Using Blockchain to push Software-Defined IoT Components onto Edge Hosts

Mayra Samaniego
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada

mayra.samaniego@gmail.com

Ralph Deters
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada

deters@cs.usask.ca

ABSTRACT
Moving IoT components from the cloud onto edge hosts helps in reducing overall network traffic and thus minimizes latency. However, provisioning IoT services on the IoT edge devices presents new challenges regarding system design and maintenance. One possible approach is the use of software-defined IoT components in the form of virtual IoT resources. This, in turn, allows exposing the thing/device layer and the core IoT service layer as collections of micro services that can be distributed to a broad range of hosts.

This paper presents the idea and evaluation of using virtual resources in combination with a permission-based blockchain for provisioning IoT services on edge hosts.

CCS Concepts
• **Computer systems organization → Embedded and cyber-physical systems**

Keywords
IoT; Edge Computing; Blockchain; Virtual Resources; Software-Defined IoT

1. INTRODUCTION
The Internet of Things (IoT) is based on the idea that by connecting physical devices with the web, new and richer interactions between devices, services and users are possible. Gubbi et al. [1] identify two views in IoT namely thing-centric and cloud-centric. The thing-centric view focusses on the enhancement of a thing and richer user experiences when engaging it. Smart objects [2] or enchanted devices [3] like an umbrella that actively seeks out its owner to warn her/him of rain are the most prominent examples in this category. The cloud-centric approach [1,4,5] move the focus away from the thing/device towards the IoT services and applications that process large data streams emanating from large collections of things/devices. Consequently, this view is primarily concerned with the requirement to scale e,g, handle/manage large numbers of connected devices.

This cloud-centric view typically implies three core layers, namely thing, cloud and application layer. The bottom or thing layer compartmentalizes the "network(s) of things" [1] e.g. sensor networks. The primary purpose of this layer is to handle all thing related aspects (e.g. provisioning, maintenance, configuration, etc.) and to provide an interface to the middle or cloud layer. The center or cloud layer hosts all core IoT services e.g. data storage, analytics, storage, etc. Some cloud-centric models host virtualized things (e.g. use proxies) others just the input and output streams of the things. The third and final layer is hosts the IoT application e.g. surveillance, monitoring, managing, etc. Since the applications cannot directly engage the things, they use the services and abstractions offered by the cloud/middle layer.

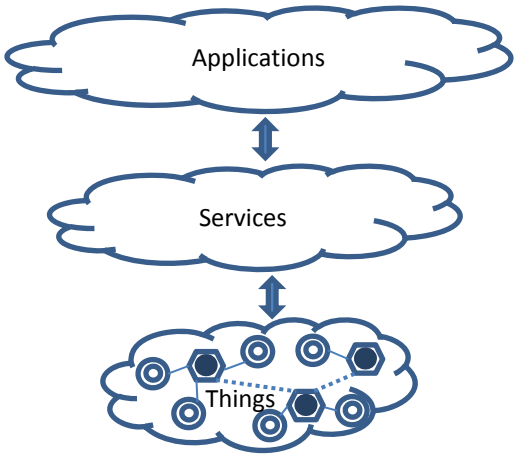

Figure 1. Three layers of cloud-centric IoT[1].

BDAW'16, November 10-11, 2016, Blagoevgrad, Bulgaria

DOI: http://dx.doi.org/10.1145/3010089.3016027

A key challenge in the cloud-centric approach is the bandwidth needed for moving the data streams into the cloud and the network latency that results from separating the things from the computation. This, in turn, leads to a marginalization of things since they are reduced to configurable data sources. Inter-device communication or the use of services to improve the capabilities of the device are therefore not the focus of cloud-centric solutions. To overcome the well-known latency issues of cloud-centric solutions [6] it has been suggested to move computation closer to the things (edge of the IoT system) e.g. via fog-computing [7,8] or edge-computing [9]. However, this, in turn, raises new issues e.g. how to enable the provisioning of services across cloud, fog and edge hosts.One possible approach is the use of software-defined IoT components in the form of virtual IoT resources. This in turn allows exposing the thing/device layer and the core IoT service layer as collections of micro services that can be distributed to a broad range of hosts. This paper presents the idea of using software-defined IoT components [10,11] called virtual resources and the use permission-based blockchains as a means for distribution.

2. Software-Defined IoT Components

Software-defined networking (SDN) [12, 13] is a management concept that centers on using abstraction to enable the decoupling the control plane (determine destinations of traffic) and data plane (forwarding traffic). Adopting this concept in the IoT space has led to the rise of Software-defined IoT (SD-IoT) [14]. SD-IoT uses abstraction to simplify provisioning and customization of its components. Network Virtualization [15] goes beyond SDN by focussing on the virtualization of all components resulting in the ability to define customized virtual networks.

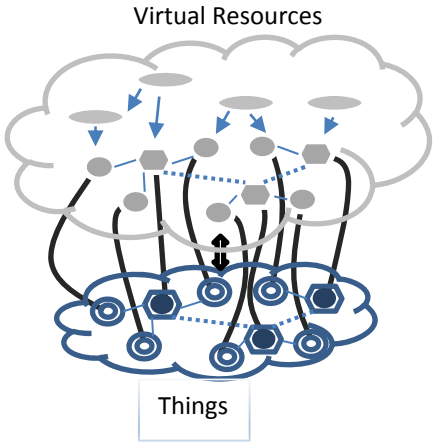

Figure 2. Virtual Resources.

