Optimal endorsement for network-wide distributed blockchains

Optimal endorsement for network-wide distributed blockchains

Iman Lotfimahyari, Paolo Giaccone
Dipartimento di Elettronica e Telecomunicazioni - Politecnico di Torino - Torino, Italy
e-mail: firstname.lastname@polito.it

Blockchains offer trust and immutability in non-trusted environments, but most are not fast enough for latency-sensitive applications. Hyperledger Fabric (HF) is a common enterprise-level platform that is being offered as Blockchain-as-a-Service (BaaS) by cloud providers. In HF, every new transaction requires a preliminary *endorsement* by multiple mutually untrusted parties called organizations, which contributes to the delay in storing the transaction in the blockchain. The *endorsement policy* is specific to each application and defines the required approvals by the *endorser peers* (EPs) of the involved organizations.

In this paper, given an input *endorsement policy*, we studied the optimal choice to distribute the endorsement requests to the proper EPs. We proposed the OPEN algorithm, devised to minimize the latency due to both network delays and the processing times at the EPs. By extensive simulations, we showed that OPEN can reduce the *endorsement latency* up to 70% compared to the state-of-the-art solution and approximated well the introduced optimal policies while offering a negligible implementation overhead compared to them.

***Index Terms*—Blockchains, Hyperledger Fabric, Endorsement policy**

I. Introduction

Nowadays, blockchains have become more and more relevant in many ICT applications, since bring trust between different entities where trust is either nonexistent or unproven. They can improve security and privacy while offering a decentralized structure. The provided immutability brings visibility and traceability, beneficial for ICT applications, such as banking, supply-chain, IoT, healthcare, and energy sectors [1], [2], [3].

A blockchain is public if it is open to everyone to read otherwise it is private. But, if a node needs permission to participate in validating transactions, then the blockchain is permissioned otherwise it is permissionless [4]. In contrast to public permissionless blockchains like Bitcoin, many enterprise applications require performance that permissionless blockchains are unable to deliver. Furthermore, many use cases necessitate knowing the identity of the participants, such as in financial transactions where notary service regulations must be followed. Private permissioned blockchains, such as Hyperledger Fabric (HF) [5] and Corda [6], meet such requirements. In HF, a transaction must be endorsed (i.e., approved) by the organizations constituting the blockchain, according to a specified endorsement policy. This guarantees a mutual agreement between non-trusted parties, similarly, in the physical world, to a receipt declaring an asset transfer between two parties, signed by both parties.

HF uses an architecture called Execute-Order-Validate for transactions, enabling the definition of endorsement policies. During the execution phase, the client sends the transaction to some Endorser Peers (EPs), based on the user's specified endorsement policy. Each EP processes the transaction by only simulating it without applying the results on the blockchain. The simulation result, denoted as "endorsement", is signed by the EP and returned to the client. Finally, if the endorsement policy is satisfied, the signed and endorsed proposal of the transaction will be sent to the blockchain nodes to be stored.

The endorsement delay experienced by a client is affected mainly by two components: i) the network delay between the client and the EPs and ii) the processing delay at each EP. The network delay mainly depends on the network congestion and the propagation delays, whereas the processing delay depends on both the CPU capability and the computation load of each EP. Because network and processing delays are time-varying and hence difficult to predict, optimally selecting EPs is hard. Note that a selection algorithm choosing just the best EP based on the minimum experienced delays will concentrate the endorsement requests to the same EPs, increasing the network congestion and the processing load, thus increasing the overall endorsement delays. In this work, we propose an *optimal EP selection policy* minimizing the endorsement delays. The main idea is to send redundant endorsement requests to multiple EPs. The adopted spatial diversity increases the chance of having the best EPs among the selected ones. The benefit of the proposed approach can be captured by a simple queueing model in which a task is sent in parallel to multiple servers, each with its queueing system, to minimize task completion time.

In this paper, our novel contributions are as follows.

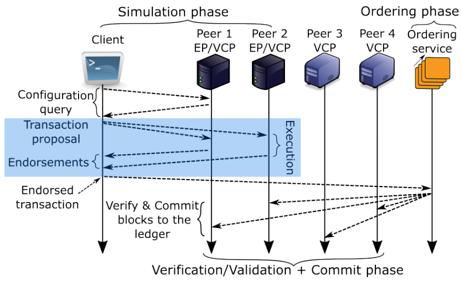


Fig. 1: Transaction processing phases in HF highlighting all the message interactions between the involved entities

(i) We highlight the role of the network and processing delays in the overall endorsement delay. (ii) We refer to a simple analytical model, based on the classical theory of queueing systems, to evaluate the effect of redundancy in selecting the EPs and to compute the optimal number of EPs. (iii) We propose an optimization approach denoted as *OPtimal ENdorsement (OPEN)* based on the analytical results, leveraging the history of endorsement delays. (iv) We demonstrate through extensive simulations that OPEN outperforms the state-of-the-art solution and accurately approximates other optimal policies while having a much lower implementation overhead compared to them.

The rest of this paper is structured as follows. Sec. II describes the HF architecture and then focuses on the endorsement phase delay by introducing the network model and the EP selection problem. Sec. III explains a simple analytical model, derived from classical results on queueing theory, to find the optimal number of EPs in a simplified scenario. In Sec. IV, we propose an EP selection algorithm based on the optimal replication factor computed analytically in Sec. III, able to operate in a generic scenario. In Sec. V, we assess by simulation the performance of our proposed approach and compare it with the alternatives proposed and with the state-of-the-art solution. In Sec. VI we discuss the related work. Finally, we draw our conclusions in Sec. VII.

II. Hyperledger Fabric architecture and endorsement

A. Hyperledger Fabric architecture and protocol

The Execute-Order-Validate approach enables the simulation of the transactions before the agreement of the participants on recording the results in the Hyperledger Fabric blockchain. We describe the role of the entities which are participating in the simulation phase of Fig. 1.

The *client* is responsible for preparing the transaction proposal of the users' transactions and sending it to the Endorser Peers (defined below) based on the specified endorsement policy. If the client receives enough endorsements before a specific time-out, it forwards them to the ordering service; otherwise, the client can retransmit the same proposal in the hope of receiving enough endorsements in time.

The *Peer* is the element responsible for the following tasks. The *Endorser Peer* (EP) simulates/executes the transaction received from the client application, based on the current values of the world state. The *Verifier/Committer* (VCP) receives a block of simulated transactions from the ordering service and verifies their legitimacy to mark them as validated or invalidated. Then it appends the verified block to the blockchain, comprising all the transactions (validated or invalidated). To update its copy of the ledger, an EP is typically a VCP at the same time. The peers are owned by various *organizations* that are blockchain members. An organization can be as small as individuals or as large as a multi-national corporation.

The *endorsement policy* defines the logical conditions to validate a transaction in terms of the EPs on a channel that must execute a transaction proposal. In Sec. II-B, we will describe in detail the representation of the endorsement policy. The definition of an endorsement policy is at the organizational level, which means any EP of that organization can represent that organization in the endorsement policy. A transaction should pass three phases to be stored in the blockchain, as shown in Fig. 1.

