Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 94 (2016) 144 – 151

Procedia Computer Science

The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016)

Using Provenance and CoAP to track Requests/Responses in IoT

Emmanuel Kaku[a*], Richard .K. Lomotey[b], Ralph Deters[a],

[a]*University of Saskatchewan, Saskatoon, Canada*
[b]*Pennsylvania State University, USA*

Abstract

Until recently, not much attention has been drawn to the need to provide documentary evidence for ensuring reliability, transparency and, most importantly, tracing the source of requests/responses in the Internet of Things. The knowledge of provenance is considered as a key component in establishing the above-mentioned issues. Most research, to a large extent, focus on requesting data, which is based on user inference and decision making, by utilising provenance information. However, little or nothing has been done regarding requests and responses and, most importantly, from the machine perspective. Consequently, this paper proposes a light-weight prototype system for tracing the source of requests/responses using provenance information over CoAP in the Internet of Things. We also provide performance evaluation of the prototypic system using metrics such as response time (ms) and throughput (KB/s). Finally, findings from our experiment are presented and discussed.

Keywords: IoT; Provenance; REST; URI; Meta-data; CoAP.

1. Introduction

The use of the Internet has now been extended to encompass billions of internet-connected objects. These internet-connected objects were initially everyday things that were not considered part of the Internet. They range from cars,

* Corresponding author. Tel.: +1-639-317-7021.
E-mail address:emk508@mail.usask.ca

doi:10.1016/j.procs.2016.08.023

buildings, animals, chairs, shirts, TV's, bicycles to mention but a few. They can be tangible or intangible things with limited resources in terms of battery, bandwidth and computational power. Apart from devices such as phones, laptops, switches, routers etc., which are already computerized, these things have identities and are equipped with embedded chips, sensors, actuators and some elements of communication technologies such as Wifi, Bluetooth, etc.). Having the above technologies has made them intelligent, enabling them to sense, self-configure, self-maintain, and interact with one another. As a result, they are able to generate huge amount of data[1,23]. These data are pushed to the cloud for massive analysis which ultimately translates to useful information that has the potential of transforming human lives in so many ways, change business strategies and models and, most importantly, generate economic revenue for the society. This paradigm shift in computing is the Internet of Things (IoT). Some key driving elements of this shift comprise identification, sensing, communication, humans, cloud-computing and actuating technologies[23]. IoT is presently all around us: in our homes, cars, and even on our physical bodies. It is currently connecting citizens to their cities, linking patients to health services and bringing clients closer to companies. With this shift pervading all spheres of human life, Gartner Inc. and Cisco[20, 21] predicted that the total number of devices connected to the Internet would hit 20 to 50 billion by 2020. IDC, a top-notch research institution, also predicted that 30 billion connected devices will generate revenue of 1.7 trillion dollars for the IoT ecosystem by 2020[22]. Such astounding numbers from these predictions indicate the significant impact that IoT will make in our lives in the near future.

Additionally, new exciting possibilities have surfaced in both research and business domains including manufacturing, transportation, smart cities, green energy, e-health, retail and personalized user applications [2,3,4]. The application of this fast-growing technology is undoubtedly beneficial. However, one major problem that seems ignored is the issue of being able to track requests/responses in the midst of the interaction among these devices, which are able to request and provide services as well as data to each other. Again, many have envisioned some issues such as transparency, reliability and confidentiality of data and privacy[5,6,8,1]. Moreover, most research application of provenance in IoT, to a large extent, have focused on requesting data. The provenance data decision making have also been from user perspective rather than intelligent machines. Some researchers have proposed integration of provenance in the Internet of Things in order to solve the above-mentioned issues[6]. Provenance, as defined by W3C "*is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness*"[11]. This implies that having documentary evidence about the history of requests/responses, together with its origins and its processes, enable good data access, transparency and decision making with regards to tracing the source of requests/responses. This paper, however, seeks to deviate from the traditional norm by exploring how machines can infer about requests/responses in the Internet of Things. For instance, if a request (GET/PUT/POST/DELETE) is made by a mobile device to change the state of a thing within IoT, an intelligent machine, such as raspberry pie, might want to trace the lifecycle of such request, its timestamp, its traversal path, to its current state associated with its response. To achieve this, this paper proposes a provenance-based light-weight prototype, which utilizes CoAP, a machine to machine communication protocol meant for small, resource-constraint devices. Furthermore, we provide performance evaluation of this prototype against metrics such as response time (ms) and throughput (KB/s).

