Development of a Simulated Portable Mesh Network via ESP8266-Based Devices with Utility Application

Dennis A. Martillano
College of Computer and Information Science, Malayan Colleges Laguna, Pulo Diezmo, Cabuyao City Laguna 4025, Philippines
Email: damartillano@mcl.edu.ph

John Patrick M. Asiddao, Roy Carlos B. Dupaya, Ken Brian B. Lavina, and Patricia Alexandra A. Sy
Malayan Colleges Laguna, Pulo Diezmo, Cabuyao City Laguna 4025, Philippines
Email: jpmasiddao@live.mcl.edu.ph, rcbdupaya@live.mcl.edu.ph, kbblavia@live.mcl.edu.ph, paasy@live.mcl.edu.ph

***Abstract*—**Mesh network is very effective for node-to-node communications. In networks that require and demand devices for home monitoring and control, uncommon topology like Mesh is viewed to have a more effective advantage. However, when it comes to networks, Mesh Network is almost left untouched in most studies simply due to the existing network solutions such as WLAN or Bluetooth. The existence of low-powered WPAN devices fill the demand in integrating devices into mesh networks. However, constant configuration could be very complex and difficult for a mesh using existing WPAN devices since nodes constantly move in and out dynamically in the mesh network. This study focused on the development of a simulated mesh network through ESP8266 enabled devices to deliver a straightforward setup and portable connectivity. The study also developed a utility application that integrates an algorithm for node network operations, and a facility that keeps track of mesh network information including delays, node connections, and data transmission. Three(3) portable ESP8266 devices were used and configured in the study. The devices were integrated in a simulated mesh network and subjected to network processes including identity tagging, dynamic connection, routing, One-way Delay and Payload Size Tests. Results of One-Way Delay and Payload Size Tests indicate consistency of transmission and receiving of data of nodes connecting and disconnecting to the simulated network. This can be used as basis in extending further into more high-end nodes like handheld devices and even computers. Similarly, the utilization of dedicated algorithm for Mesh in this study proved that portability can be achieved to avoid loss of connection when nodes abruptly fail in the network.

***Index Terms*—**Mesh network, internet of things, WPAN Devices, ESP8266, payload size

I. Introduction

Wireless mesh network is being applied to various types of wireless technology and networks to allow devices to send data to each node without the use of network cables. Devices such as IEEE Zigbee, Digi Xbee, WiFi, and Bluetooth are some of these devices. However, some of these devices are very tedious to be configured since they require a third-party software [1]. These processes are usually being done repeatedly for every use, and these devices can only perform up to their specification, which may not be advantageous to nodes that move in and out of the network dynamically [2]. Due to these disadvantages, there is a need to propose a Wireless Mesh Network (WMN) that has the nature of configuring the nodes distributed which makes it advantageous when it comes to range [3]. WPAN devices like Zigbee were designed to handle issues, which also answers the problem of providing low-power consumption for devices but still have the capability to deliver what the device needs. However, devices that act as nodes in a wireless mesh network could be either stationary or mobile and could come from different sets of wireless technologies within the same architecture of a WMN. In mesh, nodes can move, new nodes can be connected and added, and existing nodes might leave or be disconnected to the network as well. As a consequence, factors like dynamic changes in routing paths and delays in message sending can happen, which can be very difficult to configure for existing WPAN devices [2].

To support wireless mesh, integrating the network into a distributed setup through mobile devices via a utility application can be novel innovation utilizing mesh networks [3]. This will allow simulation of network topology and permits tracing and diagram functions to read and process information about the network. There are existing network utility software that provide these features however, they do not specifically cater to a mesh network. Despite the scalability and configuration advantages of mesh networks, massive research on this is not converted to actual deployment, which limits the available resources and utilities that can be used by network specialists.

