Performance Evaluation of ESP8266 Mesh Networks

IOP Conf. Series: Journal of Physics: Conf. Series **1230** (2019) 012023 doi:10.1088/1742-6596/1230/1/012023

Performance Evaluation of ESP8266 Mesh Networks

Yoppy[1], R Harry Arjadi[1], Endah Setyaningsih[2], Priyo Wibowo[1], M I Sudrajat[1]

[1] Research Center for Quality System and Testing Technology (P2SMTP-LIPI), Indonesia
[2] Universitas Tarumanagara, Indonesia

E-mail: yoppy.chia@gmail.com

Abstract. Existing WiFi mesh networks are usually implemented on high-end or PC grade platforms. However, the open source community has been recently developing a unique mesh network library targeted for the low-cost and resource limited ESP8266 platform. The so called *painlessMesh* library enables two or more ESP8266 modules to self-configure and to form a WiFi mesh network. This might open up new potential for the ESP8266 to be used in wider application areas. The library is in an early development stage and not much is known about its performance. Therefore, this paper aims to evaluate to what extent the ESP8266 painlessMesh network can perform, in terms of one-way delay and data rate. Measurements showed that a 2-node network has a delay of 2.49 ms. A network consisting of higher number of nodes tends to have an increased network delay even for the same hop distance. Meanwhile, data rate measurements showed that for the case of 10-byte payload a node can receive up to 461 messages/sec. Whereas for payload of 4400 bytes, the node can receive up to 28 messages/sec. Furthermore, it can be reported that payload greater than 4400 bytes starts causing incomplete and erroneous messages.

1. Introduction

When it comes to low cost WiFi SoCs (System on a Chip) in the IoT world today, the ESP8266 is undoubtedly on top of the list. Despite its inexpensive price, the ESP8266 features a number of crucial core and peripherals, such as 32bit 80MHz Xtensa Tensilica processor, 32KB instruction RAM, 80KB user-data RAM, 4MB flash ROM, RF frontend for IEEE 802.11 b/g/n WiFi, GPIO, I^2C, ADC, and UART. The ESP8266 is able to run both the TCP/IP stack and application codes in itself without using an external host microcontroller. Moreover, the necessary sources for development of the ESP8266 are made open and accessible. Combination of the above factors has drawn the attention of many developers to explore and experiment with it.

Mesh network has been implemented in several wireless protocols, such as LoRa [1], [2], [3], Bluetooth Low Energy [4], [5], [6] and ZigBee [7]. It has also been realized in WiFi technology, mainly with PC grade hardware or running on a Linux operating system [8], [9], [10], [11]. Those WiFi mesh libraries are designed for scalability to accommodate large scale networks.

Normally, WiFi devices including the ESP8266 are operating in a star network in which one or more stations (STA) are connected to a central access point (AP). However, since the ESP8266 also has the feature of functioning as both STA and AP at the same time [12], the open source community has been developing a library that enables the ESP8266 nodes to form a mesh network. The so called **painlessMesh** library is unique in which it allows the low cost and resource limited ESP8266 devices to form a WiFi mesh network, whereas other mesh network solutions are mainly implemented on high-end platforms. However, the performance of this relatively new library is still not much known. Therefore,

this paper aims to evaluate to what extent the ESP8266 painlessMesh network can achieve, particularly in terms of one-way delay and data rate.

2. Background

The so called *painlessMesh* is a WiFi mesh network library specifically written for the ESP8266 platforms, and is actively maintained by the community. It is called painlessMesh for it is intended to be auto-configure and easy to setup. It is an ad-hoc network which requires neither routing plan nor central controller, and all nodes are equal.

The painlessMesh library is unique for it enables the resource limited ESP8266 device to form a mesh network. Two or more modules, also called nodes, with the same SSID (service set identifier) will be automatically connected to form a mesh network. The topology is in fact not a full mesh network. Due to constrained resources, cyclic paths are actively avoided so that routing tasks can be greatly simplified. The formed network actually is more resembling a star network. However, it differs from a usual WiFi star network that typically consists of a central access point to which one or more stations are connected. In the painlessMesh network, each node can serve as both an access point for other nodes to connect to, and also a station connecting to another node.

