International Journal of Smart Sensor Technologies and Applications
Volume 1 • Issue 1 • January-March 2020

Large-Scale Software-Defined IoT Platform for Provisioning IoT Services on Demand

Chau Thi Minh Nguyen, University of Technology Sydney, Australia
Doan B. Hoang, University of Technology Sydney, Australia

ABSTRACT

Internet of things (IoT) has developed into an interconnected platform infrastructure for providing everyday services. Emerging end-to-end IoT services are being developed for local and multiple distributed regions. To realize the on-demand services in a timely and economically beneficial way, programmability and reusability are crucial for provisioning and reusing IoT resources. Existing IoT platforms are rigid and cannot be easily adapted to accommodate new services. This paper proposes a programmable large-scale software-defined IoT model for provisioning IoT services on demand with two levels of management and orchestration. One orchestrates services over geographically distributed clusters and the other orchestrates services over IoT devices within a cluster. The model entails the design of IoT-specific controllers, software-defined virtual sensors, and a new protocol for managing resource-constrained but enriched devices. The model allows provisioning and resource-sharing of end-to-end IoT services on demand. Implementation results demonstrate the feasibility and efficiency of the proposed model.

KEYWORDS

IoT Services Provisioning, Provisioning Services On-Demand, Software-Defined IoT Model, Software-Defined Virtual Sensor (SDVS)

1. INTRODUCTION

The Internet has changed our world and brought with it many technical, economic, and social benefits by connecting people. It is expected that the Internet of Things will create enormous value by interconnecting people and everyday things toward a green IoT environment where people are provided with not only smart but also environmental IoT services (Solanki & Nayyar, 2019). In fact, IoT has already enabled many emerging applications and services critical to our life in various domains from personal healthcare to critical infrastructures, and large-scale systems as smart cities (Krishnamurthi, Nayyar, & Solanki, 2019). However, this enormous potential is limited by existing IoT systems/platforms, which are mainly closed ecosystems that are vertically developed and deployed in their own IoT infrastructure and have incompatible standards, formats, semantics, and proprietary protocol and interfaces (Do, Le, Paul Lin, & Tung, 2019). Security concern also prevents the sharing

DOI: 10.4018/ijssta.20200101.oa1

of IoT resources among IoT applications (Jain, Jain, & Nayyar, 2020; Nayyar, Jain, Mahapatra, & Singh, 2019). As a consequence, a number of major issues have been identified with the current generation IoT systems/platforms:

The vast number of devices and connectivity-service infrastructure. As projected, there would be about 41.6 billion IoT devices in 2025 (Framingham, 2019). The challenge here is how to manage the complexity of the interconnecting infrastructure and to serve both local communities and geographically distributed communities covering a large or global community effectively. This also poses a challenge to application management when a thousand of IoT devices are under the management of an application network (Ramachandran & Krishnamachari, 2019).

The huge number of IoT services and service provisioning. IoT devices are capable of interacting with their environment, performing some basic functionality as well as connecting with others. A huge number of services have already emerged to take advantage of these capabilities, and the challenge is to automate the provision of services on demand.

The massive amount of resources and resource sharing. Collectively, through interconnectivity, IoTs present a massive amount of available resources and services to be shared. The challenge is in the developing of algorithms and supporting infrastructure for sharing and reusing resources.

In order to address these challenging issues, we investigate technologies, network architectures, device capabilities, protocols, and programmable mechanisms for orchestrating services on demand.

On connectivity and networking architecture. Software-Defined Networking (SDN) enables network programmability and fine-grained flow-based automated management that are not available with traditional distributed networks. Through the logically centralized knowledge of the whole network, an SDN controller can configure network devices automatically to deal with network dynamics. Many networks are currently being deployed for these purposes (Habibi, Baharlooei, Farhoudi, Kazemian, & Khorsandi, 2018). IoT-programmability, however, is still far from expectation. We innovate the Software-Defined Networking concept for the IoT domain and investigate a large-scale architecture spanning both the SDN domain and IoT domain for end-to-end applications in the manufacturing industry.

On device capability. The highly resource-constrained nature of IoT devices regarding energy, computing power, storage, and wireless connectivity prevents a direct application of the wired SDN technique to the IoT world. Efforts have been attempted to address this issue; systematically enriching devices with resource sharing capability remain open challenges (Do et al., 2019). We investigate the virtualization technology to enhance and supplement the programmability of physical devices.

On communication and management protocol. Sensors/IoT devices are not network routing devices, and heavy protocols for programming network flows in network devices are not applicable to IoT devices. Efforts have been made to address this management issue with limited success. We investigate simple protocols for porting the SDN paradigm to IoT networks to compensate for the nature of IoT devices.

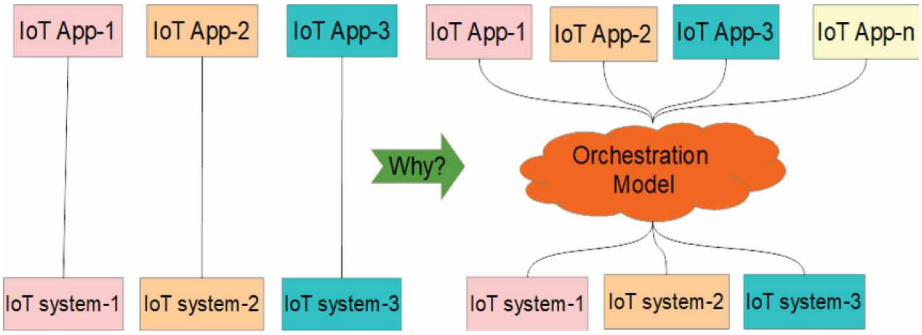
On programmable mechanisms for orchestrating services on demand. We enrich the capability of IoT devices with virtual functions and interface virtualization for orchestrating and programming services. We investigate algorithms and mechanisms for service orchestration.

On resources and services reusing and sharing. IoT services are emerging on a daily basis; both resources and services have to be shared among services for cost-effectiveness. We investigate a programmable platform for sharing the underlying IoT resources and provisioning them on demand.

This paper addresses these major issues by proposing and investigating a Large-Scale Software-Defined IoT (LSSD-IoT) model and associated techniques for provisioning end-to-end services on demand (as shown in Figure 1). It provides a distributed architecture for scheduling and allocating resources for IoT services. It allows the integration of independent IoT systems to SDN-based systems. It enables a configurable IoT infrastructure in the orchestration of IoT services on demand.

Different from traditional approaches, we apply emerging technologies including Software-Defined Networking (SDN) and Network Function Virtualization (NFV) in orchestrating IoT resources

Figure 1. The need for LSSD-IoT model



for provisioning IoT services on demand. Currently, the two techniques combine to provide flexible programmability and control of the wired network, but there remain challenges in integrating them to the wireless systems like WSN/IoT. We propose the LSSD-IoT model that not only limits the challenges but also leverages SDN and NFV advantages for not only a local IoT system but also a large-scale IoT domain. Our proposed model enables programmability and flexibility in the control and management of a joined network of wired network and IoT network in the provision of IoT services on demand.

This paper makes the following contributions:

- It proposes an LSSD-IoT and a control and management approach, allowing the provision of services on demand in a large-scale IoT platform infrastructure;
- It proposes a programmable software-defined IoT cluster (SD-IoTC) controller for managing and programming IoT clusters belonging to an LSSD-IoT platform infrastructure;
- It proposes a programmable software-defined Internet of Things (SD-IoT) model for managing and programming IoT devices within an IoT cluster;
- It deploys a simple protocol for the control-management of software-defined virtual sensors (SDVS) and their associated represented devices;
- It provides an implementation and evaluation of the proposed architecture in provisioning IoT services on demand.

The remaining of this paper is organized as follows. Section II reviews related work. Section III presents an overall LSSD-IoT model. Section IV describes the SD Cluster level. Section V describes the SD Device level. Section VI details the implementation of the proposed LSSD-IoT platform. Section VII demonstrates the performance evaluation. Section VIII concludes this paper.

2. RELATED WORK

The architecture of a WSN/IoT system plays an important role in the efficient utilization of WSN/IoT resources, for example, energy efficiency (Kamyabpour & Hoang, 2010). SDN and NFV have successfully enabled a central control and management of the wired network. Recent research has put effort into applying the principles in addressing IoT issues but there are remaining challenges regarding the IoT resources management aspect like device configuration, mobility of IoT devices, and virtual functions (Alam et al., 2020). Many efforts have attempted to build a large-scale IoT infrastructure for providing IoT services, but they address different aspects of a whole picture regarding the developing SDN/NFV-based infrastructure for developing IoT services over a geographical area. For example, (Habibi et al., 2018) has proposed a virtualized SDN-based end-to-end architecture for fog networking to manage the complexity of the joint network. They are interested in the orchestration

of integrated Cloud, Fog, and network resources for the development of IoT applications over a wide area. Nevertheless, they fail to consider network issues facing IoT systems as well as to provide a performance evaluation of the proposed architecture. In addition, (Do et al., 2019) developed and deployed an SDN/NFV-based network infrastructure for IoT. It builds and implements applications to slice end-to-end multiple network segments in accordance with the requirements of deploying IoT services from different providers. (Muñoz et al., 2018) proposed a model of integration of IoT, transport SDN, and edge/cloud computing for dynamic distribution of IoT analytics and congestion detection. Being evaluated and validated with joint experimentation, the work is well-performed in dynamically provisioning IoT analytics between the edge and cloud network and in controlling of bandwidth congestion. However, it lacks consideration to the control and management of data flow circulating within each IoT system, which may contribute to the improvement of data collection in distributed IoT systems. Also focusing on SDN/NFV-enabled network in IoT infrastructure, (K, Mansoor, Ahmad, & Kim, 2018) proposed a two-layer architecture to control network resources in terms of fault tolerance and load balancing in order to meet requirements of IoT applications, but it limits its contribution to the idea only.

