2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

Supporting IoT Multi-Tenancy on Edge-Devices

Using Blockchain to enable edge-hosted virtualization of resources

Mayra Samaniego
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
mayra.samaniego@usask.ca

Ralph Deters
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
deters@cs.usask.ca

***Abstract*—IoT systems that support multi-tenancy tend to use cloud-hosted device/thing abstraction or virtualization. By abstracting the IoT edge components e.g. as data streams or by virtualizing the sensors/actuators, it becomes possible to avoid or resolve access conflicts. The price for this cloud-centric approach is an increased latency. This paper presents the idea of light-weight virtual resources that can be hosted on edge devices and thus offer the same abstraction/virtualization without the latency mentioned above.**

Keywords—IoT, Virtual Resource, Software-Defined IoT, Edge Computing, Multi-Tenancy

I. MULTI-TENANCY & IOT

Multi-tenancy [1] refers to an architecture that supports multiple user groups (tenants) to share one or more applications or services. To support the logical separation of tenants sharing applications or services, they must operate within different instances/contexts. Multi-tenancy has been studied within the context of data [2] and within the context of cloud-hosted services [3,4,5]. Within the context of IoT Cherrier et al. [6] identified control flow, access rights [7,8,9] and conflicting settings for actuators as critical challenges. However, as Gubbi et al. [10] point out, current thing-centric and cloud-centric IoT systems tend to ignore the multi-tenancy issues.

Thing-centric IoT focusses on the enhancement and richer interactions of a device for a user [11,12] (e.g. the umbrella that informs the owner of possible rain [12]) and consequently falls into the class of single tenant systems.

Cloud-centric approaches [10,13,14] shift the focus away from the individual things towards IoT services and applications that process large data streams emanating from large sets of things. As shown in figure one, cloud-centric IoT systems identify three layers, namely thing, service, and application. The bottom or thing layer consists of the "network(s) of things" [10] e.g. sensor networks. The bottom layer handles all thing related aspects (e.g. provisioning, maintenance, configuration, and so forth) and provides an interface to the middle or cloud layer. The middle layer hosts all core IoT services e.g. data storage, analytics, storage, etc. The IoT application e.g. surveillance, monitoring, managing, etc. reside in the third layer.

It is important to note that the application (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 as event/data streams (figure 2) to the application layer.

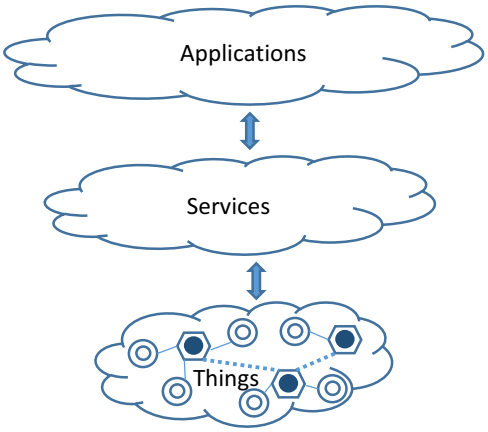

Fig 1: Three layers of cloud-centric IoT [10]

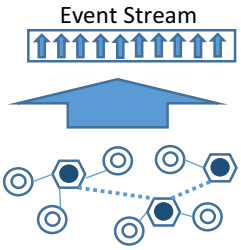

Fig 2: Event-Streams of Things

By exposing data-streams, direct manipulation of the underlying things is avoided and therefore there is no need for explicit multi-tenancy support. The obvious drawback of this design is a static sensor & data stream-centric IoT. Enabling more advanced IoT systems requires the possibility to manipulate the sensors e.g. change their configuration directly and to engage actuators. This, in turn, introduces the need to include concepts for multi-tenancy support in IoT.

978-1-5090-5880-8/16 $31.00 © 2016 IEEE
DOI 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.38

The remainder of the paper is structured as follows. Section 2 focusses on the system design and communication patterns within IoT. Section 3 presents the concept of virtual resource and is followed by the evaluation of the approach. Tha paper concludes with a summary and future work.

II. IoT System Design & Communication Patterns

While IoT systems can be achieved using a variety of system and communication patterns (e.g. agent-based) the service-oriented architecture (SOA) has been widely adopted. SOA [15,16] is based on the notion of well-defined, loosely coupled services. Using the enterprise service bus (ESB) pattern [17], it becomes possible to build reliable and highly customizable solutions. However, the SOA web service protocol stack (WS*) requires significant computational and network resources. Representational State-Transfer (REST) builds on the work of Fielding[18] and has emerged as an alternative to SOA web services. Robinson [19] identifies within his 4-level web maturity model two core REST patterns, namely

1. use of verbs (CRUD [20]) and
2. hypermedia controls.