Every node is aware of the whole network topology which is updated periodically every 3 seconds. Every node tells its neighbors about other nodes it is directly or indirectly connected to. A node's station which is not connected to any access point will actively look for an access point that has the strongest signal but is *not* listed yet in its network topology. This mechanism prevents the formation of cyclic paths. So there will be only a single path between a pair of nodes. Figure 1 shows the diagram of a painlessMesh network.

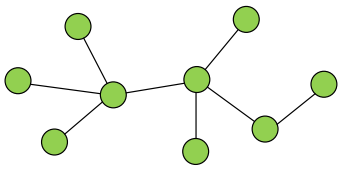

Figure 1. Diagram of painlessMesh network.

The painlessMesh works on layer 3 but not based on IP addressing. Instead, each node is distinguished by a unique 32-bit number taken from the chip MAC address. Communication in the mesh is done via JSON formatted messaging. Although it is less efficient than low level binary messaging, it is easier for users to understand and to incorporate in Javascript or web applications. There are two types of user message, i.e. single and broadcast message. Single message is sent from a node to another specific node, whereas broadcast message is addressed to all other nodes connected in the network. All nodes are time synchronized with a precision of less than 10 ms. This is useful for the nodes to run synchronous tasks [13]. The painlessMesh library used in this experiment is of version 1.0.1.

Mesh network offers its usefulness by allowing a collection of ESP8266 nodes to cover larger area through multi-hop messaging. Any nodes within reachable range will be automatically forming a mesh network. Therefore, the mesh network also leverages the network reliability by automatically joining to a neighboring network in the event of disconnection. These features may open up new potential applications.

For example, this might find applications in smart LED lighting in passenger trains. The lighting in every coach is controlled by a node. Every node is connected to its neighboring nodes and such that forming a linear network. This solution might be impractical in the usual star topology because the total length of a passenger train is most likely out of the WiFi coverage.

3. Measurement methods

3.1. One way delay

One-way delay measurement is illustrated in Figure 1, and is calculated as follow:

$$One\ way\ delay = \frac{(t_3 - t_0) - (t_2 - t_1)}{2} \quad (1)$$

t_0 : delay measurement request is sent,
t_1 : the request reaches the responding node,
t_2 : a reply is sent back to the requesting node, and
t_3 : the reply arrives at the requesting node.

It is effectively the half of round trip time. Four cases of number of nodes are put on trial, i.e. 2 nodes, 3 nodes, 10 nodes, and 16 nodes. In the first three cases all modules employed are the NodeMcu ESP8266, whereas in the last one, it is a mix of 10 modules of NodeMcu and 6 modules of Wemos D1 Mini, like shown in Figure 3. Despite of being different modules, they have the same memory and processing power since they are employing the same ESP8266 WiFi chips. All modules are flashed with codes that are doing minimum tasks. The user codes barely do nothing other than blinking an LED and printing messages to serial port. In this way, it can be regarded as the best performance that can be achieved.

One of the nodes in the mesh network is assigned as the measuring node that sends delay measurement requests to all other nodes. The measuring node is connected to a computer for logging purposes.

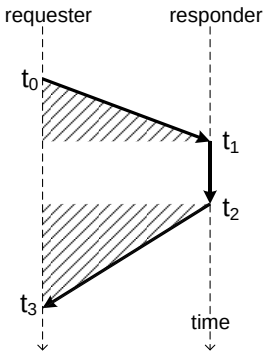

Figure 2. Delay measurement.

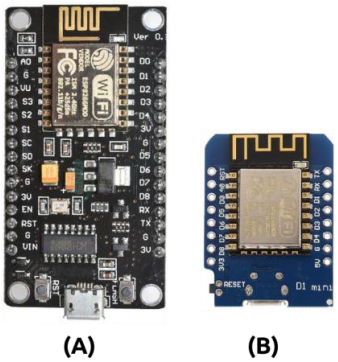

Figure 3. NodeMcu (A) and Wemos D1 Mini (B)

While it is possible to force a station to connect to a specific access point, in this experiment all nodes are left free and randomly connected to available APs. In the case of 2 nodes, a straightforward point-to-point network is formed. Likewise, in the case of 3 nodes a simple network is created, in which the measuring node is in the middle of the other two nodes. However, for the case of 10 and 16 nodes, more complex topologies are formed, like shown in Figure 4 and Figure 5, respectively. Node X represents the measuring node.