Some other works have attempted to address issues at different levels in the development of a large-scale IoT system. (Ojo, Adami, & Giordano, 2016) has introduced an SDN/NFV-based IoT architecture that enables new IoT services upon existing IoT infrastructure. By leveraging SDN and NFV, authors (Mouradian, Jahromi, & Glitho, 2018) proposed a distributed gateway as a virtual network function that is chained in the SDN-based IoT system in large-scale disaster management. Another work (Yu, Xue, Kilari, & Zhang, 2018) has proposed a fog of things (FoT) paradigm that joins the fog and cloud computing toward the on-demand Internet of Things. It provides a layer-based FoT architecture with expected features of each layer, but it lacks a performance demonstration for its operation. It limits contribution to the development of the fog network architecture, which can be managed by the SDN controller. The work proposed by (Chendeb, Agoulmine, & El-Assaad, 2020) has introduced an SD-IoT architecture where the SDN controller directly manages IoT devices within an IoT system. This will face a challenge when there is a massive increase in the number of IoT devices.

3. LSSD-IOT MODEL

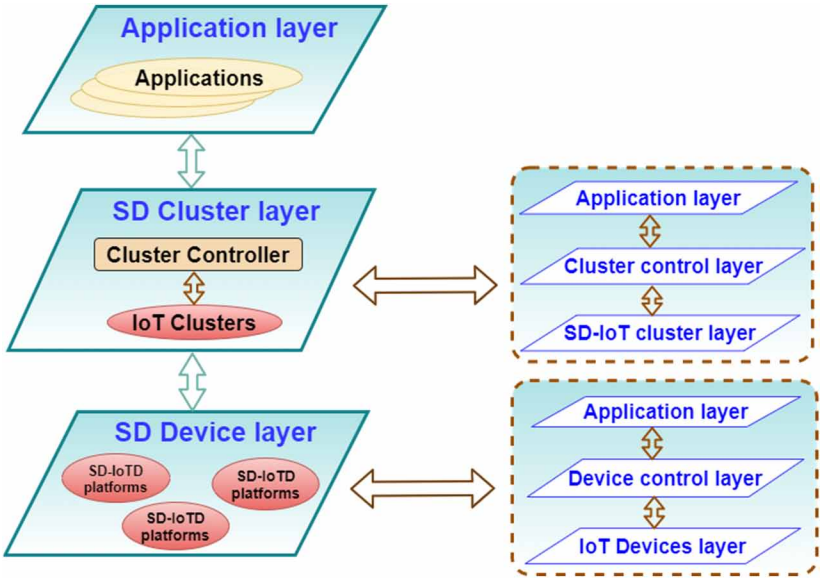
This section describes our proposed LSSD-IoT model that allows resource sharing, management, and programmability of heterogeneous sensor resources for provisioning end-to-end IoT applications on demand.

3.1. Overall Architecture

The high-level architecture of the LSSD-IoT model has three principal layers (Figure 2): the application layer, the software-defined cluster layer, and the software-defined device layer:

- **Application Layer:** This layer consists of end-user applications that utilize LSSD-IoT communications and services. Through the cluster controller, the applications influence the behavior of the underlying clusters by orchestrating, allocating, and coordinating them to provide the end-to-end requested services;
- **Software-Defined (SD) Cluster Layer:** This layer consists of IoT clusters and a cluster or central controller. Each cluster is responsible for its own local region. Each orchestrates, provisions, and manages services assigned by the controller. Each cluster can be considered as a virtual component that represents the capability of its underlying resources. The controller utilizes the collective capability of all underlying clusters to provide services requested by the application;
- **Software-Defined (SD) Device Layer:** This layer comprises IoT devices that are located in widely separated geographical areas. They are organized into groups to form clusters. Each cluster

Figure 2. LSSD-IoT model



is a platform with its own local controller to orchestrate, provision, and manage local services allocated based on the capability provided by IoT devices in the local cluster.

3.2. LSSD-IoT Features

3.2.1. Hierarchical Management

The model involves two levels of management and orchestration. The top level Software-Defined-IoT Cluster (SD-IoTC) orchestrates on-demand services over multiple clusters over a wide area. The bottom level Software-Defined-IoT Device (SD-IoTD) orchestrates on-demand services over sensors/IoT devices of a cluster. Each of the SD-IoTD clusters is a representation of an IoT local cluster.

The SD-IoTC consists of an SD-IoTC controller, multiple SD-IoTD clusters, and a communication channel between the controller and its clusters. The SD-IoTD is composed of multiple SD-IoTD clusters, each SD-IoTD has an SD-IoTD controller and multiple software-defined virtual sensors (SDVSs) and their underlying IoT devices within the cluster, and a communication channel between them. The LSSD-IoT system thus centrally controls and manages IoT devices/resources indirectly through the SD-IoTC controllers and the SD-IoTD controllers.

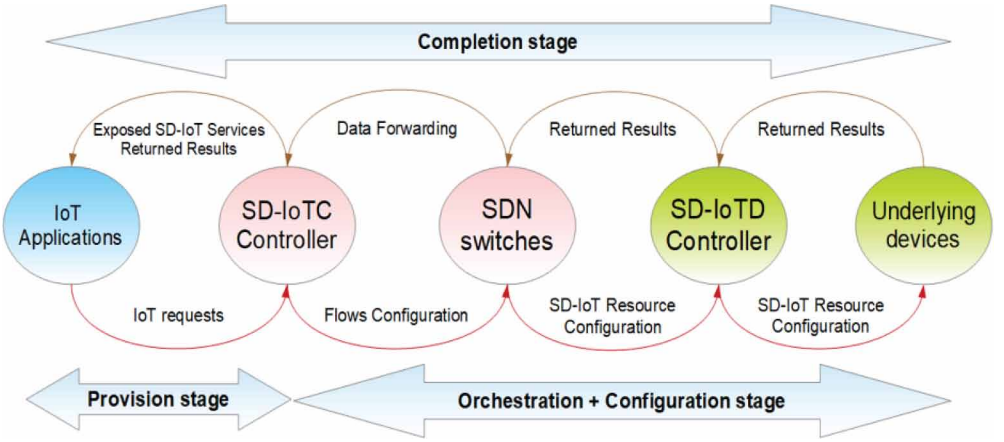
3.2.2. Provision of IoT Services on Demand

The provision of IoT services on demand is the ability to provide service whenever requested. This translates to the automation of orchestration, coordination, and management of IoT resources according to the service requirements device may provide various types of IoT services since it may include different sensors/actuators or functions.

Through the SD-IoTC and SD-IoTD controllers, the LSSD-IoT enables the automation of flexible and scalable provision of IoT services on demand. An IoT service can be established by chaining services of individual IoT clusters in various configurations: parallel, sequential, or complex configurations.

The process of an SD-IoT service provision via the LSSD-IoT system includes four stages (as presented in Figure 3). When receiving an IoT request, the SDN controller analyses the request and

Figure 3. The overall procedure of provisioning IoT services on demand via LSSD-IoT platform



accordingly orchestrates needed IoT clusters to achieve the required services. The SD-IoTC controller also instructs the SD-IoTD controller on how to forward the results to IoT service collection points. In accordance with the configuration of the SD-IoTC controller, each engaged IoT cluster orchestrates its IoT devices to obtain the required services and then delivers achieved results to the expected destination.

3.2.3. Programmability

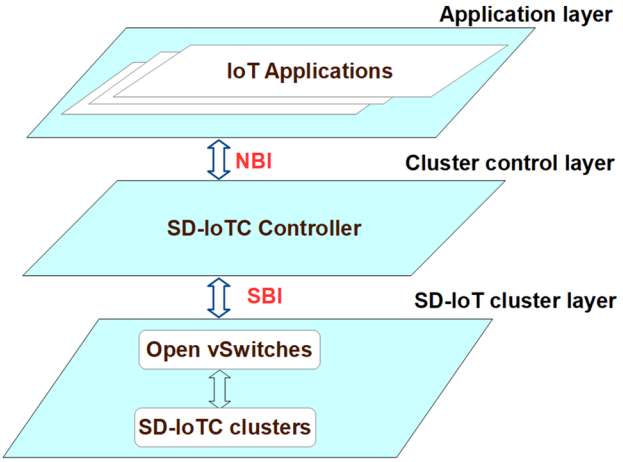
The LSSD-IoT enables the programmability of both transporting devices (switches) and IoT devices. At the cluster level, the SD-IoTC controller programs data flows over the core network to deliver IoT requests to relevant IoT clusters and transport the data/results to intended destinations. At the device level, SD-IoTD controllers orchestrate and program IoT devices within their own clusters according to the requirements of the SD-IoTC controller. The SD-IoTD controller configures SDVSs and via which programs IoT devices to achieve required services and deliver results to required destinations. The SD-IoTD controller, with its global knowledge of the underlying IoT resources and the status of the current service-provisioning tasks, can orchestrate and provide appropriate responses to application requests. The service request may be met partially or fully after the negotiation phase of the orchestration. If for some reasons such as insufficient resources to satisfy the service, an alternative arrangement may be suggested to the service requestor.

