A Network Coding Based Information Spreading Approach for Permissioned Blockchain in IoT Settings

Mumin Cebe
Department of ECE
Florida International University
Miami, Florida 33174
mcebe@fiu.edu

Berkay Kaplan
Dept. of Computer Science and Eng.
The Ohio State University
Columbus, Ohio 43210
kaplan.298@buckeyemail.osu.edu

Kemal Akkaya
Department of ECE
Florida International University
Miami, Florida 33174
kakkaya@fiu.edu

ABSTRACT

Permissioned Blockchain (PBC) has become a prevalent data structure to ensure that the records are immutable and secure. However, PBC still has significant challenges before it can be realized in different applications. One of such challenges is the overhead of the communication which is required to execute the Byzantine Agreement (BA) protocol that is needed for consensus building. As such, it may not be feasible to implement PBC for resource constrained environments such as Internet-of-Things (IoT). In this paper, we assess the communication overhead of running BA in an IoT environment that consists of wireless nodes (e.g., Raspberry PIs) with meshing capabilities. As the the packet loss ratio is significant and makes BA unfeasible to scale, we propose a network coding based approach that will reduce the packet overhead and minimize the consensus completion time of the BA. Specifically, various network coding approaches are designed as a replacement to TCP protocol which relies on unicasting and acknowledgements. The evaluation on a network of Raspberry PIs demonstrates that our approach can significantly improve scalability making BA feasible for medium size IoT networks.

CCS CONCEPTS

• **Networks** → **Network performance analysis**; • **Security and privacy** → *Information-theoretic techniques*;

KEYWORDS

Blockchain, Permissioned Blockchain, Information Spreading, Communication Complexity, Internet of Things (IoT)

ACM Reference Format:
Mumin Cebe, Berkay Kaplan, and Kemal Akkaya. 2018. A Network Coding Based Information Spreading Approach for Permissioned Blockchain in IoT Settings. In *EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '18), November 5–7, 2018, New York, NY, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3286978.3286984

MobiQuitous '18, November 5–7, 2018, New York, NY, USA

ACM ISBN 978-1-4503-6093-7/18/11…$15.00
https://doi.org/10.1145/3286978.3286984

1 INTRODUCTION

Since its application to cryptocurrencies, blockchain became a disruptive technology that started to be used in many different domains such as healthcare, energy, logistics, and forensics [5, 6, 8, 10]. Depending on the application domain, structure of blockchain and its components in terms of transaction verification and consensus are re-designed for such cases to fit the requirements and needs. For instance, structure-wise, there can be private (permissioned) or public blockchains. In the case of former, only authorized nodes can join a blockchain and they approve transactions by a special mechanism called BA which is different than the traditional hash-based computations that are used in public blockchains.

Blockchain also enables distributed trust and eliminates the need for a centralized authority for certain operations. Such feature makes blockchain a good candidate to be employed for IoT where resource constrained devices collect and exchange data for various purposes. For instance, in energy systems, smart meters can form a distributed mesh network and run certain application related algorithms (e.g., demand response, key management, firmware updates, etc.) by exchanging messages among themselves. Similarly, IoT devices in a smart home or within an autonomous vehicle may exchange data which may be stored in a distributed ledger for integrity.

Such potential of blockchain for IoT has been recognized by many stakeholders such as IBM [1] and researchers that started to trigger a few studies in this emerging domain [9]. For instance, the work in [15] proposes to store sensors data on a blockchain. Another study [2] is focusing to store identities of IoT devices on blockchain and then the authentication between interacting parties is accomplished through blockchain. As can be seen, all these studies use blockchain as a service to store some IoT related data. Construction of a decentralized autonomous blockchain within IoT devices is very rare. The most related study that constructs an autonomous blockchain is [7]. In this study, the authors propose to build a local blockhain for smart home to controlling and auditing the data.

Despite the potential benefits of IoT and blockchain integration, adapting blockhain to IoT is not straightforward and carries significant overheads such as, *high computation* to solve proof-of-work, and *long latency* to reach a consensus. These overheads may distress IoT devices in terms of energy and bandwidth requirements. For instance, BA protocol requires information exchanges among all the nodes to reach a consensus. Requiring communication among every node may lead to flooding which eventually makes scalability impossible to achieve given that the communication needs to be reliable too (i.e., re-transmissions will be inevitable).