B. Standard form of an endorsement policy

HF provides a very flexible way to define an endorsement policy. We will show that any endorsement policy, despite its complexity, can be reduced to a standard form. In HF, the definition of an endorsement policy is based on a syntax that allows the operators "AND", "OR" and "k-OutOf" to be applied to a set of organizations and nested expressions [7]. In particular, the operator "k-OutOf-E" returns true whenever at least k expressions within set E are satisfied. Despite the complexity of the policy expression, we prove that the following proposition holds:

Proposition 1. *Any endorsement policy obtained by combining arbitrarily "AND", "OR", and "OutOf" operators is equivalent to the policy:*

$$OR(St_1, St_2, ...) \tag{1}$$

where each St_i is either a single organization or the conjunction ("AND") of different organizations.

Proof. In the case of expressions based on only "AND" and "OR" operators, thanks to the distribution principle in logic expressions, we can transform the original expression into the target form (1). In the case of "k-OutOf$(e_1, e_2, \ldots, e_m)$ operator, where e_i is a single expression, by definition this holds:

$$k\text{-OutOf}(e_1, e_2, \ldots, e_m) = \text{OR}(\{\text{AND}(E)\}_{E \in \Omega})$$

being Ω the set of all $\binom{m}{k}$ combinations of k expressions from the set of m. Now, since any expression with the "OutOf" operator is equivalent to one with only "AND" and "OR", by following the previous reasoning, such expression can be reduced to the expression (1). □

The policy, defined at the organization level, must be mapped into a policy defined at the EP level since the endorsement requests should be sent to the proper EPs. So, getting the endorsement from a specific organization requires receiving it from *any* of its EPs, which is equivalent to the policy 1-OutOf$(p_1, p_2, \ldots)$, where p_i are the EPs within the organization. Revisiting Proposition 1 applied at the policy expression at the EP level, we can claim:

Proposition 2. *Any endorsement policy defined at the organization level can be expanded into an endorsement policy defined at the EP level as follows:*

$$OR(St_1^{'}, St_2^{'}, ...) \tag{2}$$

where each of $St_i^{'}$ is either a single EP or the conjunction ("AND") of different EPs.

The result of Proposition 1 allows investigating only one standard form of endorsement expression, independently from the original expression complexity. Now, by using Proposition 2, we will have the endorsement expression extended at the EP level. At this level, the final endorsement will be just in the form of the OR between the conjunction ("AND") of different EPs of different organizations, as in (2).

For example, consider a scenario with three organizations and two EPs in each of them. If the endorsement policy is "2-OutOf(o_1, o_2, o_3)", we can rewrite it as:

$$\begin{aligned} &\text{2-OutOf}(o_1, o_2, o_3) = \\ &\qquad \text{OR}(\{\text{AND}(p_{ij}, p_{i'j'}), \forall i, \forall i' \neq i, \forall j, \forall j'\}) \end{aligned} \tag{3}$$

where o_i is organization i, and p_{ij} is the EP j of organization i. The expanded version in (3) lists all the possible combinations of the EPs that can satisfy the endorsement policy according to the standard form.

C. Endorser peer (EP) selection algorithm

In our work we focus on the EP selection algorithm, starting from the standard form of the endorsement policy. The *endorsement delay* is the amount of time the client waits, from sending the endorsement request until receiving the first endorsement reply that satisfies the endorsement policy. The response delay from an EP is the sum of two components: the network delay and the processing delay at the EP. The *network delay* depends on the propagation delay and the queueing delay along the path to the EP, which is affected by the time-variant congestion conditions. The overall *processing delay* depends on the *queuing* at the EP before being served and the *computation time* at the EP, which depends on the CPU speed and the instantaneous CPU load and resource contentions. Because the standard form of any endorsement policy comprises an overall "OR" operator, as in (2), the endorsement latency corresponds to the *minimum* delay to get a valid statement. Also, each statement is based on an "AND" operator between EPs, so the delay of each statement depends on the *maximum* response delay of all EPs included in a statement. In summary, the endorsement latency depends on the "fastest" group of EPs forming a statement, while the delay of each group depends on the "slowest" EP within the group.

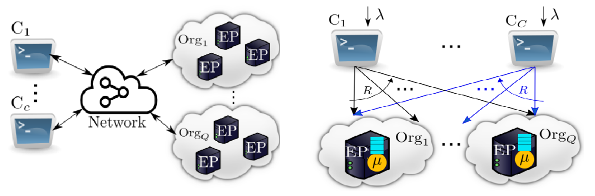
D. System model for the endorsement phase

Without loss of generality, we consider a fixed network topology connecting C clients with Q organizations, each of them with a generic network connecting the internal EPs, depicted in Fig. 2a. We assume that all nodes in the system are always available, the routing is fixed, and the links have enough bandwidth to prevent network congestion caused by the endorsement protocol. Thanks to the service discovery process in HF, we consider only the most updated EPs.

III. Background on optimal replication in queueing systems

Now, we discuss an analytical model to compute the optimal number of EPs for each transaction, derived from classical results on task replication in a queueing system, as explained in Sec. VI. For the sake of readability, we report the adopted notation in Table I.

We consider a simplified model as shown in Fig. 2b, with one EP in each organization. We assume 1-OutOf-Q as the endorsement policy, which corresponds to OR$(p_1, p_2, \ldots, p_Q)$ in its standard form. For now, we neglect the network delays and concentrate just on processing delays. We suppose each client generates endorsement requests according to a Poisson process with rate λ. Each client selects at random R EPs to send the endorsement request. R will be denoted in the



(a) General network model (b) R requests for 1-OutOf-Q

Fig. 2: Network model and the endorsement policy sending each request to R organizations/peers in parallel.

TABLE I: Notation

C	number of clients
Q	number of organizations
p	endorser peer (EP)
λ	arrival rate of new transactions to each client
μ	inverse of computation time for the EP server
U	utilization factor in each EP server
R	redundancy factor
W_i	waiting time needed for the ith request to be served
S_i	inter-arrival time between the ordered version of $\{W_i\}_i$
γ	normalized load factor for the worst case $R = Q$
$\hat{R}_k$	optimal R for policy k-OutOf-Q
L_k	endorsement latency for policy k-OutOf-Q
$\mathcal{P}$	set of all available EPs
$\mathcal{P}_e$	set of selected EPs
$\mathcal{P}_e^{\text{old}}$	set of previously selected EPs
T	probe sampling period
TX^n	transaction with local sequence number n
x_p^k	endorsement latency of TX^k for peer p
t_p^{resp}	virtual response delay of a peer for the new TX
t_p^{busy}	virtual time at which EP p is not busy anymore
τ_p^{proc}	processing delay for the current endorsement request
τ_p^{net}	network delay for EP p
τ_p^{queue}	queueing time experienced by the TX at EP p
d_{cp}	the network delay between client c and EP p

following as *redundancy factor*. To model the processing time variability at the EP, we assume an exponentially distributed processing time with an average $1/\mu$, coherently with past works [8], [9]. Thus each EP can be modeled as an M/M/1[1] queue with arrival rate $\lambda RC/Q$ and service rate μ. We define the utilization factor for each EP as $U = \lambda RC/\mu Q$. Thus, for the request traffic to be sustainable, $U < 1$ and the endorsement request arrival rate must satisfy $\lambda < \mu Q/RC$. We now claim:

Proposition 3. *Under a sustainable arrival rate of endorsement requests and a random selection policy with R EPs, according to the endorsement policy 1-OutOf-Q, it holds for the endorsement latency L_1:*

$$E[L_1] = \frac{1}{\mu - \dfrac{\lambda RC}{Q}} \left(\frac{1}{R}\right) \qquad R \in [1, \ldots, Q] \tag{4}$$

[1]In classical queueing theory, an M/M/1 queue has a single server, arrivals follow a Poisson process and service times are exponentially distributed [10].

