Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2018) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018)

An Information-centric Approach for Slice Monitoring from Edge Devices to Clouds

Binh Minh Nguyen[a,*], Huan Phan[a], Duong Quang Ha[a], Giang Nguyen[b]

[a] *School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam*
[b] *Institue of Informatics, Slovak Academy of Sciences, Bratislava 845 07, Slovakia*

Abstract

Internet of Things (IoT) has enabled physical devices and virtual objects to be connected to share data, coordinate, and automatically make smart decisions to server people. Recently, many IoT resource slicing studies that allow managing devices, IoT platforms, network functions, and clouds under a single unified programming interface have been proposed. Although they helped IoT developers to create IoT services more easily, the efforts still have not dealt with the monitoring problem for the slice components. This could cause an issue: thing states could not be tracked continuously, and hence the effectiveness of controlling the IoT components would be decreased significantly because of updated information lack. In this paper, we introduce an information-centric approach for multiple sources monitoring issue in IoT. The proposed model thus is designed to provide generic and extensible data format for diverse IoT objects. Through this model, IoT developers can build smart services smoothly without worrying about the diversity as well as heterogeneity of collected data. We also propose an overall monitoring architecture for the information-centric model to deploy in IoT environment and its monitoring API prototype. This document also presents our experiments and evaluations to prove feasibility of the proposals in practice.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Internet of Things; cloud computing; edge computing; information-centric; IoT slice monitoring; data sensing, resource utilization; activity log

1. Introduction

Thanks to the capability of connecting physical and digital worlds, Internet of Things (IoT) has opened huge number of opportunities to generate data, which helps people understand more deeply and widely their living environments [1]. Based on the understandings, people can create smart operation scripts for surrounding devices in order to make life more convenience. Along with the development, IoT service number has also increased exponentially with many different types covering almost all fields from smart home, transportation, health-care to environment monitoring, industry automation and so forth. It can be realized that those intelligent services have to continuously update

* Corresponding author. Tel.: +84-24-3869-6124 ; fax: +84-24-3869-2906.
E-mail address: minhnb@soict.hust.edu.vn

1877-0509
Peer-review under responsibility of the Conference Program Chairs.

state information from the vast number of IoT components at the same time to make reasonable decisions during their operations. However, the problem is that the collected information can be different types including sensing data, resource consumption/utilizations, and event logs recorded on individual IoT objects.

The issue of data collection presented above relates closely to the slicing IoT resource approach that allows programmers to develop easily applications atop without worrying about differences of underlying technologies like networks, gateways, platforms, clouds and even sensors[2] via a well-defined programming interfaces. However, most of existing solutions just have created unified data models and programming framework which focuses on management problem. They still lack a mechanism which enables to collect and monitor states from different sources in the manner of IoT component slices in real-time.

An example with a smart management system of an office building can be given for the monitoring problem. Whenever an officer and his/her colleagues need to arrange online meetings in a room, they use the resource management system to register in advance. The meeting rooms are equipped with sensors to control lights, air conditioning, heating and status (on/off) of all devices, making it much easier for these officers to personalize climate. The meeting room thus is monitored by an automated temperature control system, which automatically turns on/off the heating/air conditioning before and after the meeting. These sensors can also adapt to match the year time, outside temperature, outside sunlight and to the need of presenters in automatic set-up of appropriate parameters for all resources before the meeting. It ensures his/her personal preferences are met while the room automatically adjusts those preferences in an environmentally friendly way.

There are several main challenges when dealing with IoT monitoring problem in the example above. Firstly, there are a huge number of devices that are dispersed everywhere in real environment[3]. Secondly, according to their functionalities as well as vendors, the devices belong to diverse types and they can use many different technologies, models and designs. Thirdly, to enable IoT applications work as expected in the heterogeneous environment, monitoring solution also must be developed in the manner of providing a single unified view for all monitored data. With those requirements, the IoT monitoring solutions will require having a certain information-centric model, which can deal with heterogeneity and extensibility factors of IoT.

In this paper, we concentrate on designing an information-centric model for IoT slices monitoring dealing with the problem of many data types as well as sources presented above. The model will complement state information of IoT slice things in real-time, thus allow managing more effectively smart systems than the existing solutions.

The rest of this paper is organized as follows. In Section 3.1, we analyze the IoT slices monitoring problem, then a proposed approach is introduced here. Section 2 discusses existing works that are related to the IoT monitoring topic. Section 3 describes our model design, proposed deployment architecture, its monitoring components. While experiments and evaluations are presented in Section 4, Section 5 concludes the paper with some future works.

