PleBeuS: a Policy-based Blockchain Selection Framework

Eder J. Scheid, Daniel Lakic, Bruno B. Rodrigues, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
[scheid,rodrigues,stiller]@ifi.uzh.ch, daniel_lakic@access.uzh.ch

***Abstract*—Due to the growing interest in the blockchain (BC), several applications are being developed, taking advantage of the benefits that such technology promises to deliver, such as removal of Trust Third Parties (TTP) to verify transactions and data immutability. However, these applications require certain aspects, such as high transaction throughput or data privacy, that early BC implementations (*e.g.,* Bitcoin) did not provide. Thus, a myriad of novel BC implementations was developed, which introduced the issue of choosing the right implementation for a specific use-case. This paper presents a framework, called `PleBeuS`, to address this selection issue by allowing users to specify policies that rule the automatic selection of the BC that data will be stored. The selection process relies on a cost-aware approach and considers both public and private implementations and their technical characteristics. Moreover, `PleBeuS` communicates with a BC-agnostic interoperability API to enforce transactions. The evaluation of the `PleBeuS` prototype showed that it is possible to automatically select a BC-based on user policies, considering cost thresholds and technical details (*e.g.,* BC throughput, deployment), and reduce manual interaction.**

***Index Terms*—Policy-based Management; Blockchain; Interoperability.**

I. Introduction

The blockchain (BC), introduced in 2009 with the release of Bitcoin [21], has attracted attention due to the paradigm shift in the financial market that it proposes. This shift is represented by the removal of Trusted Third Parties (TTP), such as banks, to verify monetary transactions between individuals, which led to the high speculation on the value of their underlying coins, called cryptocurrencies. This attention is visible due to the creation of more than two thousand cryptocurrencies [10], such as Ethereum [4] and Ripple [7]. However, the employment of the technical infrastructure that supports these cryptocurrencies (*i.e.,* the BC) is not limited to the financial market. Thus, being employed in several applications [15], increasing purpose-specific BCs implementations, tackling a myriad of fields, *e.g.,* agriculture [28], automotive [11], and pharmaceutical goods [3].

Each new BC implementation focuses on improving different characteristics, such as transaction throughput [27] or privacy [32]. For example, in the context of the pharmaceutical industry, where Internet-of-Things (IoT) devices monitor the temperature of drugs during transport (*i.e.,* cold-chain), high transaction throughput is required due to the constant monitoring to ensure compliance with standards. However, when sensitive data about users (*e.g.,* health-related data [12]) is involved, it must remain confidential; thus, the BC implementation should support encryption mechanisms. It can be seen that different types of data (*e.g.,* temperature readings and health-related information) require different storage solutions and, consequently, different BC implementations. Thus, the question that drives this paper is how to select the most fitting BC based on user requirements considering a variety of BCs.

One answer to address this question is to employ already established management concepts, such as Policy-based Management (PBM). PBM was first proposed in the context of distributed systems [25] and later successfully employed in network management [26] to reduce the complexity of managing several network devices with vendor-specific configurations. Applying PBM in the BC context, policies can be employed to express the requirements that a BC must satisfice to store the incoming data. Moreover, the complexity of managing the data in multiple BCs is reduced because policies can be adjusted to reflect new requirements without altering the underlying implementation of BC-enabled applications. Directions toward the combination of PBM and BC selection were presented in [23] in a high-level architecture along with a BC-agnostic interoperability Application Programming Interface (API).

Thus, this paper presents a Policy-Based BC Selection framework, called `PleBeuS`, that relies on a cost-aware approach to automatically select the BC that fits the requirements specified by end-users in the form of policies. These policies consider technical BC characteristics, such as block time, transaction throughput, and deployment type, to guide the selection. This paper details major technical details of policies and the decision-making process, whereas past efforts focus on research directions [23] and technical details of BC-specific adapters [22], which only enforce transactions, *i.e.,* propagate a BC-dependent transaction to miners, to the selected BC. The scientific contributions of this paper are *(i)* automated BC selection based on policies, *(ii)* abstraction and characterization of BC implementations, *(iii)* cost-aware selection algorithm and policy switching mechanism, and *(iv)* user-tailored BC-agnostic interaction.

The remainder of this work is structured as follows. Section II presents a background on blockchain and on PBM. Section III provides a description `PleBeuS` and technical details. Section IV presents an evaluation and discussion on the prototype. Section V describes related work. Finally, Section VI summarizes the paper and presents future work.

II. Background

This section details the background on the two core concepts related to the solution proposed in this paper, *(i)* BC, and *(ii)* Policy-based Management (PBM). In Section II-A, the BC concept, and its technical characteristics are presented. These characteristics are crucial for this paper, as they are used as selection criteria. PBM is briefly described, due to its wide employment, in Section II-B within the networking context.

A. Blockchain (BC) Characteristics

In essence, a BC is a distributed append-only immutable ledger that relies on cryptographically appended transactions organized in blocks to remove the need for a TTP to verify new transactions within a distributed environment of potentially unknown stakeholders. The aspects that vary from a BC implementation to the other are described below.

1) Consensus Mechanism: Consensus mechanisms are a crucial component of BC implementations, as they secure the network against double-spending attacks by verifying transactions, and maintain the correct state in each peer. For a more in-depth discussion on the various BC consensus mechanisms and their differences, one can refer to [30]. The most popular consensus mechanisms are described in the next items.

