Blockchain Platforms Overview for Industrial IoT Purposes

Nikolay Teslya, Igor Ryabchikov
SPIIRAS
St.Petersburg, Russia
teslya@iias.spb.su, i.a.ryabchikov@gmail.com

***Abstract*—There are a lot of blockchain platform implementations available today. To be integrated into the smart space for Industrial IoT the blockchain platform should support not only token exchange but also smart contract distribution and launching, fault tolerance consensus mechanism and equivalence between participants to create and implement new blocks and contracts. The paper provides analysis of the most used consensus mechanisms, specific features of public (permissionless) and private (permissioned) blockchains. Also a description of blockchain platforms that satisfy the requirements for the IIoT platform development is provided. By the result of the analysis the platform and specific modules have been selected for implementation of blockchain for industrial IoT platform.**

I. Introduction

The organization of interaction between the smart factory components both internally and with other factories is one of the main issues for Industry 4.0 concept. Until now, there are a lot of solutions based on the using of the IoT concept, which allows uniting many components into a single information space and providing information exchange between them. Regarding to the industry such union can be based on the concept of Industrial Internet of Things (IIoT), which is the use of IoT for the interaction of physical, virtual and social industrial components in a single information space also known as smart space. However, production becomes more and more decentralized and several problems appear, among which the following can be highlighted: the need to provide interoperability between components in the smart space and between smart spaces; trust between the participants of the information space; control over the distribution of resources (such as maintenance time, energy, etc.) and finished products [1].

To provide the interoperability between the smart space components, the ontology and ontology matching mechanism can be used. Such approach is already widely described and used in the number of projects, e.g. [2], [3]. The solution of the trust problem between components is more complicated due to heterogeneity of the participants. It can be solved with the help of digital signature [4] and access control [5] mechanisms that needs central partner that can provide trust and access control for all the components of IIoT. Control over resources distribution and finished products can be carried out using a database accessible to all components. These solutions are quite complex and require the deployment of complex infrastructure to provide fault tolerance, performance and availability. At the same time, there is an active development of blockchain technology, which provides a simpler solution for the problems presented above.

The blockchain is a distributed transaction ledger supported by a set of nodes that does not have a single point of trust and failure. Each node performs the validation of the added transactions, and the consensus mechanism allows the arrangement of nodes in the transaction order. In other words, blockchain technology allows the creation of a common information space of many independent participants with generally accepted rules for its modification (including modification rights) in which the participants do not trust each other in complete manner. Abstracting, blockchain can be perceived as a trusted platform for the execution of programs, providing certain guarantees of reliability and consistency. These guarantees depend on the specific implementation.

The operation principles of the blockchain technology can be summarized as follows. Participants in the blockchain network share the common information space (state) and the modification rules (program code). In a number of implementations of the technology, there are also persist built-in mechanisms for the distribution of new software code, called smart contracts. Each participant (in accordance with the accepted rights) can make a call to the shared program code to modify the state. The call is made through the creation of transactions. The participant notifies the others about the transaction that he would like to make. Transactions of all participants for a certain period of time are collected in blocks for the purpose of bundling and reducing the overhead costs associated with reaching a consensus. After that, in accordance with the implemented mechanism of consensus, the parties agree on the next received transaction block. Each new block contains a hash of the previous block, thereby forming a linked list. By sequentially executing all transactions starting from the first block, the current state can be restored. The consensus mechanism for the most of blockchain technology implementations, makes it possible to realize the so-called coherence in the long run, ensuring that in case of the absence of changes after some time from the last update all participants will share the common state. In accordance with the implementation of the mechanism for achieving consensus, the difference between the states of a pair of nodes may be delay (the node has not yet had time to receive and apply a new transaction block) or a temporary branch (two different valid blocks of transactions were received by different nodes).

