Hosting Virtual IoT Resources on Edge-Hosts with Blockchain

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*—Current IoT systems tend to be cloud-centric which in turn introduces network latency and constrained interaction with sensors and actuators. This paper presents the idea of using restful micro-services called Virtual Resources. A Virtual Resource is a software-defined IoT management construct that enables multi-tenancy support and load distribution onto edge hosts. The paper presents a performance analysis of our Golang implementation of Virtual Resources in various settings.**

Blockchain; Virtual Resources; Edge-Computing; Fog; IoT

I. CLOUD-CENTRIC & THING-CENTRIC IOT

By connecting physical devices to the Internet, it becomes possible for them to interact with other devices, services, and applications. This Internet-of-Things (IoT) can be used to enhance the capabilities or interactions with the connected device (e.g. smart device) and/or to remotely access/control the connected devices (e.g. sensor, actuator). This, in turn, leads to the problem of how to avoid access conflicts in case multiple parties/tenants interact with the IoT network/system. In their classification of IoT, Gubbi et al. [1] distinguish between the thing-centric IoT that focusses on the devices and the cloud-centric approach in which devices are the edge components of the IoT network. In the thing-centric approach, the focus is on the enhancement of the device and the consequently richer user experiences [2,3]. In the cloud-centric approach [1,4,5] the focus shifts away from the individual things towards IoT services and applications that process large data streams emanating from large sets of things. The cloud-centric approach consists typically of three layers, namely things, cloud, and applications layer. Please note that the applications (3rd layer) and things (1st layer) are decoupled and can only interact via the middle layer (2nd layer). The middle layer exposes the IoT edge components either as abstractions (e.g. as data streams) or as virtualized resources (e.g. proxies) thus avoiding multi-tenancy issues. But the price for moving computation away from the edge into the cloud is a significant latency when engaging the things. Moving the virtual resources and abstractions closer to the edge of the IoT network is an obvious solution. However, moving part or parts of computation from the homogeneous, resource-rich, centralized cloud-computing environment towards the IoT edge devices present the following challenges:

- Supporting virtualization / abstraction on constrained & heterogeneous edge components
- Secure & safe software distribution for edge hosts
- Access control management for edge-hosted software

The remainder of the paper is structured as follows. Section 2 focusses on the concept of the virtual resource as a software-defined entity. The evaluation of the proposed edge-hosted virtual resources is presented in section 3. The paper concludes with a summary and future work in section 4.

II. SOFTWARE-DEFINED IOT & VIRTUAL RESOURCES

Software defined components and virtualization has emerged in recent years as a pattern for building IoT systems based on the success of Software-defined Networking and Network Virtualization. Software-defined Networking Software-defined networking (SDN) [6,7] 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 lead to the rise of Software-defined IoT (SD-IoT) [8]. SD-IoT uses abstraction to simplify provisioning and customization of its components. Network Virtualization [9] goes beyond SDN by focussing on the virtualization of all elements resulting in the ability to define customized virtual networks. We view this approach conceptually superior to the SD-X since it allows the definition of multiple and separate virtual networks onto of a physical one. Virtualization within IoT e.g. virtual sensors [10] is common. However, these approaches always focussed on combining or abstracting individual components nor 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 "view" on one or more IoT components. The virtual resource is a digital artifact that is linked to any other IoT component including other virtual resources. Virtual resources are therefore restful micro services that process requests by either engaging other restful services or by using their 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.The virtual resource is, therefore, a mechanism for:

- Handling multi-tenancy since each tenant uses their set of virtual resources.
- Distributing loads since these micro-services can be hosted closer to the edge

978-1-5090-4314-9/16 $25.00 © 2016 IEEE
DOI 10.1109/CIT.2016.71

- Enabling controlled low-latency access to things as well as a mechanism for distributing loads.

