Comparison of single- and multiple entry point PBFT for IoT blockchain systems

Jelena Mišić
Ryerson University
443 PUBLICATIONS **4,477** CITATIONS

Vojislav Misic
Ryerson University
397 PUBLICATIONS **3,639** CITATIONS

Xiaolin Chang
Beijing Jiaotong University
136 PUBLICATIONS **886** CITATIONS

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。

Comparison of single- and multiple entry point PBFT for IoT blockchain systems

Jelena Mišić[1], Vojislav B. Mišić[1], Xiaolin Chang[2]
[1]Ryerson University, Toronto, ON, Canada
[2]Beijing Key Laboratory of Security and Privacy in Intelligent Transportation
Beijing Jiaotong University, Beijing, China

***Abstract*—This work deals with problem of deployment of Practical Byzantine Fault Tolerance (PBFT) consensus algorithm in Internet of things (IoT) applications requiring strict order among records linked in blockchains at multiple geographical points. Due to the needs of reliability and coverage of larger geographical areas it becomes necessary to extend current PBFT systems with single entry node towards multiple entry nodes. In this work we model PBFT systems with single and multiple entry points and compare their performance. We have implemented multiple entry system using CSMA/CA algorithm over fully connected P2P networks. We have compared two systems with four orderers against increasing system load and against increasing geographical coverage. Our results indicate that system with four entry points has double capacity over the system with single entry point.**

I. Introduction

Recently it was observed that records kept in IoT applications run by multiple providers can be organized in the so-called consortium blockchains. It may be expected that multiple operators will keep their data on blockchain and participate in block creation, even though they don't necessarily trust each other. The lack of trust, as well as possible software and hardware faults, bring forth a Byzantine component among the nodes which are creating blocks. To allow for efficient blockchain creation and maintenance, a modification of Practical Byzantine Fault Tolerance (PBFT) [3] is used. In this system there are $n = 3f + 1$ nodes which run consensus algorithm that tolerates at most f faulty or malicious nodes. The original PBFT [3] deals with read/write operations in distributed databases, but it was used to insert transactions in blockchain [11] for potential use in Hyperledger Fabric [1]. However, this approach uses single entry into the ordering system.

In this work we compare classical PBFT deployment in blockchain architecture using single entry point against the architecture where every node can accept ordering requests and submit it to ordering service. We assume that transactions are validated outside the ordering system which focuses only on total order among them. For the case with multiple proposing orderers we have developed a protocol for contention avoidance among orderers. (Following the terminology from Hyperledger Fabric [1], [11], servers that work on consensus in blockchain are known as *orderers*.) Protocol is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol [14], [13], [7] so that the entry node has to reserve the medium prior to submitting a request for ordering. Contention is possible during medium reservation and inserts additional latency in the protocol. We compare the performance of two approaches for systems with four orderers against increasing system load and network coverage.

本文对于有多个请求者的情况下设计了一个协议，从而避免了请求者出现竞争。

The paper is organized as follows. Section II describes system architecture. In Section III, we present the medium reservation protocol, followed by the PBFT consensus protocol with multiple entry points. Analytical model of the ordering service is given in Section IV, and model of CSMA/CA contention among orderers is given in Section V. Performance evaluation is presented in Section VI. Finally, Section VII concludes the paper.

II. System Architecture

The IoT system is comprised of a number of IoT domains controlled by at least one proxy each. The proxies collect data from IoT sensors managed by one or more gateways [8] and validate data before sending it for ordering and inclusion in the replicated ledger. Unlike financial systems such as Bitcoin or Ethereum, data validation is simpler and faster; in particular, does not involve backward searches through existing records to ascertain the existence of sufficient funds or the like. However, some pre-processing involving machine learning can also be performed.

Each proxy is connected to at least one orderer. The orderers execute the actual consensus protocol and maintain the replicated ledger from which the data is read by IoT system client applications. Ordering servers are interconnected using overlay network using TCP connections.

Each orderer keeps a memory pool to store data records that will subsequently be packaged in blocks. The content of the memory pool is updated by record/transaction insertion service that will be explained in next Section. Assuming there are $n_{ord} = 3f + 1$ orderers, the insertion service is capable to achieve identical total order of transactions/records in memory pools belonging to all orderers in the presence of up to f faulty orderers. Faults may occur due to hardware or software faults, but also through 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 [6], [3].

