BRUSCHETTA: An IoT Blockchain-Based Framework for Certifying Extra Virgin Olive Oil Supply Chain

Antonio Arena
Information Engineering dept.
University of Pisa
Pisa, Italy
antonio.arena@ing.unipi.it

Alessio Bianchini
GEOSTECH s.r.l.
Livorno, Italy
bianchini@geostech.it

Pericle Perazzo
Information Engineering dept.
University of Pisa
Pisa, Italy
pericle.perazzo@iet.unipi.it

Carlo Vallati
Information Engineering dept.
University of Pisa
Pisa, Italy
carlo.vallati@iet.unipi.it

Gianluca Dini
Information Engineering dept.
University of Pisa
Pisa, Italy
gianluca.dini@iet.unipi.it

***Abstract*—Urban population is expected to continuously grow in size. The smart city concepts allows to handle the new challenges and issues created by this growth by applying a wide range of technologies that can provide citizens with a better living environment. Smart agriculture will play an important part of smart cities, as a sustainable and high quality food supply chain is crucial to facilitate the grow of human agglomerates. In this context, European laws imposes very strict requirements in the food industry, in order to ensure that food provenance is always guaranteed. Such fine-grained traceability can be only achieved by applying state-of-the-art technologies. In this paper, we present BRUSCHETTA, a blockchain-based application for the traceability and the certification of the Extra Virgin Olive Oil (EVOO) supply chain. EVOO is an emblematic food product for Italy, but it is also one of the most falsified ones. BRUSCHETTA provides a blockchain-based system to enforce the certification of this product by tracing its entire supply chain: from the plantation to the shops. The goal is to enable the final customer to access a tamper-proof history of the product, including the farming, harvesting, production, packaging, conservation, and transportation processes. BRUSCHETTA leverages Internet of Things (IoT) technologies in order to interconnect sensors dedicated to EVOO quality control, and to let them operate on the blockchain. We also provide a support for the correct tailoring of the BRUSCHETTA blockchain system, and we propose a mechanism for its dynamic auto-tuning to optimize it in case of high loads.**

***Index Terms*—Smart Cities, Smart Agricolture, Blockchain, Hyperledger Fabric, Food supply chain monitoring, Performance evaluation**

I. Introduction

Due to the rapid growth of the population density in urban cities, much higher requirements are set for municipality governors to manage all aspects in urban living. Recent technologies developments are offering the possibility to revolutionizing many aspects of the cities by making them *smarter* [1]. The adoption of technologies, such as the Internet of Things (IoT), is expected to play a crucial part in improving city functions to ensure, on one side, a sustainable growth, on the other, to improve citizens' living conditions [2].

Smart agriculture plays an important part in smart cities, to create a sustainable and high quality food supply chain [3]. State-of-the-art technologies are already improving the efficiency of the food production process. In order to ensure that quality standards are satisfied, European laws imposes very strict requirements in the food industry, which can be met only by massively employing novel technologies [4]. Fine-grained traceability, in particular, is imposed, in order to ensure that complete provenance can always be certified.

In this paper, we present *BRUSCHETTA*, a blockchain-based application for the traceability and the certification of *Extra Virgin Olive Oil* (EVOO). EVOO is an emblematic product for Italy, known and appreciated in the entire world. Unfortunately, it is also one of the most falsified food products [5]. BRUSCHETTA provides a blockchain-based system to enforce the certification of this product by tracing the entire process of production: from the plantation to the shops. Its goal is to enable a final user (the one that buys the product) to access a tamper-proof copy of the entire history of the product, that is the farming process, harvesting, production, packaging, conservation, and transportation, for example through its own smartphone. BRUSCHETTA leverages IoT technologies in order to interconnect sensors dedicated to EVOO quality control, and to let them operate on the blockchain. We also provide a support for the correct tailoring of the BRUSCHETTA blockchain system, and we propose a mechanism for dynamic auto-tuning of parameters that can maintain our system suitable for an industrial scenario, even in case of high network loads.