Virtualization has been proposed within IoT e.g. virtual sensors [16]. However, these approaches focussed on abstracting individual components ignoring the potential of defining virtual IoT systems. To enable the creation of virtual IoT systems on top of an existing IoT system we propose the concept of a virtual resource. A virtual resource is a restful service that offers a "view" on one or more IoT components. Virtual resources are restful micro services that process requests by either engaging other restful services or by using their own internal state. By defining these views on top of existing components, it becomes possible to create N virtual IoT systems on top of an existing one. Virtual resources are a design construct and can, therefore, be implemented in a variety of ways. We choose to use of the language Go and implement virtual resources as go-routines that are hosted in go programs. To create views at runtime, go function must be provided to the system e.g. via the blockchain.

3. Blockchain

A blockchain [17] is decentralized ledger that contains connected blocks of transactions. The fundamental concept behind the blockchain is that tamper-proof storage of approved transactions. Valid/verified transaction are stored in the form of a block (lists of transactions) that is linked to the previous one. A blockchain starts with an initial or genesis block. Upon creation of a new block it the hash value of the preceding block is entered. Once a new block is formed, any changes to a previous block would result in different hashcode and would thus be immediately visible to all participants in the blockchain. Consequently, blockchains are considered tamperproof distributed transaction ledgers. Originally designed as the distributed transaction ledger for BitCoin [17], the idea of using blockchains has spread.

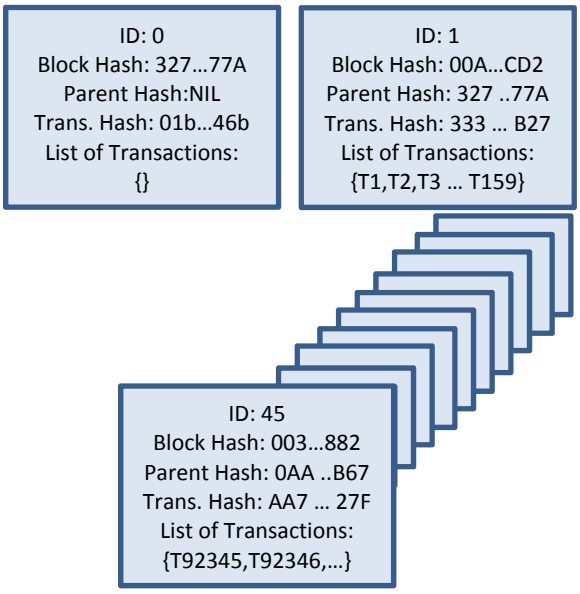

Figure 3. Sample Blockchain.

In the context of IoT the most well-known example of blockchains is IBM's ADEPT system [18] that is built on Bluemix. ADEPT can store the configuration of IoT devices and as a mechanism for pushing code onto devices.

4. Evaluation

To evaluate the use of edge-hosted virtual resources three basic experiments we wanted to evaluate communication costs, using edge-hosted data stores and using a blockchain for storing state of virtual resources.

4.1 Communication Costs

To assess the data transmission costs of edge-hosted virtual resources, we use two Intel Edison Arduino boards. As shown below, these boards consist of an Uno R3 compatible Edison breakout board and an Intel Edison module. The Edison module is a System on a Chip (SoC) comprising a 500 MHz dual-core, dual threaded Intel Atom and a 100 MHz 32-bit Intel Quark microcontroller.

Figure 4. Intel Edison SoC with Edison breakout board.

The two boards used in the experiments are both connected to the same shared WiFi network (same access point). Each board runs one virtual resource. One virtual resource sends 1000 sequential requests to the virtual resource on the other board. After sending the request, the virtual resource waits for the acknowledgment and only then continues with the next request.