Proof. From Fig. 2b, let λ' be the average incoming rate of the requests for the queue of each EP such that: $\lambda' = \lambda RC/Q$ We define W_i as the waiting time of a request to be served at the ith EP, which is the sum of queuing time and the serving time of the request in the ith EP. From M/M/1 well-known properties [10], W_i are i.i.d. and exponentially distributed with mean: $E[W_i] = 1/(\mu - \lambda')$. Observe that: $L_1 = \min(W_1, W_2, \ldots, W_R)$ where W_i are i.i.d.. From basic properties of the exponential distribution, L_1 is exponentially distributed with mean:

$$E[L_1] = E[W_i]/R \tag{5}$$

and finally get (4). □

By computing the first derivative of (4) with respect to R, we can prove the following:

Proposition 4. *Let $\hat{R}_1$ be the optimal value of R that minimizes $E[L_1]$ for the policy 1-OutOf-Q.*

$$\hat{R}_1 = \frac{\mu Q}{2\lambda C} \tag{6}$$

In summary, the optimal number of EPs changes with λ. For low arrival rates, R must be large to exploit the spatial diversity, without incurring additional overhead in the processing times. For high arrival rates, conversely, R is small to reduce the load on the EPs. Notably, for the sake of readability, we omitted from (6) the clipping to the interval $[1, Q]$ and the rounding procedure to find the optimal integer value of R. We can now extend the result of Proposition 3 to a generic OutOf policy.

Proposition 5. *Under a sustainable arrival rate of endorsement requests and a random selection policy with R EPs, according to the endorsement policy k-OutOf-Q, it holds for the endorsement latency L_k, for any $R \in [k, \ldots, Q]$:*

$$E[L_k] = \frac{1}{\mu - \dfrac{\lambda CR}{Q}} \left(\sum_{i=0}^{k-1} \frac{1}{R-i}\right) \tag{7}$$

Proof. Using the same definition of W_i as adopted in the proof of Proposition 3, we can define L_k as the endorsement latency for the policy k-OutOf-Q. Now L_k can be computed as the kth order statistic as follows, $L_k = (W_1, W_2, \ldots, W_R)_{(k)}$, recalling the fact that W_i are i.i.d. and exponentially distributed, we can define S_i as the time interval between the ordered version of the W_i (i.e., $S_i = W_{(i+1)} - W_{(i)}$). Thanks to the theory of order statistics [11], S_i is exponentially distributed with average:

$$E[S_i] = E[W_i]/(R - i) \tag{8}$$

By combining (5) and (8), for any $R \in [k, \dots, Q]$:

$$E[L_k] = \sum_{i=1}^{k-1} E[S_i] + E(L_1) = \sum_{i=0}^{k-1} \frac{E[W_i]}{(R-i)} \quad (9)$$

and we get (7). □

The optimal value of $\hat{R}$ can be computed analytically as well. We impose sustainable request arrivals, i.e., $U < 1$, *for any* R to guarantee sustainable arrivals also in the case $R = Q$, it must hold $\lambda < \mu/C$. Thus, we can set:

$$\lambda = \gamma \frac{\mu}{C} \quad (10)$$

with $\gamma \in (0, 1)$ being the load factor. By substituting (10) into (6), we can obtain the optimal number of EPs for 1-Out-Of-Q policy as:

$$\hat{R}_1 = Q/(2\gamma) \quad (11)$$

We can repeat the same derivation also for $\hat{R}_k$, i.e., for a generic k-OutOf-Q policy.

A. Numerical evaluation

In Fig. 3 we reported the endorsement latency computed in the function of γ and R, obtained by substituting (10) in (7). As expected, we observe a minimum endorsement latency obtained with $R = R_k$, as computed analytically, which depends on the load γ. Due to the difficulty to estimate the load in practical scenarios (which may not be stationary), for $k = 1$, we propose heuristically choosing $R = Q/2$ as a sub-optimal redundancy in our proposed approach, discussed in the following. This choice is robust since it is optimal at high load and at low load the latency increase is limited. Indeed, for $\gamma = 0.5$ the increase is no more than 8% compared to the optimal value, and for $\gamma = 0.1$ no more than 12%. Thus for $k = 1$, $R = Q/2$ appears to be a practical solution, which will be exploited when devising online EP selection algorithms in Sec. IV.

The redundancy effect can be limited due to the number of organizations/EPs or applied endorsement policies, as they affect the number of statements generated by Proposition 2. With fewer final statements, there would be less space for redundancy. Indeed, systems with less restrictive and less complex endorsement policies (e.g., majority policies) benefit more from redundancy, while organizations benefit from adopting more EPs to increase reliability.

IV. Practical endorsers selection algorithms

Now, we concentrate on the 1-OutOf-Q policy, since it is coherent with the standard form of any endorsement policy. Without loss of generality, we assume just one client in the system ($C = 1$). We assume that the client is aware of all needed information including available/most-updated EPs, thanks to the configuration query request, as shown in Fig. 1, which leverages the available service discovery process.

We propose an optimization procedure to select the EPs, denoted as `OPEN`, whose main goal is to minimize the endorsement response delay. `OPEN` considers the past response delays experienced by the previously selected EPs and selects the EPs with the lowest delays. This choice is motivated by the high temporal correlation between the response delays of an EP, due to queueing in the network and in the EPs. Notably, the history is meaningful only for recently selected EPs, otherwise, it is obsolete. Therefore, it is possible that a highly loaded EP which was not recently requested becomes among the least loaded ones and is worth again sending the request to it. To address this, `OPEN` probes non-selected EPs by sending gratuitous endorsement requests, which are still considered in the evaluation of the endorsement policy. Furthermore, in `OPEN` pending requests are considered indicators of possibly congested EPs, which are chosen at a lower priority.

The pseudocode of `OPEN` is provided in Fig. 4. Let TX^n be the transaction with sequence number n, evaluated locally at the client. Let x_p^n be the measured response delays of TX^n for any EP $p \in \mathcal{P}$. Let $\mathcal{P}_e^n$ be the set of selected EPs for TX^n. For each transaction, we initialize all EPs as eligible to be selected (ln. 2). Just for the first transaction, `OPEN` initializes the history of response delays to a dummy value and selects all EPs as selected endorsers (ln. 3-6). For a generic transaction, all response delays are initialized to a dummy value (ln. 7-9). Then the EPs are selected based on a procedure described in the next paragraph (ln. 10). Now `OPEN` sends the endorsement request for TX^n to the computed set of EPs (ln. 11) and updates the measured delays (ln. 12). A new instance of the procedure would start if a new transaction TX^{n+1} is generated. Note that the procedure ends when all the responses are received.

