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。

Trade-offs in large blockchain-based IoT system design

Jelena Mišić[1], Vojislav B. Mišić[1], and 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*—The well known Practical Byzantine Fault Tolerance (PBFT) consensus algorithm is not well suited to blockchain-based Internet of Things (IoT) systems which cover large geographical areas. To reduce queuing delays and eliminates a permanent leader as a single point of failure, we use a multiple entry, multi-tier PBFT architecture and investigate the distribution of orderers that will lead to minimization of the total delay from the reception of a block of IoT data to the moment it is linked to the global blockchain. Our results indicate that the total number of orderers for given system coverage and total load are main determinants of the block linking time. We show that, given the dimensions of an area and the number of orderers, partitioning the orderers into a smaller number of tiers with more clusters will lead to lower block linking time. These observations may be used in the process of planning and dimensioning of multi-tier cluster architectures for blockchain-enabled IoT systems.**

***Index Terms*—PBFT; Internet of Things (IoT); blockchain; performance evaluation**

I. Introduction

Blockchain is emerging as the storage technology of choice for a number of Internet of Things (IoT) systems [13]. In this case, the task of the underlying consensus algorithm is to ensure global ordering of IoT data records, rather than validation which is mainly performed by IoT proxies that collect this data from sensor nodes. Permissioned character of individual orderer nodes means that well established consensus protocols such as Practical Byzantine Fault Tolerance (PBFT) [3] can be used. In our previous work, we have extended PBFT to allow multiple entry points so as to remove the limitations imposed by the traditional single-entry scenarios; in addition to security improvements, we have shown that performance improvements may be obtained as well [11].

However, the need to cover large geographic areas introduces challenges that the original PBFT and its many derivatives are not well equipped to resolve. To address this issue, we have recently proposed a multi-tier, multiple-entry PBFT architecture that consists of a hierarchical structure with multiple tiers and the bandwidth reservation protocol that allows any node to initiate and lead a consensus round [12].

In this work, we consider a multi-tier PBFT architecture comprised of a number of tiers with multiple clusters each, except for the top tier which contains a single cluster only, and focus on partitioning a given number of orderer nodes into individual clusters. Different clusters may belong to different service providers that own and operate IoT sensor devices and appropriate proxies that collect and validate IoT data, package it in blocks, and submit for linking into a global blockchain for subsequent use by various client applications. Tiered architecture allows efficient coverage of large geographical areas whilst limiting the data transmission delays and, consequently, the total delay between submitting a proposal and linking it into the global blockchain.

We analyze the performance of this system with an analytical model [12], and formulate an optimization problem to minimize the total linking delay experience by a block of IoT data submitted to the system. By comparing the results obtained with two and three tiers, we show that better results are obtained with fewer tiers that have more clusters each, which is useful in the process of planning and dimensioning the consensus architecture for IoT systems that utilize blockchain technology.

The paper is organized as follows. Related work is discussed in Section II. In Section III, we present the basic structure of multi-tier architecture in which the multiple entry point PBFT consensus at each tier is implemented with the aid of medium reservation protocol. The analytical model of the ordering service within a cluster, followed by the derivation of global block linking time, are presented in Section IV. Performance evaluation is presented in Section V. Finally, Section VI concludes the paper.

II. Related work

Application of Byzantine Fault Tolerance (BFT) algorithms in the area of permissioned blockchains has generated a lot of research interest. Most popular among them is Practical Byzantine Fault Tolerance (PBFT) algorithm [3]. PBFT allows a committee of independent orderer nodes (a leader and a number of replicas) to vote for a proposal which has to be accepted by more than two-thirds of replicas; in other words, PBFT can tolerate up to f malicious or faulty replicas out of a total of $3f + 1$ orderers. This work has inspired a lot of extensions in different directions.

A number of researchers have attempted to reduce complexity of PBFT. Assuming that orderers don't behave in Byzantine manner, voting can be streamlined or even completely avoided [8]. In a recent work [7], some ordering nodes were assigned communication and execution function so that they can talk

directly to clients [7]. Lack of communication steps is compensated with use of threshold cryptography [2]. However, in reality, some orderers or clients can be Byzantine and they can even collude.