2. Related work

In this section, we analyze and discuss several researches dealing with the IoT monitoring problem related to our approach. We roughly categorize the existing studies into two groups, namely:

IoT resource and information modeling approach. There were many researches which propose centralized management models for IoT resources based on programming abstraction layer. De Suparna et al.[4] introduced the semantic resource model using Web Ontology Language Description Logic (OWL-DL). Thus, the authors represented things as objects with many related properties. The advantage of this semantic annotation is that these things linked together in a structure architecture and hence their generated data can be associated. In[5], information model approach is used for integration, configuration and utility of heterogeneous product data in a distributed environment. The authors also proven that they could control the life-cycle of IoT services with their proposed framework. Other studies mention specifically about each IoT resource including cloud[6], and[7], network[8]. Thus, the works considered resource manageability in a heterogeneous environment. While[6] and[7] brought forward a resource overlay on top of cloud as programming objects, the authors of[8] focus on dividing a large network into small pieces to manage. We examine the researches presented above as premise for developing our system from the view of modeling approach. However, these works still do not reach the monitoring issue for diverse IoT things now. Besides, unlike resource modeling for management, the monitoring problem dealt in our study requires a data model meeting all metrics that belong to different IoT sources such as clouds, networks and devices.

IoT monitoring. Several studies have introduced separately monitoring solutions for cloud, network and IoT devices. In[9], Aceto, Giuseppe et al. listed many current platforms and services for cloud resource monitoring that belong to both commercial and open sources. However, the authors summarized that cross cloud monitoring was still at an early stage. This aspect requires a high heterogeneity of tools and an exchanged information. The EU project mOSAIC[10] proposed a framework with API for monitoring cloud applications. The approach slices vertically the application architecture from software functionalities to virtual hardware to monitor. Authors of[11] described designs for a novel general-purpose monitoring tool, which covers all aspects of cloud metrics. The tool was tested with Yahoo Cloud Serving Benchmark (YCSB) using OpenNebula middleware. Other works such as[12], [13], and[14] concentrated on QoS and SLA monitoring to audit resource providers commitments. In the term of network management, in line with the development of Software-Defined-Network (SDN) and Network Function Virtualization (NFV), there were a lot of monitoring efforts such as works of Chowdhury et al.[15], and Nguyen[16]. However, under the view of IoT with components as described in the Things layer of Figure 1, until now there are not any works, which propose a generic approach for monitoring all IoT data sources including clouds, networks and IoT devices simultaneously.

In comparison with the existing works presented above, our proposal emerges with the following contributions: (1) proposing a novel information-centric model and its deployment architecture for IoT slices monitoring problem. (2) proposing a solution for the problem of gathering data from multiple sources from edge devices to clouds at the same time. And (3) proposing a solution for the problem of having multiple diverse data types to allow creating API queries for different IoT contexts.

3. Designing information-centric model

3.1. Information-centric approach

The smart management system scenario described in Section 1 already shows that in the monitoring domain, many metrics correspond with a lot of things exist in practice. Even each component has many features to monitor. Hence, a comprehensive solution which not only enables to represent and model all metrics in unified format but also has extensibility is required for IoT monitoring. Thank for this, the data is aggregated from multiple sources so that user can make the best control decision. In Fig. 1, many metrics is depicted by Monitoring Metric layer and the Aggregation layer illustrates the query requirement, which use monitored data about thing states.

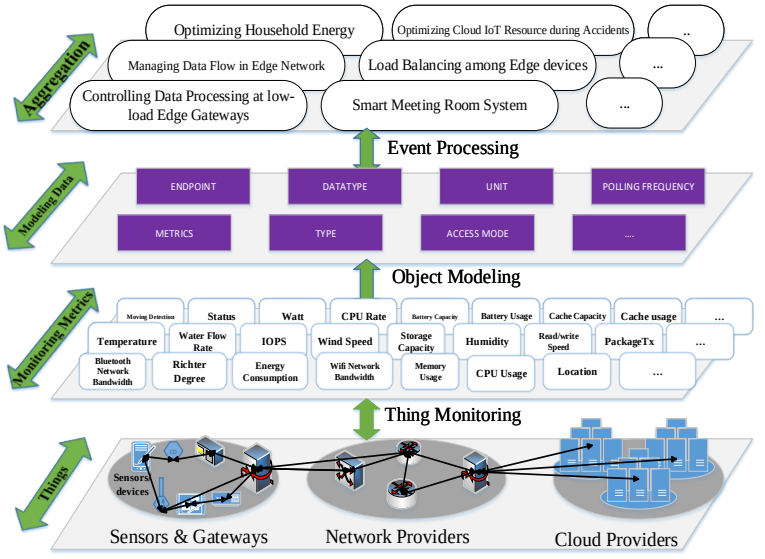

Fig. 1: IoT monitoring scenario, difficulties, and information-centric approach

To deal with the IoT monitoring problem, we classify the IoT monitored objects into three types as follows. (1) *Sensing data* produced from things (e.g. temperature, humidity values). (2) *Device resources utilizations* are CPU, memory, battery status, bandwidth and so on. (3) *Device action logs* are recorded to trace all activities occurring on the devices.

