{r,s} = \frac{\lambda_s}{n_{ord}}$ requests per second. In order to be deployed in the model, the arrival rate has to be scaled to the number of request arrivals per slot, i.e., to $\lambda_r = \lambda_{r,s}\delta$.

Offered load for the individual orderer then can be calculated as $\rho_r = \lambda_r \overline{T_r}$.

B. Computing Access Probability for Orderers

Access probability for orderer needs to be derived using a Markov chain which models the CSMA/CA process. However, since we consider non-saturated system, queuing model for the request queue at the orderer is also necessary to calculate the node idle probability. These two models have to be jointly solved.

Discrete time Markov chain (DTMC) for orderer's bandwidth reservation attempts is presented in Fig. 5; it contains transitions at end of slots. This DTMC has two dimensions; horizontal dimension corresponds to the value of backoff counter, while vertical dimension represents the backoff stage. Each state of DTMC is denoted with probability $\sigma_{i,j}$, where range of indices is $i = 0 \ldots m_r$ and $j = 0 \ldots W_i - 1$. The parameter π_0 represents the probability that orderer request queue is empty after the

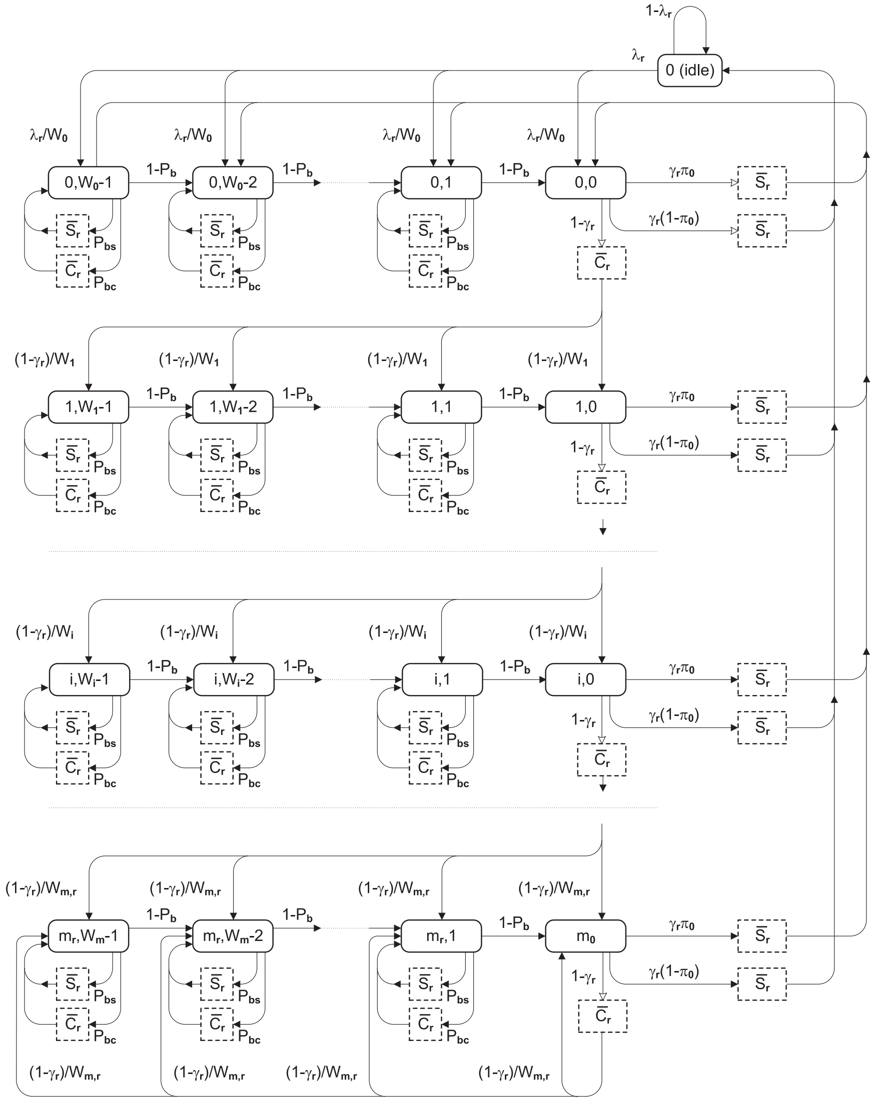


Fig. 5. Markov chain modeling orderer's attempts to achieve bandwidth reservation.

request departure; this parameter is part of the queuing model which will be presented later.