Increasing robustness to Byzantine failures [1], [4], [15] typically focuses on combatting the impact of a faulty leader (primary). Remedies include change of primary on demand [3] or in round robin fashion [15]. In [1], function of primary and all replicas is concurrently executed on all nodes.

The use of a permanent leader/primary has another downside: it introduces significant queuing delays at the leader prior to actual processing. However, most research works focus on request processing time but ignore queuing delays.

A recent proposal [6] organizes voting nodes into two layers where the primary (leader) submits a request only to m replicas in the first layer. Those replicas achieve consensus amongst themselves, and then initiate act as sub-primaries to initiate consensus within second-tier groups of replicas with n nodes each. The decision is then passed back to the client.

Problems when dealing with larger geographical areas received much less attention [5], [9]. To address this issue, recent work has introduced PBFT system where clients can send their blocks to any orderer which can then contend to win the leader position for a single consensus round. Despite additional protocol steps, this multiple-entry scheme reduces queuing load per each orderer and reduces total access time compared to single-entry system [11]. This scheme also addresses variable propagation delays among the ordering nodes and points that total system capacity depends on the geographical coverage of the PBFT cluster [12]. This provides the foundation for the work described here which aims to further improve capacity, scalability and interoperability of the whole system.

III. System Architecture

The consensus system consists of tiers of clusters in which orderers from a cluster at a given tier communicate with orderers in nearby clusters in tiers above and below, but not with other clusters at the same tier. An example architecture with $j = 3$ tiers is schematically presented in Fig. 1. Cluster within tier i in architecture with total of j tiers is denoted with indices (i, j) where $1 \leq i \leq j$; the single top-tier cluster has the index $1, j$. System with j tiers has $N_{i,j}$ homogenous clusters per tier with $n_{i,j}$ orderers per cluster.

Orderer nodes within a cluster are fully interconnected using TCP connections that implement the broadcast medium. In addition, orderers are connected with orderers in closest lower- and higher-tier cluster. Clusters may overlap geographically without interference due to the nature of broadcast medium implemented over overlay network of TCP connections. This allows orderer nodes owned and operated by different service providers to be colocated in the same area.

IoT proxy servers act as clients that prepare blocks of transactions and submit them to the closest orderer in the lowest tier cluster. Data validation is performed by the IoT proxies and, possibly, by the orderers in the lowest tier cluster, while clusters in the higher tiers perform ordering of transaction blocks only.

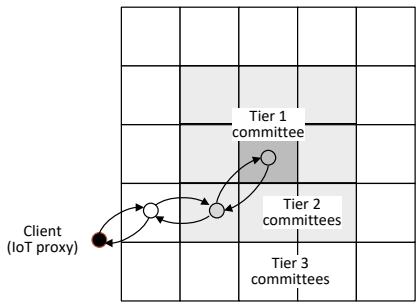


Fig. 1. Multi-tier system architecture.

The orderer which has received the block begins bandwidth reservation by broadcasting a Request to Send (RTS) message to all orderers in its cluster. If there is no competing RTS request within a time period, referred to as vulnerable period, approximately equal to maximum one-way propagation delay within the cluster [11], the orderers respond with a Clear to Send (CTS) message.

排序节点接收到区块后，通过广播RTS消息给所有同一簇中的排序节点进行带宽预约，当收到足够数量CTS消息时表明预约成功。

However, another orderer may have sent a competing RTS; when the first sender receives one or more competing RTS messages within the vulnerable period, it will enter the CSMA/CA backoff procedure similar to the one employed in IEEE 802.11 Wireless LAN Medium Access Control (MAC) standard: namely, it will wait for a random period of time before attempting to send RTS again.

当出现RTS冲突后，排序节点就会进行一个回退过程。在等待一个随机时间后，再次尝试发送RTS消息。

When the initiating orderer receives $2f$ RTS messages, it can initiate a contention-free PBFT consensus with the usual PREPREPARE, PREPARE, and COMMIT phases [3]. The PREPREPARE message contains the actual transaction block; upon receiving it, the orderers in a cluster check it and respond with a PREPARE intra-cluster broadcast. Upon receiving at least $2f + 1$ such messages, orderers (including the leader) respond with a COMMIT broadcast. When at least $2f + 1$ COMMIT messages are received, the block has been accepted and the cluster then sends it to the closest orderer in the nearest higher-tier cluster where the procedure is repeated.

