Hindawi
Mobile Information Systems
Volume 2018, Article ID 7157948, 13 pages
https://doi.org/10.1155/2018/7157948

Research Article

Mesh Network Convergence Management System Using Software-Defined Network

Junho Kim,[1] SeungGwan Lee,[2] and Sungwon Lee[3]

[1]*Department of Computer Science and Engineering, Kyung Hee University, Seoul, Gyeonggi-do 17104, Republic of Korea*
[2]*Humanitas College, Kyung Hee University, Seoul, Gyeonggi-do 17104, Republic of Korea*
[3]*Department of Software Convergence, Kyung Hee University, Seoul, Gyeonggi-do 17104, Republic of Korea*

Correspondence should be addressed to SeungGwan Lee; leesg@khu.ac.kr

Received 14 May 2018; Revised 16 August 2018; Accepted 10 September 2018; Published 17 October 2018

Academic Editor: Paolo Bellavista

The Internet penetration rate is rising all over the world. In South Korea, there is no area which does not have Ethernet ports and Wi-Fi. So many people are familiar with network, Internet. However, there are still many countries and regions that do not know the weather today because the Internet is not available yet. To solve this problem, many people are studying the mesh network and actually building it and making efforts to spread the Internet. But, in reality, software that can build and manage such a mesh network is insufficient. In order to solve this problem, this paper proposes Gathering of Organization Treating Humble Ad-hoc Management (GOTHAM) and describes the results. GOTHAM is designed to solve three problems that exist in mesh network users. The first is that the mesh network is difficult to install, and the second is that there is no mesh network topology visualization software for batman-adv. And finally, there is no mesh network integration system for administrators. This paper focuses on these three problems and explains the GOTHAM that combines software-defined network (SDN). In addition, this paper describes three modules of GOTHAM. GOTHAM-setting, GOTHAM-main, and GOTHAM-GUI are explained in detail, and how these three modules work together is described. And also, we evaluate and analyze the performance of file transfer function using flow control which is a user application in the GOTHAM-main module. GOTHAM's goal is not to be used for research but to actually run and let people use right now. That's why GOTHAM is an open source project. All the software used in GOTHAM is open source. And also, we use hardware which is inexpensive and easy to get anywhere.

1. Introduction

Recently, with the development of Internet technology and high penetration rate of the Internet, the Internet has become a commonplace. In South Korea, there are no buildings without wired Ethernet cables, and you can see dozens of Wi-Fi in every building. As such, the penetration rate of the Internet is increasing explosively, and the place where the Internet is not available seems to be awkward. But turning around a little bit, the situation changes completely. According to ITU data in 2016, the global penetration rate of the Internet does not exceed 60 percent [1]. To solve this problem, there are currently more than 10 mesh network open source communities, and over 50,000 mesh network nodes they distribute and manage. It is a trend to expand network using mesh network because it is expensive and time-consuming to install network in every land of the Earth by wire. As some of the 50,000 nodes are being used for research for the development of mesh networks, the use of special purpose mesh networks will increase as well as the general availability of the Internet [2, 3].

Although the use of the mesh network is increasing, the software that can effectively manage the network and the manpower who can control the network are insufficient. Organizations that are now researching and deploying many mesh networks are moving to mesh networks faster and more reliably. It is also important and necessary to develop mesh network technology itself. However, unlike these organizations, we wanted to develop software for administrators and common users who can easily and quickly install

and manage the network. We also wanted to create a platform and system dedicated to the mesh network that can provide the necessary functions as seen from the user's point of view and the manager's point of view.

This paper describes GOTHAM (Gathering of Organization Treating Humble Ad-hoc Management), open source software. GOTHAM was designed and built to solve three main problems. First, it is difficult to install a mesh network. For ordinary users who do not know the network, it is difficult to construct a mesh network, and there is no software to build a mesh network automatically. The people who want to construct a mesh network still have to access all the nodes and modify the configure file. I wanted to solve this problem the most. Second, there is no mesh network topology visualization software. Currently there is no web-GUI for batman-adv, but users and administrators need a visualization tool to see the topology. Visualization is a very important factor when a network administrator installs a network. It is also necessary information for network learners. So, we tried to solve this problem. Finally, there was no integrated system suitable for the mesh network. Centralized controllers are not suitable for the mesh network that supports fully distributed networks. GOTHAM is composed of three modules. There are GOTHAM-setting, GOTHAM-main, and GOTHAM-GUI. GOTHAM-setting module constructs mesh network automatically and sets basic network configuration. GOTHAM-main module manages the network (flow control, network information management, etc.). GOTHAM-GUI module visualizes mesh network topology. GOTHAM is a program for the users who use mesh networks and administrators who manage mesh networks. Later, we will discuss the three modules of GOTHAM, along with the open source technologies used for GOTHAM.

2. Related Work

This section describes the techniques used for GOTHAM. Since GOTHAM is an open source, all the software used in GOTHAM is an open source.

2.1. BATMAN, Batman-adv, and batctl. Better Approach To Mobile Ad-hoc Networking (BATMAN) is a routing protocol for the ad-hoc (mesh) network [4]. All nodes do not have all routing information like optimized link state routing protocol (OLSR), and only the best link information between nodes is memorized [5, 6]. The routing protocol such as OLSR is not suitable for unstructured, dynamically changing topology and unreliable wireless ad-hoc networks [7–10]. Therefore, the BATMAN routing protocol does not remember the entire node status, and each node stores only the information that is the best next hop between the nodes. Because the entire topology changes periodically, each node manages the information itself, rather than the information of the entire node. Based on this information, the packet is sent only to the best one according to the originator in routing. To obtain the link information of these nodes, BATMAN uses a special message called the originator message (OGM). The OGM has an originator address, a sender node address, and a sequence number. When a node sends an OGM to all the nodes around it that are connected to it, those nodes add their own address to the OGM and broadcast again. In this way, all nodes periodically generate OGMs and transmit and receive messages to each other, and calculate the transmission quality (TQ) indicating the link state. Echo quality (EQ) and receive quality (RQ) are required to obtain the TQ. All this information is obtained through the OGM, and finally, the TQ is measured together with the formula TQ = EQ/RQ and the hop count [11]. Figure 1 shows how to calculate TQ with hop count.

Batman-adv is a practical implementation using the BATMAN routing protocol [12]. It has been developing since 2007 and is being developed until now. Batman-adv works on layer 2 and is being developed as a Linux kernel module to reduce overhead. Therefore, it can be used without a separate installation where the current Linux kernel is installed. Batman-adv consists of C language and provides many functions such as gateway (Internet connection, bandwidth control, etc.), bridge loop avoidance, network coding, and AP isolation.

batctl is also written in C and is intended for configuration and debugging of batman-adv. As a configuration tool, you can add and remove interfaces to the mesh network. And you also can change and verify the batman-adv configuration (OGM interval setting, etc.). Furthermore, you can modify and check the batman-adv functions and do things like gateway announcement.

As a debugging tool, you can ping by the MAC address, and check traceroute nodes, log confirmation, originator list, and client list. Installing batctl is essential for easy handling of batman-adv.

2.2. Hazelcast. Hazelcast is open source, providing in-memory data grid (IMDG). It uses the Apache 2.0 license and allows multiple programs to share multiple data structures in a distributed environment [8]. The biggest feature of Hazelcast is that it does not need a master member. Each cluster member functions identically. Another feature is that all data exists only in memory. This allows Hazelcast to run quickly. Also, if there is a problem with a particular cluster, the data are safe unless there is a problem with all cluster members. All data are copied and distributed as shown in Figure 2. This makes it possible to maintain data more securely.

Hazelcast has three advantages [13]. First, there are many versions of clients that support various languages such as Java, C++, and Python. Second, it supports various data structures such as map, queue, multimap, and list and also supports lock and semaphore. So, the users can use data in Hazelcast widely. Finally, it is light. Hazelcast is provided as a Jar file, not a specific program.