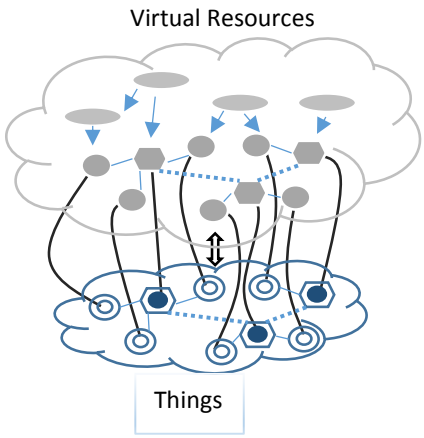

Figure 1. Virtual Resouce

Given that virtual resources are a design construct many different implementations are possible. We choose to use the language Golang and implement virtual resources as concurrent go-routines in a Golang program. Each virtual resource is a function defined in the language Golang. The source-code of all virtual resources is stored in a code repository e.g. database. At runtime, the administrator selects the virtual resources and compiles them into one or more Golang applications. The Golang functions are added to the source-code of a go host program that will be compiled for the target platform e.g. Edison. The administrator places the code onto a permission-based blockchain (e.g. Bluemix, Multichain) and informs the edge component to download the code from the blockchain. Virtual resources can be deactivated either by notifying the host program to block (and terminate) its go-function or by recompiling a new host program (without the virtual resource). Adding or changing virtual resources requires the creation of new host program.

To engage the virtual resources REST [11] was chosen over SOA [12,13]. In contrast to the SOAmodel that is based on stateless services, REST builds on the concept of stateful resources. Leonard Robinson [14] identifies within his 4-level web maturity model two patterns that have become popular within REST. The first is the basic CRUD [15] (Create Read Update Delete) approach that follows a basic data-centric style. CRUD has gained significant interest due to the natural mapping on HTTP verbs. Create, read, update and delete can be achieved via POST, GET, PUT/PATCH and DELETE. The second pattern that is less widespread is the use of hypermedia controls. Unlike the data-centric CRUD pattern, the hypermedia control pattern focuses on the use of embedding links into the responses of the request. By offering links, the server provides the client with possible next steps and ways to obtain further information. Choosing REST enforces a resource-oriented view onto the edge components (aka things). Within REST, the edge components are stateful entities and the interaction can be mapped to the CRUD operations. A particularly interesting aspect of using REST to model the thing layer is the possibility to define virtual resources on top of existing things. As shown in figure 1 the elements of the things layer are linked to their virtual counterparts called (atomic) virtual resources. Virtual resources can be combined into composite virtual resources. Composite virtual resources can again be combined with other composite resources. This, in turn, leads to a the emergence of trees with atomic virtual resources as root elements and composite virtual resources as child nodes. Each virtual resource is a RESTful microservice and also maintains its own access control list to manage access [16,17]. Cisco's CoAP (Constrained Application Protocol) [18,19] embraces the presented REST design patterns and offers a well-defined application level protocol. CoAP assumes that each CoAP message is contained in a single UDP packet (otherwise, it has to be spread over N UDP packets). The CoAP package size varies from the minimum 4 bytes (simple GET requests) to a maximum of 1024 bytes. CoAP follows the HTTP request/response interaction model between application endpoints, supports the built-in discovery of services and resources, and includes fundamental concepts of the Web such as URIs and Internet media types. By implementing the virtual resources as Golang micro-services that communicate via CoAP, it becomes possible to host them on a variety of platforms e.g. edge computing devices like Edison Arduino boards. A key challenge in the context of using edge devices is the secure deployment of code. One of the most effective ways to overcome this problem is by use of a permission-based blockchain. Code or its signature is pushed onto the blockchain, and the edge devices pull it from the blockchain. As shown wth IBM's ADEPT [20] system, it is possible to push code onto IoT devices without compromising the safety and security of the devices.

III. Evaluation

To evaluate the use of edge-hosted virtual resources, 2 Intel Edison Arduino boards are used. 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 (used for underlying Arduino board). The two boards used in the experiments are both connected to the same shared WiFi network (same access point). In the first experiment, 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 2. Edison-Arduino Edge Device.

