Context-based consensus for appendable-block blockchains

Roben C. Lunardi*†, Maher Alharby‡, Henry C. Nunes*, Avelino F. Zorzo*, Changyu Dong‡ and Aad van Moorsel‡
*Pontifical Catholic University of Rio Grande do Sul (PUCRS) - Porto Alegre, Brazil
†Federal Institute of Education, Science and Technology of Rio Grande do Sul (IFRS) - Porto Alegre, Brazil
‡ Newcastle University - Newcastle upon Tyne, UK
E-mail: roben.lunardi@restinga.ifrs.edu.br, M.W.R.Alharby2@newcastle.ac.uk, henry.nunes@edu.pucrs.br,
avelino.zorzo@pucrs.br, changyu.dong@newcastle.ac.uk, aad.vanmoorsel@newcastle.ac.uk

***Abstract*—Blockchain technology has been applied to various applications (*e.g.*, smart buildings and smart cities) that typically run in an environment of smart devices, known as Internet-of-Things (IoT). To support these applications, different blockchain architectures, data structures and consensus algorithms have been proposed, tailored to IoT. One such proposal, appendable-block blockchain, is a promising blockchain framework for use in IoT environments. It provides a scalable data structure that allows parallel insertions between independent nodes. However, it has some limitations, in particular related to the possible eclipse attack by malicious gateways and the lack of consensus for transactions insertion. To solve these issues, we propose a new consensus mechanism for appendable-block blockchains, called context-based consensus. Using context-based consensus, information can be inserted in parallel across devices (called context) while ensuring that light-weight consensus is performed to guarantee that a transaction is well-formed and it is placed in the correct order. We implemented context-based consensus and show that using multiple contexts reduces latency and increases the throughput of transaction insertions when compared to consensus without contexts or using single transaction consensus.**

***Index Terms*—Consensus, Blockchain, IoT, appendable-block**

I. Introduction

Blockchain technology is being adopted to facilitate decentralization and ensure security in areas such as education [1], healthcare [2], general Internet-of-Things (IoT) [3], real estate registries [4] or supply chains [5]. The underlying infrastructure in these applications is the Internet of Things and different blockchain architectures [6] [7] and data structures [8] [9] have been proposed to deal with the challenges offered by IoT, including security challenges such as limitations to the hardware capacity, sensitivity of device information, or the use of devices in botnets [10].

Appendable-block blockchain [11] is such a blockchain proposal for IoT, proposed for permissioned and private IoT environments. Appendable-block blockchains use an hierarchical architecture and a bespoke data structure (with separated insertion of blocks and transactions), which allows to insert transactions in parallel across nodes. In appendable-block blockchains, consensus is only performed when creating and inserting a new block for a node [12]. Once the block is created for a specific node, the node can attach transactions to the block. There is no consensus to insert such transactions in appendable-block blockchains. That is, nodes have to trust its gateway - a full node that controls the access to other nodes to the blockchain - to insert valid transactions in their blocks. In addition, appendable-block blockchains assume that devices connect to only one gateway at a time. Consequently, appendable-block blockchains are susceptible to misuse and attacks through malicious or tampered gateways. Such gateways can compromise the insertion of information (e.g., insert an invalid execution of a smart contract) and can eclipse devices or hide devices information (not inserting that into the blockchain).

To tackle these problems, we propose a new consensus mechanism for appendable-block blockchains called *context-based consensus*. This mechanism allows parallel insertion aggregated in independent contexts. Each context can have different nodes and can be defined by the consortium of organizations that control the blockchain. Every transaction inserted in the blockchain has to pass through a validation procedure and agreement between nodes, providing trust among nodes. Additionally, by using a parallel approach (separated in contexts), this context-based consensus can increase the throughput of inserted transactions when compared to single context insertion (traditional consensus). As a consequence of parallelism, this consensus can help reduce the latency to insert transactions in the blockchain.

We evaluate the performance of context-based consensus through a prototype implementation. In the evaluation, we use a smart building scenario, composed of 1,000 devices and 10 gateways and compare the performance for two different scenarios, under various transaction pool configurations and two distinct update approaches. The results show that using multiple contexts can reduce latency and increase the throughput of transaction insertions when compared to consensus for single transaction insertion or consensus for single context, achieving a total latency under 550ms and throughput above

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. Avelino F. Zorzo is supported by CNPq (315192/2018-6) and FAPERGS. This work was supported by the INCT Forensic Sciences through the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq − process # 465450/2014-8). Also, we thank the support from IFRS.

978-0-7381-0495-9/20/$31.00 ©2020 IEEE
DOI 10.1109/Blockchain50366.2020.00058

100 transactions per second.

II. Background - Appendable-block blockchains

Due to hardware limitation (memory and processing power), IoT devices are susceptible to attacks that expose sensitive information [13], lead to catastrophic situations [14], or be tampered with to be used as a botnet [15]. In order to solve many IoT security issues, Christidis and Devetsikiotis [10] proposed the adoption of blockchain in IoT.

Many proposals presented different ways to use existing blockchains in IoT, in particular using a hierarchical architecture [6] or using blockchain as a service [7]. However, for many applications that use IoT devices, latency and throughput are important factors that should be considered when designing a solution [3]. Consequently, consensus algorithms [16] [17] [12] were adapted to existing blockchains and new data structures, such as appendable-block blockchains [11] and directed acyclic graphs (DAG) [9], were proposed to be used in IoT environments. In our work, we intend to improve appendable-block blockchains to solve some security issues and to provide a new consensus mechanism to the transaction insertion in the appendable-block blockchains.