This DTMC can be solved by setting balance equations for each state and solving the system of equations. The key variable in this DTMC is access probability $\tau_r = \sum_{i=0}^{m_r} \sigma_{i,0}$ and we will use it together with success probability γ_r to express other state probabilities.

First, we need to set the balance equation for idle probability $P_i = P_i e^{-\lambda_r} + \tau_r \gamma_r \pi_0$. After approximating the number of request arrivals with first two terms of Taylor series expansion, we get $P_i = \frac{\tau_r \gamma_r \pi_0}{\lambda_r}$.

Then, we set balance equations for the first backoff stage as:

$$\sigma_{0,W_0-1} = \frac{P_i \lambda_r}{W_0} + \frac{\tau_r \gamma_r (1-\pi_0)}{W_0} + \sigma_{0,W_0-1} P_b \tag{39}$$

All other state probabilities are derived in the similar manner.

After summing all state probabilities for this DTMC and equating the sum with one as normalization condition we get equation which can be solved for orderer access probability (whilst keeping π_0 as a parameter):

$$\begin{aligned}\tau_r = \Bigg(& \frac{\gamma_r \pi_0}{\lambda_r} + \sum_{j=0}^{m_r} \gamma_r (1-\gamma_r)^j + \\ & \sum_{j=0}^{m_r} \frac{(W_j - 1)\gamma_r (1-\gamma_r)^j}{2(1-P_b)} (P_c \overline{Cr} + P_s \overline{Sr}) \\ & + \frac{(W_{m_r} - 1)(1-\gamma_r)^{(m_r+1)}}{2(1-P_b)} (P_c \overline{Cr} + P_s \overline{Sr}) \\ & + \gamma_r \overline{Sr} + (1-\gamma_r)\overline{Cr} \Bigg)^{-1}\end{aligned} \tag{40}$$

1) Queuing Analysis for Orderer: Orderer request buffer can be modeled as a M/G/1 system with FCFS discipline. Probability distribution of queue service time is equal to request service time, the PGF of which is presented in (38). Its LST $T_r^*(s) = T_r(e^{-s})$ is then used to calculate probability distribution of the number of request arrivals during request service time as

$$A_r(z) = T_r^*(\lambda_r - \lambda_r z) \tag{41}$$

We follow standard M/G/1 queue analysis steps [26] and obtain PGF for the number of requests in orderer queue after a request departure as

$$\Pi_r(z) = \frac{A_r(z)(1-\rho_r)(1-z)}{A_r(z) - z} = \sum_{i=0}^{\infty} \pi_{r,i} z^i \tag{42}$$

Coupling between DTMC and M/G/1 systems, eqns. (38) and (40), is achieved with parameters γ_r, τ_r, and π_0. This gives the last equation in the model as

$$\pi_{r,0} = \Pi_r(0) = 1 - \rho_r \tag{43}$$

By solving the system of equations (33), (34), (40) and (43) for given value of request arrival rate we can get numerical values for system descriptors.

Request response time is equal to the sum of request waiting time in the queue and request servicing time. We denote LST of request response time (also known as access time) as $S_r^*(s) = W_r^*(s)T_r^*(s)$ where $W_r^*(s)$ denotes LST of the waiting time of request in orderer's queue. These LSTs can be found by noting that the number of packets left after a departing packet is equal to the number of packets which arrive while this packet was in the system:

$$\Pi_r(z) = W_r^*(\lambda_r - \lambda_r z)T_r^*(\lambda_r - \lambda_r z) \tag{44}$$

which, after appropriate substitution, leads to the LST for request response time of

$$S_r^*(s) = \frac{s(1-\rho_r)T_r^*(s)}{s - \lambda_r + \lambda_r T_r^*(s)} \tag{45}$$

with mean value and variance of

$$\begin{aligned} \overline{S_r} &= \overline{T_r} + \frac{\lambda_r T_r^{(2)}}{2(1-\rho_r)} \\ var(S_r) &= T_r^{(2)} + \frac{\lambda_r T_r^{(3)}}{3(1-\rho_r)} + \frac{\lambda_r T_r^{(2)}\overline{S_r}}{(1-\rho_r)} - \overline{S_r}^2 \end{aligned} \tag{46}$$

where $T_r^{(2)}$ and $T_r^{(3)}$ are second and third non-central moments of ordering request service time.