The captured data from three monitoring types above would help create a complete slice view about IoT contexts. As a result, developers can find the best ways to realize operations of IoT devices and applications. However, due to distributed and heterogeneous features of IoT, it cannot design completely a system from top to down for meeting all deployment contexts. Consequently, we still rely on existing and available IoT solutions (e.g. platforms, clouds and

network service tools) for managing and controlling things in effective way. In this direction, the existing solutions maintain their models and management ways, but we support them via *drivers*, *plug-ins*, *agents* and *data transformers*. Then, we develop *centric model* to centralize monitored data into a single unified form. According to user and thing demands, this core data model is generic and can be extended with diverse data types. By this way, the extension does not influence the core model to ensure stability for our approach. Based on the single unified data form, we can implement aggregation APIs with collected data gathered from multiple IoT sources at the same time as desired.

3.2. Information model

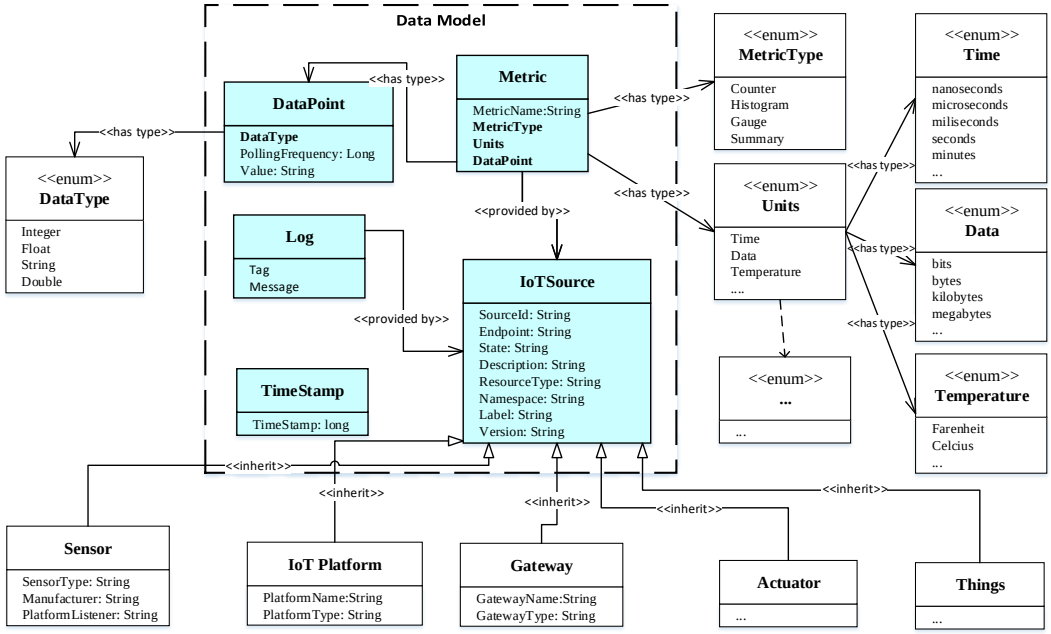

Fig. 2: Key components of the IoT monitoring information-centric model

As presented in previous Sections, we design a single unified model to centralize monitored data collected from diverse IoT objects. With the goal, our model provides generic and extensible characteristics for different data types as well as various things. Figure 2 shows our information model design, which involves five key components, namely:

- *IoTSource* enables data sources (e.g. sensors, IoT platforms, gateways, cloud servers, network services and so on) to declare their basic information to integrate into the model. Each object thus inherits attributes of this component.
- *Timestamp* represents time points when monitoring data are collected from sources.
- *Metric* is used to express diverse metric types of the collected data.
- *DataPoint* is contained by the Metric component to complete values information for metrics collected from IoTSource.
- *Log* represents log data (e.g. activities, tracing) written on IoT objects.

With the proposed model presented above, new IoT data sources (i.e. devices, gateways, platforms, networks, clouds) can be integrated easily with the information model by defining new classes, which inherit *IoTSource* attributes. Furthermore, those new classes also can be added their specific information according to source features or user demands as described in Fig. 2. This capability shows that our model can provide the extensibility to meet the heterogeneity requirement stated in Section 3.1.

