Computers, Materials & Continua
DOI:10.32604/cmc.2021.017528
Article

Tech Science Press

Towards a Dynamic Virtual IoT Network Based on User Requirements

Faisal Mehmood[1], Shabir Ahmad[2,3], Israr Ullah[1], Faisal Jamil[1] and DoHyeun Kim[1,*]

[1]Department of Computer Engineering, Jeju National University, Jeju, 63243, Korea
[2]Department of I.T. Convergence Engineering, Gachon University, Seongnam-Si, 461-701, Korea
[3]Department of Software Engineering, University of Engineering & Technology Mardan, Mardan, 23200, Pakistan
[*]Corresponding Author: DoHyeun Kim. Email: kimdh@jejunu.ac.kr
Received: 02 February 2021; Accepted: 07 April 2021

Abstract: The data being generated by the Internet of Things needs to be stored, monitored, and analyzed for maximum IoT resource utilization. Software Defined Networking has been extensively utilized to address issues such as heterogeneity and scalability. However, for small-scale IoT application, sometimes it is considered an inefficient approach. This paper proposes an alternate lightweight mechanism to the design and implementation of a dynamic virtual network based on user requirements. The key idea is to provide users a virtual interface that enables them to reconfigure the communication flow between the sensors and actuators at runtime. The throughput of the communication flow depends on the data traffic load and optimal routing. Users can reconfigure the communication flow, and virtual agents find the optimal route to handle the traffic load. The virtual network provides a user-friendly interface to allow physical devices to be mapped with the corresponding virtual agents. The proposed network is applicable for all systems that lie in the Internet of Things domain. Results conclude that the proposed network is efficient, reliable, and responsive to network reconfiguration at runtime.

Keywords: Internet of things; software-defined networks; virtual network; inter-process communication; cloud of things

1 Introduction

The Internet of Things (IoT) is a system of interconnected computing devices that is capable of exchanging data over a network without requiring human-to-computer or human-to human interference [1]. With the development of IoT, there has been a significant increase in data traffic from various applications in different domains. IoT applications promise to bring immense value to our lives. IoT could be the next frontier in the race due to its revolutionary computing capabilities. The rising influence of the Internet has made connectivity ubiquitous. IoT has applications in industrial and domestic fronts such as smart homes, smart farming, smart cities, health care and industrial automation [2].

IoT applications promise to reshape entire industries and bring immense value to our lives [3]. IoT provides services in many fields such as healthcare, finance, retail, and manufacturing.

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

There are some advantages of IoT such as it can access information from anywhere, it has also improved communication between connected computing devices, and automating tasks reduced human interaction. There are some challenges as well such as IoT devices generate a huge amount of data that requires a secure and reliable mode of communication and storage [4]. If one device is corrupted, most chances are that the connected devices will also be corrupted. IoT does not have any international standard for compatibility, so it is hard for devices from different vendors to communicate with each other. In the proposed system we have provided the solution to the above-mentioned problems. The virtual network is the virtual representation of the physical IoT network. Each physical device has a corresponding virtual agent in the virtual network.

The communication between physical devices is performed via the virtual agent—the performance of the protocol matters in a real-time system for accuracy and efficiency. We have used a lightweight Message Queuing Telemetry Protocol (MQTT) protocol for communication between physical devices and virtual agents. Each virtual agent is an independent process running in the virtual network. Virtual agents communicate and share data via an inter-process communication (IPC) mechanism.

Network configuration is the process of setting a network's flow, control, and operation to support an organization's network communication. By network configuration, we mean to say the communication flow between IoT devices in the network. This paper aims to design a virtual network that allows users to dynamically change the network configuration based on user requirements. The user can register physical devices by providing basic details such as device name, device type, device model. The coordinator will initialize the virtual object for the corresponding registered devices. The proposed system provides a user-friendly web interface to map physical devices with virtual objects. Mapping physical devices with virtual objects will allow them to communicate with each other by assigning a unique topic. After mapping devices with virtual objects, the user can control the network communication flow by creating a route for the devices in the virtual network. Each virtual object is an independent process running in the virtual network; if for some reasons (device failure, link failure), the virtual object is inactive, the virtual object can communicate with the virtual destination object via an alternative route. If the destination physical device is inactive, the coordinator will keep sending packets to the device until it is active. In this study, the network's flow and control can be dynamically changed based on the user requirement.

IoT is emerging and evolving rapidly. Various technical solutions are proposed for multiple purposes. The most commonly used approach is the concept of Software-Defined Networks (SDN). While SDN offers many advantages to IoT applications in terms of heterogeneity, flexibility, and reconfigurability, however, for small-scale applications, it is considered inefficient and thus, a lightweight SDN-inspired virtual networking approach remained an unaddressed challenge. During this study, some research gaps were identified, such as IoT device compatibility issues, IoT security, and IoT virtualization. This study focuses on the virtual IoT network by proposing a mapping and routing mechanism for IoT devices. The contribution of this paper is as follows:

- Introduce a mapping mechanism between the physical device and virtual agent. The relationship can be one-to-one, one-to-many, and many-to-many between physical devices and virtual agents.
- Introduce dynamic routing mechanism based on user requirements between virtual agents in the IoT network.
- Finally, implement the coordinator in a virtual IoT network for inter-process communication between virtual agents.

The rest of this paper is organized as follows. In Section 2, the Literature review is described briefly. In Section 3, the proposed architecture of the virtual IoT network is explained. Section 4 presents the implementation of the virtual IoT network, and Section 5 elucidates the experimental environment. Section 6 exhibits the results achieved during the experiment, and Section 7 concludes the paper and highlights future directions.