978-1-7281-1689-1/19/$31.00 ©2019 IEEE
DOI 10.1109/SMARTCOMP.2019.00049

The rest of the paper is organized as follows. Section II introduces the main technological aspects of the blockchain technology. Section III introduces the BRUSCHETTA system model and our threat model. Section IV presents and discusses the results of our performance evaluation of BRUSCHETTA. Section V presents the proper tailoring of the BRUSCHETTA blockchain system and the dynamic auto-tuning mechanism. Finally, Section VI concludes the paper.

II. Blockchain

A *blockchain* can be defined as an immutable ledger for recording *transactions*, maintained within a distributed network of mutually untrusted *nodes*. A blockchain is a list of ordered blocks, where each block stores a variable-size list of transactions. Nodes can generate and read transactions and/or participate in the *consensus protocol*, which allows nodes to agree on which transactions compose a block and in which order. The nodes that participate in the consensus protocol are called *peers*. The peers execute a consensus protocol to validate transactions, group them into blocks that include a hash value that bind each block to the preceding block. Different consensus protocols are possible, with different security properties.

The first and most widely recognized application of blockchain is the *Bitcoin* cryptocurrency [6], which allows the nodes to enable digital money transfers on an untrusted network of nodes without the financial brokering of a trusted third party, such as a central bank. This application, like other financial applications, use a class of blockchains called *public* or *permissionless*. In a public blockchain virtually anyone can participate, and every participant is anonymous. Public blockchains typically involve a native cryptocurrency and often use a resource-demanding consensus protocol, such as the *Proof of Work* (PoW) protocol, and economic incentives for peers to participate in such protocol.

In addition to cryptocurrency and financial applications, the blockchain technology is promising in several other scenarios like smart homes [7], smart grids [8], healthcare [9], smart cities [10], and so on. However, many of these scenarios require performance characteristics that the permissionless blockchain technologies are unable (presently) to deliver, such as transaction low latency and high throughput. In addition, in many scenarios, knowing the identity of the participants is a hard requirement, such as in the case of financial transactions where Know-Your-Customer (KYC) and Anti-Money Laundering (AML) regulations must be followed. For this purpose, *permissioned* blockchains have been introduced. Permissioned blockchains run among a set of known, identified participants. By relying on the identities of the peers, a permissioned blockchain can use traditional Byzantine-fault tolerant (BFT) consensus protocols [11], [12], which do not require many resources.

III. BRUSCHETTA System Model

The EVOO supply chain involves four different parties: (i) the *farmers*, which are responsible for olives farming and harvesting; (ii) the *makers*, which are responsible for the transformation process from olives to EVOO, and for the packaging process; (iii) the *couriers*, which are responsible for olives and EVOO transportation; (iv) the *sellers*, which are the final destination of the production process and are responsible for EVOO distribution to final users.