Periodically, an orderer will group a number of records (typically, transactions or groups thereof) 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. Proxies and client applications can then request those blocks in order to access the data therein. As the blocking and blockchain updating uses the same consensus protocol as the record insertion service, we do not consider it in detail here.

III. Single and multiple entry PBFT

In single entry PBFT a single leader node receives requests from the clients stores them in the queue and sequentially submits them to other orderers using PBFT without any contention. It is possible to change the leader in different time periods [3].

A. Multiple entry and medium reservation protocol

Multiple entry PBFT allows any of the ordering nodes to lead a round of record insertion without contention i.e. to ensure atomicity of record insertion. This is achieved by including the protocol for medium reservation. This protocol is based on fully connected overlay TCP network among orderers which effectively achieves a broadcast communication medium. This protocol requires that each of the orderers continuously listens to all of its peers at the corresponding TCP sockets.

Prior to initiating reservation, an orderer must check whether the previous record insertion is completed. If so, the orderer sends a Request to Send (RTS) message simultaneously over all TCP connections to its peers which respond with Clear to Send (CTS) messages – but not immediately, as will be explained below. CTS message sent from a peer indicates the promise not to engage in bandwidth reservation on behalf of other orderers until the current request has been completed. In this manner, the RTS/CTS sequence enables *atomic* multicast of subsequent PRE-PREPARE message.

However this procedure is prone to *virtual collisions* of RTS requests due to finite signal propagation time. To ensure reliability, it is necessary to calculate the following two parameters: the duration of the *vulnerable period* which starts upon receipt of the first RTS and lasts as long as it is possible to receive another one, and the duration of the timeout period after which a CTS cannot possibly 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) during the vulnerable period, it interprets it as a virtual collision, aborts its reservation attempt, and enters the backoff phase before attempting to send this RTS again.

The receiving orderer waits for the vulnerable period before sending the CTS response to a RTS request. If no other RTS is received during the vulnerable period, the orderer sends the CTS back to the initiating orderer. However, if another RTS message is received during that period, the receiving orderer declares a collision and sends no CTS back to any of the initiating orderers.

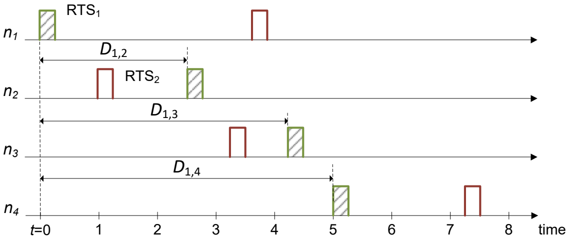


Fig. 1. Timing of RTS requests and the vulnerable period.

B. Duration of timeout 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 $t_0 + \tau$, respectively. The value of τ ranges between zero and one-way propagation time between nodes i and l, i.e., $0 \leq \tau < D_{i,l}$ as can be seen in Fig. 1 for $n_{ord} = 4$, $i = 1$, $l = 2$ and sample destination node $w = 4$. Upper limit of $\tau < D_{i,l}$ 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 $\tau + D_{l,w}$, respectively. These requests may arrive to different orderers in different order and, thus, lead to conflicting CTS messages if virtual collision is not detected.

The maximum time period between the arrival of conflicting RTS messages is labeled as the vulnerable period, during which the node that received a RTS should not send the corresponding CTS. Considering Fig. 1, the time of arrival of first RTS to node w will be $\min(D_{i,w}, D_{l,w} + \tau)$. 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

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

where index j corresponds to other orderers in the network – in this example, nodes n_2 and n_3.

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

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

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

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

One-way propagation delays can be measured using round trip time and further estimated using exponential weighted moving average EWMA [5] or using the method of kernels [12], [9]. 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 should be chosen as a small fraction of maximum vulnerable period.

C. PBFT ordering protocol

The remainder of the ordering protocol generally follows the stages of PBFT with the note that single entry and multiple entry versions are different only in the first phase:

In single entry PBFT, when leader completes one record insertion and obtains $2f + 1$ COMMIT messages, it can start new cycle with PRE-PREPARE message.

In multiple entry PBFT, the requesting orderer must receive $2f$ CTS replies for bandwidth reservation before it can start atomic broadcast of the PRE-PREPARE message containing processed client request from the head of the orderer's queue. Note that an orderer which did not return CTS reply may attempt to transmit a RTS for the new request during processing of current request. However, its/their ID is known to all other orderers so they will not accept it until the current request is processed.