2.3. OVS and OpenFlow. Open virtual switch (OVS) is a Linux-based virtual software switch that abstracts the networks and dynamically controls network resources [14]. OVS can act as a bridge between VMs and external networks

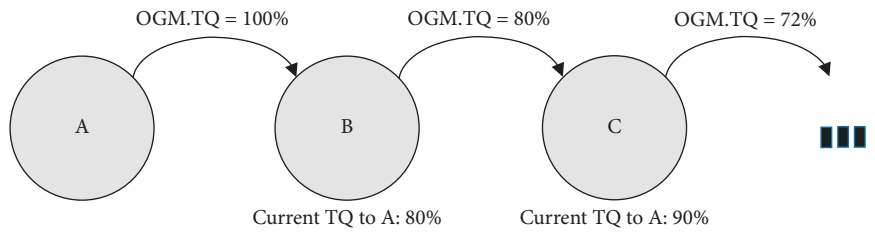

Figure 1: BATMAN transmit quality propagation.

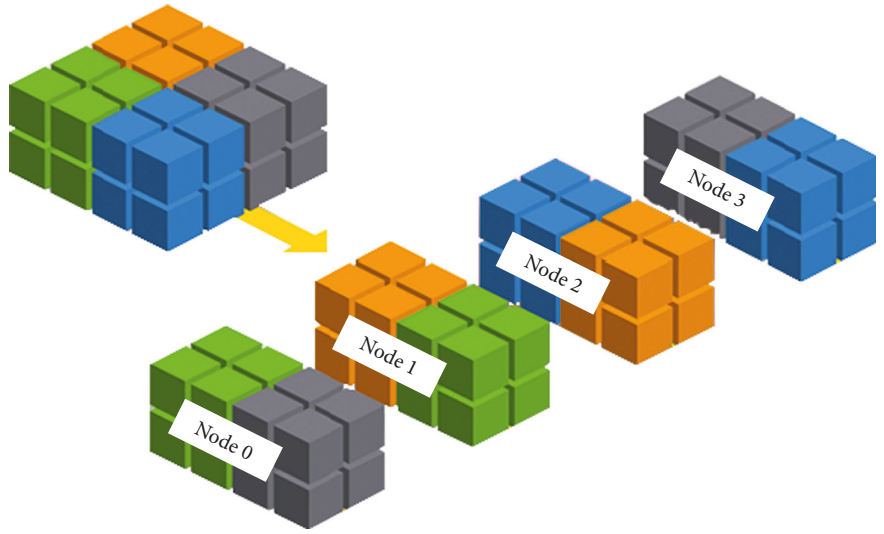

Figure 2: Hazelcast data distribution.

within the hypervisor. In addition to these basic switches and bridges, it provides services such as VLAN isolation and traffic filtering for security and provides monitoring functions such as Netflow, sFlow, SPAN, and RSPAN. OVS handles flows in two ways. In the case of a packet coming from the external network, there are two methods that process packets. The slow path method is made by new flow rules for unidentified packets. And the fast path method is processed by a conventional flow tool for identified packets. There are so many good functions in OVS. However, the main reason for using OVS for GOTHAM is that it supports this basic switch virtually, but it supports OpenFlow.

OpenFlow is the standard interface for implementing SDN and is led by the ONF (Open Networking Foundation) group [15]. You can determine the flow of packets according to the internal flow table, which allows you to forward packets to a specific port on the switch or discard packets. Also, it is possible to modify the source MAC address, IP address, and the port number of the transport layer [16].

The first application of SDN was centralized network management, but research on wireless mesh networks using SDN was also steadily conducted. As a research field called software-defined wireless mesh networks (SD-WMN) in general, many researches have been conducted to improve existing wireless mesh networks performing distributed control/management through SDN [17–19].

In the SD-WMN architecture, the network is monitored and managed in a centralized manner or a hybrid-approach using both centralized and distributed manner. Although the use of SDN simplifies the WMN management, it is still not clear whether the SDN architecture, which was originally designed for wired links, can effectively handle the dynamic topology of WMNs. Bao et al. addresses the traffic balancing issue introduced by node mobility [20]. To reduce the response time of the SDN controller for dynamic network topology, a two-layer supervised learning model is proposed that helps the controller to predict the node mobility and link failure probability. Then, an alternative route selection scheme is proposed, which achieves optimal traffic balancing while minimizing the control plane overhead.

Huang et al. proposed architecture and traffic orchestration scheme for efficiently controlling and managing limited radio resources of data plane and control plane through SD-WMN [21].

In [22], by applying SDN to the wireless mesh network, the authors gain an unprecedented opportunity to design more efficient and cost-effective solutions. In WMNs, the process of service provisioning incorporates tightly correlated steps, including access point association, gateway selection, and backhaul routing. In most of the prior studies, these steps were investigated as independent NP-hard problems. Sajjadi et al. presented a structured scheme to address the demands of end-users over SDN-aware WMNs [22]. In contrast to most of the former works, formulation proposed by the authors takes the key characteristics of wireless networks into account, especially for multichannel

multiradio WMNs. In addition, for applying solution to the large-scale scenarios, the authors introduce a heuristic algorithm that can find a suboptimal solution in polynomial time.

In [23], SDN is applied to the wireless mesh network to formulate an optimization problem to determine the flow update order with the objective of minimizing the overall congestion during reconfiguration, referred to as the switching cost.

Actually, there are many issues about wireless mesh networks such as a cluster header election method. And, we hope the SDN-based approach can be applied into these issues in further research [24, 25] (Figure 3).

3. GOTHAM

Gathering of Organization Treating Humble Ad-hoc Management (GOTHAM) is designed to solve three problems. (1) It is difficult to install the mesh network. (2) There is no software that visualizes the mesh network topology. (3) There is no system or controller for the mesh network. We tried to solve it. In addition, the most important thing was to make GOTHAM available to many users. We tried to make it available in the real world, not for research. As a result, GOTHAM has been open sourced. And it will be an open source. This section describes the purpose of GOTHAM and its detailed functions.

3.1. Purpose of GOTHAM. GOTHAM was created to solve the above three problems. That's why GOTHAM allows the mesh network users to create or join mesh networks without any configuration. Also, you can see the current mesh network topology on the web browser. Finally, it was created as a fully distributed system without a central server, and the corresponding functions are implemented. Figure 4 is whole of the GOTHAM architecture. It can organize mesh network automatically with Bluetooth Beacon. After constructing the network, GOTHAM can manage the mesh network using IMDG. So, the network manager can access all nodes through whatever node you want. And GOTHAM can use OVS. Because of this, it can control packet flow. For the users, it provides file transfer function. And, for the administrators, it provides control packet flow and congestion in the mesh network.

GOTHAM is targeted at the indoor mesh networks. We want GOTHAM to be used by students who are in the school or the people who have to make network quickly and easily. When the first node is turned on, it senses the surrounding mesh network through the Bluetooth Beacon and subscribes if it exists, and if it does not exist, it forms an arbitrary mesh network itself. GOTHAM-setting is a module that automatically constructs the mesh network. Thereafter, there is a GOTHAM-main which manages the network on the mesh network, and this module controls the flow or the information of the network. Finally, the module that visualizes the network topology is GOTHAM-GUI. GOTHAM-GUI allows the users to view information visualized on the Web. Details are given in the next part.

Another feature of the mesh network using GOTHAM is the introduction of the concept of SDN. GOTHAM has two planes, that is, the control plane and the data plane. They are separated from each other. All nodes physically form a network through two different mesh networks. Control message such as the GOTHAM message, ICMP (Internet Control Message Protocol), and ARP (Address Resolution Protocol) uses the control plane, and packets using other TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) pass through the data plane. This was possible because OVS and OpenFlow were used to specify the appropriate planes for each port. If GOTHAM-setting is activated, it will be set automatically. Of course, ordinary users accessing the AP will also use the normal data plane. By introducing the SDN concept, the packet for the network control can be guaranteed speed regardless of the state of the data plane. In addition, the control flow can be managed stably and, if necessary, the other plane can be used for backup purposes if the plane has a problem. This is covered in more detail in the file transfer function of the GOTHAM-main module.