IX. Performance Evaluation

The focus of this section is the impact of the number of orderers and geographical cluster coverage on the multiple entry PBFT system performance under increasing total system load.

To this end, we model the communications in the ordering system and evaluate request response time and system capacity. We have conducted numerical experiments using Maple 13 by Maplesoft, Inc., Waterloo, ON, while simulation results were obtained from a custom made simulator built with Anylogic by Anylogic, Inc., Oakbrook Terrace, IL.

We have obtained performance results for ordering systems with $n_{ord} = 4, 7, 10$ orderers, allowing 1, 2 and 3 faulty orderers, respectively. All orderers are fully interconnected with TCP connections having throughput of 2 Mbps each. Orderers have been randomly placed in Cartesian system with horizontal and vertical coordinates following same uniform distribution in the range $0 \ldots R$ (in meters). In this work we considered two cluster area sizes resulting in the corresponding single-dimensional component of one-way delay in the range $0 \ldots 5$ ms (small area cluster) and $0 \ldots 10$ ms (medium area cluster) as derived in Section VII. Those cluster coverages correspond to square areas with dimensions roughly of 300 × 300 and 900 × 900 km, respectively, including delays in routers; the estimations were obtained by running `traceroute` program [15] over Canadian provinces Ontario and Manitoba.

TABLE I
Mean Values of Maximum One-Way Delay and Mean Record Insertion Time. All Values in Milliseconds.

	$n_{ord} = 4$		$n_{ord} = 7$		$n_{ord} = 10$	
delay span	$\overline{\mathscr{D}}$	$\overline{T_{ins}}$	$\overline{\mathscr{D}}$	$\overline{T_{ins}}$	$\overline{\mathscr{D}}$	$\overline{T_{ins}}$
0 . . 5	3.82	22.9	4.18	25.1	4.37	26.23
0 . . 10	7.64	45.8	8.36	50.2	8.74	52.5

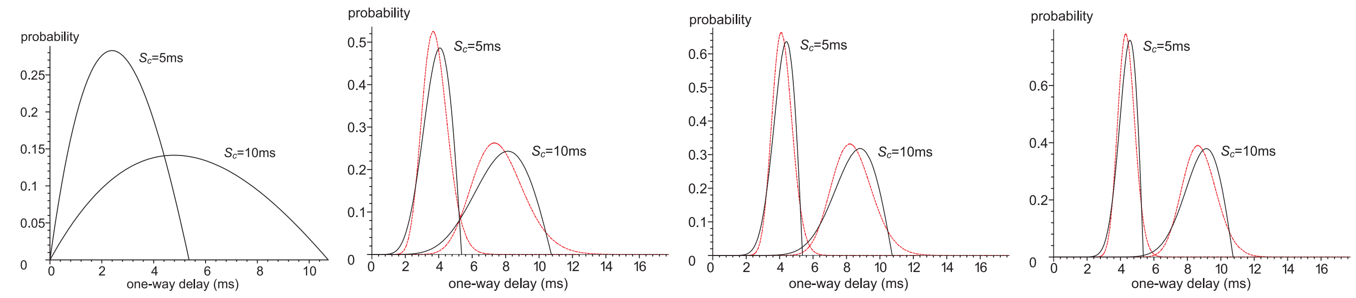
Backoff slot for contention resolution among orderers was set to $\delta = 1$ ms and all timing figures are represented as multiples of backoff slots, i.e., in milliseconds.

Minimal contention window was set to $W_{\min} = 4$ since the number of orderers is not large and contention should be low to moderate. Size of an ordering request was set to l_{db} =1Kbyte so that it can contain a batch of 5 to 10 transactions, depending on transaction size. Payload of RTS and CTS messages was set to $rc = 160$ bits and overall header size was $H_{all} = 800$ bits.

We begin by presenting probability distributions of one-way delay and maximum one-way delay for clusters with 4, 7, and 10 orderers together with the corresponding Gamma approximations represented with red dashed lines. In both cases, we consider clusters of small and medium areas, i.e., with $S_c = 5$ and 10 ms, respectively. The results are shown in Fig. 6; separate diagrams show the probability distributions of maximum one-way for 4, 7 and 10 orderers, respectively. As can be seen, the fit between the calculated distributions and their Gamma approximations is quite good.