The process continues until consensus is reached in the top-tier cluster. The block is then linked in the global blockchain and this decision is propagated to lower-tier clusters and, ultimately, to the IoT proxy server that submitted the block.

当底层簇中对于区块达成共识之后，簇头将发送该区块到更高层簇中，最近的排序节点，重复PBFT过程。这个过程将一直进行下去，直到共识达到最顶层簇。随后将该区块链接到全局区块链上，并将结果传输到下层簇，最终达到物联网代理服务器。

Bandwidth reservation is utilized at each cluster individually, regardless of the tier in which it resides. In this manner, IoT proxies can always submit their request to the nearest orderer in the lowest-tier cluster; block load is shared among all clusters in a given tier and among all orderers in a given cluster; and, last but not least, failure or Byzantine behavior of any orderer node cannot stall the operation of the consensus architecture.

代理收集交易并验证交易，之后打包成区块提交到最底层的Order，Order接收到后，也会验证交易。

IV. Modeling the Multi-tier PBFT Architecture

Analytical model for the tiered architecture consists of separate submodels for one way propagation delay over the

edge of the cluster, bandwidth reservation service and block insertion service without contention, block insertion service under CSMA/CA contention, waiting time in the orderer's queue before block reaches head of the queue, and interconnection model of multiple clusters. In addition, we need an optimization model over the whole system.

For simplicity, we assume that clusters are square shaped with a side of $0 \leq x, y \leq L_{i,j}$ where $L_{i,j}$ denotes physical dimension of the cluster at level (i, j), with nodes uniformly distributed over the area of the cluster. Given the propagation delays over TCP connections and queuing delays, it may be more useful to consider one-way propagation delays over the cluster edge instead of actual physical dimensions. If we denote the one-way propagation delay in tier $1 \leq i \leq j$ as $D_{i,j}$, the one-way delay through all tiers is $D = \sum_{i=1}^{j} D_{i,j}$, while the round-trip time through all tiers is limited by $2D$.

Record insertion time without contention: Record insertion time without contention consists of time to perform bandwidth reservation using RTS/CTS signalling and time to perform PREPREPARE, PREPARE and COMMIT voting phases. Probability distribution of maximum one-way delay was derived in [12] under the assumption of uniform node placement within the area of the cluster. It depends on the one way propagation delay over the cluster edge $D_{i,j}$ and also on the number of orderers in the cluster $n_{i,j}$.

In this model we approximated the upper bound for the RTS/CTS signalling time with three maximum one-way delays within the cluster, which is also the time required for the three PBFT phases in the cluster. However, when the number of tiers increases, the computation quickly becomes intractable and some simplification is necessary. Based on the model from [12], mean maximum one-way delay in the cluster (i, j) may be approximated by

$$\mathcal{D}_{i,j} = 0.95 D_{i,j} - 0.5 D_{i,j} e^{-0.26(n_{i,j}-1)} \tag{1}$$

Coefficient of variation of maximum one-way delay decreases with the number of orderers: its largest value is around 0.4 for 4 orderers, but rapidly decreases to 0.2 and 0.07 for 7 and 10 orderers, respectively.

RTS/CTS signalling completion and the corresponding timeout can be modeled by a Gamma probability distribution with mean $\overline{T_{cr,i,j}} = 3\mathcal{D}_{i,j}$ and coefficient of variation $\vartheta_{i,j} < 0.04$ or smaller. Laplace-Stieltjes transform (LST) of maximum time to complete RTS/CTS signalling is $T^*_{cr,i,j}(s) = (1 + br_{i,j}s)^{-cr_{i,j}}$, where $br_{i,j} = \vartheta_{i,j} 3\mathcal{D}_{i,j}$ and $cr = 1/\vartheta_{i,j}$.

By the same token, we can approximate maximum total record insertion time in a cluster without contention as the sum of six maximum one-way delays, i.e., $\overline{T_{it,i,j}} = 6\mathcal{D}_{i,j}$. Squared coefficient of variation of this time $\mathcal{V}_{i,j}$ is less than 0.01 for 4 orderers in the cluster, and even smaller for larger number of orderers. The corresponding probability distribution can be approximated with a Gamma distribution with parameters $\beta_{i,j} = \mathcal{V}_{i,j} 6\mathcal{D}_{i,j}$ and $\gamma_{i,j} = 1/\mathcal{V}_{i,j}$ giving the LST of distribution of record insertion time without contention as $T^*_{it,i,j}(s) = (1 + s\beta_{i,j})^{-\gamma_{i,j}}$.

