

A Framework for Blockchain Interoperability and Runtime Selection

Philipp Frauenthaler*, Michael Borkowski*, Stefan Schulte*

* *Distributed Systems Group, TU Wien, Vienna, Austria*

{p.frauenthaler, m.borkowski, s.schulte}@infosys.tuwien.ac.at

Abstract—The suitability of a particular blockchain for a given use case depends mainly on the blockchain’s functional and non-functional properties. Such properties may vary over time, and thus, a selected blockchain may become unsuitable for a given use case. This uncertainty may hinder the widespread adoption of blockchain technologies in general.

To mitigate the impact of volatile blockchain properties, we propose a framework that monitors several blockchains, allows the user to define functional and non-functional requirements, determines the most appropriate blockchain, and enables the switchover to that chain at runtime. Our evaluation using a reference implementation shows that switching to another blockchain can save cost and enable users to benefit from better performance and a higher level of trust.

Index Terms—Blockchain interoperability, blockchain metrics, runtime selection

I. INTRODUCTION

In the past few years, cryptocurrencies have gained significant public attention [13, 28]. The first and most prominent cryptocurrency is Bitcoin, proposed in 2008 by Satoshi Nakamoto [8, 19, 24]. While blockchains have proven to be suitable as distributed ledgers for recording transactions in Bitcoin and other cryptocurrencies [20, 24], blockchain technologies have also the potential to be applied in other use cases, e.g., the Internet of Things or business processes [11, 17, 22].

The suitability of a particular blockchain for a given use case depends on various properties, e.g., the cost of writing data into that blockchain, the time until a data record is permanently included and thus remains unchanged with sufficient probability, the transaction throughput, the network’s overall hash rate, or the distribution of the hash power among miners and mining pools [27]. Blockchain properties vary over time, e.g., the network hash rate may decrease¹. Variations of properties may cause a blockchain to become unsuitable for a given use case and, in further consequence, may hinder the widespread adoption of blockchain technologies in general, since the uncertain suitability for a given use case in the future poses significant risk for engineers evaluating the utilization of blockchains [1].

To facilitate the adoption of blockchain technologies, we introduce a general-purpose framework for storing arbitrary data on blockchains. The framework abstracts technical details and offers interfaces for reading data from and writing data

into multiple blockchains. To mitigate the impact of volatile blockchain properties, the proposed framework provides a switchover functionality allowing to switch to another, more beneficial blockchain at runtime. The framework monitors multiple blockchains, calculates their individual benefits and determines the most beneficial one based on user-defined requirements. Furthermore, the framework is able to react to various events such as a rapid decrease of a blockchain network’s hash rate or a steadily increase of the cost of writing data into a blockchain. Beyond volatile blockchain properties, the proposed framework is also able to meet changing user demands by selecting a more appropriate blockchain. The combination of the blockchain selection algorithm and the switchover functionality enables users to benefit from low cost, better performance, and a higher level of trust. We use a reference implementation supporting Bitcoin, Ethereum, Ethereum Classic, and Expanse to evaluate our framework.

Summarizing, the contributions of our work are as follows:

- We identify concrete metrics relevant for the monitoring and the runtime selection of blockchains.
- We propose a mechanism for determining the most beneficial blockchain based on user-defined requirements.
- We describe the requirements and the technical design of the proposed framework.
- We evaluate the benefits of the proposed framework in terms of cost, performance, and trust using a reference implementation.

The remainder of this paper is structured as follows: In Section II, we further motivate our work. Section III presents relevant blockchain metrics, a mechanism for determining the most beneficial blockchain, and the switchover functionality. Section IV provides an evaluation of the presented work and Section V gives an overview of related work. Finally, Section VI concludes the paper.

II. MOTIVATION

Our work aims to overcome issues regarding volatile blockchain properties, changing user demands, and the selection of an appropriate blockchain among many. In the following, we elaborate on these issues in more detail.

In the recent past, cryptocurrency users witnessed several price fluctuations, e.g., Bitcoin’s market price reached an all-time high close to 20,000 USD in December 2017 [3] and declined about 75% from this peak until November 2018 [9].

¹<https://etherscan.io/chart/hashrate>

The sensitivity of cryptocurrencies for price fluctuations influences the cost of writing data into a blockchain, e.g., through varying transaction fees.

Furthermore, in November 2018, the community has witnessed a “hash war” between the supporters of two competing hard forks of Bitcoin Cash (Bitcoin Cash ABC and Bitcoin Cash SV) [21]. To prevent the own fork to get damaged or even destroyed by the competitors, both opposing sides collected as much hash power as possible, leading to a centralization of both hard forks [12]. Additionally, a considerable amount of hash power has been shifted from Bitcoin to Bitcoin Cash during the peak time of the “war”. Due to this shift, Bitcoin’s hash rate decreased by seven percent [21]. In general, a decreasing network hash rate may lead to a loss of trust in a blockchain since it becomes easier for malicious nodes to get control over more than 50% of the overall hash power, enabling them to perform a 51% attack [15]. Since changes leading to forks are constantly encouraged by community members, such conflicts may also emerge in the future.

Another important aspect of a blockchain is its degree of decentralization. Many miners collude with others through mining pools [20]. If the number of participating nodes grows over time, the pool’s overall hash rate increases as well. Mining pools concentrating more than 50% of the hash power can perform any strategy available to a single majority miner [8].

In blockchain networks, all transactions are replicated on every network node, increasing storage requirements and thus affecting scalability [27]. Currently, on average, public blockchains like Bitcoin or Ethereum can only process 3–20 transactions per second [26]. Despite multiple attempts to solve this problem (e.g., increasing the size limit of Bitcoin blocks or transferring values off-chain by using the Lightning Network²), scalability is still an open issue [14, 26]. Thus, an increasing workload may raise the time it takes until new transactions are mined. Additionally, users may compete more intensively to get their transactions included in one of the next blocks, possibly leading to higher transaction fees. Since it is difficult to predict the workload a blockchain is confronted with, the progression of cost and performance is unclear.

A blockchain project aiming to improve scalability is Corda, developed by the R3 consortium³. The consortium addresses the scalability problem by reducing the replication of transactions across network nodes. This approach may negatively affect availability and data integrity, but also improve privacy [27]. Corda is not the only blockchain project addressing particular requirements. Since the advent of Bitcoin in 2008, a diverse range of blockchains has emerged, resulting in solutions with many different features and configurations. Differences range from cost efficiency, storage and performance to decentralization and access restrictions [27].