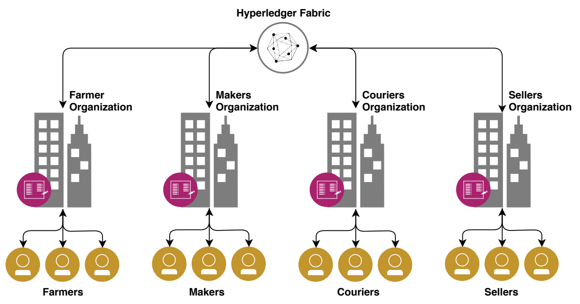

Fig. 1. BRUSCHETTA architecture

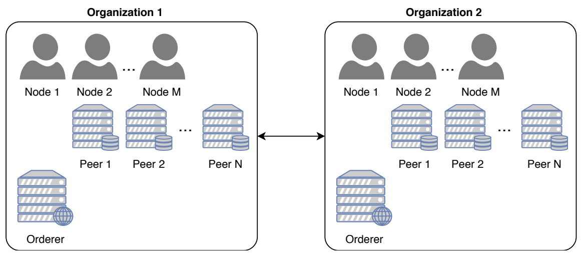

Fig. 2. Hyperledger Fabric model

In this section we present BRUSCHETTA, a blockchain-based application for the traceability and the certification of EVOO. The goal of BRUSCHETTA is to monitor every production process in order to obtain critical information for estimating the quality of the final product. We made BRUSCHETTA publicly available on Github[1]. The BRUSCHETTA architecture is illustrated in Fig. 1. As it can be seen from the figure, BRUSCHETTA uses *Hyperledger Fabric* as blockchain technology [13].

Hyperledger Fabric, or simply *Fabric*, is a popular, open-source, permissioned blockchain. The key points of Fabric is that it is highly modular and does not require a native cryptocurrency to incentivise the consensus protocol executions. Fabric natively supports *smart contracts*, which can be defined as trusted distributed applications that gain their security from the blockchain and the underlying consensus among the peers. Fig. 2 presents the Hyperledger Fabric model. Fabric users are grouped into two or more *organizations*. In every organization we have three types of users depending on their role on the blockchain, namely *nodes*, *endorsing peers* and *orderer nodes*. Nodes are users that can only generate new transactions and read the ledger history. Endorsing peers are users that are responsible for verifying that a transaction follows all the *endorsing policies* linked to the node that generated the

[1]BRUSCHETTA available at: http://bit.ly/bruschetta_unipi

transaction. In particular, an endorsing policy is a rule that defines the necessary and sufficient conditions for considering valid a transaction. Finally, orderer nodes are users that are responsible for ordering transactions, grouping them into new blocks, and executing the consensus protocol.

Following the above mentioned conventions of Hyperledger Fabric, users in BRUSCHETTA are grouped in four different organizations, which respectively represent the four different parties involved in the EVOO supply chain, as shown in Fig. 1. Every organization owns several peer nodes and orderer nodes in order to implement all the basic functions of Fabric. BRUSCHETTA is enriched with several endorsing policies so that: (i) a node belonging to an organization can not play the role of a node belonging to another organization; (ii) a transaction recording a product transfer between two nodes belonging to different organizations is included in Hyperledger Fabric when both parties agree on the quality of the product transferred.

A. BRUSCHETTA Processes

BRUSCHETTA divides the EVOO supply chain in six processes: (i) *olives farming process*, (ii) *olives harvesting process*, (iii) *olives transport process*, (iv) *olives transformation process*, (v) *packaging process* and (vi) *oil transport process*. We designed different *profiles* for every process that group critical information for the relevant process. Specifically, every profile represents one or more transactions that are generated by nodes and included in the Hyperledger Fabric blockchain, so that the entire history of every process can be reconstructed when it is required.

The main factors that can affect the EVOO quality during the olives farming process are categorized in: (i) the weather conditions, i.e., temperature, humidity and air pollution, (ii) the chemical treatments on plantations of olive trees, and (iii) the chemical composition of the fields. All this factors are collected during the farming process by a set of sensors deployed in every plantation, which periodically include data in a *olives farming profile*.

During the olives harvesting process, instead, the critical aspects that affect the EVOO quality are the time period in which the olives are harvested and how long the harvested olives are stored since the transformation process starts. Moreover, the harvesting method, i.e., if they are harvested by hand or by using specific machines, sensibly affects the acidity of olives and consequently the final EVOO quality [14]. In the harvesting process data are grouped and included in a *olives harvesting profile* by farmers.

The olives transport process, i.e., the olives transportation from the farmers to the makers is a very critical process and must be carefully monitored. In particular, the olives temperature during the transport assumes is a critical parameter, and it must be periodically monitored and included in a *olives transport profile* by temperature sensors deployed on couriers' vans.