2 Related Work

A virtual object [5,6] is an object that does not exist in the real world. It is like a substitute or virtual representation, a semantically enriched real object which can acquire, analyze, and interpret information about its context to provide services. According to recent studies [6–9], a virtual object can be regarded as the physical sensor and actuator devices' digital counterparts. Virtualization can make heterogeneous objects inter-operable through the use of semantic descriptions, enabling them to acquire, analyze, and interpret information about its context to make decisions and act upon the virtual objects.

Wireless Sensor Networks (WSNs) are the primary components of the IoT. They are ubiquitous and used in many applications. WSNs nodes are becoming more powerful, and the question arises on how multiple applications can share the same WSN infrastructure. The authors in this paper [10] present a survey on WSN virtualization. Virtualization can be classified into two categories: Node-level virtualization and network-level virtualization. Node-level virtualization allows multiple applications to run their tasks concurrently on a single sensor node, whereas network-level virtualization is the digital representation of a physical network.

If the number of virtual network requests is known, it is called virtual network embedding approaches, but virtual network requests arrive randomly in real situations. The arrival and departure of virtual network requests may affect the resource utilization of physical network infrastructure. The authors in this study [11] proposed a modified genetic algorithm for resource allocation of virtual networks to resolve this issue. The authors reset the parameters and operations corresponding to different network conditions. The results indicate that the proposed algorithm achieves a balanced workload and better convergence.

The controller is the most fundamental part of SDN. The purpose of the controller is to monitor the network traffic and reciprocate to network changes dynamically. Controllers in SDN support a different set of features and affect the quality-of-service (QoS) in SDN. The authors propose a hierarchical control plane-based cluster (HCPC) to overcome the single-point-of-failure (SPOF), performance, and scalability issues. The results in Mininet indicate that the HCPC environment with an optimum controller achieves an improved QoS [12,13].

Many existing systems provide virtualization services for physical devices by creating their virtual objects. Virtual objects are connected to create a network to provide various services. When too many services are composed, management is a challenging task and becomes a cumbersome task. The authors in this study [14] propose' Service and Virtual Objects Management (SVOM),' which effectively organize the services and virtual objects and facilitate the management tasks. The authors performed a simulation of the proposed system for scalability analysis to study the IoT network size's impact on key performance measures like response time, packet delivery ratio, and throughput [15].

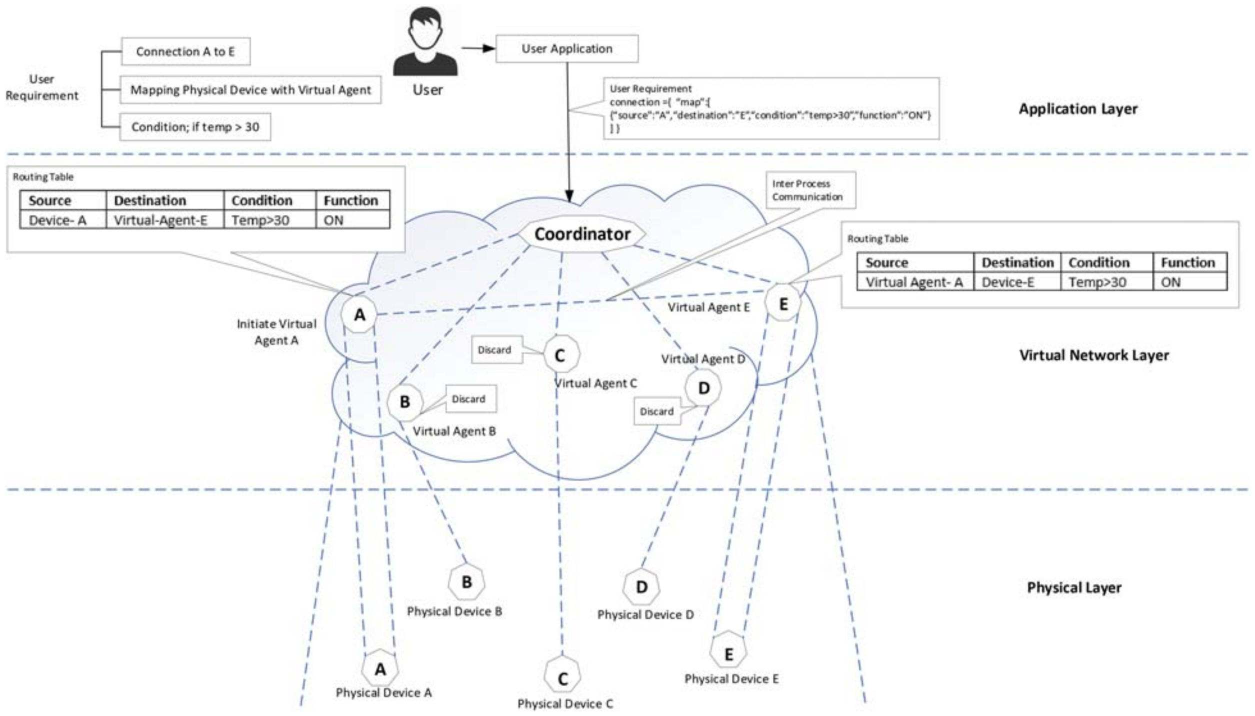
A lot of research has been conducted on IoT device virtualization. In this concept, the IoT devices are accessed through their corresponding virtual objects. In this research work [16], the authors perform a virtual objects network simulation to conduct performance analysis of

IoT network virtualization. The experiments were conducted by varying packet sending rates and network sizes. Results conclude that traffic congestion can be avoided by adjusting data rates by optimizing the virtualization server. The installation of optimized routing rules and policies on routers can be challenging for the controllers. Specific problems may arise, such as link failure identification and restoration. There are several schemes to handle link failure in SDN. The authors performed a comparison between different methods in terms of scalability, routing information access, latency, robustness, configuration, overheads of routing, controller, and switches. A simulation was performed of the Naval tactical networks and DCN using the ODL controller for reactive and proactive approaches to determine the recovery time and throughput comparison. Results indicated that the reactive approach caused an extra burden due to controller intervention [17,18].