Due to their various features and properties, different blockchains are not equally suitable for a given use case,

inevitably leading to the question which blockchain meets the requirements of a user to the largest extent [27]. As described above, blockchain properties vary over time. Such variations may impact the suitability of a particular blockchain for a given use case. Assuming a decentralized application relied on Bitcoin Cash during the “hash war”, application users would have been completely reliant on a few miners, contradicting the requirement of decentralization. In such a case, it may be appropriate to switch to another blockchain providing similar features along with a higher degree of decentralization. Furthermore, also user requirements may change over time and thus, another blockchain may become more appropriate for a certain use case [16], e.g., demanding access restrictions offered by permissioned blockchains.

Summarizing, engineers seeking to utilize a blockchain for their applications face a diverse range of blockchain technologies with different features and configurations. Furthermore, a former technology decision may become outdated due to variations of blockchain properties or changing user demands. In order to overcome these issues, a solution is required that monitors multiple blockchains, determines the most appropriate one based on user preferences, and enables a switchover between blockchains at runtime. Such a framework is supposed to continuously monitor several blockchains. In case another blockchain becomes more appropriate than the currently used one, the framework is expected to suggest switching to that chain, i.e., to route subsequent operations (e.g., reading or writing data) to the new blockchain. Additionally, during the switchover, a user-defined amount of data stored on the currently used blockchain could be moved to the target chain. This, for instance, may be essential if a certain amount of data is needed on the target blockchain for further processing or in case the community is losing trust in the currently used blockchain. In the next section, we will introduce how our framework implements the needed functionalities.

III. APPROACH

In order to address the requirements outlined in the previous section, the proposed framework consists of three main components: The *Monitoring Component* continuously surveys information about each supported blockchain and calculates metric values. Based on these metric values, the *Blockchain Selection Algorithm* calculates each blockchain’s benefit and selects the most beneficial one. In case another blockchain is more beneficial than the currently used one, a switchover is suggested. The *Switchover Component* provides the possibility to switch from one blockchain to another.

In the following, we first introduce in Section III-A the blockchain metrics supported by the Monitoring Component. In Section III-B, we discuss the Blockchain Selection Algorithm, while Section III-C presents the functionality of the Switchover Component. Finally, the technical design of the framework is discussed in Section III-D.

²<https://lightning.network/>

³<https://www.r3.com/>

A. Blockchain Metrics

In order to select the most appropriate blockchain, users should be able to define particular selection metrics, which are then applied in order to assess how a blockchain matches the needs of the user. In the following, we present eight blockchain metrics relevant for the comparison of different blockchains. They can be categorized into cost-related metrics (M1-3), performance-related metrics (M4-5), security-related metrics (M6-7), and reputation (M8). Notably, the framework presented in this paper allows to extend the metric model by further metrics, if necessary. Therefore, the discussed metrics should be considered as exemplary. The main requirement for a metric to be added to the metric model is that it is measurable (which is the case for the cost, performance, and security metrics below) or can be defined by the user (which is the case for reputation metrics).

1) *Cost of writing 1 KB of data into a blockchain (M1)*: This metric represents the cost of writing one kilobyte of data into a blockchain (in USD). Since many blockchains allow their users to prioritize transactions by specifying transaction fees, the cost may vary depending on the fees the user is willing to pay. The calculation is based on the transaction fees provided by the framework user. In case no fees are provided, the framework automatically determines fees that will cause submitted transactions to get included within a pre-defined number of blocks. Since we exemplarily use Bitcoin, Ethereum, Ethereum Classic, and Expanse in the reference implementation, we set this number to six as a trade-off between cost and performance. In case other blockchains are connected to the framework, a different block number may be more appropriate. By using the introduced metric, the framework can determine the cheapest blockchain. We have selected one kilobyte as basis for the calculation of this metric, since this amount is sufficient to store various kinds of meta data, e.g., log events. In case another amount is more appropriate, it can be changed without any restrictions.

2) *Cost of retrieving 1 KB of data from a blockchain (M2)*: This metric specifies the cost of retrieving one kilobyte of data from a blockchain (in USD). Typically, reading data from a blockchain is free of charge. Nevertheless, we introduce this metric since the proposed framework is intended to be applicable for a wide range of use cases, including possible invocations of non-read-only smart contract methods for retrieving data from a blockchain (e.g., access logging). This metric enables the framework to compare different blockchains regarding cost of retrieving data. Analogous to M1, the amount of data used for the calculation can be changed without any restrictions.

3) *Exchange rates (M3)*: This metric represents the current exchange rate between USD and the native cryptocurrency of a particular blockchain, e.g., the market price for one Bitcoin in USD. Exchange rates are required for calculating the cost of interacting with a particular blockchain.

4) *Inter-block time (M4)*: The inter-block time specifies the rolling average of the time (in seconds) it takes to mine a block and is calculated on the basis of all blocks that have

TABLE I
METRIC DATA TYPES.

Metric	Data type
M1	Decimal ≥ 0
M2	Decimal ≥ 0
M3	Decimal ≥ 0
M4	Decimal ≥ 0
M5	Decimal ≥ 0
M6	Mapping (key: string, value: decimal ≥ 0)
M7	Decimal ≥ 0
M8	Integer ≥ 0 and ≤ 10

been mined during the last 24 hours. The inter-block time is used as an indicator of a blockchain’s performance.

5) *Transaction throughput (M5)*: The transaction throughput represents the rolling average of the number of transactions that are processed per second and is calculated based on all transactions that have been mined during the last 24 hours. Analogous to M4, this metric is used to observe a blockchain’s performance.

6) *Degree of decentralization (M6)*: This metric specifies the distribution of the network’s hash power among miners or mining pools. The framework provides a mapping between miner addresses and their proportion of mined blocks. The proportion is specified in percent and is calculated from the blocks that have been mined during the last 24 hours. In case of Ethereum-based blockchains, uncle blocks are also taken into account. This metric allows the identification of miners that control large amounts of hash power. Miners controlling more than 50% of a network’s hash power can tamper with the blockchain, since they are able to generate more blocks, enabling them to master the longest chain [18].

7) *Network hash rate (M7)*: This metric specifies the hash rate the network has performed in the recent 24 hours. The hash rate is computed from the current difficulty and from the blocks that have been mined during the last 24 hours. In case of Ethereum-based blockchains, uncle blocks are also taken into account. This metric allows to observe the progression of the network’s hash rate, enabling the identification of significant declines.

8) *Reputation (M8)*: The reputation is an integer value between 0 and 10, and indicates the degree of renown a blockchain is associated with. It may reflect various properties such as trust, frequency of new feature releases, number of forks, community consensus and controversies, security concerns, etc. The value 0 indicates the worst reputation, whereas the value 10 represents an excellent reputation. This metric is introduced to compare different blockchains by their renown.

