Software-Defined Virtual Sensors for Provisioning IoT Services On Demand

Chau Nguyen, Doan Hoang
School of Electrical and Data Engineering, Faculty of Engineering and IT, University of Technology Sydney
15 Broadway, Ultimo NSW 2007, Australia
Thiminhchau.nguyen@student.uts.edu.au, doan.hoang@uts.edu.au

ABSTRACT

Internet of Things (IoT) has been increasingly developed to provide essential IoT services ranging from personal health, smart homes to smart cities, and critical infrastructures. Sensor/IoT devices are indispensable elements in these systems/services. However, they are too rigid to permit reconfiguration for changes after their implementation. This makes it difficult to provision IoT services on demand and causes inefficient utilization of resources. Software-defined networking (SDN) and Network function virtualization (NFV) are emerging solutions to the programmability of network functions. Provisioning IoT services on demand is a natural utilization of programmability. Inspired by the benefits of SDN-NFV programmability, this paper proposes a software-defined virtual sensor (SDVS) that enables the programmability of IoT devices in accordance with IoT applications on demand. The paper presents the design and implementation of the proposed SDVS and demonstrates its use in an on-demand IoT services scenario.

CCS Concepts

• **Software and its engineering → Software as a service orchestration system**

Keywords

Services on-demand, IoT Services Provisioning, Software Sensors, Software-defined Internet of Things, Software-defined Virtual Sensor, Software-defined Internet of Things Controller.

1. INTRODUCTION

The most important element in an Internet of Things (IoT) system is the underlying resource that provides necessary data for IoT applications. An effective IoT architecture must allow the control plane to orchestrate, control, and manage the underlying resources to provide services for the application requests on demand. Deployment of various IoT applications faces many challenges due to its large scale, resource limitation, and heterogeneous environment that accommodates a huge number of devices/sensors with various capabilities of sensing, actuating, computing, and communicating. In many existing IoT applications, overlaid deployment of IoT devices leads to difficulties in the interaction and sharing information between the devices and the applications. The key challenge is the programmability of various sensor nodes in response to diverse IoT application demands.

Among programmable solutions to the programmability of wireless sensor networks and Internet of Things (WSN/IoT) systems, an emerging Software-Defined Networking (SDN) technique has been proposed.

However, applying the SDN paradigm to sensor/IoT networks faces serious challenges due to the limitations on the capability of these devices and their interconnecting protocols [1]. The complementary Network Functions Virtualization (NFV) can partially address these challenges. NFV technology is utilized to virtualize networking functions as well as enhance the functionality of underlying sensors/IoT devices. This technology can be applied readily to the WSN/IoT environment for creating a virtual representation of sensors/IoT devices that can serve multiple IoT applications simultaneously. This virtual representation offers a solution to enrich the features of limited sensors/IoT devices. This paper focuses on the design and application of a software-defined virtual sensor (SDVS).

It goes without saying that an "intelligent entity" has the ability to sense and interact with its environment in a meaningful way in order to achieve the entity's end goal. A sensor in its simplest form is just a physical element or device that transforms some features of the environment into a quantifiable measure for decision making by higher functional levels.

In the Internet of Things, a "thing" is basically defined as an object that can perform a function (or a sensing service), has the ability to connect to a network for collaboration and an identity (address, names, etc.) so that the device can be called upon to perform its intended service.

A "smart object" is somewhat a glorified term for an intelligent IoT thing, but the term is rather vague without further qualifications as the word "smart" implies a continuum of degrees of intelligence.

In the Industrial Internet of Things (IIoT), the term cyber physical system (CPS) is defined to mean a system that possesses the capability for sensing and interaction with its environment, the capability for doing the computation, and the capability for communicating with other cyber physical systems.

In its simplest form, a simple sensor is limited in its functionality (simple wires for sensing the temperature), does not have the capability to perform data preprocessing, or make its output available for further deployment and hence very limited in its application.

Attempts have been made to create more useful/intelligent sensors. One may surround the basic sensing function with computing and communicating capabilities by embedding in the hardware components such as operating systems and wire/wireless intelligent interfaces.