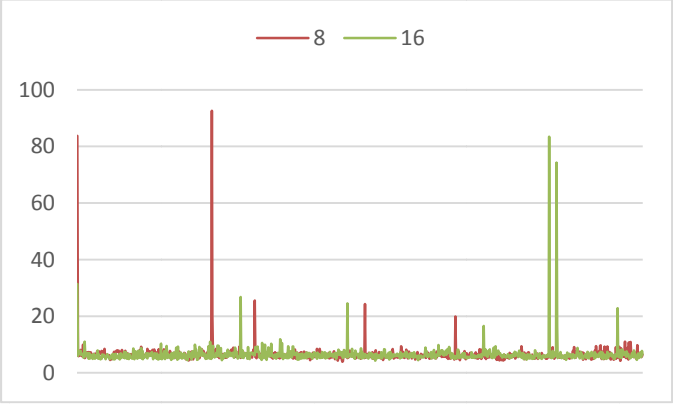

Figure 5. 8 & 16 bytes

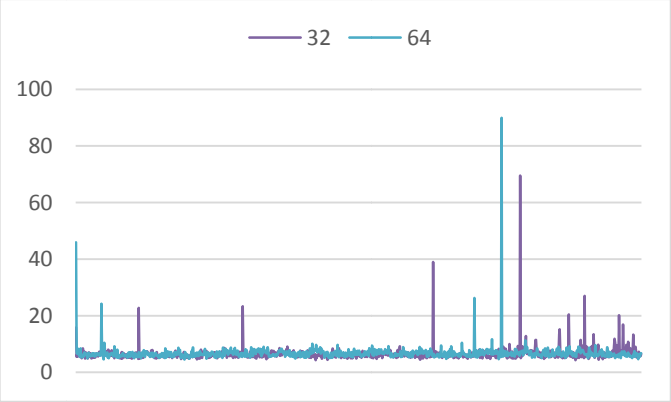

Figure 6. 32 & 64 bytes

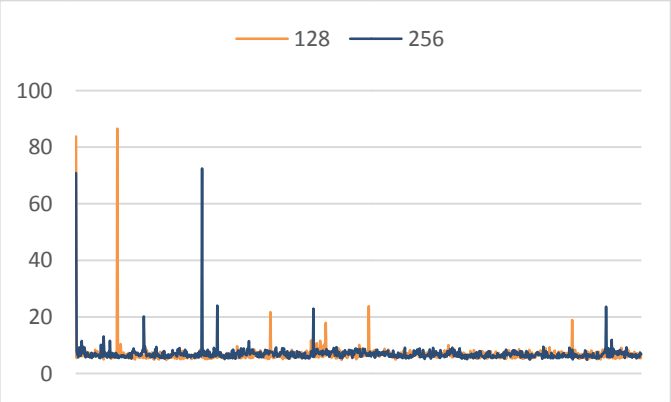

Figure 7. 128 & 256 bytes

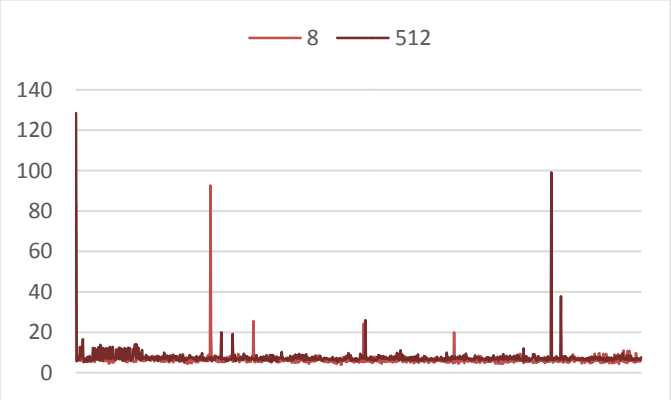

Figure 8. 8 & 512 bytes

The figures 5 – 8 show that (with a few outliers) the time to send 8-512 bytes of data is on average somewhere between 6-8 milliseconds.

To evaluate the impact of multiple virtual resources on edge-devices the number of is increased to 10. In this new experiment, one board hosts ten concurrently running virtual resources. Each virtual resource sends 100 requests to the virtual resources hosted on the other board. The payload in the different runs is 8, 16, 32, 64, 128, 256, 512

and 1024 bytes.Each figure shows the data for one of the ten concurrently running virtual resources. Please note that the following figures show the times of 100 requests for a randomly selected virtual resource.