The rest sections of this paper are structured as follows. Section 2 discusses related works. Section 3 describes the details of the system and design and the performance analysis conducted. Section 4 discusses the limitations. Finally, Section 5 focuses on our findings and future works.

2. Related Work

This section provides related works regarding provenance in IoT. Provenance, to a large extent, has been studied and applied in many different domains such as databases[12], cloud computing[13], scientific workflows[14], grid and distributed computing[15]. Most research conducted focus on tracing the origin of data with the aim of establishing the authenticity, trust and the quality of data. Moreover, provenance still continues to be one of the major issues emanating from the concept of the Internet of Things. Indeed, some amount of research has been done in the realm of the Internet of Things[16,17]. Data Provenance in the framework of the Internet of Things requires an extension to include who, timestamp and time periods of processes on data in addition to why, where and when. Bauer et al, 2013[6], proposed a conceptual architectural model by providing IoT connection points to data provenance which comprised of an Information Model from the IoT infrastructure model serving as an infrastructural interface enabling the incorporation of data provenance. To connect the two concepts, the IoT architecture was used as the basis for the common architecture. Furthermore, it was extended to include a provenance event handling component which handled provenance information using algorithms for collection, verification, categorization and selection. Again, they proposed requirements for provenance needed to be adhered to during the process of information collection through to the retrieval stage. According to them, there is still need for more storage capabilities and in addition to that, authentication mechanisms for security and trustworthiness on collected information and accessing users also have to be integrated into a common architecture.

Eduardo et al[5,18] proposed a lightweight semantic model and a prototypic mobile-enabled software which captured IoT device information, their provenance, their capabilities and use in order for end users to infer about them. They investigated the Trusted Tiny Things project which consisted two case studies; the first case was a deployment of a passive Near Field Communication tag implemented by Aberdeenshire council to access real-time timetable information about bus stop with the use of a smartphone whereas the second case study examines in-car black boxes which tracks and captures information about diving behavior, vehicle location with a range of sensors and further transmit it to an insurance company. They pointed out key questions pertaining to user privacy. In the second scenario for instance, the authors raised questions such as; "what kind of data is being recorded? When and where is the data being sent? Who is using the data? Is the data being sent to other third party companies? For what purpose? They assert that using provenance in both scenarios had the possibility of enabling users understand the lifecycle of the data and the purpose for which it was intended for. They proposed a demonstrator application to enable them evaluate their approach. This was based on the two case studies. The result of their key findings show that, on capturing sensor data it was possible to reason about personal information and moreover, establishing a connection with sensor observations to data sources could enable inference of useful information on users performed actions. Similarly, Kolozali et al.[17] proposed a stream tagging framework for real time IoT data as a way of supporting dynamic incorporation into the web space. They suggested addition of meta-data to IoT data stream and explained that the framework comprised four main parts namely; virtualization, middleware, reliable processing and semantic labeling. Virtualization handles access to diverse data sources, middleware deals with the communication by using an Advanced Message Queue Protocol. In addition, the authors' state that a reliable information module handles the extraction of dynamic and diverse sources of data and performs processing and aggregation but takes into account accuracy and trust. Another light information model used for providing summary and reliable IoT data stream was proposed. According to them this information model included stream annotation ontology, quality of service as well as quality of Information and provenance. To evaluate the performance of the framework, different data stream, raw and amassed were tested against their annotated data. The key findings of their research showed that the framework performance in all cases recorded increase in 99.4% and 96.2% with data size and average message exchange time used as performance metrics Furthermore, they suggested adding a wide set of data streams and making use of computer network which would enable the capturing of real-time road traffic to perform analysis on large number of users. Additionally, they suggested further work on stream annotation ontology that will provide effective coverage of stream analysis techniques mostly used by researchers and to generalize the model in order to blend them with research tools in existence. In contrast to the above methods Jie et al[3] identifies resource scheduling issues in distributed and parallel computing environment and proposes a scheduling algorithm based on provenance for logistic chain in IoT. This provenance based algorithm takes into consideration user's appraisal and assigns jobs to effective providers with good quality of service. To test the effectiveness of this provenance based algorithm, a simulation was performed with three test cases which provided excellent results. According to the author's, ongoing work comprises an information service that will provide detailed information on jobs and providers data, a workflow engine capable of managing jobs assigned to several providers and a tool to establish reliability and coherence of diverse user feedback. They further touched on the need for efficient management and storage capabilities of provenance information which Bauer et al[6] pointed out. Also, Qiannan et al, 2013[2] similarly propose a smart sensor data collection strategy as well as algorithms to identify and trace infected food in IoT food supply chain. According to them Self-adaptive Dynamic Partition Sampling (SDPS) Strategy enable the efficient and intelligent collection and management of data emanating from sensors. Infected sources are identified and potentially infected food in the IoT food supply chain is eliminated using their proposed tracing and backtracking algorithms discussed in their work. Qiannan et al, 2013[2], evaluated their proposed system through simulation which showed that SDPS could a trace an accuracy of 97.8% with small average of sample percentage relative to the traditional sampling methods.