Figure 5 shows the user's perspective when GOTHAM and mesh network work together. Users can use multiple applications regardless of the network. To do this, we use Bluetooth Beacon and batman-adv, and GOTHAM is used to create and manage it. TOX is a P2P (peer-to-peer) messenger as one of your applications. TOX is well suited for mesh networks because it runs on P2P without a central server. Video call is also possible with normal text transmission and file transfer.

3.2. Function of GOTHAM. As mentioned above, GOTHAM consists of three modules. These three modules are explained in detail in this part. In Figure 6, the lowest "node setting" is the GOTHAM-setting module [26]. The middle part is GOTHAM-main, and the "Web-GUI" part is GOTHAM-GUI. Each node is physically separated. However, they are connected to each other by a mesh network and share information with IMDG on the network. Thus, although physically separated, data can be obtained in its own node. So, Figure 6 illustrates this. First, we will start with GOTHAM-setting which starts for the first time.

3.2.1. GOTHAM-Setting Module. The GOTHAM-setting module activates the Bluetooth Beacon when the mesh node is powered on and senses the Bluetooth Beacon around it. It applies the universally user identifier (UUID) format defined in GOTHAM to determine whether the mesh network exists or not. If the mesh network exists, it transmits service set identifier (SSID) and channel information to batman-adv based on the sensed Bluetooth Beacon information and subscribes to the mesh network. However, if the mesh network does not exist, it sends arbitrary SSID and channel information to batman-adv and forms a new mesh network. The reason for using Bluetooth Beacon is that it is much faster than the surrounding contents search time compared to the wireless LAN [27]. Additionally, people can detect BLE (Bluetooth low energy) with the mobile phone or

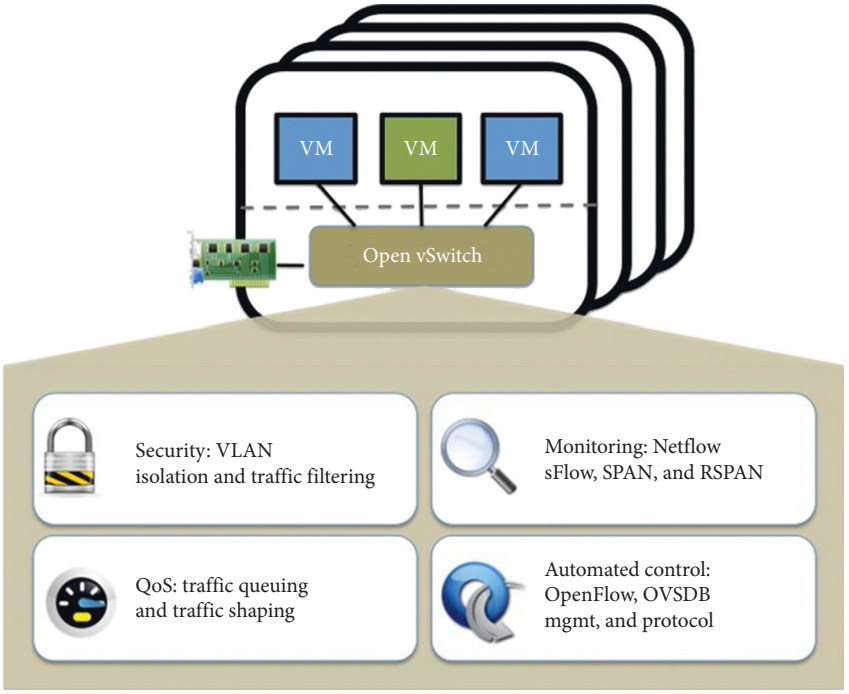

Figure 3: Open virtual switch.

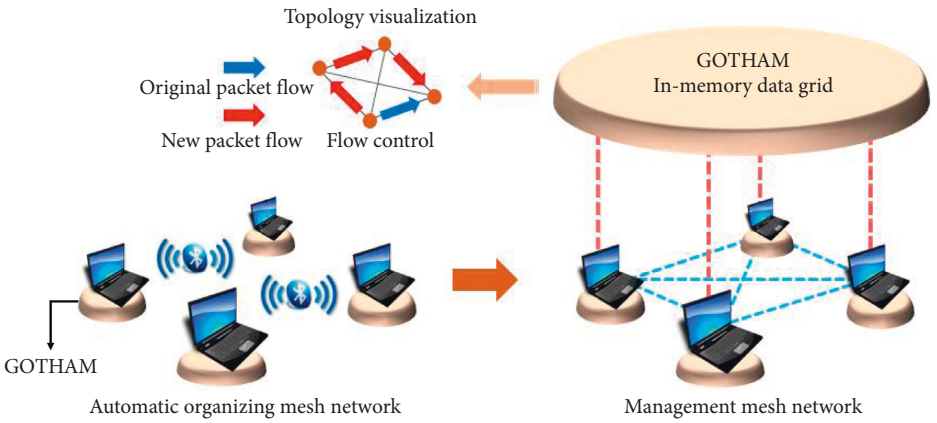

Figure 4: GOTHAM architecture.

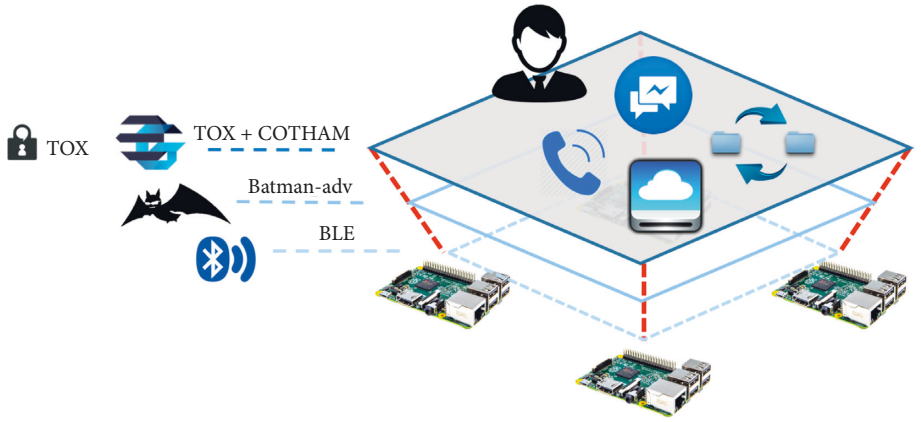

Figure 5: User's view on the mesh network.

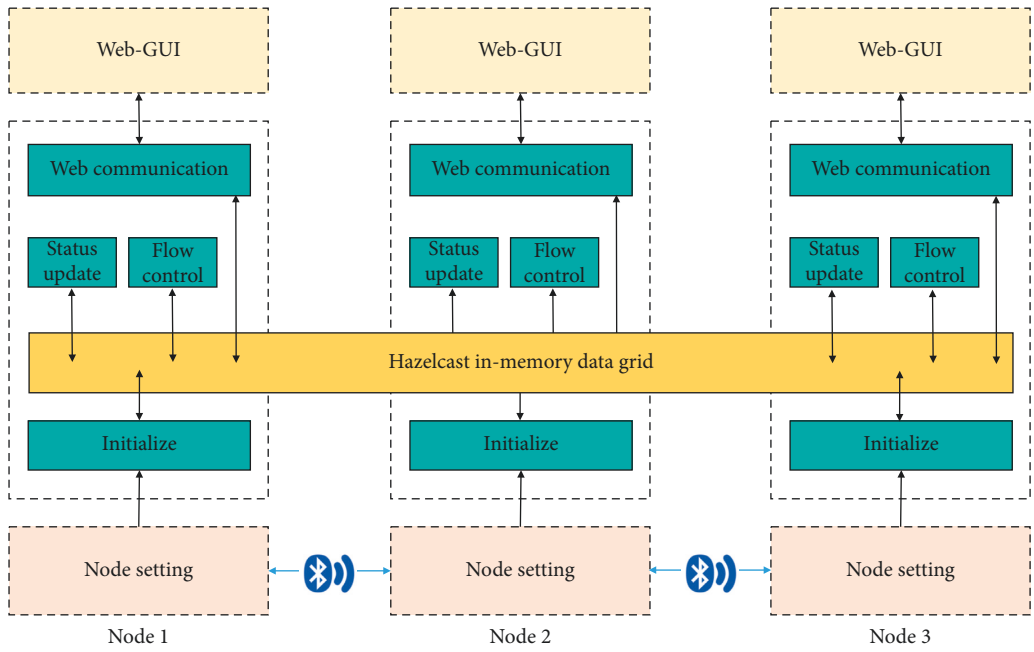

Figure 6: GOTHAM module architecture.

laptop easily without the mesh node. Bluetooth Beacon is the optimal technology because we planned GOTHAM in-house.