Simplified model for request contention: The CSMA/CA protocol is modeled by extending the CSMA/CA model from [12] to address the variability of block insertion time under variable cluster dimensions.

Backoff slot $\delta_{i,j}$ for contention resolution among orderers was set according to the maximum one-way delay among orderers within the cluster in a given tier.

Data rate for headers and payload in all clusters was R bps giving the number of bits that can fit into a single backoff period as $\Delta_{i,j} = R\delta_{i,j}$. Let N_h denote the number of bits needed for all headers of protocol stack in all clusters, while RTS and CTS messages including headers) and request payload take rpl and rs bits, respectively. LST of the time to transmit the request (in slots) is, then, $T^*_{blk,i,j}(s) = e^{-s(N_h+rpl)/\Delta_{i,j}}$.

To arrive at total block insertion time, we need to model the round trip time needed to communicate a block to an orderer within a cluster in the next layer and to receive the acknowledgement. Assuming that this time to reach orderer in the next tier is exponentially distributed with mean $D_{i,j}$ it has LST $T^*_{D,i,j}(s) = \frac{1}{1+sD_{i,j}/\delta_{i,j}}$ when $i < j$, and $T^*_{D,j,j}(s) = exp(0)$, otherwise.

Consequently, when expressed in backoff slots per cluster, the lengths of the RTS/CTS and request are, respectively:

$$rts_{i,j} = cts_{i,j} = \frac{N_h + rs}{\Delta_{i,j}} \tag{2}$$

$$l_{i,j} = \frac{N_h + rpl}{\Delta_{i,j}} \tag{3}$$

LST for the duration of total insertion time without contention in cluster (i, j) is $Sr^*_{i,j}(s) = e^{-s(rts_{i,j}+cts_{i,j})} T^*_{it,i,j}(s)$, while LST for the duration of unsuccessful bandwidth reservation in cluster (i, j) is $Cr^*_{i,j}(s) = e^{-s(rts_{i,j}+cts_{i,j})} T^*_{cr,i,j}(s)$.

If the orderer access probability within the cluster is denoted as $\tau_{i,j}$, and given the total request arrival rate $\lambda_{i,j}$ per cluster (i, j) and number of orderers in cluster $n_{i,j}$, probability that node will have to freeze its backoff counter due to successful transmission from other orderer(s) is $Ps_{i,j} = (n_{i,j}-1)\tau_{i,j}(1-\tau_{i,j})^{n_{i,j}-2}$, while probability of freezing the backoff counter due to collision among other orderer RTS messages is $Pc_{i,j} = 1 - (1 - \tau_{i,j})^{n_{i,j}-1} - (n_{i,j} - 1)\tau_{i,j}(1 - \tau_{i,j})^{n_{i,j}-2}$. Finally, probability of RTS collision experienced by the target orderer is $Pr_{i,j} = 1 - (1 - \tau_{i,j})^{n_{i,j}-1}$.

The LST for the duration of the period between the backoff countdowns within cluster i, j, expressed in backoff slots, is

$$\beta^*_{i,j}(s) = \frac{(1 - Pr_{i,j})e^{-s}}{1 - (Pc_{i,j} Cr^*_{i,j}(s) + Ps_{i,j} Sr^*_{i,j}(s)} \tag{4}$$

Backoff process is conducted over uniformly distributed random periods of time in the range $0 \,..\, W_k - 1$ where $0 \leq k \leq m$ is the index of the backoff attempt. Value W_k is known as backoff window and it is doubled in each subsequent attempt until backoff limit is reached, i.e. $W_k = 2^k W_0$, for

$k \leq m$, and $W_k = 2^m W_0$ for $k > m$. LST for the duration of k-th backoff attempt is

$$B^*_{k,i,j}(s) = \frac{1 - \beta^*_{i,j}(s)^{W_k}}{W_k \left(1 - \beta^*_{i,j}(s)\right)} \tag{5}$$

The total LST for the duration of block insertion time under contention can be calculated as

$$T^*_{i,j}(s) = \frac{\sum_{k=1}^{m} \prod_{l=0}^{k-1} B^*_{k,i,j}(s)(Pr_{i,j}Cr^*_{i,j}(s))^{k-1}(1-Pr_{i,j})Sr^*_{i,j}(s)}{\sum_{k=1}^{m} \prod_{l=0}^{k-1} B^*_{k,i,j}(0)(Pr_{i,j}Cr^*_{i,j}(0))^{k-1}(1-Pr_{i,j})Sr^*_{i,j}(0)} \tag{6}$$

with mean value of $\overline{T_{i,j}} = -\frac{dT^*_{i,j}(0)}{ds}$.