The transformation process, i.e., the process in which EVOO is made starting from olives, represents the most critical one in the EVOO supply chain. The transformation process is divided into six tasks, namely, *de-leafing*, *washing*, *crushing*, *malaxing*, *decanting* and *separation*. Temperature represents a key parameter in these tasks. All the temperature values are collected during the transformation process by a set of temperature sensors deployed on the makers' factories and included in a *olives transformation profile*.

The packaging process consists of bottling EVOO. It is important to keep the temperature constant during this process in order to preserve the characteristic of EVOO, such as colour and scent. The temperature values collected by temperatures sensors during the packaging process are included in a *packaging profile*.

The temperatures still remains a key parameter in the oil transport process for preserving EVOO characteristics. In particular, during this process the temperature should not have large variations. For this reason, the temperature values are collected by temperature sensors deployed in couriers' vans and included in an *oil transport profile*.

Finally, sellers and end-users can easily retrieve the entire supply chain of a single bottle of EVOO, by using a web application[2] which reads the blockchain transactions and reconstructs the entire history of the EVOO.

B. Threat Model

As we have just said, the EVOO production process involves multiple industrial parties with, typically, conflicting objectives. For example, a party may be interested in claiming false statements about its production process to take economic advantages at the cost of poor quality of the final product. In our scenario, we can identify a set of internal and external threats.

From the inside, a node of the system may be interested in degrading the reputation of another node by assuming the victim identity and starting to transmit data that nominally degrade the quality of the product. For example, an adversary farmer A tries to transmit malicious olive farming profiles for a plantation of farmer B. However, BRUSCHETTA avoid this behaviour by using a set of implemented endorsing policies. Another adversarial scenario is related to a malicious entity which is interested in changing its own profiles or the profiles of another node. Both situations are avoided by the immutability of Hyperledger Fabric blockchain.

From the outside, an external adversary is interested in compromising the correctness of the system, for example by deleting, tampering or stealing data. Data tampering and deleting are easily avoided by enforcing immutability. Data stealing is instead avoided by securing data with encryption, or by enforcing endorsing policies for implementing reading permissions for the Hyperledger Fabric transactions.

IV. Performance Evaluation

In order to test the proposed solution a performance evaluation based on simulations is carried out. The analysis, in

[2]Web application available at http://bit.ly/bruschetta_webapp

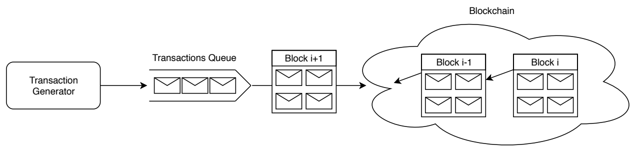

Fig. 3. BRUSCHETTA simulation model on OMNet++

TABLE I
SIMULATION PARAMETERS

Max transactions per block N	$[10, 20, 50, 100, 200]$
Transaction rate λ	$[1, 0.1, 0.01, 0.005]$ txs/s
Consensus algorithm execution time et	$[25, 50, 200, 500]$ ms
Maximum block generation time T	30 s

particular, evaluates the performance of BRUSCHETTA when different settings of the Hyperledger Fabric are considered, in order to find the proper set of values. The focus is on measuring the time required to store on the blockchain a new value in order to ensure that the new data is published with a delay that makes a real deployment of BRUSCHETTA feasible.

To this aim, a model of the BRUSCHETTA system is implemented in OMNeT++[3], a popular event-based simulator written in C++ and freely available. The simulator has been adopted for its modular design that allows a rapid and simple definition of novel simulation scenarios leveraging a standard set of existing modules. The BRUSCHETTA system is modelled in OMNeT++ as shown in Fig. 3. The Hyperledger Fabric is modeled as a M/M/1 queuing system with bulk departures, i.e. transactions are dequeued in groups. Each group represents a new block in the blockchain and can have a variable number of transactions up to N (*max transactions per block*). A new block is generated whenever there are sufficient enqueued transactions to fill a block, at least N, or when a timer expires, which ensures that two blocks are generated within a maximum block generation time T. In order to model the time required for the consensus algorithm to converge, a fixed execution time et is introduced to delay the publication of the new block on the blockchain after its generation. Transactions are generated by a generator module following a Poisson process characterized by a *transaction generation rate* λ.