We now discuss how `Select-Endorsers` function operates. Inspired by our previous result in (11), it selects $|\mathcal{P}|/2$ EPs chosen among the ones that experienced the lowest response delays, based on the measures for the last transaction TX^{n-1}. The choice is challenging when one or more responses are still pending for TX^{n-1}, and

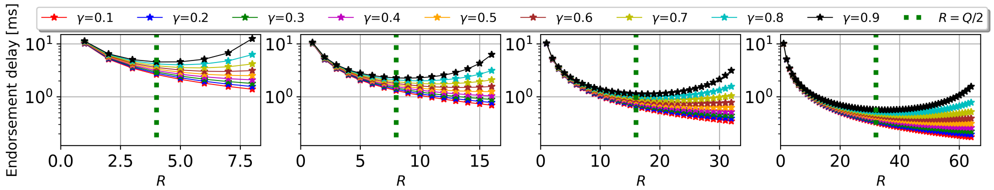


Fig. 3: Endorsement latency where $Q \in [8, 16, 32, 64]$ (left to right). The green line represents $R = Q/2$.

procedure OPEN(n) ▷ Process TX^n
  $e_p^n \leftarrow$ **true**, $\forall p \in \mathcal{P}$ ▷ Init the eligibility vector for TX$^{(n)}$
  if $n = 1$ **then** ▷ Just for the first transaction
    for $p \in \mathcal{P}$ **do**
      $x_p^0 \leftarrow x_p^1 \leftarrow -1$ ▷ Init the response delay history
    $\mathcal{P}_e^1 \leftarrow \mathcal{P}$ ▷ Select all the available peers
  else ▷ Consider a generic transaction
    for $p \in \mathcal{P}$ **do**
      $x_p^n \leftarrow -1$ ▷ Init the measured delays for TX$^{(n)}$
    $\mathcal{P}_e^n \leftarrow$ `Select-Endorsers()`
  `Send-Endorsement-Requests`(TX^n, $\mathcal{P}_e^n$)
  $X^n \leftarrow$ `Update-Response-Delays()`

Fig. 4: Pseudocode of the `OPEN` algorithm for TX^n

1: **procedure** SELECTENDORSERS( )
2:   $d_{\max} = \max_{p \in \mathcal{P}_e^{n-1}} \{x_p^{n-1}\}$ ▷ Max measured delay for TX^{n-1}
3:   **for** $p \in \mathcal{P}_e^{n-1}$ **do** ▷ For EPs used for TX^{n-1}
4:     **if** $x_p^{n-1} = -1$ **then** ▷ Not yet response from EP p
5:       $e_p^n \leftarrow$ **false** ▷ Make the EP Not-eligible for TX^n
6:       **if** $d_{\max} = -1$ **then** ▷ No delay measured for TX^{n-1}
7:         $x_p^{n-1} \leftarrow x_p^{n-2}$ ▷ Use past delays
8:       **else**
9:         $x_p^{n-1} \leftarrow d_{\max} + \epsilon$ ▷ Speculate the delay
10:   **for** $p \in \mathcal{P} \setminus \mathcal{P}_e^{n-1}$ **do** ▷ For EPs not used for TX^{n-1}
11:     $x_p^{n-1} \leftarrow x_p^{n-2}$ ▷ Use past delays
12:   $\mathcal{P}_e^n \leftarrow$ `Eligibile-EPs-with-min-delay`($|\mathcal{P}|/2$,X^{n-1})
13:   $p \leftarrow$ `Random-EP`($\mathcal{P} \setminus (\mathcal{P}_e^n \cup \mathcal{P}_e^{n-1})$) ▷ Select probe EP
14:   $\mathcal{P}_e^n \leftarrow$ `Replace-slowest-EP`($\mathcal{P}_e^n$, p) ▷ Embed the probe EP
15:   **return** $\mathcal{P}_e^n$ ▷ Selected EPs augmented with the probe EP

Fig. 5: Pseudocode for `SelectEndorsers`

the algorithm key idea is that the corresponding EPs are considered as congested and thus should not be selected for the current transaction TX^n.

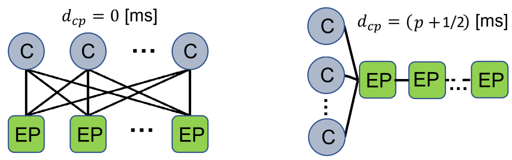
The pseudocode is reported in Fig. 5. It calculates the maximum delay measured for TX^{n-1} (ln. 2). For each EP in $\mathcal{P}_e^{n-1}$ such that the response is not received yet, we mark the corresponding EP as non-eligible (ln. 3-5). There are two cases. The first case is the special one in which no responses have been received for TX^{n-1}, thus the algorithm speculates the delay equal to the delay of TX^{n-2} (ln. 6-7). The eligibility assigned to the EPs will lead to selecting the other $|\mathcal{P}|/2$ EPs compared to the previous ones. The second case is the typical one in which at least some responses have been received for TX^{n-1} (ln. 8). For the EPs used in TX^{n-1} and for which no response has been already received, the speculated delay is equal to the maximum delay $d_{\max}$ plus some constant ϵ, chosen enough small to be negligible compared to the average network and processing delays (e.g., 1 ns) (ln. 9). This will model the fact that the actual delay is unknown, but for sure it is strictly larger than $d_{\max}$. Finally, for all the other EPs, not used for TX^{n-1}, the delays are speculated to be equal to X^{n-2} (ln. 10-11). Now, the EPs are sorted based on the X^{n-2} delay values and the half best will be selected (ln. 12). A random EP from not selected ones will be chosen as the gratuitous probe EP (ln. 13). The slowest EP from $\mathcal{P}_e^n$ will be replaced with the gratuitous probe EP (ln. 14), and $\mathcal{P}_e^n$ will be returned to the main OPEN process (ln. 15).

V. PERFORMANCE EVALUATION

We developed an event-driven simulator using OMNeT++ [12]. We considered a scenario with $C = 8$ clients and $Q = 8$ organizations, each of them with 1 EP, thus $|\mathcal{P}| = Q$. The endorsement requests are generated according to a Poisson process at each client and we set the normalized load $\gamma \in [0.1, 0.9]$. Then fixing $\gamma = 0.5$, we considered more scenarios by varying $Q \in \{8, 16, 32, 64\}$, each organization with the number of EP $\in \{1, 2, 4, 8\}$, and $C \in \{8, 40, 125, 1000, 8000, 32000\}$ clients; in each scenario we fixed all parameters except one. To understand the performance under non-stationary requests, we also considered a Poisson-modulated process with squared-wave cyclo-stationary load, with a period equal to 1200 ms, duty cycle 50%, and normalized load $\gamma = 0.5$. To consider the effect of different kinds of computation, we assume the computation time of each EP to be either exponentially distributed or bi-modal distributed with an average equal to 10 ms, whose value has been achieved from our practical measurements in HF EPs. In the bi-modal case, we assumed that, with a given probability, the computation time is constant with the value $1/\mu_1$, otherwise its value is $1/\mu_2$. Table II shows the coefficient of variations (Cv) for the adopted setting. To model the heterogeneity in the computing power and resources of the EPs, we considered a *non-homogenous scenario* in

TABLE II: Settings for different distributions of the computation time with different coefficient of variation (Cv).

