Federated Byzantine Agreement to Ensure Trustworthiness of Digital Manufacturing Platforms

Johannes Innerbichler
The IoT Group, Salzburg Research
Jakob Haringer Strasse 5/II, 5020 Salzburg, Austria
+43 662 2288-419
johannes.innerbichler@salzburgresearch.at

Violeta Damjanovic-Behrendt
The IoT Group, Salzburg Research
Jakob Haringer Strasse 5/II, 5020 Salzburg, Austria
+43 662 2288-427
violeta.damjanovic@salzburgresearch.at

ABSTRACT

In this paper, we explore the use of the Stellar Consensus Protocol (SCP) and its Federated Byzantine Agreement (FBA) algorithm for ensuring trust and reputation between federated, cloud-based platform instances (nodes) and their participants. Our approach is grounded on federated consensus mechanisms, which promise data quality managed through computational trust and data replication, without a centralized authority. We perform our experimentation on the ground of the NIMBLE cloud manufacturing platform, which is designed to support growth of B2B digital manufacturing communities and their businesses through federated platform services, managed by peer-to-peer networks. We discuss the message exchange flow between the NIMBLE application logic and Stellar consensus logic.

• Security and privacy→System security→Distributed system security • Security and privacy→Network security→Security protocols.

Keywords

Trustworthiness; Consensus Protocol; Federated Byzantine Agreement; Distributed Ledger Technology; Federated Manufacturing Platform; Smart Manufacturing.

1. INTRODUCTION

One of the major challenges of federated digital manufacturing platforms is to ensure trust between platform instances (nodes) and their participants, in a way that enforces trustworthiness of collaboration platforms, the integrity of performed actions and measurements, and correctness of their recording, which further encourages new organizations to join and extend their businesses to new collaboration models and new communities. Since some of the key benefits of the Distributed Ledger Technology relate to trust and transaction acceleration for the Internet of Things (IoT), in this paper, we explore the use of distributed ledgers and consensus protocols to ensure trust and reputation between various participants collaborating via the NIMBLE collaborative manufacturing platform. The development of NIMBLE is funded through the EU H2020 programme (website: https://www.nimble-project.org/), and it is designed as a federated, cloud-based collaborative platform for multi-sided B2B trade and enterprises. The platform supports business process negotiation and subsequent execution, between manufacturers, suppliers, service and logistics providers, software and cloud service providers, and retailers. Its business models require contractual relationships between all participants and their services affiliated with the platform. For each instance, the platform provides a set of services and offers additional, specifically tailored services for interoperation at regional, sectorial or topical levels.

The federated nature of the platform increases the complexity and range of enterprise interoperation, creating new business opportunities, reducing the need for traditional practices (e.g. financial, regulatory), but many issues are as yet, unresolved, e.g. those related to trust and reputation of the participants, data manipulation and integrity, jammed communication and spoofing. Hence, in this paper we explore the applicability of distributed trust algorithms to maintain agreement and trust through the NIMBLE distributed network and supply chains. Specifically, we explore the use of a consensus mechanism that employs the Federated Byzantine Agreement (FBA) algorithm, which is implemented within the Stellar consensus protocol (website: https://www.stellar.org/).

Paper organization. Section 2 briefly describes our motivation to explore the potential of Distributed Ledger Technology and federated consensus mechanisms for establishing secure and valid interoperation between federated platform instances. Section 3 reviews trust mechanisms in distributed systems, which are used for sharing content that is collectively confirmed (agreed) through the consensus. Here, we look at the Byzantine Fault Tolerance (BFT) mechanism, Distributed Ledger Technology (DLT) and a variety of existing community consensus mechanisms. Section 4 introduces the FBA algorithm for trust and reputation in NIMBLE, and describes its FBA background models (e.g. tiered quorum formation) and FBA consensus phases in the NIMBLE context. Section 5 illustrates the message flow between the Stellar Consensus Protocol (SCP) and NIMBLE applications. Section 6 presents our conclusions.