3.2.4. Virtualization, SDN, and NFV

By leveraging the SDN paradigm, the logically centralized SD-IoTC controller may i) obtain a global view of available SDN switches and the IoT clusters connected to the switches; ii) program data flows between the networking devices to forward IoT requests and direct IoT results to intended destinations. SDN OpenFlow is not appropriate for resource-constrained IoT devices, to gain the benefits of software-defined technology, a new control protocol and an enriched model of IoT devices are developed.

Using NFV technology, networking functions can be virtualized and hosted by physical servers. The model enables the virtualization of physical sensors/IoT devices-associated services as well as networking functions. The SDVS is a virtualized function representing IoT services from an IoT device. With the support of the SDN technique, traffic flows between these virtual SDVSs can be controlled and managed. This makes it possible for provisioning IoT services on demand.

Figure 4. SD cluster layer architecture



4. SOFTWARE-DEFINED CLUSTER LAYER

This layer inherits much of the SDN architecture. It has 3 layers (Figure 4): the application layer, the SD-IoTC control layer, and the SD-IoT cluster layer.

The application layer is the same as the application layer of the overall LSSD-IoT, housing end-user applications. The SD-IoTC control layer contains an SD-IoTC controller to perform both SDN functionality and IoT-specific service provisioning and coordinating functions. Instead of just SDN devices, they are replaced by SD-IoT clusters. Each cluster comprises an Open vSwitch and a host representing an SD-IoT platform below. The OpenFlow and orchestration protocols are used for the communication between the SD-IoTC controller and SD-IoT clusters.

4.1. SD-IoTC Architecture

4.1.1. SD-IoTC Controller

The SD-IoTC controller is a software element extended and developed from the Floodlight SDN controller to handle IoT devices and their service orchestration. It communicates with its connected SD-IoT clusters.

The SD-IoTC controller houses a set of components (Figure 5). These functions allow the SD-IoTC controller to i) process IoT requests coming to the LSSD-IoT system, ii) control, manage, and orchestrate IoT clusters/devices, and iii) store temporary IoT services that can be shared between multiple IoT applications. Details of each additional component are as follows.

SD-IoT Service via Web-GUI is an interface for users to specify IoT demands for the LSSD-IoT system. It also displays available IoT services and the status of IoT service provision.

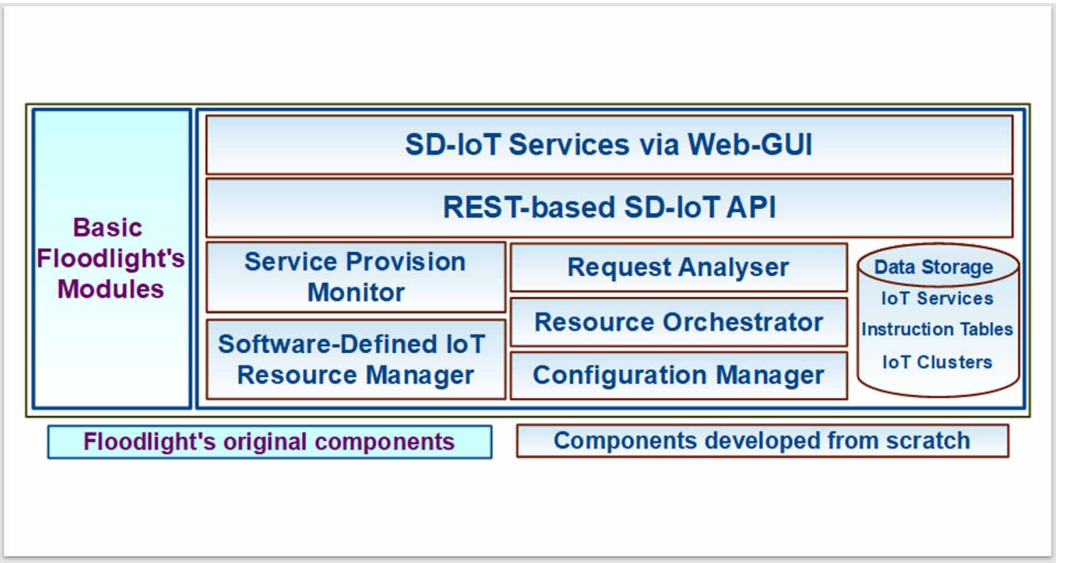
REST-based SD-IoT API is a Northbound Interface that provides abstractions of IoT resources to the application layer.

Request Analyser analyses input requests and provides specific requirements for the Resource Orchestrator.

Software-defined IoT Resource Manager manages SD-IoT resources. It leverages SDN devices to update connected SDIoT clusters and their capabilities. The statistics are collected from other module applications (basic Floodlight's Modules).

Resource Orchestrator orchestrates SD-IoT clusters to provide required IoT services. In accordance with IoT requirements and available SD-IoT resources, it selects the most appropriate SD-IoT cluster to handle the IoT request.

Figure 5. SD-IoTC controller architecture



Configuration Manager programs the core network and SD-IoTD controllers according to the instructions from the Resource Orchestrator. The Configuration Manager leverages the SDN resources to forward IoT requests to appropriate SDIoT clusters. It computes the flows for delivering IoT service requests and IoT service results to the intended destinations.

Data storage stores information about the SD-IoTD systems connected to the LSSD-IoT system. The information includes locations of SD-IoTD clusters, their service capability, and temporary collected data.

Service Provision Monitor monitors and updates the status of the resources being used in the service provisioning. Moreover, it always checks the response from an SD-IoT cluster to see if it can handle the allocated task and tells the Resource Orchestrator to re-schedule the task if necessary.

4.1.2. SD-IoTC Controller Orchestration Mechanism

The SD-IoTC controller orchestrates services based on the remaining resources of the clusters, resources and services already provisioned. The operation is illustrated via the pseudo-code below. The algorithm is for orchestrating the IoT services on demand. It aims to efficiently utilize existing IoT resources by sharing their resources in response to on demand IoT services. Some sets of parameters have been used for the Orchestration mechanism. Regarding the input, there is a set of IoT requests (SRQ) asking for IoT services. The expected output should be a list of SD-IoT clusters (sdiot identification as sdiot name) and associated requests (sdiot request). Each input request of SRQ is analyzed to know the demand from users like required services (sid) in which location (reqLoc), actions associated with the service (reqAct), how long the service is demanded (reqPeriod), and how often data related to the service is sent to the user (reqFreq), and where is the data collection point (reqDstAddr). According to the requirements, the orchestration algorithm process each request (RQ: sdiot request) from the incoming set of requests (SRQ) and get a list (L1) of sdiot requests having the same requirements including reqAct and sid. Based on the information, the algorithm figures out IoT clusters (sdiot cluster) being able to provide the requested services. The orchestrator connects to the database to check the capability of the potential candidates (L2) in the database to choose the best candidate to take responsible for the associated request.

4.2. Resource Orchestration Mechanism

```
Input: a set of sdiot requests SRQ, incoming IoT request RQ, and
updated SD-IoT resources RS
Output: a set of SD-IoT clusters (sdiot name) and associated
requests (sdiot request)
Switch required action (reqAct) in RQ do
case ”GET”:
for each required service (sid) in RQ do
L1 = getSdiotReqWithGET(SRQ,sid,reqAct); //get a list of sdiot
requests having the same requirements for sid and reqAct
if the size of L1 is 0 then
L2 = getListSdiotByAreaIdAndSid(list sdiot clusters,sid,areaId);//
get a list of SD-IoT clusters with related sid and AreaId
Best sdiot = PickBestSdiotFromL2 (L2,sid)//get the SD-IoT cluster
with least task from L2
else if the size of L1 is 1 then
L3 = getSdiotWithGET(L1);//get SD-IoT cluster’s name
Best sdiot = L3
else
L4 = getListSdiotWithGET(L1)//get a list of SD-IoT cluster
involved in the reqAct
Best Sdiot = PickBestSdiotWithLeastState(L2);//pick the SD-IoT
cluster with least task
for each sdiot name in L4 do
if the sdiot name is Best Sdiot do
Best sdiot = sdiot name
end if
end for
end if
update the list of (sdiot name, sdiot requests)
end for
case ”SET ON”:
for each required service (sid) in RQ do
L2 = getListSdiotByAreaIdAndSid(list sdiot clusters,sid,areaId);//
get a list of SD-IoT clusters with required sid and area
Pick best sdiot from L2
update the list of (sdiot name, sdiot requests)
end for
case ”SET OFF”:
for each required service (sid) in RQ do
L2 = getListSdiotByAreaIdAndSid(list sdiot clusters,sid,areaId);//
get a list of SD-IoT clusters with required sid and area
Pick best sdiot from L2
get the list of (sdiot name, sdiot requests)
end for
return the list of (sdiot name, sdiot requests)
```

This means that if an IoT request needs an IoT service that is currently provisioned for another request, the controller reuses, if it can be accommodated, for the incoming request without further configuration on the underlying SD-IoT resources. Moreover, if the SD-IoTC controller receives

a response from an SD-IoTD controller that it cannot achieve the required services, the SD-IoTC controller re-orchestrates other SD-IoTC clusters to handle the related requests.