Cv (Bimodal)	0.0	0.5	1	2	5
$P(\mu = \mu_1)$	0.5	0.6	0.75	0.9	0.98
$1/\mu_1$ [ms]	10.0	5.9	4.2	3.4	2.85
$1/\mu_2$ [ms]	10.0	16.1	27.3	70.0	360.0



(a) S1 (bipartite) (b) S2 (linear)

Fig. 6: The two synthetic network topologies adopted for test scenarios in our simulations.

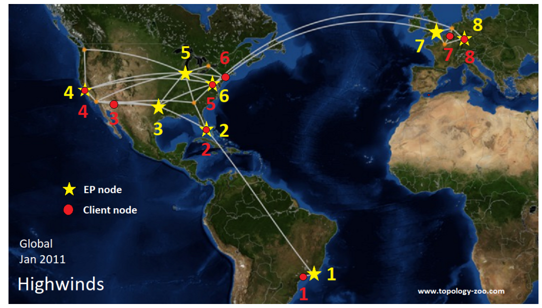


Fig. 7: Real network topology (S3) showing the EPs and clients placement with interconnecting network topology

1: **procedure** DSLM(n) ▷ Process TX^n
2: **if** $n = 1$ **then** ▷ Just for the first TX
3: $l_p \leftarrow \bar{x}_p \leftarrow x_p^0 \leftarrow 0, \forall p \in \mathcal{P}$ ▷ Init EP load and delay values
4: $\mathcal{P}_h \leftarrow$ `Select-`$|\mathcal{P}|$`/2-random-peers` ▷ Random half EPs
5: **for** $p \in \mathcal{P}_h$ **do**
6: $\bar{x}_p \leftarrow [\alpha \bar{x}_p + (1-\alpha) x_p^{n-1}]$ ▷ Average delay
7: **return** $\arg\min_{p \in \mathcal{P}_h} \{(\bar{x}_p^{0.5} + 1) q_p\}$ ▷ Choose min product delay queue length.

Fig. 8: Pseudocode for DSLM adapted to our model

which we assigned different average computation times to different EPs (i.e., $(2, 4, 6, 8, 12, 14, 16, 18)$ ms) where the computation time of each EP is exponentially distributed. We considered three scenarios for the network model, two of them are synthetic and the last one is real. Let d_{cp} be the network delay between client c and EP p. In the first scenario, denoted as S1, the network delays are negligible compared to the processing times at the EP, i.e., $d_{cp} = 0$ (Fig. 6a). In the second scenario, denoted as S2, we set linearly increasing delays between any client and the EPs, similarly to a linear topology where all clients are closer to the first EP, i.e., $d_{cp} = (p + 1/2)$ ms for $p \in [1, Q]$. This implies similar delays from each EP to any client while on average the total network delays are comparable to the processing times at the EPs (Fig. 6b).

In the third scenario, denoted as S3, we selected the *Highwinds* network from [13], shown in Fig. 7, as a real world-wide scenario where the link delays are calculated based on the physical distance between the geographical position of the nodes (using the Haversine formula) and the propagation speed is $2/3$ the speed of light. The clients here can be divided into two groups: (i) *far clients* placed in nodes $1, 7, 8$, and (ii) *centered clients* placed in nodes $2, 3, 4, 5, 6$.

We measure the average endorsement latency as the main performance metric. The endorsement latency is calculated from the moment the endorsement request is sent out from the client until the first response is received by the client. For comparison, we considered three EP selection algorithms, namely RND, OOD, and DSLM, where the first two are proposed by us.

1) Random EPs (RND)

RND is the policy adopted in the analytical model of Sec. III. Every endorsement request is sent to R randomly chosen EPs. If $R = Q/2$, the policy is denoted as *RND-half*. If R adapts to the load according to the rule $R = Q/(2\gamma)$, as in (11), it is denoted as *RND-load*.

2) Dynamic Stochastic Load Minimization (DSLM)

Dynamic Stochastic Load Minimization (DSLM) was proposed in [14] and the pseudocode of the version adapted to our system model is shown in Fig. 8. Just for the first TX, DSLM initializes the load l_p and the measured response delay x_p^0 of any EP p (ln. 2-3). Typically, it randomly selects half of the EPs (ln. 4) and evaluates heuristically the load on each selected EP by the product of the square root of the response delay and the corresponding queue length (ln. 5-6). The average is obtained with an exponential moving average with parameter α. Finally, DSML returns the EP with the lowest estimated load among the selected ones (ln. 7).

3) Oracle Optimal Delays (OOD)

As a reference for all the endorsement algorithms, we define an online Oracle-based Optimal Delays (OOD) EP selection policy that minimizes the endorsement latency given a fixed replication factor R, denoted as OOD-R. The pseudocode of OOD-R is provided in Fig. 9. We assume an oracle that knows in advance the response delay of any endorsement request if sent to a specific EP. Thus, the oracle knows for any EP p: (i) the absolute time t_p^{busy} at which the EP will finish (or has finished) to serve the last received endorsement request TX^{n-1}, (ii) the processing time τ_p^{proc} of the endorsement request

1: **procedure** OOD-$R(n)$ ▷ Process TX^n
2: **for** $p \in \mathcal{P}$ **do** ▷ For each EP
3: $t_p^{\text{resp}} = \max\{t^{\text{now}} + \tau_p^{\text{net}}, t_p^{\text{busy}}\} + \tau_p^{\text{proc}}$ ▷ Compute response delay
4: $p = \arg\min_{p \in \mathcal{P}} \{t_p^{\text{resp}}\}$ ▷ Choose the min response delay EP
5: $\mathcal{P}_e =$ `Find-`$(R-1)$`-EPs-with-largest-`$\{t_p^{\text{resp}}\}$
6: **return** $\{p\} \cup \mathcal{P}_e$ ▷ Return endorsement set

Fig. 9: Pseudocode of OOD-R

TX^n, and (iii) the overall network delay τ_p^{net} between each client and the EP. Thus, if sent to EP p, the response to TX^n will be received from EP p at a predicted time t_p^{resp} (ln. 3) equal to:

$$t_p^{\text{resp}} = \max\{t^{\text{now}} + \tau_p^{\text{net}}, t_p^{\text{busy}}\} + \tau_p^{\text{proc}} + \tau_p^{\text{net}} \quad (12)$$

since if at the time of arriving the request to an EP its queue is empty, then the request will be served at $t^{\text{now}} + \tau_p^{\text{net}}$, otherwise at t_p^{busy}. Then, the request will be processed for τ_p^{proc} and the response will be sent back, experiencing τ_p^{net} delay. Now OOD-R chooses the EP with the smallest predicted time to minimize the response delay (ln. 4). The remaining $(R-1)$ endorsement requests (if any) will be sent to the EPs in decreasing order of predicted time (ln. 5). This allows to load the "slowest" EPs with requests whose responses will be received late and thus reduces the load on the "fastest" EPs, for the sake of future endorsement requests.

It should be noted that, in the case of RND-load, the request arrival is assumed to be stationary, thus, the system load can be estimated with high accuracy. Also, OOD is implementable with enough control information, but obtaining this information would need instantaneous communication with the EPs, which is challenging to accomplish in a practical situation. So, both algorithms are not practical in a real scenario.