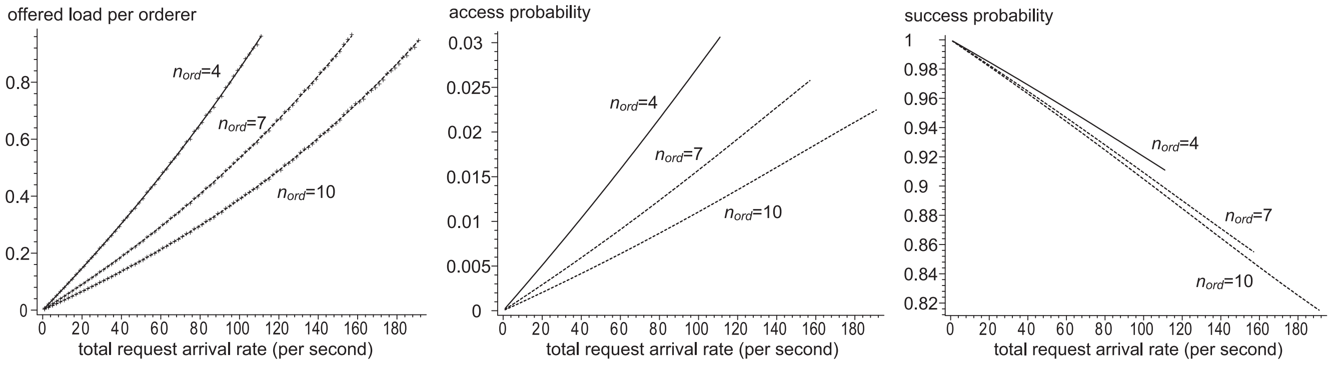
As can be seen from Table I, mean maximum one-way delay obtained from (8) and mean record insertion time from (18) both increase with the number of orderers and with the span of the one-way delay. If we compare values for small and medium cluster sizes we note that the ratio between the delays is roughly 1:2 which matches ratio of maximum delays in the one-way delay distribution.

Assuming that IoT proxies are geographically uniformly distributed around orderers and/or that they can perform load balancing towards orderers, each orderer will receive the same portion of the total external and internal request rate, $\lambda_{r,s} = \lambda_s / n_{ord}$. Internal request rate should be kept at few percent of the external rate by having moderately large blocks. We use the total request rate (in requests per second) as the independent variable and number of orderers and one-way delay span as parameters. In all diagrams below, the numerical results for 4, 7, and 10 orderers are presented with solid lines, dotted lines, and long dashed lines, respectively while simulation results are shown with cross symbols.



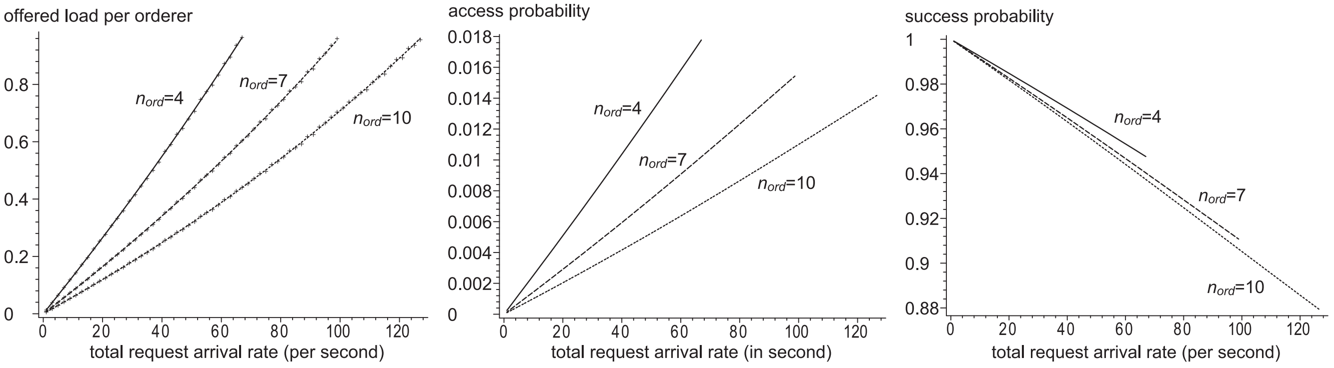
(a) (b) (c) (d)

Fig. 6. Probability distribution of one-way delay and maximum one-way delay, together with their Gamma approximations (red dashed lines), for small ($S_c = 5$ ms) and medium ($S_c = 10$ ms) cluster sizes. (a) One-way delay. (b) Maximum one-way delay, 4 orderers. (c) Maximum one-way delay, 7 orderers. (d) Maximum one-way delay, 10 orderers.