Other attempts rely on a bare minimum hardware platform and implement all additional intelligent features in software. In other words, with a layer of software over the basic hardware sensor, the composite device acquires intelligence and is able to perform desired functions and to provide services as envisaged.

In many situations, a sensor is purely a software entity (algorithm, object, middleware layer) that either emulates a physical sensor or a software object that provides higher level services to the user, relying on logical information available within a system.

On computing, additional functionality may include various preprocessing functions that preprocess data (noise removal or compressing) before making it available. On communication, additional functionality may include transmitting and receiving components that can interface with different communication wired or wireless protocols depending on the application. On storage, raw data collected by a sensor may need to be stored locally and temporarily for local preprocessing before transporting to a data collector, depending on the availability of the connectivity at the time.

With the advance of digital transformation, Industry 4.0 specifically, every element of a manufacturing factory has its digital twin as an essential component of the overall smart manufacturing industry. A digital twin of a component is a purpose-built software entity that mirrors the essential features of the component. A digital twin can range from a cyber physical system (smart sensor, smart conveyor belt, a production line) to a production process or a complete factory. These are emulated software components that an intelligent controller, orchestrator, or an enterprise can use to simulate, test out the operation and the intended product before committing them to an actual production and manufacturing process.

Many emerging services can now be implemented as cloud services, where a cloud can provide necessary virtual resources on demand (based on a pool of physical resources) and provision services as requested. In this infrastructure of virtualised resources, all computing, storage, and networking resources are implemented in software. Cloud can thus provide services with 5 defined features: on-demand self-service, rapid elasticity, broadband network access, measured service, and resource pooling.

Clearly, a cloud-based IoT service can be provisioned once elements of the service can be virtualised and orchestrated.

In this paper, we propose, design, and implement a software-defined virtual sensor (SDVS) that embraces basic features of the above discussed devices.

The rest paper is organised as follows. Section II discusses related work on the development of virtual sensors. Section III provides a detailed description of the proposed SDVS. Section IV describes the SDVS architecture and software implementation. Section V describes the use case scenario and performance evaluation. Section VI concludes the paper with directions for future research.

2. RELATED WORK

IoT applications are increasing rapidly in number, and they all demand increasingly sophisticated and capable IoT devices (sensors/actuators, sensor nodes, or IoT things). However, most of IoT devices possess limited capabilities such as power supply, computation capability, communication protocol, and memory/storage capacity. To mitigate, remove these difficulties or enhance physical capabilities for responding to emerging IoT demands, many attempts have been made for operation and resource efficiency, real-time response, autonomous device configuration, programmable control, flexible management, and on-demand provision of services.

Many attempts exploit virtual sensors/objects to deal with the above requirements. Examples of virtual objects have been discussed in [2]. A virtual sensor can play various roles. It can act as an IoT gateway to communicate with other components. It can represent one or multiple physical sensors to provide a sensing data type or aggregated data that may be collected from one or multiple physical sensors from one or several WSN platforms over a large-scale area [3]. In addition, a virtual sensor may share complex tasks with physical sensor/IoT devices by enhancing and supplementing itself with additional software functions [4]. It may also be a communication interface between an application and physical sensors, for example, SenseWrap [5] that enables an application to communicate with any kind of physical sensors without knowing device-specific details. Without providing a specific solution to the above issues, but [6] contributes to providing an overall picture where middleware solutions can fit in and proposing a Sensor Web Enablement framework where sensors need to be treated in a uniform, interoperable, and platform-independent manner.

Nevertheless, there remain many challenging issues [2] that need to be considered for future development of the IoT world regarding virtual sensors and their applications: i) a lack of common association between a virtual and real objects; ii) the balanced tradeoff between the number of replicas of the same information and their reusability; iii) the interoperability concern that is the consequence of virtual objects of different IoT systems having different APIs; iv) the scalability issue relating to the management of virtual object life cycle; v) the future creation of virtual objects that may autonomously and adaptive interact with the surrounding environment in order to support dynamic deployment of IoT applications.