A. Simulations results

For a fair comparison between OOD and other approaches, in all test scenarios we selected OOD-half, i.e., with the same R as OPEN, and RND-half, and slightly smaller R than RND-load, for which $R \in [Q/2, Q]$. Only DSLM has a completely different redundancy factor ($R = 1$).

1) Homogenous scenario

The left graphs in Fig. 10 show the simulation results for a homogenous scenario with all EPs with computation times that are exponentially distributed with the same average.

In scenario S1 (left-up), all delays are purely due to processing in the EPs. Since DSLM does not exploit redundancy and for $\gamma = 0.1$ the queuing at the EP is negligible, its response delay is around 10 ms, equal to the computation time. By increasing the load and hence the queueing, the average delay increases slightly. Instead, by exploiting the redundancy all the other approaches can get smaller delays, by a factor of 2 to 6. As expected, OOD-half achieves the best average endorsement latency among all solutions. At low loads, due to the maximum redundancy factor (i.e., $R = 8$), RND-load performs closer to OOD-half by always having the fastest EP among its selection. By increasing the arrival rate, for both RND-half and RND-load, the endorsement latency increases as the selection of EPs is not efficient as OOD-half, which knows in advance the best EP. At high loads, for RND-load, R is almost 4, hence RND-load shows similar results to RND-half. OPEN has a redundancy factor $R = 4$, as RND-half, but selects EPs with smaller estimated delays. At low loads, OPEN has a small advantage over RND-half, as the queueing is almost negligible. As the offered load increases, the higher queueing makes OPEN more efficient, also thanks to the higher frequency by which the response delays are estimated.

In scenario S2 (left-middle), as expected, OOD-half is the best algorithm, and DSLM is outperformed by all other solutions by a factor of 2 to 3. Due to the linearly increasing network delays, the effect of redundancy in EPs becomes less dominant, so the delay's improvement in scenario S2 is less than in S1. On the other hand, at low loads, OPEN acts slightly better than RND-half compared to S1, thanks to being aware of the network delays. At high loads, OPEN behaves close to RND-half since the queueing delays become dominant to network delays.

In scenario S3 (left-down), again OOD-half is the best approach, DSLM is outperformed by all other solutions by at least a factor of 2. Due to the different network delays, on average much larger than the computation times, the redundancy is less effective, thus a lower delay improvement is experienced in S3 compared to S2, and S1. OPEN performs quite similarly to RND-load in low loads even with a half number of selected EPs, and much better in high loads. OPEN completely outperforms RND-half in all loads since it exploits mainly the EP with lower network delays.

2) Non-homogenous scenario

The simulation results for a non-homogenous scenario are reported in the middle graphs of Fig. 10. As a reminder, now the average computation times for the EPs are different, but the overall average is the same as in the homogenous scenario. In all three scenarios S1, S2, and S3, as DSLM is not able to exploit redundancy, it is not able to reduce its average latency. On the other hand, by exploiting redundancy, all other approaches can

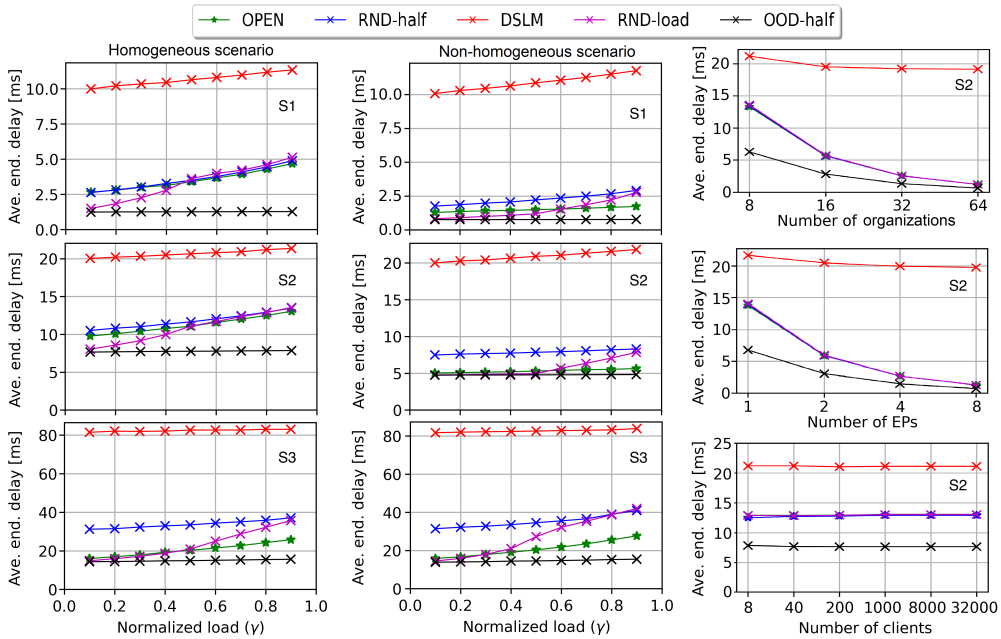


Fig. 10: Average endorsement delay for i) average computation time of 10 ms for each EP: S1 (left-up), S2 (left-middle), S3 (left-down), ii) different average computation times from [2 to 18] ms for each EP: S1 (middle-up), S2 (middle-middle), S3 (middle-down), and iii) the different number of organizations, EPs, and clients in scenario S2, for $\gamma = 0.5$ (right-most graphs).

reduce their average latency, where OOD-half achieves the lowest latency thanks to its global knowledge of the system.

In scenario S1, RND-load reduces the latency more than RND-half, since the higher redundancy factor increases the chance of selecting EPs with lower average computation times. With the same redundancy factor as RND-half, OPEN reduces the most the delays for all loads, as it employs the latency history to select EPs with lower average computation times. In scenario S2, a similar behavior as in S1 is observed for all the algorithms. OPEN, by exploiting the delay history comprising both the average computation times and the network delays, achieves the best performance by almost approaching OOD-half. In scenario S3, we observe almost similar results as in scenario S3 of the homogeneous case, as the variation in the average computation times is still negligible to the average network delays.

3) Scaling the number of organizations, EPs, and clients.

The simulation results for larger scenarios are shown in Fig. 10 (right). We consider the S2 scenario, to get a heterogeneous system in terms of network delays, and we fixed $\gamma = 0.5$. By increasing the number of organizations, the overall number of EPs increases, thus the endorsement latency is reduced for all the algorithms exploiting redundancy, as shown in Fig. 10 (right-up). The same behavior is observed when the number of EPs in each organization increases (see Fig. 10 (right-middle)). The similarity with the previous graph is that we are considering the 1-OutOf-N policy here, which by recalling (2), for this endorsement policy there is no difference between two EPs of the same organization or different EPs of different organizations.

According to Fig. 10 (right-down), changing the number of clients has no effect on the approaches. Note that increasing the number of clients will reduce the efficiency of the information gained by OPEN and it will converge to the RND-half results for homogeneous cases with less dominant network delays.

4) Bi-modal computation times

The simulation results are shown in Fig. 11. In scenario S1, for constant computation time (Cv=0), the redundancy is not beneficial for delay reduction, while at high loads it can increase the EPs' queue length and thus the delay. For larger Cv, all the algorithms, except DSLM, decrease the average endorsement delay. This is because the average of the minimum between a sequence of i.i.d. random variables is smaller when the variance is larger. All the solutions, except for DSLM, behave

TABLE III: Average endorsement delays for cyclo-stationary input rates with different scenarios.