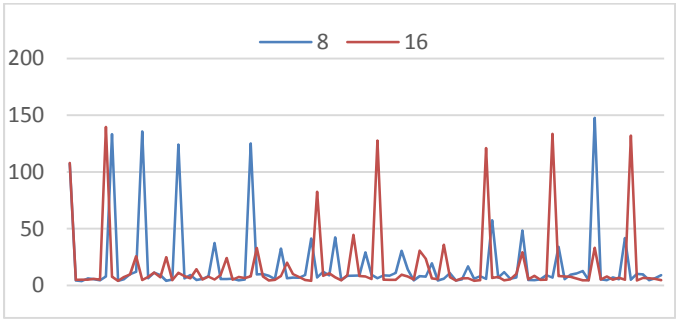

Figure 9. 8 & 16 bytes

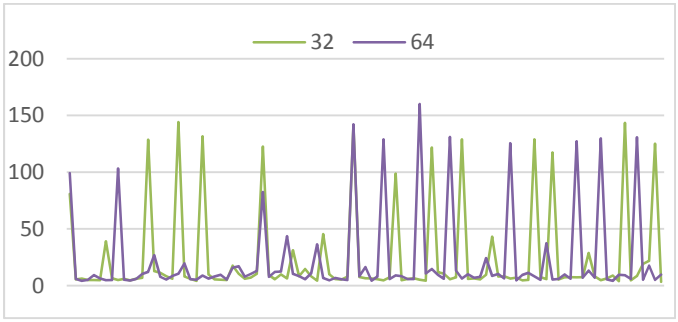

Figure 10. 32 & 64 bytes

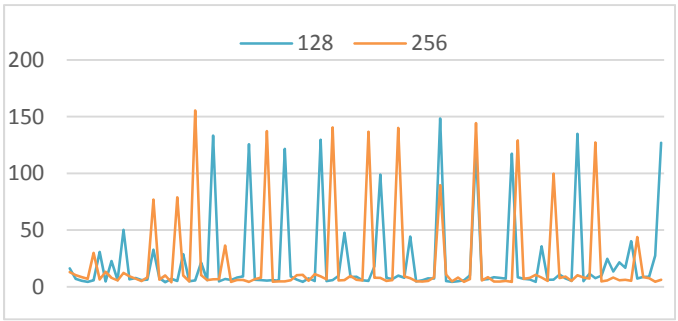

Figure 11. 64 & 128 bytes

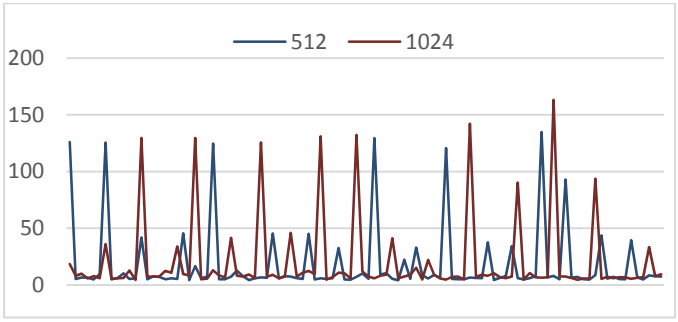

Figure 12. 512 & 1024 bytes

Figures 9-12 indicate that running 10 concurrent virtual resources that continuously send/receive data does result in increased latency. The averages for the various payload are now around 22 milliseconds. Also many spikes can be observed. Given that the virtual resources are go-routines (lightweight threads) the peaks can be attributed to the scheduler's context switches. For a variety of reasons, it is advisable to encrypt data. Given the limited resources available on the Intel Edison (500 MHz dual-core, dual-threaded Atom) symmetric encryption (AES) is desirable. In the following experiments, the same two boards (fig. 4.1) are used. Again the boards are connected to a WiFio network, and again each board hosts ten concurrently running virtual resources. However, the payload is now kept constant (256 bytes), and all data is encrypted using AES (both boards share the secret key). Again the first experiment is conducted with only two virtual resources, each hosted on one board.

The first virtual resource generates 1000 sequential requests that are sent to the second virtual resource. To evaluate different arrival times, a delay is introduced. The payload was kept at 256 bytes that are encrypted using AES. The virtual resource waits 0, 50, 100,150, 200, 250, 300 milliseconds before issuing the next request. Please note that the following figures show the results for 1000 results.

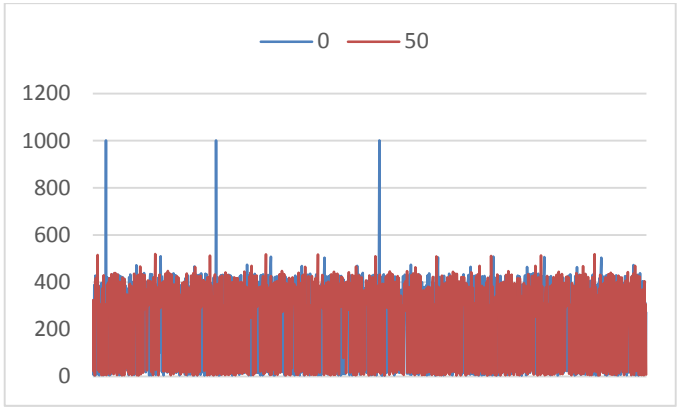

Figure 13. 0 & 50 milliseconds delay

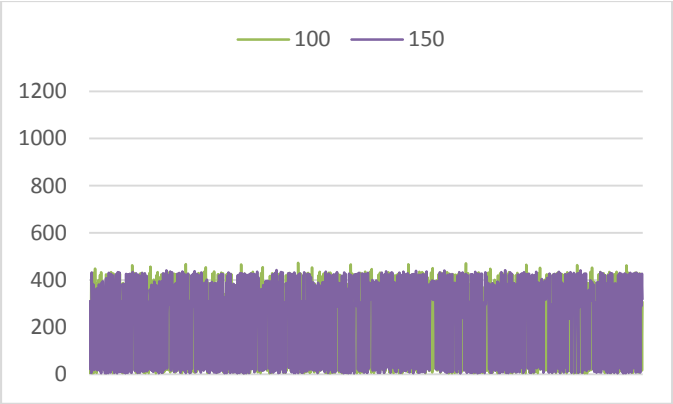

Figure 14. 100 & 150 milliseconds delay

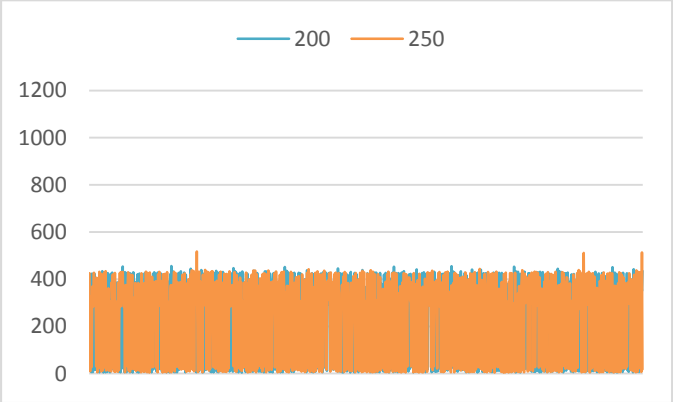

Figure 15. 200 & 250 milliseconds delay

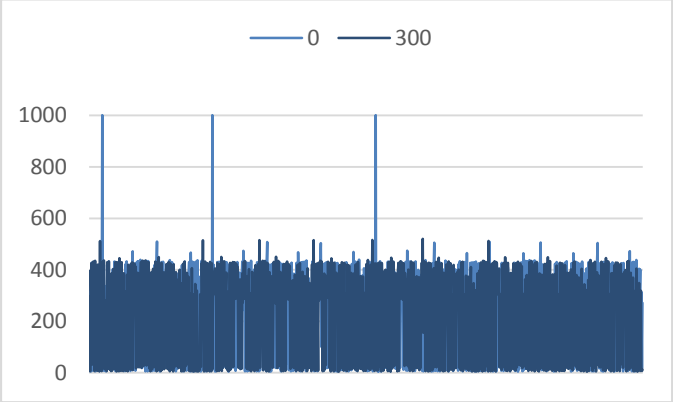

Figure 16. 0 & 300 milliseconds delay

Figures 13 – 16 show that by adding the encryption the time has increased to an average of 225 – 240 milliseconds.