Recently, a number of pieces of research have attempted to leverage SDN and NFV paradigm to introduce new approaches to not only existing but also upcoming IoT issues such as architecture, security, management, programmability and management of IoT infrastructure [7]; or provision of advanced services regarding network virtualization, data distribution, quality of services/experience. However, only a few works consider the benefits of SDN/NFV-based mechanism in programmability, control, and management of IoT devices in order to provide a complete SDN/NFV-based solution to IoT service provision. [8] has proposed a software-defined device provisioning framework for improving the scalability of IoT platforms, but it fails to consider how to deal with limitations of physical devices in the provision of IoT devices to respond to IoT applications on demand. [9] has proposed an SDN&NFV-based paradigm, that using virtual images to replace physical devices, for effectively providing collected data to users.

3. SOFTWARE-DEFINED VIRTUAL SENSOR (SDVS)

Typically, a physical sensor node is composed of four key units responsible for computing, communicating, sensing functions, and power. The computing subsystem controls the other units and computes demanding tasks. It includes a processor and a storage unit. The processor unit may operate in various energy-saving modes as Sleep or Off-Duty, Active or On-Duty, sensing unit on duty, and transceiver on-duty mode. The storage unit comprises a flash memory, containing the program code for a node, and a RAM, storing sensing data and necessary data for computing tasks. The communicating subsystem allows a sensor node to communicate with other nodes or with the base station by using a short-range radio. The power subsystem includes a battery. The sensing subsystem translates physical phenomena into electrical signals.

However, depending on their application, IoT devices may have different functionalities and may deploy different underlying technologies. In terms of communication, they may use different wireless technologies that entail different routing protocols, addressing schemes, data encodings, and data formats. Regarding the IoT platform, they may rely on different development environments, programming languages, processors, memories, and communication networks [10]. IoT devices can be of various types, but they typically include common elements such as i) identification, ii) sensing and actuating, iii) computation, iv) communication interfaces, and v) management. They may consist of several communication interfaces such as audio/video, memory and storage, Internet connections, and I/O interfaces for sensors.

To shield an IoT device from the above-mentioned dependencies, to overcome the limitations of physical sensors/devices, to allow autonomous device configuration and management, and to allow services programmability, in accordance with the SDN and NFV principles, we propose a virtual sensor as a software representation of an IoT device with the following definition and properties.

3.1 SDVS - Capability

SDVS is a software entity that functions as a virtual sensor that addresses the above-mentioned limitations of physical sensors/actuators and possesses capabilities for adapting itself in interacting with the surrounding environment and its controller for providing desired services. Importantly, an SDVS is flexible in communicating with its attached end devices (sensors/actuators) regardless of their specific communication protocols.

Specifically, an SDVS can be considered as an enriched version of a physical sensor or a group of physical sensors by virtue of the additional software layer on top and flexible sets of plug-in sensor interfaces.

An SDVS can provide value-add functions over that available from its underlying sensors through its software implementation. An SDVS may be also able to provide some local processing and management such as data pre-processing and local configuration. An SDVS may also include some storage to deal with other local issues rather than sending them to remote handlers. An SDVS can emulate a sensor, actuator, or provide a digital twin of a device for designing and testing preproduction of service in an Industrial IoT application. By software-defined, we mean that an SDVS can be flexibly designed, programmed, configured, and managed autonomously according to its intended application.

Essentially, an SDVS is defined by its representation type and an interface to its underlying resources.

3.2 SDVS – Representation Types

In accordance with the definition of an SDVS, representations of an SDVS are classified into two main groups: i) sensing and ii) functioning services. Regarding the first category, an SDVS may represent a single sensing service that is accumulated from one or multiple physical/virtual sensors. The second category represents a functional counterpart or an advanced functional element of a physical/virtual sensor.

3.2.1 Representing a Physical Sensor

A constrained physical sensor may need the additional support of a virtual sensor to become a programmable entity in an IoT system. Particularly, sensing and functioning services of the physical sensor can be i) programmed to provide IoT services to multiple applications.