Metrics M1–M7 are calculated automatically by the framework’s Monitoring Component. For M8, manual user input is required, since this metric highly depends on the subjective assessment of the framework user.

An overview of each metric’s data type is given in Table I.

TABLE II
WEIGHTS USED BY THE FRAMEWORK.

Weight	Meaning
0	No importance
1	Very low importance
2	Low importance
3	Medium importance
4	High importance
5	Very high importance

B. Blockchain Selection Algorithm

Next, we introduce concepts for comparing different blockchains and for selecting the most appropriate one. To give the user the opportunity to define which metrics are of high and low importance, respectively, we first introduce a weighted ranking system used to calculate a blockchain’s benefit. For this, each blockchain metric is assigned a user-defined weight indicating its importance when calculating an overall ranking of blockchains. Table II shows six possible weights offered by the framework.

Furthermore, the user has to specify a *Score Assignment Function* (SAF) for each blockchain metric. The SAF maps a concrete metric value to a score, as shown in (1). D_i denotes the data type of metric i .

$$\text{SAF} : D_i \mapsto \{0, 1, 2, 3, 4\} \quad (1)$$

The SAF is applied in order to normalize different data types and ranges of data. Through this, it is possible to combine the different metrics and use them for a weighted ranking of blockchains, despite the fact that the data ranges and data types of the single metrics differ.

The score calculated by the SAF quantifies how well a metric satisfies a certain property, e.g., an inter-block time of 100 seconds may be rewarded with a score of 2. A complete example for how a SAF could be defined by the user is given in Section IV.

In Table III, five possible score values and their meanings are presented. The SAF of a metric is applied to the corresponding metric value of each supported blockchain. Assuming the framework supports Bitcoin and Ethereum, the SAF of M1 is applied to the cost of writing data into the Bitcoin blockchain and to the cost of writing data into the Ethereum blockchain. In a next step, the SAF of M2 is applied to the cost of reading data from the Bitcoin blockchain and to the cost of reading data from the Ethereum blockchain. The same procedure is repeated for M3–M8. In the presented example, the entire process results in two score values for each metric (one for Bitcoin and one for Ethereum). By providing weight and score assignments, the user is able to customize the internal logic of the framework to meet desired needs.

The benefit of a blockchain \mathcal{B} is calculated by summing up each metric’s weighted score, as shown in (2), where n denotes the number of blockchain metrics, weight_i represents the user-defined weight of metric M_i and $\text{score}_{\mathcal{B}[i]}$ dubs the score value

TABLE III
SCORE DEFINITIONS USED BY THE FRAMEWORK.

Score	Meaning
0	Does not satisfy
1	Partly satisfies
2	Substantially satisfies
3	Almost satisfies
4	Fully satisfies

TABLE IV
AN EXAMPLE OF A WEIGHTED RANKING WITH TWO BLOCKCHAINS.

Metric	Weight	Blockchain A		Blockchain B	
		Score	W. Score	Score	W. Score
M1	5	4	20	3	15
M2	3	4	12	4	12
M3	4	4	16	2	8
M4	5	2	10	4	20
M5	3	3	9	3	9
M6	3	3	9	3	9
M7	5	3	15	4	20
M8	4	3	12	2	8
Total	32	26	103	25	101

of metric M_i for blockchain \mathcal{B} (obtained by applying the SAF to the value of metric M_i for \mathcal{B}).

$$\sum_{i=1}^n \text{weight}_i \cdot \text{score}_{\mathcal{B}[i]} \quad (2)$$

The presented formula is applied for each supported blockchain. The blockchain with the highest benefit is chosen as the most beneficial one. Table IV shows an example of a weighted ranking, where two blockchains are evaluated. Blockchain A has a benefit (total weighted score) of 103, whereas Blockchain B has a benefit of 101. Therefore, blockchain A is considered as more beneficial.

The introduced weighted ranking system allows the quantification of a blockchain’s benefit on the basis of user-defined weights and score assignments. However, it does not offer a mechanism for enforcing certain requirements (e.g., an inter-block time lower than or equal to 60 seconds) a blockchain *must* fulfill under all circumstances, regardless of its benefit. The example shown in Table IV outlines a situation where Blockchain A has a worse score for M4 (inter-block time) than Blockchain B, but A is still the most beneficial one due to the scores of other metrics.

However, it might be the case that it is of utmost importance to the user that for a particular metric a certain threshold is met, e.g., in the example just mentioned, that the inter-block time (M4) has at most a value of 60 seconds. Thus, the Blockchain Selection Algorithm is adapted to consider only those blockchains that fulfill such additional requirements, i.e., only those blockchains that satisfy these additional requirements serve as candidates for the weighted ranking system. All other blockchains are not further regarded.

For that purpose, we introduce the *Metric Validation Function* (MVF). As shown in (3), this function maps an 8-tuple to a 9-tuple. D_i represents the data type of metric M_i .

$$\text{MVF} : D_1 \times \dots \times D_8 \mapsto \{\text{true}, \text{false}\}^9 \quad (3)$$

The left-most (first) value of the returned 9-tuple indicates whether metric M1 is valid (i.e., whether it satisfies user requirements), the second value indicates whether metric M2 is valid and so forth. The last (right-most) value of the 9-tuple represents the validity of a blockchain and may be the result of a combination of the single boolean values, e.g., linking the single boolean values to a propositional formula. This enables the user to specify more complex requirements. The single boolean values are needed for further decisions during the switchover (see Section III-C). The concrete implementation of the proposed function has to be provided by the user.

C. Switchover Functionality

As described in Section II, a switchover is the process of routing all subsequent operations (e.g., read and write operations) to another blockchain. Depending on user preferences, the switchover is either performed fully automated once a more beneficial blockchain is detected, or has first to be approved by the user.

Furthermore, the framework allows the user to define the amount of already existing data records that should be moved from the currently used blockchain to the destination chain during a switchover. This amount may depend on the metric(s) that caused the switchover. For instance, if the community is losing trust in the currently used blockchain, it may be essential to transfer *all* data stored on the currently used blockchain or at least data of a specific period of time to the destination blockchain.

In order to customize the framework’s logic for determining the amount of data that should be transferred, the user can specify a custom strategy. Whenever a more appropriate blockchain is detected, i.e., a switchover is suggested by the framework, this custom strategy is triggered. The framework forwards each metric’s weighted score and the validation results (obtained from the MVF) of both the currently used blockchain and the suggested chain to the user-defined strategy. The presence of this information enables the user to define a strategy that is able to determine the amount of data on the basis of those metrics that cause the currently used chain to be less appropriate than the suggested one.