The objective of the study is to develop a simulated mesh network using ESP8266 Modules whilst providing portable connectivity among WiFi devices, which can be used as a new approach of wireless networking for future implementation of the network. Specifically, the study dealt with the following: (1) Development of portable devices using ESP8266, integrated into a microcontroller that feeds ESP8266 configuration data and autoconfigure in a mesh cluster, (2) Designing of painless-mesh based

Manuscript received September 15, 2021; revised April 12, 2022.
doi:10.12720/jcm.17.5.350-357

algorithm module that will find alternative connections when a node fails or disconnects, which would allow network operations to continue, and (3) Develop a utility mobile application for the simulated mesh network that would monitor and keep track of vital mesh information

II. Literature Review

In a wireless mesh network, as the number of routing nodes increases, the complexity of configuring the wireless nodes also increases [4]. Access to the transmission medium is usually shared, thereby creating competition among routing nodes.

While mesh network may be uncommonly integrated, it is considered reliable and provides an opportunity to prevent failure of one node to affect the network. This is due to the fact that the rest of the nodes can still communicate with each other, either directly or using one or more intermediate nodes. This implies that a wireless mesh network can self-form and self-heal [5].

Wireless Mesh Network has been used in various applications. For example, mesh network was used in a Stroke Monitoring Appliance that utilize wireless nodes that form a wireless mesh network for wearable devices. The wearables feature a sound transducer that couples to the wireless mesh network to communicate patient data over the wireless mesh network to detect a heart attack or a stroke attack [6]. Mesh network can easily be integrated with high-speed Internet connections as well as other communication frameworks including Internet of Things [3]. It is for this reason that Wireless Mesh Network is also being used in Agriculture, where a wireless sensor network sends real-time data of the climatological and other environmental properties, which can be relayed to nodes or to a central repository. These show that mesh networks can effectively be used to wirelessly connect nodes of different forms, and eventually communicate to an entire city using inexpensive and existing portable technology.

Mentioned studies of mesh network show the use of WPAN devices based on the IEEE-802.15.4 standard. This is true in order to introduce routing algorithm suited for the sensing application within the range. However, with the exception of too much configuration in WPAN devices, the elevating concept of IoT pose the need for underlying technologies much suited to realize Internet of Things [7]. This entails the need to provide devices like ESP8266-based modules, with capabilities for 2.4 GHz Wi-Fi (802.11 b/g/n, supporting WPA/WPA2). Wi-Fi Mesh Network Library (painlessMesh) specifically written for the ESP8266 platforms were studied and implemented [8]. The painlessMesh library is unique for it enables the resource-limited ESP8266 device to form a mesh network. In this setup, two or more nodes, with the same SSID (Service Set Identifier) can automatically be connected to form a network. The intention is to allow auto-configuration and easy set up a network that requires neither a routing plan nor a central controller, where all nodes are equal. This type of mesh network where each node can serve as both an access point for other nodes to connect to, and as a station connecting to another node can be extended, programmed, and managed for portability.

When it comes to design issues with respect to the protocols, mesh network can be described in a layer-wise manner. In a simulated mesh network, routing protocol must consider design of different routing metric, routing overhead and robustness, and effective use of support infrastructure including utility application that can solidify need for node balancing, route adaptability, and monitoring of vital mesh network information [9]. Wireless mesh networks have not yet witnessed mass-market deployment, thereby producing very limited simulations that can be used for large-scale or actual networks [10]. Among vital mesh network information that must be monitored for simulations are One-way delay (round trip time) and Data rate, which can be further be evaluated and observed by series of node induced data transmission via Pay Load size tests.