4.3. Communication Interfaces

The communication between the application layer and SD-IoTC control layer is via a Northbound Interface (NBI), e.g., REST-based API. The SD-IoTC control layer communicates with the cluster layer through a Southbound Interface (SBI).

4.4. SD-IoT Clusters

An SD-IoT cluster consists of an SDN switch and a host representing an SD-IoT platform. SDN switches are networking devices connecting SD-IoT platforms to the LSSD-IoT system. They report on the connected IoT platforms. They allow the SD-IoTC controller to configure data flows between IoT clusters to deliver IoT requests to a proper SD-IoT platform or returned results to data collection points. Hosts connected to SDN switches represent IoT clusters that comprise required sensors/IoT devices. The hosts can be considered as SD-IoTC clusters that represent the underlying SD-IoT platform.

5. SOFTWARE-DEFINED DEVICE LAYER

This layer contains many clusters (SD-IoT clusters). Each cluster is a local platform. All platforms have the same 3-layer architecture: the application layer, the device controller layer, and the IoT device layer (as shown in Figure 6).

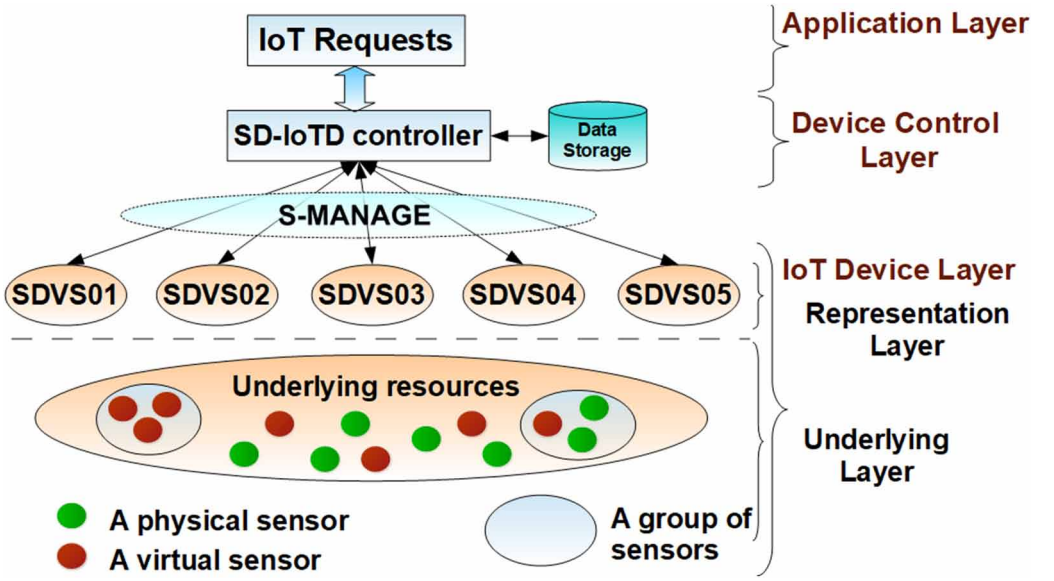
- **Application Layer:** This layer allows developers to deploy their IoT applications by utilizing an abstraction of the underlying IoT infrastructure. The abstraction is provided by the SD-IoTD controller in the orchestration layer. The SD-Cluster layer acts as the application layer to the platform.
- **Device Controller Layer:** This layer accommodates the SD-IoTD controller. It is a bridge between the application and the IoT device layer. It provides the underlying resources with an interface to update their status, attributes, and sensor services. With the knowledge of both requirements of the application layer and capabilities of the IoT resources within the IoT device layer, it can orchestrate IoT services on demand;
- **IoT Device Layer:** Hosts the SD-IoTD resources. This layer contains IoT devices belonging to the clusters. These devices include both virtual and physical devices (sensors, actuators, cyber-physical systems). In particular, virtual devices and IoT devices are software components that can represent plugin physical devices, emulate physical devices, and logical sensing functions. A brief description of a virtual sensor (SDVS) is described in the SDVS section. Communication between the SD-IoTD controller and the SDVS is through a southbound protocol, S-MANAGE as described in the S-MANAGE section.

Essentially, the device layer is designed with two sublayers: the representation layer and the underlying resources layer. The underlying layer contains physical and/or virtual sensors/IoT devices that perform their intended functions. The representation layer is a software component that enriches the devices with additional functionality including programmability, configurability, communications and networking, computation, and storage.

5.1. SD-IoTD Controller

The SD-IoTD controller is a bridge between an IoT cluster and the LSSD-IoT system. With respect to the local IoT cluster, it is the manager that orchestrates, coordinates, and programs the IoT devices within the cluster to provision service demands from the local application. Regarding the LSSD-

Figure 6. SD-IoT model



IoT system viewpoint, the SD-IoTD controller represents its own IoT cluster. It is responsible for reporting its capability in provisioning IoT services on demand and orchestrating its resources to provide requested services (T. M. C. Nguyen, Hoang, & Dang, 2018).

5.2. S-MANAGE Protocol

The S-MANAGE is proposed as a control-and-manage protocol between the SD-IoTD controller and virtual sensors (SDVS) within the representation layer. It enables the SD-IoTD controller to manage and program the SDVSs within the representation layer to achieve the required services and deliver them to the right destinations (Chau Nguyen & Hoang, 2019). This protocol is specifically designed to eliminate the heavy barrier of the OpenFlow protocol and to provide configurability features relevant to IoT constrained devices.

5.3. SDVS

The SDVS is introduced as an interface (or representation layer) between SD-IoTD model and physical sensors/IoT devices. The SDVS is a software entity that functions as a virtual sensor that addresses the limitations of physical sensors/actuators/IoT devices and possesses capabilities for adapting itself in interacting with the surrounding environment and its controller for providing desired services. It provides the SD-IoTD controller with the capability of its represented devices (C. Nguyen & Hoang, 2020).

6. PLATFORM IMPLEMENTATION

6.1. Implementation Platform

The implemented platform integrates both cluster and device layers to form the large-scale IoT on-demand service platform. The SD-IoTC Controller implements all relevant components to interpret application requests, orchestrate, provision, coordinate services of the underlying clusters. An application interface is made available for users to specify their requests for the IoT services and to present the provisioned services to the application. The SD-IoTC clusters are implemented as

software entities that represent individual clusters of IoTs. It includes an information base that holds the knowledge of its local IoT environment, resources, usage. The SD-IoTD Controller implements all local components for provisioning local IoT services on demand (T. M. C. Nguyen et al., 2018). The Coordination protocol is designed for orchestrating sub-requests between the SD-IoTC controller and an SD-IoTC cluster.

At the cluster level, there is a user interface for users entering requests and tables providing information regarding i) available SD-IoT resources, ii) requirements of IoT requests, iii) acknowledgment from IoT applications, and iv) results of resource orchestration and execution status.

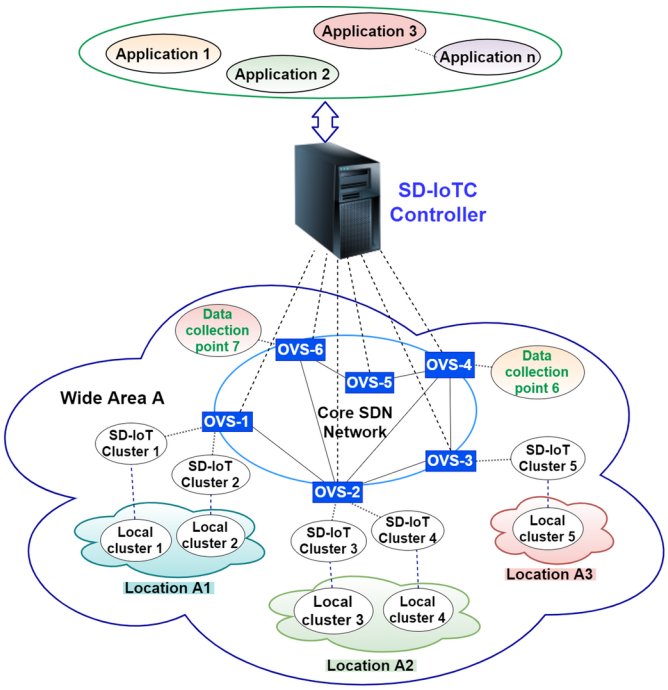
At the device level, there is also a user interface for receiving users demands or requests from the SD-IoTC controller; and tables presenting i) available SDVSs, ii) results of resource orchestration, iii) the number of messages exchanged between each SDVS and the SD-IoTD controller, and iv) SDVS's forwarding table, configuring table and sensor services. The S-MANAGE protocol is deployed for the communication and management between the SD-IoTD controller and its underlying sensors. The SDVS entity is implemented to provide the controller the capability-enriched underlying resources.

We have implemented the SD-IoTC controller as a new application module within the Floodlight controller (Big-Switch-Network, 2019).