According to our best knowledge, several existing approaches fail to provide dynamic mapping and routing mechanisms between physical devices and their corresponding virtual agents. The devices are either mapped based on a one-to-one or one-to-many relationship. The proposed mapping algorithm allows users to dynamically map one-to-one, one-to-many, and many-to-many relationships based on their requirements and enable users to control the communication flow between devices and virtual agents. This study attempts to propose a novel approach towards a dynamic virtual IoT network based on user requirements.

3 System Model

This work's primary purpose is to design a virtual IoT network that can be dynamically configured based on the user requirement. This section briefly explains the proposed architecture of the virtual network. In Fig. 1, there are three layers: Physical layer, virtual network layer, and application layer.



```
User Requirement
connection ={ "map":[
{"source":"A","destination":"E","condition":"temp>30","function":"ON"}
] }
```

Routing Table

Source	Destination	Condition	Function
Device- A	Virtual-Agent-E	Temp>30	ON

Routing Table

Source	Destination	Condition	Function
Virtual Agent- A	Device-E	Temp>30	ON

Figure 1: Proposed architecture of virtual IoT network

We have used Arduino, Raspberry Pi, and different sensors and actuators in our experiment. The physical layer consists of physical IoT devices such as Raspberry Pi, Arduino, sensors, and actuators. Different sensors and actuators are used in this experimentation. Sensors collect data from the surroundings and exchange data with other devices to perform specific actions, whereas actuators are used to control devices. Actuators receive commands to operate accordingly. As IoT resources have limited memory and processing power, it is not a good approach to execute tasks on the resource end. For this purpose, we have availed cloud computing services. We have used Amazon Web Services (Elastic Compute Cloud EC2) to deploy a virtual network. All the necessary processing is performed on the cloud.

The virtual network layer represents the virtual representation of a physical network layer. The virtual network is deployed on the Amazon cloud, where all the configuration, installation, and deployment are done. In the virtual network layer, the coordinator runs as an independent process responsible for registering physical devices, initializing virtual agents, virtual mapping agents with physical devices, and routing between physical devices via inter-process communication. Each physical device has a corresponding virtual agent in the cloud. The coordinator initializes the virtual agent after registering a physical device and making it. The virtual agent keeps sending packets to the physical device after some interval to get the device's status, whether it is active or inactive. Virtual agents communicate with each other via inter-process communication and communicate with the physical devices via MQTT. The coordinator is the parent process, whereas virtual agents are the child processes. The coordinator controls all the virtual agents; if the virtual agent stops responding, then the coordinator will restart the virtual agent process and update its information in the routing table. After updating the routing table, the virtual agent is synchronized with the physical device, and the network's communication flow is resumed. The coordinator is responsible for registering physical devices, virtual agents' initialization, virtual mapping agents with the physical devices, and creating a route for communication between virtual agents and physical devices. The route is set by the user using a client application and sent to the coordinator. The coordinator validates the route and makes it functional in the virtual network. This study's primary purpose is that the user can dynamically change the flow and control of communication without changing the physical network. The routing table is updated based on user requirements. If the routing path is updated in the middle of communication, the coordinator will update the routing table and virtual agents.

In the application layer, the user has access to the web application to register physical devices. Users can view the list of registered IoT devices and virtual agents. The web application allows the user to map physical devices with the virtual agents and request the coordinator. The coordinator validates and sends back the acknowledgment. After successful mapping, the user can create a route between devices for communication. The route based on user requirement is sent to the coordinator; the coordinator validates the route and sends back the acknowledgment.

Fig. 2 illustrates the proposed system's sequence flow in terms of essential components, i.e., physical network, virtual network, and the coordinator, of the virtual network and its functions. We have presented a case scenario of two devices, i.e., physical device A and physical device E. Each physical device sends a request to the coordinator for registration. The coordinator validates and initiates the corresponding virtual agent. After registering the physical device and initializing virtual agents, the user can view the list of devices by accessing the web application. Users can map the physical devices with the corresponding virtual agent by sending the request to the coordinator, and the coordinator broadcasts the request to the active virtual agents. The virtual agents then map with the physical device and synchronized. The next step is to create a route between physical devices to communicate with each other and perform some specific action. In this figure, the user creates a route between device A and device E.