2. MOTIVATION

NIMBLE offers the technology for establishing a multi-sided federation of digital platforms, enabling communication and collaboration between the various country-, regional- or topic-centered platform instances and participants. For example, Figure 1 shows NIMBLE platform instances in three countries, Germany, Italy and Spain. The overall platform architecture topology is distributed, and requires non-centralized algorithms to compute trust and reputation through a federated ecosystem.

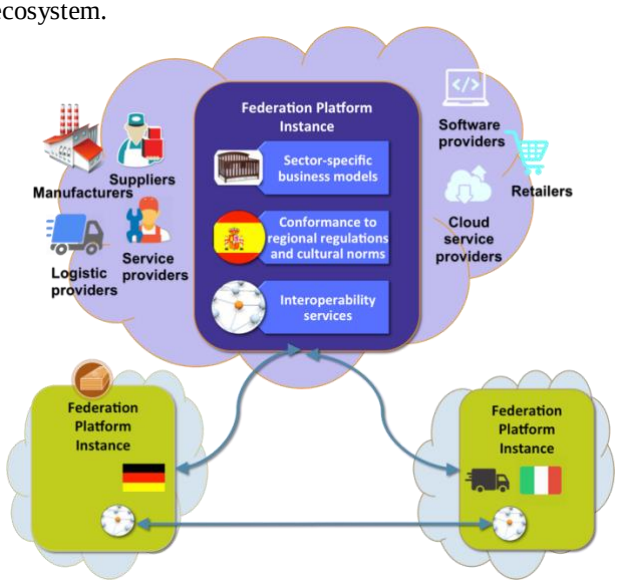

Figure 1. NIMBLE federated instances.

One of the platform's requirement towards a federated ecosystem is to support open membership enabling new organizations with modest resources to join the platform, to extend their businesses to new communities, and to create innovative business models through collaborations, and thus fulfill their aspirations for economic growth. The main challenge for such a business ecosystem is to ensure that participants' actions and transactions via the platform are recorded correctly in order to achieve trustworthiness at the system level. Distributed Ledger Technologies and fault tolerant system behavior are new elements that need to be integrated into systems development methodologies.

3. BACKGROUND AND RELATED WORK

This section summarizes (i) background mechanisms for DLT, (ii) community consensus mechanisms, e.g. Proof-of-Work, Proof-of-Stake, Proof-of-Importance, Byzantine Agreement and Federated Byzantine Agreement, and (iii) consensus protocols, e.g. Ripple and Stellar.

3.1 Byzantine Fault Tolerance in distributed systems

BFT algorithm is created to address the Byzantine Generals Problem, which is a logical dilemma explained in [1]. It suggests a scenario in which a group of Byzantine generals and their armies surround an enemy city that they plan to attack. The attack preparation involves sending a messenger from one army to the next, because in order to be successful, all armies must attack at the same time. However, the generals know that there are one or more traitors involved in the communication, who will try to confuse the others. The BFT algorithm ensures that the agreement for attack will not be compromised through untrustworthy messages. In other words, it needs to guarantee that (i) all loyal parties decide upon the same plan of actions, and (ii) a small number of traitors cannot cause the loyal parties to adopt a bad plan [1]. The potential solutions could range from a solution with *oral messages* (every message that is sent is delivered correctly; the receiver of the message knows who is the sender; the absence of a message can be detected) to a solution with *signed messages* where anyone can verify the authenticity of the loyal party's signature.

Similarly, in distributed digital manufacturing environments with multiple actors, the Byzantine Generals Problem can be used to simulate the risk of producing incorrect or inconsistent outputs that can lead to breakdown of the system. The failures in distributed systems can occur either as (i) *omission failure* i.e. not receiving a request, or failing to respond to a request, and (ii) *execution failure*, due to sending incorrect or inconsistent data, or responding to a request incorrectly. The authors in [1] showed that Byzantine resilient (fault tolerant) systems that implement BFT solutions are expensive in traditional networks: they require significant amounts of time and numbers of messages in order to guarantee the reliability of the system.