6.2. Implementation Scenario

To demonstrate the application of the proposed LSSD-IoT model, we develop a scenario that requires IoT services on demand (as depicted in Figure 7). In the scenario, five local locations are geographically distributed over a wide area, A. An application needs to be developed to provide the weather conditions of these locations, which are located in three areas A1, A2, A3. Ideally, each location should be served by a local IoT system which is more responsive to deal with specific local issues concerning resources, response time, and mobility. LLSD-IoT system is designed to address such an on-demand service by orchestrating and distributing resources and subservices appropriately

Figure 7. Implementation scenario



to locations as needed. The LSSD-IoT system, in this case comprises 5 local IoT subsystems. IoT systems 1 and 2 are in A1. IoT systems 2 and 4 are in A2. IoT system 5 is in A3.

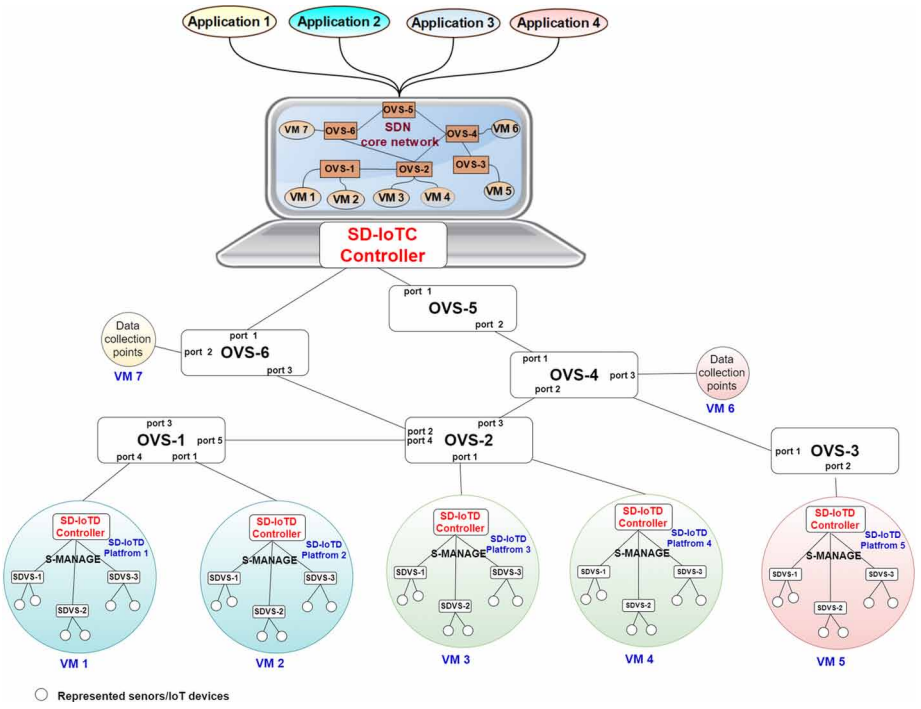
6.3. Implementation Set Up

We implement a prototype as shown in Figure 8. All components of the proposed architecture are built in software. Seven hosts house local IoT clusters or data collection storage. Five SD-IoTD systems run on five hosts, and two IoT storage applications run on other hosts. Each SD-IoTD platform contains a network of five SDVSs, each of which represents five sensor types.

All the main components are built on a PC running Ubuntu 16.04 LTS with detailed configuration: Memory:16GB; Processor: IntelR CoreTM i7-7600U CPU @ 2.8GHz x 4; OS type:64-bit; Disk:235GB.

A Web-GUI is developed by leveraging the bootstrap container based on the Spring framework (https://projects.spring.io/spring-framework/). Users can request for IoT services by accessing the GUI via a link:http://controller-IP-address:8080/ui/pages/sdiot.html. The SD-IoTC controller is extended from the Floodlight SDN controller, a well-known open platform for controlling and managing SDN devices. The SD-IoTC controller is developed in Eclipse Java EE IDE, version: Photon Release (4.8.0). A network of SDN devices is built in the Mininet simulator. In our implementation, we use Open vSwitch 2.5.5 and OpenFlow 1.3 (0x04). SD-IoTD systems are Java-based platform developed in NetBeans 8.2. These systems are run in Mininet hosts connected to SDN devices. It is connected to its own database built in MySQL. We have implemented three main software components of the proposed SD-IoTD model: the SD-IoTD controller, the S-MANAGE protocol, and the SDVS. The databases for SD-IoTC controller and SD-IoTD controllers are built in MySQL, version 5.7.27-0ubuntu0-16.04.1.

Figure 8. Detailed implementation of the LSSD-IoT platform



7. PLATFORM EVALUATION

In this section, we evaluate the proposed model and the implemented platform on two aspects: the implemented capability of the platform for provisioning large-scale IoT services on demand and the performance of the platform in terms of orchestration, coordination, configuration programming, and service provisioning. The implementation results demonstrate the feasibility as well as the efficiency of the proposed model in the provision of IoT services on demand over a large-scale IoT domain.

7.1. Implementation Capability

We evaluate the components and the capabilities that contribute to the provisioning of large scale IoT services on demand. Three aspects are demonstrated: i) Orchestration tasks and allocating of resources, ii) Configuration of resources to perform the allocated tasks, iii) Management of tasks, and the overall service.

The capability is expressed by the following features.

7.1.1. At the Cluster Level, the SD-IoTC's Capabilities

- Update the capability of SD-IoT resources (Figure 10);
- Handle various requests demanding one or multiple IoT services and one or multiple IoT requests at any time according to available SD-IoT resources (Figure 11);
- Orchestrate SD-IoT resources to process one or multiple IoT requests at any time (Figure 13);
- Re-orchestrate an IoT request if any allocated SD-IoT resource cannot handle an assigned request (Figure 13a);
- Program flows to transmit results to desired data collection points and monitor the execution status of the provision of IoT services on demand (Figure 12).

7.2. At the Device Level, the SD-IoTD Controller

- Automatically update its available resources to the SD-IoTC controller (Figure 10);
- Update the capability of SDVSs and the database about the status of the SDVSs (Figure 14a);
- Handle one/multiple requests at any time (Figure 14c);
- Process various types of requests for one or multiple services at any time (Figure 14c);
- Respond to the SD-IoTC controller if it cannot handle an assigned task (Figure 14b);
- Orchestrate SDVSs to achieve required IoT services and send obtained results to desired destinations (Figure 15).

7.3. Interfaces for Control and Management

The programmability and orchestration of the LSSD-IoT model are demonstrated via the capability of orchestrating and configuring the underlying resources for provisioning IoT services on demand. Figure 9 illustrates the overview of orchestration and programmability of the LSSD-IoT model happening at two levels. Three key components involved in the operation, include SD-IoTC controller, SD-IoTD resources, and users/IoT applications. The LSSD-IoT system orchestrates its SD-IoTD resources to provide IoT services for the users.

7.4. Orchestration and Scheduling at Cluster Level

Users are provided with an interface to specify their requirements, e.g., IoT services in which area, required period, how often results being sent to which destination. The users are also provided with available SD-IoT services (Figure 10). The resources are updated whenever an SD-IoTD cluster joins the LSSD-IoT system. In the area A1 (ManagedArea), there are currently five IoT clusters, e.g., sdiot01, sdiot02, with different capabilities (ProvidedServices). Each SD-IoT cluster manages two locations (ManagedLocation), e.g. LOC01 and LOC02.

Figure 9. Level of programmability and orchestration of the LSSD-IoT platform

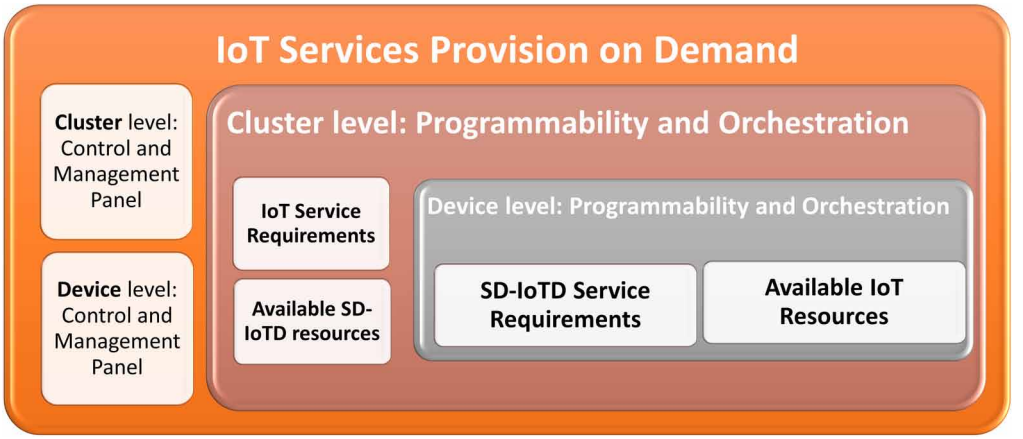


Figure 10. Available SD-IoT resources

SDIoT Resource

Cluster_Name	ManagedArea	ManagedLocation	ProvidedServices
sdiot01	Area1	LOC01,LOC02	SID01,SID03,SID05
sdiot02	Area1	LOC01,LOC02	SID02,SID03,SID04
sdiot03	Area1	LOC01,LOC02	SID01,SID02,SID03,SID04,SID05
sdiot04	Area1	LOC01,LOC02	SID01,SID02,SID03,SID04
sdiot05	Area1	LOC01,LOC02	SID02,SID03,SID04,SID05

The LSSD-IoT handles one/multiple requests that are marked with IoTReq ID at any time (Start Time) (Figure 11). The input requests may require one or multiple IoT services (Req Service). The user monitors the status of IoT service achievement (Completion Status) of required services if it is OnGoing or Completely Done. OnGoing indicates that the systems are still obtaining required services, while Completely Done shows that all required services have been sent to desired destinations after a required period. As for Completely Done status, take IoTReq ID number 2 for example, all sub-requests of request number 2 have been Completely Done, (Figure 13b) so the Completion Status of the request is Completely Done (Figure 11). Meanwhile, regarding the IoTReq ID number 6, only two out of five sub-requests have been done (Figure 13b), so its Completion Status is still OnGoing (Figure 11).