- *Proof-of-Work (PoW)* requires competition between nodes to calculate the hash value of the next block that will be appended in the BC. When a node finds a hash that is below a defined threshold, it broadcasts the hash to all the nodes for verification and inclusion in the BC [21].
- *Proof-of-Stake (PoS)* selects block validators pseudo-randomly depending on their stake in cryptocurrency [24]. The concept behind PoS is that nodes with higher stakes contribute toward securing the BC.
- *delegated Proof-of-Stake (dPoS)* employs a selection process based on a defined set of validators, which take turns to verify and create new blocks. Even though this approach allows for a scalable solution, it introduces a higher degree of centralization due to the limited number of validators [24].

2) Deployment Type: The deployment type of a BC implementation can be classified into different categories depending on its write and read permissions. Public and private categories relate to data visibility (*i.e.,* which information users are able to access/read), and permissioned and permissionless categories data writability (*i.e.,* who is allowed to append information) [31]. The combination of these categories (*i.e.,* private, public, permissioned, and permissionless) affect who can control and manage the integrity of the BC and form the following categorization:

- *Public permissionless BCs* grant write/read access to any peer within the network.
- *Public permissioned BCs* grant read access to any peer, but write access is restricted to selected peers.
- *Private permissionless BCs* grant write/read access to selected peers within a closed network.
- *Private permissioned BCs* manage write/read access by a centralized organization.

3) BC Performance: Measuring the performance of a BC (*i.e.,* BC throughput) is not a straightforward task as it depends on different factors, such as block time, block size, and transaction size. BC throughput can be measured in different metrics, such as byte per milliseconds (byte/ms) [8] or transactions per second (tps) [16]. In this paper, the performance of a BC is measured in tps as it is more precise to grasp than byte/ms.

- *Block Time* defines the period that new blocks are appended in the BC by miners or validators.
- *Block Size* defines the size of each block in the BC; this size is directly related to the number of transactions that can be included in a block.
- *Transaction Size* determines the size of a standard transaction (*i.e.,* exchange of funds) in the BC.
- *BC Throughput* is calculated by dividing the *Block Size* by the *Transaction Size*, which results in the maximum number of transactions that can be appended in a block. This value is divided by the *Block Time* to achieve the maximum theoretical tps of a BC.

4) Data size: Refers to the amount of arbitrary data (*i.e.,* not transaction-related) a single transaction is able to store. Transaction-focused BCs (*e.g.,* Bitcoin) are not designed for storing arbitrary data because they were not conceived as a database, but rather as a secure means of funds exchange. However, with the popularization of the BC, other BCs began to allow data to be embedded in a dedicated field in the transaction (*e.g.,* Ethereum). Thus, employing the BC as an immutable database.

5) Transaction Costs: Public BCs often rely on providing incentives for miners to secure the BC, *i.e.,* verify transactions to avoid double-spending. These incentives are in the form of block reward and transaction fees. The block reward does not affect the cost of a transaction because it is employed to mint new coins. However, the calculation of transaction fees depends on the implementation of the BC and its consensus mechanisms. For example, Stellar assumes a fixed base fee [13] for each operation performed by the transaction, and in Bitcoin miners select the transactions with higher fees to maximize earning; thus, transactions with higher fees tend to be included in the Bitcoin BC faster.

6) Turing-complete Smart Contracts (SC): SCs contain a set of rules under which the parties to that SC agree to interact with each other [1]. When these rules are met, the agreement is automatically enforced. SCs are triggered by addressing a transaction to the contract. In that case, execution of the SC is enforced independently and automatically in a prescribed manner on every node (miner) in the BC, using the data that was included in the transaction as input [9]. While transaction-focused BCs (*e.g.,* Bitcoin), have SC capabilities, they are constrained. The reason for that is that Bitcoin's script is not Turing complete since it is not able to solve the halting problem [17]. This restricts the functionality SCs of these BCs are able to provide.

B. Policy-based Network Management (PBNM)

Policy-based Network Management (PNBM) is an already established concept being studied for several years [29]. PBNM is defined as "*..the usage of rules to accomplish decisions*" [26]. In the network management context, rules (*i.e.,* policies) are often specified in an Event-Condition-Action (ECA) model. For example, an ECA rule can be: when incoming packets on router R (*event*), if destination port equals to `(80 or 443)` and source IP address is in range `130.60.156.1/24` (*condition*), then forward the packet to device DPI on `130.60.156.160` (*action*). The employment of rules to guide the management of resources was initially proposed in the context of distributed systems [25] and then applied to network management with success. Thus, it can be seen that PBNM principles can be abstracted to be applied in other contexts, such as deciding which BC to store data based on rules.

The PNBM architecture, defined in RCF 3060 [20], can be represented by four main components, *(i)* Policy Management Tool (PMT), *(ii)* Policy Decision Point (PDP), *(iii)* Policy Enforcement Point (PEP), and *(iv)* Policy Repository. In such an architecture, the PMT is responsible for allowing users to author policies, policy refinement/translation following the Policy Continuum [26], and detecting conflicts. The PDP is responsible for managing which policy is active or inactive, and deciding which policy is going to be enforced in the PEP. In the network management context, the PEP is defined as any network device (*e.g.,* routers, switches, and middleboxes), which rules can installed or actions performed. The last component, the Policy Repository, stores and maintains policy conditions, actions, and related management data (*e.g.,* users, domains, and configurations).