3.2.2 Representing a Sensing Service of Multiple Physical/Virtual Sensors

That provides the same sensing service type. A virtual sensor can collect a kind of sensing reading from multiple physical/virtual sensors. The reading can be used for two purposes: aggregating the reading values to produce new average value and deriving a data type that cannot be produced by a single sensing type. For the first purpose, the temperature of small towns is collected for further production of an average temperature of a city. For the second purpose, a proximity sensor value is produced by interpolating light reading and variance in the light intensity [3].

3.2.3 Representing a Subset of Services of One/Multiple Physical Sensors

An IoT application may require several sensing services that need to be exposed to the application as a single service. For example, an SDVS may have to abstract different data types from a physical sensor and provide aggregated data to the application [11].

3.2.4 Representing a Subset of Services of One/Multiple Virtual Sensors

For example, a heat index is derived from moisturizer and temperature reading. Thus, to evaluate the heat index of a campus including many buildings, to the SDVS necessarily to provide average heat indexes collected from multiple virtual sensors that represent heat indexes of buildings within the campus.

3.2.5 Representing a Functional Counterpart of a Constrained Physical Sensor

The SDVS can enhance limited functionalities of a physical sensor, for instance, i) addressing and naming; ii) search and discovery, iii) mobility management, and iv) accounting and authentication. Examples of these cases are discussed in [1].

3.2.6 Representing an Advanced Functioning Service of a Physical Sensor

The SDVS enables an advanced function to be deployed on a physical sensor. The additional element enhances the physical sensor's capabilities according to application demands. For instance, an application-specific sensor is used for another application that has a different communication protocol. Thus, the SDVS allows the physical sensor to communicate with the application via the required communication protocol.

3.3 SDVS - Features

To embrace the SDN and the NFV principles and to fulfil the defined representations discussed above, the SDVS is composed of the following features.

3.3.1 *Fundamental Properties*
It has all the characteristics of a represented sensor/IoT device. When representing a service, it provides collected results regarding the service. When representing a physical sensor, it has all the characteristics and functions of the sensor/device. When representing a group of physical sensors, it has not only the information of the represented sensors but also additional functions that enable it to handle communication with the physical sensors and high-level tasks. It is also the same as the case of an SDVS representing a group of virtual sensors.

3.3.2 *Initiation*
It is initiated by the SD-IoT controller when i) an IoT device or a sensor node joins an SD-IoT cluster, or ii) there is a call for a new IoT service.

3.3.3 *Location*
An SDVS can be placed at the controller or at the physical device according to the resource orchestration approach of the controller.

3.3.4 *Activation Period*
An SDVS is activated according to application demands, so it would be in idle or deleted when it is not involved in any IoT service provision or its represented sensor is no longer existed.

3.3.5 *Identification*
Each SDVS is identified by a name, ID, and an IP address. A name and ID enable it to locally communicate with each other. An IP address allows it to globally communicate with the controller or applications on the cloud/the Internet.

3.3.6 *Configuration and Management*
An SDVS is managed and configured by the SD-IoT controller via a manage protocol. The configuration is deployed by the controller and represented by instructions in forwarding and configuring tables. The SDVS follows the instructions to know how to process an incoming packet as well as program underlying sensors, respectively.

3.3.7 *Association Between the SDVS and Physical/Virtual Sensor(s)*
Regardless of many representation types, there are four main kinds of association between an SDVS and physical/virtual sensors: i) one-to-one, ii) one-to-many, iii) many-to-one, and iv) many-to-many. As for one-to-one association, an SDVS represents a physical/virtual sensor or a service belong to the device. Regarding one-to-many association, there are three cases such as i) the SDVS may represent a set of physical/virtual sensors; ii) the SDVS collects information of a homogeneous service from a variety of physical/virtual sensors; or iii) the SDVS represents a subset of services of a physical sensor. In the case one physical/virtual sensor has many services and each of them is represented by one SDVS, so it results in the many-to-one association. The many-to-many association is a combined method of other associations.