3.3. Data integration

In most cases, data collected from different things have diverse formats. As a result, our information model must unify them into a single type. Figure 3 describes our data integration flow, where distributed agents through drivers gather monitoring data from things and transfer them to transformation service to integrate data. By this way, the adapter number of *Data Transformation* corresponds with the data format quantity. The monitoring data later are extracted to get information from every field before they are constructed into our unified model as presented by Fig. 2. For instance, a sample of temperature sensing data format gathered from sensors managed by OpenHAB platform[17] is defined as follow:

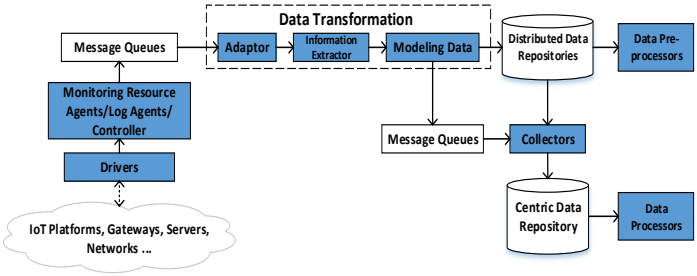

Fig. 3: Data Integration Flow

```
Temperature 30.2
Location San Francisco
Property current
Unit C
```

Meanwhile, representation format of temperature sensing data collected from sensors connected with HomeAssistant platform[18] is defined in the following way:

```
{
  "name": "Outside",
  "device": "weather-ha",
  "type": "air",
  "data": {"temp": "24C"}
}
```

Due to the format differences, these temperature sensing data are modeled by the proposed information-centric model, but they still keep required information. The following list shows a sample XML that is used to store data gathered from HomeAssistant platform above.

```
<Metric>
  <MetricName>Temperature</MetricName>
  <Units>
    <Data>Celsius</Data>
  </Units>
  <DataPoint>
    <DataType>Float</DataType>
    <Value>24</Value>
  </DataPoint>
</Metric>
<IoTSource>
  <SourceId>outside-sensor-temp</SourceId>
  <SourceType>Sensor</SourceType>
  <State>active</State>
  <Description>{"device": "weather-ha", "type": "air"}</Description>
  <PlatformType>HomeAssistant</PlatformType>
  <Version>v1</Version>
</Resource>
```

Due to the need of unifying multiple data types and formats, as shown on the list above, the new created data of proposed centric model could be larger than the size of original formats. However, according to our monitoring architecture presented in Sucbsection 3.5 below, there are several data repositories proposed to deploy in IoT environment. Then developers can manage the data quantity stored in each layer to avoid overload. In order to show the scalability of architecture, a changing sensor number test was carried out and its results are presented in the next Section.

3.4. Model properties

As mentioned before, because of the heterogeneity and distribution of objects in IoT environment, our information model must have the extension and adaptation abilities to ensure that data can be centralized effectively. In this design, the proposed information model has the following properties:

- *Abstraction*. Normally, an abstraction hides details of an entity. In terms of modeling data collected from many IoT objects, there are many "entities" that need to be mentioned. For example, in Fig. 2, Units are attributes of the *Metric* class. However, it contains a lot of types like time (e.g. nanosecond, microsecond), temperature (e.g. Farenheit, Celcius), data (e.g. bit, byte). Thus, at the highest level, we use Unit attribute to represent a large set of parameters.
- *Generalization*. An important property of our model is the extensibility, which allows many IoT object data to integrate under a single unified format. Specifically, an IoT data source can be connected with the *IoTsource* class by declaring well-defined parameters. In other words, those parameters are inherited by different IoT object classes during declaration process. However, the data source objects do not need to provide all information for the *IoTsource*, they only inherit/declare required parameters to identify themselves.
- *Containment*. In our model, data of one class could be contained by another class, which is of different nature. Such containment provides a logical interrelationship between two diverse classes to make the module rich in required information. For example, the *Datapoint* class is contained by the *Metric* to determine data types and values received by an IoT objects.

The abstraction, generalization, and containment properties of our information model can be mathematically defined by the formula: $\mathfrak{R} = \left\{(c_i, c_j)|c_i \in E_i, c_j \in E_j\right\}$, where $\mathfrak{R}$ represents relationships among c_i and c_j, which are parameters of classes E_i and E_j respectively.

