CrossChain: An Open Platform for Cross-layer Blockchain Research

Ryan Xu, Hakkyung Lee, Ning Zhang, Xuan Zhang
Washington University in St. Louis, St. Louis, MO, United States

***Abstract*—There is an increasing number of possible applications to combine the fields of Blockchain and Internet of Things (IoT) devices. In this paper, we introduce CrossChain, a platform for researchers to start exploring the effects of running a blockchain on an IoT device - specifically a Raspberry Pi. Examples of three different consensus mechanisms were explored and findings are presented below and more elaborately in our repository. A visualization tool provides graphs for several important attributes monitored in the system.**

I. Introduction

It has been more than a decade since a blockchain was introduced to the world for the first time. Blockchain was initially invented by Satoshi Nakamoto as a method of implementing Bitcoin, a decentralized cryptocurrency. With the emergence of Bitcoin, blockchain itself has been developed in various ways, and many trials have been going through to integrate blockchain into various fields until now. Meanwhile, the Internet of Things (IoT), a concept of connecting physical devices mutually via the Internet, has started to appear around a similar period. Cisco System estimated IoT to be started between 2008 and 2009 [7]. Since then, tech companies, regardless of their size, have dived into the IoT field and started releasing devices to connect the world like never before. Recently, Facebook unveiled their Portal product line - with Google following with their own Google Home Hub - in an effort to connect people even more. With two new fields explosively arising over the last decade, a question naturally arises of what possibilities are there to combine these two fields? There exist many exciting proof-of-concept applications but rarely could we find actual steps on how to set up blockchain on an IoT device [4].

We set out to explore blockchain and IoT ourselves. We chose to use Raspberry Pi Model 3B+ (RPi) as our IoT device due to its massive popularity and moderate power - we used a number of these devices to construct a blockchain network. In terms of Operating System (OS), we started out using the Raspbian Stretch, the official OS dedicated for RPi, but later moved to Ubuntu 18.04 LTS for ARM64 due to its lack of 64-bit support. The methodology of switching OS is documented in the later section.

When it comes to selecting which blockchain to run on our RPi, we wanted to explore various distributed consensus algorithms. We considered its popularity within the blockchain field and accessibility of its community for smooth troubleshooting. Moreover, while some blockchains officially support RPi, we excluded them from our list if we could not run a full node for configuring a network. Since we could not test consensus algorithms exhaustively due to our limited time, we narrowed our choices down and selected 4 blockchains, where one of each can represent the behavior of a particular consensus algorithm. The consensus algorithms we agreed on are Proof-of-Work (PoW), Proof-of-Stake (PoS), Proof-of-Authority (PoA), and Crash Fault Tolerance (CFT) based algorithms, and each of them is available via Ethereum Geth, Qtum, Ethereum Parity, and Hyperledger Fabric, respectively. However, unfortunately, after numbers of trials, we have concluded that Hyperledger Fabric is not plausible on RPi due to lack of supporting official base image for ARM architecture and limited hardware memory.

We present our study and experience of installing current state-of-the-art blockchain packages on RPi throughout this paper and introduce CrossChain - a platform to help visualize and monitor the system impacts of different consensus mechanisms on RPi. Using CrossChain, a user can easily keep track of some blockchain information such as current block number, number of transactions within the generated block, and the size of the block. System information such as CPU usage and network data can be retrieved and visualized as well.

II. Related Work

Blockchain in combination with IoT has only started being an active research area recently. Conoscenti et al. [4] conducted a systematic literature review and found 18 use cases of blockchain, but only 4 were specifically designed for IoT. Nonetheless, the potential of the combination cannot be understated as Ksheri outlines in his column [5].