III. Results and Discussions

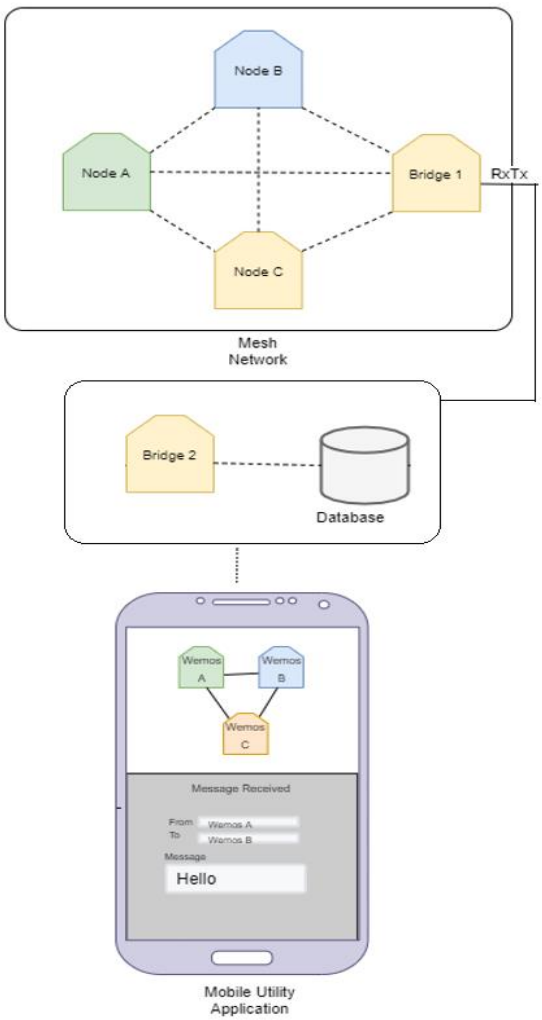

Fig. 1. Mesh network set-up

The setup of the Mesh Network during the functionality testing of the study consists of five (5) ESP8266 Enabled

Module devices. The three (3) devices acted as nodes while the other two (2) devices acted as a bridge. The bridges are computer-programmed in this context. Fig. 1 shows the mesh set up used in this study.

The final prototype of the actual Mesh Network setup contains three (3) ESP8266 modules: Node A, Node B, and Node C as seen in Fig. 1. The nodes can be logically expanded to more than three (3), but not limited to the actual mesh network. The other two(2) ESP8266 modules acted as a bridges that communicate with each other via serial RxTx.

One (1) module is connected to the Wi-Fi while the other one is connected to the actual mesh network. The Actual Mesh Setup is wirelessly connected to Bridge 1 via WiFi module whilst Bridges 1 and 2 are wired connected via RxTx (Receive and Transmit). Bridge 2 is then connected to a dedicated database that connects to the utility app. The Database served as the repository of processes and data transmissions, which are used to help the utility mobile application in sending commands to transmit packets, view online devices or active nodes, help the bridge check for new commands from the utility app, return results, and update connected devices.

A. *Modified painless Mesh-based Agent and Algorithm*

The study modified and utilized a painlessMesh library of ESP devices to simulate the mesh network. In order to achieve objectives, agents and algorithms were developed and built under painlessMesh. These agents and algorithms can perform tagging identity of nodes in the network, finding routes and alternative routes for all nodes, and identifying nodes that are connected and disconnected in the mesh network. Agents' algorithm can be viewed in Fig. 2, Fig. 3 and Fig. 4 respectively.

PseudoCode Algorithm in Identity tagging:

1. *Get hardware/MAC address of the device*
2. *Generate a value with the shifted bits of the hardware address*
3. *Set the value as the device's id in the mesh network*

```
uint8_t MAC[] = {0, 0, 0, 0, 0, 0};
if (WiFi.softAPmacAddress(MAC) == 0) {
  Log(ERROR, "init(): failed to get mac address\n");
}
uint32_t nodeId = tcp::encodeNodeId(MAC);
if (nodeId == 0) Log(ERROR, "NodeId set to 0\n");
```

```
uint32_t encodeNodeId(const uint8_t *hwaddr) {
  uint32_t value = 0;

  value |= hwaddr[2] << 24;
  value |= hwaddr[3] << 16;
  value |= hwaddr[4] << 8;
  value |= hwaddr[5];
  return value;
}
```

```
nodeId = mesh.getNodeId();
mesh.setName(nodeId);
```

Fig. 2. Painless-based agent algorithm

PseudoCode Agent in Finding Routes and Alternative Routes

1. *Get the list of APs (Access Points) in the network*
2. *Sort the APs by signal strength (RSSI)*
3. *Connect to the AP with the best signal strength (RSSI)*

```
aps.sort([](WiFi_AP_Record_t a, WiFi_AP_Record_t b) {
  return a.rssi > b.rssi;
});

connectToAP();
```

Fig. 3. Agent in finding route and alternative routes

PseudoCode Agent in Identifying connected and disconnected Nodes in Mesh Network:

1. *Get the list of known node ids*
2. *Iterate through each node id*
3. *Traverse the node tree*
4. *check if there is a connection to the given node id*

```
for (int i = 0; i < deviceCount; i++){
  if(mesh.isConnected(deviceIds[i]))
    doc[String(deviceIds[i])] = true;
  else doc[String(deviceIds[i])] = false;
}

bool contains(protocol::NodeTree nodeTree, uint32_t nodeId) {
  if (nodeTree.nodeId == nodeId) {
    return true;
  }
  for (auto&& s : nodeTree.subs) {
    if (contains(s, nodeId)) return true;
  }
  return false;
}
```

Fig. 4. Agent in identifying connected/disconnected nodes

B. *One-way Delay*

One-way delay measures the time that a packet transmits from node1 to node2 across the network. The formula contains t0, t1, t2, and t3. t0 is the time when the sender sent a message, t1 is the time when the receiver received the message, t2 is the when the receiver sent a reply, and t3 is when the sender received the reply from the receiver. The equation (1) shows the formula of One-Way Delay as discussed in a study [9].

$$One - way\ Delay = (t3 - t0) - (t2 - t1)/2 \quad (1)$$

For the One-way Delays, the study sent messages with different bit sizes. For example, a specific node sent a message containing "Hello World", with a bit size of 88 bits. When sending this message, t0 is recorded as 1946.045 seconds in the mesh network's time. The t1 was recorded after the receiver node received the request. Here, t1 was recorded to be 1946.083 seconds. Subsequently, the receiver sent back a reply, which recorded t2 to have a value of 1946.088 seconds. The sender then received the reply and the time for t3 will be recorded, which is 1946.139 seconds.

Following the formula of One-Way Delay, we get a value of 0.0445 seconds with the process having a duration

of 0.094 seconds. This is done by subtracting the start time of the requests (t0) from the end time (t3). We then can compute the network's data rate by dividing the bit size by the duration. Here we got a value of 936.1702428 bits per second.

The average One-Way Delay of all tests done is 0.02947 seconds, with an average duration of 0.06667 seconds, and a data rate of 1072.2469363222 bits per second.

Fig. 5 depicts the behavior of One-Way Delay Test and Data Rate when subjected to different messages of different bits. Higher One-Way Delay is expected to occur to messages with higher bits. Data Rates show stability when data bits are the same throughout the duration. For example, transmitted data for test number 4,5, and 6 are all 56 bits. Graph shows stability of expected and computed One-Way delay for the 3 tests.

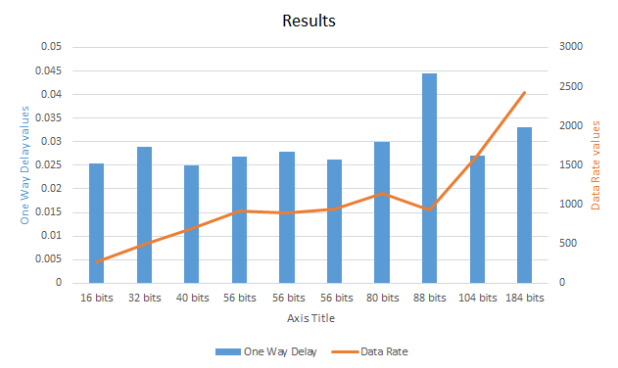

Fig. 5. One-way delay result

C. Payload Size

Payload Size Testing provides an instrument that measures the messages sent/received per second based on the payload duration time since the sending/receiving started up until it is finished, the number of messages sent per second out from the sender, and messages received per seconds at the receiver. Equation (2) shows formula for Payload Size [9]. This part of the testing measured the maximum performance that can be achieved by the ESP8266. The study performed Payload Size Tests on three(3) different messages as seen in Table I.

m/s = messages sent/received per second
m = number of messages
t_s = time payload sending/receiving started
t_f = time payload sending/receiving finished

$$m/s = \frac{m}{(t_f - t_s)} \tag{2}$$

As part of the payload size testing, variance computation is also performed for each run test of sending and receiving. Equation (3) shows variance formula. This allowed the developers to view the behavior of the message rate in sending and receiving.

$$\text{variance } \sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n} \tag{3}$$

where:

x_i = the i^{th} data point
$\bar{x}$ = the mean of all data points
n = the number of data points