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.

Every ordering peer 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 in order to be able to proceed. After that, peer j transmits a COMMIT message acknowledging the proposed sequence numbers of records in its memory pool simultaneously over all its TCP sockets. When at least $2f + 1$ COMMIT messages, including its own COMMIT, have been received, 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 bandwidth reserved with RTS/CTS message exchange and allows peer j to start bandwidth reservation for new proposals, if any.

In multiple entry case each orderer achieves atomic broadcast by sending RTS after backoff over all of its TCP connections and listening on all of them for the CTS. If any other content is received during the vulnerable period, the orderer will declare a RTS collision and perform new backoff before the request retransmission. During the backoff process 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, this scheme can be modeled similar to CSMA/CA in 802.11 DCF. Receipt of CTS means that ordering service has been reserved for target orderer to insert it request in memory pools of all other orderers. This reservation is terminated by receipt of $2f + 1$ COMMIT messages; after that, all orderers inform the client that submitted the record that its request has been accepted.

每次进行原广播之前，都要发送RTS，直到收足够的CTS，
否则就删除RTS冲突，然后回退一个新的时间窗口

对于每个memory pool的容量，应该给出假设，具有足够大的空间，能够容纳任意数量的加以或者记录

IV. Model of Record Insertion Time

Single entry and multiple entry PBFT differ only in medium reservation phase. We will first develop the model for medium reservation phase and proceed with other common phases. 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 $dx_i, dy_i \in \{0 \,..\, R\}$. 采用笛卡尔坐标系记录节点i的横纵坐标

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 $x \in \{0 \,..\, S_c\}$ and vertical $y \in \{0 \,..\, S_c\}$ components which contribute to the total delay using geometrical principles.

假设一对节点之间的路由器数量与它们之间的距离成正比

两个节点单项延时

Cumulative distribution function (CDF) of one-way delay can be derived using constant s_0 as: 这个公式是怎么的出来的？其中变量u是什么？

$$D(u) = \frac{(6u^2\pi(S_c)^2 + 3u^4 - 16S_c u^3)}{6(S_c)^4 s_0} \tag{4}$$

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 (3).

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

$$P(\max(D_1) < u) = P(D_{1,2} < u) \cdots P(D_{1,n_{ord}} < u) \tag{5}$$

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

$$\max(D_1(u)) = \prod_{i=2}^{n_{ord}} D_{1,i}(u) = D(u)^{n_{ord}-1} \tag{6}$$

Since all one-way delays in the network follow the same probability distribution, (6) also represents maximum one-way delay for any orderer in the network, i.e., $\mathscr{D}(u) = \max(D_1(u))$. Then, we can compute probability density function (pdf) and

Laplace Stieltjes Transform (LST) of maximum one-way delay as

$$\mathscr{d}(u) = \frac{d(\mathcal{D}(u))}{du}$$

$$\mathcal{D}^*(s) = \int_{u=0}^{\infty} e^{-su}\mathscr{d}(u)du \tag{7}$$

Probability distribution from (3) 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

$$td(x) = \int_{u=0}^{x} d(u)d(x-u)du$$

$$tD(u) = \int_{x=0}^{u} td(x)dx \tag{8}$$

Maximum of this sum can be obtained as follows. Assuming for the moment that $n_{ord} = 4$, $i = 1$, and $w = 4$, as shown in Fig. 1, the following holds for the initiating orderer o_1 with respect to the destination orderer $w = o_4$:

$$P(\max(tD_1) < u) = P(tD_{1,2,4} < u) \cdot P(tD_{1,3,4} < u) \tag{9}$$

leading to the expression where we calculate CDF of maximum two hop delay since by definition $tD(u) = P(tD \leq u)$:

$$\max(tD(u)) = (tD(u))^{n_{ord}-2} \tag{10}$$

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

$$\theta(u) = \frac{d(\Theta(u))}{du}$$

$$\Theta^*(s) = \int_{u=0}^{\infty} e^{-su}\theta(u)du \tag{11}$$

Further we need to sum maximum two hop delay with one hop delay and find the distribution of the maximum. Probability density function (pdf) and CDF of the sum, respectively, are given with

$$twd(y) = \int_{u=0}^{y} d(u)td(y-u)du$$

$$twD(u) = \int_{x=0}^{u} twd(y)dy \tag{12}$$

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

$$\mathcal{W}(u) = (twD(u))^{n_{ord}-1}$$

$$\omega(u) = \frac{d(\mathcal{W}(u))}{du}$$

$$\mathcal{W}^*(s) = \int_{u=0}^{\infty} e^{-su}\omega(u)du \tag{13}$$

Timeout value for the issuer of RTS request can be chosen as $TO = \overline{\mathcal{W}} + k\sqrt{var(\mathcal{W})}$, where $k \in (1\,..\,4)$.