To evaluate the impact of multiple virtual resources on edge-devices the number of is increased to 10.

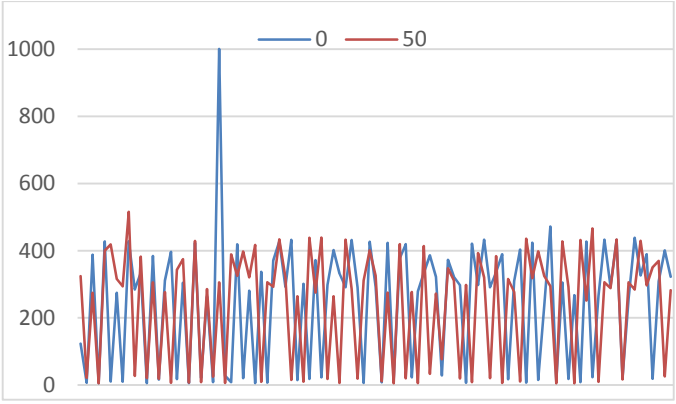

Figure 17. 0 & 50 milliseconds delay

In this new experiment one board hosts 10 concurrently running virtual resources. Each virtual resource sends 100 requests to the virtual resources hosted on the other board. The payload is kept at 256 bytes and each virtual resource waits 0, 50, 100, 150, 200, 250, 300 milliseconds before issuing the next request. Please note that the following figures show the results for 100 request/responses.

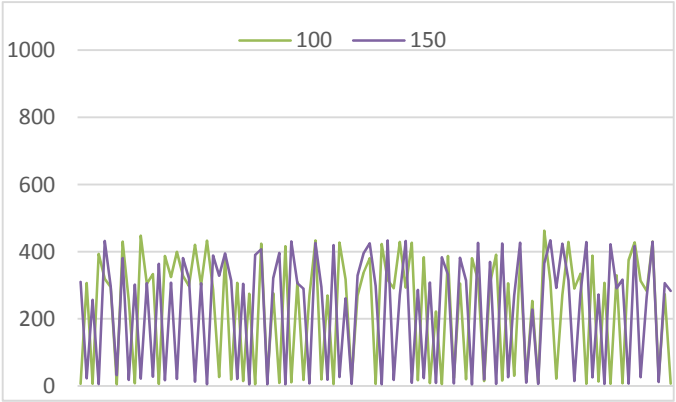

Figure 18. 100 & 150 milliseconds delay

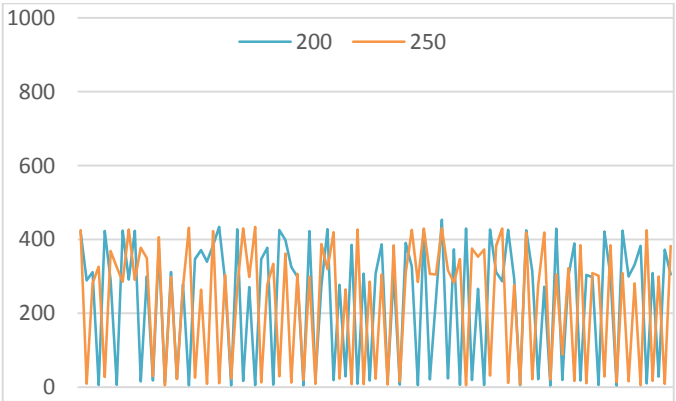

Figure 19. 200 & 250 milliseconds delay

As can be seen in the figures 17 – 19, the adding of delays has no significant impact on the performance. Obviously, the time for sending/receiving has increased. However, it is important to note that the time for encryption and decryption is now included in the sending of requests.

4.2 Edge-Hosted Data-Stores

Enabling virtual resources to store data in a persistent manner is of great practical importance. Given the limited computational resources of the presented Intel Edison board, it is advisable to use single-board-computers like the raspberry pi as hosts for the database. In the following tests two raspberry pi 2 are used (Raspbian 3.18.7-v7+, 900 MHz quad-core ARM Cortex-A7 CPU, 1GB Ram). One raspberry pi 2 hosts ten virtual resources and the other the Elasticsearch DB with 20GB storage). The devices are again connected via the WiFi network. The following figures show the round-trip-time (sending read request and receiving data). To measure the impact of arrival rates each of the 10 concurrently running virtual resources waits 0, 50, 100, 150, 200, 250 or 300 milliseconds after receiving the result before issuing the next request.