The parameters and the corresponding values considered in our simulations are summarized in Table I. Different values of transactions per block, transaction generation rate and execution time are considered. In order to measure the performance the *transaction delay* is collected. The metric is defined as the time between the generation of a transaction and its publication on the blockchain. The transaction delay is considered to measure the time required for the transaction to be validated and published, thus providing an indication whether BRUSCHETTA is sufficiently reactive to handle the transactions in a practical implementation or not. In order to obtain statistically sound results, every simulation scenario is replicated 30 times with independent random seeds. In the following results the average transaction delay value across different replicas are shown along with the corresponding 95-th confidence intervals.

[3]OMNeT++ website: https://omnetpp.org/

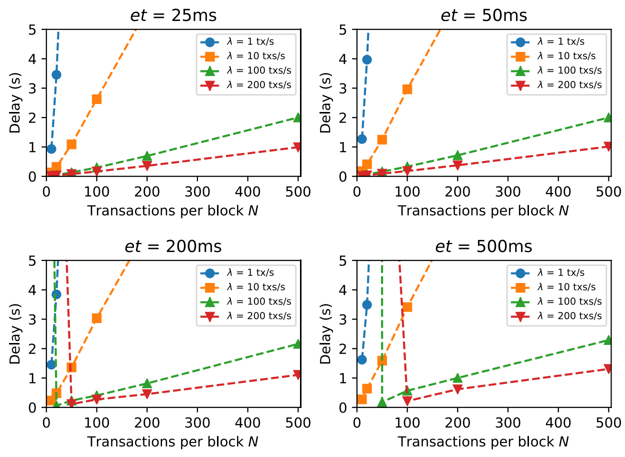

Fig. 4. Transaction delay vs transactions per block with different values of execution time

Fig. 4 reports the transaction delay obtained with an increasing number of transactions per block, considering different values of the consensus algorithm execution time. For the sake of readability the graphs show the delay up to 5s. In order to provide full detail on the results, a complete plot of the delay is reported in Fig. 5 and Fig. 6 for two specific configuration, i.e. $et = 50ms$ and $et = 200ms$, respectively.

Let us analyze first the results obtained with small execution times, i.e. $et = 25ms$ and $et = 50ms$. When a sustained transaction generation rate, i.e. $\lambda = 100txs/s$ and $\lambda = 200txs/s$, is considered, the resulting delay is significantly below T, meaning that the generation of a new block is always triggered by a sufficient number of transactions enqueued.

As expected, in these scenarios the transaction delay increases as N increases or as λ decreases. This can be explained considering that a lower λ or a larger N result in a higher queue waiting time experienced by each transaction. A lower λ results in the generation of a lower number of transactions that, in turn, results in a lower number of transactions enqueued thus requiring more time to trigger the generation of a block. Likewise larger blocks require more time to reach the number of transactions necessary to fill the block.

This behaviour is exacerbated with $\lambda = 1txs/s$ and $\lambda = 10txs/s$, which results in a delay value that rapidly increases as N increases, reaching a delay larger than 5s with $N > 50$ and $N > 200$, respectively. In order to offer a complete view of the results in Fig. 5 we report the results obtained with $et = 50ms$ with a y-scale that goes up to 20s. As can be seen the delay grows until a maximum delay of 15s is reached, meaning that a new block is generated every 30s when the