Most works including the above related works to a large extent focus on data and user perspective (response), where end users read and infer about data. However, little or nothing has been done regarding requests specifically on the part of devices. In view of this, we explore a novel approach by providing machines (devices) with the ability to infer and make decision as to where request/response came from based on provenance information in the Internet of Things.

3. System Design and Setup

In this section we present an overview of the System Architecture, System Setup, how the experiment was conducted, data collection and the results of the performance analysis.

3.1. System Architecture Overview

The system comprise of five components, namely; client, agent, server, database and the network. The client in this case can be computers, mobile phones, tablets etc. The agent can be computerized device such as Raspberry Pie, Arduino etc. The server provides resources. It can be a personal computer, workstation or high end powered computer. In fig.1 the proposed system architecture uses a standard protocol called CoAP for communication between and among devices whereas the devices connect to the database using the HTTP protocol to register their metadata information. It is a standard developed the Internet Engineering Task Force (IEFT), it provides features that go beyond HTTP. CoAP is designed for devices with constrained bandwidth, energy and computation. It runs on a UDP protocol but it is able to interoperate with the HTTP which enable integration with the web whiles maintaining support for multicast, reducing overhead and providing simplicity for such constrained environments. Additionally, just like HTTP, CoAP provides support for web URIs and also enables built-in support for resource discovery as well as enabling different types of data formats such as XML, JSON, plain text, html to mention but a few. It has a compact message encoded in a binary format and comprise of the version, type, messageid, code etc. to mention but a few in its header. Following the header, token and option, is the payload, which is optionally used. Furthermore, it is has a maximum packet size ranging from 4 to 1024 bytes[19, 28]. It is based on REST[29] and operates on a request/response communication model. In this model, client sends request to servers, and servers provide reply back with responses. The client uses GET, PUT, POST and DELETE in order to access or change the state of resources[29]. Metadata refers to information that describes entities, such as the clients, agents and servers together with processes associated with them. For instance, requesting a resource and receiving a response by entities, is termed process in this context. In other words, the life cycle of requests/responses associated with these individual entities including clients, agents and servers is what makes up provenance information. Basically, in our system the client creates a CoAP request, store along with its provenance information (name, URI, date, to, method) in CouchDB, in document (request table) to access or change the state of a resource that is hosted on a server. The client can be any device such as mobile phone, tablet, computer etc. The client then logs into the request table, record its provenance information (URI, name, time etc.) along with whom the request is meant for in a database. The agent then intercepts the CoAP request, further log into the request table and then records its metadata information (URI, name, time etc.) together with the destination device (URI) of which the request is meant for. In addition to this, the agent forwards the CoAP request to the server. Finally, the server receives the CoAP request and also records its provenance information (URI, name, time etc.) in the request table of the database together with the sender's URI. Furthermore, the server then record in a separate response tables its provenance information (URI, name, time etc.) in the database together with the URI of device (agent) who sent the request on behalf of the client. The agent receives the CoAP response, log into the response table its provenance information (URI, name, time etc.) together with the recipient's URI and forwards the response to the client device. The client finally, receives the CoAP response and then log in its provenance information (URI, name, time etc.) together with the URI of the device (agent) that sent the response. In our research, the URI is sent via the CoAP payload which is used as a unique identifier for each IoT device in our prototype. Based on the captured information, which forms part of the life cycle of the IoT devices (provenance), they make use of the digitally captured information in order to trace the sources of requests/responses in the light-weight prototype system. The diagram in Fig.3 shows the interaction between the IoT devices in our prototypic system and how provenance information is captured.