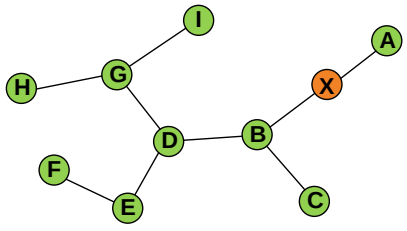

Figure 4. Mesh network of 10 nodes.

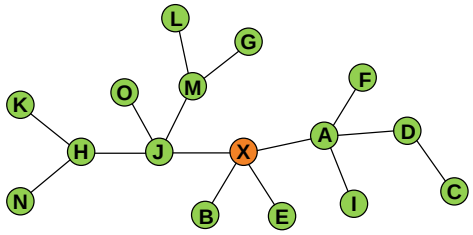

Figure 5. Mesh network of 16 nodes.

3.2. Data rates

The following measurement approach is used to find out the maximum data rate that can be achieved by a node in the mesh network. To minimize the overhead of managing multi-node topology, only two nodes are employed in the data rate measurements. One node is designated as a packet sender, while the other as a receiver. To obtain the best performance, user tasks are set minimal. Other than sending or receiving messages, the user tasks are only blinking LED and printing information to the serial port.

The sender is programmed to send messages at various payload sizes (from 10 to 4400 bytes) and rates (from 200 to 1000 messages/second). The actual number of messages received at the receiver is recorded. For each payload size, at least 100 seconds of measurement timespan are captured, and the average value is then calculated.

4. Measurement results

4.1. One-Way Delay

Box plots in Figure 6, Figure 7, Figure 8, and Figure 9 demonstrate the distribution of the measured delays. Interpretation of the box plot can be explained as follow. The top and bottom of the box represent the third and the first quartiles. The height of the box is defined as interquartile range (IQR). The line in the box is equal to its median. While both whiskers signify the highest and lowest datum within 1.5xIQR extending from the third and first quartile, respectively. The small circles represent individual occurrences that are regarded as outliers.

Two-node network delay, as shown in Figure 6, is distributed between 2.32 ms and 3.54 ms, with 75% of data lie below 2.58 ms and a median of 2.49 ms. Being the simplest network consisting of two nodes, this result could be regarded as the best performance that can be achieved by the ESP8266 painlessMesh. Figure 7 summarizes the distribution of 1634 measured delays in a 3-node network in which one node is acting as the requesting node, while the other two nodes (A and B) are functioning as the responding nodes.

In Figure 8 and Figure 9, it is observed that maximum delays in the 10-node network are occasionally higher than those of in the 16-node network. Assuming there are no flaws in the library implementation, these counterintuitive results may be caused by the complex memory management and tasks scheduling that are happening on the lower layer. Longer delays may be caused by shortage of free memory available to the nodes at some points in time.

During experiment, it is observed that a 16-node mesh network is quite unstable which is indicated by failing to join all nodes into a single network. Higher number of nodes exhibits even more probability to broken network.

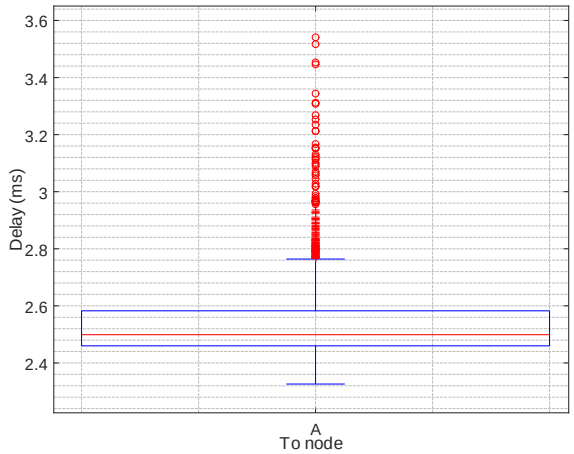

Figure 6. Two-node network delay. Delay request is sent every second. 2023 measurements are collected.