The first (1) is the now dominant CRUD [20] (Create Read Update Delete) approach that is data-centric. Given the clear read-write semantic of the four verbs/operators, it becomes possible to leverage caching and thus improve the performance. The second (2) 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 information to obtain further information. While SOA might be applicable for the 2nd and 3rd layers in IoT [21,22] it is fairly obvious that its resource needs render it useless on the 1st layer. REST, however, offers an architectural design pattern that applies to the first layer as well as all higher layers. Choosing REST enforces a resource-oriented view onto the edge components (aka things).
In addition to the architectural design pattern, it is important to consider the communication patterns. The two most important IoT communication patterns are

- Publish-Subscribe and
- Resource-oriented communication.

Within IoT communication, MQTT [23,24] is the most often used one due to is low overhead, easy implementation, and support from all leading vendors. MQTT uses a broker/router based publish-subscribe architecture. MQTT offers decoupling with respect to time, space, and synchronization. The classical MQTT deployment follows a hub-spoke model in which nodes are linked directly to sensors and actuators. MQTT uses TCP for communicating with the message brokers. This, in turn, can lead to high communication costs. Consequently a UDP based MQTT for sensors (MQTT-S) [25] was developed. The main drawback from our point of view is the level of abstraction. MQTT is a message-oriented protocol which means that it is content agnostic and only focusses on the delivery of messages.

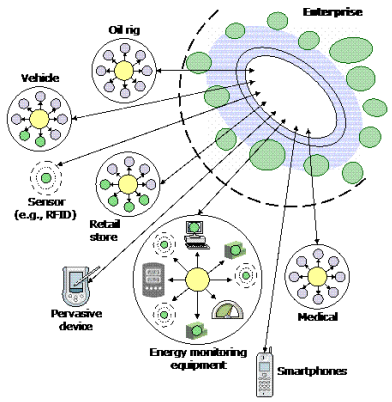

Fig 3: MQTT machine to machine example (as presented in IBM online article on MQTT [26])

The most widely used resource-oriented IoT communication protocol is Cisco's CoAP (Constrained Application Protocol). CoAP embraces the presented REST design patterns and offers a well-defined application level protocol. Cisco views CoAP as the key component in its fog-computing effort that focusses on providing a light-weight cloud closer to the edge [27,28,29].

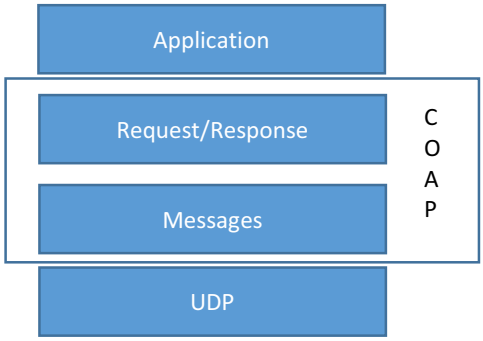

Fig 4: CoAP Protocol

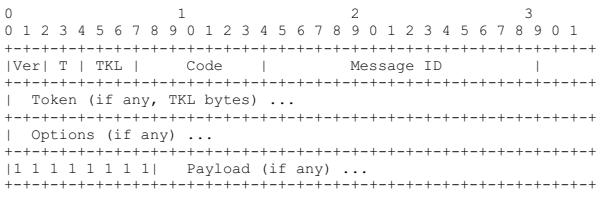

Fig 5: CoAP Message

CoAP [30,31,32,33] is a transfer protocol designed for machine-to-machine communication and optimized for use with constrained nodes and constrained networks. CoAP specifications use the UDP protocol as a transport protocol. While there are no real constraints on the size of CoAP messages/packets it is assumed 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.

By embracing the REST pattern, CoAP enforces a resource-oriented view on IoT components e.g. edge devices. This,in turn, provides a unified way to abstract and engage all elements.

III. 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 (SDN) [34, 35] 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) [36]. SD-IoT uses abstraction to simplify provisioning and customization of its components.