3.5. Deployment architecture

Deployment architecture for our information-centric IoT monitoring system is illustrated by Fig. 4, where IoT environment is divided into three layers, including:

- *Monitored Sources*. This layer covers all IoT monitoring objects like sensors, IoT platforms, cloud services, network tools, and devices. Dealing with the heterogeneity problem, we only inherit existing functionalities, and information models provided by IoT data sources categorized into this layer.
- *Distributed Data Collectors*. To monitor a huge number of heterogeneous objects distributed everywhere, we propose an intermediate deployment layer to collect, cache, pre-process data. Components of this layer thus are distributed very closely to things. We use agents, and drivers to collect monitoring data, which involve three types: sensing data, resource utilizations, and logs. The collected data are transformed through adapters to model into our single unified format (as information model presented in Section 3.2). While cache data can be sent to distributed repositories, most heavy data are pushed to the highest layer. Also, in the intermediate layer, management queries from sky to things will be traversed management query/data pre-processor and configurators/controller components. However, some queries as well as small data processing can be made in this intermediate layer.
- *Centric Data Collectors*. This is the highest layer with a number of powerful computing resources. Collected data from lower layers are aggregated centrally at data collectors before it is stored in centric repositories. Analytics applications extract data from the repository for mining purposes, which generate management queries to send back to things (via the *Distributed Data Collectors* layer). The heavy operations of this layer can be mentioned covering store and processing of very large data.

Otherwise we also built a monitoring API prototype based on Python language for IoT application developers in easier way to get data and manage monitored objects. Moreover, smart processing can be defined and executed simply through the API. For example, the API offers function to allow gathering sensing data from a temperature sensor. If

the data is larger than 80 degrees, the sample disaster application will send a query to increase sensor's data acquisition frequency.