Figure 7 shows the Bluetooth Beacon UUID format used by the GOTHAM-setting module [26]. Among the packets of Bluetooth Beacon, the usable part is UUID (128bit, 16byte). Let's look at the contents of Figure 7 one by one. VERSION represents the version of the Bluetooth Beacon that is currently formed. VERSION information changes when the public network information is changed, the master IP is changed, or the SSID or CHANNEL is changed. Master IP was used to connect the server-client GOTHAM before the first Hazelcast was introduced, but now it has a Dynamic Host Configuration Protocol (DHCP) server, and it will display the IP (Internet Protocol) of the first node to be turned on. And SSID and CHANNEL are essential items for forming a mesh network, and they are the most important items of UUID of Bluetooth Beacon. MY IP represents its own IP, and 2.4 GHz and 5 GHz represent the number of 2.4 GHz and 5 GHz wireless interface cards available to the nodes. Currently, it is not only used for display, but it will be able to create a new mesh network later or share data more quickly through Multi-RAT.

The GOTHAM-setting joins the mesh network using Bluetooth Beacon as described above. In addition, basic network setting such as OVS, hostapd, and DHCP daemon (DHCPD). Also, if there is an available extra interface, it runs the hostapd program to operate the access point (AP) mode for ordinary users who do not use the mesh node and cannot use the mesh network. This allows common users to access the mesh network. And it gains the IP address from the master node that presents Master IP. Batman-adv provides DHCP request packets deliver. So, all of nodes in the mesh network can get the IP address. The IP address is important and necessary for other software, such as GOTHAM-main or P2P application. When you join the mesh network and the basic network setting is finished, the GOTHAM-main module is executed, and the module is terminated.

The GOTHAM-setting module should be fast and be able to handle many programs properly, such as hostapd and OVS. So it was built with Bash and Python.

3.2.2. GOTHAM-Main Module. The GOTHAM-main module is the central part in Figure 6 managing the entire network and providing many functions for the users and administrators. It was made into JAVA to avoid subordination of O/S. When the GOTHAM-main module starts, it first forms IMDG with Hazelcast. After all the nodes have connected to each other, each node updates its node information in real time. If a GUI request comes in, it collects information of all nodes and delivers them. Since it does not always have the entire node information, GOTHAM collects information only when necessary. It can reduce total traffic so it is so effective. The feature of GOTHAM-main is that it can distribute the information of the entire node even if there is no specific server in the mesh network. It also maps IP and MAC (media access control) addresses. Therefore, the user can manage the network based on the MAC address or the name specified by the user without knowing the IP of each node. This concept is based on the existing content centric network (CCN) and can manage name-based networks. CCN is that it does not utilize IP which represents location information, and the terminal can acquire the content through the name [28]. GOTHAM is not CCN, but it uses CCN's concept that it uses the content's name instead of IP. Actually, GOTHAM uses the IP address but it conceals the IP address. Currently, it has a mapping of

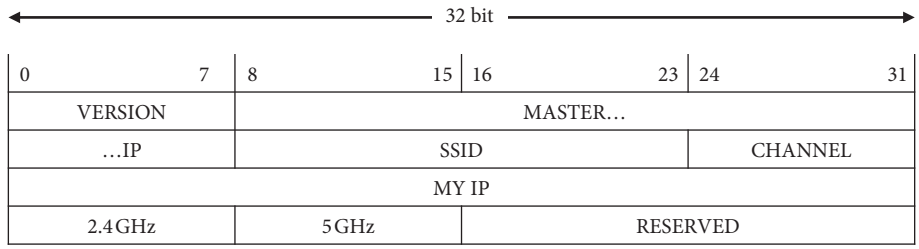

Figure 7: GOTHAM Bluetooth Beacon UUID packet format.

the IP address of the corresponding node and the name of the node through Hazelcast. Thus, the user can view the information and access the node through the information. Another characteristic is that if the data rate is low, the control plane is opened to a certain extent to guarantee a data rate higher than a certain rate. When a user transfers a file, it measures the speed in real time and performs the flow control accordingly. So, if the speed is low, the file can be transferred to two paths.

Figure 8 shows a sequence diagram of a file transfer function that automatically activates when a user's data transfer rate falls below a certain level using flow control. When the file transfer starts, a TCP session for the control plane and a TCP session for the data plane are established. Then, when the connection is completed, the user sends the name of the file on the desired node, and if the file exists, the file is transmitted in units of 100 MB. After the first 100 MB arrives, it measures the speed and decides whether to create a thread according to its speed. And it sends 100 MB again to the two paths, or just the data plane, just like the first sending. GOTHAM-main repeats the process until you get all the files. This technique is link aggregation. GOTHAM combined two physical paths to one logical path. It measures the speed in real time, performs link aggregation, and sends the file, so users can adjust the path to suit the situation at that time. Users can be applied to situations that are much more mobile than static link aggregation and is effective in a real network environment.

In this paper, GOTHAM divided into three types according to speed. The top speeds of the network were divided into "Good" types up to 70 percent, "SoSo" types up to 50 percent, and "Poor" types below. GOTHAM divides 100 MB: "SoSo" sends 70 MB to the data plane and 30 MB to the control plane. In case of "Poor," the ratio of the data plane to the control plane is 5 : 5. Performance evaluations based on this feature are discussed in detail later. With this function, the user can guarantee Quality of Service (QoS) more than a certain rate when exchanging necessary data. GOTHAM's file transfer function is very useful because of the unstable mesh network.

Taken together, GOTHAM-main is the heart of GOTHAM and provides an environment for managing the entire network. In addition, it has information and functions to manage the network. Most of the GOTHAM-main code is about managing the whole network information such as link state and node state. On the user side, it has a file transfer function, making it useful for both administrators and users.

3.2.3. GOTHAM-GUI Module. The GOTHAM-GUI is a module for visualizing the mesh network topology on the web, using TOMCAT as a server and using de JavaScript and WebSocket [29, 30]. When a user makes a GUI request, GOTHAM-GUI requests GOTHAM-main to send the whole node information. And it draws it to the GUI with information. GOTHAM-GUI runs on all nodes. If you had a server-client structure, all the nodes had access to the one node for visualization. This increases the traffic on the network, and if there is a problem with the central web server, the entire network user is not provided with the visualization function. However, since the GOTHAM-GUI is spread over all nodes, the problem of congestion traffic to one side disappears. In addition, even if the server of a specific node is dead, the access to the other node can be visualized, and the stability is also excellent.

The GOTHAM-GUI was originally created for the Open Network Operating System (ONOS) application, so we used d3 javascript, WebSocket, and JAVA, which are used by ONOS. Currently, it will work as an independent program, but in the future, we will develop an ONOS application that can be used by the SDN controller to control future mesh networks.

Figure 9 shows the whole sequence diagram of GOTHAM. The first time the user powers the mesh node, the GOTHAM-setting module runs automatically. The GOTHAM-setting senses Bluetooth Beacon and sets its Bluetooth Beacon. And it sets batman-adv and basic network. After that, it runs the GOTHAM-main module. Then, GOTHAM-main is executed and makes IMDG. After that, it updates information of network in real time. Then, if the user requests the GUI, the GOTHAM-GUI runs and visualizes the network topology in the web browser to the user.

4. Performance Evaluation and Analysis

This section describes the mesh network testbed using GOTHAM. We also measure the performance of the file transfer function using flow control of GOTHAM-main.