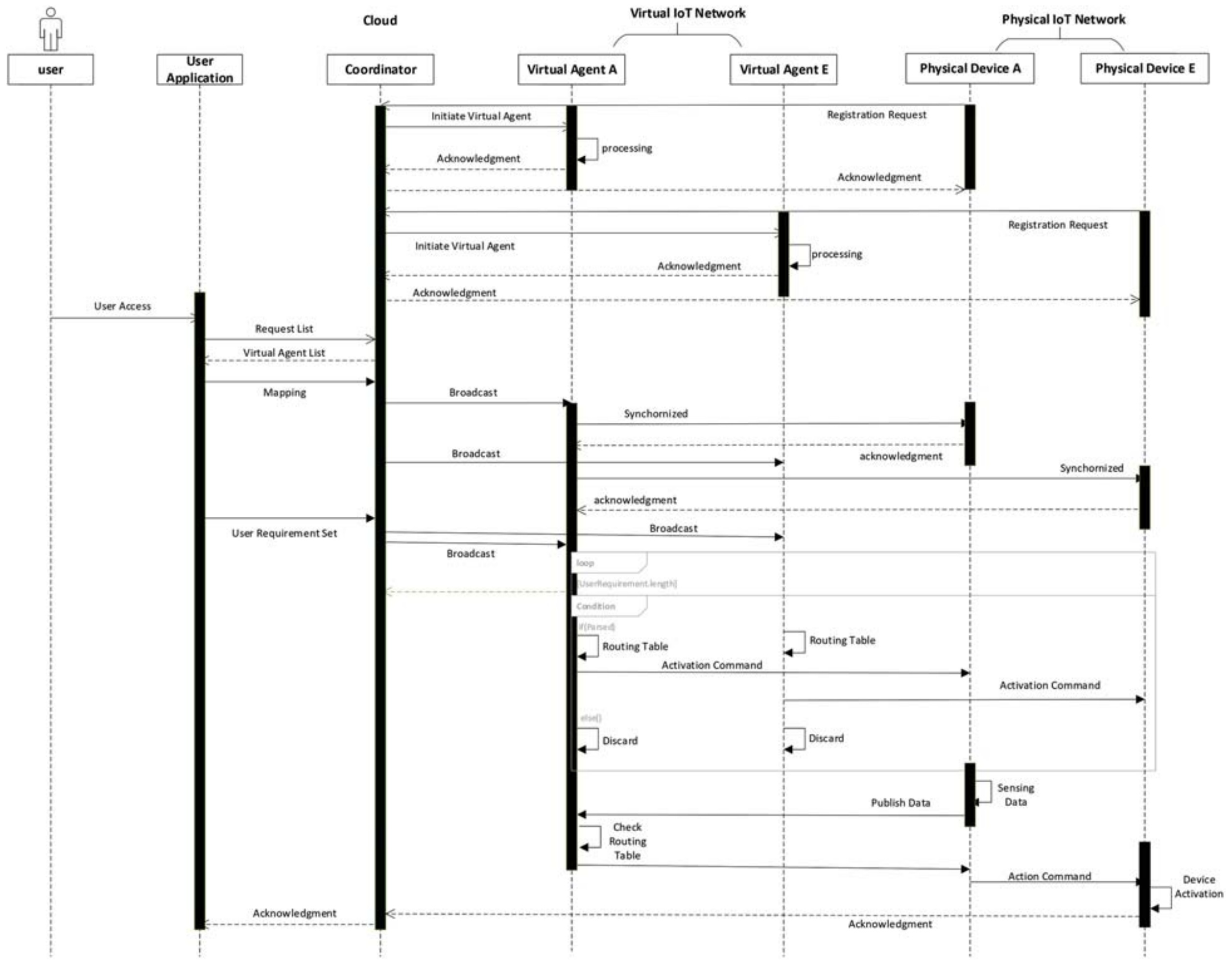


Figure 2: Sequence diagram of virtual IoT network

The user requirement is sent to the coordinator; the coordinator broadcasts the active virtual agents' request. In this case, virtual agents A and E will accept the request, and all other virtual agents will discard the request. The corresponding virtual agent will keep the information in the routing table. Physical device A is a sensor and publishes data to the virtual agent A. The virtual agent will check the routing table and forward the command to the virtual agent E. Virtual agent E is mapped with device E. It will send a command to device E. Device E will perform the specific action and send back the coordinator's acknowledgment.

4 Implementation Environment

This section provides an overview of the tools and technologies used in implementing and evaluating the proposed virtual IoT network. This study has used two different resources, i.e., Raspberry Pi [19] and Arduino [20].

The web server is developed using Node.js, which is the very first Javascript-based server-side framework. Furthermore, we have used the MQTT protocol in our experiment for communicating with low-constrained devices. It is a lightweight protocol that is based on publish/subscribe model. We have used different sensors and actuators to perform the experiments. In this study, sensors are categorized into two types; location-based sensors and environmental-based sensors. Location-based sensors are used to detect any object's motion or track some object's location. The latter collect data from surroundings and exchange data with other devices to perform some specific action.

We have proposed a mapping mechanism between physical devices and virtual agents in the virtual IoT network, as shown in Algorithm 1.

Algorithm 1: Mapping physical IoT devices with virtual agents

```
1: $P = \{PS_1, PS_2, PA_3, \ldots, P_x\}$
2: $V = \{VS_1, VS_2, VA_3, \ldots, V_y\}$
3: **while** $P_x$ in P **do**
4:     Select $P_x$ from P
5:     **for all** $V_y$ in V **do**
6:         **if** user selects $V_y$ **then**
7:             Mapp[ ] $\leftarrow P_x V_y$
8:         **else**
9:             print (select at least one virtual sensor or actuator)
10: Mapping (Mapp[ ])
11: **function** MAPPING (Mapp[ ])
12:     **for all** *i* in Mapp[ ] **do**
13:         **if** successful **then**
14:             db $\leftarrow$ Mapp[i]
15:             result = success
16:         **else**
17:             result = error
18:     return result
19: **end function**
```

It illustrates the mapping methodology between the physical devices and virtual agents. Set P includes registered physical sensors and actuators, whereas set V consists of the physical devices' virtual agents. While Loop will execute as long as the user selects from the set of physical devices P. First, the user will choose sensor or actuator from set P. List of virtual agents will be shown in the Foreach loop's execution. If the user selects the virtual agent, it will be mapped with the physical device; otherwise, the user will be prompted to choose at least one virtual sensor or actuator. As said in earlier sections, the user can map physical device and virtual agent based on one-to-one, one-to-many, and many-to-many relationship and is stored as JavaScript Object Notation (JSON) array on the client-side. The JSON configuration, in turn, is passed to the coordinator in the virtual network for validation and execution. The mapping is stored in the database and used for communication between the physical device and virtual agent. Communication between the physical device and the virtual agent is via MQTT protocol. MQTT protocol is based on publish/subscribe model. A unique I.D. is created when a device is registered in the virtual network. While mapping the physical device and virtual agent, a unique topic is created to exchange data. The unique topic is created by the MQTT broker and is sent back to the physical device. The physical device uses that unique topic to communicate with the virtual agent. The physical device publishes the data, whereas virtual agents are subscribed to that device.