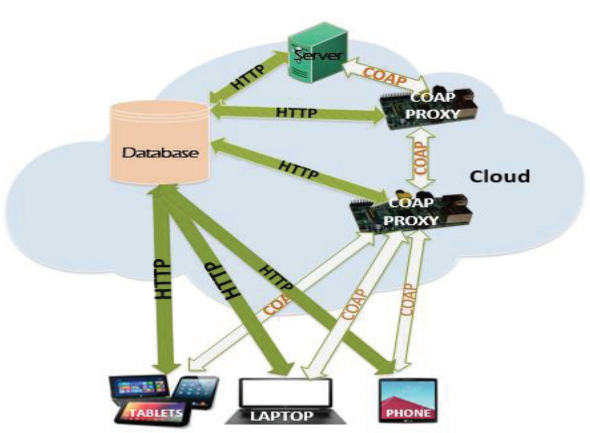

Fig. 1. proposed system architecture

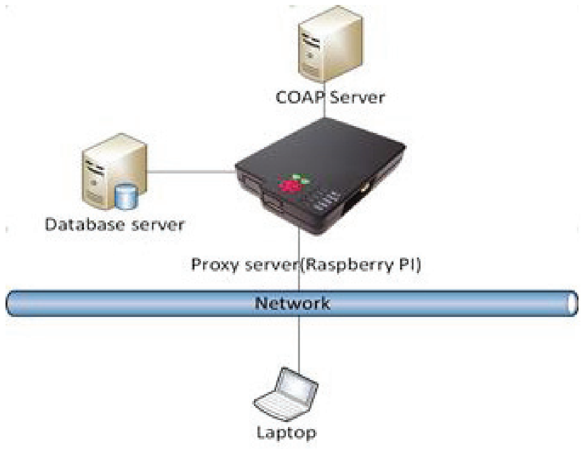

Fig. 2. design/setup

3.2. System Setup

We describe in this subsection how our system was setup. The setup consists of hardware and software components. The hardware consists of laptop, desktop and raspberry pie. In the architecture, the server is a Lenovo M series Intel Core(TM) i7-3770 with speed of 3.40 GHz and a RAM of 32GB. The agent is a Raspberry pie with 800 MHz of speed and a RAM of 1GB whereas the client is a laptop with an Intel Core(TM) i7-3612QM, 2.10GHz speed and RAM of 8 GB. The client and server run windows 8 operating system whereas the raspberry pie 2 (agent) runs on Linux. The application software written in Golang is an open source programming language developed by google. This prototype system was built from an existing package/library written by Dustin Sallings which provides a CoAP client and server implementation[5]. Although the application is installed on each machine, they all perform different functionalities based on the role they play. The server hosts a free open source NoSQL database called CouchDB. This system is extensible since it can support multiple NoSQL databases as well as multiple agents. Moreover, for communication to occur among the devices a network connection was established using Ethernet technology. The server, raspberry and the client connects together with an Ethernet cable via a switch. It should be noted here that wireless technology can be used as well. All the devices are connected via a network. The lightweight prototype of the proposed system architecture, the design and Setup is shown in fig.1 and fig.2 respectively.

3.3. Experiment and Data Collection

This subsection provides an understanding of how the experiment was conducted, and how the data was collected and aggregated. In the experimental phase, a performance application was developed and a sample of payload sizes comprising 8, 16, 32, 64, 128, 256, 512 and 1024 bytes were selected. These payload sizes represent unique URI naming scheme used to identify each device in our prototype system. The application was made to run on the client (laptop) and the experiment performed 100 times for each payload size. To sample fairly good data set from this varying payload sizes some important data metrics such as response time (ms) and throughput (KB/s) were identified. The response time refers to the time taken for a client to send CoAP message to the agent, from the agent to the server and vice versa. Also, for the response time, the time taken for each device to write to a database using HTTP, a web transfer protocol, is taken into consideration. The response time is measured in milliseconds (ms). The throughput indicated above refers to the amount of CoAP data sent over a network per second and it is measured in Kilo Bytes per second (KB/s). During the experiment, factors such as network connectivity, number of devices as well as the time taken to write to the database were taken into account to better produce a reasonable result from the experiment. After the test was conducted, the response time data as well as throughput data was collected as csv format in a log file. All data captured on response times were summed up and averaged. The same was done for the throughput data.