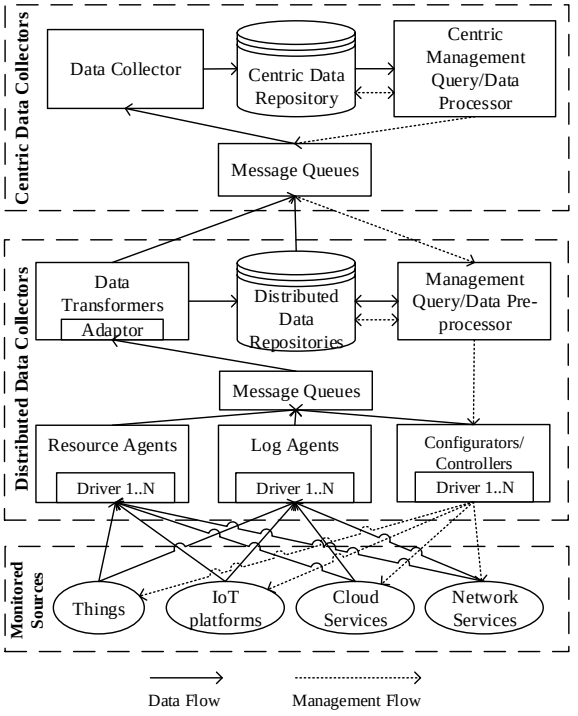

Fig. 4: Monitoring Architecture

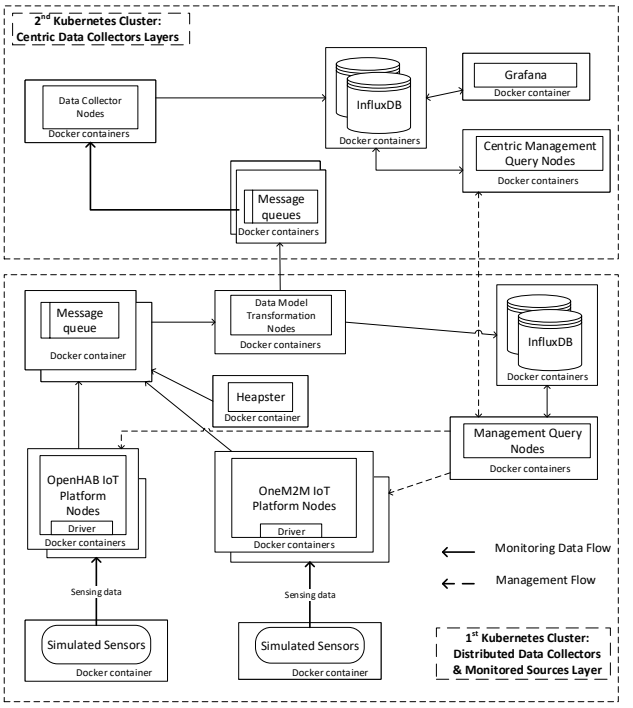

Fig. 5: Experimental Testbed

4. Experiments and evaluations

4.1. Experimental setup

Because of having a large-scale IoT testbed with full data sources from things, gateway nodes, networks, and clouds is roughly difficult in practice, in order to evaluate the feasibility of our information-centric model for the multiple sources monitoring problem, at the current research progress, we use servers and their defined-network to simulate IoT environment. Figure 5 describes our experimental testbed setup with two server clusters. The first cluster is used to simulate gateways and sensors for the *Distributed Data Collectors* layer. Meanwhile, the second cluster plays the role of providing services belong to the *Centric Data Collectors* layer. All our services (e.g. IoT platforms, agents, proposed ingredients and so on) are deployed on Docker containers. We use Kubernetes[19] to orchestrate and manage the machines. In addition, while Flannel[20] is exploited to define network routers, Heapster[21] is employed as resource utilization monitoring tool. The reason for our Docker ecosystem choice is that they could be deployed easily on many current light-weight physical devices such as Raspberry PI, Intel Galileo and so forth (to be considered as IoT gateways). Message queues in our tested are created by Mosquitto[22] to transmit data between components as well as layers. In *Centric Data Collectors*, collected data are pushed to time series database InfluxDB[23]. We deploy OpenHAB and OneM2M[24] IoT platforms on IoT gateway containers. Emulated sensors thus are managed and sent data to these platforms.

The main goals of our tests are to show, evaluate, and prove the abilities of monitoring and collecting data from many IoT sources as well as modeling the gathered information in our centric format. To perform these tasks, we carry out three experiments corresponding with three different scenarios, namely:(1) Changing the data generation rate of emulated sensors, and (2) tracking time delay in the case of sending data from the sensors to centric repository.

4.2. Changing sensor data generation rate

In this test, we create 10 emulated temperature sensors deployed in our testbed presented in Subsection 4.1. Next, we increase the sensors generated data frequency from 10 to 40 messages per minute for each. Based on the proposed framework prototype, we monitor resource usages (CPU, Memory, Network bandwidth) of every container, and sensing data generated from sensors through APIs provided by the IoT platforms. Meanwhile, event occurring on the machines are logged in the form of timestamp. All monitored data are modeled using our information-centric model before manifesting by diagrams. The monitored resources are calculated by averaging obtained figures.