Based on the result of the Payload Size Testing, the developers sent three consecutive messages with the same text and bit sizes to test the consistency of the message rate being sent and received. The actual message rate by the sender and receiver nodes provided consistent results having acceptable % differences. While the results of the actual and programmed messages per seconds are not totally equal–which is expected to happen–the result is still acceptable since the % difference is below 10% [11] (See Table II)

TABLE I: SAMPLE PAYLOAD SIZE RUN TEST

Test No	Message Rate when sending				Message Rate when receiving		
	Programmed	Actual	% Diff	Rem	Actual	% Diff	Rem
1	10	10.99	9.44%	Accept	6.89	1.22%	Accept
2	10	11.00	9.55%	Accept	6.71	3.85%	Accept
3	10	10.86	8.29%	Accept	7.32	4.87%	Accept
	Ave	10.95	9.09%	Accept	6.98	3.31%	Accept
	Var	0.004		Var	0.06		

TABLE II: SAMPLE PAYLOAD SIZE VARIANCE OF ACTUAL TEXT

Text Value	1st Test Run	2nd Test Run
wakokok	0.026536776	0.0039159284
malayan	0.0067875201	0.026601466
hello world	0.0070587734	0.001304594

The payload size results suggests that the message rate when sending and receiving has a low variance value which indicates a good rate for each test run. Similarly, the variance of the test runs is not moving away from each other displaying consistency. This suggests that the results of each transmission in sending or receiving does not move away from the expected mean.

D. Developed Utility Application

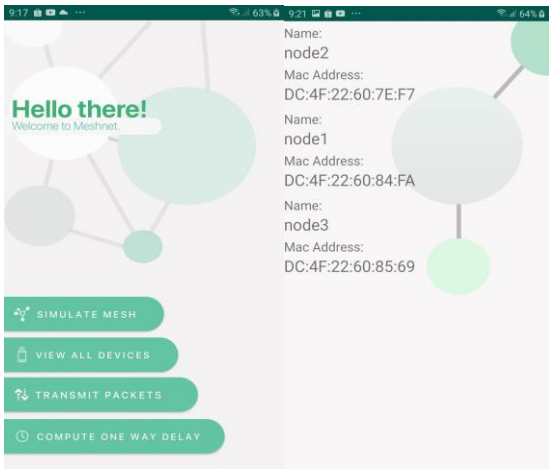

Fig. 6a. Main and view menu

The developed utility app connects to the simulated network via a WIFI module. Fig. 6a-6e show the screenshot of the developed application. The Main Menu of the Meshnet (utility app) features four options available (from top to bottom): Simulate Mesh, View All Devices, Transmit Packets, and Compute One Way Delay. In the View Devices module, users can view all online and connected devices in the mesh network as well as device information consisting of the devices' names and MAC addresses. The list automatically updates once a node disconnects from the mesh network.

In the Transmit Packets module, the app allows a node to node sending a packet (packets will only be limited to text). If the transmission was successful, the app will display the results which include the measured delay and the bit size of the packets.

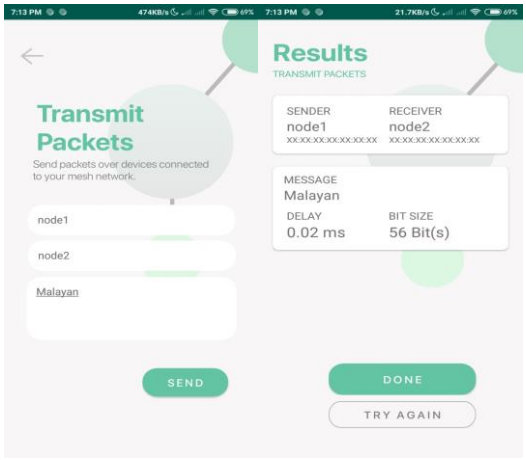

Fig. 6b. Transmit packet module

In the One-Way Delay module, the app will measure the time that a packet takes from node1 to node2 that travels across the network. The result will display alongside the values of the time the sender sent the message (t0), the time when the receiver received the message (t1), the time when the receiver sent a reply (t2), and the time when the sender received the reply from the receiver (t3).