Blockchain use in IoT is still in an incipient stage. One use of blockchain is to leverage the power of smart contracts to automate policy enforcement. Researchers propose concepts on using smart contracts to facilitate the data integrity upon entry in the supply chain[E], and to create a cross-platform IoT collaboration framework[C], where both two works focus on the implementation of the policy. Other researchers investigated setting up a blockchain network for IoT devices: [F] set up a simulation of IoT using RPis and the RPis are full nodes; several other researchers realized the IoT devices can be very limited in terms of the operating system, storage, or computation power, so they designed the blockchain network in a different way: [A] designed a blockchain-based framework for smart home data transaction between IoT device; [B] proposed an access management framework. Both of the work put the burden of storage to outside machines, not IoT devices

themselves. [D] took a different approach, while insisting using a large public chain, [D] managed to make a light client, who does not store the chain data, for IoT devices in smart city use scenario. There are works focusing on the performance too: [B]'s follow-up work did a scalability evaluation on the framework. The framework is tested with an increasing number of virtual clients and the latency, throughputs are measured.

In our work, we took the same assumption F held: we use RPis to simulate IoT devices, a computer with an operating system but limited hardware resource. Built upon this presumption, we designed and implemented an infrastructure for inspections of blockchains with multiple consensus protocols from a performance perspective.

III. Compatibility of Distributed Consensus Algorithms on the IoT device

Blockchain can be categorized into two groups at a high level: a public and a permissioned blockchain. A public blockchain is a network where everyone can participate and contribute to maintaining the network. On the other hand, a permissioned blockchain limits the participants of the network. Within these classifications, each blockchain adopts a different distributed consensus algorithm. While PoW, PoS, PoA and CFT-based consensus are considered on RPi, it turns out that Hyperledger Fabric, which uses CFT-based consensus algorithms, couldn't be hosted on our RPis due to some limitations. Here is a list of classified limitations appearing throughout this section: architecture, memory, processing power, and chain specific. A brief overview is shown in Table I.

A. Proof-of-Work

PoW utilizes a cryptographic hash for mining a new block. Many blockchains, including Bitcoin, are based on PoW. Since the mining process on PoW-based blockchain is a brute-force, it requires a huge amount of processing power. As the number of blocks and participating miners increase, the difficulty of solving the hash problem scales exponentially. Keeping this in mind, Ethereum was considered as PoW-based blockchain due to its popularity and robust documentation. The following details the attempts of running Ethereum with PoW on RPi and observations.

1) Ethereum Geth on Raspbian: Raspbian Stretch was considered as a starting point. Geth, a Go implementation of Ethereum client, was installed without any error. Connecting RPis each other to create a private network and sending transactions from one to another was also successful. However, mining on RPi was impossible. Whenever the mining command was called, an out-of-memory and memory allocation failure error appeared. At first, we assumed that it was due to limited memory. However, after diving into various Ethereum forums, we are reasonably certain as of now Geth requires 64-bit for mining. Since Raspbian Stretch only supports 32-bit, Ubuntu for ARM64, a 64-bit OS, is considered as a next target.

2) Ethereum Geth on Ubuntu: On the same RPi, Ubuntu 18.04 for ARM64 was installed. After following the same steps to set up Geth and create a private network, a full node was successfully set up. This time, mining on RPi was possible, but with a certain limitation. In a private blockchain of three mining RPis, only 13 blocks were mined in the span of 18 hours, even with the initial difficulty of 0. The problem was that RPi could not handle the scaling difficulty. To mitigate this issue, Geth had to be modified and recompiled. The difficulty was hardcoded to be 0x50 so that it remains a constant. This value, which is a heuristic, allows RPi to mine one block per 10 seconds on average. Thus we concludes that PoW on RPi is possible, but there still exists a processing power limitation.

B. Proof-of-Stake

Although PoW is the mainstream consensus, wasting a massive amount of computational power is one of its most critical drawbacks. As an alternative, PoS was proposed. The mining process on PoS-based blockchain mainly depends on the total amount of cryptocurrency a node is staking. Given that PoS is energy efficient and less computationally heavy, PoS seemed to be well-suited for IoT devices. Initially, Ethereum with PoS was considered as a candidate. However, it turns out that it failed to run on RPi. Qtum was considered as an alternative.