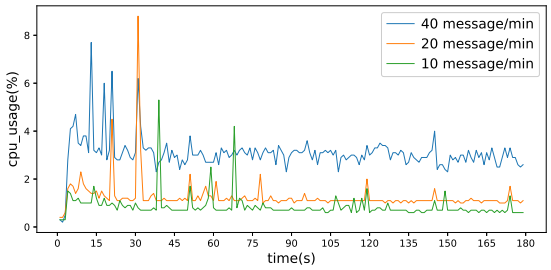

Fig. 6: OpenHAB nodes: CPU usage

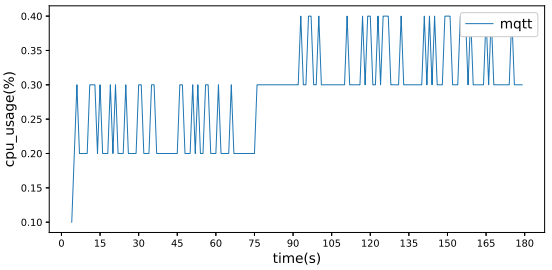

Fig. 7: MQTT node: CPU usage

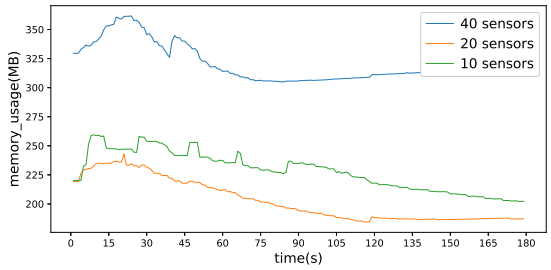

Fig. 8: OpenHAB node: memory usage

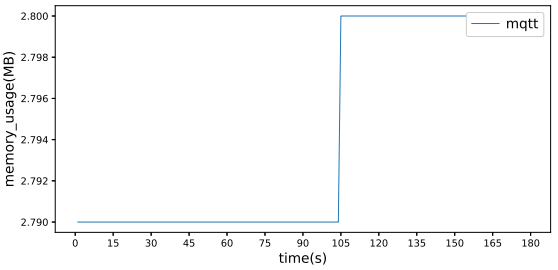

Fig. 9: MQTT node: memory usage

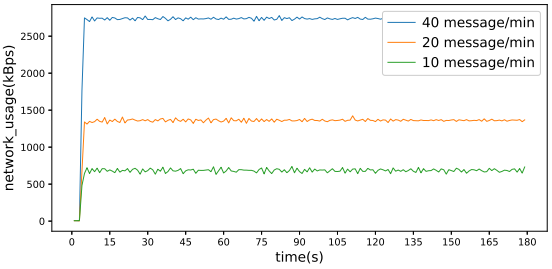

Fig. 10: OpenHAB node: network output

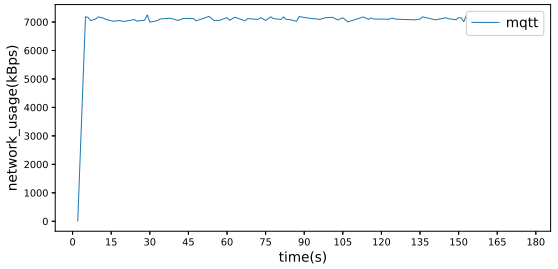

Fig. 11: MQTT node network output

Our achieved sample monitoring data are shown by 8 figures from Fig. 6 to Fig. 13 above. Respectively, Fig. 6, Fig. 8 and Fig. 10 illustrate resource usages of three OpenHAB platform nodes deployed on *Distributed Data Collectors* layer. Besides, Fig. 7, Fig. 9 and Fig. 11 show the resource consumption of message queue node deployed on *Centric Data Collectors* layer. In Fig. 9, the memory usage of queue node increases to maximum at 105-th second because of the generated data from sensors, the queue is full, and it uses entire memory of the node. Finally, Fig. 12 and Fig. 13 express the sensing data values and query frequencies of tested sensors. We monitor the data at *Centric Data Collectors* layer.

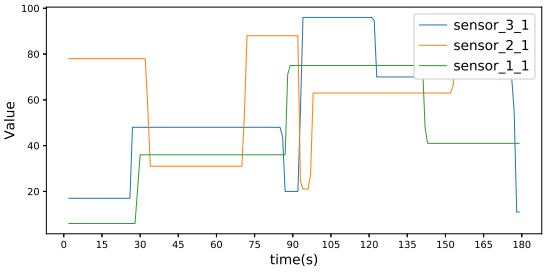

Fig. 12: Sensing values: Celcius degree

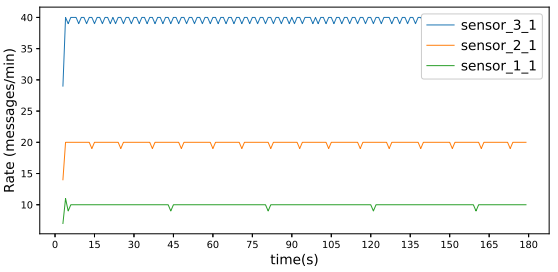

Fig. 13: Sensor sending data frequencies

An important observation can be made from this test: resource consumption of gateway nodes increases in direct proportion to the raise of data generation rate. The reason is that IoT services at *Distributed Data Collectors* layer need to collect and process increasing collected data amount. Through achieved outcomes of this experiment, we demonstrate that our framework prototype and information model can be used well for monitoring and gathering all three data types (resource utilization, sensing data and log tracing) from all three layers.

4.3. Time delay of data sending using monitored logs

In the third test, we increase the sensor number from 10 to 100 and track the transmission time delay when sending temperature data from sensors to repository in the *Centric Data Collectors* layer using log agents deployed on everything in our testbed. The goals of this experiment are to check the ability of recording logs across gateway nodes and evaluate the delay changes when increasing data amount (i.e. sensor numbers increase). In this direction, required time is separated into two stages, involving: (1) transmission time from sensors to IoT platform gateway and (2) transmission time from these nodes to the *Data Centric Collector* repository.

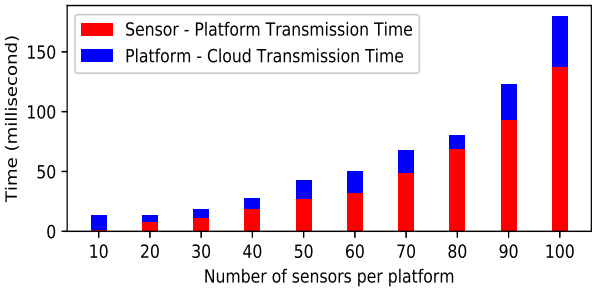

Fig. 14: Delay time by sensor number

Figure 14 represents the delay time increase when sensor number is augmented. It can be explained by reason: collected data amount increases in proportion to the sensor number. A thing can be recognized easily here: the transmission time of (1) are higher than (2). This phenomenon can be analyzed as follows. Due to the heterogeneity of deployment environment, data must be transmitted across several services as well as nodes in the *Distributed Data Collectors* layer, therefore they need more time to traverse. Inversely, because the intermediate node number among gateway to the *Centric Data Collectors* layer is smaller, data sent from platforms to centric repository is faster. With the obtained results, we proven that our framework prototype and its information-centric model can monitor and centralize effectively log data in IoT environment.