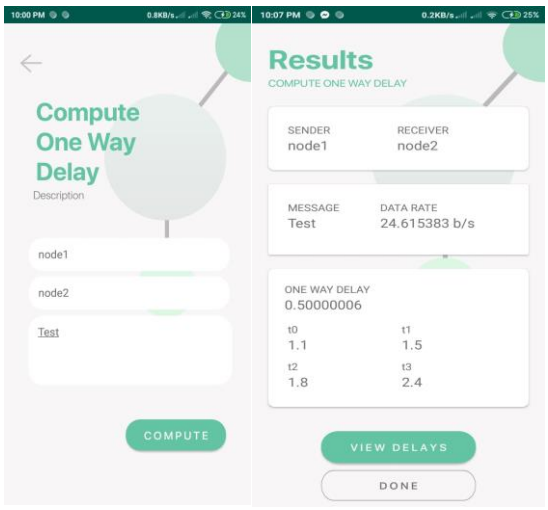

Fig. 6c. One-way delay module

In the modules (Fig. 6d and 6e), users can simulate a mesh network without a device simply by adding virtual devices to the list, which they can also update or remove later on. With the use of added virtual devices, users are able to virtually transmit packets and compute one-way delays within the simulated mesh network.

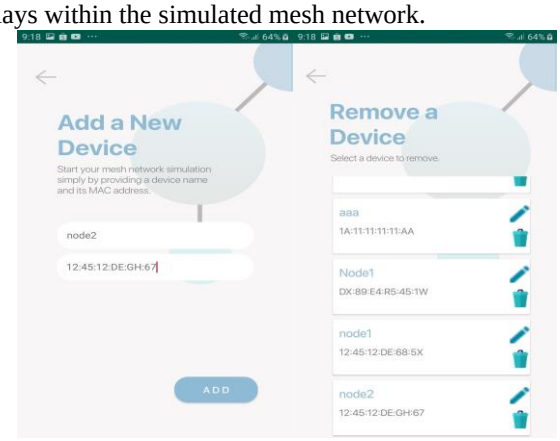

Fig. 6d. Adding/Removing device module

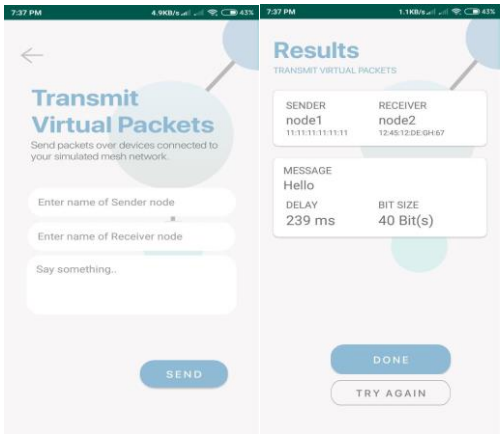

Fig. 6d. Transmit virtual packet module

E. Mesh Simulation Logs

Node Self Identification Using Chip ID

On startup, a node will immediately identify itself using its unique board/chip ID so each node will be able to differentiate each other and know which node to send data.

```
CONNECTION: newConnectionTask(): adding 3947617551 now= 22844152
{"command":"init","nodeId":"576749303","nodeMac":"DC:4F:22:60:7E:F7","nodeName"
CONNECTION: stationScan(): Malayan
```

Fig. 7a. Node self identification logs

Sending data from one node to another

In Fig. 7b, a sender node receives a command from the bridge to send a one-way delay request from another node. After sending the request, the receiver node receives the request and then sends a reply to the sender node. After the sender node receives the reply, it sends the result to the bridge and updates the database which is then picked up by the mobile application. The sender was able to identify which node to send the data to and the receiver was able to identify the original sender and send back a reply because of the unique chip ID they identified themselves with when they connected to the network. Payload Test output can be viewed in Fig. 7c.