The aim of the paper is to analyze available blockchain solutions in order to be implemented for Industry 4.0 case. The

main questions to be analyzed are implemented consensus algorithm, publicity of network, smart contract support, platform to be used for providing all these functions in IIoT platform. These parameters are partly depending on each other, for example, the consensus mechanism is depending on the publicity of the network. To be appropriable for IIoT platform [1] blockchain should support smart contracts, support block generation without mining procedure and consensus mechanism that can work with a low number of users.

The rest of the paper is structured as follows. Section 2 provides information about papers that studies blockchain implementations. Section 3 provides descriptions of consensus mechanisms. Section 4 describes permissonless and permissioned implementations of blockchain network and their advantages and disadvantages. Section 5 provides some blockchain platforms and modules that can be used for integration of blockchain and IIoT.

II. State of the art

At the moment there are many implementations of the blockchain technology. Publication search showed the existence of at least 20 platforms. Their main differences are the implementation of the consensus mechanism, the transactions validation mechanism (and, consequently, the guarantees of reliability and consistency) and functionality (for example, support only currency exchange or blockchain for common purposes with the ability to create smart contracts). In the case of a common-purpose blockchain (with the support of smart contracts), there are differences in available state storage structures, as well as opportunities to limit the performance of smart contracts.

As blockchain technology has gained popularity not so long ago, many of the proposed projects are at an early stage, so they do not have qualitative documentation describing the algorithms that support the claimed guarantees and therefore are not suitable for the implementation of production projects. The review of existing implementations of blockchain technology (in particular, the consensus mechanisms implemented in them) is presented in [6], which provide enough material to conclude their consistency – the satisfying for the declared guarantees, as well as features that were not clearly documented. In the scope of this paper, basically, the projects described in [6] were studied. The peculiarities of their work (outside the consensus mechanism) were studied and correlated with the requirements of this project (organization of a common smart space of independent participants).

Next sections will describe in detail consensus mechanisms used in implementations and publicity of blockchain network.

III. Consensus mechanisms

The main difference between various implementations of blockchain technology is the mechanism of consensus. First of all, it is characterized by the possible types of inconsistency and the requirements / assumptions necessary for correct work. At the moment there are different types of mechanism for reaching consensus. The most popular mechanisms are presented and described below:

- Proof-of-work (PoW) [7];
- Proot-of-elapsed-time (PoET)) [8];
- Proof-of-stake (PoS) [9];
- Byzantine Fault Tolerance (BFT) [10];
- Federated Byzantine Agreement (FBA) [11];
- Various combinations of the above algorithms.

A. Proof-of-work

The Proof-of-work mechanism provides the ability to create transaction blocks by any participant also known as miners randomly at approximately equal intervals. The participant who will create the next block is not determined in advance. After creating a valid block, the participant distributes it to the network, the rest participants should accept it, validate and proceed to create the next block. But a situation is possible in which several valid blocks will be created (by different participants). This situation is viewed as a conflict that will be resolved by selecting a longest chain that is defined by consensus of more than 51% of participants. Since only one chain can be active, unaccepted block is added to the block tree and marked as orphaned. This makes it possible to implement an attack, in which an attacker tries to convince a participant that a certain transaction has been made (for example, a payment for services), perform the desired actions (get a service), and then cancel the transaction by creating a new valid block, thereby receiving the service and having returned payment. With Proof-of-work, the probability of generating a valid transaction block is proportional to the computational power that the participant has (for the generation it is necessary to solve the complex mathematical problem). So the miner or group of miners should control more than 50% of computational power of whole network to successfully implement this type of attack. That's why it also called "51% attack". For large blockchain networks with a large number of participants and a uniform distribution of computing power, the probability of this attack is small due to the extremely high computational and electric power needed by one attacker to overcome total power of other participants. Unfortunately for small blockchain networks, which are using this kind of consensus mechanism this attack can be real. The examples of blockchain networks that currently implements this type of consensus mechanisms are Bitcoin [7], Ethereum [12].