4.1. GOTHAM Testbed. Raspberry Pi was mainly used as a mesh node. We created a testbed using 6 Raspberry Pi and two laptops, each with two wireless LAN cards, and one Bluetooth Beacon dongle. The specifications of the testbed are shown in Table 1.

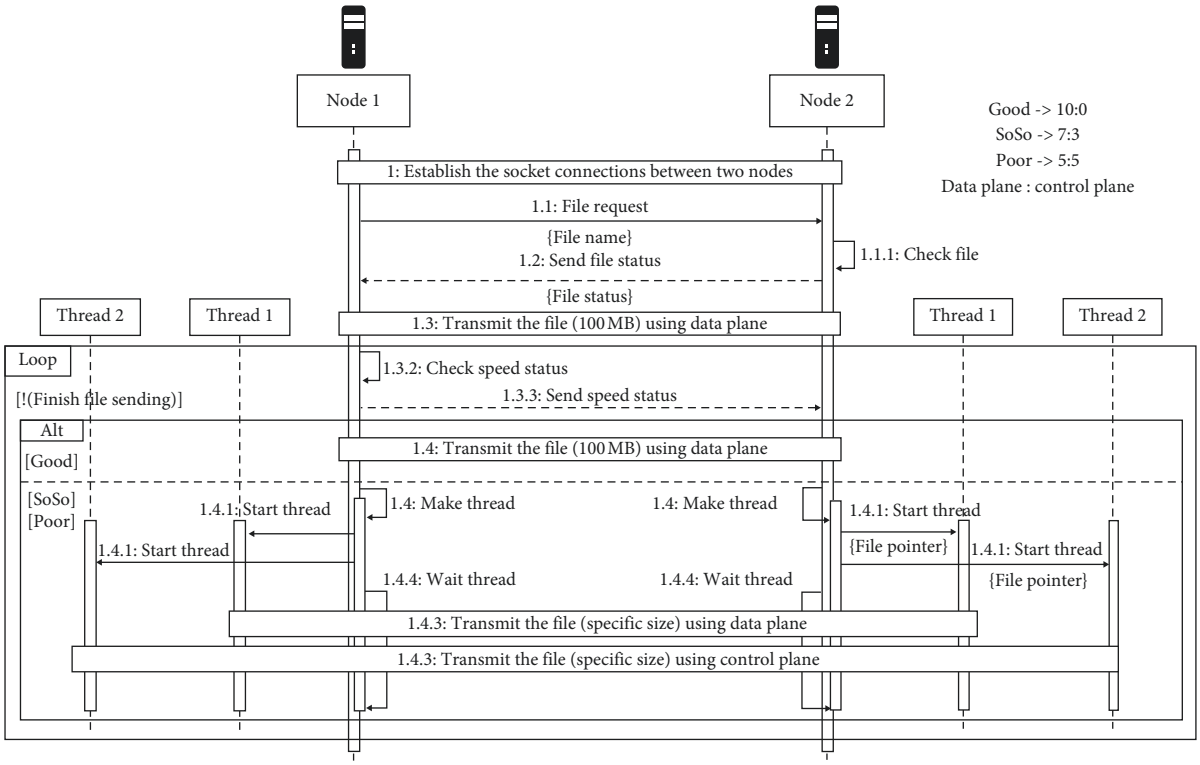

Figure 8: GOTHAM-main file transfer sequence diagram.

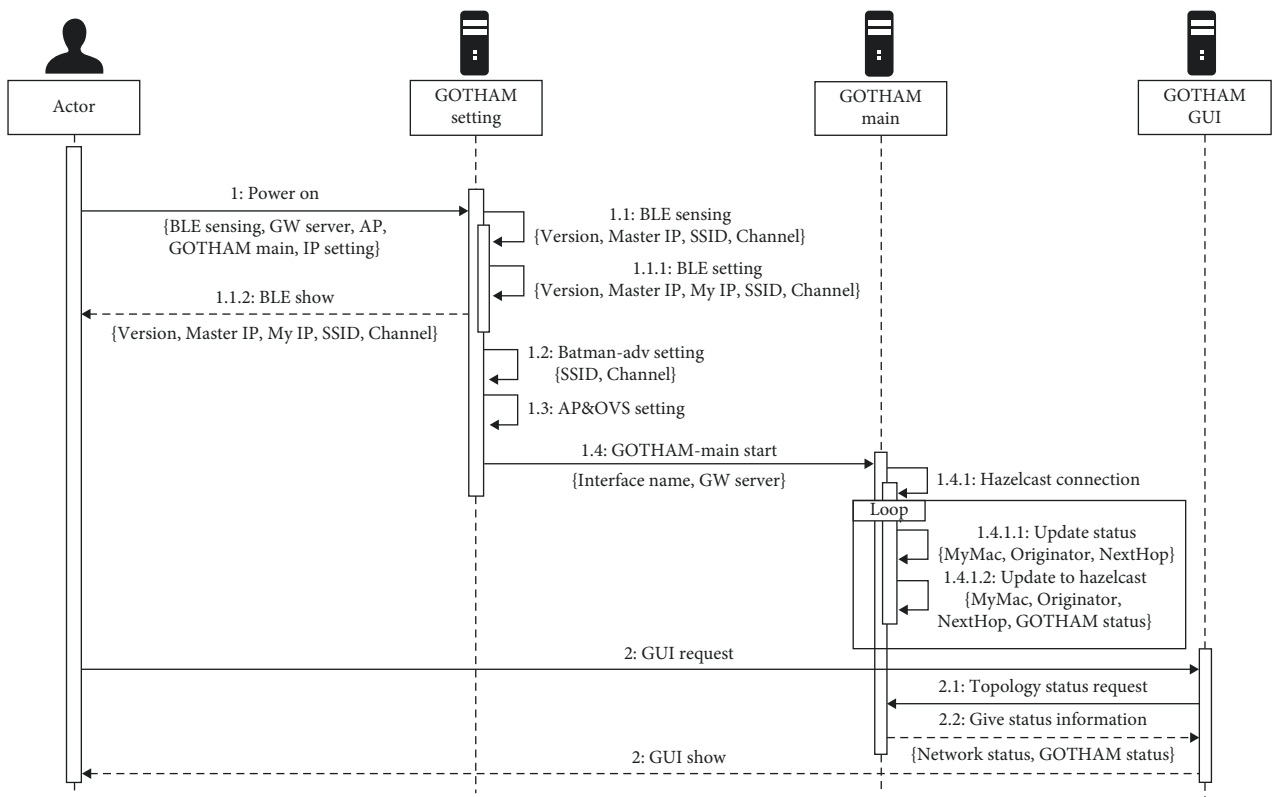

Figure 9: GOTHAM sequence diagram.

Table 1: GOTHAM testbed specification.

Component	Description
Raspberry Pi 2	(i) QUAD Core Broadcom BCM2836 CPU (ii) 1 GB RAM/16 GB micro-SD card (iii) Ubuntu mate 16.04 LTS/GNU/Linux 4.1.19-v7+ (iv) Hostapd v2.4 (v) OVS v2.5.0 (vi) Batman-adv v2015.0 (vii) batctl v2016.0-2
Laptop #1	(i) Intel i5 CPU (ii) 4 GB RAM/500 GB HDD (iii) Ubuntu mate 16.04 LTS/GNU/Linux 4.1.19-v7+ (iv) OVS v2.5.0 (v) Batman-adv v2015.0 (vi) batctl v2016.0-2
Laptop #2	(i) Intel Pentium CPU (ii) 2 GB RAM/160 GB HDD (iii) Ubuntu mate 16.04 LTS/GNU/Linux 4.1.19-v7+ (iv) OVS v2.5.0 (v) Batman-adv v2015.0 (vi) batctl v2016.0-2
WLAN interface	(i) ipTIME N100UM/Ralink RT3070 (ii) 2.4 GHz
Bluetooth interface	(i) NEXT 204BT (ii) Bluetooth CSR 4.0

As shown as Table 1, we used Ubuntu mate as the default O/S. Ubuntu mate is a light weight version of Ubuntu. So, it can be used on small embedded devices like Raspberry Pi. Absolutely, we can install the desktop computer or laptop. It is used because of its wide range of application and its light weight.