1) Ethereum Prysm on Ubuntu: Ethereum has been planning and working on switching from PoW to PoS. Prysm, the only PoS-supported Ethereum client, is tested on RPi for the first target. Installing Prysm can be done in two ways: using Docker and Bazel, a build tool. Docker method did not work due to architecture compatibility. While installing Bazel was successful, building Prysm via Bazel was impossible due to the out-of-memory issue. Even after setting up a swap memory, RPi crashed in the middle of the building process. Concluding that running Prysm is not possible with our RPi due to memory limitation, Qtum, a next candidate, was tested.

2) Qtum on Ubuntu: Since Qtum officially supported ARM architecture, installing was relatively simple. However, to set up a private network, regtest mode had to be used, which comes with limited functionalities. Note that in regtest mode, while it is possible to run PoS, it automatically generates a block for every 30 seconds. With that said, Qtum provides less control over the staking process. It is either joining the staking process or not. Moreover, a user cannot control the staking amount. As soon as the confirmation of 500 blocks is met for each block, the balance acquired from that block goes into staking automatically. Thus it is concluded that PoS is possible on RPi, with chain specific limitations.

C. Crash Fault Tolerance Protocol

Crash Fault Tolerance (CFT) is a protocol in which a consensus can be reached even with a certain number of faulty nodes. However, unlike Byzantine Fault Tolerance (BFT) protocol, CFT cannot maintain the network under the presence of malicious nodes. CFT is often used for a permissioned blockchain, because the participants are usually identified

TABLE I: Current state of support of blockchain on RPi
(O : Supported △ : Partially supported X : Not supported)

	Ethereum (Geth)	Ethereum (Parity)	Ethereum (Prysm)	Qtum	Hyperledger Fabric
Availability on RPi	△	O	X	O	X
Configuring a private network	O	O	X	△	O
Consensus algorithm	Proof-of-Work	Proof-of-Authority	Proof-of-Stake	Proof-of-Stake	Crash Fault Tolerance based (Raft, Kafka)
Operating System	Ubuntu ARM64	Ubuntu ARM64	N/A	Ubuntu ARM64	N/A
Limitations	Processing power	Chain specific	Architecture, Memory	Chain specific	Architecture, Memory

ahead of time. Hyperledger Fabric, one of the most popular permissioned blockchain, leverages this fact by adopting CFT-based consensus algorithms because when deployed within a single enterprise, or operated by a trusted authority, fully Byzantine fault-tolerant consensus might be considered unnecessary and an excessive drag on performance and throughput [3]. Hyperledger Fabric currently supports Raft and Kafka, a CFT implementation based on "leader and follow" model.

1) Hyperledger Fabric on Raspbian: Hyperledger Fabric, a popular permissioned blockchain framework, uses Docker images to build the binaries. Unfortunately, up-to-date images built for ARM architecture was not found. Although images that third parties compiled for ARM was found, all were either very outdated or not for Raspbian. Hyperledger Fabric community also stated that as of now Hyperledger Fabric does not officially support 32-bit. On top of that, we recently noticed that the official wiki page recommends at least 4GB of memory for running Hyperledger Fabric, which is another obstacle for RPi. Since RPi does not meet the hardware requirement, running on different OS was not considered and it is concluded that running Hyperledger Fabric is not possible due to architecture compatibility and memory limitation.

D. Proof-of-Authority

PoA is yet another type of consensus algorithm, which was proposed as an alternative of PoW. Similar to PoS, PoA does not require a huge amount of computing power for generating blocks, which makes it suitable for a low power device. However, while PoS requires staking the monetary value to become a validator, PoA utilizes an identity of the validator as a form of staking [2]. Since staked identity relates to the reputation of the validator, misbehaving will directly harm the reputation of the validator in the real world. In this way, the validators will highly likely align securing the network with keeping one's reputation intact.