3.4. Performance Analysis

In this section, performance analysis is done based on the experiment and data collected in subsection C from section III. To better understand the performance of the prototypic system, some metrics indicated below were identified:

- Payload sizes
- Response/Request time interval (milliseconds)
- Throughput (Kilobyte per second)

The payload sizes are selected from a range of 8, 16, 32, 64, 128, 256, 512 and 1024 in bytes. The messages could vary in length and the reason for choosing these different sizes of the payload is to represent the varying length of the messages. The experimental setup, involves a client (laptop), server (desktop) and an agent (Raspberry Pie - a small computer with constrained computation, memory and bandwidth). As explained in section 3 subsection 3.1, the client sends a request and identifies itself with a Uniform Resource Locator (URI) in its payload and further registers its metadata information such as client's name, and method etc. in the database.

Similarly, the agent gets the request from the client (URI), identifies itself with its URI in the payload and further creates a request on behalf of the client to the server (URI) and vice versa. The experiment was conducted hundred (100) times to obtain the average data of both response time and throughput. All the processes, excluding our application running on each of the experimental device, were halted. Ethernet was used for the network connection. After performing the experiment, the data generated was plotted in a graph Fig. 4 and Fig. 5 respectively.

In Fig. 4, the x-axis (horizontal) represents the sizes of the payload from 8 to 1024 bytes with each size being doubled, whereas y-axis (vertical) in Fig. 4 represents the response time in milliseconds. We infer that the response time falls in the range of 18.7 ms to 24.4 ms. Although there is a bit of fluctuation in the response time it shows that the response time increases steadily along with increase in payload size. The fluctuation could be caused by writing provenance information into the database, network bandwidth competition and some background processes on running on the devices. The writing of provenance meta-data is via HTTP which uses the same network and as a result competes for bandwidth. Another factor is background processes. On each device, we terminated as many processes as we could but it is worth noting that some system processes were unable to terminate. Although, the response time is very fast, around 21ms, any effect from those processes easily creates those fluctuations. On the other side, the throughput of the y-axis in Fig. 5 approximately doubles as payload doubles in size. Because the response time stays in a relative level, the doubled throughput is as a result of the doubled payload size.