3.3.8 *Stored Data*
An SDVS needs to store information about itself and its represented sensors. The information of the represented sensor includes identification; computing and communicating characteristics; current, past, and future status related to handling tasks and duty cycle modes. The statuses are stores in the list of attributes of the virtual sensor. In addition, it has its own identification, configuration information, including forwarding and configuring tables and statuses.

3.3.9 *Communication Interfaces*
There are four communication types that can be with i) its underlying sensors, ii) other SDVSs, iii) the controller, and iv) other cloud services/applications. It needs protocols to communicate with the controller or other SDVSs, IoT applications, and its represented sensors, respectively.

3.3.10 *Security*
An SDVS is managed by a controller that is responsible for managing its life-cycle. The SDVS can be created, modified, transferred, deactivated, activated, or deleted by the controller. However, the controller can delegate the control to an application/user, another SD-IoT controller, or other SDVSs with limited permission.

3.3.11 *Mobility*
An SDVS can be moved between SD-IoT systems or within an SDN domain, or to the cloud.

3.3.12 *Service Advertisement*
An SDVS informs and updates its available services to the controller.

3.3.13 *Service Provision*
SDVS stores temporary sensor services collected from its represented sensors and reuses them to quickly respond to application requests.

3.3.14 *Programmability*
An SDVS can be programmed to operate according to its configuration set up by the controller. It is responsible for communicating with represented sensors via their specific protocol.

3.3.15 *IoT Interface*
An SDVS is an interface between underlying IoT resources and the controller, applications/users, or the cloud.

4. SDVS ARCHITECTURE AND IMPLEMENTATION

4.1 Overall Architecture
To fulfil this design, the SDVS requires three main elements organized in the architecture, as depicted in Figure 1.

4.1.1 *Represented Entity*
Includes attributes and functions of the represented underlying entity, for example, battery level, driver communication protocol, attached sensors, processing capability, memory.

4.1.2 *Software-Defined Function (SDF)*
Are communicating or computing functions which can be programmable via a manage protocol.

4.1.3 *Data Storage*
Stores metadata of the represented entities, sensing data, and instructions for configuration of represented entries or for forwarding results to desired destinations. Particularly, it temporarily stores actual data collected from its represented entities.

4.2 Software Implementation
In software implementation, an SDVS is composed of four elements (as illustrated in Figure 2): i) identification and address for communicating with both SD-IoT controller and represented

IoT devices, ii) sensing/actuating services of represented IoT device, iii) storage data element storing information about sensing/actuating services, and iv) advanced functions that can be flexibly installed such as communicating and computing functions.

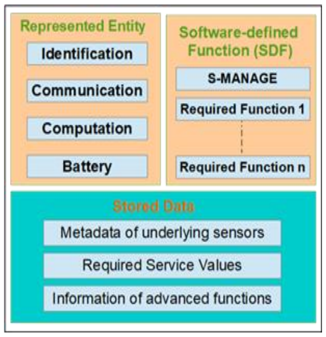
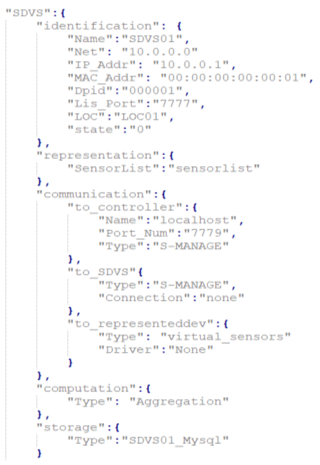


Figure 1. SDVS Architecture.



```
"SDVS":{
    "identification": {
        "Name":"SDVS01",
        "Net": "10.0.0.0"
        "IP_Addr": "10.0.0.1",
        "MAC_Addr": "00:00:00:00:00:01",
        "Dpid":"000001",
        "Lis_Port":"7777",
        "LOC":"LOC01",
        "state":"0"
    },
    "representation":{
        "SensorList":"sensorlist"
    },
    "communication":{
        "to_controller":{
            "Name":"localhost",
            "Port_Num":"7779",
            "Type":"S-MANAGE"
        },
        "to_SDVS"{
            "Type":"S-MANAGE",
            "Connection":"none"
        },
        "to_representeddev":{
            "Type": "virtual_sensors"
            "Driver":"None"
        }
    },
    "computation":{
        "Type": "Aggregation"
    },
    "storage":{
        "Type":"SDVS01_Mysql"
    }
```

Figure 2. SDVS In Software.