We have proposed a routing mechanism based on user requirements in the virtual IoT network. Algorithm 2 illustrates the routing mechanism based on user requirements in the virtual network. The relationship between physical devices and virtual agents can be one-to-one, one-to-many, and many-to-many. Set VS includes initialized virtual agents of sensors, and set V.A. includes initialized virtual agents of actuators. The user will create a route between physical devices and pass it to the coordinator. The coordinator will broadcast the user requirements to the virtual agents. User requirement includes source, destination, condition, and function. The source is from where the data is coming from, and the destination is where it should reach. The condition and function are to be set by the user e.g., if the temperature is greater than 30, then turn On the fan. The corresponding virtual agents will accept the user requirements while other virtual agents will discard them. Each virtual agent maintains a routing table where routing information is stored.

Algorithm 2: Routing mechanism based on user requirement in virtual networks

```
1: VS = {$VS_1$, $VS_2$, $VS_3$, ... $VS_x$}
2: VA = {$VA_1$, $VA_2$, $VA_3$, ...$VA_y$}
3: function COORDINATOR(userrequirements)
4:     if user.req.source $\varepsilon$ VirtualSensors then
5:         source = user.req.source
6:         condition = user.req.condition
7:     else user.req.source $\varepsilon$ VirtualActuators
8:         destination = user.req.destination
9:         function = user.req.action
10: end function
11: Routing(user-requirements)
12: function ROUTING(user-requirements)
13:     if physical devices || virtual agents are inactive then
14:         send packets until active
15:     else if physical devices || virtual agents are active then
16:         subscribe and publish data
17:     else
18:         find alternative shortest path
19: end function
```

The user requirements are passed to the routing function, where it validates whether the corresponding devices are active or not. If the relation is one-to-one and inactive, it will keep sending packets until the devices become active. If the device is not responsive for 5 min, the coordinator will consider it a faulty device and notify the virtual network administrator. If the relation is one-to-many or many-to-many, and devices are inactive for a long time, it will find an alternative path through other virtual agents for the communication flow. In case devices and virtual agents are active, it will publish data to the required destination

5 Experimental Environment

This section explains the experimental environment of the proposed system. A total of 160 sensors and actuators are utilized to create a physical IoT network. IoT devices are first registered and then initialized by their corresponding virtual agents to form a virtual IoT network. Part of the IoT testbed is portrayed in Fig. 3. The details of the proposed algorithm are given in Section 4. The mapping of physical devices with the virtual agents is performed, and a route is created based on the user requirements for the communication flow among them. The results are compared between the physical and virtual IoT networks. The number of IoT devices are varied from 5 to 160 to observe the network latency, fault-tolerance, and packet delivery ratio. Results indicate that the proposed virtual IoT network is efficient and reliable as compared to the physical network.

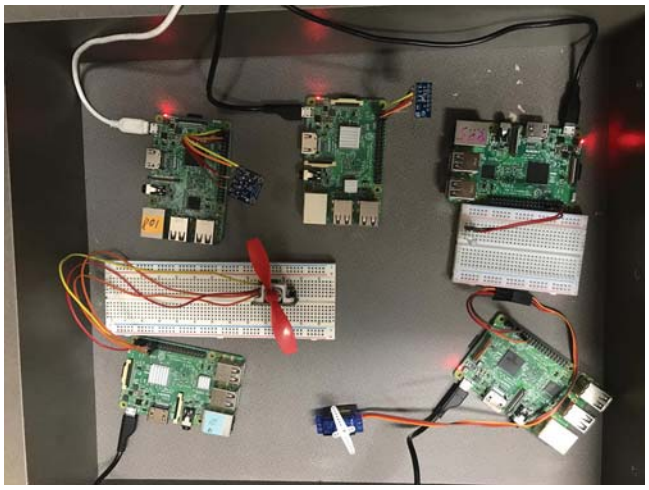
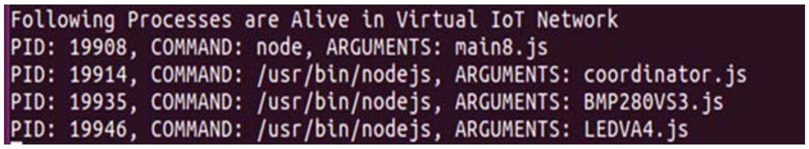