Network Virtualization [37] goes beyond SDN by focussing on the virtualization of all components resulting in the ability to define customized virtual networks. We view this approach conceptually promising for IoT since it allows the definition of virtual networks as overlays on a physical one. Virtualization within IoT is not a new concept e.g. virtual sensors [38]. However, these virtualization approaches always focussed on combining or abstracting individual components not defining virtual IoT systems.

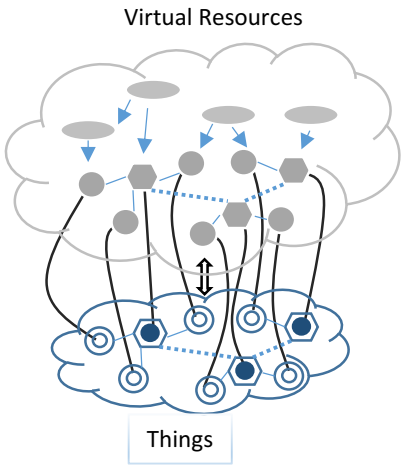

Fig 6: Virtual Resources & Things

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 provides an instance/context for accessing on one or more IoT components. As shown in figure 6, virtual resources are digital artifacts that can be linked to any other IoT component including other virtual resources. The virtual resource itself is a micro-service that exposes the interface as CoAP verbs. As shown in figure four, the virtual resource is stateful. To support multi-tenancy an access-control list (ACL) and hierarchical locks are needed. The ACL allow to define the access privileges for users/virtual resources, and the hierarchical locks provide support in case of conflicting writes.

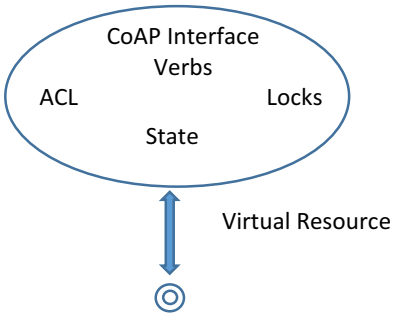

Fig 7: Virtual Resource

As mentioned, virtual resources are restful micro services that process requests by either engaging other restful services or by using their own internal state. By defining virtual resources on top of existing virtual resources, it becomes possible to create N virtual IoT systems on top of an existing one. The virtual resources represent a directed acyclical graph (DAGs) with things as the bottom elements. These virtual resources that form the DAG are engaged either by the things that push state changes to the virtual resources or requests to a virtual resource. Plese note that thing events bubble up and requests bubble down. Given that virtual resources are a design construct they can be implemented in a variety of ways. We choose to use the language Golang given the possibilities to compile code for different platforms and the single deployment feature. 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.

The key challenge in moving virtual resources onto edge hosts it the secure and safe deployment of code. A fairly new way of distributing code is the use of a permission-based Blockchain like Bluemix or Multi-Chain. By treating code like an asset it becomes possible to move it securely onto and off devices using one of the many permission-based blockchains. Depending on the used technology the retrieval of uncompiled Golang code (or pre-compiled code) is within 300 – 800 milliseconds based on available network connectivity.

Fig 8: Edge Host (Edison Arduino board)

IV. Evaluation

To evaluate the use of edge-hosted virtual resources, the following experiments were conducted (please see [39,40] fort additional information, results, and analysis:

1) Communication costs between edge-hosted virtual resources

2) State-synchronization of edge-hosted virtual resources with a blockchain

Communication costs between edge-hosted virtual resources

To evaluate the raw data transmission performance of edge-hosted virtual resources, 2 Intel Edison Arduino boards are used.

Please see Samaniego et al. [39] for additional results and analysis.

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).

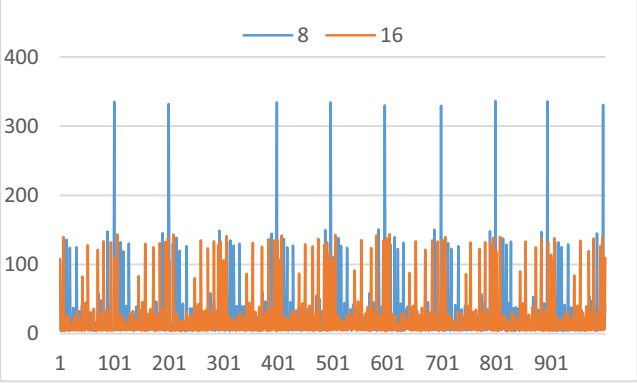

Fig 9: 8 & 16 byte payload

The two boards utilized in the experiments are both connected to the same shared WiFi network (same access point). Each board runs multiple virtual resources. 10 concurrently executing virtual resources on one board engage a single virtual resource on the other board. Each of the 10 concurrently executing virtual resources sends 100 requests. After sending the request, the virtual resource waits for the acknowledgment and only then continues with the next request.