To address this issue, in this paper, we offer a network coding based approach that can reduce transmissions that are otherwise excessive when a reliable unicast protocol such as TCP is used. Network coding enables fast relaying of data packets by slicing them into small pieces of generations and rely on computation at the receiver side to recover the whole packet. This study is the first attempt to address the communication overhead of BA in the IoT context. Our approach addresses both *long latency* to reach a consensus and *poor scalability* due to enormous communication overhead for disseminating blocks to the whole network.

Specifically, our proposed protocol works as follows: The leader in permissioned blockchain applies linear network coding methods [11] to distribute the block to parties in terms of generations. Each party acknowledges the receipt of the generation and then, again uses linear network coding distribute the same block until reaching consensus according to practical BA. To adapt to our cases, we also applied some fine tuning for the settings of the network coding mechanisms. We proposed two new alternative options to set the re-tranmission period for a generation for reduced consensus times.

In order to test our proposed approach, we built a wireless mesh network testbed using Raspberry-PIs to mimic an interconnected IoT environment. We compared our network coding based approach with TCP-based information spreading on the testbed. The results demonstrated that our approaches based on network coding can significantly reduce the consensus reaching time while decreasing the number of transmissions and hence providing energy efficiency for resource constrined IoT devices.

The rest of this paper is organized as follows. In the next section, we give a broad background about public and permissioned blockchains and BA protocol. In Section III, we explain the details of the proposed information spreading approach to reach consensus between nodes. Section IV contains experimental setup and performance evaluation. Finally, Section V concludes the paper.

2 BACKGROUND

In this section, we provide the necessary background on types of blockchain and BA protocol before we move on to detail our approach.

2.1 Public and Permissioned Blockchain

Blockchain is now a very well-known technology that contains a series of ordered chain of blocks stamped to each other. The chain structure is built by linking the hash of preceding block to the current one, the current one to the succeeding one, and so on. This establishes a strong tie between blocks which guarantees the order of blocks and provides an implicit strong immutability mechanism [13].

The blockchain structure in broad sense is categorized as public and permissioned [14]. The most widely-used Blockchains, such as Bitcoin and Ethereum, fall into public Blockchain category where everyone is able to read and write from/to the ledger without any restriction (i.e., there is no membership requirement). However, in PBC [3], the participants form a members-only club, and thus only the consortium of participants are able to update the blockchain.

The process of adding new blocks to the chain is carried out via a protocol, which establishes consensus such that all participants confirm the new block. The consensus protocols may differ among blockchains. For instance, Bitcoin uses proof-of-work based consensus protocol which requires computation intensive hash puzzle. On the other hand, PBC such as Hyperledger uses some sort of BA protocol [4] which is explained next.

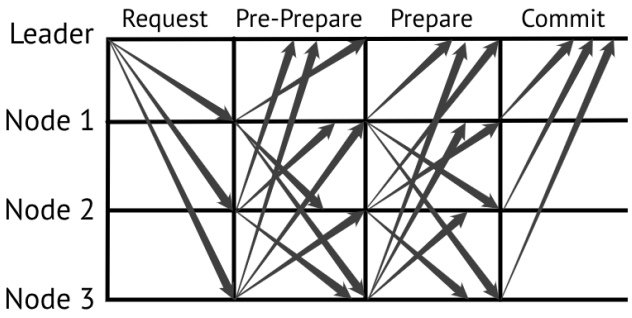


Figure 1: An illustration of how BA protocol works with replicated nodes.

2.2 Byzantine Agreement Protocol

BA method is well-known consensus protocol in distributed computing in case of nodes failures [4]. This algorithm provides reliability of data or computation, even though arbitrary nodes conduct malicious actions, especially when sending and receiving messages that were crafted to disrupt the consensus protocol.

BA is recently started to be used in permissioned blockchain to establish consensus in a byzantine setting where some of the nodes assumed to act maliciously to deteriorate the system. In this setting, where there is n nodes, a consensus can be achieved if at least $(2n-1)/3$ number of nodes act honestly. Honesty means providing correct information to the other participants. In permissioned blockchain, there are two different types of nodes called *Leader* and *Validator*. First, a randomly selected Leader builds a block from transactions. This block is then distributed by leader to the Valitador nodes for verification. Validators will check the transactions within the block and distribute it again to the other Validators after signing it as shown in Figure 1. Each node, then again, distribute the block captured from the other node. This distribution process continues until each Validator node collects individually signed version of the block from the other ones. After $n-1$ version of blocks are gathered, the Validators check differences between blocks. If $(2n-1)/3$ of these blocks are valid, the Validator nodes inform the Leader about confirmation and add the block to its local chain.