(a) (b) (c)

Fig. 7. Loading descriptors for an orderer node within small area cluster ($S_c = 5$ ms). (a) Offered load. (b) Medium access probability. (c) Probability of successful request transmission.

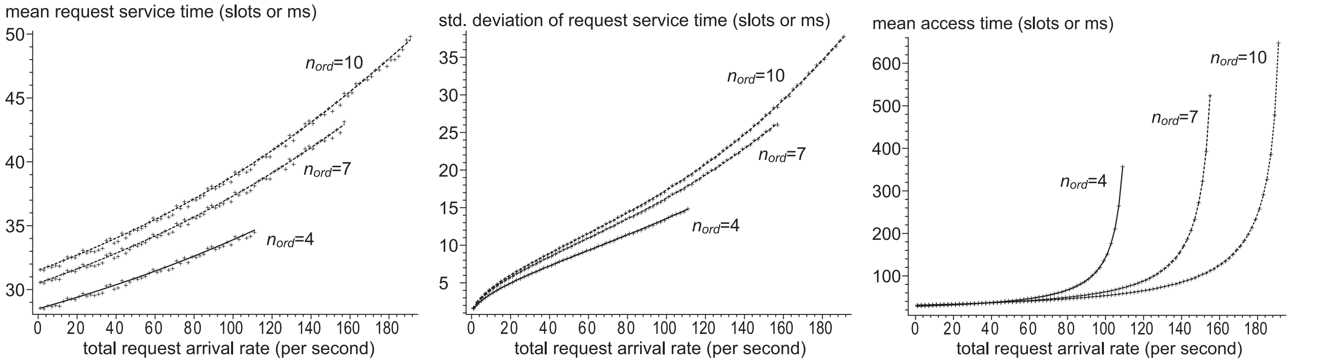


(a) (b) (c)

Fig. 8. Loading descriptors for an orderer node within medium area cluster ($S_c = 10$ ms). (a) Offered load. (b) Medium access probability. (c) Probability of successful request transmission.

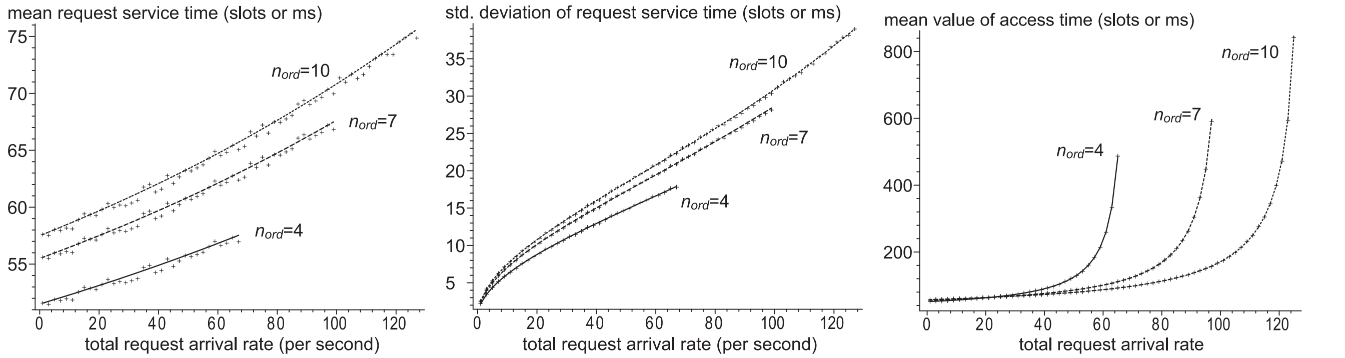
Figs. 7(a) and 8(a) present offered load per orderer for small- and medium area clusters, respectively. The medium area cluster reaches stability limit $\rho_r \approx 1$ (i.e. maximum capacity) around 64, 98, and 125 batch requests per second for 4, 7, and 10 orderers, respectively, which corresponds to individual load of 16, 14, and 12.5 batch requests per second per orderer. In small area cluster, the corresponding capacity limits are higher due to smaller request service times. However, the increase is only about 70, 58 and 52% respectively on account of larger contention among orderers, as will be confirmed with following figures.