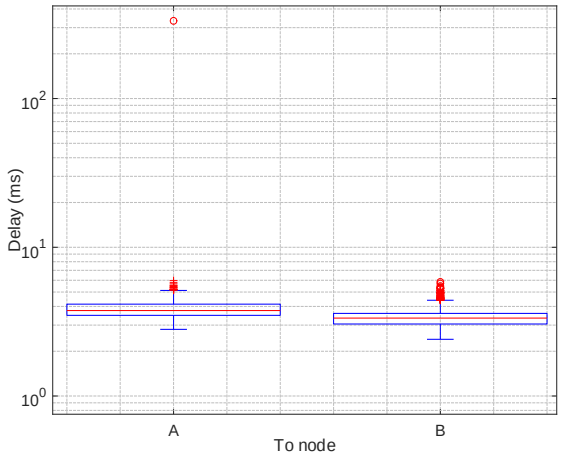

Figure 7. Three-node network delay. Delay request is sent every two seconds. 1634 measurements are collected.

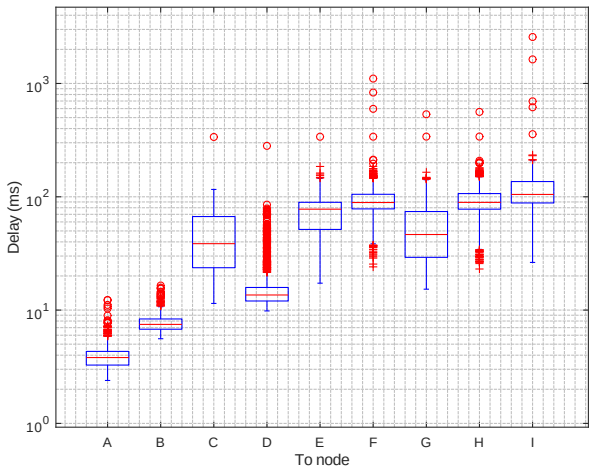

Figure 8. Ten-node network delay. Delay request is sent every two seconds. 1400 measurements are collected.

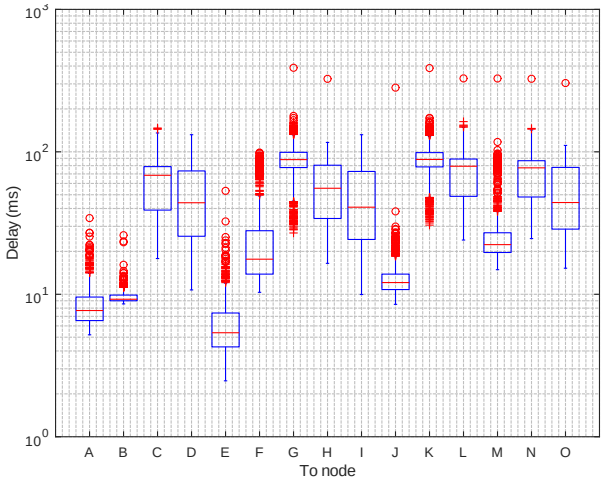

Figure 9. Sixteen-node network delay. Delay request is sent every two seconds. 1230 measurements are collected.

4.2. Data Rate

Table 1 shows the average data rate measurement results. It is observed that at higher payload size, the actual number of messages sent out from the sender can be less than what is programmed. For example, at payload of 1500 bytes, the sender is programmed to send 200 messages/second. In fact, it can only send 149.6 messages/second. This happens because the resource limited ESP8266 is not fast enough to process the due tasks.

On the receiver side, it can also be seen that the capability of receiver to process incoming messages is lower than that of the sender has dispatched. For instance, at payload size of 10 bytes, the sender is sending 1000 messages/sec. However, the receiver can only receive 461 messages/sec. The other messages are lost.

These measurement results can be regarded as the maximum performance that can be achieved by the ESP8266. Also, it can be reported that payload larger than 4400 bytes starts causing corrupted messages.