Scenario	S1 [ms]	S2 [ms]	S3 [ms] centered clients	S3 [ms] far clients
OOD-half	1.2	7.1	13.6	16.9
OPEN	2.9	11.7	17.9	24.2
RND-load	3.1	11.2	19.2	23.7
RND-half	3.2	12.2	25.9	43.6
DSLM	10.8	22.8	65.9	110.3

similarly for low and high loads.

In scenario S2, also for Cv=0, the redundancy reduces the average delay. The reason is that DSLM considers the computation load at the EPs obliviously of the network delays, which are dominating the computation times. But, in the other approaches, redundancy increases the chance of selecting the EP with lower network delays. By increasing Cv, redundancy can reduce the latency even more, by benefiting from the variability in the computation times. At low load ($\gamma = 0.2$), OPEN performs quite well as it also selects EPs with lower network delays. RND-load is performing slightly better as it sends to all EPs. OOD-half is even better than RND-load with a small margin, thanks to the lower load guaranteed by setting $R = 4$. At high load ($\gamma = 0.8$), RND-load adopts $R = 5.7$ (on average) and the corresponding queueing penalizes the overall response delay. OPEN acts slightly better thanks to the smaller value of R.

In the S3 scenario, all approaches are not affected by Cv, as the variability in the computation times is compensated by the network delays which vary between 0 ms and 7 times the average computation time. At low load ($\gamma = 0.2$), OPEN selects closer EPs in terms of network delays and outperforms RND-half by a factor greater than 2, while being very close to OOD-half. RND-load achieves the same results as OPEN by selecting all the EPs (i.e., $R = 8$), which include the closest EP as well. At high load ($\gamma = 0.8$), as in scenario S2, RND-load is penalized by the queueing. OPEN reduces the endorsement delay up to 70% compared to DSLM. Notably, differently from OPEN, RND-load may not select the closest EPs. As expected, for both loads OOD-half performs the best, since it always selects the minimum combination of the network delay and the processing delay.

5) Cyclo-stationary request process

We compared OPEN with other approaches under Poisson-modulated cycle-stationary load. We evaluated the average endorsement delays by using an exponential moving average. The results are provided in Table III. In S1 and S2, OPEN, RND-half, and RND-load showed almost constant average endorsement latency, while DSLM and ODD results are the highest and the lowest respectively. Interestingly, all the results for different approaches in S1 and S2 are very close to the results gained from Fig. 10 (left) for $\gamma = 0.5$, even if the load was changing periodically. This means that all of them are robust to load change in homogeneous scenarios.

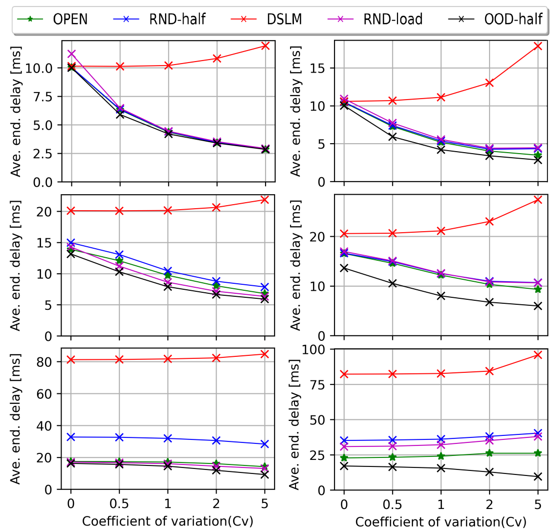


Fig. 11: Average endorsement latency under bimodal computation times: i) normalized load: $\gamma = 0.2$ (left), $\gamma = 0.8$ (right), ii) scenarios: S1 (up), S2 (middle), S3 (down)

In S3, as a non-homogenous real scenario, OPEN shows a small difference of the average endorsement latency between centered and far clients (recall their definition in Sec. V). This difference (6 ms) is negligible compared to the average network delays in S3 (50 ms). The same behavior is observed for RND-load. On the other hand, in RND-half the performance depends heavily on the client's position; even in the case of centered clients, RND-half experiences more endorsement latency than OPEN with far clients. As expected, OOD-half achieves the best endorsement latency with minimum difference regardless of the client's position. DSLM performs the worst with latencies about 4 times larger than OPEN.

These results show that OPEN adapts to load changes even in the presence of unbalanced network delays. Also, OPEN outperforms RND-half and DSLM, while it shows similar results to RND-load and to OOD-half.

VI. Related works

Different works modeled analytically the endorsement process in HF. [9] modeled the EPs as M/M/1 queues and considered the propagation delays in the network model, coherently with our work. It showed that using a pure "AND" endorsement policy, compared to "OR" or "k-OutOf-Q" policies, significantly increases the endorsement delay by increasing the number of organizations. Similarly, [15] showed the same results by modeling HF using stochastic reward networks. They also observed that for "OR" and "k-OutOf-Q" policies the latency decreases by increasing the number of EPs within the same organization, similar to the effect of increasing R in our work.

[8] modeled HF using Generalized Stochastic Petri Nets and showed that for high request arrival rates, the endorsement phase is a performance bottleneck of HF. This is coherent with the motivation of our work, focusing on optimizing the endorsement phase. [16] considered four organizations and showed that simple endorsement policies based on "AND", "OR" and "k-OutOf-Q" operators, experience the minimum latency. [17] showed that using "k-OutOf-Q" policy, increasing k decreases the throughput and increases the latency. This is coherent with our system model since the endorsement latency will be the maximum among k request delays. [18] optimized the HF configurations to improve the throughput and reduce the delays. Coherently with our results, they showed the equivalence between the "1-OutOf-Q" policy and the "OR" among all organizations. Our results in Sec. II-B generalize such property.

Some works tried to improve endorsement phase of HF. [14] proposed a way to select the best EP for "1-OutOf-Q" endorsement policy in HF v1.4. They introduced an algorithm running in each EP, called DSLM, to calculate the EP's load by considering multiple resource metrics within an EP. For each request, only half of the EPs are probed to get their actual load, coherently with $R = Q/2$ adopted in OPEN. A version of DSLM tailored to our system model has been considered in Sec. V as an alternative approach to be compared with OPEN. [19] showed that the failed transactions due to timeouts are affected by the number of statements within the "AND" operator defined in the endorsement policy. Such failures increase the latency and waste of resources due to re-transmissions at the application level.

[20] suggested a way to reduce the possibility of endorsing conflicting transactions. They proposed a cache mechanism inside the EPs to record some data of the recently endorsed transactions and drop the conflicting proposal before execution. Recall that, in the endorsement phase, no execution results will update the world state, so transactions with similar initial world states can propose different updates for the world state. This early drop of the proposal before execution will reduce the computing and network resources by reducing the chance of transaction failure at the validation phase. [21] removed unnecessary operations for pure read requests, by modifying the EPs algorithm to differentiate the process of pure read transactions from mixed read/write ones. This reduced the latency and resource consumption in the endorsement phase.