When the number of orderers at both cluster area sizes is increased, the offered load per orderer decreases, which decreases the queue length of any single orderer and decreases



(a) (b) (c)

Fig. 9. Moments of request service time and system access (i.e., response) time for an orderer node in small area cluster ($S_c = 5$ ms). (a) Mean value. (b) Standard deviation. (c) Mean system access (response) time.



(a) (b) (c)

Fig. 10. Moments of request service time and system access (i.e., response) time for an orderer node in medium area cluster ($S_c = 10$ ms). (a) Mean value. (b) Standard deviation. (c) Mean system access (response) time.

medium access probability, as can be seen in Figs. 7(b) and 8(b) for small- and medium size clusters, respectively. Contention also increases due to random access, as can be seen from the diagrams in Figs. 7(c) and 8(c) which show probability of successful RTS request transmission. For the same offered load, success probability decreases slightly with the number of orderers, showing the impact of RTS collisions. However access probabilities and transmission success probabilities show that the level of contention close to capacity limits is higher in small area cluster compared to the medium area one. Again, this is a consequence of higher offered load and increased orderer activity on the medium.

Figs. 9 and 10 show the first two moments of request service time and mean system access time per orderer for small- and medium area clusters, respectively. As before, numerical results are presented with lines and simulation results with points. We notice that simulation results are within 1 to 2.5% around numerical ones, which corroborates the validity of our approach. Mean request service times increase super-linearly with request traffic intensity and/or number of orderers, due to increased frequency of RTS collisions close to the capacity limit and freezing of backoff activity due to successful request service.

Standard deviation increases sub-linearly for offered loads smaller than 0.6, at which value it goes through an inflection point and begins to increase super-linearly: from few backoff slots to around 50% of the mean value of request service time. since it is affected by frozen time of backoff counter and also with jump increases of service time after RTS collisions. Standard deviation of request service time also increases with the number of orderers since backoff time is uniformly distributed and its impact increases with the increase of the number of ordering nodes.

Mean system response time, which includes waiting time and request service time, is shown in Figs. 9(c) and 10(c) for small- and medium area clusters, respectively. Small area cluster scenario has around 60% larger capacity than its medium area counterpart, but in both diagrams curves for 4, 7, and 10 orderers are shifted apart due to higher total system capacity. For very low load, system response time is mostly affected by request insertion time and it increases with the number of orderers. With the increase of total load, access time mainly increases due to waiting time in queue which increases with offered load; the increase occurs earlier if the system has fewer orderers. Waiting time is mostly affected with the number of requests in the orderer queue which has distribution close to Poisson

TABLE II
TOTAL SYSTEM CAPACITY PER CLUSTER IN BATCH REQUESTS PER SECOND (ONE BATCH CONTAINS 5 TO 10 TRANSACTIONS).

cluster area	$n_{ord} = 4$	$n_{ord} = 7$	$n_{ord} = 10$
small	109	155	190
medium	64	98	125

distribution. Since request service time is small, for medium to high request traffic intensity, mean access time and its standard deviation have similar values, with standard deviation reaching about 90%, 102% and 110% of the mean value for 4, 7 and 10 orderers respectively. Higher number of orderers increases chance of RTS collisions and increases variability of access time.

These results are conveniently summarized in Table II which shows total system capacity per cluster obtained from the diagrams. As can be seen, system capacity increases with the number of orderers. Moreover, system capacity is higher in small area cluster than in the medium area one, but the ratio is nowhere near 1:2 observed for one-way delays, mostly on account of increased collisions.

From the data shown in Figs. 7 to 10, we can conclude that use of multiple entry points in the PBFT system brings forth two problems.