3.2 Distributed Ledger Technology

Distributed Ledger Technology (DLT), commonly called blockchain, is an emerging distributed data architecture for processing digital transactions over a business network. It tracks both tangible (i.e. car, house) and intangible (i.e. brand, copyright) assets involved in transactions, and facilitates the process of recording performed transactions. It can be also seen as a critical enabler of Digital Identity with the potential to minimize fraud and enable asset provenance and full transaction history [2]. Other key utilities of DLT and blockchains are *contract management* between two parties involved, *regulatory compliance*, *tokenization* for the authentication of physical items, when the items are paired with a corresponding digital token. The authors in [2] emphasize the following benefits of blockchains in financial scenarios:

- *Transaction immutability* – eliminates inclusion of an enforcer of trust in the ecosystem;
- *Transparency between all participants* – provides transparency for historical and real time transactions;
- *Transaction autonomy* – guarantees transaction execution under mutually agreed conditions and accelerates business outcomes.

3.3 Community consensus mechanisms

DLT and blockchain uses BFT and community consensus to legitimate transactions. In blockchain, new transactions are added into new blocks, to the end of the chain, broadcast to all the nodes, and can never be changed or removed once accepted by the network. If members of the community send inconsistent, inaccurate or malicious transactions information to others, the reliability of the blockchain breaks down. Hence, the consensus mechanisms are necessary in blockchain systems, to protect against the Byzantine Generals Problem.

There are several approaches to consensus mechanisms, supporting both reputation and trusted identity claims [3]:

- Proof-of-Work (PoW) algorithm [4][5] is designed to protect against ill-behaviour of the participants who do not possess the majority of the system's computing power. PoW is a basis of Bitcoin, requiring from anyone who wants to add new information to the blockchain to perform a work-intensive task, e.g. must use information from the existing blockchain [6]. PoW takes a fair amount of time to execute, which guarantees a practical protection against manipulation of the blockchain, enjoying a measure of protection against "51% attacks" [3][7]. An alternative solution for PoW relies on node votes and majority consensus in order to root out faults. The downside to this strategy is that it provides protection against Byzantine faults only as long as a relatively large majority of nodes on the blockchain continues to act legitimately [6]. Although BFT has been studied in Distributed Systems for a long time, after the Practical BFT (PBFT) was introduced in 1999 [8], there were no practical implementations of BFT until the emergence of the PoW algorithm.
- Proof-of-Stake (PoS) algorithm [9] calculates consensus based on parties that have posted some collateral to prove their value. This opens the possibility of so-called "nothing at stake" attacks, in which parties that previously posted some collateral but later spent the money, can go back and rewrite history from a point where they still had stake. To mitigate such attacks, systems combine PoS and PoW, or delay refunding collateral long enough for some other consensus mechanism to establish a checkpoint. Some other approaches based on PoS are Leased Proof-of-Stake (LPoS) and Delegated Proof-of-Stake (DPoS) [10]. LPoS allows holders to lease their balances to staking nodes, which increases the weight of the staking nodes and their chances of being allowed to add a block of transactions to the blockchain. DPoS enables holders to use their balances to elect a list of nodes with the opportunity to stake blocks of new transactions and add them to the blockchain.
- Proof-of-Importance (PoI) algorithm considers factors such as balance, reputation (determined by a separate purpose-designed system), and the number of transactions made to and from a specific address, which indicates productivity of network nodes that should be rewarded [10].
- Byzantine Agreement [11] ensures consensus in a fast and efficient way, enforcing trust and helping a small non-profit organization to keep more powerful organizations, such as banks or CAs, honest. Complicating matters, however, all parties must agree on the exact list of participants, and attackers must be prevented from joining multiple times and exceeding the system's failure tolerance.
- Federated Byzantine Agreement (FBA) overcomes situations in which malicious parties are joining many times in order to outnumber the honest nodes, and create the Byzantine General Problem. FBA determines decentralized quorums by allowing each node to select quorum slices – individual trust decisions made by each node that together determine system-level quorums. FBA avoids complete lists of accepted participants that are necessary for ensuring consensus on a system level, and supports open membership that promotes organic network growth. It also has modest computing and financial requirements, in comparison to PoW and PoS.
- Tendermint is based on PoS algorithm, designed with the goal to provide more efficient consensus solution then PoW-based systems [12]. Over time, it became a general purpose blockchain consensus engine.
- HoneyBadgerBFT is the first efficient asynchronous BFT protocol, presented in [13].
- Ripple Consensus Protocol is designed to target financial institutions. It provides a distributed ledger solution to facilitate cross-border payments. Its consensus mechanism uses probabilistic voting to calculate consensus success [14].
- Stellar Consensus Protocol allows for solving problems through reaching consensus among network nodes [3]. It targets individual participants, rather than financial institutions. Stellar has a strong focus on technology, and uses an API based on the External Data Representation (XDR) standard [15].