The main idea of OPEN is to send multiple replicas of the same request to multiple peers. This approach has been deeply investigated in the literature on queueing theory, motivated by the problem of optimal job assignment to servers. As the literature is huge, we focus just on a few papers for the sake of space. In the generic literature about distributed systems, several works [22], [23], [24] investigated the effect of sending replicas of a job to more than one randomly selected server and waiting for the first response to exploit redundancy, as in OPEN. These works introduced redundancy to reduce the job completion time and overcome server-side variability, where a server might be temporarily slow, due to many factors like garbage collection, background load, or even network interrupts. [25] showed that, besides its simplicity, in many cases, redundancy outperforms other techniques for overall response time. [26], by decoupling the inherent job size from the server-side slowdown, described a more realistic model of redundancy and showed that increasing the level of redundancy can degrade the performance, coherently with our observations in Sec. III. [27] showed that a major improvement results from having each job replicated to only two servers, coherently with our Fig. 3 which shows that for the 1-OutOf-k policy, the endorsement latency decreases mostly when varying R from 1 to 2. On the contrary, in our work, we have considered the optimal value of R that minimizes the endorsement latency, which may be greater than 2. [28] showed the reverse relation between the incoming load and the optimal number of replicas, coherently with (11), and experimentally obtained the optimal redundancy factor in different job arrival rates and for different service times. Also, [29] theoretically demonstrated that, when replicating the job to multiple servers, the best choice in case of low (or, high) loads is to replicate to all (or, only 1) servers, coherently with (11) and with the operations of OPEN, which adapts the replication factor to the instantaneous load.

VII. Conclusions

We addressed the problem of minimizing the endorsement latency in HF. Leveraging some results obtained in a simplified queueing model, we proposed the OPEN algorithm to choose multiple EPs for each transaction by taking into account the measurements from the past requests, in a realistic scenario. Through simulations with OMNeT++, we showed that independently from the scenario, OPEN is robust and achieves performance remarkably close to the optimal oracle-based approach (OOD) and outperforms state-of-the-art solutions.

OPEN has been validated only by extensive simulations. Beyond the scope of this work, we implemented OPEN in HF to validate the proposed approach in a realistic setting. The experimental results of the first version of the proof-of-concept are very promising. We leave the optimization of the design of the client-based OPEN solution and its extensive experimental validation for future work.

REFERENCES

[1] L. Hughes, Y. K. Dwivedi, S. K. Misra, N. P. Rana, V. Raghavan, and V. Akella, "Blockchain research, practice and policy: Applications, benefits, limitations, emerging research themes and research agenda," *Journal of Information Management*, 2019.

[2] S. F. Wamba and M. M. Queiroz, "Blockchain in the operations and supply chain management: Benefits, challenges and future research opportunities," *International Journal of Information Management*, 2020.

[3] I. Abu-elezz, A. Hassan, A. Nazeemudeen, M. Househ, and A. Abd-alrazaq, "The benefits and threats of blockchain technology in healthcare: A scoping review," *International Journal of Medical Informatics*, 2020.

[4] S. Solat, P. Calvez, and F. Naït-Abdesselam, "Permissioned vs. permissionless blockchain: How and why there is only one right choice," *Journal of Software*, 2020.

[5] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger Fabric: a distributed operating system for permissioned blockchains," in *ACM EuroSys*, 2018.

[6] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: An introduction," 2016. [Online]. Available: https://docs.r3.com/en/pdf/corda-introductory-whitepaper.pdf

[7] "Endorsement policies." [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/endorsement-policies.html

[8] P. Yuan, K. Zheng, X. Xiong, K. Zhang, and L. Lei, "Performance modeling and analysis of a Hyperledger-based system using GSPN," *Computer Communications, Elsevier*, 2020.

[9] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for Hyperledger Fabric blockchain network," *Information Processing & Management*, 2021.

[10] P. Purdue, "The M/M/1 queue in a Markovianenvironment," *Operations research*, vol. 22, no. 3, pp. 562–569, 1974.

[11] H. A. David and H. N. Nagaraja, *Order statistics*. John Wiley & Sons, 2004.

[12] "Omnet++." [Online]. Available: https://omnetpp.org/

[13] "The Internet Topology Zoo," http://www.topology-zoo.org.

[14] C. Liu, M. Li, Y. Wang, Y. Wang, D. Huo, and Y. Chen, "Achieve better endorsement balance on blockchain systems," in *CSCWD*. IEEE, 2021.

[15] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of Hyperledger Fabric (permissioned blockchain network)," in *IEEE NCA*, 2018.

[16] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing Hyperledger Fabric blockchain platform," in *IEEE MASCOTS*, 2018.

[17] L. Hang and D.-H. Kim, "Optimal blockchain network construction methodology based on analysis of configurable components for enhancing Hyperledger Fabric performance," *Blockchain: Research and Applications*, 2021.

[18] S. Figueroa-Lorenzo, J. Añorga, and S. Arrizabalaga, "Methodological performance analysis applied to a novel IIoT access control system based on permissioned blockchain," *Information Processing & Management*, 2021.

[19] J. A. Chacko, R. Mayer, and H.-A. Jacobsen, "Why do my blockchain transactions fail? a study of hyperledger fabric," in *ACM SIGMOD*, 2021.

[20] F. Lu, L. Gan, Z. Dong, W. Li, H. Jin, and A. Y. Zomaya, "A cache enhanced endorser design for mitigating performance degradation in Hyperledger Fabric," in *IEEE International Conference on Blockchain*, 2018.

[21] M. Kwon and H. Yu, "Performance improvement of ordering and endorsement phase in Hyperledger Fabric," in *IOTSMS*. IEEE, 2019.

[22] N. B. Shah, K. Lee, and K. Ramchandran, "When do redundant requests reduce latency?" *IEEE Trans. on Communications*, 2015.

[23] A. Vulimiri, O. Michel, P. B. Godfrey, and S. Shenker, "More is less: Reducing latency via redundancy," in *ACM HotNets*, 2012.

[24] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Low latency via redundancy," in *ACM CoNEXT*, 2013.

[25] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Perf. Evaluation Review*, 2015.

[26] K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, and B. Van Houdt, "A better model for job redundancy: Decoupling server slowdown and job size," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3353–3367, 2017.

[27] K. Gardner and S. Zbarsky, "Analyzing response time in the redundancy-d system," *ACM SIGMETRICS, Perf. Evaluation Review*, 2015.

[28] J. Hollinghurst, A. Ganesh, and T. Baugé, "Latency reduction in communication networks using redundant messages," in *ITC*, 2017.

[29] G. Joshi, E. Soljanin, and G. Wornell, "Efficient replication of queued tasks for latency reduction in cloud systems," in *Allerton Conference on Communication, Control, and Computing*, 2015.

Iman Lotfimahyari received his BSc and MSc in Electronics Engineering from IA University in 2003 and 2007, respectively. In March 2020, he received his second MSc in Telecommunication Engineering from Politecnico di Torino, Italy, and joined the Telecommunication Networks Group of Politecnico di Torino as a Ph.D. student. His current research interests involve programmable data planes for SDN and blockchains.

Paolo Giaccone received the Dr.Ing. and Ph.D. degrees in telecommunications engineering from the Politecnico di Torino, Italy, in 1998 and 2001, respectively. He is currently a Full Professor in the Department of Electronics, Politecnico di Torino. His main area of interest is the design of optimal network control algorithms.