- Increasing the number of orderers decreases offered load per orderer but also increases maximum one way delay and contention on the medium. Increased contention creates RTS collisions and, consequently, increases request service time. This counter-action gradually diminishes the benefit of adding more orderers. This can be seen from Figs. 9(a) and 10(a), and also from Figs. 7(c) and 8(c), where distance between curves for 7 and 10 orderers is smaller than distance between curves for 4 and 7 orderers.
- Increasing physical distance among the nodes increases the time needed for inserting a record into the distributed ledger. This increases offered load per node for the same total request arrival rate. In this case, the only way to extend system capacity is to increase the number of orderers which accentuates the problem of contention. As the result, request access probability is lower and system saturation occurs due to large request insertion time. Indeed the capacity of a medium area cluster with 10 orderers in only about two-thirds of the value obtained with a comparable small area cluster, even though the area covered by the medium size cluster is four times as large.

It is worth noting that, from the aspect of PBFT application, increasing the number of orderers also increases system fault tolerance.

X. CONCLUSION

In this paper, we have presented a PBFT ordering service suitable for covering certain geographical areas by connecting to proxies that collect data from IoT domains. Consensus algorithm ensures that orderers have identical order of transactions in their memory pools so that they can be recorded in a blockchain ledger. The service allows ordering requests to be accepted through any ordering peer, thus avoiding performance penalty and eliminating a single point of failure present in other PBFT implementations. This is accomplished through a contention resolution scheme, based on the classical CSMA/CA technique, which ensures atomic insertion of records into orderer memory pools.

We have modeled multiple entry ordering service and evaluated it for small- and medium size geographical clusters, and for system size of 4, 7 and 10 orderers. We have observed the two-way action of increased number of orderers through decreased offered load and increased maximum one-way delay and contention on the medium which limits system performance. System capacity per cluster will increase with further decrease of cluster dimensions. However, such small cluster can cover only city-wide areas; when larger scale coverage is needed, further interconnection and consensus schemes among the clusters are necessary. This will be the subject of our future work.

REFERENCES

[1] I. Abraham, G. G. Gueta, D. Malkhi, M. Yin, and M. K. Reiter,“HotStuff: BFT consensus in the lens of blockchain,” 2019, *arXiv:1803.05069*.

[2] Y. Amir, B. Coan, J. Kirsch, and J. Lane, “Prime: Byzantine replication under attack,” *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 4, pp. 564–577, Jul./Aug. 2011.

[3] E. Androulaki *et al.* “Hyperledger fabric: A distributed operating system for permissioned blockchains,” in *Proc. 13th EuroSyst. Conf.*, 2018, p. 30.

[4] P.-L. Aublin, S. Ben Mokhtar, and V. Quéma, “RBFT: Redundant byzantine fault tolerance,” in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst.*, 2013, pp. 297–306.

[5] J. Baek and Y. Zheng, “Identity-based threshold signature scheme from the bilinear pairings,” in *Proc. Int. Conf. Inf. Technol.: Coding Comput.*, 2004, pp. 124–128.

[6] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2001, pp. 514–532.

[7] E. Buchman, “Tendermint: Byzantine fault tolerance in the age of blockchains,” Ph.D. thesis, Univ. Guelph, Ontario, Canada, 2016.

[8] M. Castro and B. Liskov, “Practical Byzantine fault tolerance,” *OSDI: Symp. Operating Syst. Des. Implementation*, New Orleans, LA, USA, Feb. 1999.

[9] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, “Making Byzantine fault tolerant systems tolerate Byzantine faults,” in *Proc. 6th USENIX Symp. Networked Syst. Des. Implementation*, Boston, MA, USA, 2009.

[10] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” *J. ACM*, vol. 32 no. 2, pp. 374–382, 1985.

[11] G. G. Gueta *et al.*, “SBFT: A scalable and decentralized trust infrastructure,” in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2019, pp. 568–580.

[12] *IEEE Standard Local Metropolitan Area Networks - Specific Requirements - Part* 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Standard* 802.11-2016 (Revision of IEEE Standard 802.11-2012), Dec. 2016, pp. 1–3534.

[13] L. J. Kleinrock, *Queuing Systems, Theory*, vol. I. New York, NY, USA: Wiley, 1972.

[14] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, “ZYZZYVA: Speculative Byzantine fault tolerance,” *ACM Trans. Comput. Syst.*, vol. 27, no. 4, pp. 1–39, Jan. 2010.

[15] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Boston, MA, USA: Addison-Wesley Longman, 2016.

[16] L. Lamport, R. Shostak, and M. Pease, “The Byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[17] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, “The honey badger of BFT protocols,” *ACM SIGSAC Conf. Comput. Commun. Secur.*, pp. 31–42, 2016.