Appendable-block blockchain was proposed to be used in IoT environments using a different data structure that allows the insertion of transactions after a block was inserted in the blockchain [8] [11]. It adopts a layered IoT architecture, composed of devices, gateways and service providers. In this architecture, devices produce data and send them to the gateways who append these data to the blockchain. Devices can be understood as light-nodes in the blockchain, *i.e.*, they do not store blockchain data. Gateways are responsible for controlling the access and insertions in the blockchain. Service Providers can be understood as middleware to access the blockchain information from the gateways.

Similar to other blockchains, appendable-block blockchains have transactions stored inside blocks. The difference is that every node (device, gateway or service provider) has a unique block assigned to it, *i.e.*, identified by its public key. The block for a node is created and attached to the blockchain when the node submits the first transaction (it can be understood as a genesis transaction). After having the block attached to the blockchain, the node can insert new transactions into it, resulting in a chain of transactions, *e.g.*, each transaction is linked to the previous one. More details in Fig. 1.

To handle block insertions, appendable-block blockchain allows to use different consensus algorithms. For example, it can use simplified witness-based insertion or Practical Byzantine Fault Tolerance (PBFT) [12]. Every time that a new device (a device that does not have its public key in a block) tries to connect to a gateway, it starts a consensus algorithm to insert a new block. After the block is inserted, the gateway can update that block with transactions sent by the device without committing the consensus algorithm, leading to many issues such as: (*i*) the insertion of invalid data; (*ii*) the inconsistency in the block's data when a device connects to multiple gateways at the same time; (*iii*) as devices

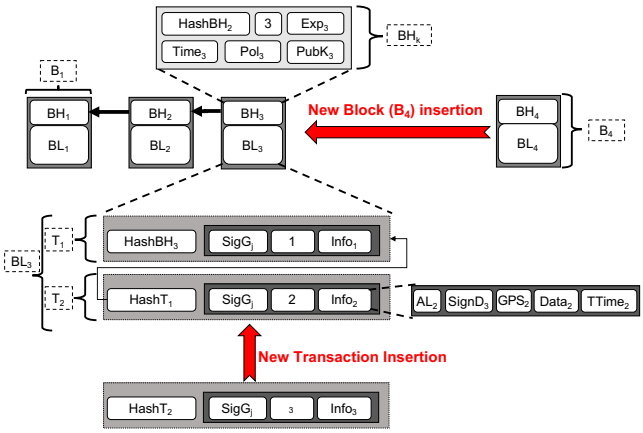


Fig. 1: Appendable-block blockchain block insertion and transaction insertion (adapted from [12]).

are connected to only one gateway per time, devices can be eclipsed by malicious gateways, *i.e.*, the communication and information produced by a device is intercepted by the gateway. To solve these security issues, we present in Section III a proposal for a context-based consensus algorithm.

III. Context-based consensus algorithm

Currently, SpeedyChain [11] - first proposed appendable-block blockchain - presented a solution that allows inserting new transactions into already inserted blocks. The consensus is only performed when creating and inserting new blocks, and there is no consensus to insert transactions into blocks [12]. However, some issues were not properly addressed by the current version of SpeedyChain. Firstly, current proposals of appendable-block blockchains have a communication protocol that allows a device to connect to only a single gateway. As a consequence, transactions produced by devices can be omitted by a malicious gateway. Secondly, the consensus is performed only to insert new blocks, which means that invalid transactions can be included. Finally, scalability can be a problem as the usage of consensus in the current appendable block blockchain would be performed individually for each transaction, leading to latency issues as we will present in the evaluation.

We propose a context-based consensus algorithm to address the limitations of the current version of SpeedyChain. In particular, to solve the first issue, we propose that every device should connect to a minimum initial set of gateways. When a device connects to multiple gateways, this eliminates a malicious gateway from cheating as other gateways will detect it, differently to what would happen if a device was connected to a single gateway. A simplified version of the connection protocol is presented in Fig. 2. The main steps are described as follows:

1) Device a (represented *Dev a* in Fig. 2) sends a Hello message with its own public key *Dev a Pubkey*, *e.g.*,

for encryption using the asymmetric cryptography, to gateway A (*Gw a*);

2) *Gw a* verifies if *Dev a PubKey* is in a block header of the blockchain, *i.e.*, a block for that device was inserted previously in the blockchain:
   a) In the case that the *Dev a PubKey* is not in a block header and the device is allowed to access the network, *Gw a* starts a consensus to include a new block for *Dev a* containing its *PubKey*;
      i) Other gateways (*Gw b* and *Gw c* in Fig. 2) verify the proposed new block, they vote (signed voting) and send the result back to the gateway that started the consensus;
3) After the consensus (if the block is considered valid), the block containing *Dev a PubKey* is inserted in the blockchain. Then, if *Dev a PubKey* is in the blockchain, *Gw a* and *Dev a* can establish an encrypted channel using symmetric cryptography;
4) After a device connects to a gateway, they can exchange information. Our proposal allows a device to send the same transaction to multiple gateways. These multiple connections with gateways can help to avoid a device from being eclipsed by a tampered gateway. Allowing the connection to multiple gateways is an improvement to appendable-block blockchain, as discussed previously. Also, it is important to note that devices' transactions have a timestamp and a digital signature.
5) Any update from a device is a transaction in the blockchain.