B. Proof-of-elapsed-time

The Proof-of-elapsed-time mechanism is similar to Proof-of-work, but for the random block creation implementation, the solution is not based on a resource-intensive task, but on special hardware (in particular, in the Hyperledger Sawtooth it is used specific Intel CPU instruction set called Intel Software Guard Extensions (SGX) that allows applications to run *trusted code* in a protected environment [13]). It allows to wait for a random period of time and, due to a cryptographic signature, to prove that the user has made an expectation. This solution allows to reduce the cost of block creation (there is no need to spend resources on calculating a mathematical problem), but it has an obvious drawback — the network's performance depends on hardware or virtual environment performance. The network will lose its efficiency in case the hardware is hacked (it's enough to remind two last vulnerabilities for Intel processor — Spectre and Meltdown, that causes decrease of system performance to 2-11%

due to the patches [14]) and the mechanism of operation becomes known. Also, not to mention that hardware developers and manufacturers will have possibilities to influence the network by backdoors in hardware architecture. In addition, there is still a problem of influence through investment of funds due to the probability of block generation that depends on the amount of Intel processors a participant has. This protocol is implemented in the Hyperledger Sawtooth blockchain network [13].

C. Proof-of-stake

In general, proof-of-stake approach can be described in the following way: the more part the user has in the blockchain network (the part can be manifested in the form of currency, assigned weights, some contribution to the work / development of the network), the more influence he can to render. Net implementation of the approach in public networks implies random selection of the next participant in the network who will be responsible for creating the block. The likelihood of choice is proportional to its stake. This approach has more variables than Proof-of-work (for example, how to punish the creators of the blocks for failing to fulfill their duties, or how to cope with the problem of possible branches that increase the probability of double waste), so there is no generally accepted implementation that has proven its stability. There are implementations combining Proof-of-stake and other approaches, for example, proof-of-work or Byzantine Fault Tolerance. An example of implementation of that mechanism in the blockchain technology is Peercoin [15].

D. Byzantine Fault Tolerance (BFT)

Byzantine Fault Tolerance is the property of information system to be tolerant to fault one or more of it component. It originates from the Byzantine General problem that assumes need to reach consensus between three generals is conditions that one of them could send conflicting information to others [16]. A consensus mechanism based on the BFT is used in the permissioned blockchain networks (in networks whose members are known to each other). In blockchain network this mechanism allows the achievement of consensus on the condition when less than a third of all nodes are faulty including malicious nodes that are interfering with the overall goal. Violation of the conditions can lead to a lack of progress, or branching (for example, if more than two-thirds of all nodes are nodes that are maliciously working together). The peculiarity of this protocol in comparison with those described before is that it does not allow the rollback of the state — the accepted valid block is final and can not be replaced. But this also involves a drawback — progress is possible only with the performance of a certain part of the nodes, while the above mechanisms are able to function even when only one node is running. There are implementations combining this mechanism with a proof-of-stake in the sense that the quantity condition (more than 2/3) is applied to the sum of the participants' weights, and not to their number. This mechanism is the most popular among permissioned nodes. The mechanism is used in a variety of private blockchain networks implementations — Hyperledger Fabric [17], Tendermint [18], Corda [19], Exonum [20] and others.

E. Federated Byzantine Agreement

The Federated Byzantine Agreement is similar to the Byzantine Fault Tolerance mechanism but allows each participant to indicate their own list of trusted participants when reaching a consensus, rather than sharing common assumptions. This mechanism is implemented in the projects Ripple [21] and Stellar [22].

IV. Permissioned and permissionless blockchain platforms

In accordance with the type of mechanism for reaching consensus, so-called private (permissioned) and public (permissionless) blockchain network can be built. A publicly accessible blockchain network allows connection of any participant, providing equal opportunities. Guarantee of the reliability of the network is provided mainly based on the assumption of the presence of a large number of participants and invested funds, so that the disruption of the system becomes too expensive. The advantage is the ability to create global blockchain network, but at the moment global networks suffer from a scaling problem - with a large number of participants, the allowable specific intensity of transactions is extremely low (since the distribution in the blockchain means only duplication, but not scaling). Consensus mechanisms that allow the development of global public networks are Proof-of-work, Proof-of-elapsed-time and Proof-of-stake.