[18] J. Mišić, M. Ali, and V. Mišić, “Protocol architectures for CoAP IoT domains,” *IEEE Netw.* vol. 32, no. 4, pp. 811–817, Jul./Aug. 2018.
[19] J. Mišić and V. Mišić, “Bridging between IEEE 802.15.4 and IEEE 802.11 networks for multiparameter healthcare sensing,” *IEEE J. Sel. Areas Commun. Wireless Ser.*, vol. 27, no. 4, pp. 435–449, May 2009.
[20] J. Mišić, V. B. Mišić, and X. Chang, “Kernel based estimation of domain parameters at IoT proxy,” *IEEE Globecom*, Abu Dhabi UAE, 2018, pp. 1–6.
[21] J. Mišić, V. B. Mišić, and X. Chang, “Comparison of single- and multiple entry point PBFT for IoT blockchain systems,” in *Proc. IEEE 92nd Veh. Technol. Conf.: VTC2020-Fall, Workshop New Netw. Architect. Powering Internet-of-Things*, Oct. 2020.
[22] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” Accessed: Jan. 11, 2021, [Online]. Available: https://bitcoin.org/bitcoin.pdf
[23] D. Ongaro and J. K. Ousterhout. “In search of an understandable consensus algorithm,” in *Proc. (USENIX) Annu. Tech. Conf.*, 2014, pp. 305–319.
[24] J. Sousa and A. Bessani, “From Byzantine consensus to BFT state machine replication: A latency-optimal transformation,” in *Proc. 9th Eur. Dependable Comput. Conf.*, 2012, pp. 37–48.
[25] J. Sousa, A. Bessani, and M. Vukolić, “A Byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform,” in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2018, pp. 51–58,.
[26] H. Takagi. *Queueing Analysis*, vol. 1: *Vacation and Priority Systems*. North-Holland, Amsterdam, The Netherlands, 1991.
[27] G. S. Veronese, M. Correia, A. N. Bessani, and L. C. Lung, “Spin one’s wheels? Byzantine fault tolerance with a spinning primary,” in *Proc. 28th IEEE Int. Symp. Reliable Distrib. Syst.*, 2009, pp. 135–144..
[28] M. Wand and M. Jones. *Kernel Smoothing*. 1st ed., Chapman & Hall, Inc., 1995.
[29] Y. Xiao and J. Roshdal. “Throughput and delay limits of IEEE 802.11.” *IEEE Commun. Lett.*, vol. 6, no. 8, pp. 355–357, Aug. 2002.
[30] Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, and Y. Zhang, “A. blockchain-based nonrepudiation network computing service scheme for industrial IoT,” *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3632–3641, 2019.
[31] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, “Resource trading in blockchain-based industrial Internet of Things,” *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3602–3609, 2019.
[32] H. Zhai, Y. Kwon, and Y. Fang, “Performance analysis of the IEEE 802.11 MAC protocols in wireless LANs,” *Wireless Commun. Mobile Comput.*, vol. 3, pp. 2586–2590, 2003.

Jelena Mišić (Fellow, IEEE) is Professor of computer science with Ryerson University in Toronto, Ontario, Canada. She has authored or coauthored four books, more than 125 papers in archival journals and close to 190 papers at international conferences in the areas of computer networks and security. She is on editorial boards of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE IOT JOURNAL, IEEE NETWORK, COMPUTER NETWORKS AND AD HOC NETWORKS. She is a Fellow and Member of ACM.

Vojislav B. Mišić (Senior Member, IEEE) is a Professor of computer science with Ryerson University in Toronto, Ontario, Canada. His research interests include performance evaluation of wireless networks and systems and software engineering. He is on the editorial boards of IEEE TRANSACTIONS ON CLOUD COMPUTING, AD HOC NETWORKS, PEER-TO-PEER NETWORKS AND APPLICATIONS, and *International Journal of Parallel, Emergent and Distributed Systems* of ACM.

Xiaolin Chang is Professor with the School of Computer and Information Technology, Beijing Jiaotong University. Her current research interests include edge/cloud computing, network security, security and privacy in machine learning.

Haytham Qushtom is working toward the Ph.D. degree in computer science with Ryerson University in Toronto, Ontario, Canada. His research interests include blockchain technology, Internet-of-Things, and distributed computing systems.