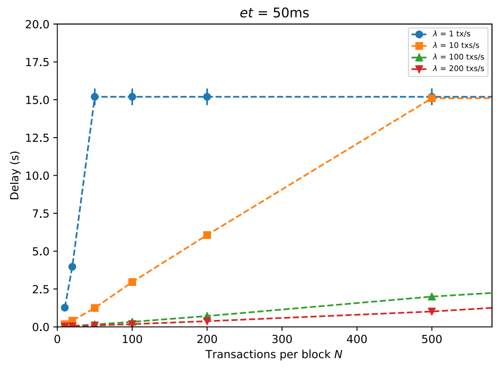

Fig. 5. Transaction delay vs transactions per block with $et = 50ms$

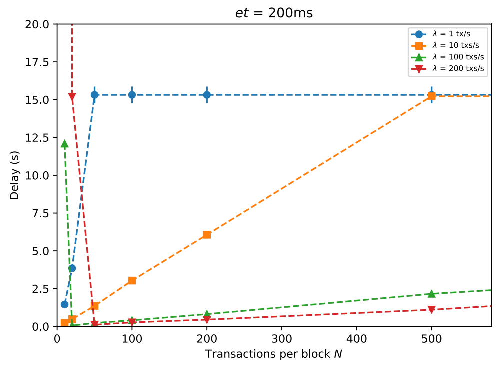

Fig. 6. Transaction delay vs transactions per block with $et = 200ms$

timer expires, as the transaction generation rate is not sufficient to trigger the generation of new blocks.

A similar behaviour is observed with larger execution times, i.e. $et = 200ms$ and $et = 500ms$, and low transaction generation rates, i.e. $\lambda = 1tx/s$ and $\lambda = 10txs/s$: the transaction delay increases up to 15s the maximum value. A different behaviour is, instead, observed when larger execution times, i.e. $et = 200ms$ and $et = 500ms$, and high generation rates are considered, i.e. $\lambda = 100txs/s$ and $\lambda = 200txs/s$. Specifically, the transaction delay starts from a very high value, decreases up to a minimum, and then starts to increase again. In order to obtain an insight on this behaviour in Fig. 6 we report the delay with a scale up to 20s. As can be seen the delay obtained with a very small N, i.e. below 20, is even higher than 15s. This very high value is due to the fact that the execution time of the consensus algorithm becomes a bottleneck with these configurations. With small blocks and a high transaction generation rate, blocks can be created continuously, in other words a block is generated after another. The system cannot register the transactions with the same rate they arrive, consequently they are enqueued for a long time. As N increases, the delay reduces as less blocks are created, until a minimum delay is reached. After this minimum the delay starts increasing again, due to the fact that transactions have to wait longer in order to reach the minimum number of transactions required to trigger the creation of a block.

TABLE II
SIMULATION PARAMETERS WITH NETWORK LOAD VARIATIONS

Max transactions per block N	10
Maximum block generation time T	30 s
Consensus algorithm execution time et	200 ms
Transaction generation rates λ	$[10, 40, 100]$ txs/s
Transaction generation rates changing time t	$[2500, 4000]$ s

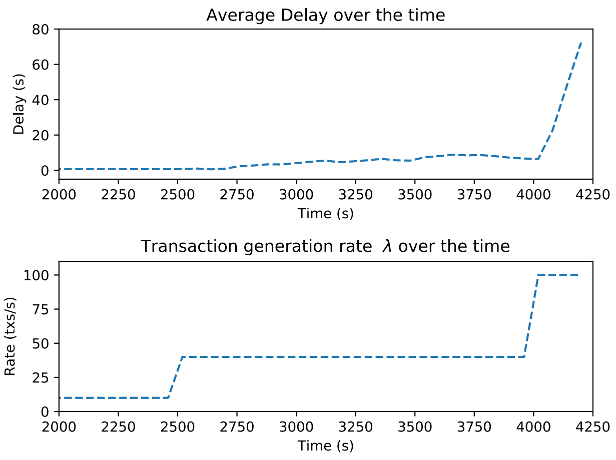

Fig. 7. Average transaction delay with variable mean transaction inter-arrival times

V. Dynamic Auto-Tuning Mechanism

In a business scenario, transaction delay represents a critical performance index that may affect the entire supply chain. Moreover, in a realistic situation, the transaction generation rate may vary over the time and in an unpredictable way. This is particularly true when IoT devices are involved in generating new transactions. For this reason, a static configuration of the Fabric blockchain may be unsuitable due to the high variability of the incoming transactions.