Due to policies being managed by users, conflicting policies can arise. Policy conflicts emerge, when there are contradictory policy configurations (*e.g.,* two policies within the same activity time frame). As a result, two or more policies are activated, which results in contradictory statements [19]. Policy conflicts can arise during *(i)* policy specification (*i.e.,* static) or *(ii)* runtime (*i.e.,* dynamic). The former arise during the creation of the policies and can be eliminated by using static analysis [5]. While the latter is detected during the enforcement of policies and must to be solved automatically to ensure the correct operation of the system without interruptions [6].

III. Policy-based BC Selection Framework

This paper proposes a PBM-based framework, called `PleBeuS`, to aid in the selection of the most appropriate BC to store data based on user requirements. `PleBeuS` allows users to represent their requirements in the form of policies that are used as inputs for filtering and selection algorithms. Subsequent sections present the `PleBeuS` framework, detailing architectural components, policy parameters, and the employed cost-aware selection algorithms.

A. Architecture

The architecture of `PleBeuS` follows the PBM concept; thus, it is divided in the PMT, the PDP, and the `Bifröst` API [22] acting as the PEP. Figure 1 depicts `PleBeuS` architecture and its components. The components depicted using a dashed line, such as the transaction costs service, are external solutions and were not implemented.

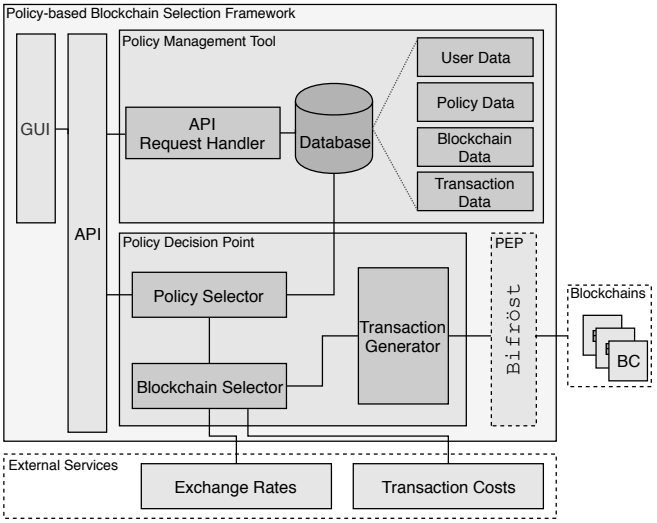

Fig. 1: `PleBeuS` Architecture

- **API Request Handler:** Allows to configure and manage policies, which are then stored in the database. It is a Web service that interacts with a *Graphical User Interface (GUI)*, which executes REST API calls in the background. The GUI allows users to create new policies, delete, or edit existing policies and presents cost and transaction statistics.
- **Database:** The database stores users, policies, BC information (see Table I), and the transactions that were enforced by the PEP.
- **Policy Selector:** Upon receiving a request to store data in a BC, this component retrieves the active policy that matches the time frame and the user that sent the request. The retrieved policy is then forwarded to the *Blockchain Selector* to execute the selection algorithms.
- **Blockchain Selector:** This component implements the selection algorithms (*cf.* Section III-C), being responsible for retrieving the transaction costs for each BC and their exchange rate to calculate the cost thresholds in the currency defined by the user. Moreover, once a BC or more were selected, it sends the selected BC to the *Transaction Generator*.
- **Transaction Generator:** This component receives the selected BC and constructs a transaction in the format required by the `Bifröst` API, *i.e.,* an HTTP request containing the data to be stored and the selected BC.

TABLE I: Summary of BCs and Characteristics

BC Name	Type	TPS	Block Time [s]	Max. Arbitrary Data	Turing-complete SCs	Fees
Bitcoin	Public	4 - 7	600	80 bytes	No	variable
Ethereum	Public	15 - 25	15	46 kBytes	Yes	variable
Stellar	Public	1000 - 4000	5	28 bytes	No	base fee
EOS	Public	250 - 3996	0.5	256 bytes	Yes	variable
IOTA	Public	500 - 800	60	1300 bytes	No	none
Hyperledger	Private	variable	20 (default)	20 bytes	Yes	none
Multichain	Private	variable	15 (default)	80 bytes	No	none

B. Policy Parameters

The policy parameters allowed in `PleBeuS` are divided into two different categories, *(i)* BC-specific and *(ii)* externally driven. These two categories and related parameters are detailed in the next sections.

1) BC-specific Parameters: These parameters are bound to different BC implementations characteristics and only change if a hard-fork occurs in the BC. Hence, the characteristics of the BC implementations are mapped one-to-one as parameters for a policy. They are used to filter BC implementations based on their characteristics and not result in a policy switch. Table I presents the summary of the BCs supported by the PEP (*i.e.,* `Bifröst` API) including their key characteristics.