Let us 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 of times needed for PRE-PREPARE, PREPARE and COMMIT are all equal to $\mathcal{D}^*(s)$. Then, LST for the record insertion time for single entry system is $T_{ins}^*(s) = (\mathcal{D}^*(s))^3$ while for multiple entry system without collisions with other orderers within ordering service it is $T_{ins}^*(s) = \mathcal{W}^*(s)(\mathcal{D}^*(s))^3$. LST for the total insertion time (taking into account submission time from the proxy and reply time to the proxy) is $T_{ins,tot}^*(s) = T_{ins}^*(s)e^{-l_d s}(T_c^*(s))^2$.

V. Model for medium reservation protocol

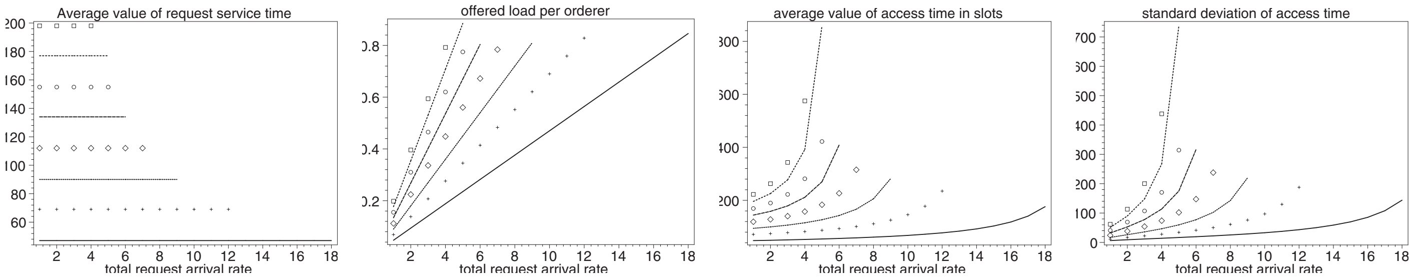
Using the distributions for timeout and multiple entry record insertion time, we can model CSMA/CA protocol for collision-free access for ordering peers. This is necessary in order to perform atomic bandwidth reservation for further PBFT steps. It is inspired by Distributed Coordinated Function (DCF) 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.

每个与其他orderers拥有TCP连接的peer orderer可以载波监听，并且每个PBFT的步骤都采用多播的形式。

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 the 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. 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. 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. The behavior of this system can be modeled following the approach presented in [7], [13], [14]; the details are omitted due to space limitations.



(a) Mean request service time.

(b) Offered load.

(c) Mean access time.

(d) Standard deviation of access time.

Fig. 2. Performance as seen by leading orderer node in system of four orderers with single leader.

Finally, with the distribution of record insertion time (which includes possible RTS collisions for multiple orderers) we developed M/G/1 queueing model for record inserting nodes.

VI. Performance Evaluation

The focus of performance evaluation is to compare ordering system with $n_{ord} = 4$ orderers with a single entry point to that same system in which entry can be made from any ordering node. Total load to the system is expressed as number of ordering requests per second (rps) λ_s. All TCP connections among orderers have throughput of 2Mbps. Orderers are randomly placed in Cartesian system with horizontal and vertical coordinates following same uniform distribution in the range $0\,..\,R$ (in meters), resulting in the corresponding single dimensional component of one-way delay in the range $0\,..\,S_c$ where S_c ranges between 20 and 90 milliseconds. Symbols for each value of S_c are presented according to Table I. For each value of one-way delay we have increased total input load up to the offered load of $\rho \approx 0.9$ which we will assume to be system capacity.