As show in Figure 10, we have actually created a testbed [31, 32]. All these 8 nodes automatically construct a mesh network with each other when the devices are turned on. And we can see and control the whole mesh network with GOTHAM. In Figure 10, there are no supplementary battery. But if we do not have power points, we used the supplementary battery for the power source. It is so efficient to make the mesh network at outdoor. Figure 11 shows the screen shot as GOTHAM-GUI.

In Figure 11, all node's best next hops are a originator itself. The GOTHAM-GUI displays a logical link due to the characteristics of the mesh network. It draws a picture with the best next hop information which is generated by batman-adv. Therefore, there may be some differences from the physical signal.

Figure 12 is a drawing of various topology states drawn with GOTHAM. We tested many cases with Raspberry Pi. Nevertheless, we were able to confirm that all the topologies were drawn well.

4.2. Testbed for Performance Evaluation. In this section, we will explain the testbed that is built to perform performance

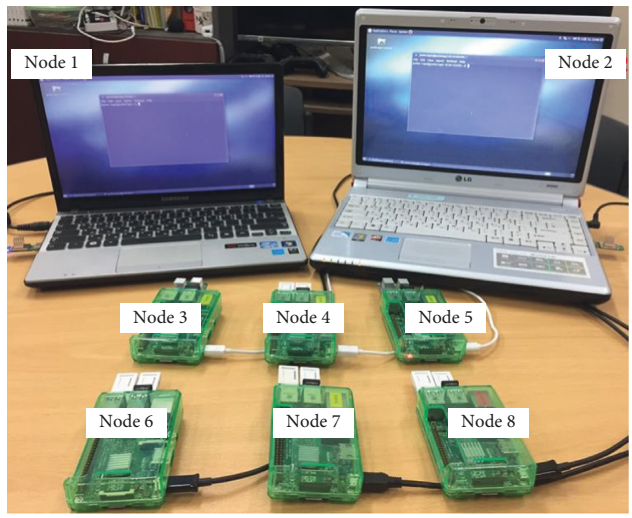

Figure 10: GOTHAM testbed.

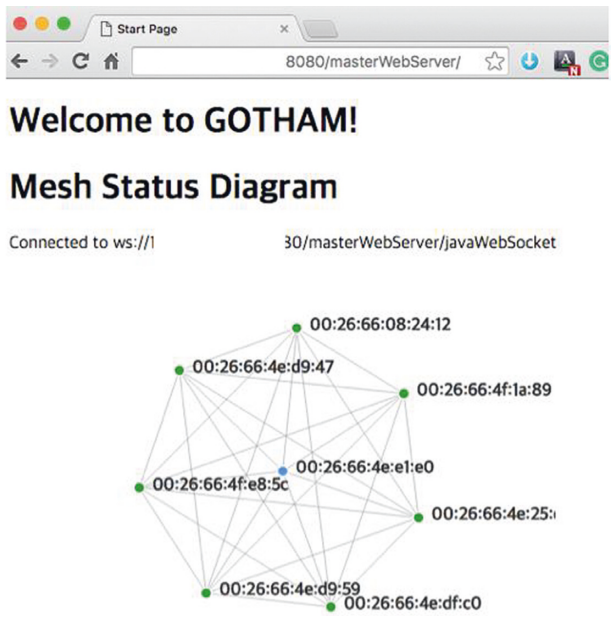

Figure 11: GOTHAM-GUI.

measurement of the file transfer function using flow control in GOTHAM-main (Table 2).

The GOTHAM testbed created above is made of Raspberry Pi, but Raspberry Pi's sending packets over two or more interfaces could not deliver as much bandwidth as hardware problems. Therefore, the GOTHAM file transfer function created in this paper could not be measured properly. To solve this problem, we used two desktops to build a wired mesh network and experimented on it. Batman-adv supports wired mesh network. So this experiment can check only performance of the GOTHAM file transfer function without wireless interference.

4.3. Result on Performance Evaluation. We sent a total of 400 MB files for testing. After 100 MB transmission, the

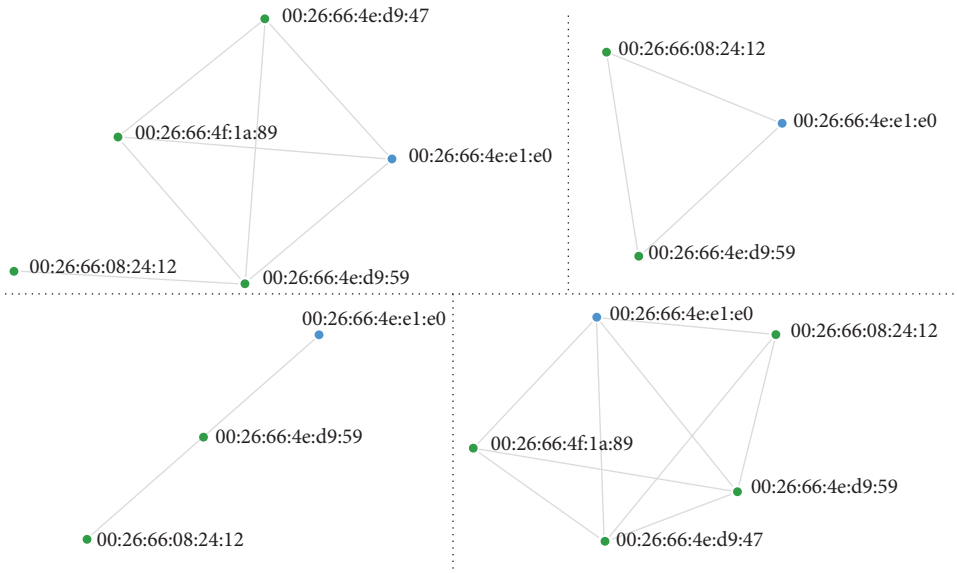

Figure 12: GOTHAM-GUI topology example.

Table 2: GOTHAM testbed specification for file transfer.

Component	Description
Desktop	(i) Intel Xeon CPU (ii) 16 GB RAM/1 TB HDD (iii) Ubuntu 16.04.1 LTS (iv) GNU/Linux 4.4.0-45-generic (v) OVS v2.5.0 (vi) Batman-adv v2015.2 (vii) batctl v2016.0-2

Table 3: File transfer rate.

Speed status	Data plane	Control plane
Good (OnePath)	100	0
SoSo (TwoPath)	70	30
Poor (StrongTwoPath)	50	50

speed was measured and the transmission speed was divided into 3 parts as "Good," "SoSo," and "Poor." If it is "Good" as shown in Table 3, it is sent to one path. In the case of "SoSo" and "Poor," it is sent to two paths, which is divided according to the ratio of sending the file.

Figure 13 shows the amount of data over time when sent to OnePath, TwoPath, and StrongTwoPath under normal circumstances. When measured with iperf, the maximum transmission rate was 11.9 MB/s and the maximum transmission rate of all paths was 23.8 MB/s with 11.9∗2. When analyzing the graph in Figure 13, OnePath sent 400 MB, totaling 37.5 sec, which translates to 10.66 MB/s when converted to speed. TwoPath was measured at 27.5 sec, and the speed was measured at 14.55 MB/s. Finally, in StrongTwoPath, the speed was 20 MB/s with 20 sec. As you can see, StrongTwoPath was the fastest as you would expect. And it was measured to be similar to the maximum transmission rate for OnePath sent to one path. Also, StrongTwoPath's speed is similar to the maximum transmission rate for total path.

Figure 13 can demonstrate StrongTwoPath is faster than TwoPath. And TwoPath is also faster than OnePath. In this test, there are no congestion in the network. In the network, there are only packets which are generated by control software such as GOTHAM, batman-adv, and system packet. It means that all packets used only the control plane. We will analyze and explain in more detail the later part.

Figure 14 shows the graph of average bandwidth and speed every 2 seconds based on the graph in Figure 13. It shows the maximum speed when two lines are one path and the maximum speed when two paths are used. The two lines are red lines in Figure 14. When using OnePath, it is confirmed that it does not exceed 11.9 MB/s. However, in the case of TwoPath and StrongTwoPath, we can see that the control plane is used with the data plane.