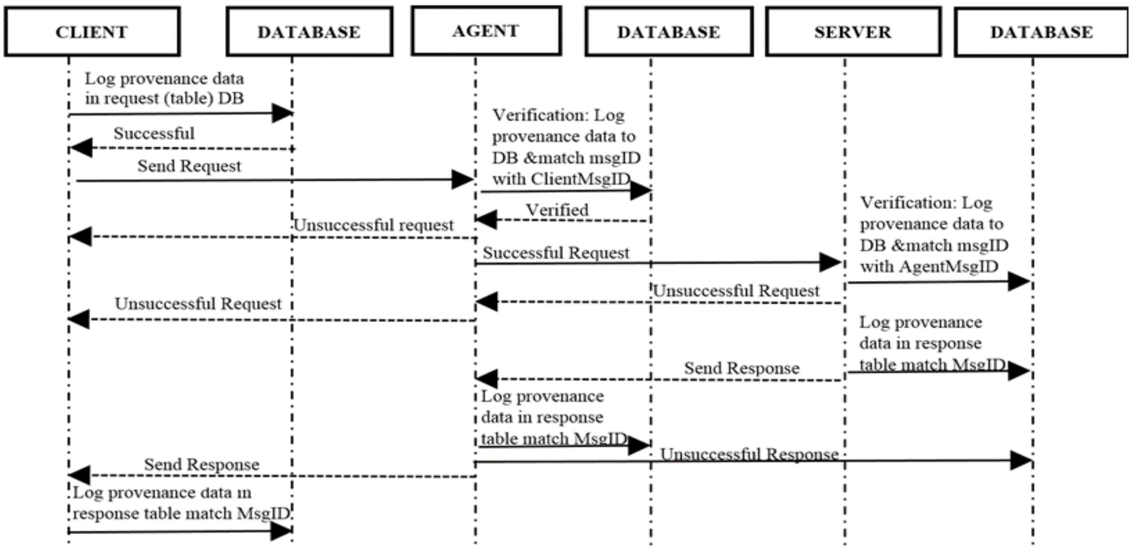
CLIENT
DATABASE
AGENT
DATABASE
SERVER
DATABASE
Log provenance data in request (table) DB
Successful
Send Request
Verification: Log provenance data to DB &match msgID with ClientMsgID
Verified
Unsuccessful request
Successful Request
Verification: Log provenance data to DB &match msgID with AgentMsgID
Unsuccessful Request
Unsuccessful Request
Log provenance data in response table match MsgID
Send Response
Log provenance data in response table match MsgID
Unsuccessful Response
Send Response
Log provenance data in response table match MsgID

Fig. 3. a sequence diagram showing the interaction in the prototypic system using provenance information

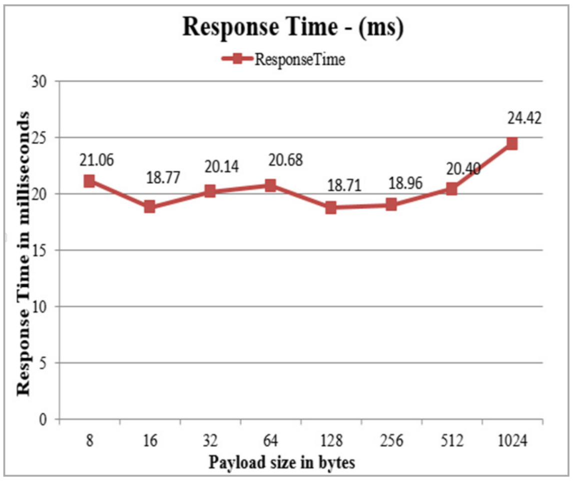

Fig. 4. average prototype response time (ms)