- *Public vs. Private*: Users may choose whether the data to be stored should be open to the public or not. If the transactions contain sensitive data, private BCs should be selected to ensure privacy, while taking advantage of the BC immutability for posterior audits.
- *BC Throughput*: Represents the minimum amount of tps that a BC implementation must supports.
- *Data Size*: Determines the minimum amount (in bytes) of arbitrary data, *i.e.,* non-transaction related, that a BC-specific transaction must allow to include.
- *Turing-completeness*: Determines whether the selected BC must support complex SCs or not.

2) Externally Driven Parameters: These parameters are not static and are susceptible to external factors, *e.g.,* the time of the day. They do not map any BC characteristics, but rather map user preferences and are responsible for policy switching, *i.e.,* determining which policy is active.

- *Cost Thresholds*: Users can set a maximum cost amount they are willing to spend in a specific interval. When the accumulated transaction costs reach the defined threshold, the currently active policy switches to the next defined policy with a higher threshold. `PleBeuS` validates the policies to prevent that previous intervals does not exceed the threshold of the upcoming intervals.
- *Cost Interval*: When defining cost thresholds, users need to define for which interval the thresholds are valid. `PleBeuS` supports five intervals, *(i)* **daily**, *(ii)* **weekly**, *(iii)* **monthly**, *(iv)* **yearly**, and *(v)* **default**. When a call to store data is performed, the calculated costs are added to each of the intervals for the current user. Default specifies the fallback policy when all thresholds are reached.
- *Fiat Currency:* Currently, `PleBeuS` supports **CHF**, **EUR**, and **USD**. This currency is used for converting cost calculations in a specific cryptocurrency to a fiat currency.
- *Cost Profile:* This parameter is used in case multiple BC implementations fulfill the criteria defined by the policy. Users can choose between **performance** and **economic** cost profiles. In case **performance** is selected, `PleBeuS` selects the most performant BCs from the BC set. Consequently, for **economic**, the BC with the lowest transaction costs is selected from the BC set. In terms of policy priority, `PleBeuS` prioritizes performance over economic policies within the same interval.
- *Transaction Split:* In case the transaction split is set to **false**, `PleBeuS` selects a single BC for all transactions. However, if set to **true**, the transactions are sent, in a round-robin scheme, to the set of BCs to increase performance, executing the selection algorithm based on the *Cost Profile*.
- *Time Frame:* The time frame specifies when the policy should be active, given that all the other constraints (*e.g.,* cost thresholds) are fulfilled. This parameter can either be set as valid for the whole day or at a specific time frame of the day (*e.g.,* **08:00 - 17:00**). An overlap of time frames will result in a policy conflict.
- *Preferred BCs:* Provides the user the possibility to select the most relevant (for the user) BCs. This parameter can range from 1 to n BCs. However, if a single BC is selected, the selection algorithms are not executed.

C. Cost-aware BC Selection Algorithm

The BC selection process first applies straightforward filtering, following a divide and conquer approach, with a complexity of $\mathcal{O}(\log n)$, removing BCs whose characteristics do not match with the selected ones. These filters are defined in the policy and applied to reduce the size of the BC set that will be used as input for the BC selection algorithm. The following procedure defines the priority of the filters. Let $F(f(x), P)$ be the collection of filters, where $f(x)$ is the filtering function (*e.g.,* minimum tps, BC type, Turing-complete, block time, and minimum data size) and P the priority ranging from 1 (*i.e.,* highest priority) to the number of filters (*i.e.,* lowest priority). Moreover, let BC_{pool} be the available BCs, and let BC_{set} be the resultant set after applying the filters. Thus, $BC_{set} \leftarrow BC_{pool}$ then $\forall f(x) \in ordered(F(f(x), P))$ do $BC_{set} \leftarrow f(BC_{set})$.

After the filters are applied, based on the *Cost Profile* defined by the user in the policy, PleBeuS executes two algorithms to select the most appropriate BC from BC_{set}. If the user specified a *Cost Profile* that prioritizes **performance**, PleBeuS executes Algorithm 1. This algorithm selects the BC that presents the highest *tps*, meaning that the data will be immutably recorded in a BC as fast as possible, disregarding the costs. However, if two BCs present the same tps value, the algorithm prioritizes the one with lower costs.

Algorithm 1 Fastest BC Selection

```
1: procedure SELECTFASTESTBC(bcSet, costs)
2:     fastest ← bcSet[0]
3:     for all bc ∈ bcSet do
4:         if (bc.tps > fastest.tps) then
5:             fastest ← bc
6:         else
7:             if (bc.tps == fastest.tps) and (costs[bc] < costs[fastest]) then
8:                 fastest ← bc
9:             end if
10:        end if
11:    end for
12:    return fastest
13: end procedure
```

Algorithm 2 is executed by PleBeuS for policies containing the **economic** *Cost Profile*. This algorithm selects the BC that presents the lowest transaction costs from the BC set. If two BCs present the same transaction costs, then the algorithm prioritizes the BC with the highest *tps*, as there is no impact on the cost, and the transaction is included faster.

It is important to notice that the bcSet is a pre-processed set based on the data to be stored. For example, if the size of the data to be stored is above the maximum data supported by a BC transaction, then, this BC is not appended in the bcSet.