5. USE CASE SCENARIO AND IMPLEMENTATION EVALUATION

5.1 Use Case Scenario

For the practical realization of the proposed SDVS, we develop a use case scenario in which SDVSs represent physical IoT devices to provide IoT services on demand. In the scenario, there are a number of IoT devices along a street. Sensing readings from the devices may be used for providing traffic load along the street, weather condition of an area, or to adjust traffic light according to traffic load over the street. IoT devices are under the control and management of a controller. Each IoT device is attached with sensors such as light, camera, proximity, temperature, and movement. Sensing readings and actuators from the device can be achieved and executed on demand, respectively.

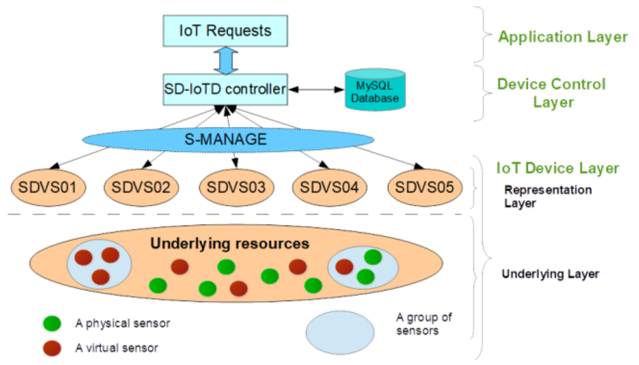


Figure 3. SD-IoT Model.

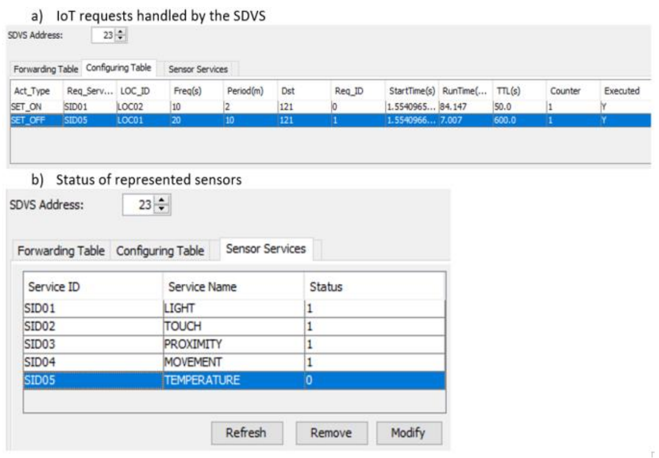
5.2 Implementation of the Overall SD-IoT Model

To demonstrate the above scenario, we implement a software-defined internet of things (SD-IoT) system that is introduced in our previous work [12] (as depicted in Figure 3). The system represents a cluster of IoT devices. It is composed of three main components: i) an SD-IoT controller, ii) a number of SDVSs, and iii) an S-MANAGE protocol that is a communication protocol between the controller and SDVSs. The SDVSs are representations of physical IoT devices. The SD-IoT controller controls and manages the SDVSs using the S-MANAGE protocol. Detailed implementation of the SD-IoT model, S-MANAGE's design and performance evaluation can be found in our previous work [13]. All components of the model are written in Java, using NetBeans 8.2.