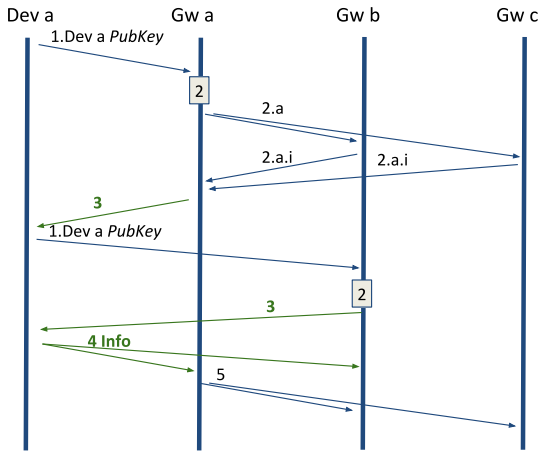


Fig. 2: Device connection simplified protocol.

As mentioned previously, every device can connect and send its update (a new transaction) to multiple gateways at the same time. For instance, *Dev a* is connected to three gateways (Gw a, Gw b and Gw c), as depicted in Figure 2.

As presented previously, appendable-block blockchains allow nodes to insert transactions into their own blocks at the same time, independently from each other. However, there are two remaining issues. One of them is that the current version of SpeedyChain does not present consensus at transaction level, *i.e.*, a gateway - to which a device is connected - inserts the transaction in the device's block ledger and sends it to the other gateways. In this case, different gateways can insert the same transaction in the blockchain (duplicating the same transaction), or in the worst case, propose an invalid transaction, *e.g.*, wrong result of smart contract execution. The other issue is that using one consensus procedure for each inserted transaction, without changing the way how transactions are inserted in the current version of appendable-block blockchains, can lead to scalability problems. A solution to these problems is to separate devices into different contexts. Consensus will be executed inside each context, and then propagated to gateways from different contexts. Thus, a new field called context should be added to the Block Header (presented previously in Fig. 1. This will allow to define that a device will participate in a specific context. The definition of which context a device will be part of is made by gateways during the device's block insertion. The rules to define this can be based on the type of information handled (gas sensor, lightning sensor, etc.) or other definitions that an organization/consortium will agree upon previously. In our proposal, we assume that the definition of contexts is based on existing predefined rules.

Context-based consensus consists of different contexts, where each context contains a number of devices. Consensus is performed in a context independently from other contexts. Consequently, each context can have different consensus or different parameters to be considered to append new information in the blockchain. Gateways can participate in consensus of different contexts, allowing them to participate in different consensus mechanisms (see Fig. 3). For example, Context Blue (*CB*) is composed of a set of gateways $\{GW\ A,\ GW\ B,\ GW\ C,\ GW\ D,\ GW\ E\}$, Context Yellow (*CY*) = $\{GW\ E,\ GW\ F,\ GW\ G,\ GW\ H,\ GW\ I,\ GW\ J\}$, and Context Red (CR) = $\{GW\ E,\ GW\ F,\ GH\ K,\ GW\ L,\ GW\ M,\ GW\ N\}$. In this example, the consensus algorithm used in CB can be different from the consensus algorithm used in CY and CR.

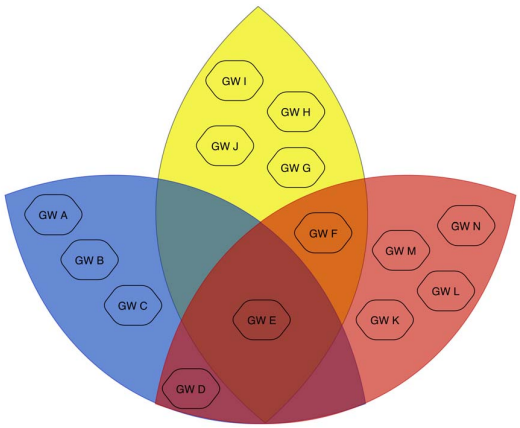


Fig. 3: Gateways in three different contexts.

After a consensus is performed inside a context, a gateway can share/propagate the new consented set of transactions to the gateways from the other contexts. For example, after a consensus in *CR*, *GW F* can share new information from *CR* with gateways in *CY*. This can be performed in two different approaches:

- (*i*) using the existing approach by sending the transactions with signed votes to a list of known gateways that do not participate in the context in which the consensus was performed, *e.g.*, *GW D* in Fig. 3 can share a set of valid transactions from CB with gateways to known gateways from CR. Every gateway that receives a transaction from known devices can share that with other gateways;
- (*ii*) using a new approach by sending transactions from a context or specific devices when they are requested by a gateway (on demand), *e.g.*, if *Gw I* wants to ensure that it has an updated view from a device from CB.

Approach (*i*) is similar to what is adopted in dBFT [18] and any other consensus algorithm that has a limited group of nodes performing consensus. This approach maintains an updated view (but not synchronous) of all transactions from every node. One issue that this approach can have is related to scenarios of a large amount of contexts and, consequently, many messages are exchanged between gateways to update all gateways' ledgers. However, the number of messages will be far less than performing the consensus by all gateways in the blockchain.