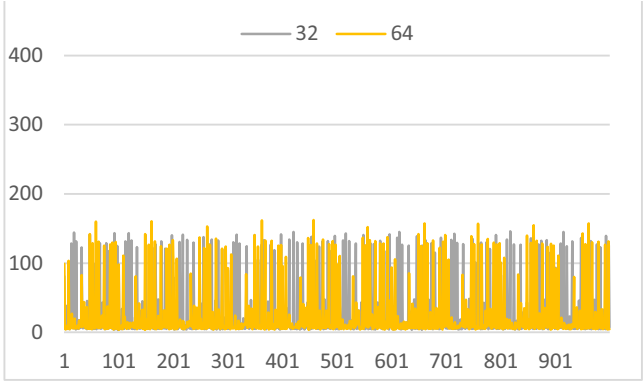

Fig 10: 32 & 64 byte payload

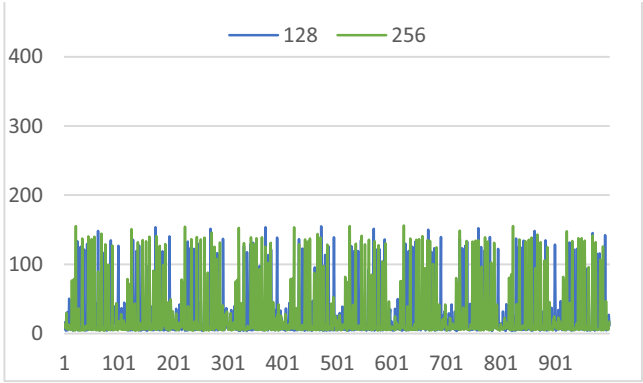

Fig 11: 128 & 256 byte payload

Please note that the 10 concurrent request-response times are shown in one continuous graph. The requests 1-100 are from virtual resource 1, the requests 101 -200 are from virtual resource two and so forth.
As can be seen, the round-trip times average 22 milliseconds with peaks around 130 milliseconds.

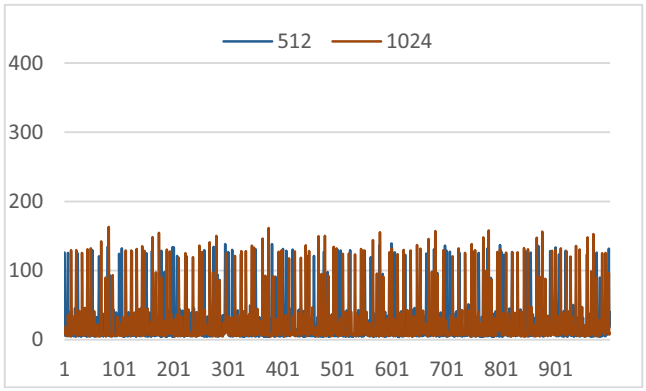

Fig 12: 512 & 1024 byte payload

The next set of experiments evaluates the use of AES encryption in the communication. Now the 10 concurrently running virtual resources send encrypted CoAP messages to the virtual resource on the other board. 256 bytes was chosen as the payload. Each experimental run uses a delay between 0 and 300 milliseconds which indicated how long each concurrently running virtual resource waits before issuing the next request.