Algorithm 2 Most Economic BC Selection

```
1: procedure SELECTECONOMICBC(bcSet, costs)
2:     economic ← bcSet[0]
3:     for all bc ∈ bcSet do
4:         if (costs[bc] < costs[economic]) then
5:             economic ← bc
6:         else
7:             if (costs[bc] == costs[economic]) and (bc.tps > economic.tps) then
8:                 economic ← bc
9:             end if
10:        end if
11:    end for
12:    return economic
13: end procedure
```

IV. Evaluation and Discussion

In order to verify PleBeuS's functionality regarding the policy switching mechanism, and the BC selection algorithm, a prototype was implemented and evaluated in three scenarios with different sets of defined policies. These scenarios are defined in Section IV-A, and the results from each scenario are presented and discussed in Section IV-B. PleBeuS's code and datasets are available online [18].

A. Evaluation Scenarios

Each scenario combined both public and private BCs, and varied in parameters, such as the BC Set (*i.e.,* BCs available for selection), costs thresholds intervals (*e.g.,* daily, weekly, monthly, and yearly), which trigger a policy switch, and the profile (*e.g.,* performance or economic). The time frame parameter was not considered because the cost interval triggers the switch of policies. The evaluation scenarios and the defined policies are described in the next sections.

1) Scenario #1 - Private BCs: For the first scenario, six policies were defined (see Table II) with the BC Set containing only private BC implementations (*e.g.,* HyperLedger (HYP), Multichain (MLC) and Postgres (PSG)). PSG was defined as the BC for the default policy. Even though private BCs do not require transaction fees as public BCs do, they incur other costs, such as hardware and server maintenance, and support. Thus, an arbitrary cost of CHF 0.01 per transaction was considered, triggering the switch of policies.

TABLE II: Policies Defined in Evaluation Scenario #1

No.	BC Set	Interval	BC Type	Cost [CHF]	Profile
1	All private BC	daily	private	5	performance
2	HYP, MLC	daily	private	8	economic
3	MLC, PSG	weekly	private	20	performance
4	HYP, MLC	monthly	private	50	performance
5	All private BC	yearly	private	100	economic
6	PSG	default	-	-	-

2) Scenario #2 - Public BCs: In the second evaluation scenario, only public BC (*e.g.,* Bitcoin (BC), Ethereum (ETH), Stellar (XLM), EOS (EOS), and IOTA (MIOTA)) were selected to compose the BC Set. Nine policies were defined in this scenario (see Table III), in which the profile alternated between Performance and Economic to evaluate both parameters in a balanced manner. Ethereum was arbitrary selected as the BC for the default policy.

TABLE III: Policies Defined in Evaluation Scenario #2

No.	BC Set	Interval	BC Type	Cost [CHF]	Profile
1	All public BC	daily	public	2	performance
2	BTC, ETH, EOS	daily	public	10	economic
3	EOS, IOTA, XLM	weekly	public	15	performance
4	BTC, EOS	weekly	public	30	economic
5	ETH, EOS, IOTA	monthly	public	40	performance
6	BTC, EOS, XLM	monthly	public	60	economic
7	ETH, EOS	yearly	public	80	performance
8	All public BC	yearly	public	100	economic
9	ETH	default	-	-	-

3) Scenario #3 - Private and Public BCs: The last scenario combined both public and private BCs to evaluate the behavior of `PleBeuS` in heterogeneous environments. Seven policies were defined (presented in Table IV), with Postgres being the selected BC for the default policy. Moreover, to increase performance, the split parameter was set to true, meaning that the data will be stored in the BC Set in a round-robin scheme.

TABLE IV: Policies Defined in Evaluation Scenario #3

No.	BC Set	Interval	BC Type	Cost [CHF]	Profile	Split
1	All private BC	daily	private	15	performance	✓
2	All private BC	daily	private	30	economic	✓
3	All BC	weekly	indifferent	1500	performance	✓
4	All public BC	monthly	public	4000	performance	✓
5	All public BC	monthly	public	8000	economic	✓
6	All BC	yearly	indifferent	15000	economic	✓
7	PSG	default	-	-	-	-

B. Results and Discussion

For each scenario, 10000 `storeData` requests were sent to `PleBeuS` API, which generated a total of 10000 BC transactions. Each request represented one collected random data point from a generic sensor, *e.g.,* temperature, or humidity. The performance (*i.e.,* tps) of the public BCs followed the lower spectrum of the values from Table I. For the private BCs, the values from the performance evaluation conducted in [22] were selected. The results of the conducted evaluation in each scenario are depicted in Figure 2.

The policy switching mechanism is influenced by two parameters, *(i)* cost threshold, and *(ii)* time frame. As the time frame was not considered in the evaluation, the switch in the active policy was determined by the cost threshold. Figure 2a, Figure 2d, and Figure 2g represent the number of transactions executed per policy in the scenarios. It can be seen in Figure 2a that the default policy (No. 6) is never activated in scenario #1 because the 10000 transactions were executed within the yearly cost interval (*i.e.,* 100 CHF) with policy No. 5 active for half of the total transactions, *i.e.,* 5000 transactions. Figure 2b confirms this behavior, depicting the cost thresholds for each policy, where the cost for policy No. 6 is exactly 50 CHF. Even though the same yearly cost was defined in scenario #2, the switching occurred in a different pattern (see Figure 2d) due to the higher costs of interacting with public BCs. As soon as the accumulated cost reaches the yearly threshold in the last policy (No. 8), the default one is activated, and the accumulated cost reaches 400 CHF.