Approach (*ii*) can be adopted as a mechanism to avoid many update messages and, also, it can be used as a mechanism to update gateways that do not participate in the same context when requested. As the gateways do not participate in the consensus for that context, the information about transactions may not be used by that gateway. Additionally, this approach can be used to reduce the amount of data that is stored in each gateway. These data can be required if a gateway needs to use them for some processing, decision making, or they are requested by a Service Provider. This approach does not affect the replication of block headers, but can compromise the reliability and the number of copies of transactions.

Each context can have different configurations or different consensus algorithms. Each round can be defined by a set of transactions, that can be designed in different configurations:

- (*a*) one transaction (from that context) per time;
- (*b*) a set of transactions (from that context) generated during the time required to perform the previous context without a limit of transactions;
- (*c*) fixed maximum number of transactions per consensus.

Configuration *a* presents the same configuration used by the current version of appendable-block blockchain to insert transactions, *i.e.*, one transaction per time. This configuration can have a reduced latency to process the transaction for a device in a scenario that gateways are not overloaded. It is a simple approach, and each gateway can start a round of consensus. However, it can lead to a high number of consensus performed in a scenario with a high number of devices or a high rate of updates from devices from a context. In the end, the latency can be increased by the bottleneck in gateways.

Configuration *b* presents the same configuration available in many blockchain, *i.e.*, a limited set of transactions for each consensus. This configuration can reduce the number of consensus performed in the same context and, as a consequence, reduce the number of messages. However, it can lead to more time spent to verify all transactions and, as a consequence, more time can be required to perform the consensus. The gateway that starts the consensus (also known as leader) has to use all transactions produced in the context. However, this approach can increase the number of messages exchanged before the consensus (every gateway will have to send proposed insertions to the leader when requested). A problem with this approach is that overloading the gateways with many new transactions can lead to a time-consuming consensus.

Configuration *c* presents an alternative configuration, which may help to avoid high latency (or starvation) of transactions in overloaded situations. However, this configuration can increase the latency to insert a single transaction, but the number of messages exchanged will be reduced. The gateway that starts the consensus (also known as leader) has to use a limited set of transactions from the context. A problem that can happen is when too many transactions are produced in a small amount of time, *i.e.*, this approach can have a problem to handle an overloaded situation.

A context-based approach can reduce the number of messages exchanged to perform consensus for the transactions. However, some issues can happen when using this approach. For example, gateways that participate in many contexts can have issues regarding the high number of consensus messages, *e.g.*, GW E (in Fig. 3) participates in all contexts. A maximum amount of contexts for each gateway should be defined. Also, scenarios in which a small number of gateways participate in the consensus for a particular context is susceptible to Sybil and 1% attacks, similarly to shard approaches [19] or consensus with limited gateways [18].

Lunardi *et al.* [12] discussed the usage of PBFT in appendable-block blockchains to insert new blocks. That approach can still be used to insert new blocks, *i.e.*, having different consensus to block insertion similarly to the ones used for transaction insertion. In the next subsections, we present the algorithms for each different configuration.

A. Configuration a

New *data* that are produced by a node from a specific context (C_j) will be processed by a gateway (Gw_i) from that context, and it will be sent for a consensus. The *prepareConsensus(Tm)* and *commitConsensus(Tm)* functions used can be different for each scenario. Although, we assume, in this work, that operations are the same as used in PBFT [20], *i.e.*, every node receives a copy of the transaction in the prepare phase, and sends the vote to every other gateway (in the same context C_j) approving or not the new transaction on commit phase.

Algorithm 1 Perform transaction consensus - Config. a

Require: $Info_{\mathrm{m}}$ and Di
1: $validInfo \leftarrow$ **verifyInfo**$(Info_{\mathrm{m}})$
2: **if** $validInfo$ is **true then**
3: $T_{\mathrm{m}} \leftarrow$ **createTransaction**$(B, Info_{\mathrm{m}}, NPK_{\mathrm{i}})$
4: **for all** Gw_{i} in D_{j} **do**
5: **prepareConsensus**(T_{m})
6: **end for**
7: **for all** Gw_{i} in D_{j} **do**
8: $responseList \leftarrow$ **commitConsensus**(T_{m})
9: **end for**
10: **if** $|positive(responseList)| > minResponses$ **then**
11: **addTransaction**(T_{m}))
12: **end if**
13: **end if**

B. Configuration b

A gateway Gw_i will receive new *data* produced by a node from a specific context C_j. After processing that data, Gw_i will send it to a transactions list (or pool) and then process it in the next consensus.

Algorithm 2 Send transactions pool - Config. b and c

Require: $Info_{\mathrm{m}}$ and device NPK_{i}
1: $validInfo \leftarrow$ **verifyInfo**$(Info_{\mathrm{m}})$
2: **if** $validInfo$ is **true then**
3: **sendTransactionPool**$(Info_{\mathrm{m}}, BH_{\mathrm{b}})$
4: **end if**

Differently from *Configuration a*, we assume, in *Configuration b*, that operations prepare and commit use a set of transactions that will be voted as valid or not. It is important to note that variable z (line 1 in Alg. 3), which represents the limit of transactions, is set to zero (no limit is used in Configuration b). Also, we assume that a leader is elected for each consensus round. Similarly to *Configuration a*, we based the prepare and commit phases in what is adopted by PBFT [20]. Thus, every node receives a copy of the set of transactions in the prepare phase. After that, on the commit phase, every node sends the vote (approving or not each transaction in the set) to all other gateways (in the same context C_j). As a result, there is a list of votes (from all gateways) for each transaction.