5. Conclusion and future work

This paper presents a novel approach for IoT monitoring problem. Thus, an information-centric model is proposed to provide an effective monitoring slice view of diverse IoT objects for developers. This model offers the ability of general and extension to centralize and integrate monitoring data collected from multiple sources with different types. We bring forward a deployment architecture for the information-centric approach with several key components, which are configured in three monitoring layers. Our data model and proposed architecture are experimented in a simulation IoT environment with gateways, emulation sensors, IoT platforms, processing servers based on Docker ecosystem solutions. Although this research still is in the initial stage, it fosters works focusing on IoT slice management

in very near future. Currently, we continue to enlarge and enrich our experimental testbed to connect to physical, virtual devices, and existing IoT tools (e.g. other IoT platforms, SDN solutions, and clouds) deployed in surrounding environment.

Acknowledgements

This research is supported by Vietnamese MOET's project No. B2017-BKA-32 "Research on developing software framework to integrate IoT gateways for fog computing deployed on multi-cloud environment", Slovak VEGA project 2/0167/16 "Methods and algorithms for the semantic processing of Big Data in distributed computing environment", and EU H2020-777435 DEEP-HybridDataCloud project "Designing and Enabling E-infrastructures for intensive Processing in a Hybrid DataCloud".

References

1. Cao, Tien-Dung, et al. "IoT Services for Solving Critical Problems in Vietnam: A Research Landscape and Directions." IEEE Internet Computing 20.5 (2016): 76-81.
2. Massonet, Philippe, et al. "End-To-End Security Architecture for Federated Cloud and IoT Networks." Smart Computing (SMARTCOMP), 2017 IEEE International Conference on. IEEE, 2017.
3. Nguyen, Binh Minh, et al. "Multiple Peer Chord Rings Approach for Device Discovery in IoT Environment." Procedia Computer Science 110 (2017): 125-134.
4. De, Suparna, et al. "Service modelling for the Internet of Things." Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on. IEEE, 2011.
5. Cai, Hongming, et al. "IoT-based configurable information service platform for product lifecycle management." IEEE Transactions on Industrial Informatics 10.2 (2014): 1558-1567.
6. Nguyen, Binh Minh, Viet Tran, and Ladislav Hluchy. "Abstraction approach for developing and delivering cloud-based services." Computer Systems and Industrial Informatics (ICCSII), 2012 International Conference on. IEEE, 2012.
7. Georgakopoulos, Dimitrios, et al. "Discovery-Driven Service Oriented IoT Architecture." 2015 IEEE Conference on Collaboration and Internet Computing (CIC). IEEE, 2015.
8. Lpez, Toms Snchez, et al. "Resource management in the Internet of Things: Clustering, synchronisation and software agents." Architecting the Internet of Things. Springer Berlin Heidelberg, 2011. 159-193.
9. Aceto, Giuseppe, et al. "Cloud monitoring: A survey." Computer Networks 57.9 (2013): 2093-2115.
10. Rak, Massimiliano, et al. "Cloud application monitoring: The mOSAIC approach." Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. IEEE, 2011.
11. Montes, Jess, et al. "GMonE: A complete approach to cloud monitoring." Future Generation Computer Systems 29.8 (2013): 2026-2040.
12. Alhamazani, Khalid, et al. "Cloud monitoring for optimizing the QoS of hosted applications." Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on. IEEE, 2012.
13. Emeakaroha, Vincent C., et al. "DeSVi: an architecture for detecting SLA violations in cloud computing infrastructures." Proceedings of the 2nd International ICST conference on Cloud computing (CloudComp10). 2010.
14. Tran, Dang, et al. "A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness." Procedia Computer Science 108 (2017): 365-374.
15. Chowdhury, Shihabur Rahman, et al. "Payless: A low cost network monitoring framework for software defined networks." Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014.
16. Nguyen, Hung. "Monitoring virtualized network." U.S. Patent Application No. 13/560,968.
17. Smirek, Lukas, Gottfried Zimmermann, and Daniel Ziegler. "Towards Universally Usable Smart HomesHow Can MyUI, URC and openHAB Contribute to an Adaptive User Interface Platform?." (2014).
18. Mirocha, Åukasz. "Smart home as the Internet of Things black-box: User's agency in open and proprietary Smart Home ecosystems." Digital Ecosystems: 119.
19. Bernstein, David. "Containers and cloud: From lxc to docker to kubernetes." IEEE Cloud Computing 1.3 (2014): 81-84.
20. CoreOS (2015). Flannel. https://github.com/coreos/flannel.
21. CoreOS (2014). Heapster. https://github.com/kubernetes/heapster.
22. MQTT, Mosquitto Open Source. "v3. 1/v3. 1.1 Broker." (2016).
23. Ganz, Julian, Matthias Beyer, and Christian Plotzky. "Time-series based solution using InfluxDB."
24. Swetina, Jorg, et al. "Toward a standardized common M2M service layer platform: Introduction to oneM2M." IEEE Wireless Communications 21.3 (2014): 20-26.