The amount of data to be transferred is specified by a date range. For instance, if the user-defined strategy specifies a range between 01.02.2019 and 28.02.2019, all data records mined during this range are copied to the destination chain. It should be noted, that – while we use the term “transfer data” to describe that data is copied from one blockchain to another – data can of course not be deleted from the original blockchain. The goal of data transfers is merely to make sure that no data gets lost. For instance, if the number of miners for a particular blockchain rapidly decreases and this is reflected in a very low degree of decentralization (M6), the chance is

very high that a malicious, powerful attacker can perform a 51% attack on the blockchain, thus rendering the data in the blockchain useless.

In order to prevent the framework from performing multiple switchovers within a short period of time, e.g., due to frequent variations of the order in the weighted ranking system, we introduce a *switchover suppression period*. This period can be defined by the user. The framework suppresses subsequent switchover suggestions until the switchover suppression period elapses, regardless of how many changes are occurring in the ranking system. Thus, at most one switchover is suggested every period, preventing immediate switchovers between blockchains. If no timespan is defined, the framework starts the switchover immediately after a more beneficial blockchain has been detected (and automatic switchovers are enabled).

D. Technical Design

Figure 1 presents an overview of the framework’s architecture. The proposed framework consists of the *Core Logic* and a number of *Blockchain Proxies*. The depicted external network nodes serve as bridges between the framework and blockchain networks (e.g., the Ethereum network). They are used by the framework for interacting with a blockchain’s network, e.g., for submitting new transactions or requesting new blocks.

The Core Logic communicates with Blockchain Proxies and consists of the three major components presented in Sections III-A to III-C: The Monitoring Component, the Blockchain Selection Algorithm, and the Switchover Component. Based on data requested via the Blockchain Proxies, the Core Logic validates each blockchain’s metrics by applying MVF, calculates each blockchain’s benefit, determines the most beneficial chain, and provides the functionality for switching to another blockchain. The Core Logic is agnostic to a blockchain’s technical details. Instead, the according Blockchain Proxy translates data from a particular blockchain into a neutral format that can be processed by the Core Logic. For each supported blockchain, such a proxy is implemented. A proxy abstracts interactions with the underlying blockchain by providing an interface used by the Core Logic. Examples for interactions are writing new data records into a blockchain, reading data records from a blockchain, and requesting new blocks.

We have selected Bitcoin, Ethereum, Ethereum Classic, and Expanse for the prototypical implementation of the proposed framework⁴. The first three blockchains have been chosen due to their popularity, whereas Expanse has been selected since storing data is very cheap due to the low market price. Notably, the proposed framework is not restricted to these blockchains. The mentioned blockchains should be considered as exemplary. In case further blockchains should be supported, the framework can be extended by providing additional proxy implementations.

In order to save disk space, the framework only keeps blocks in memory that are needed for calculating each blockchain’s

⁴<https://github.com/pf92/blockchain-interop>

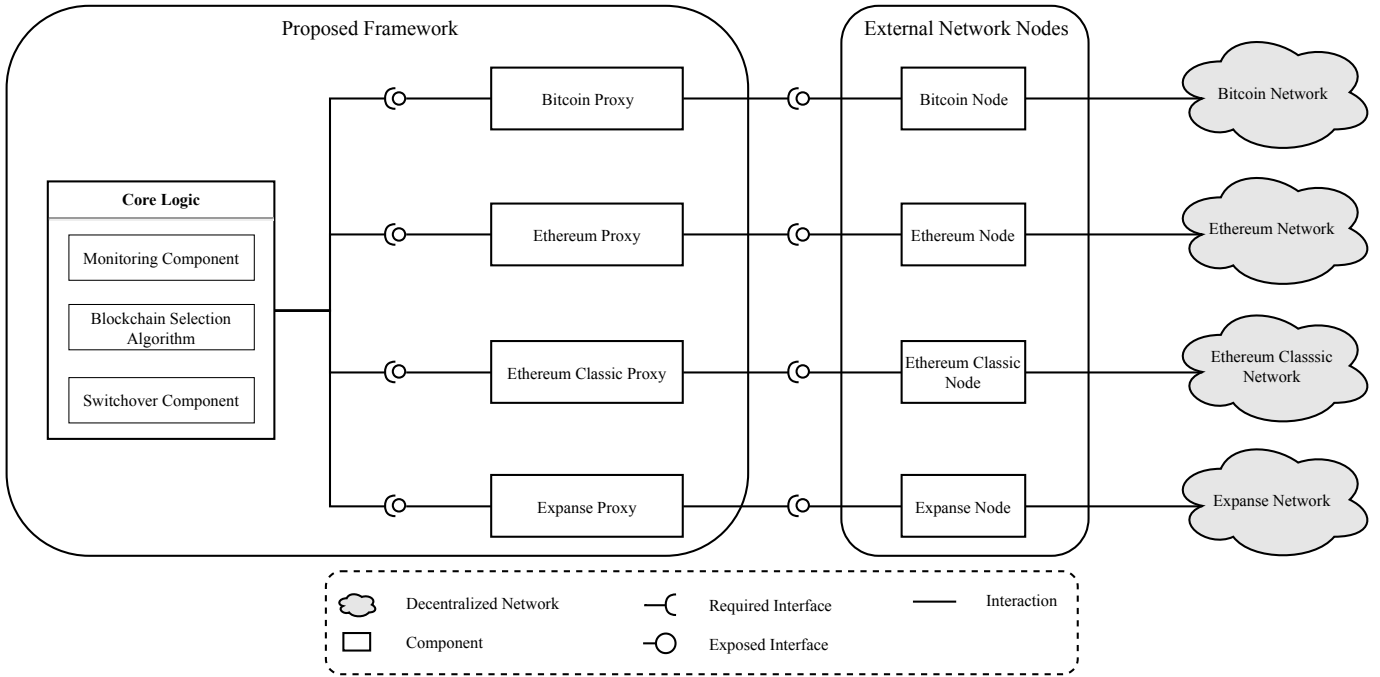


Fig. 1. Architecture of the proposed framework.

metrics, i.e., only blocks which have been mined during the last 24 hours are stored.

The framework’s design incorporates the reactive programming paradigm. In the reactive programming paradigm, data flows and the propagation of changes play a key role. If a data source changes its value, the change is propagated through the entire topology, i.e., each operator or observer that is part of the topology or is registered to receive notifications is informed about changes [2]. In the framework’s architecture, external network nodes act as data sources. Once a new block is received by the framework, subsequent computation steps are triggered in order to recalculate the metrics discussed in Section III-A. Each new block affects the calculation of the metric values, and changes of metric values affect the blockchain selection algorithm, i.e., the weighted ranking system and the MVF, as discussed in Section III-B.