5.3 Performance Evaluation

5.3.1 Programmability of SDVS

An SDVS is expected to have the following features (as presented in Figure 4). It is assumed that the SDVS possesses software divers and plug-in interfaces for various types of underlying physical sensors. We are in the process of deploying a Raspberry Pi for housing SDVSs and physical sensors' plug-ins.



a) IoT requests handled by the SDVS

SDVS Address: 23

Forwarding Table | Configuring Table | Sensor Services

Act_Type	Req_Serv...	LOC_ID	Freq(s)	Period(m)	Dst	Req_ID	StartTime(s)	RunTime(...	TTL(s)	Counter	Executed
SET_ON	SID01	LOC02	10	2	121	0	1.5540965...	84.147	50.0	1	Y
SET_OFF	SID05	LOC01	20	10	121	1	1.5540966...	7.007	600.0	1	Y

b) Status of represented sensors

SDVS Address: 23

Forwarding Table | Configuring Table | Sensor Services

Service ID	Service Name	Status
SID01	LIGHT	1
SID02	TOUCH	1
SID03	PROXIMITY	1
SID04	MOVEMENT	1
SID05	TEMPERATURE	0

Figure 4. Programmable Features of the SDVS.

5.3.1.1 Handling Multiple IoT requests at a time

As shown in Figure 4a, the SDVS23 currently handles two IoT requests that have Req_ID "0" and "1".

5.3.1.2 Configuring Represented Sensors in Accordance with IoT Demands

As displayed in Figure 4b, the SDVS23 currently represents five sensor types. It can configure the represented sensors to achieve required services such as sensing and actuating. For example, the sensor service SID05 is configured to be OFF (Status is "0"), while other services are ON (Status is "1").

5.3.1.3 Configuring and Updating Status of Represented Sensors According to Required Parameters Associated with IoT Requests

As presented in Figure 4a, the SDVS allows the IoT requests to specify metrics related to required services. Take the Req_ID "1" for example, the request requires service SID05 in location LOC01, for 10 minutes. Achieved results need to be sent to a desired destination (as Dst "121") every 10s. The SDVS also records the time (StartTime) when it starts handling (Executed is "Y") the request, the remaining time for handling the request (TTL). Moreover, it counts the number of requests interested in a specific service (Counter value).

5.3.1.4 Updating the SD-IoT Controller

The SDVS can update the controller about changes in its environment and status of its represented sensors, so the controller can resolve conflicts among IoT requests. Thanks to the updated relevant information, the controller can muster availability of underlying resources, and hence orchestrate them for incoming requests.

5.3.2 *Efficiency of SDVS*

The following results demonstrate the efficient utilization of the proposed SDVS in provisioning IoT services on demand.

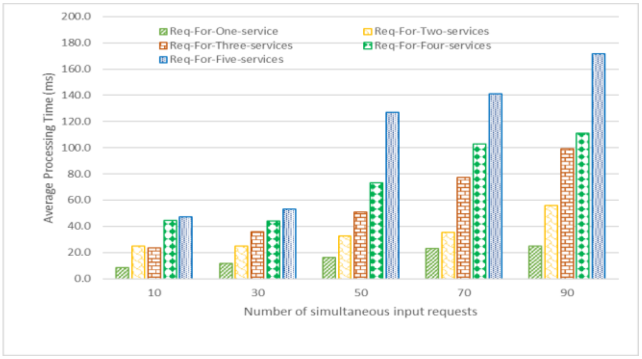


Figure 5. The controller's processing time for one per multiple concurrent requests.

Figure 5 presents the processing time of the SD-IoT model in response to one or multiple simultaneous application requests. The number of IoT services per request may vary from one to five. We try to send a number of requests to the SD-IoT model, and the number is increased by 20 from 10 to 90. The system needs more time to process a higher number of requests as well as requests demanding for a higher number of services. However, while the number of requests increases 9 times (10 to 90), the total processing time rises about 3 times regardless of the number of required services per request. The system can enable SDVSs to be shared between applications with similar interests, so it can reduce the time for reconfiguring physical devices.