Offered load for an individual orderer can be calculated as

$$\rho_{i,j} = \frac{\lambda_{i,j}}{n_{i,j}} \overline{T_{i,j}} \tag{7}$$

By applying Discrete Time Markov Chain [12] we can obtain expression for orderer's medium access probability $\tau_{i,j}$, and then solve it together with with (7) to obtain $\tau_{i,j}$ and $\overline{T_{i,j}}$.

Global block linking time: An orderer request queue can be modeled as a M/G/1 system with FCFS discipline, hence the LST for block insertion time within the cluster [14] may be obtained as

$$S^*_{i,j}(s) = \frac{s(1-\rho_{i,j})T^*_{i,j}(s)}{s - \lambda_{i,j}/n_{i,j} + (\lambda_{i,j}/n_{i,j})T^*_{i,j}(s)} \tag{8}$$

with mean value of $\overline{S_{i,j}} = \overline{T_{i,j}} + t\frac{\lambda_{i,j}T^{(2)}_{i,j}}{2n_{i,j}(1-\rho_{i,j})}$, where $T^{(2)}_{i,j}$ is second non-central moment of ordering request service time.

A given block must be inserted in all j tier blockchains until it is linked in the global blockchain for given area. To calculate the time to do that we need to take into account block and acknowledgement transmission and propagation times. This gives LST for the total block linking time as

$$T^*_{tot,j}(s) = \left(\prod_{i=1}^{j} S^*_{i,j}(s) T^*_{Sc,i,j}(s)^2\right) T^*_{blk,i,j}(s)^{j-1} \tag{9}$$

Time periods are expressed using backoff slots in individual clusters; upon transformation to absolute time, mean value of the total round trip time can be expressed as

$$\overline{T_{tot,j}} = \sum_{i=1}^{j} \overline{T_{i,j}}\delta_{i,j} + \sum_{i=1}^{j} \frac{\lambda_{i,j}T^{(2)}_{i,j}}{2n_{i,j}(1-\rho_{i,j})}\delta_{i,j} + 2D + (j-1)\frac{(N_h + rpl)}{R} \tag{10}$$

V. Performance Evaluation

We can now find the optimal distribution of a constant number of orderers into clusters at different tiers, 'optimal' referring to the configuration that results in minimum block insertion time as per (10). We assume that the total area is a square with a side that corresponds to a given one-way propagation delay D; that the system-wide request arrival rate λ is uniformly distributed over all lowest-tier clusters; and that all clusters in a given tier have the same size, expressed through their respective delays $D_{i,j}$ and corresponding duration of backoff period $\delta_{i,j}$. The optimization problem can be expressed as

$$\begin{aligned} &\text{minimize} && \overline{T_{tot,j}} \\ &\text{subject to:} && \sum_{i=1}^{j} N_{i,j}n_{i,j} = C \\ & && \sum_{i=1}^{j} D_{i,j} = D \end{aligned} \tag{11}$$

We have solved it using Maple 13 by Maplesoft, Inc., Waterloo, ON. We have assumed that $D_{1,j} < D_{2,j} \ldots < D_{j,j}$, since each cluster (i,j), $1 \leq i < j$ receives a multiple of the output of lower-tier clusters $(i+1,j)$, and that the number of clusters at higher tiers gets progressively smaller, i.e., $N_{i,j} < N_{i+1,j}, 1 \leq i < j$.

The total area was set to correspond to maximum one-way delay of $D = 25$ms along each edge of a square. This value corresponds to the distance of well over a thousand kilometers, depending on the number of intermediate routers. We have considered the architecture with two and three layers only, with the total number of orderers varied from 80 to 184 for both low- and high load systems.