C. Configuration c

Similar to *Configuration b*, all *data* produced by a device from a specific context C_j will be processed by Gw_i from that context. Also, a list will be processed in the consensus (Alg. 2). We assume, in *Configuration c*, that operations prepare and commit use a set of transactions with a predefined limit (z in line 1 in Alg. 3) that will be voted as valid or not. Also, we assume that a leader is elected for each consensus round. Also, every time that a consensus is finished a new one will be started but with a maximum amount of transactions per time, *i.e.*, the consensus is based on the time for each round but with a limited amount of z transactions.

IV. Evaluation

In order to evaluate context-based consensus algorithms in appendable-block blockchain, we performed testing with

Algorithm 3 Perform transaction consensus - Config. b and c

Require: $transactionPool$ and C_{j}
1: $setT_{\mathrm{m}} \leftarrow$ **getTransactions**$(transactionPool, z)$
2: $validInfo \leftarrow$ **verifyInfo**$(Info_{\mathrm{m}})$
3: **if** $validInfo$ is **true then**
4: $T_{\mathrm{m}} \leftarrow$ **createTransaction**$(B, Info_{\mathrm{m}}, NPK_{\mathrm{i}})$
5: **for all** Gw_{i} in D_{j} **do**
6: **prepareConsensus**$(setT_{\mathrm{m}})$
7: **end for**
8: **for all** Gw_{i} in D_{j} **do**
9: $responseListperT \leftarrow$ **commitConsensus**$(setT_{\mathrm{m}})$
10: **end for**
11: **for all** T_{k} in $responseListperT$ **do**
12: **if** $|valid(responseList)| > minResponses$ **then**
13: **addTransaction**(T_{k}))
14: **end if**
15: **end for**
16: **end if**

a different number of contexts, different configurations and different approaches for updating the nodes. Also, we used the Core Emulator [21] to create a container-based network to emulate network equipment, gateways and devices. For all executed tests, a network with ten (10) gateways and 1,000 devices was adopted in order to emulate a smart building.

We present the description scenarios used in the evaluation in Table I, the configurations used in the context-based consensus in Table II and the approaches used to propagate the transactions after consensus in Table III. The emulation was performed in a Virtual Machine (VM) with 6-core processor, 16GB of memory and 64MB of graphics memory running Ubuntu 18.04 operating system using a Virtual Box hypervisor over a Macbook Pro with 2.3 GHz 8-Core Intel Core i9 processor, 32GB DDR4 memory.

TABLE I: Evaluated scenarios.

Scenario	Description
1	1,000,000 transactions sent by 1,000 devices, varying from 1 to 10 contexts, where all gateways participate in all contexts
2	1,000,000 transactions sent by 1,000 devices, varying from 1 to 10 contexts, each context having exactly 5 gateways (gateways can participate in more than one context)

TABLE II: Evaluated configurations.

Configuration	Description
A	All contexts using PBFT for a single transaction
B	All contexts using PBFT with no limit of transactions per consensus
C	All contexts using PBFT with limited number of transactions (100, 1000, and 10000) per consensus

TABLE III: Evaluated approaches.

Approach	Description
I	After the consensus, transactions are sent to all gateways that do not participate in the context
II	After the consensus, transactions are not sent to gateways that do not participate in the context

In *Scenario 1*, we intend to show how multiple contexts can perform when all gateways participate in all contexts. This is

to show the most demanding scenario due to high processing and communication demand. In *Scenario 2*, we intend to show the impact of limiting the number of contexts that a gateway can participate in on latency and throughput. Unlike *Scenario 1*, in this scenario it is not possible for a gateway to participate in all contexts as there are only five gateways in every context. Hence, it is possible to have some gateways that participate in multiple contexts.

Also, we evaluated the 3 configurations proposed in Section III and presented in Table II. These different configurations were evaluated to show how the number of transactions in each consensus affects the throughput and latency in context-based consensus. Finally, as presented in Table III, we used 2 different transaction update approaches: inserting transactions in all gateways that do not participate in the consensus, or not inserting them while they are not requested. These approaches were evaluated only for Scenario 2. Thus, we used only the approach I code for Scenario 1 since all gateways belong to all contexts, *i.e.*, they do not need any additional updates. Consequently, we executed 150 different tests, as a result of different combinations of scenarios, transaction limit configurations in each context, different transaction propagation approaches and a different number of contexts used in each test.

A. Results

We used two metrics to evaluate context-based consensus for transactions in appendable-block blockchains, *i.e.* **latency** and **throughput**.

1) Latency results: latency was calculated based on the time spent from creating a transaction to inserting it in the blockchain. Consequently, the latency captures the whole time spent in different processes such as the time it takes to propagate the transaction to gateways, the time the transaction spends in the transaction pool, and the time spent in the consensus. It is important to note that the evaluation was performed in a local network, where the communication times are reduced.

We can observe in Table IV the average (with the 95% confidence interval) transactions latency (in milliseconds) in all scenarios, approaches and configurations. Hence, lower latency results are better. Due to space limitation, we present only the results for 1, 2, 4 and 8 contexts. The first row indicates the scenario (1 or 2), the configuration (A, B and C, where C can take 100, 1,000 and 10,000 transactions) and update approach (I or II). For each scenario/configuration/approach, we collect results from 1 to 10 contexts (for one context, all devices in that context; for two contexts, half of the devices in each context; and so on).