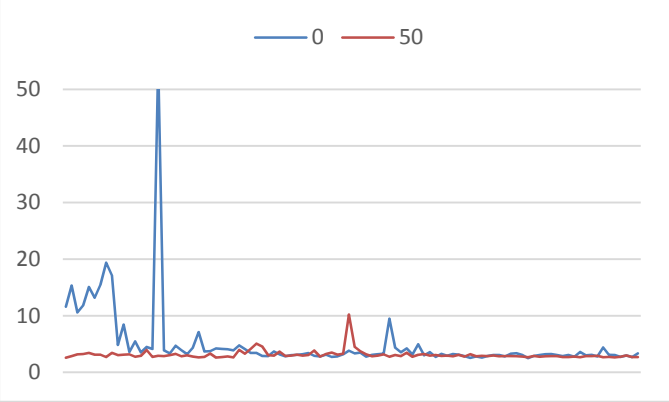

Figure 20. 0 & 50 milliseconds delay

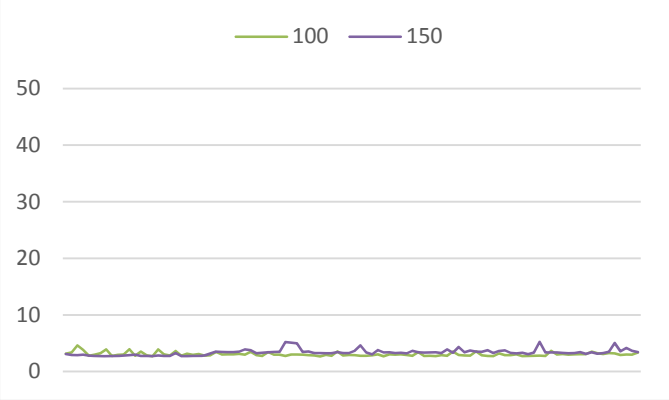

Figure 21. 100 & 150 milliseconds delay

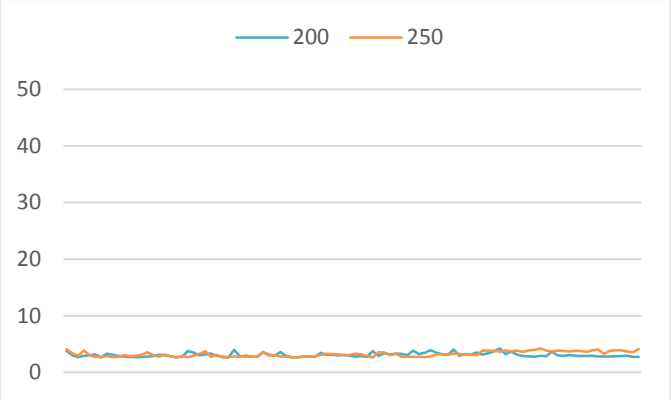

Figure 22. 200 & 250 milliseconds delay

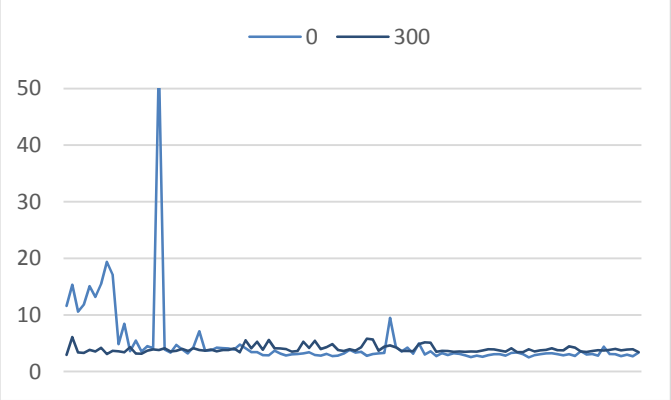

Figure 23. 0 & 300 milliseconds delay

Figures 20 – 23 show that the round-trip time on the raspberry pi machines is excellent (3-4 milliseconds). This shows that using even basic single-board-computers with SD memory cards (64GB) delivers exceptional performance.

4.3 Using Blockchain

IBM's ADEPT [18] system shows that a permission-based blockchain can be used for distribution of code and the storing of configuration data. IBM's ADEPT is based on PBFT and the use of Bluemix that serves as a Blockchain as a Service (BaaS) cloud offering. We analyzed Bluemix's performance [XX] and detected at high loads failures in receiving/processing transactions. We, therefore, decided to use a Multichain as an alternative. To test the performance of this approach a single Edison board (fig. 4.1) is used.

One virtual resource is hosted on the Edison board and issues 100 sequential write requests to the cluster-based blockchain (Multichain). 712 bytes are sent as a JSON string each time from a virtual resource. Please note that this includes 256 bytes of AES encrypted data. Again delays between 0 and 300 milliseconds are used.