The LSSD-IoT system configures and monitors data flows between sources of IoT services (Sender of Results) and destinations of IoT results (IoT Application Receiver). It also provides information on the achieved services (Required Sdiot Services). The records are shown in Figure 12. By default, SD-IoTD clusters can achieve all required services, but if any SD-IoTD cluster cannot handle a request, it sends a response (Sdiot Response) regarding its incapability to the SD-IoTC controller. Hence, the SD-IoTC controller re-orchestrates other SD-IoTD resources to process the request (Figure 13a). As shown in Figure 13a, IoT request number 4 cannot be handled by sdiot01, so sdiot05 is orchestrated to handle request number 4. Thus, after re-scheduling the resources, the SD-IoTC controller set the value for the Sdiot Response to New replacement of sdiot01 for IoT Req ID 4.

Figure 11. IoT request status

IoT Requests

IoTReq_ID	Req_Area	Req_Service	Start_Time	Completion_Status
1	Area1	SID01	1571550404154	Completely Done
2	Area1	SID04,SID02	1571550416288	Completely Done
3	Area1	SID03,SID04,SID05	1571550427916	OnGoing
4	Area1	SID03,SID04,SID05	1571550441393	OnGoing
5	Area1	SID01,SID02,SID03,SID04,SID05	1571550451705	OnGoing
6	Area1	SID05,SID03,SID04,SID01,SID02	1571550462736	OnGoing

Figure 12. IoT application announcements

IoT Application Announcements

IoT_Application_Receiver	Sender_of_Results	Required_Sdiot_Services
dest6	From sdiot03	results about services SID01 for big request ID number 1
dest6	From sdiot03	results about services SID05 for big request ID number 1
dest7	From sdiot01	results about services SID01 for big request ID number 2

The SD-IoTC controller updates the status of SD-IoTD resources allocated to an IoT request at a specific time (Figure 13b). For an IoT request (IoTReq ID), a set of information associated with each IoT service (Req Service) is recorded: the responsible SD-IoTD cluster (Allocated Cluster), the response status (Sdiot Response), the execution status (IsExecuted), and the processing status (Status) which is "OnGoing" or "Completely done". When an SD-IoTD cluster completes a task, it informs the SD-IoTC controller of its completion and marks the Status "Completely Done."

7.5. Orchestration and Scheduling at Device Level

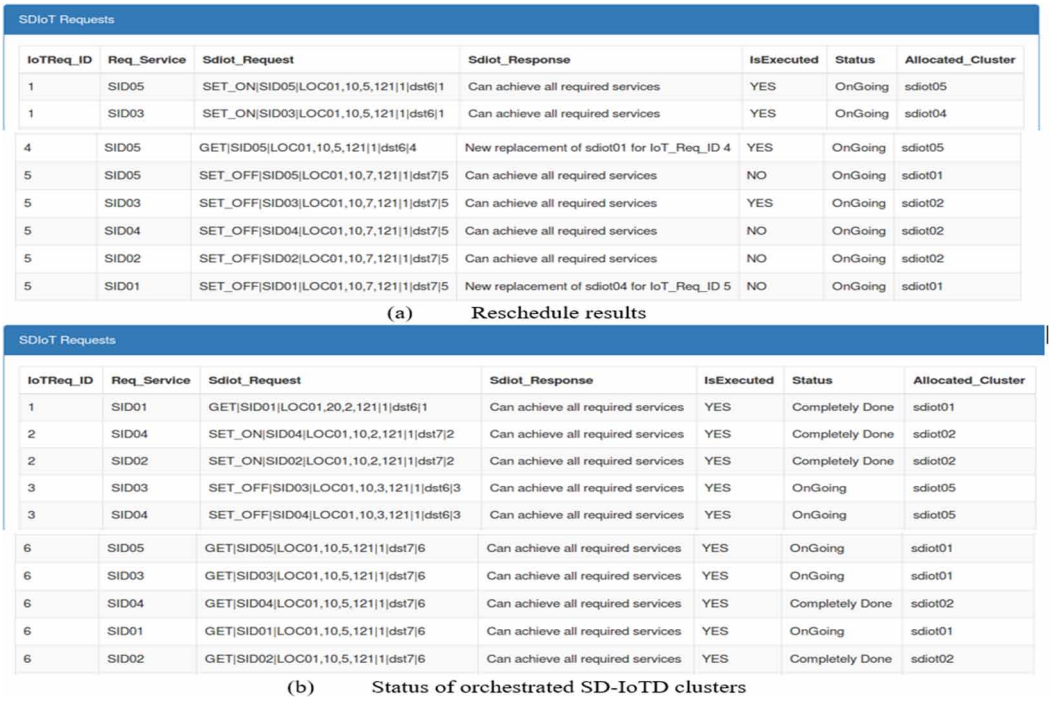
The control and management at each SD-IoTD cluster have been shown in the control panel of the SD-IoTD controller (Figure 14 and Figure 15).

The management panel includes three main windows presenting information collected from the application layer, the orchestration layer, and the device layer. The application window allows users to input their IoT requests or admit IoT requests from the SD-IoTC controller via a user interface. The operation of the SD-IoTD controller is illustrated via multiple tables. The first table (Figure 14a) shows all SDVSs under the controller's management. For example, SDVS01 in location LOC01 has the State 9 which means the SDVS currently handles 9 tasks. The second table (Figure 14b) displays appropriate SDVSs for a service request. For instance, SDVS01 is orchestrated to provide services, including SID01, 02, 03, 04, 05. The third table (Figure 14c) reveals the results with resources provided for application requests. This table shows that SDVS01 has achieved the required services (IsExecuted) and sent results (Results) to the required destination. The last table (Figure 14d) displays the number of packets sent out by SDVSs. SDVS01 has sent 51 data packets.

The device window is about the SDVS. It provides information regarding the Forwarding table, Configuring Table, and Sensor services provided by the SDVS.

The SDVS can be configured to provide desired services (Figure 15c) and forward data to the required destination (Figure 15a). The SDVS shows a list of capable services and their associated status (Figure 15b). It also provides the details of each provisioned service (Figure 15a), including the processing start time (Start-Time), the processing duration (RunTime), and the remaining time

Figure 13. SD-IoTC controller orchestration



SDIoT Requests

IoTReq_ID	Req_Service	Sdiot_Request	Sdiot_Response	IsExecuted	Status	Allocated_Cluster
1	SID05	SET_ON|SID05|LOC01,10,5,121|1|dst6|1	Can achieve all required services	YES	OnGoing	sdiot05
1	SID03	SET_ON|SID03|LOC01,10,5,121|1|dst6|1	Can achieve all required services	YES	OnGoing	sdiot04
4	SID05	GET|SID05|LOC01,10,5,121|1|dst6|4	New replacement of sdiot01 for IoT_Req_ID 4	YES	OnGoing	sdiot05
5	SID05	SET_OFF|SID05|LOC01,10,7,121|1|dst7|5	Can achieve all required services	NO	OnGoing	sdiot01
5	SID03	SET_OFF|SID03|LOC01,10,7,121|1|dst7|5	Can achieve all required services	YES	OnGoing	sdiot02
5	SID04	SET_OFF|SID04|LOC01,10,7,121|1|dst7|5	Can achieve all required services	NO	OnGoing	sdiot02
5	SID02	SET_OFF|SID02|LOC01,10,7,121|1|dst7|5	Can achieve all required services	NO	OnGoing	sdiot02
5	SID01	SET_OFF|SID01|LOC01,10,7,121|1|dst7|5	New replacement of sdiot04 for IoT_Req_ID 5	NO	OnGoing	sdiot01