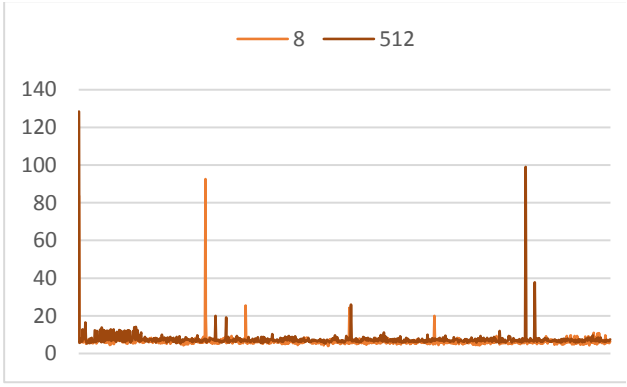

Figure 3. Sending 8 & 512 bytes between devices

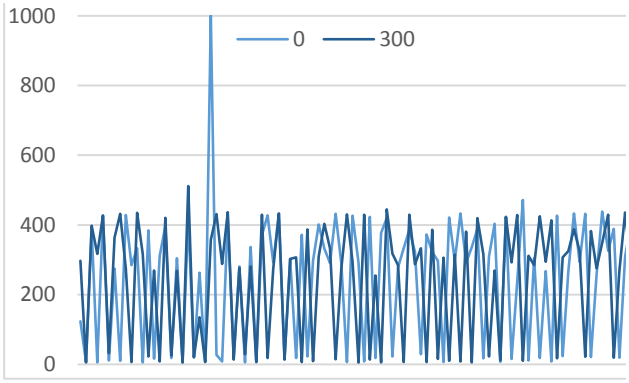

Figure 4. 0 & 300 milliseconds delay

Figure 3 shows that the performance is very good since in the 8-byte and 512-byte CoAP payload scenario. To ensure safe and secure data transmission, 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.

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 is kept at 256 bytes and each virtual resource waits 0 and 300 milliseconds before issuing the next request. AES us used to encrypt and decrypt the messages. Please note that the following figures show the results for 100 results. As can be seen in figure 4, the adding of delays has no significant impact on the performance. Compared with figure 2, the time to send the data has increased. However, it is important to note that the time for encryption and decryption is included in the sending of requests.

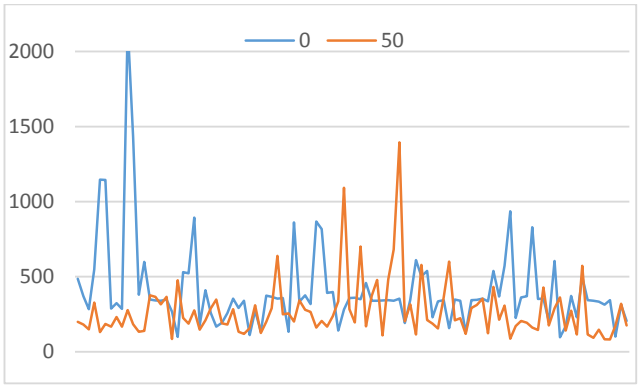

Figure 5. 0 & 50 milliseconds delay

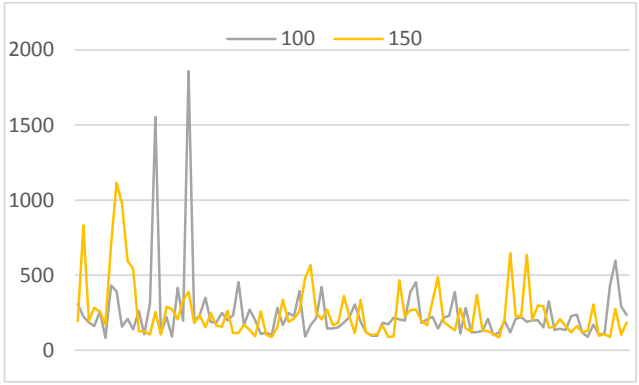

Figure 6. 100 & 150 milliseconds delay

A critical issue in the provisioning of virtual resources on edge hosts is the persistent storage of state. In the third experiment, the use of the permission-based blockchain (Bluemix) is used. To test the performance of this approach, a single Edison board is used. Again ten concurrently running virtual resources issues 100 sequential write requests to the cloud-hosted blockchain (IBM Bluemix).

712 bytes in total are sent 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. The figures 5,6,7,8 show data that fluctuates significantly. The reason for these fluctuations is primarily due to the cloud-hosted IBM Bluemix. Since it is offered as a free BaaS (blockchain as a service) it tends to experience significantly fluctuating loads. While varying greatly in its response time, it is important to note that our experiments show that permission-based blockchains can be used to store virtual resource state data.