4. FEDERATED BYZANTINE AGREEMENT FOR TRUST AND REPUTATION IN BUSINESS PLATFORMS

Since in decentralized and distributed systems there are no central authorities to govern interactions and agreements between participants, trust and reputation of participant parties is still a major challenge. Similar problems occur in federated ecosystems, where low entry barriers should spur the organic growth of the systems. The NIMBLE platform is designed to support collaboration between companies interacting with each another on a daily basis, and here, trust and reputation are becoming major concerns.

To support trustworthiness between companies registered in one platform instance and collaborating with other

companies either from the same instance or from another NIMBLE instance, it could be necessary to associate a federated identity to each company and provide appropriate mechanisms which allow multiple authorities to access and validate globally recognized entities. Depending on the business context, federated identities can vary and have a different representation across platform instances. Hence certain collaboration scenarios may require additional authentication and verification mechanisms.

Furthermore, it would be necessary to make a clear distinction between delegated identities and federated identities. In a system with delegated identities, the identity management is outsourced to another system, while in a system with federated identities, every participant can keep its own entity information in multiple nodes (e.g. NIMBLE instances). In the following, we explore the applicability of FBA to ensure trust and reputation between the platform instances and their participants in the NIMBLE ecosystem.

4.1 FBA background mechanisms

In FBA, a consensus protocol ensures that all participants agree on updating a replicated state that is called slot (e.g. transaction ledger), which helps participants to avoid contradictory states [3]. Each participant in the FBA system can safely apply update *x* in a specific slot when it has safely applied updates in all other slots upon which a specific slot depends, and when it believes that all other participants will agree on update *x* for that specific slot. In FBA language, this is described as "participant has *externalized* update *x* for a specific slot".

4.1.1 Quorum slices and quorum intersections

Agreement in FBA is accomplished by allowing every participant to decide on its own set of trusted neighbors, some of which may exhibit various types of non-rational behavior e.g. malicious behavior, unavailability, random errors, etc. A set of participants that is sufficient to reach agreement is called *quorum*, while a *quorum slice* is a subset of the quorum that can be selected based on arbitrary criteria, e.g. reputation or financial arrangements [3]. A participant agrees to a specific statement if there exists at least one quorum slice, which also agrees to the same statement. Another important property for ensuring the safety of an FBA-based system is *quorum intersection*. If a system lacks quorum intersection, quorums can independently agree on contradictory statements. In other words, quorum intersection exists iff any two of quorums share at least one node [3].

4.1.2 Tiered quorum in federated platforms

In NIMBLE, we apply a tiered quorum structure, in which each platform instance is represented by a node (see Figure 2). The top-level tier is composed of instances (e.g. *ESP-1 (Spain)*, *IT-1 (Italy)*, and *D-1 (Germany)*), which are governed by well-known and trusted state authorities and, therefore, enjoy a high level of trust. In the example in Figure 2, every top-level node agrees to a statement iff at least two other nodes at the same level agree on the same statement. Sub-level tiers are constituted from nodes within a specific country (e.g. *ESP-2*, *ESP-3*, *ESP-4*, *ESP-5*), and must find trust from at least one node in the top-level tier.

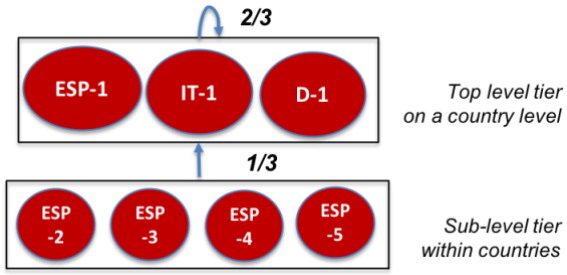

Figure 2. Tiered quorum structure for ensuring trust between federated instances in NIMBLE.