Figure 3: Sensors and actuators used in the experimental environment

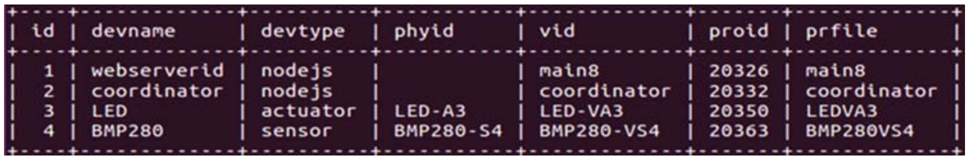
Fig. 4 represents a table that consists of alive processes running in the virtual network. These processes are the virtual agents of their corresponding physical devices. The process with ID 19908 is the parent process, whereas other processes are child processes. The parent process represents the virtual network deployed on the cloud. In child processes, the coordinator is the primary process responsible for registering physical devices, initializing virtual agents, mapping physical devices with virtual agents, and creating a route between physical devices via a virtual network.



```
Following Processes are Alive in Virtual IoT Network
PID: 19908, COMMAND: node, ARGUMENTS: main8.js
PID: 19914, COMMAND: /usr/bin/nodejs, ARGUMENTS: coordinator.js
PID: 19935, COMMAND: /usr/bin/nodejs, ARGUMENTS: BMP280VS3.js
PID: 19946, COMMAND: /usr/bin/nodejs, ARGUMENTS: LEDVA4.js
```

Figure 4: Active processes in a virtual network

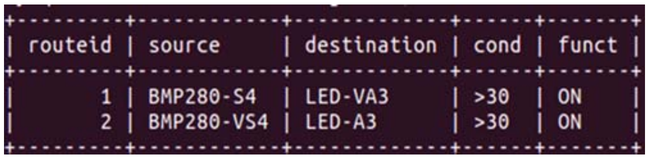
Fig. 5 represents the list of registered physical devices and initialized virtual agents with their process I.D. Each device, when registered, is assigned a physical I.D. After the validation and authorization of physical I.D., the virtual agent is initialized, and a unique process I.D. and process profile are created. Process profile includes inter-process communication (IPC) configuration, port I.D. at which the process is running, the unique topic for publishing and subscribe messages to communicate with physical devices.



```
+----+--------------+----------+------------+-------------+-------+-------------+
| id | devname      | devtype  | phyid      | vid         | proid | prfile      |
+----+--------------+----------+------------+-------------+-------+-------------+
|  1 | webserverid  | nodejs   |            | main8       | 20326 | main8       |
|  2 | coordinator  | nodejs   |            | coordinator | 20332 | coordinator |
|  3 | LED          | actuator | LED-A3     | LED-VA3     | 20350 | LEDVA3      |
|  4 | BMP280       | sensor   | BMP280-S4  | BMP280-VS4  | 20363 | BMP280VS4   |
+----+--------------+----------+------------+-------------+-------+-------------+
```

Figure 5: Registered IoT devices

Fig. 6 represents the routing information between physical devices and virtual agents via a virtual network. The routing table includes information related to the source, destination, condition, and function. The source can be a physical node or virtual agent from where the data is coming from, while the destination can be a virtual agent in the virtual network or the physical device. The condition and function are based on user requirements. The user can set the condition and action to be performed using the web application.



```
+---------+------------+-------------+------+-------+
| routeid | source     | destination | cond | funct |
+---------+------------+-------------+------+-------+
|       1 | BMP280-S4  | LED-VA3     | >30  | ON    |
|       2 | BMP280-VS4 | LED-A3      | >30  | ON    |
+---------+------------+-------------+------+-------+
```

Figure 6: Routing table in virtual IoT network

The proposed virtual IoT network is suitable for heterogeneous networks where there is a massive number of devices. In this study, we have compared network latency between the physical network and virtual network. As the number of nodes increases, the speed of the network communication decreases. We have used a total of 160 sensors and actuators for the experiment. We have mapped physical devices with virtual agents and started the communication flow. For example, if the temperature is greater than 30, turn on the LED, Fan, or motor. We have calculated total time from source to destination for physical and virtual networks based on the communication flow. We have used the Node-IPC module to calculate the time taken to communicate between virtual agents. Results show that the proposed system has low network latency. We have also calculated the packet delivery ratio for the physical and virtual networks. Results show that the proposed approach has a better packet delivery ratio as compared to the physical network. Finally, we can conclude that speed and accuracy have been improved in the proposed method.

6 Execution Results

In this section, we present the results achieved during the experimentation. We have focused on fault tolerance, network latency, and packet delivery ratio metrics in the proposed work. Applications with low network latency are considered efficient. Fig. 7 illustrates the network latency of the proposed virtual network. Latency is the measure of the delay in the network. Network latency is a term used to measure the time taken from source to destination across the network. Minor delays in network connections are referred to as low-latency networks, whereas long delays are called high latency networks. High latency creates a bottleneck in any network communication. In Fig. 7, we can compare the network latency between the physical network and the virtual network. Results indicate that a physical network has higher latency than a virtual network. As we increase the number of nodes, the network latency also increases. The performance of the network decreased when the number of nodes increased. Network delay for 160 nodes in the physical network is 7.5 s, whereas network delay for the same number of nodes in the virtual network is 6.2 s, showing a clear improvement of about 1.3 s. Results indicate that communication via a virtual network is efficient as compared to a physical network.