The graph, as shown in Figure 14, shows that GOTHAM is using a minimum of the control plane. The data plane is congested and uses the control plane temporarily, but the load of the control plane should not be constantly increased or maintained. Therefore, we wanted to use the GOTHAM control plane but only to the minimum. You can see that the above graph works well to the desired goal according to the original purpose.

Figures 13 and 14 are shown for the graph in Figure 15. Figure 15 shows that when 0~200 MB is transmitted, iperf packets are generated in the data plane as much as the maximum bandwidth, thereby creating congestion situation. After receiving 200 MB, we made a general situation and measured again. Generally, when traffic is maximized, the graph is a blue graph when you send a file, that is, only one path. You can see that the graph's slope, or speed, changes around 200 MB. The speed of 0~200 MB transmission is about 6.25 MB/s. However, the speed of 200~400 MB transmission is about 9.76 MB/s. The "original" graph is a graph of a file sent to one path in a typical situation where no traffic has occurred. It is for comparison. It is similar with

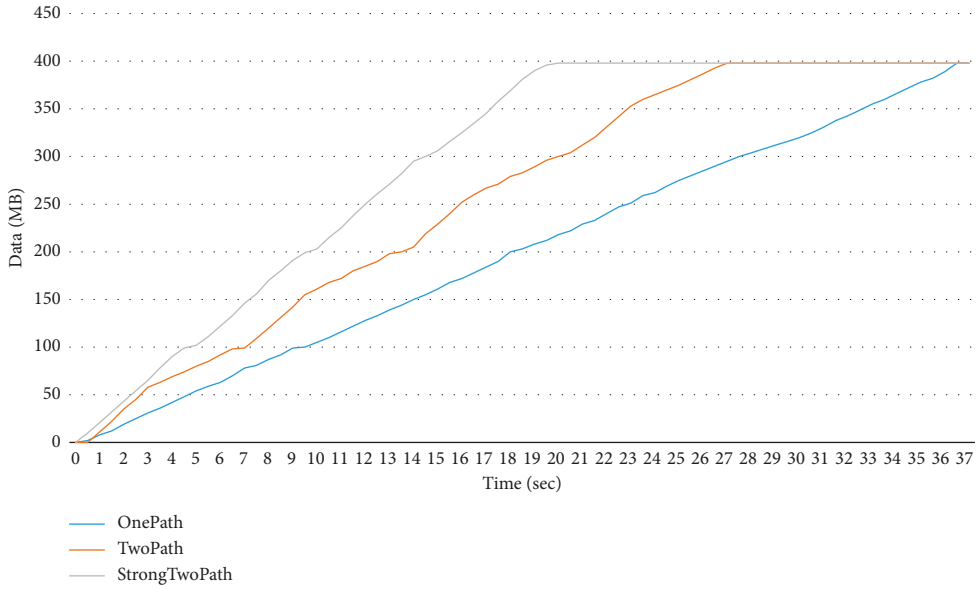

Figure 13: The number of data according to time in normal environment.

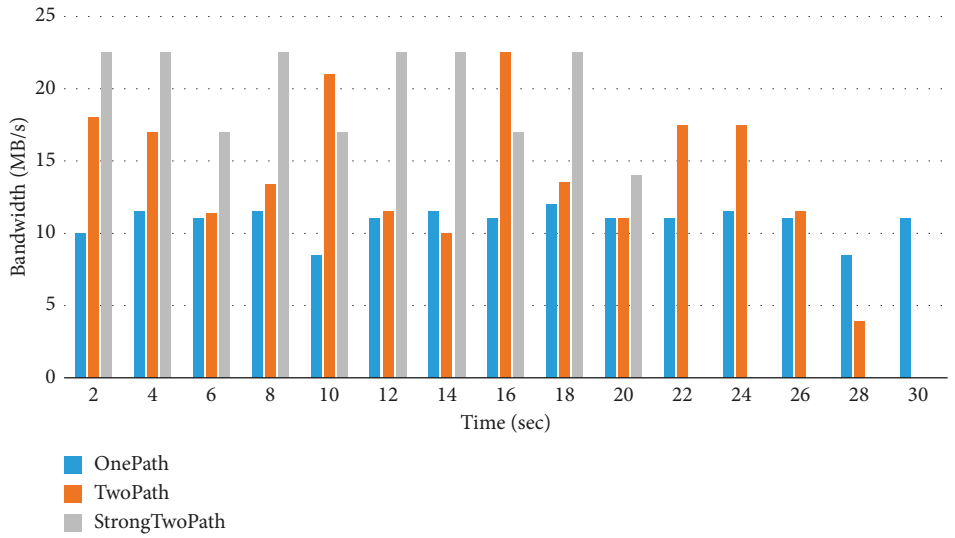

Figure 14: Bandwidth according to time in normal environment.

Figure 11's OnePath. The "TrafficGenerated with GOTHAM" graph shows the performance measured by GOTHAM's own functions. It decided to determine the path itself according to the algorithm of GOTHAM. GOTHAM was automatically sent to StrongTwoPath from 100 MB to 200 MB because the speed of 0 MB~100 MB did not come out normally. Since then, it sent files to TwoPath because it was a bit faster than 0~100 MB section. And after 200 MB, there was no congestion situation, so it sent it back to OnePath. This graph shows that when GOTHAM is executed, it is about 11.5 sec faster than when it is not. GOTHAM could send all the data in 1.5 sec delayed time compared to the normal situation. However, this numerical value is similar to the network in a normal state. The most of the users cannot feel this difference.

Through the three graphs, we could demonstrate that the regular users who wish to transfer files in the mesh network maintain a constant speed by opening the control plane to certain part even when network congestion occurs. Due to the characteristics of the mesh network, there may be a lot of traffic that is concentrated in a certain path. If you use GOTHAM at this time, the users will feel less inconvenience.

5. Conclusions

Mesh networks are increasingly used, and the use of mesh networks with special purposes such as network construction at the sea, network construction at the concert site, and a mesh network for general Internet use is increasing. GOTHAM has been designed and developed by considering

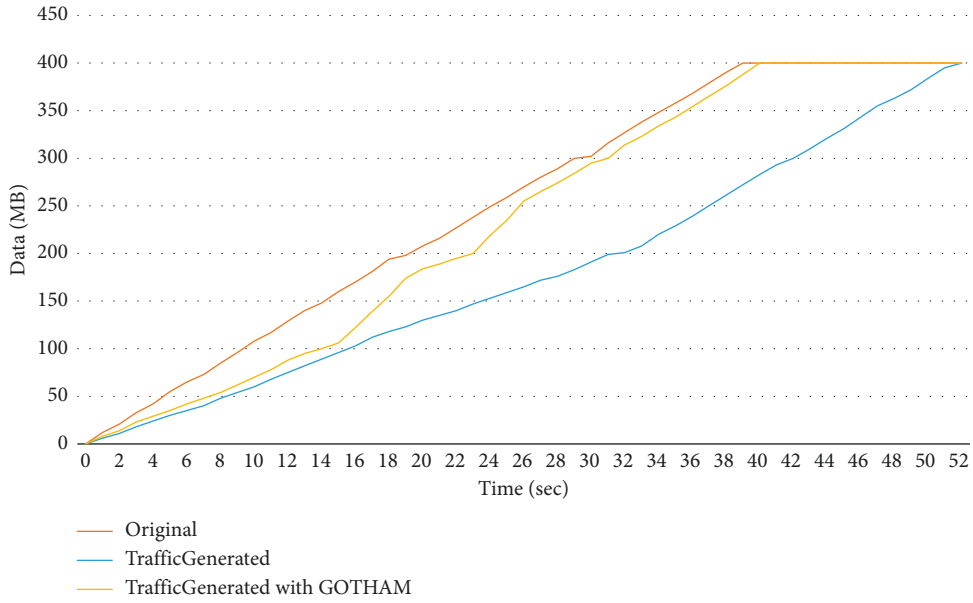

Figure 15: The number of data according to time in congestion environment.

that there is no software for the users and administrators to use the mesh network. GOTHAM started with a goal of the mesh network integrated management system, and some frame was created. The control and data planes are separated from the control and data messages through SDN-based network configuration. In addition, by adding the ability to automatically build a mesh network for the general user who does not have knowledge of the network, we made the mesh network more convenient and faster to use. We also created a network topology visualization for network administrators for users' eye candy. This makes it very simple to see the current network configuration in a web browser. As an administrator, they can access the nodes with a specific name without IP. Currently, we are mapping the node name and the IP of the node so that we can access it. But we plan to make it possible to control the node through future work. In addition, the user functions are the file transfer function using the flow control which is included in GOTHAM-main and evaluated performance. It is optimized for the mesh network when users transmit files and can provide stable speed. All of this has been tested in a Raspberry Pi and laptop environment for low-cost boards to make it easy for the ordinary user to use and optimized for both environments. This is not just for use in the laboratory, but for anyone who really needs a network.