The above presented tiered architecture increases trust, since at least two instances from the top-level are necessary to ensure system-wide agreements. In addition, not every single instance needs to be constantly available, which results in a more fault-tolerant system. FBA guarantees that all well-behaving nodes will externalize the same statement even in the presence of ill-behaved nodes.

4.2 FBA consensus phases

Agreement to a specific statement *c* requires the exchange of messages between participants (nodes) (Figure 3). The process of consensus at the level of a single node evolves in three phases, from (i) unknown, when nodes "vote for statement *c*", via (ii) accepted, when two nodes either succeed in agreement or show that statement *c* is contradictory, to (iii) confirmed, when both nodes send acceptance messages and confirm that statement *c* is true.

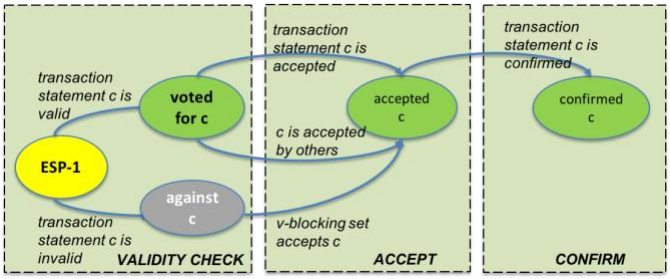

Figure 3. The consensus phases and agreement of an accepted statement *c* at a single node ESP-1.

4.2.1 Federated statement acceptance

Federated agreement at a system-wide level allows open membership, but this set-up bears the risk that a majority of well-behaved nodes can be broken. The challenge here is for the well-behaved nodes to discover ill-behaved ones and to arrive at a quorum intersection of well-behaved nodes. In FBA, there is a term called *v-blocking* that identifies failed nodes (Figure 3) [3].

4.2.2 Federated statement confirmation

Statement *confirmation* means that a node *v* claims to accept statement *c* and confirms *c* iff an intact node *v* enjoys a quorum intersection. According to Theorem 11 in

[3], once sufficient messages are delivered and checked, every intact node *v* will accept and confirm statement *c*.

4.3 FBA safety, liveness and fault tolerance

A distributed consensus protocol has to ensure system-wide safety, liveness and fault tolerance [3]. Safety is achieved if all correct instances agree or disagree on a certain statement that was initially proposed by one of the instances. The SCP solves this issue by attaching full sets of quorum slices to each propagated message.

Another important feature of the protocol is known as liveness of the system. In an FBA system, participants are not allowed to change their decisions for a statement after it was distributed to other participants. This may lead to a situation where an agreement on a statement gets stuck. Therefore, the consensus protocol has to ensure that the system agrees or disagrees with a statement after a finite amount of time. The SCP supports a federated voting mechanism (see Section 4.2), with voting of the nodes starting in a *bivalent* state (neither agrees nor disagrees with *a*). After enough votes were cast the state of the system changes to either *a-valent* (nodes vote for statement *c*) or a contradicting *ā-valent* state (nodes vote against statement *c*). The system can also end up in a stuck state, when it is not capable of finding a solution, due to the fact that nodes are not allowed to change their votes in a later phase. The SCP [3] avoids stuck states by applying neutralizable statements, which overcomes this problem.

The third feature of the protocol is known as fault tolerance. At any point in the execution of the protocol, the system should be able to recover from a failure of a node. The authors in [8] present a fail-stop model that describes situations where a node crashes and stops sending messages to other nodes. In BFT, it can be assumed that nodes fail by behaving arbitrarily, e.g. the node is taken over by an attacker and sends compromising messages to the system.

5. BRINGING STELLAR COSENSUS TO FEDERATED BUSINESS PLATFORMS

Our objective is to enable a federated system like NIMBLE to agree on statements in a decentralized manner. The role of SCP is to define well-structured communication and message exchange between distributed platform instances and their participants. Figure 4 illustrates the message flow between application logic and Stellar consensus logic.