Private blockchain networks are supported by the participants who know each other and who are interested in the efficiency of the common network. In such networks they speak about a certain "trust" of participants to each other, due to which it is possible to apply certain optimizations. Violation of trust can lead to the network being disabled until the malicious participants are corrected or removed from the network. In some implementations, a breach of trust can also lead to incorrect system operation, but it should be noted that there are implementations in which the malfunction can be limited to inoperability, but not by incorrect work. An incorrect job is a violation of the specified consistency rules (for example, double spending or violation of other rules of smart contracts). In private blockchain networks there is the possibility of flexible access rights setting: who can invoke transactions, who can participate in the mechanism of consensus and how much weight each participant has in this process. Proof-of-stake, Byzantine Fault Tolerance and Federated Byzantine Agreement protocols can be used to implement this type of networks. The comparison of blockchain network types is presented in Table I.

Another difference between blockchain platforms, in addition to the mechanism for consensus achieving, is functionality. Many implementations are aimed only at accounting and transfer of assets (currencies or objects) between the participants, including MultiChain, Chain Core and others. For the development of a common smart space, it is necessary to implement a blockchain platform of common purpose that allows the creation of smart contracts.

The architecture for integration of blockchain and IIoT is proposed in [1]. It allows to unite small and weak devices like sensors and powerful computing units through shared

information space (smart space) as well as provide trust between them through recording the signed pieces of shared information into blockchain. At the current state of research sensors are available only to provide information without signing it by own signature. Also, these parts of IIoT cannot took part in consensus making due to the week calculation power. Proceeding from the above, integration of IIoT and blockchain technologies should be based on the implementation of permissioned blockchain technology that providing high performance proceeding more than 2500 transaction per second and allow to create private smart spaces for the joint work of small groups of participants.

TABLE I. BLOCKCHAIN TYPES COMPARISON

	Permissionless blockchain	**Permissioned blockchain**
Participant identification	No	Yes
Participant actions limitation	No limits	By consensus between the rest participants
Transaction validation	All participants	Only specified participants
Control	No	Yes
Smart contracts	Yes, depends on platform	Yes, depends on platform
Consensus mechanisms	Proof-of-work, Proof-of-elapsed-time and Proof-of-stake	Proof-of-stake, Byzantine Fault Tolerance and Federated Byzantine Agreement
Platforms	Bitcoin, Ethereum, Hyperledger Fabric	Corda, Hyperledger Fabric, Hyperledger Burrow, Tendermint, Symbiont, Kadena, Quorum, HydraChain, Swirld, Exonum

V. BLOCKCHAIN PLATFORMS AND MODULES FOR INTEGRATION WITH INDUSTRIAL IOT

A. Platforms

According to the review of private blockchain network platforms implementations proposed in [6], the choice can be limited to implementations of *Corda, Hyperledger Fabric, Tendermint* and *Symbiont*, which support the mechanism of smart contracts. Other implementations (Kadena, Quorum, HydraChain, Swirld) even support the mechanism of smart contracts, but either have significant drawbacks, or their correctness is difficult to verify (in particular, for lack of detailed documentation and proprietary code). In particular, *Kadena* uses a closed implementation of the mechanism for reaching consensus, and in the documentation describing reliability guarantees, there are erroneous statements; *Quorum* in particular QuorumChain by default configuration has only one block-making node, so if it crashes the protocol halts; the implementation of *HydraChain* and *Swirld*, in the opinion of the authors [6], need more detailed review and validation.