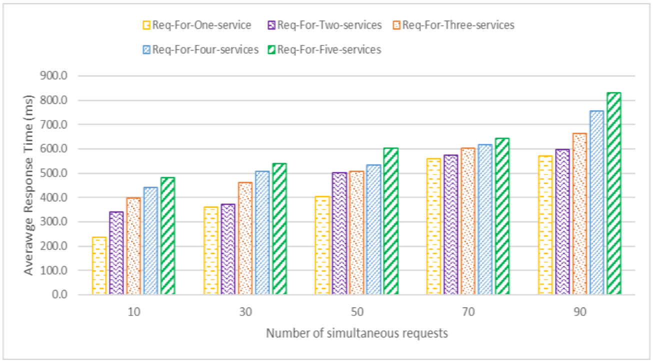


Figure 6. SDVS' s average response time for one request per multiple requests.

Response time of an SDVS to various requests: Figure 6 shows that response time of an SDVS to a request increases when there is a growth in the number of required sensor services per request. Moreover, it needs more time to reply to a rising number of incoming requests at a time. However, in comparison to an increase in the number of simultaneous incoming requests, the rise in response time of an SDVS is much lower. For example, to respond to 10 concurrent requests, regardless of the number of required services per request, an SDVS needs about 379.3ms on average. The amount is grown to about 1.8 times, while the number of concurrent requests increases by 9 times.

6. CONCLUSION

In this paper, we introduce a software-defined virtual sensor (SDVS) with new concepts to reshape the SDN and NFV technologies and support the provision of IoT services on demand. SDVS is designed to enable IoT devices to be programmable on demand in response to IoT application requests. A detailed design is provided. The implementation results demonstrate the feasibility and application of the proposed SDVS. For the next step, we will complete the integration of physical sensors into the SDVS and explore various applications. We will also investigate the integration of SDVSs into a large-scale software-defined IoT infrastructure for provisioning IoT services on demand.

7. REFERENCES

[1] Kobo, H.I., A.M. Abu-Mahfouz, and G.P. Hancke, A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. IEEE Access, 2017. 5: p. 1872-1899.

[2] Nitti, M., et al., *The Virtual Object as a Major Element of the Internet of Things: A Survey*. IEEE Communications Surveys & Tutorials, 2016. **18**(2): p. 1228-1240.

[3] Madria, S., V. Kumar, and R. Dalvi, *Sensor Cloud: A Cloud of Virtual Sensors.* IEEE Software, 2014. **31**(2): p. 70-77.

[4] Gupta, A. and N. Mukherjee. Implementation of virtual sensors for building a sensor-cloud environment. in 8th International Conference on Communication Systems and Networks (COMSNETS). 2016. IEEE.

[5] Evensen, P. and H. Meling. SenseWrap: A service oriented middleware with sensor virtualization and self-configuration. in 2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP). 2009.

[6] Bröring, A., et al., *New Generation Sensor Web Enablement.* Sensors, 2011. **11**(3): p. 2652.

[7] Omnes, N., et al. A programmable and virtualized network &amp; IT infrastructure for the internet of things: How can NFV &amp; SDN help for facing the upcoming challenges. in 2015 18th International Conference on Intelligence in Next Generation Networks. 2015.

[8] Mavromatis, A., et al. A Software Defined Device Provisioning Framework Facilitating Scalability in Internet of Things. in 2018 IEEE 5G World Forum (5GWF). 2018.

[9] Atzori, L., et al., *SDN&NFV contribution to IoT objects virtualization.* Computer Networks, 2019. **149**: p. 200-212.

[10] Ray, P.P., *A survey on Internet of Things architectures.* Journal of King Saud University - Computer and Information Sciences, 2016.

[11] Kabadayi, S., A. Pridgen, and C. Julien, *Virtual sensors: Abstracting data from physical sensors*, in *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*. 2006, IEEE Computer Society. p. 587-592.

[12] Nguyen, T.M.C., D.B. Hoang, and T.D. Dang. A software-defined model for IoT clusters: Enabling applications on demand. in 2018 International Conference on Information Networking (ICOIN). 2018.

[13] Nguyen, C. and D. Hoang, *S-MANAGE Protocol for Provisioning IoT Applications on Demand.* Journal of Telecommunications and the Digital Economy, 2019. **7**(3): p. 37-57.