We can observe that when using only one context (in all scenarios/configurations/approaches), the average latency is always higher than 10,000ms (10 seconds), indicating that using only a single context, *i.e.*, only one consensus for all transactions, is not sufficient to insert a transaction before a new one is produced by the same device (every 10 seconds). Additionally, considering two or more contexts, for almost all cases configurations/approaches, *scenario 1* presented worse results than evaluation over *scenario 2*. For *scenario 1*, the lowest transaction latency was 706.5±1.3ms using two contexts with *Configuration C* (with limit of 1,000 transactions). This value is more than 463% of the best result (**152.5±0.3ms**) in *scenario 2* (four contexts with *Configuration c* with limit of 1,000 transactions and update *approach II*). Consequently, the results show that the number of gateways in each context can impact the latency.

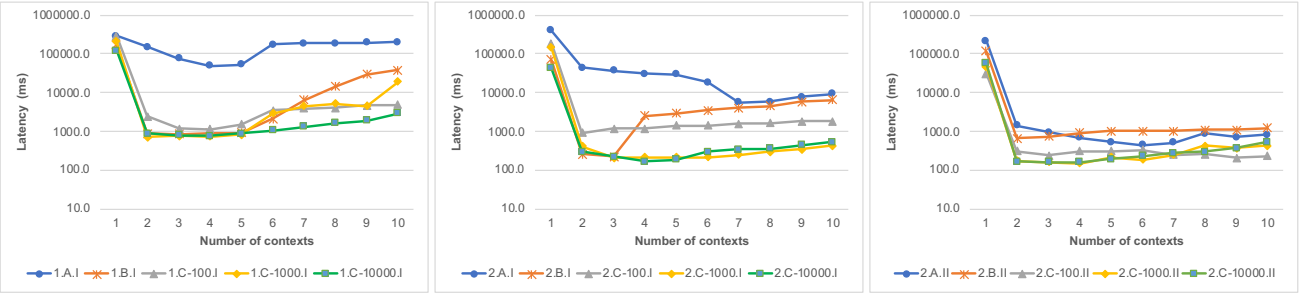
TABLE IV: Latency (in milliseconds) to insert transaction.

Scen.Conf.Appr.	Number of Contexts			
	1	2	4	8
1.A.I	284867.0±336.1	150477.9±577.2	48413.7±407.1	189562.5±731.9
1.B.I	168544.0±245.3	881.1±1.3	906.9±2.5	14328.5±32.1
1.C-100.I	287921.4±544.4	2402.2±5.4	1152.5±3.8	4106.3±8.3
1.C-1000.I	210587.2±359.4	706.5±1.3	734.2±2.2	5160.3±27.2
1.C-10000.I	122431.8±353.8	895.8±2.1	762±1.8	1661±3.9
2.A.I	402376.2±1294.3	43833.7±194.5	31170.9±217.8	5714.2±49.0
2.B.I	70143.3±226.3	256.7±0.6	2507.3±4.6	4455.8±18.1
2.C-100.I	189105.4±345.6	937.1±2.5	1186.9±2.8	1619.8±15
2.C-1000.I	145247.4±262.9	416.7±0.9	216.6±0.6	303.4±2.2
2.C-10000.I	42636.7±117.4	305.8±0.7	168.0±0.4	363.2±1.2
2.A.II	214997±804.3	1407.9±4.4	686.9±4.4	897.3±5.6
2.B.II	118258.2±484.4	670.0±2.0	924.6±2.3	1095.4±6.2
2.C-100.II	30212.2±170.7	309.3±0.8	306.1±2.8	261.7±0.7
2.C-1000.II	46555.1±275.6	173.3±0.6	152.5±0.3	430.2±1.5
2.C-10000.II	56736.4±89.3	169.5±0.4	164.6±0.4	302±0.8

In order to help to better understand the differences between transaction limit configurations and update approaches, we present the results separated in scenarios in Fig. 4 using logarithm values. We can observe that best results for two or more contexts are achieved by *Configuration c* with 1,000 and 10,000 transactions, represented respectively by yellow (with diamond) and green (with square) lines. In special for both evaluations over *scenario 2*, the average latency was under 1 second for two or more contexts using *Configuration c* with 1,000 and 10,000 transactions. Additionally, *approach II* had better general results than *approach I*. Also, it is important to note that *approach II* using *Configuration c* (100, 1,000 and 10,000) achieved an average latency lower than 550ms for 2 or more contexts. Thus, we can assume that a context with less gateways (*scenario 2*), with a configuration with a limit between 100 and 10,000 transactions, and updating by request can have a reduced latency.

2) Throughput results: in this evaluation, we considered as throughput the rate of insertion of transactions per second (tps) in the blockchain. As expected, when considering only one context, the throughput was considerably lower than with multiple contexts. Similar to what happened to latency, best results were obtained when less gateways per context were used (*scenario 2*). As a comparison, the best result was obtained when using three contexts, in *scenario 2*, with *Configuration c* with limit of 100 transactions and *approach II* (not updating), having a consensus throughput of **154.8±1.4tps**.

It is important to note that throughput is affected by many factors. The number of transactions is an important aspect, for one transaction in each consensus (*Configuration a*) means a consensus procedure that will be performed for just one transaction. Although, a high number of transactions in each consensus means more time to verify and perform consensus.