3 PROPOSED APPROACHES

3.1 Overview

In PBC, we propose removing proof-of-work based consensus when a new block is to be confirmed since this requires heavy computation that cannot be afforded by resource-constrained IoT devices. Instead, we opt to use a BA protocol. However, as this protocol also requires heavy communication among the nodes (i.e.,n^2 transmissions) that is detrimental for IoT devices, we aim to reduce

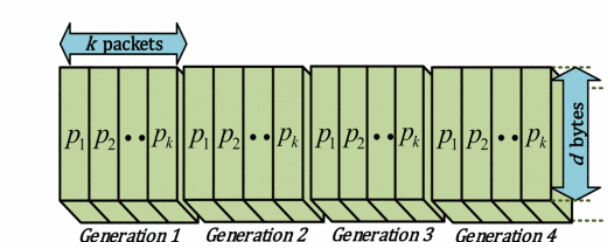


Figure 2: An illustration of how a block divided into generations

this communication overhead by utilizing the concept of network coding.

Network coding is a mechanism that does not simply forward the data, but recombines several input packets into one or several output packets [11]. In communication networks, the information is transported in the same way as cars share a highway or fluid share a pipe, and network coding breaks with this assumption. To better illustrate network coding, let's define a network pair *(G, S)*, where *G* is a finite directed multigraph and *S* is a source node in *G* and does not have any incoming nodes. At *S*, a finite number of data is multicast to other nodes, and all nodes can pass the received data to other nodes. The coding is done at the source node before the data is multicast.

As can be seen, network coding is very suitable for the settings where there is a one-to-many information spreading and the complete information does not merely depend on receiving the packets in specific orders but rather receiving a sufficient number of packets. This fits exactly to our case of the BA protocol, where for each consensus establishment, each node needs to disseminate a certain block to the other ones. In addition, using network coding is shown to bring performance improvements in static settings and thus we adopt it for IoT environment where performance would be an issue.

The details of our approach are as follows: Using network coding, we first divide the block information into g generations. Each generations contains k pieces of chunks each of which is exactly d bytes as shown in Figure 2. Each chunk is encoded via using a network coding library [12] and are broadcast over the network. Each recipients of chunks, try to decode the generation using the encoding vector. When a node completes decoding the generation successfully, it informs (i.e., acknowledges) the source node. If the source node collects all the feedback from nodes which show that every node is completed its decoding, it starts to broadcast the next generation until the block successfully reaches all the parties.

3.2 Fine Tuning for IoT Environments

Network Coding retransmission period for transmissions is critical and may cause unnecessary retransmission of packets or longer waiting time to transfer a generation. Therefore, we propose different retransmission calculation methods in order to converge faster. The proposed calculation methods are described below separately.

3.2.1 Fixed Period. It is a well-known fact that packet loss in wireless communication is not necessarily due to the network congestion. The packet may drop as a result of multi-path fading or RF interference. Hence, increasing the retransmission period may result in waiting longer for transferring a generation.

Instead, we propose setting a maximum timeout value (MTO) and waiting based on this value to retransmit a generation. The MTO is assumed to be *fixed* (and that's why this is the name of the approach) and not updated during the process. For the waiting time, we propose using two mechanisms: 1) **Regular** waiting method where we wait for exactly the MTO value; and 2) **Random** waiting method where we wait for a random time between 0 and MTO for each retransmission period.

3.2.2 Logarithmically Increasing Period. A logarithmically increasing function is increasing slower than any non-constant polynomial. In order to achieve a minimum number of retransmission timeout during network coding, we propose to use a **logarithmically** increasing function to calculate the new retransmission period. In this work, we propose the following logarithmic function to determine the MTO.

$$f(x) = 2.89 * ln(r * MTO) \quad (1)$$

where r is the number of retransmissions.

As we did in the Fixed Period approach, we apply both random and regular mechanisms to determine the individual retransmision period (i.e., waiting time).