Fig. 7b. Node transmission logs

Payload Testing logs

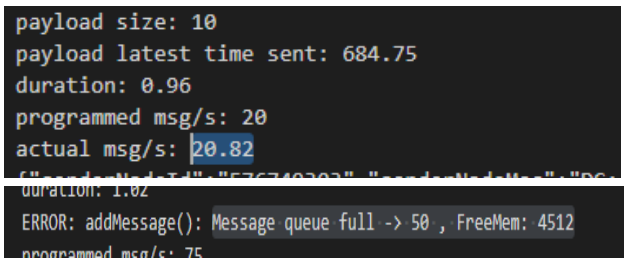

Fig. 7c. Payload testing logs after a test

IV. Conclusions and Recommendations

The study was able to simulate a portable mesh network setup by using ESP8266 devices. These mentioned devices were able to provide portable connectivity among other devices. Additionally, the study able to utilize ESP8266 devices that could provide wireless endpoints connectivity towards each other through the developed agent that identifies devices within the network. During the initial development of the agent, the devices were to identify themselves using its node name registered in the database but was later changed to its native chip ID to avoid failure when a connection problem to the database occurs.

Furthermore, the study utilized and modified a painlessMesh library that would help the ESP8266 identify each device to form the network setup as the nodes were able to connect to each other and form the mesh network. Each node can serve as both an access point for other nodes to connect to, and a station while also being capable of finding the best alternative connections when a single node disconnects. However, the devices using the painlessMesh Library to form a mesh network cannot have a simultaneous connection with both the mesh network and the internet which is why the study made use of two bridge devices communicating together via RxTx, In this setup, one is connected to the mesh network while the other one is connected to the internet.

The study was able to develop a utility app that could monitor the ESP8266 modules during the simulation of the mesh network. The app was able to detect if a new device has been connected or disconnected to and from the mesh network, displays vital mesh information such as the data rate, one-way delay, device name, and MAC address.

Above all, the study was able to exhibit a simulated process of a portable mesh network using only dedicated ESP8266 devices. While this is just a simulation using simple node technology, results of One-Way Delay and Payload Size indicate consistency which can be used as a basis in extending further into more high-end nodes like handheld devices and even computers. Similarly, the utilization of painlessMesh in this study proved that portability can be achieved to avoid loss of connection when nodes abruptly fail in the network. One or two tests indicate however a chance of a bad message rate score during the initial payload size experiment, but this is attributed to the fact that the device is still initializing in the first tests. Succeeding tests however indicate that the message rate stabilized and measured with acceptable values, nonetheless.

It is highly recommended to test in multiple nodes as this study has experimented in a three (3) to five (5) node mesh network only. This will allow stretching the Payload Size computation even further that will assess the consistency of the mesh network. The study also highly recommends encryption. Encryption was not implemented on the Mesh Network utility app as the project was intended for simulation purposes only. Future researchers are recommended to expand the scope of the project and improve the features of the utility app by adding encryption that will help improve the security of the network. Furthermore, the study is only limited to viewing and sending of messages in the form of short text and string through the network. While this is a good source of token and packets, the developers recommend larger packets and object type data which can only be possible if the nodes are also capable of handling massive packets. This can also allow Simulation Application to be adjusted in such a way that large packets from external sources like images and other file types can be transmitted in the portable network.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

D.A Martillano is the main author and responsible for the overall technical and hardware set-up, general analysis processes, and evaluation of procedures. R.C. Dupaya is in charge of executing the hardware setup including the modifications of the agents and algorithm. K.V. Lavina assisted in performing tests and completing each executing cycle for Data Collection. J.P Asiddao and P.A. Sy concentrated in the development of the utility application. Conclusion and Recommendations were conceived by the whole research team.

ACKNOWLEDGMENT

Presentation of this paper is supported by Malayan Colleges Laguna.

REFERENCES

[1] F. Solahuddin. (2016). Xbee Pocket Manual. 10.13140/RG.2.1.1540.0089. [Online]. Available: https://www.researchgate.net/publication/299448733_Xbee_Pocket_Manual

[2] S. Sajeeb, A. Uzzal, and M. Tahzib-Ul-Islam, "A survey on wireless mesh network and its challenges at the transport layer," *International Journal of Computer Engineering and Technology*, 2014.

[3] L. Yu, T. Kin-Fai, Q. Xiangdong, L. Ying, and D. Xuyang, Wireless Mesh Networks in IoT Networks, 2017, pp. 183-185.