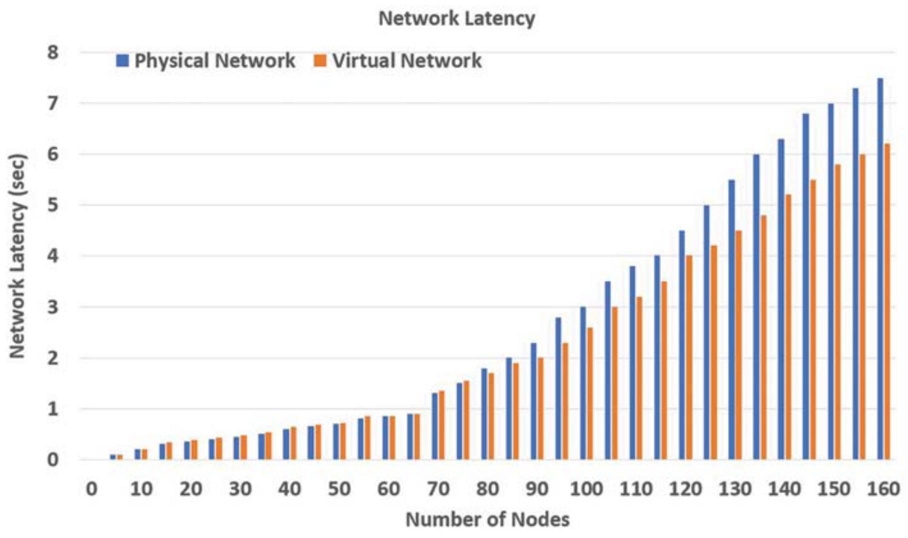


Figure 7: Network latency

Applications with less packet loss during communication are considered reliable. In real-time applications, communication without any packet loss is necessary. Fig. 8 shows the comparison of packet delivery ratio (PDR) between physical and virtual networks. We have used a total of 160 sensors and actuators and performed an experiment checking whether the packets were successfully delivered from source to destination. Packet delivery ratio (PDR) is the ratio of the number of packets sent by the source node and the number of packets received by the destination node. Results indicate that number of packets lost via a virtual network is low compared to a physical network.

Applications with high fault tolerance are considered reliable [9]. IoT devices can be faulty due to hardware, power, or communication failure. It can affect multiple services on a single device simultaneously. In this study, the coordinator keeps the information about virtual agents and their corresponding physical devices. Virtual agents are synchronized with physical devices. In case of any failure, the coordinator notifies the network administrator about the fault. We have experimented by varying devices from 5 to 160. Fig. 9 shows the fault tolerance of the physical network and the proposed virtual network. Fault-tolerance is the capability of a computer system or network to deliver uninterrupted service, despite one or more of its components failing. Fault tolerance also resolves potential service interruptions related to software or logic errors. The purpose is to prevent failure and continue operating correctly. Results indicate that a virtual network has higher fault tolerance as compared to a physical network.