The results of the evaluation of the BC selection algorithms (*cf.* Section III-C), are illustrated in Figure 2c, Figure 2f, and Figure 2i. The first two figures (2c and 2f) show a clear distinction among the BCs selected due to the selected deployment type private BCs and public BCs. Moreover, the Bitcoin BC is not selected due to its high transaction costs because of its high prices (10000 USD as of September 22, 2019 [10]). However, in the third scenario (Figure 2i), with the split parameter selected, the transactions were sent to all available BCs, with Postgres being selected 3500 times because it presents the fastest tps and cheapest costs because it is a regular database. Despite the inclusion of Postgres in the BC set, the split parameter guaranteed that others BC were selected with a tendency to private BCs being selected for the *Economic* profile. This tendency is confirmed because more than half of the 10000 transactions, approximately 6700 transactions, were included in HyperLedger, Multichain, and Postgres.

The results of the evaluation showed that if the BC set contains the same deployment type, the selection process is able to correctly minimize costs (*e.g.,* not selecting Bitcoin) or maximize performance (*e.g.,* selecting a generic database). However, if the BC set contains both private and public BCs, the priority is always given to private BCs because they present a higher tps in comparison to public BCs. Moreover, the prices of interacting with public BCs must be taken into consideration when selecting this deployment type. Sending a transaction to a public BC for each sensor measurement is unfeasible, as the cryptocurrencies prices are higher compared to private BCs. Thus, measurements should be combined and only an average from the last n measurements included in a transaction to minimize costs.

Even though private BCs are prioritized in the selection mechanism, one must take into consideration the degree of centralization that this type of deployment introduces. As described in Section II-A, private BCs are controlled by a single entity or a group of trusted entities. Thus, are not decentralized, and the property of immutability might be broken if these entities collude or the central entity behaves maliciously.

V. Related Work

The BC area is a novel research topic; thus, few work in the literature tackle the specific problem of selecting a BC among the myriad of implementations.

A complete survey of the different factors that impact the BC choice is presented in [2]. The authors detail different BC aspects (*e.g.,* basic definitions, consensus mechanisms, account models, and interaction roles) by surveying the scientific publications to provide a *vademecum*, *i.e.,* complete manual and reference guide regarding the BC technology. However, instead of providing a solution where a BC is automatically selected based on specific requirements, the authors present a flowchart that users must manually follow to decide when to use and which type of BC (*e.g.,* permissionless, full-permissioned, or open-permissioned) is the most appropriate.

Furthermore, [14] proposes a framework to determine the most appropriate BC based on defined requirements by users. The proposed framework monitors the BCs and calculates scores, using weights, for each of the defined metrics, *e.g.,* transaction throughput, reputation, and costs, in order to select the most suitable one. Moreover, they introduce a *Switchover* mechanism to copy data from one BC to the other. Even though its approach is close to the solution provided in this paper, it is still necessary to manually adjust the weights of each metric to correspond to the user's requirements, whereas `PleBeuS` reduces manual interaction.