3.2.3 Exponentially Increasing Period. In contrast to the logarithmically increasing functions, exponentially increasing functions increase more rapidly. An exponentially increasing function may result in less retransmision in network coding. The main drawback is that it may require more time to successfully transfer each generation to other nodes. We propose the following exponentially increasing function:

$$f(x) = 1.44^r * MTO \quad (2)$$

where r is the number of retransmissions.

Again, random and regular mechanisms are applied in the similar manner.

4 PERFORMANCE EVALUATION

In this section, we describe the experiment setup and present the results along with discussion.

4.1 Experiment Setup

To assess the communication overhead of the BA protocol, we built an IEEE 802.11s-based wireless mesh network that consist of Protronix Wi-Fi dongles (See Fig. 3b) attached to Raspberry-PIs (See Fig. 3a) in the Electrical and Computer Engineering Department at FIU. We first describe how we setup the testbed using IEEE 802.11s.

(a) Raspberry Pi 3.

(b) Protronix USB WiFi Adapter.

Figure 3: Components of the testbed

The IEEE 802.11s module needed a special setup procedure in Linux environment to build the mesh network. We provided this required setup procedure and the used commands in Appendix A. While building the testbed, we carefully dispersed the Raspberry PIs around aisles of the third floor in the Engineering Center. The Raspberry PIs are placed not to be in a line-of-sight position between each other. There are concrete walls and doors between them as can be seen in Figure 4. By this positioning, we try to mimic realistic conditions that reflects the path attenuation, refraction and diffraction of the signal while it propagates through space in wild. Wireless USB Adapter allows Raspberry PIs to create to a wireless network which complies with IEEE 802.11g and shows abilities of transferring data through obstacles even in a steel-and-concrete structure.

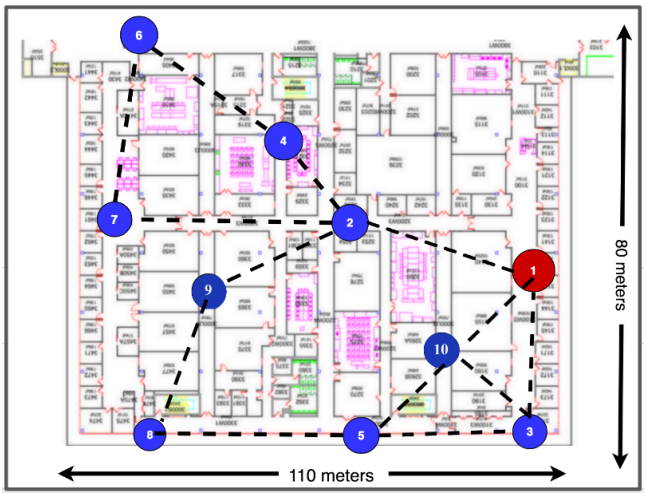


Figure 4: The layout of testbed on FIU ECE floor.

Next, to fairly compare network coding based spreading method with conventional TCP protocol, we changed some TCP stack parameters of Linux kernel to adapt it to lossy wireless environment. The first tweak is related to congestion control algorithm. The Linux kernels after version 2.4 uses Cubic congestion control algorithm in TCP stack. However, the conventional TCP congestion control algorithm works inefficiently in wireless environments which leads to large Retransmission Timeout (RTOs) and poor overall performance. This is a well-known issue in TCP research. Therefore, we enabled F-RTO algorithm which is designed specially for wireless environments to control the RTO parameter. Due to the lossy environment, TCP maximum re-transmission limit is not enough to reach consensus for the 10-nodes testbed as will be discussed later. The required parameter change details are provided in Appendix B. For our approaches, we set the MTO values to 2.5 and 20 seconds according to testbed size of 5 and 10 respectively.

4.2 Performance Metrics

In the experiments, we compared our Network Coding based approach to TCP unicasting since TCP is the currently used approach in other environments for BA protocol. For our approach, the three variations are represented as **Fixed**, **Logaritmic** and **Exponential** in the graphs. We also show Random Network Coding and Regular Network Coding separately. We used the following metrics to assess the overhead and compare their performance:

- *Total Size of the Outgoing Traffic*: This metric shows the total size of the traffic data packets that are sent by an application until reaching a consensus in BA process. Note that this metric is crucial for IoT environments since the devices will be operated and minimizing energy usage is a priority. Reducing the transmission counts will reduce the overall energy consumption.
- *Consensus Establishment Time*: This metric indicates the total elapsed time to reach a consensus via BA process. Depending on the application needs, if there is an urgency for consensus, this metric needs to be minimized.