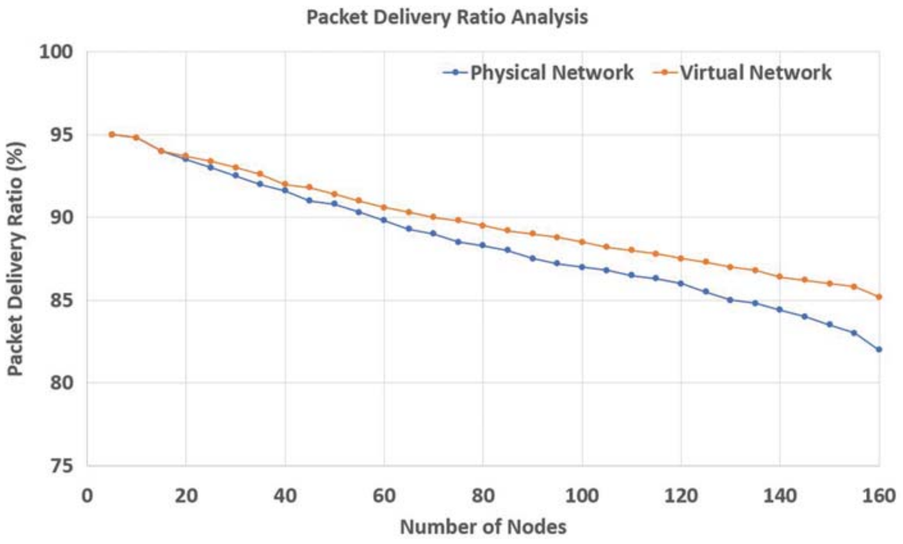


Figure 8: Packet delivery ratio

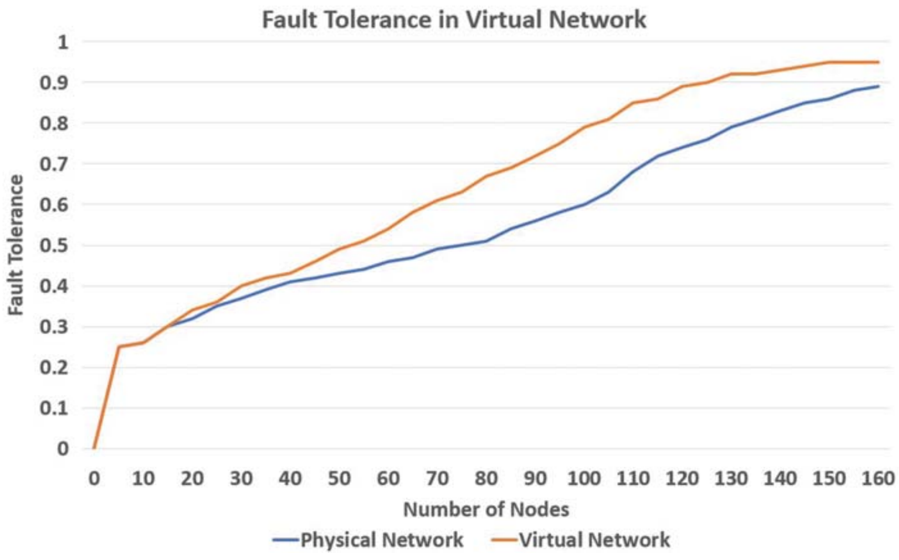


Figure 9: Fault tolerance

7 Conclusion

This paper presents an idea of a dynamic virtual network based on user requirements. In this study, we have designed and developed a virtual network virtual representation of a physical network deployed on the cloud. Each physical device has a virtual agent that a coordinator initializes. The coordinator is responsible for registering physical devices, virtual agents' initialization, mapping between physical devices and virtual agents, and creating a communication route based on user requirements. We have proposed a mapping and routing mechanism in the virtual IoT network allowing the user to map one-to-one, one-to-many, or many-to-many devices. The routing mechanism enables users to reconfigure network configuration based on user requirements. The coordinator is the primary process, whereas virtual agents are the child processes. Virtual agents communicate with each other via IPC mechanism. The communication between physical devices and virtual agents is via MQTT protocol. Results highlighted that the proposed virtual network is flexible and scalable. Thus it is concluded that virtual network has low network latency, less packet loss, high fault tolerance than the physical network in small-scale IoT applications. This paper's possible future work is to simulate the same methodology for a massive number of devices with different constraints and compared them with traditional SDN approaches. The limitation of SDN and this approach for small and large-scale IoT networks are worth investigating and can be an ideal successor for this work.

Funding Statement: This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2019M3F2A1073387), and this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1A09082919), and this research was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-01456, AutoMaTa: Autonomous Management framework based on artificial intelligent Technology for adaptive and disposable IoT). Any correspondence related to this paper should be addressed to Dohyeun Kim.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. Miorandi, S. Sicari, F. De Pellegrini and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] F. Mehmood, I. Ullah, S. Ahmad and D. Kim, "Object detection mechanism based on deep learning algorithm using embedded IoT devices for smart home appliances control in CoT," *Journal of Ambient Intelligence and Humanized Computing*, vol. 3, no. 6, pp. 1–17, 2019.

[3] S. Ahmad, F. Mehmood, A. Mehmood and D. Kim, "Design and implementation of decoupled IoT application store: A novel prototype for virtual objects sharing and discovery," *Electronics*, vol. 8, no. 3, pp. 285, 2019.

[4] F. Mehmood, S. Ahmad and D. Kim, "Design and implementation of an interworking IoT platform and marketplace in cloud of things," *Sustainability*, vol. 11, no. 21, pp. 5952, 2019.

[5] M. Nitti, V. Pilloni, G. Colistra and L. Atzori, "The virtual object as a major element of the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1228–1240, 2015.

[6] S. Ahmad, L. Hang and D. Kim, "Design and implementation of cloud-centric configuration repository for DIY IoT applications," *Sensors*, vol. 18, no. 2, pp. 474–494, 2018.

[7] S. Ahmad, S. Malik, I. Ullah, D. H. Park, K. Kim *et al.,* "Towards the design of a formal verification and evaluation tool of real-time tasks scheduling of IoT applications," *Sustainability*, vol. 11, no. 1, pp. 204–226, 2019.

[8] S. Ahmad, A. Khudoyberdiev and D. Kim, "Towards the task-level optimal orchestration mechanism in multi-device multi-task architecture for mission-critical IoT applications," *IEEE Access*, vol. 7, no. 1, pp. 140922–140935, 2019.

[9] S. Ahmad and D. Kim, "A multi-device multi-tasks management and orchestration architecture for the design of enterprise IoT applications," *Future Generation Computer Systems*, vol. 106, pp. 482–500, 2020.

[10] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow *et al.,* "Wireless sensor network virtualization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 553–576, 2015.

[11] P. Zhang, H. Yao, M. Li and Y. Liu, “Virtual network embedding based on modified genetic algorithm,” *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 481–492, 2019.

[12] J. Ali and B. H. Roh, “Quality of service improvement with optimal software-defined networking controller and control plane clustering,” *Computers, Materials & Continua*, vol. 67, no. 1, pp. 849–875, 2021.

[13] J. Ali, B. H. Roh and S. Lee, “QoS improvement with an optimum controller selection for software-defined networks,” *Plos One*, vol. 14, no. 5, pp. e0217631, 2019.

[14] F. Mehmood, I. Ullah, S. Ahmad and D. Kim, “A novel approach towards the design and implementation of virtual network based on controller in future IoT applications,” *Electronics*, vol. 9, no. 4, pp. 604, 2020.

[15] R. Gouareb, V. Friderikos and A. H. Aghvami, “Virtual network functions routing and placement for edge cloud latency minimization,” *IEEE Journal of Selected Areas in Communications*, vol. 36, no. 10, pp. 2346–2357, 2018.

[16] I. Ullah, S. Ahmad, F. Mehmood and D. Kim, “Cloud based IoT network virtualization for supporting dynamic connectivity among connected devices,” *Electronics*, vol. 8, no. 7, pp. 742, 2019.

[17] J. Ali, G. M. Lee, B. H. Roh, D. K. Ryu and G. Park, “Software-defined networking approaches for link failure recovery: A survey,” *Sustainability*, vol. 12, no. 10, pp. 4255, 2020.

[18] J. Ali and B. H. Roh, “An effective hierarchical control plane for software-defined networks leveraging TOPSIS for end-to-end QoS class-mapping,” *IEEE Access*, vol. 8, pp. 88990–89006, 2020.

[19] V. Vujovic and M. Maksimovic, “Raspberry pi as a sensor web node for home automation,” *Computers & Electrical Engineering*, vol. 44, no. 4, pp. 153–171, 2015.

[20] D. R. Patnaik Patnaikuni, “A comparative study of arduino, raspberry pi and ESP8266 as IoT development board,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, pp. 2350–2352, 2017.