Corda is basically not a real blockchain platform but provides same functionality that can be used mostly in financial operations. It offers a way of representing the shared information space and performing transactions that differs from most other implementations, through which a certain level of scaling can be achieved. In Corda, transactions are not formed in one general order. Instead of blocks, the concept of states (facts) is introduced to represent arbitrary data structures. Each participant can store a set of states of interest, and their sets may overlap. Transactions, instead of modifying a common database (for example, a key-value database), accept a set of states for input, and output a set of other states, marking the input states as used. Only unused states can be transferred to the transaction log. This makes it possible not to separate the shared database and the transaction list, but only store the data states (data structures) of the transactions to which the participants are associated. At the same time, in the process of achieving a consensus, everyone who may wish to use a particular state graph (become a participant in a transaction that accepts a child's entry) or those who are entrusted to it should participate. As a mechanism for achieving consensus, **BFT-SMaRt** [23] is used, which is used by many other blockchain implementations. There is a mechanism for replicating Corda's state graphs for new stakeholders.

This implementation may be desirable in case of full privacy is required (information about the transaction should be available only to its participants). The state chains use a small number of participants (for example, in the case of a common currency exchange, this approach may not be suitable, since the currency can move between all participants, and to keep it in mind, everyone needs to store the entire graph) and there is trust between certain participants (on which the absence of double spending can be checked). But for the paper goal (the analysis of blockchain platforms for integration of IIoT and blockchain), it is assumed that information is available to all participants, thus Corda's efforts to partition information are unnecessary. In addition, in the general case, the solution of which is directed to this work, full confidence in the participants (to verify the absence of double spending) is not allowed.

One of the most notable implementations of private blockchain network platforms at the moment is **Hyperledger Fabric**. This implementation was originally proposed by IBM, after which it was supported by the Linux Foundation community. At the moment there are large projects using Hyperledger Fabric, which may indicate its stability. For example, a joint project of Maersk (transportation and logistics) and IBM on the use of Hyperledger to track the movement of goods by stakeholders [24]. The use of blockchain technology allows to automate the workflow and increase the speed of information exchange. The main difference between Hyperledger Fabric and other implementations is the separation of the execution and validation of transactions into two phases, which can help to scale the solution. The transaction is processed as follows:

1) The initiator forms an application for the execution of the transaction, signs it with his private key and sends it to the group of responsible node-validators.

2) The validator nodes execute the requested smart contract and generate a response containing sets of values read and written to the database (Hyperledger Fabric uses a key-value database), signed with their private keys and sent to the initiator.

3) The initiator verifies that the returned replies are identical (they can differ if some nodes do not have time to apply the last blocks of transactions) and sends the request, sets of read and written values and all the signatures to the ordering nodes that form the new block and apply the consensus protocol. At the same time, it is checked that since the smart contract was executed by the validating nodes, the read values of the database