In order to show the problems stemming from a static configuration, we carried out some additional simulation experiments of the BRUSCHETTA system with OMNeT++. The parameter and the corresponding values considered in our simulation are summarized in Table II. In particular, we took in consideration three different values of transaction generation rates during every simulation experiment. We started from a low transaction rate $\lambda = 10txs/s$. Then, we switched at time $t = 2500s$ to a higher transaction generation rate $\lambda = 40txs/s$. Finally, we increased λ up to $100txs/s$ at time $t = 4000s$. Fig. 7 shows the average transaction delay during the simulation experiments. As expected, the average transaction delay remains stable with a high transaction generation rate, i.e., $\lambda = 10txs/s$. When the transaction generation rate increases to $\lambda = 40txs/s$, the transaction delay slightly increases, reaching a stable value equal to $8s$, meaning that the blockchain is still able to handle the incoming transactions. However, when the

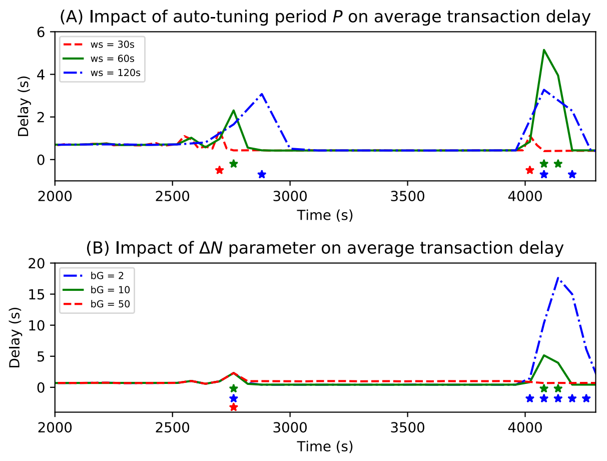

Fig. 8. (A) Transaction delay with different auto-tuning periods P and (B) Transaction delay with ΔN values. The starred points are the instants in which the dynamic mechanism changes the number of max transactions per block N.

transaction generation rate increases further to $\lambda = 100 txs/s$, the transaction delay dramatically increases, from $8s$ to almost $70s$, in a time interval of $4s$.

In order to mitigate this problem, we implemented a dynamic mechanism that periodically monitors the network state and automatically tunes the blockchain parameters. In particular, our dynamic auto-tuning mechanism is periodically executed with a fixed *auto-tuning period* P. It computes the average transaction delay in the last P seconds and compares it with the average values of the past average transaction delays. If the actual average transaction delay value is higher than the average of the past average transaction delays, the mechanism increases of a fixed value ΔN the number of max transactions per block N, so that a larger number of transactions can be handled at the same time. Otherwise, if the actual average transaction delay value is lower than the average of the past average transaction delays, the mechanism decreases of a fixed value ΔN the number of transactions per block.

The dynamic mechanism has been evaluated in OMNeT++. We took into consideration two different scenarios in our simulation experiments. In both scenarios, the transaction generation rate is increased following the same pattern reported in Fig. 7. In the first scenario, we evaluated the mechanism with different auto-tuning periods P and a fixed value ΔN equal to 10.

Fig. 8A shows the transaction delay with different auto-tuning periods P. As expected, with a low auto-tuning period, i.e., $P = 30s$, the mechanism quickly reacts to varying network conditions, keeping the transaction delay quite low. With higher auto-tuning period values, i.e., $P = 60s$ and $P = 120s$, the mechanism reacts slowly to the transaction generation rate increases. In particular, the maximum transactions per block N increases twice in order to bring the transaction delay back to reasonable values.