In the following, we further elaborate on the calculation of each metric, taking into account the four blockchains which we have selected for the prototypical implementation of our framework, i.e., Bitcoin, Ethereum, Ethereum Classic, and Expanse.

1) *Cost of writing 1 KB of data into a blockchain (M1):*

A cheap method for storing data on the Bitcoin blockchain is to use the script operation code `OP_RETURN` [23]. `OP_RETURN` marks a transaction output as invalid and accepts a user-defined sequence of up to 80 bytes [7, 23]. In order to write data records into the Bitcoin blockchain, a transaction with one input and two outputs (one output that spends the remaining coins and another one that holds the data) is sufficient. A transaction that stores 80 bytes of data in its second output has an overall size of 282 bytes. To store

one kilobyte of data, 13 transactions are required. The overall size of these 13 transactions is $12 \cdot 282 + 266 = 3,650$. 266 bytes is the size of the transaction that stores the remaining 64 bytes. The overall size of 3,650 bytes is multiplied with the user-defined transaction fees (Satoshi per byte). If the user does not provide transaction fees, an estimation of transaction fees is requested from external APIs.

Ethereum-based blockchains like the selected Ethereum, Ethereum Classic, and Expanse, offer three possibilities for storing data on the blockchain [27]. The first possibility is to store records in the data field of a transaction. In Ethereum and Ethereum Classic (both systems use the Ethereum Virtual Machine), every transaction costs 21,000 gas. For every non-zero byte that is stored in a transaction’s data field, additional 68 gas have to be paid [25]. The number of bytes stored in a transaction is bounded by the current block gas limit. A single transaction carrying one kilobyte of data would consume at most $21,000 + 68 \cdot 1024 = 90,632$ gas. As of February 2019, the block gas limit for Ethereum and Ethereum Classic is about 8,000,000, far enough for executing transactions holding one kilobyte of data.

The second possibility is to store data in logs. The cost of storing one byte in a log is eight gas. Additional 375 gas have to be paid for the `LOG` operation [25]. However, this option requires a smart contract that emits log events. Due to different ways of writing such a contract, it is difficult to calculate the cost. Nevertheless, the second option is more expensive than the first, since every input data intended to be logged as event is also encoded in the data field of the transaction submitted for the contract call.

The third option is to store data in a smart contract’s

storage. Storing a 32-byte word in a smart contract’s storage costs 20,000 gas [25]. Additional gas has to be paid for the transaction that contains the contract call and the input data.

Therefore, the first option is the cheapest one. This statement also holds for Expanse, since the aforementioned operations consume the same gas cost on the Expanse Virtual Machine⁵. In case the user does not specify a preferred gas price, the framework requests the median gas price from the external network nodes.

2) *Cost of retrieving 1 KB of data from a blockchain (M2)*: Since data records are stored in a transaction’s data field without the involvement of smart contracts, reading data records is free of charge. Hence, in the current reference implementation, this metric is always zero for all supported blockchains.

3) *Exchange rates (M3)*: The framework continuously requests the current market price in USD for cryptocurrencies associated with the supported blockchains. In the reference implementation, we use CryptoCompare⁶, an external service exposing interfaces for requesting these market prices.

4) *Inter-block time (M4)*: The rolling average of the time between two blocks is computed by applying the formula shown in (4), where n denotes the number of blocks mined during the last 24 hours. The presented formula is applied for all supported blockchains.

$$\frac{24 \cdot 3600}{n} \quad (4)$$

5) *Transaction throughput (M5)*: As shown in (5), the transaction throughput (rolling average) is computed by summing up the number of transactions stored in each block that has been mined during the last 24 hours, and by dividing this sum by $24 \cdot 3600$. n denotes the total number of mined blocks during the last 24 hours and txcount_i represents the number of transactions of block i .

$$\frac{\sum_{i=1}^n \text{txcount}_i}{24 \cdot 3600} \quad (5)$$

6) *Degree of decentralization (M6)*: We calculate the distribution of a network’s hash power in two different ways due to fundamental differences between Bitcoin- and Ethereum-based blockchains. In Bitcoin, it is sufficient to count the number of blocks for each miner that has mined at least one block during the last 24 hours and, in a further step, to divide each miner’s block counter by the overall number of blocks. For Ethereum-based blockchains, in addition to regular blocks, also uncle blocks are taken into account, since miners also spend computational power for integrating these blocks.

7) *Network hash rate (M7)*: For Bitcoin, the average number of hashes per second the network has performed in the last 24 hours is computed as shown in (6).

$$\frac{n}{144} \cdot \frac{D \cdot 2^{32}}{600} \quad (6)$$

Here, n denotes the number of blocks that have been mined during the last 24 hours. $D \cdot 2^{32}$ specifies the expected number

of hashes that have to be calculated to find a block with difficulty D [6]. In Bitcoin, D is set such that, on average, a new block is mined every ten minutes (600 seconds) [6]. Thus, 144 is the number of blocks are expected to get mined within 24 hours. Since a new block is anticipated to get mined every ten minutes, $D \cdot 2^{32}$ hashes are expected to be computed in 600 seconds, yielding an average network hash rate of $\frac{D \cdot 2^{32}}{600}$ hashes per second [6]. The term $\frac{n}{144}$ adjusts this hash rate in case less than or more than 144 blocks have been mined. We calculate the hash rate of Ethereum-based networks by summing up the *difficulty* field of each block and each uncle block mined during the last 24 hours and by dividing this sum by the number of seconds equal to 24 hours.

IV. EVALUATION

In order to evaluate the proposed framework, we investigate its benefit in terms of cost, performance, and trust by using exemplary scenarios. For this, we analyze the framework’s reaction to varying blockchain properties as well as its handling of changing user demands.

Since the framework relies on external network nodes, we set up a Bitcore⁷ node for Bitcoin, two Parity⁸ nodes for Ethereum and Ethereum Classic, and we use gexp⁹ to run an Expanse network node. Each deployed network node uses the respective main chain. For our experiments, we deploy each network node on a separate virtual machine (1 vCPU, 4 GB RAM) hosted on the Google Cloud Platform. The framework itself is executed on a MacBook Pro (late 2013, 2.4 GHz Intel Core i5, 8 GB 1600 MHz DDR3, Intel Iris 1536 MB, 256 GB SSD, macOS 10.13.6, Oracle JDK 10).