As explained above, the system had a single top-tier cluster with square shape and one-way propagation delay along each side of $D_{1,j}$, with a total of $n_{1,j}$ orderers. The number of clusters, the number of orderers within a cluster, and the one-way propagation delay were determined accordingly.

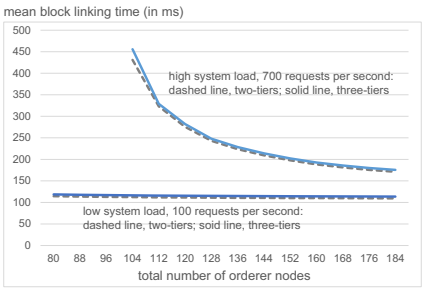
As each cluster must have $3f + 1$ orderers to achieve resilience to Byzantine faults [3], the number of orderers in each cluster and their total number have to be varied in discrete steps.

We have adopted two values for system-wide request arrival rate of $\lambda = 100$ and 700 requests per second as representatives of low and high load, respectively. Size of submitted blocks was rpl =1Kbyte. Payload of RTS and CTS messages was $rs = 160$ bits and overall header size was $N_h = 800$ bits. Throughput per TCP connection between any two orderers is 2Mbps.

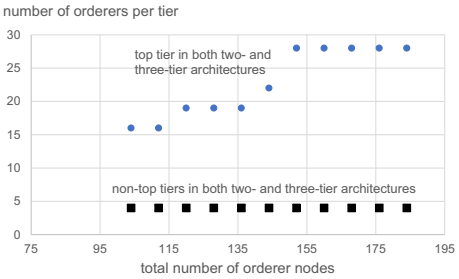
Minimum contention window was set to $W_{min} = 4$. Backoff slot for contention resolution among orderers was set to match maximum one-way delay among orderers within the cluster.

Fig. 2(a) shows minimum values for mean linking time obtained under variable number of orderers, with system request arrival rate as the parameter. Values for the two- and three-tier architecture are shown with dashed and solid lines, respectively.

Under low load conditions the mean linking time is in the range of 110 to 120ms, nearly independent of the total number of orderers. The difference between the curves is slightly above 4ms due to an additional block transmission time needed in the three-tier architecture. Under high load, minimal values of

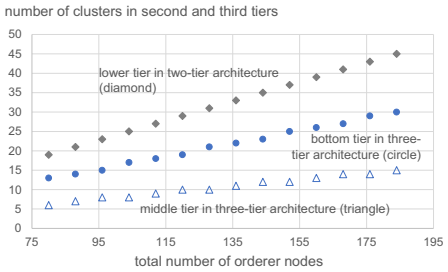


(a) Mean linking time vs. total number of orderers.

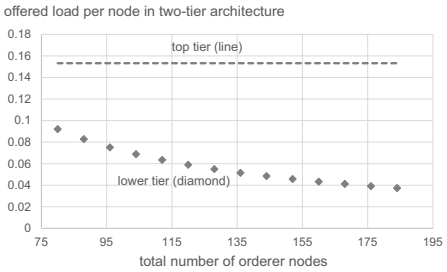


(b) Optimum number of orderers per tier.

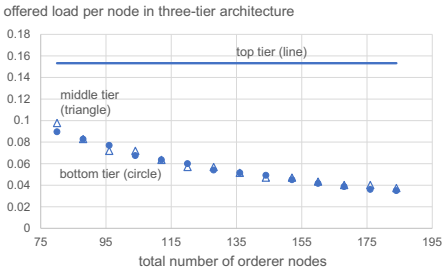
Fig. 2. Pertaining to the performance of multi-tier PBFT architectures.



(a) Number of clusters in non-top tiers leading to optimum performance.



(b) Offered load per node in two-tier architecture.



(c) Offered load per node in three-tier architecture.

Fig. 3. PBFT performance at low load (i.e., at request arrival rate 100 requests per second.

mean linking time range from 170 to 430ms, for the two-tier architecture, and from 175 to 455ms, for the three-tier one.

Mean linking time drops with the increase in the total number of orderers, partly on account of the reduction of per-cluster and, consequently, per-orderer queuing load. The highest values of mean linking time correspond to the lowest total number of orderers (104), at which point the offered load per orderer exceeds the value of about 0.8, as will be shown below.