Currently, three different PoA implementations are available on Ethereum: Aura via Parity, a Rust-based Ethereum client, Clique via Geth, and IBFT 2.0 via Pantheon, another Ethereum client written in Java. While Pantheon is an open source and capable of building a private network, it is mainly targeted for enterprise usage. In addition, the Java-based program requires lots of memory to run smoothly, which is a crucial reason Pantheon was not selected here. Between Parity and Geth, Parity with Aura consensus engine was chosen mainly because Parity provides more comprehensive official documentation regarding PoA settings.

1) Ethereum Parity on Raspbian: The first trial was building Parity from source code. Since Parity is written in Rust, it uses Cargo, a package manager for Rust, for building. While installing dependencies was successful, building via Cargo was not due to the out-of-memory issue. Since this could be an OS issue just like Geth case, Ubuntu for ARM64 is considered.

2) Ethereum Parity on Ubuntu: Again, installing Parity from the source failed due to the same error. After a bit of searching, it turns out that 1G of RAM was not enough to support one of the main dependencies, rustc, a Rust language compiler. Luckily, Parity was also officially distributed via Snap, an Ubuntu packaging manager. Using Snap, Parity was successfully installed on RPis. Furthermore, there is one benefit for using Snap; whenever there is a new release, Snap automatically keeps the version up to date. Running a full node and setting up a private network was also possible without any constraints. However, there is one chain specific limitation. Since a validator should be selected beforehand, it is hard to trust and grant the authority to a person or a group who is not known and cannot meet face to face. Assuming that PoA is used among trusted groups, it is concluded that running PoA on RPi is by far the best option.

IV. CrossChain Framework Design

Taking in our findings of the various blockchain implementations, we sought to create an introductory infrastructure to assist researchers in exploring blockchain and IoT. Introducing CrossChain, this platform supports examples of three consensus algorithms present - PoW, PoS, and PoA - on multiple RPis with Ubuntu 18.04 for ARM64. As explained before, Geth, Qtum, and Parity are used as clients, respectively. CrossChain has several aspects that can be useful to researchers.

First, thorough documentation found in the repository helps how to install, set up, and run blockchains on the RPis. With limitations and incompatibilities in linking many conventional blockchains to RPis, this is a reliable and up-to-date tutorial

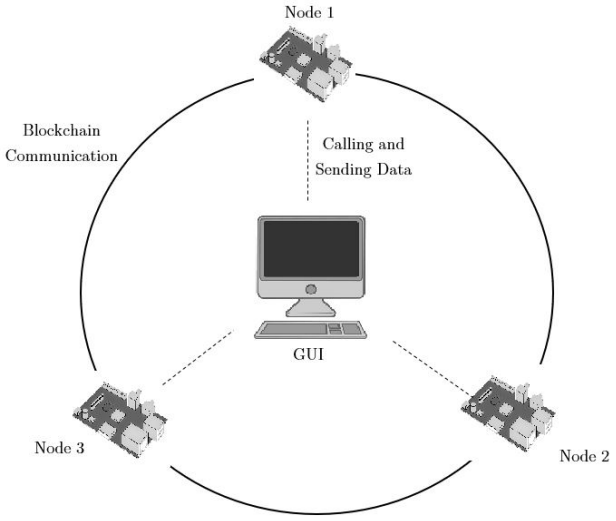

Fig. 1: System Work Flow Diagram

and resource. This hopefully will allow researchers to quickly get a network up and running without going through trial-and-error with numerous chains.

Secondly, a basic Graphical User Interface (GUI) was created to help visualize some attributes that would be helpful to researchers. For each blockchain, a block number, difficulty, number of transactions, and the size of each block can be gotten and graphed. Besides, the CPU usage of RPi in terms of percentage is graphed to comprehend the computing load on the device. Network data such as bandwidth, packets, latency, and jitter are also tracked. These attributes can be analyzed to determine how well blockchains operate on the RPis.

A user can add RPis that have been set up with the to CrossChain GUI and choose which attribute to be graphed. This helps visualizing data in a comprehensible manner. Additionally, there is an option to log all the block data collected as well as set the sample rate of network and system data. This data is packed into a CSV file that is categorized by the node, type of data, and consensus algorithm. In addition, the sample rate of the network and system features can be adjusted, though we will discuss later some minimum rate limitations. Finally, switching between blockchain can be done via GUI, though this involves starting and stopping the chain remotely. This will aid making informed comparisons between different blockchains.Though GUI at the current stage is rather rudimentary, it is an viable starting point for exploring.