In the following, four evaluation scenarios are presented. Scenario 1 analyzes exemplarily the framework’s reaction to varying blockchain metrics by emulating a decreasing hash rate. For this experiment, we select Expanse as the currently used blockchain. The experiment can also be performed with other blockchains such as Bitcoin, Ethereum, or Ethereum Classic without any restrictions. To customize the framework’s internal logic, we provide an implementation of the MVF that returns the 9-tuple (true, true, true, true, true, false, true, false) in case the network hash rate drops below 180 GH/s, otherwise (true, true, true, true, true, true, true, true). Thus, if the network hash rate drops below 180 GH/s, M7 and the corresponding blockchain are invalid (denoted by the boolean value *false*). Furthermore, we set each metric’s weight to 1 and provide for each metric a score assignment function that always returns a score value of 1, since for the conduction of this experiment it is not relevant which blockchain is selected by the framework after the detection of a decreasing hash rate. Starting at a hash rate of 200 GH/s, we emulate a decreasing hash rate reduced by 5 GH/s every 5 seconds. As shown in Listing 1, once the hash rate is

⁷<https://bitcore.io/>

⁸<https://www.parity.io/ethereum/>

⁹<http://expanse-org.github.io/go-expanse/>

⁵<https://expanse.tech/docs/developer/>

⁶<https://min-api.cryptocompare.com/>

Listing 1. Log extraction that shows the reaction of the framework in case the network hash rate of Expanse decreases rapidly.

```
13:52:25,189 - Switchover suggestion: Expanse
13:52:26,983 - Expanse network hash rate: 195.0 GH/s
13:52:31,984 - Expanse network hash rate: 190.0 GH/s
13:52:37,806 - Expanse network hash rate: 185.0 GH/s
13:52:42,807 - Expanse network hash rate: 180.0 GH/s
13:52:46,844 - Expanse network hash rate: 175.0 GH/s
13:52:46,845 - Hash rate (175.0 GH/s) violated
13:52:46,847 - Switchover suggestion: Ethereum Classic
```

under 180 GH/s, the framework suggests to switch to another blockchain (here: Ethereum Classic).

For Scenarios 2–4, we assume that the framework is used in a service-oriented architecture that is made up of different services adopted and operated by several independent and possibly competing business partners. In order to monitor the adherence to service-level agreements (SLAs), services publish relevant information to a blockchain. We conduct these scenarios based on metric values measured between 25.09.2018 and 17.10.2018.

Scenario 2 analyzes the benefit of the framework’s selection mechanism in terms of cost, performance, and trust. We assume the involved business partners want to use a blockchain that is cheap, fast and has a high level of trust. The framework is configured with the weighted ranking settings outlined in Table V. Since we assume a demand for very cheap and fast write operations, and a high level of trust, we set the weights for M1, M3, M4, M5, M6, M7 and M8 to five (highest importance). Metric M2 can be ignored (i.e., we set the corresponding weight to zero), since read operations are free and our reference implementation does not make use of smart contracts. In order to benefit from an accurate selection, we define the score assignments as granularly as possible, e.g., by considering very low cost in the score assignment.

According to their popularity and miner activity, we assume a reputation of 10 for Bitcoin and Ethereum, a reputation of 9 for Ethereum Classic and a reputation of 5 for Expanse. In order to emulate an execution on 25.09.2018, the framework is fed with the values measured on that day. According to the weighted ranking outlined in Table VI (metrics with a weight of zero are omitted), Ethereum is the most beneficial blockchain. The key points of this selection are as follows:

- By using Ethereum, the cost of writing one KB of data is approximately 24 times lower than the cost of writing the same amount of data into the Bitcoin blockchain. The cost of writing data into the Ethereum Classic and Expanse blockchains is about 154 times and about 96 times lower, respectively, than the cost when using Ethereum.
- Compared to Bitcoin with a price of 6,394.25 USD, the exchange rate of Ethereum is about 30 times lower. The price in USD for one Ether is approximately 20 times greater than the price for one token on Ethereum Classic. With an exchange rate of 0.36 USD, Expanse is the cheapest token.
- Ethereum features an inter-block time about 38 times

TABLE V
WEIGHTED RANKING SETTINGS USED FOR THE EVALUATION (M2 IS NOT LISTED SINCE IT IS ALWAYS ZERO IN THE IMPLEMENTATION).

Metric	Weight	Score Assignment
M1	5	$[0; 10^{-4}] \rightarrow 4$, $[10^{-4}; 10^{-2}] \rightarrow 3$, $[10^{-2}; 10^{-1}] \rightarrow 2$, $[10^{-1}; 1] \rightarrow 1$, $[1; \infty) \rightarrow 0$
M3	5	$[0; 50] \rightarrow 4$, $[50; 100] \rightarrow 3$, $[100; 250] \rightarrow 2$, $[250; 500] \rightarrow 1$, $[500; \infty) \rightarrow 0$
M4	5	$[0; 20] \rightarrow 4$, $[20; 40] \rightarrow 3$, $[40; 60] \rightarrow 2$, $[60; 120] \rightarrow 1$, $[120; \infty) \rightarrow 0$
M5	5	$[10; \infty) \rightarrow 4$, $[5; 10] \rightarrow 3$, $[2; 5] \rightarrow 2$, $[0.45; 2] \rightarrow 1$, $[0; 0.45] \rightarrow 0$
M6	5	Proportion (%) of the biggest miner: $[0; 22] \rightarrow 4$, $[22; 27] \rightarrow 3$, $[27; 30] \rightarrow 2$, $[30; 38] \rightarrow 1$, $[38; \infty) \rightarrow 0$
M7	5	Rates are denoted in terahashes: $[1,000; \infty) \rightarrow 4$, $[700; 1,000] \rightarrow 3$, $[400; 700] \rightarrow 2$, $[100; 400] \rightarrow 1$, $[0; 100] \rightarrow 0$
M8	5	$[8; 10] \rightarrow 4$, $[6; 8] \rightarrow 3$, $[4; 6] \rightarrow 2$, $[2; 4] \rightarrow 1$, $[0; 2] \rightarrow 0$

shorter than Bitcoin and about three times shorter than Expanse. Ethereum Classic has almost the same inter-block time as Ethereum.

- With a throughput of about 5.74 transactions per second (tps), Ethereum processes by far the greatest number of transactions, whereas Bitcoin has a rate of 2.57 tps, Ethereum Classic a rate of 0.47 tps, and Expanse handles only 0.06 tps.
- The network hash rate of Ethereum is about 16 times greater than the rate of Ethereum Classic and about 1,275 times greater than the hash rate of Expanse. The Bitcoin network mines with a hash rate approximately 217,012 times greater than that of Ethereum.
- The biggest Ethereum miner controls about 24% of the network’s hash power, whereas the biggest Ethereum Classic and Expanse miners control about 42% and 48%, respectively. The biggest miner of the Bitcoin network controls about 20%.