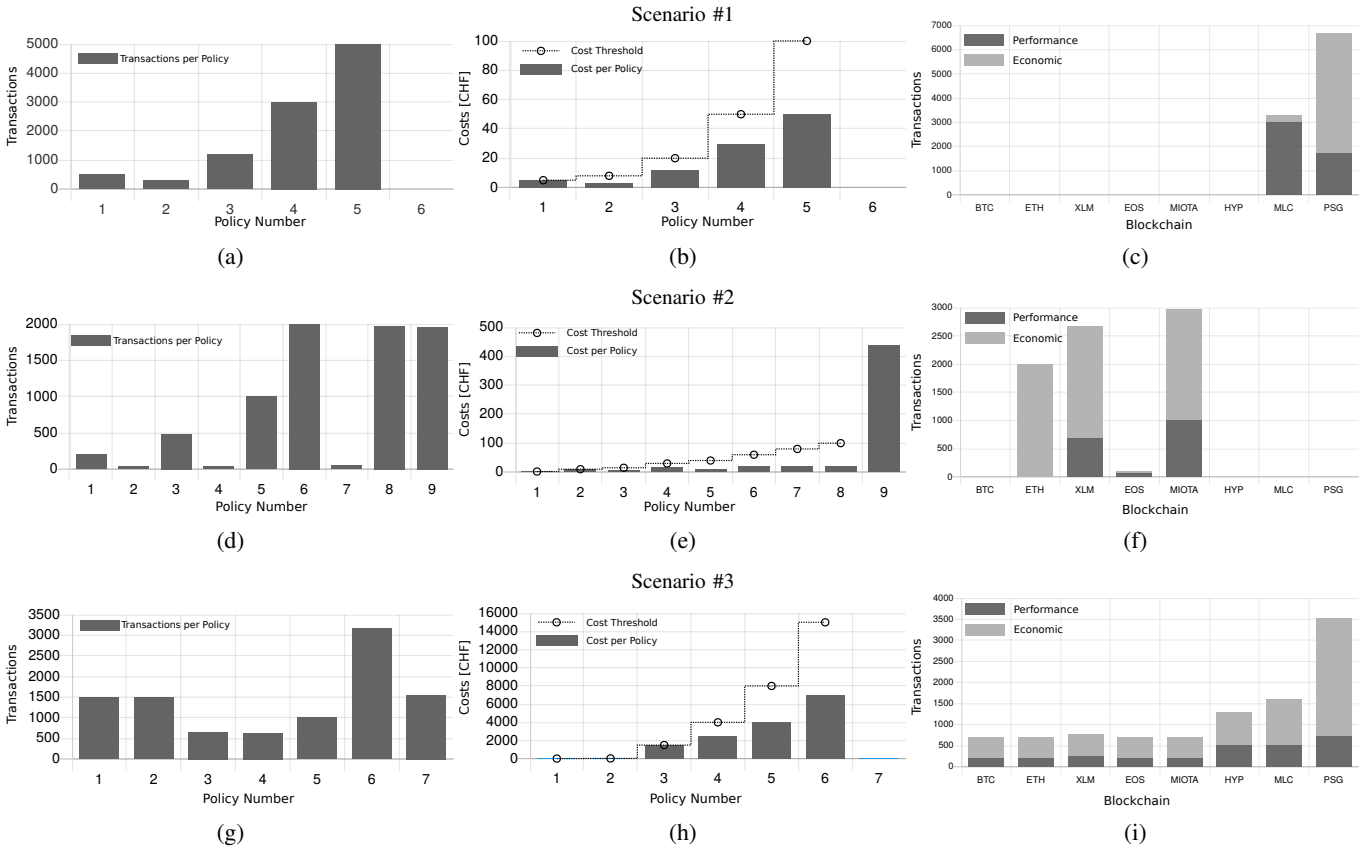

Fig. 2: Comparison of the Results in the Evaluation Scenarios, Where (a), (d) and (g) Represent Transactions Executed per Policy, (b), (e) and (h) Represent Costs per Policy, and (c), (f) and (i) Represent the Transactions Stored per BC

VI. Summary and Future Work

This paper presented `PleBeuS`, a Policy-based Blockchain Selection framework. `PleBeuS` automatically selects the most suitable Blockchain (BC) implementation that meets the requirements of users specified in the form of policies. Moreover, `PleBeuS` is able to communicate with a BC Interoperability API [22] to include the data to the selected BC. `PleBeuS` implements: *(i)* a policy switching mechanism aware of cost thresholds and time frames, and *(ii)* two BC selection algorithms that either prioritize the BC that presents the best performance in terms of transaction per seconds (tps) or the BC that presents the lowest estimated transaction price retrieved from external solutions.

`PleBeuS` was evaluated in three scenarios that contained policies that differed in cost thresholds, selection profiles (*e.g.,* performance or economic), and BC sets (*e.g.,* private BCs, public BCs, and both public and private BCs). This evaluation demonstrated that the policy switching mechanism and the BC selection algorithms behaved as expected, switching policies upon a cost threshold is reached and selecting the fastest BC implementation from a set of defined BC. `PleBeuS` can be applied in use cases where the data immutability provided by the BC is necessary, while costs should be minimized or performance constraints met, such as in the cold-chain of medicines use case [23].

Furthermore, the evaluation showed that private BCs have priority in the selection algorithm over public BCs. This behavior is because the algorithm solely relies on objective parameters, such as the number of tps. Private BCs implementations are able to achieve higher tps compared to public BCs due to their consensus mechanisms being designed to reach a consensus about the BC state as fast as possible because the set of validators are known and trusted. Thus, this behavior might change if the selection algorithm considers other non-objective parameters, such as the degree of centralization or the possibility of collusion between validators.

Future work considers multi-user interaction, improvements in the selection algorithm to take into consideration more BC parameters (*e.g.,* popularity, code maturity, BC stability, and security), detecting policy conflicts, and conducting performance and scalability evaluation of the whole selection process and overall framework.

Acknowledgments

This work has been supported by Komission für Technologie und Innovation (KTI), Switzerland (CTI-No. 26996.1 PFES-ED). Also, the authors would like to acknowledge manifold discussions with modum.io, especially Stefan M. Weber, Sacha Uhlmann (modum.io), and Thomas Bocek from the Hochschule für Technik Rapperswil (HSR).

REFERENCES

[1] M. Alharby, A. Aldweesh, and A. v. Moorsel, "Blockchain-based Smart Contracts: A Systematic Mapping Study of Academic Research (2018)," in *International Conference on Cloud Computing, Big Data and Blockchain (ICCBB 2018)*, Fuzhou, China, November 2018, pp. 1–6.

[2] M. Belotti, N. Boi, G. Pujolle, and S. Secci, "A Vademecum on Blockchain Technologies: When, Which and How," in *IEEE Communications Surveys Tutorials*, 2019, pp. 1–47.

[3] T. Bocek, B. B. Rodrigues, T. Strasser, and B. Stiller, "Blockchains Everywhere - a Use-Case of Blockchains in the Pharma Supply-Chain," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2017)*, Lisbon, Portugal, May 2017, pp. 772–777.

[4] V. Buterin, "A Next-Generation Smart Contract and Decentralized Application Platform," 2014, https://github.com/ethereum/wiki/wiki/White-Paper, Last visit March 20, 2019.

[5] M. Charalambides, P. Flegkas, G. Pavlou, A. K. Bandara, E. C. Lupu, A. Russo, N. Dulav, M. Sloman, and J. Rubio-Loyola, "Policy Conflict Analysis for Quality of Service Management," in *IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, Stockholm, Sweden, Sweden, June 2005, pp. 99–108.