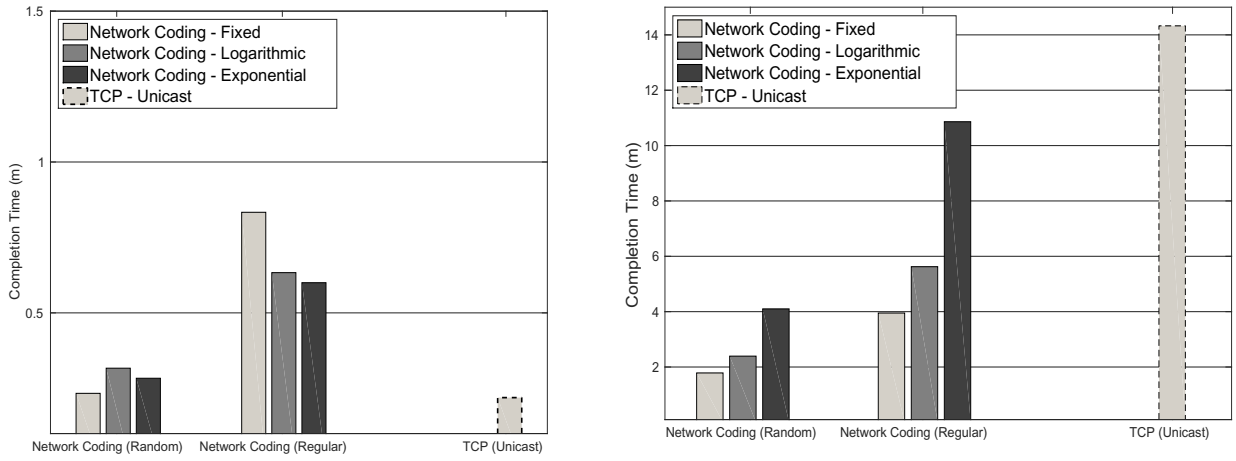
4.3 Experiment Results and Discussion

We compared the performance of our approach with TCP under two different network sizes, 5 and 10, to observe their scalability. We analyze the results under both metrics below.

4.3.1 Consensus Establishment Time. The results for Completion time for all approaches are shown in Figure 5. As can be seen, our network coding based approaches significantly reduce the Completion Time compared to TCP Unicast approach for 10 nodes and provides comparable results for 5 nodes. As expected, the real effectiveness of our approach is when the network size grows to 10 which implicates that network coding approach scales much better. Specifically, the time is reduced by a magnitude of 4 to 8. For the 5-node case, only Random Network Coding approaches provide comparable result to TCP. In particular, **Fixed** approach with random waiting performs the best as the waiting time for retransmissions does not increase and congestion is not much as there are few nodes in the network. However, the trend is reverse for the Regular Network Coding approach. We speculate that 5 nodes is too small to come up with a meaningful pattern.

For the 10-node case, the **Fixed** retransmission function takes the least time for both random and regular cases to reach a consensus when compared to the others. This is due to the fact that the randomly generated retransmission value is fixed and not growing over the time. Since it is fixed, the waiting time for the next retransmission does not increase. Similarly, since the **Logarithmic** function is slowly increasing in the beginning, its waiting time for the next retransmission is far less than the **Exponential** one. Consequently, the exponentially increasing **Retransmission** function reaches the consensus later than the other ones. Another observation is that the regular retransmission method performs worse than the random ones for each particular case. This is a result of collisions during retransmissions. Even if a node waits for an exponentially increasing time to retransmit again, the other nodes also wait for a similar amount, which implies that each node would start transmission synchronously. It causes unnecessary congestion over the network and increases the packet drop rate.

As expected, TCP takes the most time to reach a consensus in the 10-node case. This is due to the fact that TCP requires reliable connection setup and acknowledgements for each pair of communication. In addition, TCP RTO mechanism causes unnecessary



(a) Testbed Size - 5.

(b) Testbed Size - 10.