(a) Scenario 1 (b) Scenario 2, approach I (c) Scenario 2, approach II

Fig. 4: Average latency for each transaction in context-based consensus.

Additionally, increased load on gateways during parallel execution of consensus and more messages exchanged (during the consensus or receiving updates from different contexts) can affect the performance. Furthermore, it is important to note that a higher rate of transactions would affect the throughput. However, in all evaluation instances we used the same number of devices, gateways and transactions in order to have the same parameters for all 150 different performed tests.

TABLE V: Transactions throughput (transactions per second) using context-based consensus.

Scen.Conf.Appr.	Number of Contexts			
	1	2	4	8
1.A.I	30.264905±0.4	79.4±0.4	84.6±0.5	80.4±0.6
1.B.I	30.4805145±2.0	92.2±1.5	91.7±0.9	79.9±1.2
1.C-100.I	37.2878515±1.5	85.4±1.4	93±0.8	71.5±2.3
1.C-1000.I	29.6682895±2.1	92.4±1.3	89.7±1.1	81.4±0.9
1.C-10000.I	29.8±0.4	60.6±0.7	99.4±2.0	78.6±3.2
2.A.I	19.3±0.2	85.2±0.7	105.9±1.3	127.7±1.5
2.B.I	6.4±0.6	123.9±1.2	102.7±0.8	40.6±0.6
2.C-100.I	26±1.0	112.1±1.7	121.4±2.5	133.3±2.4
2.C-1000.I	8.1±0.7	122.6±1.1	108.6±0.8	99.5±0.6
2.C-10000.I	6.2±0.5	126.1±1.6	113.1±0.8	97±0.7
2.A.II	36.4±0.2	75.7±0.4	87.3±0.4	78.8±0.3
2.B.II	13.5±1.2	91.2±2.0	96.2±2.0	54.8±1.0
2.C-100.II	41.5±1.2	133.2±1.1	125.8±0.9	108±0.6
2.C-1000.II	30.7±0.9	134.9±1.1	124.7±0.9	114.9±0.8
2.C-10000.II	11.1±1.4	145.3±1.6	123.1±0.8	99±0.6

Fig. 5 shows the impact of the number of contexts, configurations and update approaches on the throughput. We can observe that increasing the number of contexts can improve throughput. This shows that parallelism of insertion using different contexts can help to improve appendable-block blockchains performance. Additionally, between 2 and 6 contexts in scenario 2 (for both approach I and II) and using configuration C (100, 1,000 and 10,000 transactions limit) the throughput is above 100 transactions per second.

V. Discussion and Threats to Validity

The evaluation presented in Section IV showed that the context-based approach can present improvements both over a single context (or non-existence of contexts) and to a single transaction insertion in the blockchain. Also, our evaluation shows that context-based consensus can guarantee lower latency and higher throughput. However, there are some threats to validity of our evaluation since the evaluation tests were performed in a controlled environment.

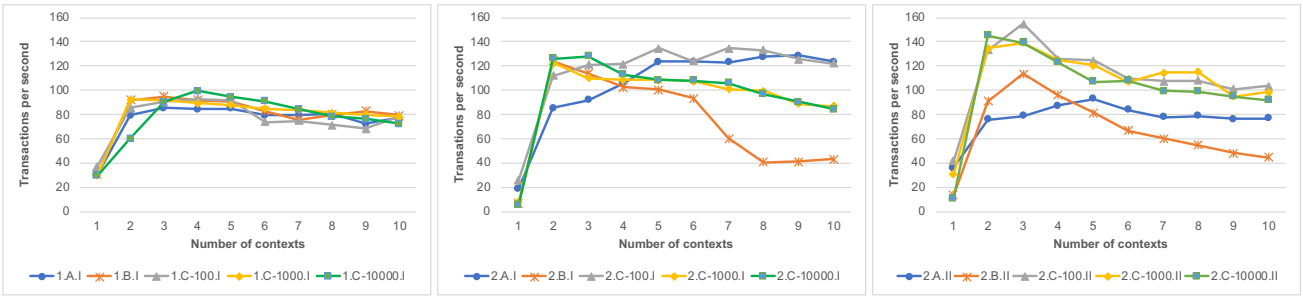
The first internal threat is the instrumentation used to perform the tests. Mainly, the hardware used to perform the evaluation can have an impact on the presented results as different hardware and network configurations lead to different results. However, the difference between scenarios, configurations and approaches is expected to be reproduced in any adopted hardware. We intend to consider different hardware in future work. The second internal threat is related to the selection of values used to set the scenarios. A different number of gateways and devices, as well, different rate of transactions per second can influence the obtained results. Moreover, smart contracts transaction on appendable-block blockchains (as proposed by Nunes *et al.* [22]) can lead to different execution results. In future work, we intend to evaluate different set of scenarios, *e.g.*, larger scenarios with higher transaction rate with smart contracts executions.

Also, context-based consensus can present some security issues, particularly due to the limited number of gateways controlling the consensus in each context. This issue is similar to the issues in the adoption of shards in blockchains [19]. Different from many shard approaches, all block headers are kept by all nodes in context-based consensus for appendable-block blockchains. This can reduce the impact of an attack, but further investigations will be discussed in future work.

VI. Final Considerations and Future Work

In this paper we propose and present context-based consensus, which supports consensus at the transaction level as well as allowing devices to connect to multiple gateways. By doing so, the context-based consensus can solve two existing issues in appendable-block blockchains, namely the eclipse attack performed by a single malicious gateway and the lack of transactions consensus.