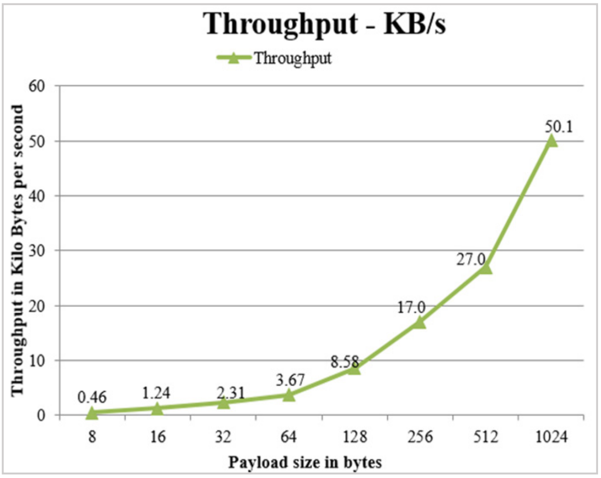

Fig. 5. average prototype throughput (KB/s)

4. Limitation

The limitation of our study is that the experiment was conducted in a localized environment. However, in future work, we look forward to carrying it out using cloud servers.

5. Conclusion and Future Work

As devices, systems, things equipped with computing chips, software's and sensors interface with the internet, our lives will be impacted in one way or the other, societies will be influenced, our business models will be impacted, and economic revenues will be generated. These things mentioned above have the capability to connect, transmit and request for services between and among themselves. This creates new opportunities for the next generation of applications and services that were initially impossible. In the midst of all this excitement, comes a challenge with tracing the source of request/response, of which the knowledge of provenance becomes a key to solving. In this paper, we propose a prototypic system that provides machines with the capability of tracking the request/response in the Internet of Things. In our approach, a provenance–based prototype was proposed to provide transparency and reliability in order to enable able machines make inference about the source and the chronological history of requests/responses. In our prototype we made use of the Constrained Application Protocol, a light weight protocol used by constrained devices for communications in the Internet of Things. We observe in our experiment that, the payload message with the highest payload size, gives a much higher throughput of 50.1KB/s. Also, we observed a 21ms fast response time but spotted a slight fluctuation as it steadily rose along with the payload size. In future studies, we look forward to testing our porotype in a production environment. Also, scaling the number of devices and the provision of visualization regarding the trace of request/response would be considered and investigated.