In the second scenario, we evaluated the mechanism with different ΔN values, keeping the auto-tuning period P fixed and equal to $60s$. Fig. 8B shows the transaction delay with different ΔN values. With a very low value of ΔN, i.e., $\Delta N = 2$, the blockchain suffers of frequent changes of the maximum transaction per block N when a burst of transactions arrives, which may be unfeasible if the transaction generation rate becomes even higher than in the considered scenario. If case of a high value of ΔN, i.e., $\Delta N = 50$, the blockchain does not suffer of transaction delays when the transaction generation rate increases. This can be explained by considering that the queue size becomes very large when N changes at time $t = 2500s$, so that the queue can easily handle a burst of incoming transaction. However, if the max transactions per block are too high, we could experience high transaction delays with low transaction generation rates. This is because the block generation is triggered only by the maximum block generation time T.

We than propose to adopt the values $P = 60s$ and $\Delta N = 10$ as standard parameters for our dynamic auto-tuning mechanism, as we obtained moderated transaction delay changes and a low number of transaction per block N changes.

VI. Conclusions

In this paper, we presented BRUSCHETTA, a blockchain-based application for the traceability and the certification of Extra Virgin Olive Oil (EVOO). BRUSCHETTA provides a blockchain-based system to enforce the certification of EVOO by tracing the entire process of production: from the plantation to the shops. The proposed BRUSCHETTA architecture allows to collect and certify data from all the different production phases, provided by users or sensors. The adoption of blockchain allows final users to access a tamper-proof copy of the entire history of the product for example from their smartphone. In order to show that the proposed system can be adopted in real industrial scenarios, a performance evaluation based on simulations was carried out. Results showed that the proposed approach can not always be suitable in real industrial scenarios, where the transactions arrival rate may vary over the time in an unpredictable way. For this reason, we proposed and evaluated a mechanism for dynamic auto-tuning of blockchain parameters in order to ensure that the information is published in the blockchain in a timed manner.

Acknowledgements

This work was partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence) and by the project PRA_2018_81 ”Wearable sensor systems: personalized analysis and data security in healthcare” funded by the University of Pisa.

References

[1] T. Heo, K. Kim, H. Kim, C. Lee, J. H. Ryu, Y. T. Leem, J. A. Jun, C. Pyo, S.-M. Yoo, and J. Ko, “Escaping from ancient rome! applications and challenges for designing smart cities,” *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 109–119, 2014.

[2] Q. Zhang, T. Huang, Y. Zhu, and M. Qiu, "A case study of sensor data collection and analysis in smart city: Provenance in smart food supply chain," *International Journal of Distributed Sensor Networks*, vol. 9, no. 11, p. 382132, 2013.

[3] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, p. 32, 2017.

[4] F. Schwägele, "Traceability from a european perspective," *Meat science*, vol. 71, no. 1, pp. 164–173, 2005.

[5] D. L. Garcia-Gonzalez and R. Aparicio, "Research in olive oil: challenges for the near future," *Journal of agricultural and food chemistry*, vol. 58, no. 24, pp. 12 569–12 577, 2010.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

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

[8] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: towards sustainable local energy markets," *Computer Science-Research and Development*, vol. 33, no. 1-2, pp. 207–214, 2018.

[9] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE, 2016, pp. 1–3.

[10] K. Biswas and V. Muthukkumarasamy, "Securing smart cities using blockchain technology," in *2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd international conference on data science and systems (HPCC/SmartCity/DSS)*. IEEE, 2016, pp. 1392–1393.

[11] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," *arXiv preprint arXiv:1707.01873*, 2017.

[12] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[13] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, p. 30.

[14] C. Saglam, Y. Tuna, U. Gecgel, and E. Atar, "Effects of olive harvesting methods on oil quality," *APCBEE procedia*, vol. 8, pp. 334–342, 2014.