In both cases, the curves for two- and three-tier architectures are very close, although the former does exhibit a slight edge (lower linking time), under high load.

Furthermore, the top-tier cluster should cover as small area as possible so as to reduce the maximum one-way delay in that cluster and, consequently, minimize the time to achieve both bandwidth reservation and consensus in that cluster. In the two-tier case, optimal results were obtained for $D_{1,2} \approx 1$ms and $D_{2,2} = 12$ ms; in the three-tier case, with $D_{1,3} = 1$ms, $D_{2,3}$ in the range 2 to 4ms, and $D_{3,3}$ in the range 8 to 10 ms.

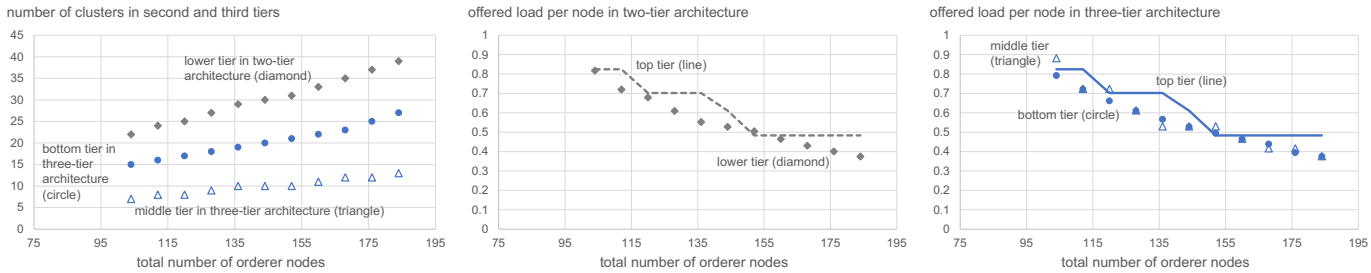
The impact of high load can be seen in the optimum number of orderer nodes in clusters at different tiers, shown in Fig. 2(b). Namely, the optimum number of orderer nodes in all lower-tier clusters is still four, which keep the contention at the minimum. At the top tier in both two-and three-tier architectures, the situation is different: the optimum number of orderer nodes in the top cluster is as high as can be, ranging from 16 to 28, due to the need to handle high system load.

Performance under low load: The diagrams in Fig. 3 describe further system parameters under low load (i.e., system-wide request arrival rate of 100 requests per second): Fig. 3(a) shows the optimum number of clusters for low- and middle-tier clusters (remember that both architectures have a single top-tier cluster), while Figs. 3(b) and 3(c) show mean offered load per node in two- and three-tier architectures, respectively.

The number of non-top-tier clusters slowly increases with the total number of orderers. In the two-tier architecture, the number of clusters increases from 19 (for a total of 80 orderers) to 45, when the total number of orderers is 184. In the same range of the number of orderers, the number of clusters in the middle and low tiers in the three-tier architecture increase from 6 and 13, to 15 and 30, respectively.

Optimum number of orderers per cluster in all tiers except the top one is only four, regardless of the number of tiers and the tier to which the clusters belong. Keeping the number of orderers low ensures minimum contention in the bandwidth reservation phase. This is corroborated by the corresponding lines for low system load in Fig. 2(a).

Despite their small number, four nodes in the top-tier cluster are more than sufficient to sustain the total system load – each node has an offered load of about 0.155, as can be seen from the corresponding dashed and solid lines in Figs. 3(b) and 3(c) corresponding to two- and three-tier architectures, respectively. In lower-tier clusters, offered load per orderer slowly decreases with the total number of orderers, as more



(a) Number of clusters in non-top tiers leading to optimum performance.

(b) Offered load per node in two-tier architecture.

(c) Offered load per node in three-tier architecture.

Fig. 4. PBFT performance at high load (i.e., at request arrival rate 700 requests per second.

orderers are available to carry the constant request load.

Performance under high load: The diagrams in Fig. 4 describe system performance under high load of 700 requests per second. The optimal number of clusters in different tiers, shown in Fig. 4(a) behaves similar to the corresponding values under low load, except that the number of clusters in the low tier of two-tier architecture now ranges from 22 to 39, while the middle- and low-tier of the three tier architecture use 7 and 15, up to 13 and 27 clusters, respectively.