Backoff slot for contention resolution among orderers was set to $\delta = 1ms$ and all timing figures are represented as multiples of backoff slots. In the case of multiple entry ordering system minimal contention window was set to $W_{min} = 4$ since the number of orderers is not large. Size of an ordering request was set to l_{db} =1 Kbyte which can contain a batch of transactions. Payload of RTS and CTS messages was set to $rc = 160$ bits and overall header size was $H_{all} = 800$ bits.

Performance descriptors for single entry ordering system are shown in Fig. 2, assuming request rate of λ_s at the leader node. Fig. 2(a) shows mean record insertion time (request service time) which covers PBFT but without any bandwidth reservation process. Lines on the graph are horizontal since record insertion time does not depend on the system load; instead, it depends only on the one-way delay distribution between the orderers.

Since record insertion time increases when system dimensions increase, offered load for leader is increasing, Fig. 2(b) while the system capacity, shown by the length of horizontal lines in Fig. 2(a), decreases. We notice that system capacity is limited to around 18rps when $S_c = 20$, i.e., with mean one-way delay and mean maximum one-way delay shown in Table I.

TABLE I
Line types for various values of S_c

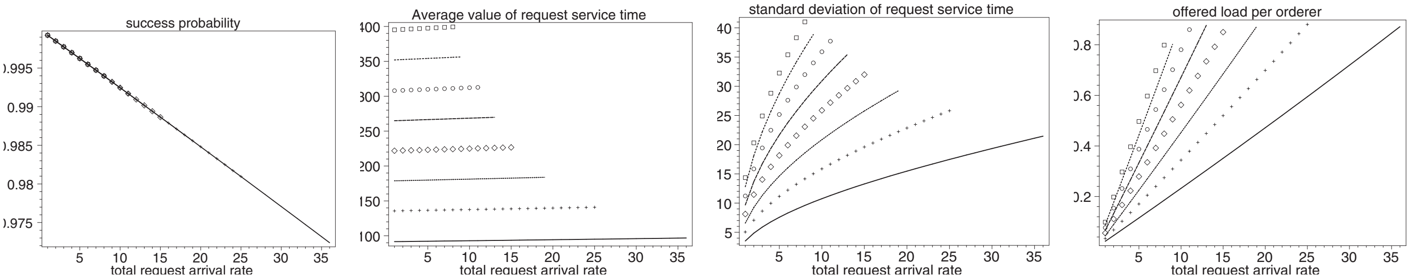
S_c	mean one way delay	mean maximum one way delay	symbol
20ms	10.25ms	14.39ms	solid line
30ms	15.39ms	21.59ms	crosses
40ms	20.5ms	28.79ms	space-dot line
50ms	25.63ms	35.99ms	diamonds
60ms	30.75ms	43.19ms	space-dash line
70ms	35.88ms	50.39ms	circles
80ms	41.01ms	57.59ms	dash-dot line
90ms	46.13ms	64.79ms	boxes

Mean request access time and its standard deviation, which includes waiting time in the leading orderer queue and record insertion time, are shown in Figs. 2(c) and 2(d). We notice that standard deviation grows with offered load but is still below the mean value for offered loads smaller than $\rho = 0.9$.

Next experiment assumes that each orderer can insert the request in the system using bandwidth reservation method. Assuming that IoT proxies can perform load balancing towards the orderers, each orderer will receive λ_s/n_{ord} of the total request rate. Horizontal axis presents total system load in rps. Bandwidth reservation takes time and significantly increases record insertion time, as shown in Fig. 3(b), compared to the case with single entry point shown in Fig. 2(a).

However, multiple entry points allow orderers to share the load which reduces offered load per orderer. Fig. 3(d) presents offered load per orderer, which is significantly higher than in the equivaent single-entry system, Fig. 2(b). For a given geographic range (i.e., value of S_c), system capacity is doubled by moving from one entry point to four. Probability of no RTS collision in the CSMA/CA bandwidth reservation process is shown in Fig. 3(a). Note that bandwidth reservation introduces additional randomness in record insertion time, whereas in a single-entry system record insertion time depends only on variability of inter-node delays.