[6] M. Charalambides, P. Flegkas, G. Pavlou, J. Rubio-Loyola, A. Bandara, E. Lupu, A. Russo, M. Sloman, and N. Dulay, "Dynamic Policy Analysis and Conflict Resolution for DiffServ Quality of Service Management," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, Vancouver, BC, Canada, January 2006, pp. 294–304.

[7] B. Chase and E. MacBrough, "Analysis of the XRP Ledger Consensus Protocol," 2018, http://arxiv.org/abs/1802.07242, Last visit June 19, 2019.

[8] S. Chen, J. Zhang, R. Shi, J. Yan, and Q. Ke, "A Comparative Testing on Performance of Blockchain and Relational Database: Foundation for Applying Smart Technology into Current Business Systems," in *Distributed, Ambient and Pervasive Interactions (DAPI 2018)*, Las Vegas, NV, USA, July 2018, pp. 21–34.

[9] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," in *IEEE Access*, vol. 4, May 2016, pp. 2292–2303.

[10] CoinMarketCap, "Coinmarketcap market capitalizations," 2019, https://coinmarketcap.com/, Last visit September 22, 2019.

[11] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "BlockChain: A Distributed Solution to Automotive Security and Privacy," in *IEEE Communications Magazine*, vol. 55, no. 12, December 2017, pp. 119–125.

[12] European Commission, "General Data Protection Regulation (GDPR) - What personal data is considered sensitive?" 2016, , https://tinyurl.com/yyrdetod , Last visit August 13, 2019.

[13] S. D. Foundation, "Fees — Stellar Developers," 2019, , https://www.stellar.org/developers/guides/concepts/fees.html , Last visit August 30, 2019.

[14] P. Frauenthaler, M. Borkowski, and S. Schulte, "A Framework for Blockchain Interoperability and Runtime Selection," 2019, , http://arxiv.org/abs/1905.07014 , Last visit August 13, 2019.

[15] W. Gao, W. G. Hatcher, and W. Yu, "A Survey of Blockchain: Techniques, Applications, and Challenges," in *27th International Conference on Computer Communication and Networks (ICCCN 2018)*, Hangzhou, China, July 2018, pp. 1–11.

[16] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the Security and Performance of Proof of Work Blockchains," in *ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*, Vienna, Austria, October 2016, pp. 3–16.

[17] B. Inc., "Script - Bitcoin Wiki," 2019, https://en.bitcoin.it/wiki/Script, Last visit August 19, 2019.

[18] D. Lakic, E. J. Scheid, and B. Rodrigues, "PleBeuS: a Policy-based Blockchain Selection Framework," 2019, https://gitlab.ifi.uzh.ch/scheid/plebeus, Last visit December 14, 2019.

[19] E. C. Lupu and M. Sloman, "Conflicts in policy-based distributed systems management," in *IEEE Transactions on Software Engineering*, vol. 25, no. 6, November 1999, pp. 852–869.

[20] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "RFC 3060 - Policy Core Information Model," https://tools.ietf.org/html/rfc3060, Last visit July 12, 2019.

[21] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009, https://bitcoin.org/bitcoin.pdf, Last visit June 19, 2019.

[22] E. J. Scheid, T. Hegnauer, B. Rodrigues, and B. Stiller, "Bifröst: a Modular Blockchain Interoperability API," in *IEEE Conference on Local Computer Networks (LCN 2019)*, Osnabrück, Germany, October 2019, pp. 1–9, Accepted. To be published.

[23] E. J. Scheid, B. Rodrigues, and B. Stiller, "Toward a Policy-based Blockchain Agnostic Framework," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2019)*, Washington - DC, USA, April 2019, pp. 609–613.

[24] R. Shaan, "*Difference Between Traditional and Delegated Proof of Stake*," April 2018, , https://bit.ly/2JWIn3f , Last visit May 5, 2019.

[25] M. Sloman, "Policy driven Management for Distributed Systems," vol. 2, no. 4. Springer, 1994, pp. 333–360.

[26] J. Strassner, *Policy-Based Network Management: Solutions for the Next Generation (The Morgan Kaufmann Series in Networking)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[27] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform," in *IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2018)*, Milwaukee, WI, USA, September 2018, pp. 264–276.

[28] F. Tian, "An Agri-Food Supply Chain Traceability System for China based on RFID Blockchain Technology," in *13th International Conference on Service Systems and Service Management (ICSSSM 2016)*, Kunming, China, June 2016, pp. 1–6.

[29] D. C. Verma, "Simplifying Network Administration Using Policy-Based Management," in *IEEE Network*, vol. 16, no. 2, March 2002, pp. 20–26.

[30] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks," in *IEEE Access*, vol. 7, January 2019, pp. 22 328–22 370.

[31] K. Wst and A. Gervais, "Do you Need a Blockchain?" in *Crypto Valley Conference on Blockchain Technology (CVCBT 2018)*, Zug, Switzerland, June 2018, pp. 45–54.

[32] P. Zhong, Q. Zhong, H. Mi, S. Zhang, and Y. Xiang, "Privacy-Protected Blockchain System," in *IEEE International Conference on Mobile Data Management (MDM 2019)*, Hong Kong, Hong Kong, June 2019, pp. 457–461.