This is confirmed by the diagrams in Figs. 4(b) and 4(c) which show the offered load per node in two- and three-tier architectures, respectively. As can be seen, individual orderer nodes carry an offered load as high as 0.8 when the total number of orderer nodes in the system is at the lowest value of 104. From that point on, the offered load per orderer decreases as more orderers are available. In the lower tiers, the decrease is close to linear; in the top tier, the offered load follows a decreasing stepwise pattern.

Optimization of the distribution of orderers manages to maintain the maximum offered load at approximately the same value for orderers in all tiers, which is why some of the data points overlap. Also, the offered load per node in the top-tier cluster is virtually the same in both two- and three-tier architectures due to the same number of orderer nodes in that cluster.

Final remarks: As can be seen, the two- and three-tier architectures exhibit similar performance level in terms of mean time needed to link a data block into the global blockchain. The three-tier architecture uses the additional tier to reduce the cluster coverage, which reduces the service time, but at the expense of an extra transmission time which turns out to cancel the benefits of reduced coverage. As the result, the two-tier architecture offers marginally better performance as the load increases. Either way, our results unequivocally show that best performance is obtained when the top-tier cluster covers the minimum area, which leads to minimum transmission delays, and has the most orderers, which reduces consensus time.

VI. Conclusions

We have presented and evaluated a tiered multiple-entry permissioned blockchain architecture suitable for wide area IoT systems. This architecture matches the needs of individual service providers to maintain their separate blockchains yet to have their data linked in a global ledger. We have modeled the multi-tier architecture to evaluate its performance. Our results show that the performance is mostly affected by the total number of orderers, as more orderers lead to better performance at high loads, and that the area covered by the top cluster (which carries the entire system load) should be as small as possible. Overall, the multi-tiered architecture shows it is capable of sustaining high system load in a wide area setting.

References

[1] P.-L. Aublin, S. Ben Mokhtar, and V. Quéma. RBFT: Redundant Byzantine fault tolerance. *IEEE 33rd Int. Conf. Distributed Comput. Syst.*, pages 297–306. IEEE, 2013.

[2] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Int. Conf. the Theory and Application of Cryptology and Information Security*, pages 514–532, 2001.

[3] M. Castro and B. Liskov. Practical Byzantine fault tolerance. *OSDI: Symp. Operating Systems Design and Implementation*, New Orleans, LA, Feb. 1999.

[4] A. Clement *et al.* Making byzantine fault tolerant systems tolerate byzantine faults. *6th USENIX Symp. Networked Systems Design and Implementation*, Boston, MA, 2009.

[5] G. S. Veronese *et al.* EBAWA: Efficient Byzantine Agreement for Wide-Area Networks. *IEEE 12th Int. Symp. High Assurance Systems Eng.*, pages 10–19, 2010.

[6] W. Li *et al.* A scalable multi-layer PBFT consensus for blockchain. *IEEE Trans. Parallel and Distributed Systems*, 32(5):1146–1160, 2021.

[7] G. G. Gueta *et al.* SBFT: a scalable and decentralized trust infrastructure. *49th Ann. IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN)*, pages 568–580. IEEE, 2019.

[8] R. Kotla *et al.* Zyzzyva: Speculative Byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27(4), Jan. 2010.

[9] A. Miller *et al.* The honey badger of BFT protocols. *ACM SIGSAC Conf. Computer and Communications Security*, pages 31–42, 2016.

[10] J. Mišić, V. B. Mišić, and X. Chang. Comparison of single- and multiple entry point PBFT for IoT blockchain systems. *IEEE 92nd Vehicular Technology Conf. VTC2020-Fall*, Victoria, BC, Oct. 2020.

[11] J. Mišić, V. B. Mišić, and X. Chang. Adapting PBFT for use with blockchain-enabled IoT systems. *IEEE Trans. Vehicular Technology*, 2021.

[12] O. Novo. Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE Internet of Things J.*, 5(2):1184–1195, 2018.

[13] H. Takagi. *Queueing Analysis*, volume 1: Vacation and Priority Systems. North-Holland, Amsterdam, The Netherlands, 1991.

[14] G. S. Veronese *et al.* Spin one's wheels? Byzantine fault tolerance with a spinning primary. *28th IEEE Int. Symp. Reliable Distributed Syst.*, pages 135–144, 2009.