Figure 5: Consensus Establishment Time for different approaches.

waiting for each packet due to random packet loses. However, the Network Coding based approaches require neither a connection setup process nor congestion control mechanisms which are adversely affected from packet losses as a result of RF interference.

4.3.2 Total Size of the Outgoing Traffic. Next, we look at the total amount of data transmitted by all the nodes. The results are shown in Figure 6. As expected, TCP generates more traffic, because a block should be transferred multiple times for each node. However, the Network Coding approaches produce much less traffic due to the fact that a single packet transfer is adequate for reaching multiple neighbors by broadcasting. The reductions are consistent for both 5 and 10-node cases and is at least in the order of 2 to 4 magnitude.

Finally, we can obeserve that the **Logarithmic** and **Exponential** Retransmision functions in the Network Coding help decreasing the traffic overhead at the expense of high consensus time. When compared to each other, it can be seen that the **Fixed** retransmission method prominently produces the most traffic. This can be attributed to more frequent retransmissions which increase the probability of medium access collisions and packet losses. Therefore, if energy savings is the most critical aspect of the application, **Logarithmic** approach would be better than **Fixed** approach to save more energy. Otherwise, if consensus time is critical, **Fixed** approach should be used.

5 CONCLUSION

In this paper, we investigated a new mechanism for reducing the communication overhead of the BA protocol that is used for PBC applied to IoT settings. Specifically, we proposed network coding to be used in wireless and lossy environments of IoT applications to reduce the number of packet losses and accelerate the consensus time. Several re-transmission timers are tried to fit the needs of IoT.

The proposed BA protocol is implemented in an actual wireless mesh network of Raspberry PI devices at FIU Engineering Center. The experimental evaluation with 10 nodes indicated that network coding can surely decrease the consensus time while also reducing the number of transmissions which enables significant energy savings for resource constrained IoT devices. Consequently, network coding can be a viable approach when applying blockchain technologies to IoT applications.

6 ACKNOWLEDGEMENT

This work is supported by the US National Science Foundation under Grant Number CNS-REU-1757761.

7 APPENDICES

7.1 Steps of Building IEEE 802.11s-based Testbed

First, we need to install *usbutils* tool and required firmware on Rasberry PIs to be able to recognize the wireless adapters by running the following commands.

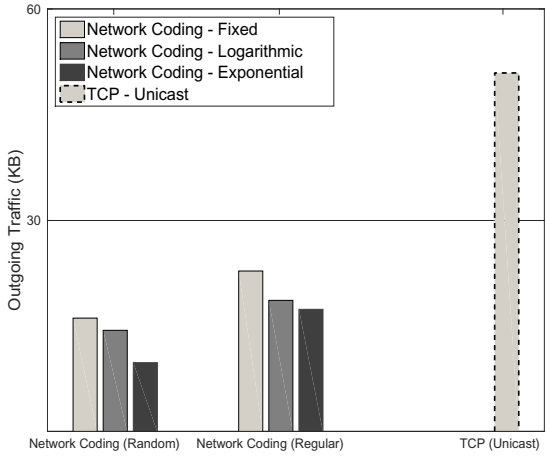
```
$sudo apt-get install wireless-tools usbutils
$sudo apt-get install fimware-ralink
```

Second, to configure wireless interface settings, we installed *iwtool* as following,

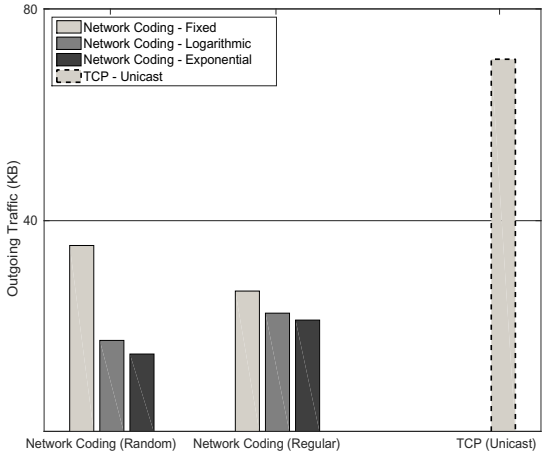
```
$sudo apt-get install iw
```

Finally, we configure the wireless interface using following commands.

```
$sudo iw dev wlan1 interface add FIUmesh type mp
$sudo iw dev FIUmesh set channel 11
$sudo ifconfig FIUmesh 10.1.1.29 netmask up
$sudo iw dev  FIUmesh mesh join
```



(a) Testbed Size - 5.