It should be noted with the current setup of CrossChain the difficulty attribute is to be taken with a grain of salt. In PoW, since Geth is modified, the difficulty of the chain is fixed and the difficulty magnitude is the same for all blocks. In PoS, the difficulty doesn't affect the chain due to the staking mechanic - the winner is determined by how much is staked and not difficulty, so the actual block difficulty is set very low so that the winner can quickly generate the block. It is apparent in QTUM as the difficulty per block is on the order of the nano scale. In PoA, the protocol does not use a difficulty parameter at all so the value is simply a placeholder.

V. CrossChain System Implementation

A. Blockchain

After many trial and error based on several online tutorials and documentation, detailed guides on how to construct the three blockchains can be found in the project repository.

PoW can be tested on Geth. As briefly explained in III-A2, Geth was modified to make mining more feasible on the RPis. Normally in PoW, the difficulty for mining a new block scales based on the number of coins mined so far as well as the number of nodes in the network, and it usually scales exponentially. Unfortunately, due to the hardware limitations of RPi, even with initial difficulty of 0, the default scaling factor is too high for the RPi to mine at a reasonable rate. For demo purposes, the difficulty is fixed to 0x50, which results in mining a block about every 10 seconds on average. This fix requires Geth to be recompiled. Again, due to limited memory, it was cross-compiled and then transferred to the RPi.

PoS can be tested with Qtum's regtest mode, where it is suitable for creating a private network and generating blocks and transactions. As Qtum's initial state is a PoW/PoS mixed stage for the initial 500 blocks, presumably for the ease of growing the balance quickly, we chose to generate 500 blocks manually and to wait for a few hours to transit into a pure PoS stage.

PoA is available on Parity. Setting up a private chain with Parity is quite similar to Geth. Simply prepare a genesis file that serves as the genesis block and instantiate the chain. Unlike PoW or PoS, PoA has a validator list, a list of authorities who can seal a block. Make sure to designate a validator beforehand to watch it creating new blocks. Furthermore, as the only validators can seal blocks, the block generation time is fixed to 5 seconds, which can be adjusted within the genesis file, and the difficulty value turns out to be meaningless as a result.

B. Graphical User Interface

The GUI was written in Python and utilizes PyQt5 and pyqtgraph packages to create the interface [8,9]. A user can add nodes to the GUI, which will create a Secure Shell (SSH) connection. The connected node uses four monitoring threads - one for each function call. Figure 2 shows a class diagram of the GUI.

The first thread tracks the blockchain depending on which consensus algorithm is selected. For PoW and PoA, Web3.py - a python library for interacting with Ethereum - is utilized. A filter is used to be notified when a new block is mined, whose data will be parsed and collected. The coinbase address is checked to see which block successfully mined. For PoS, Qtum has its command line interface, qtum-cli. There are a few more steps to obtain attributes such as block size compared to Web3, but it can be done nonetheless. Because Qtum doesn't provide a filter for a new block, the block number is compared instead - if it is different, we know a new block has been