The biggest feature of GOTHAM is that there is no centralized DB, server, and main controller. However, all nodes can be visualized without a centralized server. Also, users can use the GOTHAM file transfer function and can access every node. Another feature is that GOTHAM is an open source to everyone. Anyone who needs to build a network can install GOTHAM and use it at any time [33]. In order to have these features, we have studied and tested a lot of open source and decided on the necessary open source among them. Like the open sources which used in GOTHAM, we hope GOTHAM will be used in others open source.

Currently, GOTHAM is the beginning stage. Now we have created three modules and implemented basic functions. The goal is to enable everything in the web browser afterwards. As shown in Figure 5, the goal is for the user to be able to control everything via the web browser without knowing the GOTAHM along with the network. And, this article is based on the first author's Master's thesis [34].

Data Availability

There are no special data in this paper. The data related to the paper are described in the paper.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea Government (MSIT) (2014-3-00547, Development of Core Technology for Autonomous Network Control and Management).

References

[1] International Telecommunication Union, *Percentage of Individuals Using the Internet 2000–2011*, ITU, Geneva, Switzerland, 2012.

[2] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: commodity multihop ad hoc networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 123–131, 2005.

[3] M. Raza, N. Aslam, H. Le-Minh, S. Hussain, Y. Cao, and N. M. Khan, "A iritical analysis of research potential, challenges and future directives in industrial wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 39–95, 2018.

[4] A. Neumann, C. Aichele, M. Lindner, and M. Wunderlich, *Better Approach to Mobile Ad-Hoc Networking (BATMAN)*, Work in Progress Technical Report, 2008.

[5] D. Johnson, N. Ntlatlapa, and C. Aichele, *Simple Pragmatic Approach to Mesh Routing Using BATMAN*, in *Proceedings of 2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, CSIR, Pretoria, South Africa, October 2008.

[6] R. Sanchez-Iborra, M.-D. Cano, and J. Garcia-Haro, "Performance evaluation of BATMAN routing protocol for VoIP services: a QoE perspective," *IEEE Transactions on Wireless Communications*, vol. 13, no. 9, pp. 4947–4958, 2014.

[7] N. M. Anas, F. K. Hashim, H. Mohamad, M. H. Baharudin, and M. P. Sulong, "Performance analysis of outdoor wireless mesh network using BATMAN advanced," in *Proceedings of 16th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 1–4, Takamatsu, Japan, June 2015.

[8] L. Sicard, M. Markovics, and G. Manthios, *An Ad-Hoc Network of Android Phones Using BATMAN: The Pervasive Computing Course*, The IT University of Copenhagen, Copenhagen, Denmark, 2010.

[9] L. Barolli, M. Ikeda, G. De Marco, A. Durresi, and F. Xhafa, "Performance analysis of OLSR and BATMAN protocols considering link quality parameter," in *Proceedings of International Conference on Advanced Information Networking and Applications (AINA'09)*, pp. 307–314, Bradford, UK, May 2009.

[10] Y. Zheng, Y. Wang, Z. Li, L. Dong, Y. Jiang, and H. Zhang, "A mobility and load aware OLSR routing protocol for UAV mobile ad-hoc networks," in *Proceedings of International Conference on Information and Communications Technologies (ICT 2014)*, pp. 1–7, Dublin, Ireland, January 2014.

[11] A. Quartulli and R. Lo Cigno, *Client Announcement and Fast Roaming in a Layer-2 Mesh Network*, University of Trento, Trento, Italy, 2011.

[12] BATMAN-adv open source official site, https://www.open-mesh.org.

[13] Hazelcast open source official site, https://hazelcast.org.

[14] OVS open source official site, https://openvswitch.org.

[15] Open Networking Foundation official site, https://www.opennetworking.org.

[16] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: a survey and taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713–2737, 2016.

[17] W. Quan, Y. Liu, H. Zhang, and S. Yu, "Enhancing crowd collaborations for software defined vehicular networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 80–86, 2017.

[18] H. Zhang, W. Quan, H.-C. Chao, and C. Qiao, "Smart identifier network: a collaborative architecture for the future internet," *IEEE Network*, vol. 30, no. 3, pp. 46–51, 2016.

[19] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Network*, vol. 32, no. 5, pp. 112–117, 2018.

[20] K. Bao, J. D. Matyjas, F. Hu, and S. Kumar, "Intelligent software-defined mesh networks with link-failure adaptive traffic balancing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 266–276, 2018.

[21] H. Huang, P. Li, S. Guo, and W. Zhuang, "Software-defined wireless mesh networks: architecture and traffic orchestration," *IEEE Network*, vol. 29, no. 4, pp. 24–30, 2015.

[22] D. Sajjadi, R. Ruby, M. Tanha, and J. Pan, "Fine-grained traffic engineering on SDN-aware Wi-Fi mesh networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7593–7607, 2018.

[23] F. Yaghoubi, M. Furdek, A. Rostami, P. Öhlén, and L. Wosinska, "Consistency-aware weather fisruption-tolerant routing in SDN-based wireless mesh networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 582–595, 2018.

[24] A. Adekiigbe, A. A. Ahmed, A. S. Sadiq, K. Z. Ghafoor, and K. Abu-Bakar, "An efficient cluster head election algorithm for client mesh network using fuzzy logic control," *Journal of Internet Technology*, vol. 18, pp. 1057–1067, 2017.

[25] P. Sarigiannidis, T. Lagkas, S. Bibi, A. Ampatzoglou, and P. Bellavista, "Hybrid 5G optical-wireless SDN-based networks, challenges and open issues," *IET Networks*, vol. 6, no. 6, pp. 141–148, 2017.

[26] J. Kim, M. Seo, and S. Lee, "GOTHAM-mesh network management and visualization for batman-adv," in *Proceedings of International Conference on Information Networking (ICOIN)*, pp. 435–437, Da Nang, Vietnam, January 2017.

[27] C. Kim, S. Lee, and S. Lee, "Multi-device to multi-device (MD2MD) content-centric networking based on multi-RAT device," *Symmetry*, vol. 9, no. 12, p. 291, 2017.

[28] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of 5th International Conference on Emerging Networking Experiments and Technologies*, pp. 1–12, Rome, Italy, June 2009.

[29] D3 javascript open source official site, https://d3js.org.

[30] I. Fette, *The Websocket Protocol*, IETF, Fremont, CA, USA, 2011.

[31] J. Kim and S. Lee, "Constructing infrastructure wireless network using open source," in *Proceedings of Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 757–762, Vienna, Austria, July 2016.

[32] J. Kim, G. An, S. Ko, M. Seo, G. Kim, and S. Lee, "Openwincon-open source wireless-wired network controller for free small to medium networking," in *Proceedings of 26th Joint Conference on Communications And Information (JCCI)*, pp. 27–29, Sokcho, Korea, April 2016.

[33] OpenWinCon open source official site, https://github.com/OpenWinCon.

[34] J. Kim, "A study of mesh network convergence management system using SDN," Master's thesis, Kyung Hee University, Seoul, 2017.