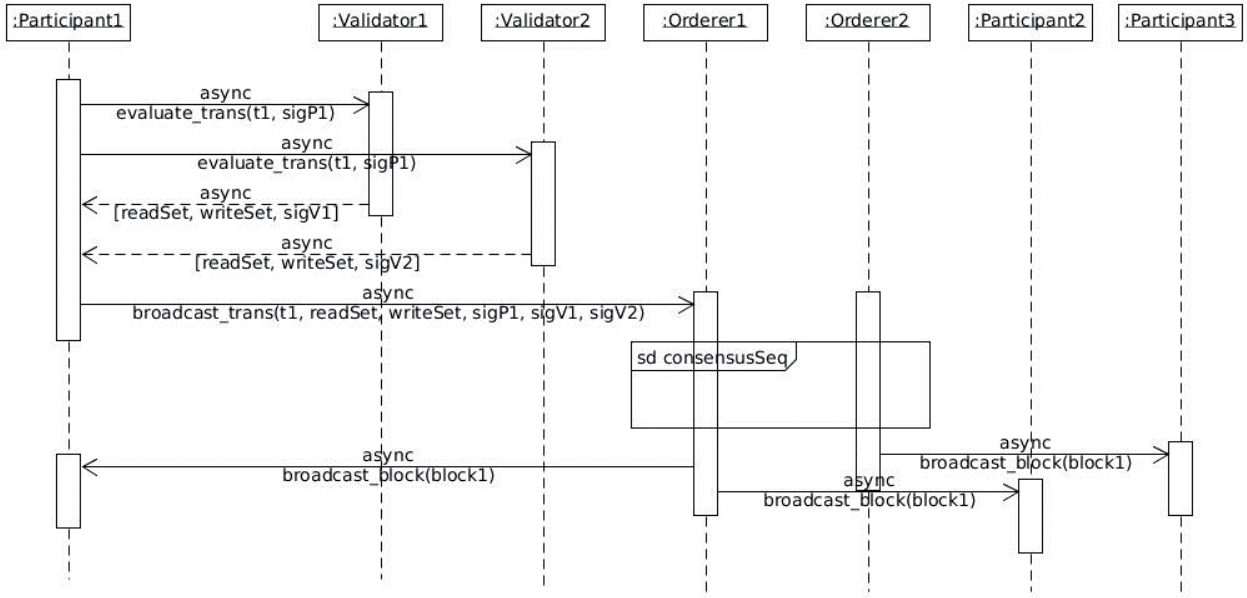

Fig. 1. The transaction process for Hyperledger Fabric, represented by a sequence diagram in UML notation

did not change (it is assumed that the result of the smart contract will not be changed).

4) The generated block is shared between all nodes of the network, which apply the described changes of the key-value database.

Figure 1 shows a diagram of the process for performing the transaction. In Hyperledger Fabric, participant nodes, validators, and sequencing nodes are logically divided, but each node can perform all three roles.

Scaling in this case can be achieved due to the fact that not all nodes should perform smart contracts, but only some validator nodes, while the others should only check the digital signatures and apply changes to the key-value database. The conditions for sufficiency of the set of nodes-validators can be set quite flexible: by enumeration, by percentage of the total number, by percentage with regard to weights, etc. For example, it can be specified that a sufficient number of nodes performing a smart contract is a third.

But this solution still has a potential weakness. If an attacker could gain access to a sufficient number of validator nodes (in accordance with the accepted sufficiency rules), he can neglect the generally accepted rules of consistency (represented by smart contracts) and implement any change in the database. As an example, let's assume that in the blockchain unit records are kept of the ownership of assets, and the possibility of transferring ownership rights to the current owner is realized. The owner of a certain asset can be written to the corresponding cell in the database. Having access to the sites-validators, an attacker can change the current owner of the asset, even without hacking the owner's node of the asset. Thus, it can be argued that the work of unhidden participants will be broken, or rather, it will continue, but it is incorrect. As a result, they can be seriously harmed. For example, a producer member can expect production orders from another participant. An attacker can create applications for production on behalf of this participant and force the producer to waste resources. For the private blockchain network this case is hard to be reproduced due to the direct control and validation by trusted nodes, but in public network this case can be real.

Thus, the separation of the validation process into two stages makes the credibility of other participants necessary. And adjusting the sufficiency of the number of validating transactions can not solve the problem — even if there are 98 sufficient out of 100 nodes, in the case of hacking 98 nodes, the remaining two non-hacked participants will also suffer. If the need for validation is indicated by all participants, then the reliability requirement will be violated — failure of one participant will lead to disruption of the whole blockchain network.

Perhaps the success of Hyperledger Fabric, in spite of this problem, is due to the fact that in its tasks there are especially trusted participants, the probability of hacking or falling of which in practice is small or would mean serious consequences for all. But in the general case, when the participants are considered equal, and the likelihood of hacking (or collusion) is real, this vulnerability can be very serious.

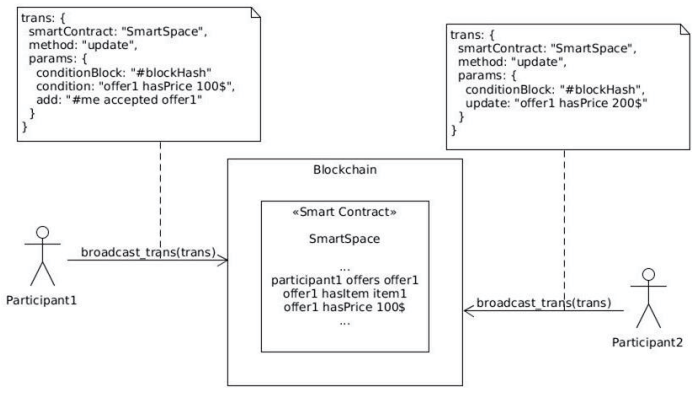

Fig. 2. Publication of a transaction with a conditional entry into the smart space