Table 1. Average data rates.

Payload size (bytes)	Messages/sec from sender *(programmed/actual)*	Messages/sec received at receiver
10	1000 / 1000	461.00
20	1000 / 1000	455.92
500	500 / 373.53	370.56
1500	200 / 149.6	139.85
2000	200 / 117.18	87.94
3000	200 / 76.42	72.80
4400	200 / 57.65	27.85

5. Conclusion

Performance of ESP8266 mesh network has been tested and evaluated. A netwok consisting of two nodes has a single hop delay of 2.49 ms. As expected, higher number of nodes tends to increase the network delay even for same hop distance. More importantly, during measurement it is observed that a network of 16 nodes seems to be unstable as it occasionally fails to join all nodes. The instability is more evident for higher number of nodes.

Evaluation is also done for data rate performance. It is found that a node can receive up to 461 messages/sec and 28 messages/sec for a payload of 10 bytes and 4400 bytes, respectively. It is also observed that sending messages with payload greater than 4400 bytes results in broken and incomplete messages.

Although exhaustive tests have not yet been done, these measurement results can give rough estimation and impression on the performance of the painlessMesh network. Based on the measurement results, users can justify the suitability of ESP8266 painlessMesh in their applications.

References

[1] Ochoa M N, Guizar A, Maman M and Duda A 2017 Evaluating LoRa energy efficiency for adaptive networks: From star to mesh topologies *Int. Conf. Wirel. Mob. Comput. Netw. Commun.* **2017**–**October**
[2] Lee H C and Ke K H 2018 Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation *IEEE Trans. Instrum. Meas.* 1–11
[3] Dvornikov A, Abramov P, Efremov S and Voskov L 2017 QoS Metrics Measurement in Long Range IoT Networks *Proc. - 2017 IEEE 19th Conf. Bus. Informatics, CBI 2017* **2** 15–20
[4] Wang S-D and Chiang K-J 2017 BLE Tree Networks for Sensor Devices in Internet of Things *2017 IEEE 15th Intl Conf Dependable, Auton. Secur. Comput. 15th Intl Conf Pervasive Intell. Comput. 3rd Intl Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr.* 1304–9
[5] Chiumento A, Reynders B, Murillo Y and Pollin S 2018 Building a connected BLE mesh : a network inference study 296–301
[6] Leonardi L, Patti G, Bello L L O and Member S 2018 Multi-Hop Real-Time Communications Over Bluetooth Low Energy Industrial Wireless Mesh Networks *IEEE Access* **6** 26505–19
[7] Rodriguez M G, Uriarte L E O, Jia Y, Yoshii K, Ross R and Beckman P H 2011 Wireless sensor network for data-center environmental monitoring *2011 Fifth Int. Conf. Sens. Technol.* 533–7
[8] Riggio R, Gomez K, Rasheed T, Gerola M and Miorandi D 2009 Mesh your senses: Multimedia applications over WiFi-based wireless mesh networks *2009 6th IEEE Annu. Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Networks Work. SECON Work. 2009* **00** 4–6
[9] Riggio R, Miorandi D, Chlamtac I, Scalabrino N, Gregori E, Granelli F and Fang Y 2008 Hardware and software solutions for wireless mesh network testbeds *IEEE Commun. Mag.* **46** 156–62
[10] Gomez K, Riggio R and Miorandi D 2009 Mice over mesh: HTTP measurements over a WiFi-based wireless mesh network *2009 7th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wirel. Networks* 1–6
[11] Ridolfi M, Van De Velde S, Steendam H and De Poorter E 2016 WiFi ad-hoc mesh network and MAC protocol solution for UWB indoor localization systems *2016 IEEE Symp. Commun. Veh. Technol. Benelux, SCVT 2016*
[12] ESP8266 Datasheet 2015 ESP8266EX Datasheet *Espr. Syst. Datasheet* 1–31
[13] https://gitlab.com/painlessMesh/painlessMesh. [Accessed: 14-Aug-2018]

Acknowledgments

This research is supported by the Ministry of Research, Technology, and Higher Education of the Republic of Indonesia through INSINAS 2018 funding scheme.