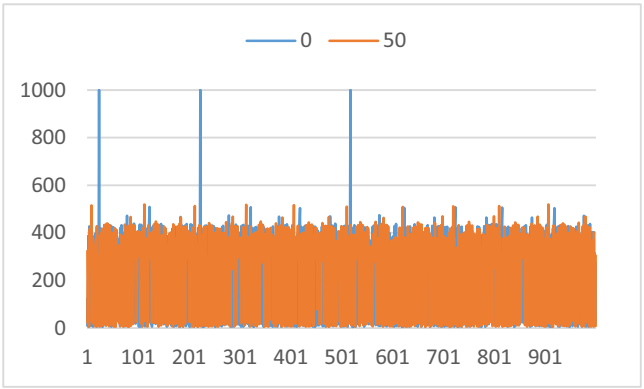

Fig 13: 0 & 50 millisecond delay

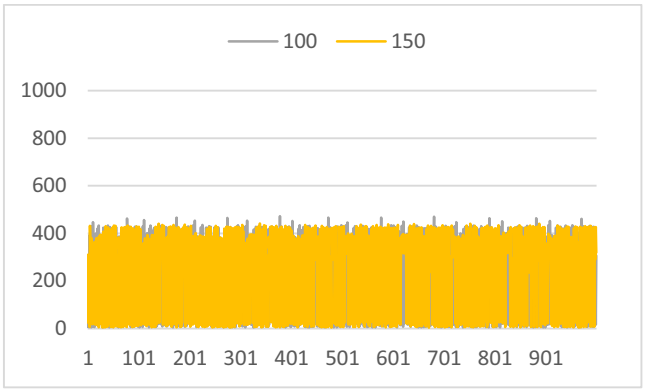

Fig 14: 100 & 150 millisecond delay

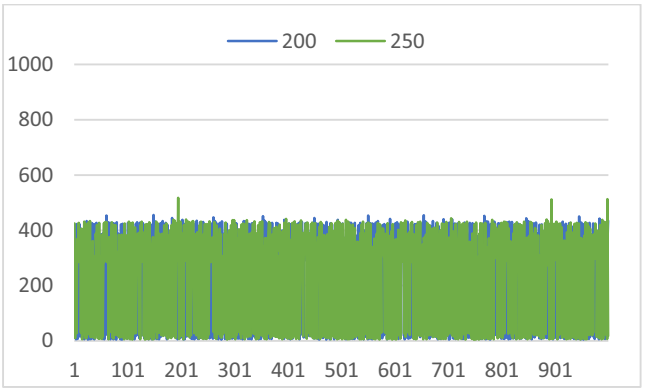

Fig 15: 200 & 250 millisecond delay

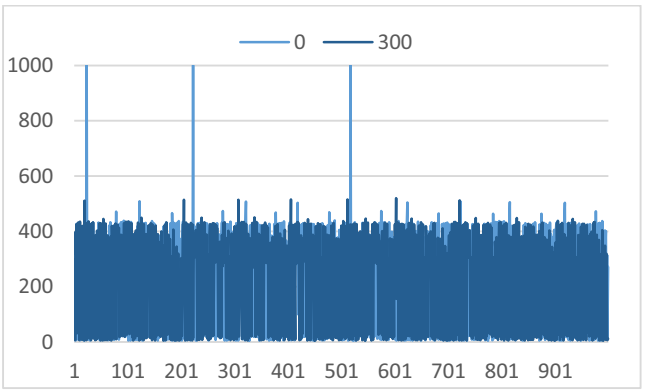

Fig 16: 0 & 300 millisecond delay

As can be seen in the figures 13-16, the adding of delays has no major impact on the performance. Compared with the figures 9-12, the time to send the data has increased. However, it is important to note that the time for encryption and decryption is now included in the sending of requests.

State-synchronization of edge-hosted virtual resources with a blockchain

To test the performance of this approach, a single Edison board is used. Again the 10 concurrently running virtual resources issues 100 sequential write requests to the cloud-hosted blockchain (IBM Bluemix).

Please see Samaniego et al. [40] for additional experimental runs and analysis.