References

1. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol.10, no. 7, pp. 1497–1516, 2012.
2. Q. Zhang, T. Huang, Y. Zhu, and M. Qiu, "A Case Study of Sensor Data Collection and Analysis in Smart City: Provenance in Smart Food Supply Chain," *Int. J. Distrib. Sens. Networks*, 2013.
3. J. Yin, J. Li, and P. Ke, "A provenance based scheduling algorithm for logistics chain in {IOT}," *2013 6th {International} {Conference} {Information} {Management}, {Innovation} {Management} {Industrial} {Engineering}*, vol. 1, pp. 324–327, 2013.
4. R. Moore-Colyer, "Microsoft showcases real-world IoT uses in oil and gas, healthcare and smart cities," 2015. [Online]. Available:http://www.v3.co.uk/v3uk/news/2437378/microsoft-showcases-real-world-iot-uses-in-oil-and-gas-healthcare-and-smart-cities. [Accessed: 27-Dec-2015].
5. E. Pignotti and P. Edwards, "Trusted tiny things: making the internet of things more transparent to users," *ASPI '13 Proc. Int. Work. Adapt. Secur.*, 2013.
6. S. Bauer, "Data Provenance in the Internet of Things," 2013.
7. Giusto; D.; Iera; A.; Morabito; G.; Atzori; L. (Eds.), "The Internet of Things," in *20th Tyrrhenian Workshop on Digital Communications*, 2010.
8. J. a Stankovic, "Research Directions for the Internet of Things," *Internet Things Journal, IEEE*, vol. 1, no. 1, pp. 3–9, 2014.
9. A. Kanuparthi, R. Karri, and S. Addepalli, "Hardware and embedded security in the context of internet of things," *Proc. 2013 ACM Work. Secur. Priv. dependability cyber Veh. - CyCAR '13*, pp. 61–64, 2013.
10. W. W. W. Consortium, "W3C Consortium Recommendation," 2013. [Online]. Available: http://www.w3.org/TR/2013/REC-prov-dm-20130430/. [Accessed: 20-Dec-2015].
11. Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *Sigmod*, vol. 34, no. 3, p. 31, 2005.
12. P. Buneman, A. Chapman, and J. Cheney, "Provenance management in curated databases," *Proc. 2006 ACM SIGMOD Int. Conf. Manag. data SIGMOD 06*, vol. 1, no. c, pp. 539–550, 2006.
13. I. Abbadi and J. Lyle, "Challenges for provenance in cloud computing," *... Pract. Proven. (TaPP'11). USENIX ...*, 2011.
14. R. BOSE, "Lineage Retrieval for Scienti c Data Processing: A Survey," *Computing*, vol. 37, no. 1, pp. 1–28, 2005.
15. D. Zhao, C. Shou, T. Maliky, and I. Raicu, "Distributed data provenance for large-scale data-intensive computing," *Proc. - IEEE Int. Conf. Clust. Comput. ICCC*, pp. 8–10, 2013.
16. J. Yin, X. Zhang, Q. Lu, and C. Xin, "IOT Based Provenance Platform for Vegetables Supplied to Hong Kong," pp. 591–596, 2012.
17. S. Kolozali, M. Bermudez-Edo, D. Puschmann, F. Ganz, and P. Barnaghi, "A Knowledge-Based Approach for Real-Time IoT Data Stream Annotation and Processing," *Internet Things (iThings), 2014 IEEE Int. Conf. on, Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Soc. Comput. IEEE*, pp. 215–222, 2014.
18. E.Pignotti and P. Edwards, "Trusted Tiny Things Project. [Online]. Available: http://t3.abdn.ac.uk/. [Accessed: 23-Dec-2015].
19. Xue, Steven, Richard K. Lomotey, and Ralph Deters. "Enabling Sensor Data Exchanges in Unstable Mobile Architectures." In Mobile Services (MS), 2015 IEEE International Conference on, pp. 391-398. IEEE, 2015.
20. Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. CISCO white paper, 1, 1-11.
21. Gartner Press Release: Gartner Say 6.4 Billion Connected "Things" Will be in use in 2016, Up 30 Percent from 2015 Retrieved April 20, 2016 from Gartner, Inc.: http://www.gartner.com/newsroom/id/3165317.
22. IDC. (2015): Explosive Internet of Things Spending to Reach $1.7 Trillion in 2020, According to IDC. Retrieved April 20, 2016 from IDC.
23. Becker, A., Sénéclauye, G., Purswani, P., & Karekar, S. (2012). Internet of Things. *Atos White Paper*.
24. Toby.J. MQTTT and CoAP, IoT Protocols, Eclipse Newsletter. Retrieved April 20, 2016 from Eclipse Foundation.
25. Github. Implementation of CoAP in go Retrieved April, 06 from Github: https://github.com/dustin/go-coap.
26. Miller, E. (1998). An introduction to the resource description framework. Bulletin of the American Society for Information Science and Technology, 25(1), 15-19.
27. Shelby, Z., Hartke, K., & Bormann, C. (2014). The constrained application protocol (CoAP).
28. Toby.J. MQTTT and CoAP, IoT Protocols, Eclipse Newsletter. Retrieved April 20, 2016 from Eclipse Foundation.
29. Fielding, Roy Thomas. *Architectural styles and the design of network-based software architectures*. Diss. University of California, Irvine, 2000.