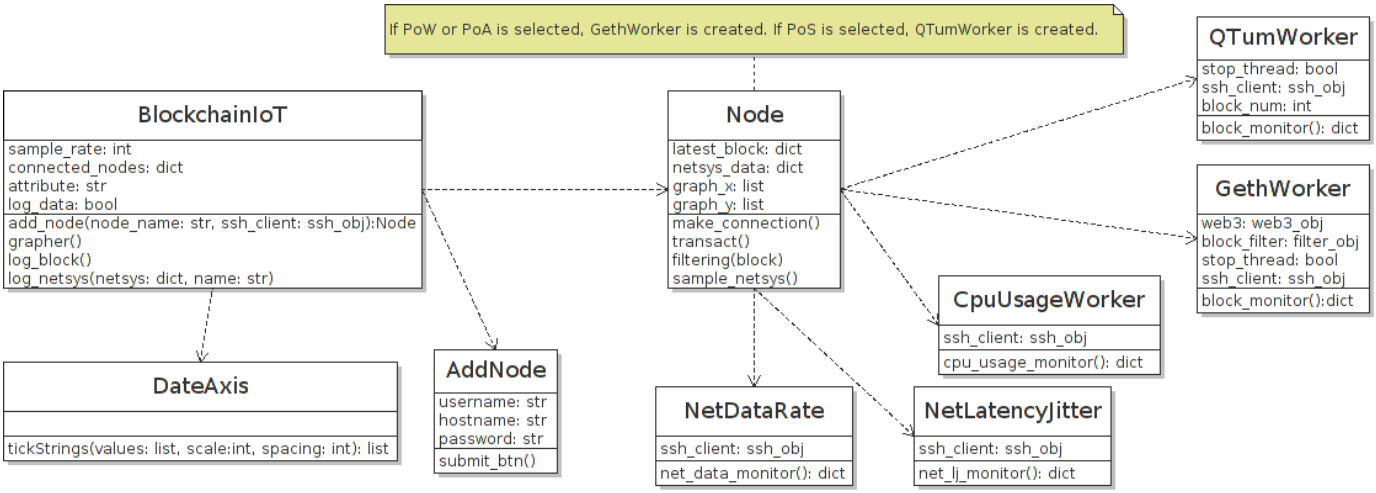

Fig. 2: GUI Class Diagram

created. It is important to note that when the chain gets longer, the time it takes to call some functions does get on the order of tens of seconds.

The second thread tracks CPU usage by parsing */proc/stat*. The third thread follows network bandwidth and packets using a tool called bwm-ng [10]. Finally, the fourth thread monitors network latency and jitter using *ping* command to test the reachability of the network. Since measuring power consumption requires external hardware, it was excluded.

All of the data is packaged in dictionaries and sent via signals to each node. A graphing function then plots whichever attribute has been selected to plot. For block-specific attributes, the graph refreshes with new data whenever a new block is created. For system and network-specific attributes, the graph refreshes with new data according to the sample rate, which can be changed. If log data is selected, information will be logged in two different JSON files per node - one for block data and the other for network and system data.

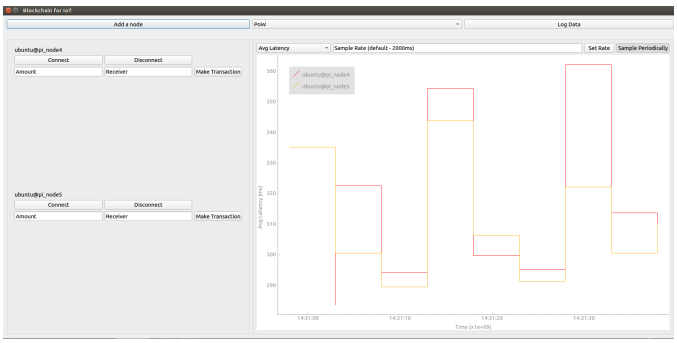

Fig. 3: Graphing Latency on GUI

VI. Evaluation

Figure 4 shows the hardware platform of the experiment. Figure 3 depicts a screenshot capture of the GUI graphing average latency for two nodes. Although CrossChain can fully

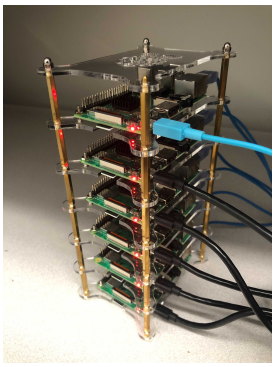

Fig. 4: Experiment Hardware Platform

supports examples of PoW, PoS, and PoA while creating visualizations for attributes, the capabilities of the RPi are limited; running a mining node as well as sampling multiple systems and network attributes can accumulate to a fairly heavy load on the system. To check the overhead in numeric values, the performance overhead of the monitor functions in Figure 2 and the run time of RPC are measured and summarized in Table II and Table III.