To evaluate the performance of context-based consensus, we implemented a prototype. The evaluation shows that performance is good, achieving a total latency under 550ms and throughput above 100 transactions per second. These results are influenced by the number of gateways in each context, the number of transactions per consensus, and the



(a) Scenario 1 (b) Scenario 2, approach I (c) Scenario 2, approach II

Fig. 5: Average transactions consensus throughput (transactions per second inserted in the blockchain).

way appendable-block blockchain in each gateway is updated with transactions from other contexts. The best results (latency under 550ms, achieving average results lower as 152.5m) were obtained using multiple contexts with a limited number of gateways and a limited number of transactions per consensus round. Throughput of over 100 transactions per second relied on PBFT as consensus algorithm. We also showed that using multiple contexts to insert transactions leads to lower latency than a single or no context. The consensus of a single transaction per time (as in the previous versions of appendable-block blockchains) tends to result in prohibitively high latency.

As future work, we intend to scale our context-based consensus by increasing the number of gateways and devices, so that it can be used in different IoT environments. Further discussion should be performed considering different consensus algorithms as well, detailed analysis and impact of different attacks, such as Sybil and 1% attacks.

REFERENCES

[1] L. Lu, J. Chen, Z. Tian, Q. He, B. Huang, Y. Xiang, and Z. Liu, "Educoin: a secure and efficient payment solution for mooc environment," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 490–495.

[2] H. Guo, W. Li, M. Nejad, and C. Shen, "Access control for electronic health records with hybrid blockchain-edge architecture," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 44–51.

[3] V. Dedeoglu, A. Dorri, R. Jurdak, R. A. Michelin, R. C. Lunardi, S. S. Kanhere, and A. F. Zorzo, "A journey in applying blockchain for cyberphysical systems," in *International Conference on COMmunication Systems NETworkS (COMSNETS)*, 2020, pp. 383–390.

[4] S. Latifi, Y. Zhang, and L. Cheng, "Blockchain-based real estate market: One method for applying blockchain technology in commercial real estate market," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 528–535.

[5] S. Malik, V. Dedeoglu, S. S. Kanhere, and R. Jurdak, "Trustchain: Trust management in blockchain and iot supported supply chains," in *IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 184–193.

[6] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2017, pp. 618–623.

[7] M. Samaniego and R. Deters, "Blockchain as a service for IoT," in *IEEE International Conference on Internet of Things (iThings)/IEEE Green Computing and Communications (GreenCom)/IEEE Cyber, Physical and Social Computing (CPSCom)/IEEE Smart Data (SmartData)*, 2016, pp. 433–436.

[8] R. C. Lunardi, R. A. Michelin, C. V. Neu, and A. F. Zorzo, "Distributed access control on IoT ledger-based architecture," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2018, pp. 1–7.

[9] I. Foundation, "IOTA - Next Generation Blockchain," Feb. 2020. [Online]. Available: https://iota.org/

[10] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[11] R. A. Michelin, A. Dorri, M. Steger, R. C. Lunardi, S. S. Kanhere, R. Jurdak, and A. F. Zorzo, "Speedychain: A framework for decoupling data from blockchain for smart cities," in *15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, 2018, pp. 145–154.

[12] R. C. Lunardi, R. A. Michelin, C. V. Neu, H. C. Nunes, A. F. Zorzo, and S. S. Kanhere, "Impact of Consensus on Appendable-Block Blockchain for IoT," in *16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, 2019, pp. 228–237.

[13] W. Tang, J. Ren, K. Deng, and Y. Zhang, "Secure data aggregation of lightweight e-healthcare iot devices with fair incentives," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8714–8726, 2019.

[14] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, 2011, pp. 4490–4494.

[15] T. Mahjabin, Y. Xiao, T. Li, and C. L. P. Chen, "Load distributed and benign-bot mitigation methods for iot dns flood attacks," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 986–1000, 2020.

[16] S. Solat, "Rdv: An alternative to proof-of-work and a real decentralized consensus for blockchain," in *1st Workshop on Blockchain-enabled Networked Sensor Systems*. ACM, 2018, pp. 25–31.

[17] X. Fan and Q. Chai, "Roll-dpos: A randomized delegated proof of stake scheme for scalable blockchain-based internet of things systems," in *15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, 2018, pp. 482–484.

[18] T. Crain, V. Gramoli, M. Larrea, and M. Raynal, "(Leader/Randomization/Signature)-free Byzantine Consensus for Consortium Blockchains," *CoRR*, vol. abs/1702.03068, 2017. [Online]. Available: http://arxiv.org/abs/1702.03068

[19] A. Hafid, A. S. Hafid, and M. Samih, "New mathematical model to analyze security of sharding-based blockchain protocols," *IEEE Access*, vol. 7, pp. 185 447–185 457, 2019.

[20] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *1999 Third Symposium on Operating Systems Design and Implementation (OSDI)*. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186.

[21] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time network emulator," in *27th IEEE Military Communications Conference (MILCOM 2008)*, 2008, pp. 1–7.

[22] H. C. Nunes, R. C. Lunardi, A. F. Zorzo, R. A. Michelin, and S. S. Kanhere, "Context-based smart contracts for appendable-block blockchains," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–9.