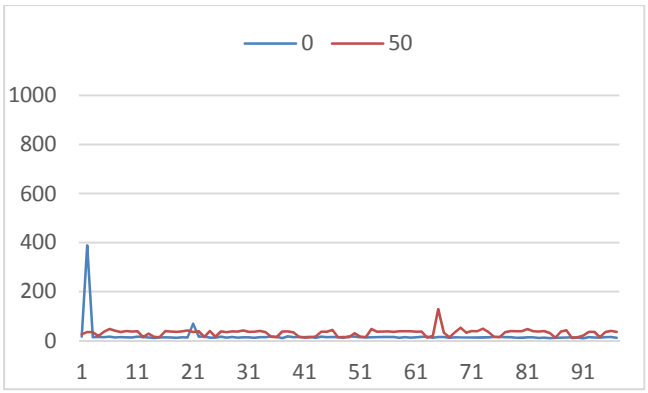

Figure 24. 0 & 50 milliseconds delay

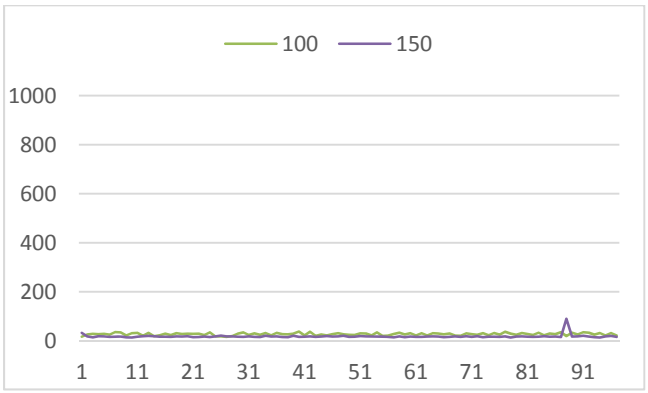

Figure 25. 100 & 150 milliseconds delay

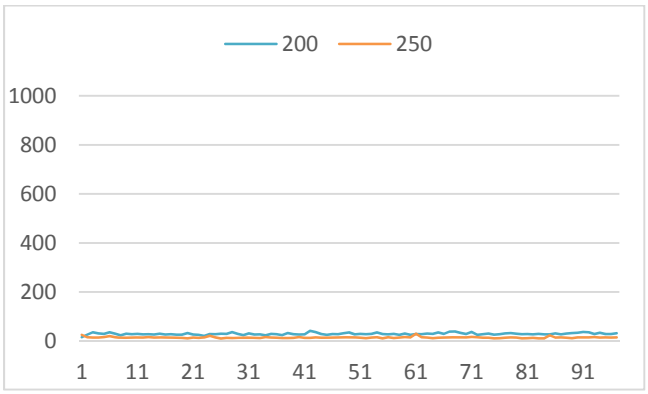

Figure 26. 200 & 250 milliseconds delay

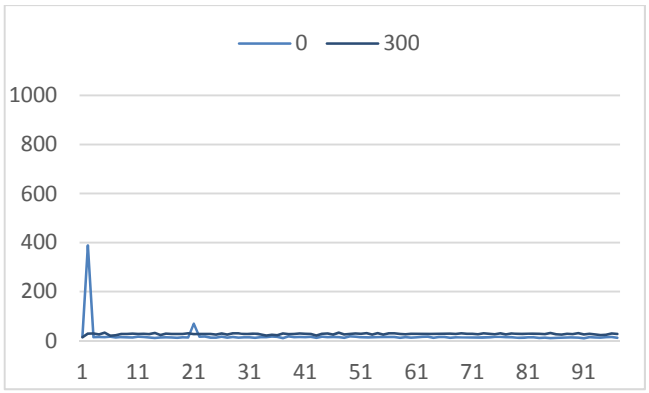

Figure 27. 0 & 300 milliseconds delay

The figures 24-27 show excellent results with roundtrip times of under 20 milliseconds. This indicates that there is nearly no overhead in communicating with the blockchain.

To test the impact of multiple virtual resources on a board in the next experiment 10 concurrent virtual resources are hosted on a single board. Each sends 100 writes to the blockchain.