(b) Testbed Size - 10.

Figure 6: The size of outgoing traffic for different approaches.

7.2 Adapting TCP Stack for Wireless Environments

The default TCP stack installed in Linux does not perform well in wireless conditions and thus leads to frequent timeouts and poor overall performance. Therefore, we changed the default settings of TCP as follows to adapt to the wireless environments.

```
$sudo sysctl net.ipv4.tcp_frto=1
$sudo sysctl net.ipv4.tcp_low_latency=1
$sudo sysctl net.ipv4.tcp_frto_response=2

$net.ipv4.tcp_syn_retries = 15
$net.ipv4.tcp_synack_retries = 15
$net.ipv4.tcp_retries2 = 45
```

The first setting enables F-RTO recovery algorithm for TCP retransmission timeouts. This parameter particularly boosts the performance of TCP stack in wireless environments. The second parameter forces TCP stack to prefer low latency rather than high throughput settings. The third one enables aggressive response to retransmision timeout in F-RTO settings. If F-RTO detects a TCP retransmission, it acts aggressively to re-transmit again. The later parameters are related to maximum retry counts. We increased these parameters accordingly.

Another important settings is related to reliability of the TCP stack. TCP is known as providing a reliable, stream-oriented connection between hosts which guarantees that the data reaches to the destination. However, a naive use of TCP stack to just send the data often fails in multi-hop wireless settings, and causes transmitted data to never arrive to its destination. To avoid this behaviour, we set **SO LINGER** socket option to keep socket alive till the packets reach the destination.

REFERENCES

[1] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *Proceedings of the Thirteenth EuroSys Conference (EuroSys '18)*. ACM, New York, NY, USA, Article 30, 15 pages. https://doi.org/10.1145/3190508.3190538

[2] Louise Axon. 2015. Privacy-awareness in Blockchain-based PKI. (2015).

[3] Christian Cachin. 2016. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, Vol. 310.

[4] Miguel Castro and Barbara Liskov. 2002. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)* 20, 4 (2002), 398–461.

[5] Mumin Cebe, Enes Erdin, Kemal Akkaya, Hidayet Aksu, and Selcuk Uluagac. 2018. Block4Forensic: An Integrated Lightweight Blockchain Framework for Forensics Applications of Connected Vehicles. *arXiv preprint arXiv:1802.00561* (2018).

[6] Ruzanna Chitchyan and Jordan Murkin. 2018. Review of Blockchain Technology and its Expectations: Case of the Energy Sector. *arXiv preprint arXiv:1803.03567* (2018).

[7] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. 2017. Blockchain for IoT security and privacy: The case study of a smart home. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*. IEEE, 618–623.

[8] Niels Hackius and Moritz Petersen. 2017. Blockchain in logistics and supply chain: trick or treat?. In *Proceedings of the Hamburg International Conference of Logistics (HICL)*. epubli, 3–18.

[9] IBM. [n. d.]. Watson Internet of Things. *https://www.ibm.com/internet-of-things/trending/blockchain* ([n. d.]).

[10] Tsung-Ting Kuo, Hyeon-Eui Kim, and Lucila Ohno-Machado. 2017. Blockchain distributed ledger technologies for biomedical and health care applications. *Journal of the American Medical Informatics Association* 24, 6 (2017), 1211–1220.

[11] S-YR Li, Raymond W Yeung, and Ning Cai. 2003. Linear network coding. *IEEE transactions on information theory* 49, 2 (2003), 371–381.

[12] lokeller. [n. d.]. Network Coding Java Library. *http://lokeller.github.io/ncutils/* ([n. d.]).

[13] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[14] Hitoshi Okada, Shigeichiro Yamasaki, and Vanessa Bracamonte. 2017. Proposed classification of blockchains based on authority and incentive dimensions. In *Advanced Communication Technology (ICACT), 2017 19th International Conference on*. IEEE, 593–597.

[15] Yu Zhang and Jiangtao Wen. 2015. An IoT electric business model based on the protocol of bitcoin. In *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on*. IEEE, 184–191.