Scenario 3 investigates the framework’s handling of changing user demands indicated by adjusted metric weights. We assume that engineers of one business partner plan to run data-intensive tests on their adopted services. Since a large amount of data is scheduled to be written into the blockchain, low cost is preferred. Furthermore, we assume that the reputation can be neglected for the test execution. This scenario is conducted based on the weighted ranking settings outlined in Table V. In order to incorporate the changed demands in the weighted ranking system, we set the weights for M6, M7, and M8 to 0. Furthermore, we assume that these changes take effect on 07.10.2018. Table VII shows the weighted ranking based

TABLE VI

WEIGHTED RANKING OF SCENARIO 2 (BASED ON VALUES MEASURED ON 25.09.2018).

Metric	Bitcoin W. Score	Ethereum W. Score	Ethereum Classic W. Score	Expanse W. Score
M1	0	5	15	15
M3	0	10	20	20
M4	0	20	20	10
M5	10	15	5	0
M6	20	15	0	0
M7	20	5	0	0
M8	20	20	20	10
Total	70	90	80	55

TABLE VII

WEIGHTED RANKING OF SCENARIO 3 (BASED ON VALUES MEASURED ON 07.10.2018).

Metric	Bitcoin W. Score	Ethereum W. Score	Ethereum Classic W. Score	Expanse W. Score
M1	0	10	15	20
M3	0	10	20	20
M4	0	20	20	10
M5	10	15	5	0
Total	10	55	60	50

on the changed settings and the metric values gathered on 07.10.2018 (metrics with a weight of zero are omitted). Since Ethereum Classic has the highest benefit, it is selected as the most beneficial chain. Due to the lack of significant variations of blockchain metrics between 25.09.2018 and 07.10.2018, Ethereum has been the most beneficial chain for the entire date range. By switching from Ethereum to Ethereum Classic, the cost of writing one KB of data has been decreased by a factor of 42. Furthermore, Ethereum Classic has almost the same inter-block time as Ethereum (approximately 14 seconds). Since Expanse has an inter-block time of 44 seconds, Ethereum Classic has been preferred, as shown in Table VII.

The intention of Scenario 4 is to show the effects of changing user requirements indicated by adjusted score assignments. For this, we assume that an inter-block time between 30 and 60 seconds is completely sufficient for conducting further service tests. Moreover, the transaction throughput becomes less important for further test executions. Due to the large amount of test data that is written to the blockchain, low cost have still high priority. This scenario is conducted based on the weighted ranking settings of Scenario 3. To reflect the changed user requirements in the weighted ranking settings, the score assignment for M4 is changed to: $[0; 60) \rightarrow 4$, $[60; 120) \rightarrow 3$, $[120; 180) \rightarrow 2$, $[180; 240) \rightarrow 1$, $[240; \infty) \rightarrow 0$. Since the transaction throughput has a lower priority, the weight for M5 is set to 3. We further assume that the weighted ranking settings are changed on 17.10.2018. Based on the metric values measured on 17.10.2018 and the changed settings, the framework selects Expanse as the most beneficial chain, since it has the highest score (as shown in Table VIII). Due to the lack of significant variations of blockchain metrics between

TABLE VIII

WEIGHTED RANKING OF SCENARIO 4 (BASED ON VALUES MEASURED ON 17.10.2018).

Metric	Bitcoin W. Score	Ethereum W. Score	Ethereum Classic W. Score	Expanse W. Score
M1	0	10	15	20
M3	0	10	20	20
M4	0	20	20	20
M5	6	9	3	0
Total	6	49	58	60

07.10.2018 and 17.10.2018, Ethereum Classic has been the most beneficial chain for the entire date range. A switchover from Ethereum Classic to Expanse enables a cost reduction by a factor of approximately 45.

Summarizing our evaluation, we see that the framework can select the most appropriate blockchain based on user preferences. Furthermore, it is able to react to volatile blockchain properties and it can handle changing user demands. These features allow users to benefit from low cost, better performance, and a higher level of trust.

V. RELATED WORK

Despite the fact that blockchain technologies have gained much research momentum in recent years, to the best of our knowledge, there are not too many approaches aiming at providing the means to switch between different blockchains.

One of the earliest contributions in the field of blockchain interoperability is the atomic cross-chain protocol (ACCS) proposed by TierNolan in 2013 [5]. This protocol enables users of different cryptocurrencies to swap their assets in an atomic fashion. Further contributions focusing on the transfer of assets are The Atomic Swap Technology (TAST)¹⁰, Tesseract [4], BarterDEX¹¹, Metronome¹², and the Republic Protocol¹³. Furthermore, sidechains aim to provide interoperability between two blockchains by locking assets on the source chain and creating them on the target blockchain. Transferred assets can only be used on one blockchain at the same time. Cryptographic proofs ensure that assets have been locked on the source chain, before new ones can be created on the target chain [10].

Beyond the trading of assets, Polkadot¹⁴, Cosmos¹⁵, and Block Collider¹⁶ aim to integrate blockchains in a more general way, e.g., by enabling communication between smart contracts located on different blockchains.

A further remarkable contribution in the field of blockchain interoperability is BTCRelay¹⁷, a smart contract running on Ethereum that verifies Bitcoin transactions. The contract acts

¹⁰<https://pantos.io>

¹¹<https://docs.komodoplatform.com/home-whitepaper.html>

¹²<https://www.metronome.io/>

¹³<https://renproject.io/>

¹⁴<https://polkadot.network/>

¹⁵<https://cosmos.network/>

¹⁶<https://www.blockcollider.org/>

¹⁷<http://btcrelay.org/>

as bridge between the Bitcoin blockchain and Ethereum smart contracts, enabling users to pay with Bitcoin for using Ethereum DAPPs.

To the best of our knowledge, there are no contributions in the field of selection of blockchains during runtime. The discussed approaches do not integrate a mechanism for selecting the most beneficial blockchain based on user-defined requirements. Furthermore, the presented approaches do not provide a functionality for switching back and forth between several blockchains and for migrating already existing data.

VI. CONCLUSION

In this paper, we have presented a framework capable of switching back and forth between blockchains at runtime. The proposed framework monitors several blockchains, calculates each blockchain's benefits according to user-defined settings, and determines the most beneficial one. Furthermore, the framework is able to react to variations of blockchain metrics and it can handle changing user demands. We have presented the framework's design in detail, using a reference implementation in Java.