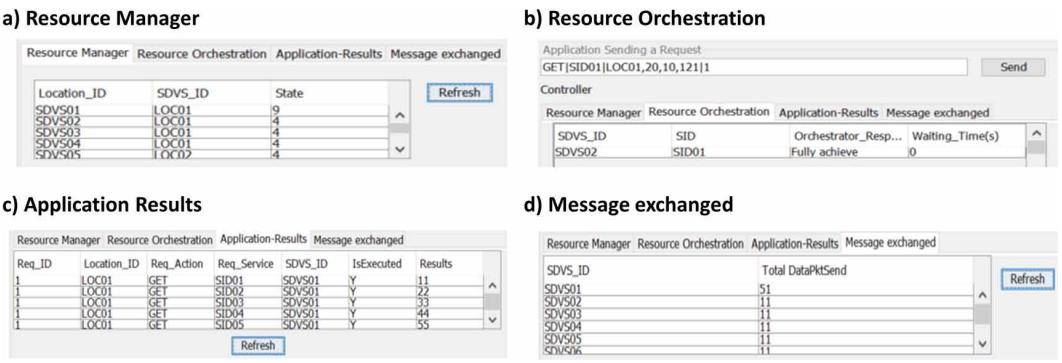
(a) Reschedule results

SDIoT Requests

IoTReq_ID	Req_Service	Sdiot_Request	Sdiot_Response	IsExecuted	Status	Allocated_Cluster
1	SID01	GET|SID01|LOC01,20,2,121|1|dst6|1	Can achieve all required services	YES	Completely Done	sdiot01
2	SID04	SET_ON|SID04|LOC01,10,2,121|1|dst7|2	Can achieve all required services	YES	Completely Done	sdiot02
2	SID02	SET_ON|SID02|LOC01,10,2,121|1|dst7|2	Can achieve all required services	YES	Completely Done	sdiot02
3	SID03	SET_OFF|SID03|LOC01,10,3,121|1|dst6|3	Can achieve all required services	YES	OnGoing	sdiot05
3	SID04	SET_OFF|SID04|LOC01,10,3,121|1|dst6|3	Can achieve all required services	YES	OnGoing	sdiot05
6	SID05	GET|SID05|LOC01,10,5,121|1|dst7|6	Can achieve all required services	YES	OnGoing	sdiot01
6	SID03	GET|SID03|LOC01,10,5,121|1|dst7|6	Can achieve all required services	YES	OnGoing	sdiot01
6	SID04	GET|SID04|LOC01,10,5,121|1|dst7|6	Can achieve all required services	YES	Completely Done	sdiot02
6	SID01	GET|SID01|LOC01,10,5,121|1|dst7|6	Can achieve all required services	YES	OnGoing	sdiot01
6	SID02	GET|SID02|LOC01,10,5,121|1|dst7|6	Can achieve all required services	YES	Completely Done	sdiot02

(b) Status of orchestrated SD-IoTD clusters

Figure 14. SD-IoTD controller orchestration



(TTL). Moreover, it shows the number of requests for a service type. With the recorded information, the SD-IoTD controller can notify the SD-IoTC controller of the waiting time before the request can be processed.

7.6. Platform Performance

In this section, we carry out the tasks of service provisioning and evaluate the performance through two performance measures: orchestration time and response time. Orchestration time is the time required for orchestrating a requested service at the cluster level. Response time is the service response time, which measures the time from when the SD-IoTC controller configures SD-IoT resources at the device level to the time the first lot of data is collected at the collection point. As illustrated in Figure 16, the

Figure 15. Device configuration

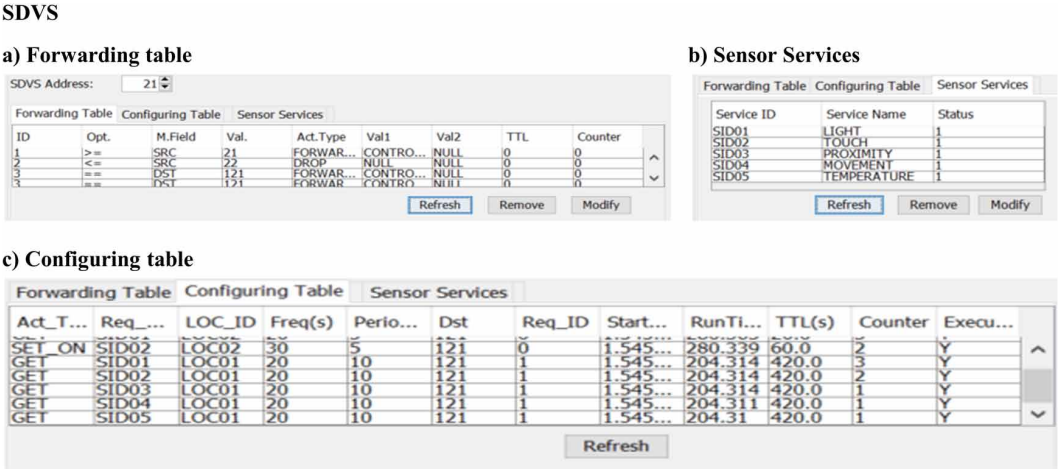
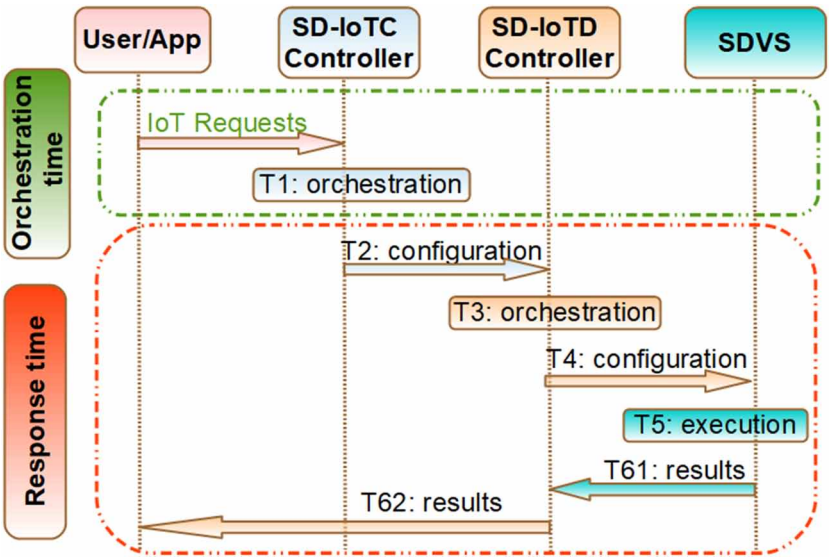


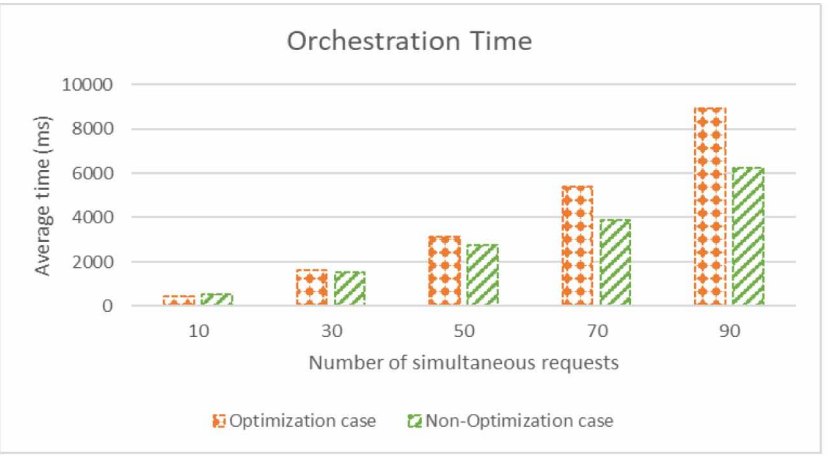
Figure 16. Timing diagram



Orchestration time is T1, the time needed for orchestrating SD-IoT clusters for serving an IoT request. Total Response time is composed of T2, T3, T4, T5, T61, and T62.

The efficiency is examined via the above two measurements. The metrics are investigated through two test cases: a) with optimization and b) without optimization. With the optimized orchestration, the SD-IoTC controller reuses existing configurations/results associated with an IoT service to provision multiple IoT service demands that share the same services. The LSSD-IoT system can save time in configuring SD-IoT resources to respond to multiple IoT requests and the reduction in the load over the transport network since the model may only need to configure IoT clusters once to achieve services required by multiple requests. On the other hand, for the non-optimization case, the SD-IoTC controller needs to orchestrate available SD-IoT resources and configure them for every incoming request.

Figure 17. Orchestration time with and without optimization



The orchestration is executed according to the availability of IoT resources and associated achieved IoT services, and current IoT requests. At an arrival time of a request, the orchestrator has to determine the amount of resources that have been allocated to existing requests and can only allocate residual resources from its resource pool to the new request.

We examine the orchestration for two cases a) with optimization (optimization case) and b) without optimization (non-optimization case). A number of simultaneous requests have been sent to the LSSD-IoT system. The requests may have one, two, three, or four common IoT services. As displayed in Figure 17, 18a, and 18b, for the two cases, the time needed for orchestrating and provisioning increases with the number of requests. Compared to the non-optimization case, the average time for orchestrating the same number of simultaneous requests for the optimization case is longer due to the computation for re-utilizing available IoT services that can be reused for provisioning IoT services to multiple IoT requests.

In contrast, the response time of the non-optimization case is much longer than that of the optimization case for provisioning IoT services to the same number of IoT requests (Figure 18). Particularly, for the optimization case, the maximum amount of time for responding 90 concurrent requests is about 50000ms, while the non-optimization case requires about 249 times higher. This is because the optimization case a) reuses available results of IoT services and shares them with multiple IoT applications, and b) minimizes the configuration for the same service type. The non-optimization case does not consider these factors, thus the orchestrating and configuring underlying resources need to be performed for every incoming request.