TABLE II: The runtime of monitor functions

CPU usage	Network Bandwidth	Network Jitter and Latency
1.0272s	0.511s	0.8248s

The first type of the overhead comes from the run time of the *ssh* commands used to get the performance stats, including CPU usage and network indicators. We set up a 3-node network of PoS and recorded the run time of each command through SSH. As these commands are consensus independent, we believe that the numbers are representative. For CPU usage and network bandwidth, three samples each node was taken and the average was calculated. For jitter and latency, the function measures the communication between two IP addresses, so we measure the run time of this command

from NodeA to NodeB, NodeC to NodeB, NodeB to NodeA, each for three times. It is worth noting that the run time of this command highly depends on the configuration, where the user can set up the number and interval of the signals sent from one IP to another IP. In our system, the default is set as 5 signals in total with the interval being 0.2s. The run time of these commands is critical to be measured especially due to the way we set up our system. Monitor threads are created according to the sample rate, so if a function has not finished before the sample rate has elapsed, a second instance of the same thread will be created, causing a crash. We took three samples of each three nodes and the average is presented below.

TABLE III: The runtime of RPC

PoW	PoS	PoA
0.37s	14.01s	0.04s

Second type of the overhead comes from the run time for the RPC used to retrieve the information needed for blockchain stats. A long run time can not only increase the wait time for the user, but also potentially lead to not up-to-date data. Although the run time for the RPC is blockchain specific, we decided to show the recorded time of the three blockchains we deployed below. For the testing purpose, we decided to use the RPC to get difficulty as the representative of the RPC. The recording is done in three nodes, each repeated three times and average is reported. As the PoA and PoW are both Ethereum, they use the same set of APIs, we only did the testing on PoW. Also, through testing we found that the run time of RPC can vary among the nodes of the same private chain and we put this finding as a problem to investigate in the future.

VII. Conclusion and Future Work

We presented our experiences trying to configure popular blockchains on RPis. While some trials did not succeed for various reasons, we managed to install a blockchain for each of the three main consensus methodologies. A Detailed explanation of how to set the chains up can be found in the project repository. We also provided a visualization tool, CrossChain, to aid researchers in exploring the reactions of the RPis while running blockchains.

There are a few avenues of future work. The first is adding more functionality to the GUI to make it more robust and automated. Secondly, as IoT devices become more powerful, it is intriguing to see how improved hardware can help the devices mine blocks more efficiently. At the time of writing this paper, the Raspberry Pi Foundation has released the Raspberry Pi 4, which has a configuration with 4 GB of RAM. Knowing the how much performance increases in relation to hardware improvements may be important in future IoT and blockchain development.

References

[1] POA Network. (2017). Proof of Authority: consensus model with Identity at Stake. [online] Available at: https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256.

[2] Hyperledger. (2019). Introduction hyperledger. [online] Available at: https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html.

[3] Conoscenti, Marco, Antonio Vetro, and Juan Carlos De Martin. "Blockchain for the Internet of Things: A systematic literature review." 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA). IEEE, 2016.

[4] Kshetri, Nir. "Can blockchain strengthen the internet of things?." IT professional 19.4 (2017): 68-72.

[5] Dorri, Ali, Salil S. Kanhere, and Raja Jurdak. "Blockchain in internet of things: challenges and solutions." arXiv preprint arXiv:1608.05187 (2016).

[6] Evans, D. (2011). The Internet of Things - How the Next Evolution of the Internet Is Changing Everything. [online] Cisco. Available at: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[7] Riverbank Computing Limited. "PyQt Whitepaper" https://www.riverbankcomputing.com/static/Docs/PyQt4-whitepaper/pyqt-whitepaper-us.pdf

[8] Luke Campagnola. (2016) Pyqtgraph https://pypi.org/project/pyqtgraph/

[9] Volker Gropp. https://github.com/vgropp/bwm-ng