Our evaluation shows that switching to another blockchain can save cost and enable users to benefit from better performance and a higher level of trust. The modular design of the framework allows future researchers to add support for further blockchains by providing additional proxy implementations.

As described in Section III, the framework is able to copy data from the currently used blockchain to the destination chain during a switchover. However, the proposed framework does not enable the migration of smart contracts between blockchains, allowing the automatic deployment of required smart contracts on the destination chain. Such a feature may be relevant if data records are managed by smart contracts rather than stored in a transaction's data field and will therefore be investigated in our future work. Furthermore, the reference implementation stores data records published to Ethereum, Ethereum Classic, and Expanse in a transaction's data field rather than in a smart contract's storage, leading to slower access times. A possible solution we want to investigate is to improve the time needed for searching data by a tracking of transactions that hold data records.

REFERENCES

- [1] M. Atzori. "Blockchain technology and decentralized governance: Is the state still necessary?" In: *Journal of Governance and Regulation* 6.1 (2017).
- [2] E. Bainomugisha, A. L. Carreton, T. v. Cutsem, S. Mostinckx, and W. d. Meuter. "A Survey on Reactive Programming". In: *ACM Comput. Surv.* 45.4 (2013), 52:1–52:34.
- [3] B. Bambrough. *Senior Banker Warns Bitcoin Has A 'Basic' Problem*. 2019. URL: <https://www.forbes.com/sites/billybambrough/2019/01/22/senior-banker-warns-bitcoin-has-a-basic-problem>. Accessed: 13.02.2019.
- [4] I. Bentov, Y. Ji, F. Zhang, Y. Li, X. Zhao, L. Breidenbach, P. Daian, and A. Juels. "Tesseract: Real-Time Cryptocurrency Exchange using Trusted Hardware". In: *IACR Cryptology ePrint Archive* 2017/1153 (2017).
- [5] *Bitcoin Forum: Alt chains and atomic transfers*. 2013. URL: <https://bitcointalk.org/index.php?topic=193281>. Forum postings between 02.05.2013 and 13.05.2017. Accessed 23.02.2019.
- [6] *Bitcoin Wiki: Difficulty*. 2017. URL: <https://en.bitcoin.it/wiki/Difficulty>. Accessed: 19.02.2019.
- [7] *Bitcoin Wiki: OP_RETURN*. 2018. URL: https://en.bitcoin.it/wiki/OP_RETURN. Accessed: 19.02.2019.
- [8] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. "SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies". In: *2015 IEEE Symposium on Security and Privacy*. 2015, pp. 104–121.
- [9] C. Bovaird. *Bitcoin Prices Have Fallen 75% From All-Time High*. 2018. URL: <https://www.forbes.com/sites/cbovaird/2018/11/19/bitcoin-prices-have-fallen-75-from-all-time-high>. Accessed: 13.02.2019.
- [10] V. Buterin. *Chain interoperability*. R3 Research Paper. 2016.
- [11] K. Christidis and M. Devetsikiotis. "Blockchains and Smart Contracts for the Internet of Things". In: *IEEE Access* 4 (2016), pp. 2292–2303.
- [12] T. Copeland. *The flaw in Bitcoin Cash*. 2018. URL: <https://decryptmedia.com/4055/bitcoin-cash-faces-centralization>. Accessed: 09.02.2019.
- [13] S. Dziembowski. "Introduction to Cryptocurrencies". In: *22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 1700–1701.
- [14] J. Herrera-Joancomartí and C. Pérez-Solà. "Privacy in Bitcoin Transactions: New Challenges from Blockchain Scalability Solutions". In: *Modeling Decisions for Artificial Intelligence*. Springer International Publishing, 2016, pp. 26–44.
- [15] I.-C. Lin and T.-C. Liao. "A Survey of Blockchain Security Issues and Challenges". In: *International Journal of Network Security* 19.5 (2017), pp. 653–659.
- [16] L. M. Maruping, V. Venkatesh, and R. Agarwal. "A Control Theory Perspective on Agile Methodology Use and Changing User Requirements". In: *Information Systems Research* 20.3 (2009), pp. 377–399.
- [17] J. Mendling, I. Weber, W. van der Aalst, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar, A. Gal, L. Garcia-Banuelos, G. Governatori, R. Hull, M. L. Rosa, H. Leopold, F. Leymann, J. Recker, M. Reichert, H. A. Reijers, S. Rinderle-Ma, A. Rogge-Solti, M. Rosemann, S. Schulte, M. P. Singh, T. Slaats, M. Staples, B. Weber, M. Weidlich, M. Weske, X. Xu, and L. Zhu. "Blockchains for Business Process Management – Challenges and Opportunities". In: *ACM Transactions on Management Information Systems* 9 (1 2018).
- [18] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun. "A review on consensus algorithm of blockchain". In: *IEEE International Conference on Systems, Man, and Cybernetics*. 2017, pp. 2567–2572.
- [19] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. URL: <https://bitcoin.org/bitcoin.pdf>. Accessed: 06.02.2019.
- [20] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.
- [21] S. O'Neal. *ABC vs SV: Assessing the Consequences of the Bitcoin Cash War*. 2018. URL: <https://cointelegraph.com/news/abc-vs-cv-assessing-the-consequences-of-the-bitcoin-cash-war>. Accessed: 09.02.2019.
- [22] C. Prybila, S. Schulte, C. Hochreiner, and I. Weber. "Runtime Verification for Business Processes Utilizing the Bitcoin Blockchain". In: *Future Generation Computer Systems* (2019).
- [23] A. Sward, I. Vecna, and F. Stonedahl. "Data Insertion in Bitcoin's Blockchain". In: *Ledger* 3 (2018), pp. 1–23.
- [24] F. Tschorsch and B. Scheuermann. "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies". In: *IEEE Communications Surveys Tutorials* 18.3 (2016), pp. 2084–2123.
- [25] G. Wood. *Ethereum: A secure decentralised generalised transaction ledger*. 2014. URL: <https://ethereum.github.io/yellowpaper/paper.pdf>. Accessed: 19.02.2019.
- [26] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen. "The Blockchain as a Software Connector". In: *13th Working IEEE/IFIP Conference on Software Architecture*. 2016, pp. 182–191.
- [27] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba. "A taxonomy of blockchain-based systems for architecture design". In: *IEEE International Conference on Software Architecture*. 2017, pp. 243–252.
- [28] A. Zohar. "Bitcoin: Under the Hood". In: *Communications of the ACM* 58.9 (2015), pp. 104–113.