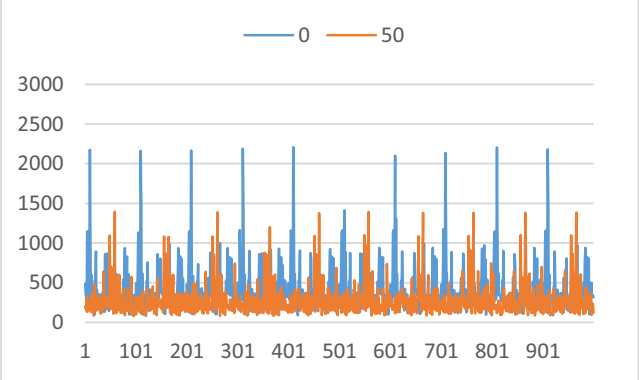

Fig 17:: 0 & 50 millisecond delay

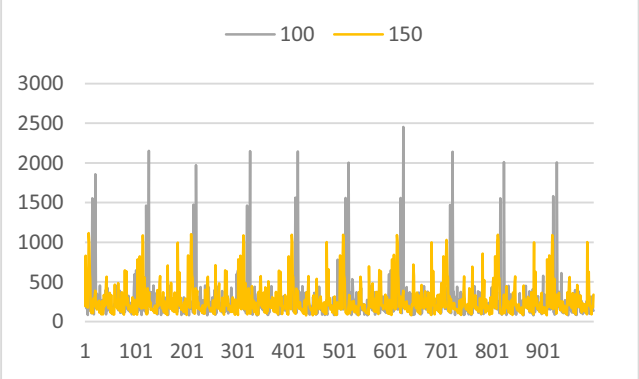

Fig 18: 100 & 150 millisecond delay

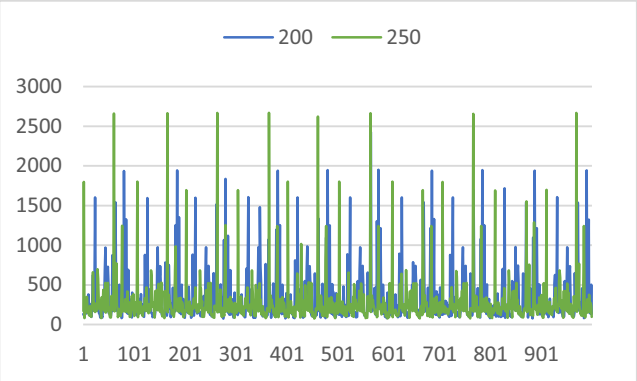

Fig 19: 200 & 250 millisecond delay

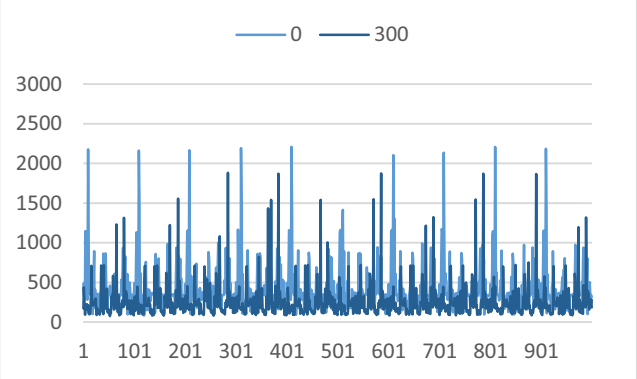

Fig 20: 0 & 300 millisecond delay

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 17-20 show data that fluctuates significantly. The reason for these fluctuations is primarily due to IBM's Bluemix. IBM's Bluemix is a 30day free BaaS (blockchain as a service) that tends to experience significantly fluctuating loads.
Please note that our experiments show that permission-based blockchains can be used to store virtual resource state data [39,40].

V. Summary & Future Work

IoT systems that support multi-tenancy tend to use cloud-hosted device/thing abstraction or virtualization. By abstracting the IoT edge components e.g. as data streams or by virtualizing the sensors/actuators, it becomes possible to **avoid or resolve access conflicts.**
However, by using hosting the abstractions/virtualizations away from the edge components in the cloud, significant latency is introduced. This paper presents the idea of light-

weight virtual resources that can be hosted on edge devices and thus offer the same abstraction/virtualization without latency.
The virtual resources are implemented as restful micro-services. A virtual resource enables the definition of views on top of existing IoT systems and therefore offer an approach for virtualizing IoT networks.
This, in turn, provides a solution to the multi-tenancy issues since each tenant can be given their own virtual IoT system. Since virtual resources are lightweight microservices, it becomes possible to offload the cloud and move computation towards the edge of the network. This, in turn, allows for richer interactions with the elements of the thing layer e.g. the sensors and actuators.The paper presents the results of an extensive evaluation of the proposed virtual resources (please see Samaniego et al. [39,40] for additional data and analysis). We show that it is possible to run virtual resources on edge devices like raspberry pi and Edison Arduino boards. Communication costs are low even when many concurrent views are executed. Use of a blockchain is shown to be an effective solution for storing data from edge hosts.

Future work will focus on the evaluation of different boards and the use of different blockchains. While Arduino boards enable fast and easy development of IoT systems, they tend to consume often too much energy.