To account for the rise of the orchestration time and response time for the optimization case, we investigate the response time for 4 cases, as shown in Figure 19. A variety of requests are sent to the LSSD-IoT platform simultaneously. These requests may have a common interest for one, two, three, or four services. As shown in Figure 19, the more services required per request, the more time needed for orchestration as well as service response. In addition, for a higher number of concurrent requests coming to the systems, the time for processing them becomes longer.

8. CONCLUSION

In this paper, we have introduced an LSSD-IoT model with new concepts to reshape the SDN and NFV technologies and overcome the limitations of IoT networked devices to support the programming of IoT services on demand. In particular, we unified the central SDN and the local SD-IoT platforms

Figure 18. Response time with and without optimization

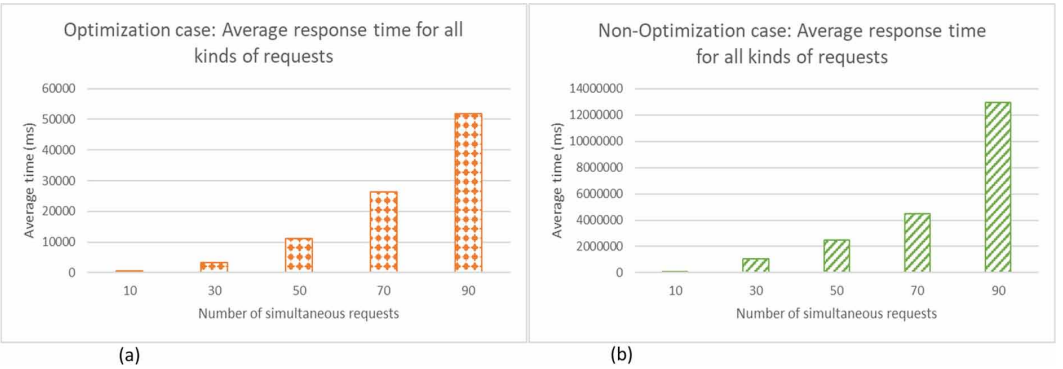


Figure 19. Orchestration time and response time for the optimization case with various input requests

for scalability in terms of the number of IT devices and their geographical coverage for programming global IoT services on demand. We developed a new SD-IoTC controller to enable it to control and manage both SDN devices and IoT devices. We developed the local SD-IoTD platform for modularized IoT domains. In addition, the new concept of software-defined virtual sensors was introduced and the effective southbound protocol (S-MANAGE) was deployed for both networked and IoT devices behavior programming and configurational management. We implemented and comprehensively evaluated the LSSD-IoT platform for end-to-end IoT applications. Our research opens up new research directions for end-to-end control and management of SDN-NFV-based IoT infrastructure in the provision of IoT services on demand. Important directions may include:

- **Intelligent QoS Schemes and Efficient Use of Sharing Resources:** As billions of devices produce a massive volume of data that causes a serious load on the transporting network. With the proposed LSSD-IoT model, QoS schemes to avoid conflicts among requirements from various IoT demands and improve the orchestration mechanism to maximize the response time in the provision of IoT services on demand;
- **Security of Shared IoT Platform:** At the cluster level, trust needs to be established between the LSSD-IoT platform and the SD-IoT platform when a local IoT system first joins a large-scale IoT platform. At the device level, mechanisms need to be developed to ensure that each IoT device in an IoT cluster is trusted for and protected from sharing their IoT services with multiple IoT applications;
- **Discovery of IoT Resources:** As the proposed SDVS acts as a programmable interface between the SD-IoT model and heterogeneous underlying IoT devices, an effective discovery mechanism to allow mobile IoT devices to easily participate in a local IoT system to share their IoT values.

REFERENCES

Alam, I., Sharif, K., Li, F., Latif, Z., Karim, M. M., Biswas, S., & Wang, Y. et al. (2020). A Survey of Network Virtualization Techniques for Internet of Things Using SDN and NFV. *J ACM Comput. Surv., 53*(2). Advance online publication. doi:10.1145/3379444

Big-Switch-Network. (2019). *Floodlight: Floodlight Is an Open SDN Controller*. Retrieved from http://www.projectfloodlight.org/floodlight/

Chendeb, N., Agoulmine, N., & El-Assaad, M. (2020). *Inspiring from SDN to Efficiently Deploy IoT Applications in the Cloud/Fog/IoT ecosystem.* Paper presented at the 8th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2020), Cancún, Mexico.

Do, S., Le, L. V., Paul Lin, B. S., & Tung, L.-P. (2019). SDN/NFV Based Internet of Things for Multi-Tenant Networks. *Transactions on Networks and Communications, 6*(6). 10.14738/tnc.66.5695

Framingham, M. (2019). *The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast*. Retrieved from https://www.idc.com/getdoc.jsp?containerId=prUS45213219

Habibi, P., Baharlooei, S., Farhoudi, M., Kazemian, S., & Khorsandi, S. (2018). *Virtualized SDN-Based End-to-End Reference Architecture for Fog Networking.* Paper presented at the 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA).

Jain, R., Jain, N., & Nayyar, A. (2020). *Security and Privacy in Social Networks: Data and Structural Anonymity. In Handbook of Computer Networks and Cyber Security*. Springer.

K, K. O., Mansoor, A. M., Ahmad, R., & Kim, S. (2018). *Efficient Deployment of Service Function Chains (SFCs) in a Self-Organizing SDN-NFV Networking Architecture to Support IOT.* Paper presented at the 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN).

Kamyabpour, N., & Hoang, D. B. (2010). *A Hierarchy Energy Driven Architecture for Wireless Sensor Networks.* Paper presented at the 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops. doi:10.1109/WAINA.2010.46

Krishnamurthi, R., Nayyar, A., & Solanki, A. (2019). *Innovation Opportunities through Internet of Things (IoT) for Smart Cities. In Green and Smart Technologies for Smart Cities*. CRC Press.

Mouradian, C., Jahromi, N. T., & Glitho, R. H. (2018). NFV and SDN-Based Distributed IoT Gateway for Large-Scale Disaster Management. *IEEE Internet of Things Journal*, *5*(5), 4119–4131. doi:10.1109/JIOT.2018.2867255

Muñoz, R., Vilalta, R., Yoshikane, N., Casellas, R., Martínez, R., Tsuritani, T., & Morita, I. (2018). Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources. *Journal of Lightwave Technology*, *36*(7), 1420–1428. doi:10.1109/JLT.2018.2800660

Nayyar, A., Jain, R., Mahapatra, B., & Singh, A. (2019). *Cyber Security Challenges for Smart Cities. In Driving the Development, Management, and Sustainability of Cognitive Cities*. IGI Global.

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

Nguyen, C., & Hoang, D. (2020). *Software-Defined Virtual Sensors for Provisioning IoT Services on Demand.* Paper presented at the 2020 5th International Conference on Computer and Communication Systems (ICCCS).

Nguyen, T. M. C., Hoang, D. B., & Dang, T. D. (2018). *A software-defined model for IoT clusters: Enabling applications on demand.* Paper presented at the 2018 International Conference on Information Networking (ICOIN). doi:10.1109/ICOIN.2018.8343223

Ojo, M., Adami, D., & Giordano, S. (2016). *A SDN-IoT architecture with NFV implementation.* Paper presented at the 2016 IEEE Globecom Workshops (GC Wkshps). doi:10.1109/GLOCOMW.2016.7848825

Ramachandran, G. S., & Krishnamachari, B. (2019). Towards a large scale iot through partnership, incentive, and services: A vision, architecture, and future directions. *Open Journal of Internet Of Things*, *5*(1), 80–92.

Solanki, A., & Nayyar, A. (2019). *Green internet of things (G-IoT): ICT technologies, principles, applications, projects, and challenges. In Handbook of Research on Big Data and the IoT*. IGI Global. doi:10.4018/978-1-5225-7432-3.ch021

Yu, R., Xue, G., Kilari, V. T., & Zhang, X. (2018). The Fog of Things Paradigm: Road toward On-Demand Internet of Things. *IEEE Communications Magazine*, *56*(9), 48–54. doi:10.1109/MCOM.2018.1701140

Chau Nguyen is currently a Ph.D. candidate in the School of Electrical and Data Engineering, Faculty of Engineering and Information Technology, University of Technology Sydney (UTS). She received the BSc in Electronics and Telecommunications from Vietnam National University, Ho Chi Minh City University of Natural Sciences, Vietnam in 2010, and the M.Sc. in Telecommunications Engineering from the University of Sunderland, the United Kingdom in 2014. Her research interests include IoT, WSNs, and SDN.

Doan B. Hoang is a professor in the School of Electrical and Data Engineering, Faculty of Engineering and Information Technology, University of Technology Sydney (UTS). He leads the Virtualized Infrastructures and Cyber Security (VICS) research group. His current research interest includes: Optimization of Software-defined infrastructures (Cloud, SDN, NFV, and IoT) and services, developing Security capability maturity models and metrics for Cyber Security, modeling IoT security and trust assessment, and IT models for Assistive Healthcare. Professor Hoang has published over 230 research papers. Before UTS, he was with Basser Department of Computer Science, University of Sydney. He held various visiting positions: Visiting Professorships at the University of California, Berkeley; Nortel Networks Technology Centre in Santa Clara, the University of Waterloo, Carlos III University of Madrid, Nanyang Technology University, Lund University, and POSTECH University.