[4] N. D. Castagnoli and N. J. Friday. (2005). Automatic route configuration in hierarchical wireless mesh networks. [Online]. Available: https://patents.google.com/patent/US7899027B2/en

[5] A. Siemuri, "Wireless mesh network implementation. smart energy systems research platform SESP," University of Vaasa, 2019.

[6] B. Tran, "Mesh network personal emergency response appliance," Patent (US, 2011/0115624A1). United States Patent Publication, 2011.

[7] L. J. H. Kumar, S. K. Deep, Y. Hadjadj-Aoul, and N. Kumar, "Wireless and mobile technologies for the internet of things," *Mobile Information Systems*, Hindawi Publishing Corporation, 2016.

[8] V. C. Gungor, *et al*. (2008). Architectures and Protocols for WMNs. Springer. [Online]. Available: https://www.utc.fr/~enataliz/dokuwiki/_media/en/springer2008.pdf

[9] C. Yoppy, R. Arjadi, S. Endah, W. Priyo, and S. Muhammad, "Performance evaluation of ESP8266 mesh networks," *Journal of Physics: Conference Series*, 2019.

[10] A. O. Adeoye. (2009). A Framework for the Self-Configuration of Wireless Mesh Networks. Semantic Scholar. Computer Science. [Online]. Available: https://pdfs.semanticscholar.org/f9cb/203809ba92a75bec11fe06a2e05f92d0f389.pdf

[11] D. A. Martillano, J. M. R. Dita, C. G. Cruz, and K. S. Sadhra, "Android based real-time industrial emission monitoring system using IoT technology," *Journal of Communications*, vol. 12, no. 11, pp. 623-629.

Dennis A. Martillano is a graduate of Bachelor of Science in Computer Engineering and holds a master's degree in Information Technology. He took up Interdisciplinary Studies (Art Studies) and Graduate Degree in Distance Education from the University of the Philippines. He completed his academic requirements in Doctor in Information Technology where he is specializing Medical Data Mining for his Dissertation in Technological Institute of the Philippines. Currently, he is a lecturer in the College of Computer and Information Science at Malayan Colleges Laguna, Philippines.

Roy Carlos B. Dupaya obtained his degree in Bachelor of Science in Information Technology with specialization in Mobile and Web Technology from Malayan Colleges Laguna. He participated in the 20th IT Skills Olympics at the University of Makati in 2019 representing Malayan Colleges Laguna for Web Design. During his internship at OOCL (Philippines) Inc., he experienced QA testing of their shipment documentation system.

Patricia Alexandra A. SY obtained her degree in Bachelor of Science in Information Technology major in Mobile and Web Technology from Malayan Colleges Laguna. She specializes in UI/UX design for web and mobile applications. In 2019, she was a part of the duo that represented Malayan Colleges Laguna in the 20th IT Skills Olympics at the University of Makati in the Web Design category. She also joined the ASEAN Data Science Explorers in 2019. During her internship at Binan City Hall, she learned and Application Lifecycle Management Advance Course and LoadRunner 12.0 by Micro Focus Software University. During his internship at the Biñan City Hall, she was able to develop an Inventory System for the Information and Communications Technology Office.

Ken Brian B. Laviña obtained his degree in Bachelor of Science in Information Technology with specialization in Mobile and Web Technology from Malayan Colleges Laguna. He is knowledgeable in several programming languages such as PHP, HTML, Javascript, C#, Java, and C++. In 2018, He participated and completed the course in Application Lifecycle Management Advance Course and LoadRunner 12.0 by Micro Focus Software University.

his research outputs are in the areas of Machine Learning, Networking and Communication, and Internet of Things.

John Patrick M. Asiddao obtained his degree in Bachelor of Science in Information Technology with specialization in Mobile and Web Technology from Malayan Colleges Laguna. He is knowledgeable in programming languages such as C#, Java, HTML, and PHP. He participated and completed the course in Application Lifecycle Management Advance Course and LoadRunner 12.0 by Micro Focus Software University. Throughout his internship, he experienced using Oracle BI Publisher in OOCL (Philippines) Inc. and developed an android-based mobile app for the Students Affairs Office in Malayan Colleges Laguna