Fig 21: TI Sensor Tag [41]

An alternative to the widely used Arduino boards is the TI Sensor Tag. Based on the CC2650 wireless MCU that according to TI uses 75% less energy than comparable platforms. Our early experiments indicate the great potential for this platform that can be linked via Bluetooth to any low-power processing node like an Edison SOC.
Besides evaluating cheaper and more energy efficient platforms for IoT we plan to focus on the fast evolving area of blockchains. Our experiments were based on the use of IBM's bluemix blockchain. IBM's approach to blockchain technology is based on using PBFT in combination with a permission system.

References

[1] Bezemer, Cor-Paul, Andy Zaidman, Bart Platzbeecker, Toine Hurkmans, and Aad'T. Hart. "Enabling multi-tenancy: An industrial experience report." In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pp. 1-8. IEEE, 2010.

[2] Jacobs, Dean, and Stefan Aulbach. "Ruminations on Multi-Tenant Databases." In *BTW*, vol. 103, pp. 514-521. 2007.

[3] Computing, Cloud. "Toward a multi-tenancy authorization system for cloud services." (2010).

[4] Guo, Chang Jie, Wei Sun, Ying Huang, Zhi Hu Wang, and Bo Gao. "A framework for native multi-tenancy application development and management." In *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, pp. 551-558. IEEE, 2007.

[5] Mietzner, Ralph, Tobias Unger, Robert Titze, and Frank Leymann. "Combining different multi-tenancy patterns in service-oriented applications." In *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*, pp. 131-140. IEEE, 2009.

[6] Cherrier, Sylvain, Zahra Movahedi, and Yacine M. Ghamri-Doudane. "Multi-tenancy in decentralised IoT." In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pp. 256-261. IEEE, 2015.

[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] Xu, Xun. "From cloud computing to cloud manufacturing." *Robotics and computer-integrated manufacturing* 28, no. 1 (2012): 75-86.

[9] Botta, Alessio, Walter De Donato, Valerio Persico, and Antonio Pescapé. "On the integration of cloud computing and internet of things." In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pp. 23-30. IEEE, 2014.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] Newcomer, Eric, and Greg Lomow. *Understanding SOA with Web services*. Addison-Wesley, 2005.

[16] Weerawarana, Sanjiva, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F. Ferguson. *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR, 2005.

[17] Schmidt, M-T., Beth Hutchison, Peter Lambros, and Rob Phippen. "The enterprise service bus: making service-oriented architecture real." *IBM Systems Journal* 44, no. 4 (2005): 781-797.

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

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

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

[21] Guinard, Dominique, Vlad Trifa, Stamatis Karnouskos, Patrik Spiess, and Domnic Savio. "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services." *IEEE transactions on Services Computing* 3, no. 3 (2010): 223-235.

[22] Spiess, Patrik, Stamatis Karnouskos, Dominique Guinard, Domnic Savio, Oliver Baecker, Luciana Moreira Sá De Souza, and Vlad Trifa. "SOA-based integration of the internet of things in enterprise services." In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pp. 968-975. IEEE, 2009.

[23] Tang, Konglong, Yong Wang, Hao Liu, Yanxiu Sheng, Xi Wang, and Zhiqiang Wei. "Design and implementation of push notification system based on the MQTT protocol." In *International Conference on Information Science and Computer Applications (ISCA 2013)*, pp. 116-119. 2013.

[24] http://www.redbooks.ibm.com/redbooks/pdfs/sg248054.pdf

[25] Hunkeler, Urs, Hong Linh Truong, and Andy Stanford-Clark. "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks." In *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*, pp. 791-798. IEEE, 2008.

[26]https://www.ibm.com/developerworks/community/blogs/aimsupport/entry/what_is_mqtt_and_how_does_it_work_with_websphere_mq?lang=en

[27] 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.

[28] 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.

[29] 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.

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

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

[32] S. Xue, R. Deters: "Resource sharing in mobile cloud-computing with CoAP", 6th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2015), 8 pages.

[33] N. Chen, H. Li, R. Deters, "Collaboration & Mobile Cloud Computing", IEEE CIC 2015, 6 pages.

[34] 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.

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

[36] 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.

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

[38] 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.

[39] M. Samaniego, R. Deters: Hosting Virtual IoT Resources on Edge-Hosts with Blockchain, IEEE CIT 2016, 4 pages.

[40] M. Samaniego, R. Deters: Using Blockchain to push Software-Defined IoT Components onto Edge Hosts, BDAW 2016, 8 pages.

[41] TI Sensor tag: http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/?INTC=SensorTag&HQS=sensortag