The described vulnerability is absent in standard implementations of private blockchain networks in which the implementation of smart contracts and validation is performed by each node after reaching a consensus on the order of transactions. In this case, if the participants are hacked / dropped, only the progress of the network is possible, but not incorrect work (in the sense of breaking smart contracts). Although, to guarantee this, some addition of the standard algorithm is necessary. In the case of Byzantine Fault Tolerance, progress is possible with a consistency of more than 2/3 of nodes. In the case of more than 2/3 of the nodes are hacked, an attacker can implement a branching of the blockchain — two uncommitted participants can consider two different chain of blocks to be valid. In this case, an attacker can manipulate the set and order of transactions in blocks. This can lead to a mismatch between the participants. For example, a general ontology of products, to which the producer refers in the description of his proposals, can be presented in the blockchain. In the event that the branching occurred, and the ontology of the product description differs from the producer and the consumer, then when the order is made, the consumer can receive not what he expected. The solution to this problem can be the inclusion in the transaction hashes of blocks that will be checked during the execution of transactions. The semantics of the application will then be the following: "Execute a smart contract X subject to the condition of the blockchain taken before block Y". If the conditions under which the participant would like to execute a smart contract are temporary (for example, the cost of a service may change in the future due to some events), the check of the desired state can be included in the transaction (Figure 2). For example, a condition ontology can be described as conditions that will be checked before the execution of a smart contract.

B. Modules

Besides using existing blockchain platforms it is possible to create own blockchain platform with ready-to-use consensus mechanisms. There are set of libraries that provides such functions. Two of them are described here: BFT-SMaRt and Tendermint.

BFT-SMaRt [23] allows to achieve a consensus on the order of messages within a cluster of nodes with the support of Byzantine fault tolerance (maintaining efficiency if maliciousness is less than a third of nodes). It should be noted that the library does not provide recovery of the message chain after a long period of inactivity, which can occur with a small number of nodes. Instead, BFT-SMaRt offers the implementation of aggregate state calculation based on the entire message sequence, but having a small size (in practice, not more than a few megabytes). Upon restoring after a long period of inactivity, the BFT-SMaRt node requests the current state and after receiving the same response from a sufficient number of participants (at least a third of them), accepts the state and continues working in the normal mode, participating in the process of reaching consensus.

Tendermint provides functions to create secure and consistent replications of objects on many machines. It consists of two chief technical components: a blockchain consensus engine and a generic application interface [18]. The consensus engine that is called Tendermint Core, controls the order of transaction recording on every node — same transactions should be recorded in the same order for all nodes. The application interface, called the Application BlockChain Interface (ABCI), enables the transactions to be processed in any programming language. Unlike other blockchain and consensus solutions, which come pre-packaged with built in state machines (like a key-value store, or a quirky scripting language), developers can use Tendermint for Byzantine fault tolerance state machine replication of applications written in whatever programming language and development environment is right for them.

CONCLUSION

The analysis provided in the paper shows that there are a lot of various implementations of blockchain platforms. They are differing by a lot of parameters, and most important are protocol of block creations, consensus for block adding, smart contract support (environment, program language, functions). To create connection between Industrial IoT and blockchain platform it is needed to create module that will unite IoT and blockchain functions. This module will allows storing information in smart space of IIoT and duplicate it with signed transaction in blockchain. Also, it will be possible to create contracts that can be processed automatically in proper conditions. The most preferable platforms for this kind of integration is Hyperledger Fabric/Burrow. They provide fault tolerant consensus mechanism as well as built-in infrastructure to create and process smart contracts. Both types of blockchain network — private and public can be created with these platforms.