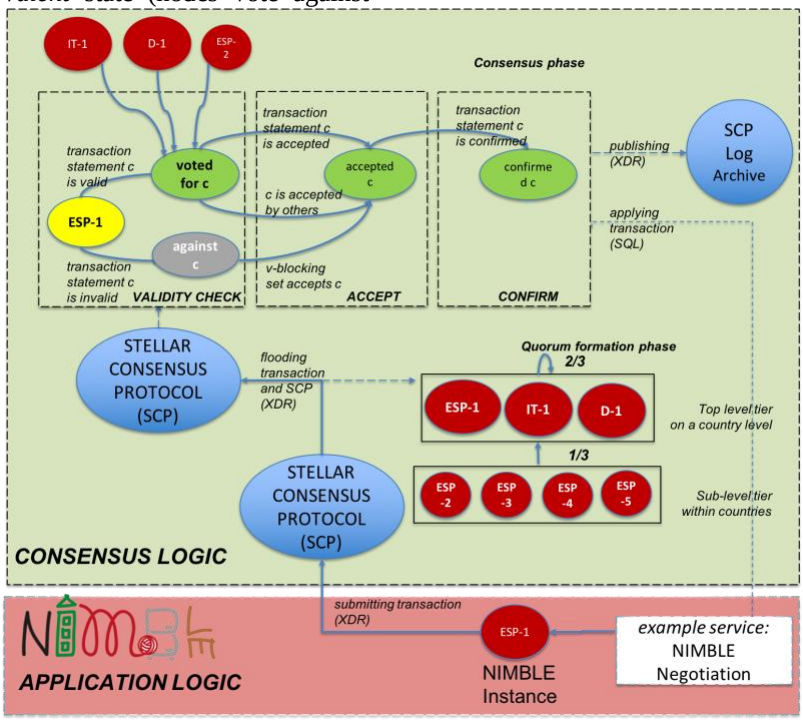

Figure 4. The message flow between NIMBLE application logic and Stellar consensus logic.

A specific NIMBLE platform instance in Spain, ESP-1 in Figure 4, is in the process of negotiating logistics details with a partner organization in Italy. To check trustworthiness of that partner, ESP-1 needs to submit a transaction to SCP. If ESP-1 is an external client to SCP submitting this new transaction, SCP contacts peers (through HTTP), submits an XDR transaction representation, and ESP-1 receives a status code of either "rejected" or "pending" [16]. If ESP-1 is not an external client to SCP but peer that already holds TCP connections to other peers, it has already defined quorum slices at the country level (top level tier). ESP-1 submits a transaction message in XDR format which is repeated to all peers (called "flooding" in [16]). SCP decides on the consensus state and the results are recorded in the SCP Log Archive (in XDR format) and sent back to the application (in our case, the NIMBLE negotiation service).

5.1 Embedded Architectural Components

Implementing SCP in NIMBLE requires several components to be added to the existing microservice architecture of the NIMBLE platform. Figure 5 shows the composition of the consensus component (green) and the platform services (red). Each consensus component will be realised in loosely coupled units, with inter-component communication executed via HTTP. The *Consensus Logic* component exchanges votes with other instances in the federated network and manages the formation of a quorum.

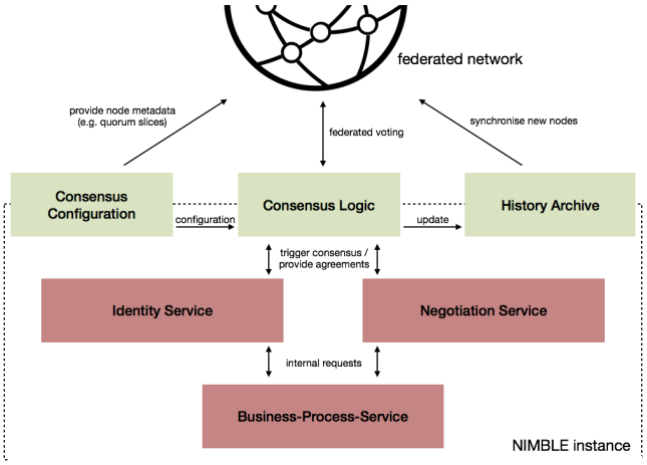

Figure 5. Consensus components embedded in the NIMBLE architecture.