Mean and standard deviation of response time (which includes waiting time and request service time) are shown in Figs. 4(a) and 4(b). Curves for $S_c = 20\,..\,90$ are shifted apart due to decreasing total system capacity; they indicate coefficient



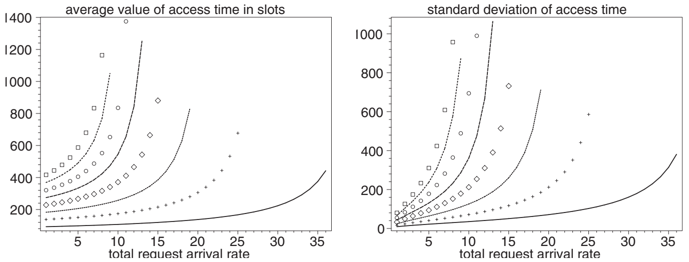
(a) Prob. of successful medium reservation.

(b) Mean request service time.

(c) Standard deviation of request service time.

(d) Offered load.

Fig. 3. Performance descriptors for an orderer node in the multiple-entry system.



(a) Mean value.

(b) Standard deviation.

Fig. 4. Descriptors of access time in the multiple-entry system.

of variation of about 0.8 for low loads, up to 1 as load increases. Curves also indicate geographical limits for the desired ranges of system load: e.g., for system load of 20rps, mean one way delay corresponds to 15.4ms (i.e., $S_c \approx 30$ms). If system load does not exceed 10rps, mean one way delay can be around 30-35ms (or $S_c \approx 60 - 70$ms).

VII. Conclusion

In this work we have compared two paradigms of blockchain deployment in IoT systems. Both paradigms accept ordering requests from surrounding IoT proxies and deploy PBFT scheme in order to establish strict record ordering. First paradigm is PBFT record ordering service with a single entry point that accepts the requests and provides serialization of requests. This scheme eliminates contention among orderers but leads to a large load at the entry point. Second paradigm allows multiple entry points, i.e., ordering requests can come through any ordering peer, but a variant of the CSMA/CA protocol is used to resolve contention. In both cases, PBFT ensures that identical order of transactions will be eventually reached at each orderer.

Our performance evaluation shows that the use of multiple entry points decrease queuing load per orderer despite contention and increased overhead of bandwidth reservation algorithm. As the result, the capacity of the same system with four orderers is doubled compared to the single entry point system. We have also outlined system capacities for a range of one-way delays and found super-linear decrease of system capacity with linear increase of one way delays. Presented results can be used for IoT system dimensioning with respect to the trade-off between one-way delays (geographical coverage), system load and reliability issues, thus avoiding single point of failure.

In future work we will evaluate impact of the number of orderers, request size, and also block formation service on system performance.

References

[1] E. Androulaki *et al*. Hyperledger Fabric: a distributed operating system for permissioned blockchains. In *Proc. Thirteenth EuroSys Conference*, page 30, 2018.

[2] V. Buterin. Ethereum: A next-generation smart contract and decentralized application platform, 2017. http://ethereum.org/ethereum.html.

[3] M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *OSDI: Symposium on Operating Systems Design and Implementation*, 1999.

[4] IEEE standard for local and metropolitan area networks – specific requirements – part 11: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, Dec. 2016.

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

[6] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[7] J. Mišić and V. Mišić. Bridging between IEEE 802.15.4 and IEEE 802.11 networks for multiparameter healthcare sensing. *IEEE Journal of Selected Areas in Communications – Wireless Series*, 27(4):435–449, 2009.

[8] J. Mišić, V. B. Mišić, and F. Banaie. Reliable and scalable data acquisition from IoT domains. In *Proceedings of IEEE GlobeCom*, Singapore, 2017.

[9] J. Mišić, V. B. Mišić, and X. Chang. Kernel based estimation of domain parameters at IoT proxy. In *Proceedings of IEEE Globecom*, Abu Dhabi UAE, 2018.

[10] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[11] J. Sousa, A. Bessani, and M. Vukolić. A Byzantine fault-tolerant ordering service for the Hyperledger Fabric blockchain platform. In *2018 48th Annual IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN)*, pages 51–58, 2018.

[12] M. Wand and M. Jones. *Kernel Smoothing*. Chapman & Hall, Inc., first edition, 1995.

[13] Y. Xiao and J. Roshdal. Throughput and delay limits of IEEE 802.11. *IEEE Commun. Lett.*, 6(8):355–357, Aug. 2002.

[14] H. Zhai, Y. Kwon, and Y. Fang. Performance analysis of the IEEE 802.11 MAC protocols in wireless LANs. *Wireless Communications and Mobile Computing*, 4:917–931, 2004.