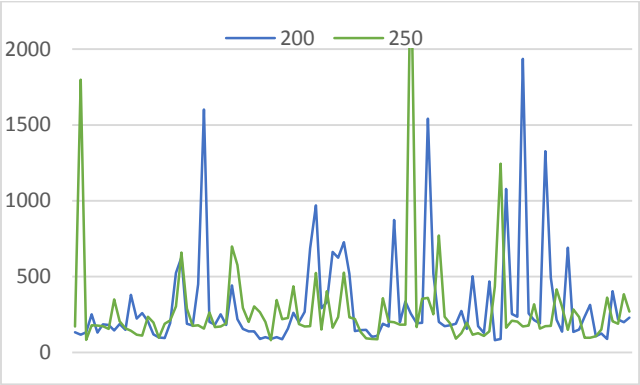

Figure 7. 200 & 250 milliseconds delay

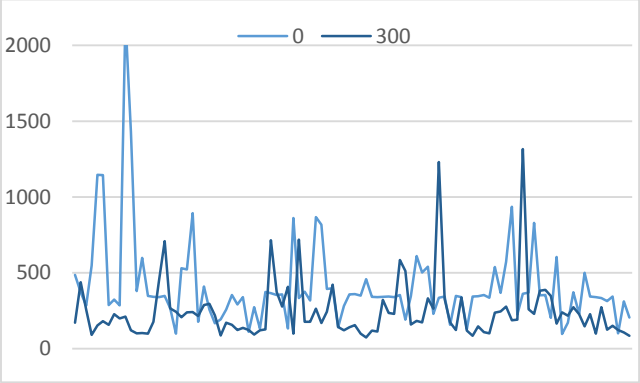

Figure 8. 0& 300 milliseconds delay

IV. SUMMARY

Current IoT systems tend to be cloud-centric which in turn introduces network latency and constrained interaction with sensors and actuators. This paper presents the idea of using restful micro-services called Virtual Resources. A Virtual Resource is a software-defined IoT management construct that enables multi-tenancy support and load distribution onto edge hosts. The evaluation of hosting the virtual resources on constrained edge-devices show that it is possible to host multiple virtual resources on a single host. Even when using AES encryption, multiple (10) virtual resources can communicate with minimal delay (around 1-1.2 seconds). Reducing the amount of virtual resources per host (e.g. to 1) leads to a delay of 20 milliseconds.

This paper also evaluates the use of cloud-hosted permission-based blockchains as an effective means for distributing code and usage as persistent data storage. We believe that using blockchains will provide a safe, secure and scalable approach for data storage of IoT devices. Future work will focus on evaluating multichain & hyperledger.

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] 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.

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

[8] 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.

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

[10] 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.

[11] Fielding R.: "Architectural Styles and the Design of Network-based Software Architectures", Dissertation University of Irvine, 2000

[12] Issarny, Valérie, Georgios Bouloukakis, Nikolaos Georgantas, and Benjamin Billet. "Revisiting Service-Oriented Architecture for the IoT: A Middleware Perspective." In *International Conference on Service-Oriented Computing*, pp. 3-17. Springer International Publishing, 2016.

[13] Pautasso, Cesare, Olaf Zimmermann, and Frank Leymann. "Restful web services vs. big'web services: making the right architectural decision." In *Proceedings of the 17th international conference on World Wide Web*, pp. 805-814. ACM, 2008.

[14] Robinson, L.: "Richardson Maturity Model"]http://martinfowler.com/articles/richardsonMaturityModel.hml

[15] CRUD: "Create Read, Update and Delete", http://en.wikipedia.org/wiki/Create,_read,_update_and_delete

[16] Hemdi, Marwah, and Ralph Deters. "Data Management in Mobile Enterprise Applications." *Procedia Computer Science* 94 (2016): 418-423.

[17] *M. Hemdi, R. Deters: "Using REST based protocol to enable ABAC within IoT systems" Proc.* IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference, 2016, 7 pages

[18] https://tools.ietf.org/html/rfc7252

[19] Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (CoAP), 2014.

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