Also, it is possible to deeply integrate IIoT and blockchain by creating own blockchain network over the IIoT infrastructure. For this purpose, the paper provides the list of libraries for creating own blockchain function. These libraries provide ready-to-use solutions for various consensus mechanisms, but developers should create own infrastructure for smart contract, for example by containers technology.

The future work will be concentrated on the first case, with developing the module for IIoT and private blockchain integration. Also, the second case will be partially used to implement deep integration for consensus mechanisms that provide higher speed of block adding and sharing between all participants.

ACKNOWLEDGMENT

The research is funded by the Russian Science Foundation (Project № 17-71-10223).

REFERENCES

[1] N. Teslya and I. Ryabchikov, “Blockchain-Based Platform Architecture for Industrial IoT,” in Proceeding of the 21st conference of FRUCT Association, 2017, pp. 321–329.

[2] A. Smirnov, A. Kashevnik, A. Ponomarev, N. Shilov, M. Shchekotov, and N. Teslya, “Smart space-based intelligent mobile tourist guide: Service-based implementation”, *Conference of Open Innovation Association, FRUCT*, 2014, pp. 126–134.

[3] A. Smirnov, A. Kashevnik, N. Shilov, S. Balandin, I. Oliver, and S. Boldyrev, “On-the-fly ontology matching in smart spaces: A multi-model approach,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6294, pp. 72–83, 2010.

[4] O. Goldreich, *Foundations of cryptography I: Basic Tools*, Cambridge: Cambridge University Press, 2001.

[5] A. Smirnov, A. Kashevnik, N. Shilov, and N. Teslya, “Context-Aware Access Control Model for Privacy Support in Mobile-Based Assisted Living”, *J. Intell. Syst.*, vol. 24, no. 3, pp. 333–342, 2015.

[6] C. Cachin and M. Vukolić, "Blockchain Consensus Protocols in the Wild," Jul. 2017.
[7] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Www.Bitcoin.Org, p. 9, 2008.
[8] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (PoET)," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017, vol. 10616 LNCS, pp. 282–297.
[9] P. Daian, R. Pass, and E. Shi, "Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proofs of Stake," *Iacr*, pp. 1–64.
[10] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," *OSDI {'}99 Proc. third Symp. Oper. Syst. Des. Implement.*, no. February, pp. 173–186, 1999.
[11] D. Mazières, "The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus," pp. 1–45, 2015.
[12] Ethereum Available: https://www.ethereum.org/ [Accessed: 01-Mar-2018]
[13] "Hyperledger Sawtooth - Hyperledger." [Online]. Available: https://www.hyperledger.org/projects/sawtooth. [Accessed: 01-Mar-2018].
[14] N. A. Simakov *et al.*, "Effect of Meltdown and Spectre Patches on the Performance of HPC Applications," no. January, pp. 1–8, 2018.
[15] S. King and S. Nadal, "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake," Cambridge University Press, Cambridge, 2012.
[16] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.
[17] Hyperledger Fabric Available: https://hyperledger-fabric.readthedocs.io/ [Accessed: 01-Mar-2018]
[18] "Introduction — Tendermint documentation." [Online]. Available: https://tendermint.readthedocs.io/en/master/introduction.html#abci-overview. [Accessed: 01-Mar-2018].
[19] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda : An Introduction," pp. 1–15, 2016.
[20] "What is Exonum - Exonum Documentation," 2018. [Online]. Available: https://exonum.com/doc/get-started/what-is-exonum/. [Accessed: 01-Mar-2018].
[21] Ripple, "Product overview," no. October, 2017.
[22] "Stellar Basics - Stellar." [Online]. Available: https://www.stellar.org/how-it-works/stellar-basics/. [Accessed: 01-Mar-2018].
[23] A. Bessani, J. Sousa, and E. E. P. Alchieri, "State Machine Replication for the Masses with BFT-SMART," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 355–362.
[24] L. W. Cong, Z. He, and J. Zheng, "Blockchain Disruption and Smart Contracts," *SSRN Electron. J.*, p. 48, 2017.