Platform services, e.g. *Identity Service* and *Negotiation Service*, communicate with the *Consensus Logic* in order to find system-wide agreements for new statements. Configurable metadata of individual nodes is saved in the *Consensus Configuration* component, whose role is also to provide necessary information (i.e. quorum slices) for finding consensus. Metadata of nodes is at the same time shared internally with the consensus logic and is publicly available for other nodes. Each agreement is stored in the *History Archive*, which provides historical information for synchronising new nodes in the network.

6. CONCLUSION

The federated nature of the NIMBLE collaboration platform poses design and development challenges, related to data and message sharing strategies, security and privacy of data and controlled data access across the platform. In this paper, we have explored federated consensus principles based on FBA, supporting open membership in NIMBLE. We have explored data flow between NIMBLE services and SCP mechanisms, and identified important components for implementation of SCP in federated ecosystems.

ACKNOWLEDGMENTS

This research has been funded by the European Commission within the H2020 project NIMBLE, No. 723810, for the period between 01 October 2016 - 31 September 2019.

7. REFERENCES

[1] L. Lamport, R. Shostak, and M. Pease, 1982. The Byzantine Generals Problem. ACM Transactions on Programming Lang. and Sys., Vol. 4, No. 3,382—401.

[2] McWaters, R.J., 2016. The future of Financial Infrastructure. World Economic Forum 2016. Online: http://www3.weforum.org/docs/WEF_The_future_of_financial_infrastructure.pdf Last accessed March 2018.

[3] Mazières, D., 2015. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus. Online: https://www.stellar.org/papers/stellar-consensus-protocol.pdf Last accessed in March 2018.

[4] Dwork, C., Lynch, N. and Stockmeyer, L, 1988. Consensus in the Presence of Partial Synchrony. Journal of the ACM 35, pp. 288–323.

[5] Dwork, C. and Naor, M., 1992. Pricing via Processing or Combatting Junk Mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pp. 139–147.

[6] C. Tozzi, 2017. Byzantine Fault Tolerance: The Key for Blockchain. Online available from: https://www.nasdaq.com/article/byzantine-fault-tolerance-the-key-for-blockchains-cm810058

[7] Eyal, I. and Sirer, E.G., 2013. Majority is not Enough: Bitcoin Mining is Vulnerable. Online available from: http://arxiv.org/abs/1311.0243 Last access Feb. 2018.

[8] Castro, M. and Liskov, B., 1999. Practical Byzantine Fault Tolerance. In *Proceedings of the 3rd Symposium on Operating Sys. Design and Implem.*, pp. 173–186.

[9] King, S. and Nadal, S, 2012. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Online: http://peercoin.net/assets/paper/peercoin-paper.pdf.

[10] Kostarev, G., 2017. Review of Blockchain Consensus Mechanisms, Waves Platform. Online available: https://blog.wavesplatform.com/review-of-blockchain-consensus-mechanisms-f575afae38f2 Accessed 2018.

[11] Pease, M., Shostak, R., and Lamport, L., 1980. Reaching Agreement in the Presence of Faults. Journal of the ACM 27, 228–234.

[12] Kwon, J., 2014. Tendermint: Consensus without Mining. http://tendermint.com/docs/tendermint.pdf Last accessed in March 2018.

[13] Miller, A., Xia, Y., Croman, K., Shi, E., Song, D., 2016. The Honey Badger of BFT Protocol. Online available from: https://eprint.iacr.org/2016/199.pdf

[14] Schwartz, D., Youngs, N., and Britto, A., 2014. The Ripple Protocol Consensus Algorithm. Online: https://ripple.com/files/ripple_consensus_whitepaper.pdf Last accessed in March 2018.

[15] External Data Representation (XDR) Standard. Online: https://tools.ietf.org/html/rfc4506.html

[16] Stellar Core Data Flow. Online available from: https://www.stellar.org/developers/stellar-core/software/core-data-flow.pdf