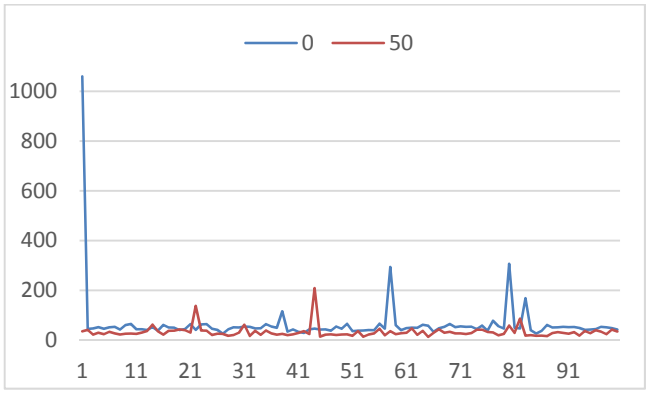

Figure 28. 0 & 50 milliseconds delay

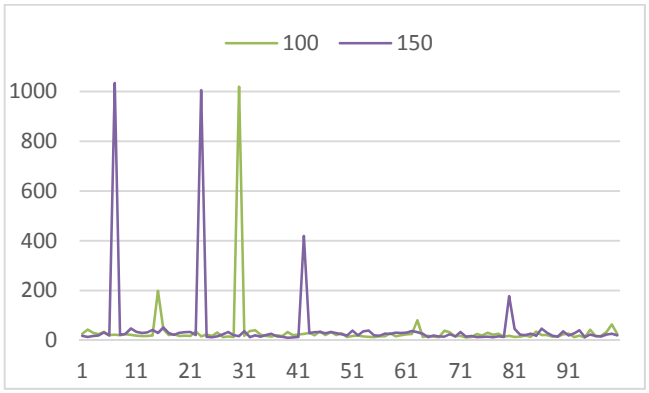

Figure 28. 100 & 150 milliseconds delay

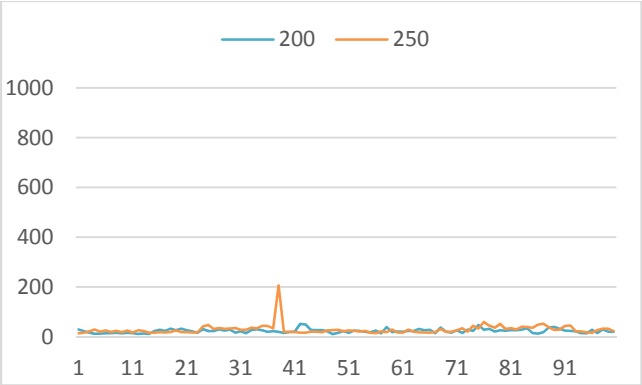

Figure 29. 200 & 250 milliseconds delay

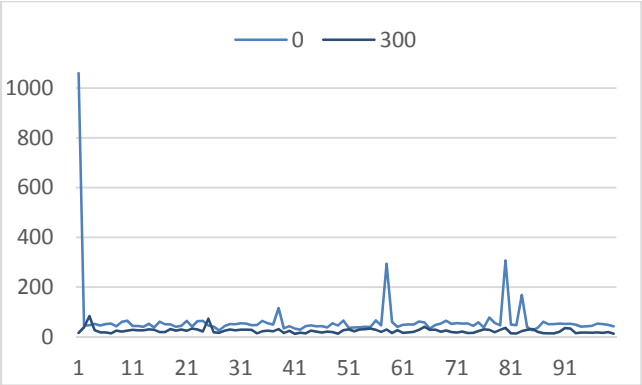

Figure 30. 0 & 300 milliseconds delay

Figures 28 – 30 show that by adding delays the performance of the blockchain writes is increased. This can be expected since the blockchain experiences with increased delays slower arrival rates which in turn result in lower loads. It is apparent from the figures that the delays help in avoiding an overload of the board's network card since the blockchain never experiences any significant load. We, therefore, conclude that the blockchain is an excellent solution for storing the state of virtual resources.

5. Summary & Future Work

Moving IoT components from the cloud onto edge hosts helps in reducing overall network traffic and thus minimizes latency. However, provisioning IoT services on the IoT edge devices presents unique challenges regarding system design and maintenance. One possible approach is the use of software-defined IoT components in the form of virtual IoT resources. This in turn allows exposing the thing/device layer and the core IoT service layer as collections of micro services that can be distributed to a wide range of hosts.

This paper presents the idea and evaluation of using virtual resources in combination with a permission-based blockchain for provisioning IoT services on edge hosts. The experiments show that a permission-based blockchain is an efficient way to store state information of virtual resources.

Our future work will focus on evaluating the impact of blockchain placement and the use of smart contracts. Since Multichain is derived from Bitcoin it implements a very basic blockchain VM. Adding smart contracts can be done by use of an oracle that executes the smart contract, or by moving to a private blockchain based on the Ethereum VM.

6. References

[1] Gubbi, Jayavardhana, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions." *Future Generation Computer Systems* 29, no. 7 (2013): 1645-1660.

[2] Sanchez, Tomas, D. C. Ranasinghe, Mark Harrison, and Duncan McFarlane. "Adding sense to the internet of things—an architecture framework for smart object systems." *Pers Ubiquitous Comput* 16, no. 3 (2012): 291-308.

[3] Rose, David. *Enchanted objects: Design, human desire, and the Internet of things*. Simon and Schuster, 2014.

[4] Doukas, Charalampos, and Ilias Maglogiannis. "Bringing IoT and cloud computing towards pervasive healthcare." In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pp. 922-926. IEEE, 2012.

[5] Biswas, Abdur Rahim, and Raffaele Giaffreda. "IoT and cloud convergence: Opportunities and challenges." In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pp. 375-376. IEEE, 2014.

[6] Bonomi, Flavio, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. "Fog computing and its role in the internet of things." In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13-16. ACM, 2012.

[7] Bonomi, Flavio, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. "Fog computing: A platform for internet of things and analytics." In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pp. 169-186. Springer International Publishing, 2014.

[8] Vaquero, Luis M., and Luis Rodero-Merino. "Finding your way in the fog: Towards a comprehensive definition of fog computing." *ACM SIGCOMM Computer Communication Review* 44, no. 5 (2014): 27-32.

[9] Grieco, Raffaella, Delfina Malandrino, and Vittorio Scarano. "SEcS: scalable edge-computing services." In *Proceedings of the 2005 ACM symposium on Applied computing*, pp. 1709-1713. ACM, 2005.

[10] Nastic, Stefan, Sanjin Sehic, Duc-Hung Le, Hong-Linh Truong, and Schahram Dustdar. "Provisioning software-defined iot cloud systems." In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pp. 288-295. IEEE, 2014.

[11] Mayra Samaniego, Ralph Deters. Hosting virtual IoT resources on edge-hosts with blockchain, IEEE CIT 2016, 4 pages.

[12] Nunes, Bruno Astuto A., Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. "A survey of software-defined networking: Past, present, and future of

programmable networks." *IEEE Communications Surveys & Tutorials* 16, no. 3 (2014): 1617-1634.

[13] Kirkpatrick, Keith. "Software-defined networking." *Communications of the ACM* 56, no. 9 (2013): 16-19.

[14] Nastic Stefan, Sanjin Sehic, Duc-Hung Le, Hong-Linh Truong, and Schahram Dustdar. "Provisioning software-defined iot cloud systems." In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pp. 288-295. IEEE, 2014.

[15] Chowdhury, NM Mosharaf Kabir, and Raouf Boutaba. "A survey of network virtualization." *Computer Networks* 54, no. 5 (2010): 862-876.

[16] Alam, Sarfraz, Mohammad MR Chowdhury, and Josef Noll. "Senaas: An event-driven sensor virtualization approach for internet of things cloud." In *Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on*, pp. 1-6. IEEE, 2010.

[17] Blockchain https://bitcoin.org/bitcoin.pdf

[18] IBM ADEPT https://www-935.ibm.com/services/multimedia/GBE03662USEN.pdf