JURNAL INFOTEL
Informatics - Telecommunication - Electronics
Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Design and Analysis of Trash Monitoring System Prototype Based On Internet of Things (IoT) Using MQTT Protocol

Achmad Auliyaa Zulfikri[1*], Doan Perdana[2], Gustommy Bisono[3]
[1,2,3]Telecommunication Engineering, School of Electrical Engineering, Telkom University
[1,2,3]Jalan Telekomunikasi Terusan Buah Batu Bandung 40257
*Corresponding email: achmadauliyaa@gmail.com

Received 23 July 2018, Revised 15 August 2018, Accepted 17 August 2018

Abstract - On this research, Internet of Things (IoT) as an advanced technology is used to monitor the height of trash from a trash can in order to give notification whether the height of trash is already reaching the maximum height limit or not. To support those needs, we used NodeMCU as a microcontroller, ultrasonic sensor, MQTT as IoT protocol, and Android to display the data. After we did the system performance test, we had the biggest result of end-to-end delay which was 2.06875 seconds when the packet delivery was set to 1000 ms with 3 active nodes and the smallest result of end-to-end delay which was 0.26055 seconds when the packet delivery was set to 100 ms with 1 active node. The biggest result of throughput was 597.17 Bytes/s when the packet delivery was set to 100 ms with 1 active node and the smallest result of throughput was 75.86 Bytes/s when the packet delivery was set to 1000 ms with 3 active nodes. The biggest result of availability and reliability was 99.905% when the packet delivery was set to 1000 ms and the smallest result was 99.833% when the packet delivery was set to 100 ms.

Keywords – Internet of Things, NodeMCU, ultrasonic sensor, MQTT

I. INTRODUCTION

Development of science and technology developed rapidly. Even each second we had experienced a new development either from the side of science or technology. The society needs towards information technology which reliable, flexible, and real-time had been fulfilled due to the rapid development of wireless technology [1]. According to research [2], with the quick development of science and technology, it could ease all of our activities and the man works. Therefore, a network that can be accessed by anyone, anywhere and at anytime is needed. One of them who meets those criteria is Internet of Things (IoT). There had been many studies that use such technology to be associated with the needs of daily life. On research [3], IoT was designed for a home security system, furthermore, on research [4], IoT was designed to detect human movement. Two researches above were some examples of the utilization of IoT in everyday life. On this research, IoT was designed to monitor the height of the trash on a trash can in the housing. It was expected to prevent the buildup of waste in every trash can so that the janitors would get a notification through the Android about the trash that had been reached the height limit and should be transported. It was also expected to ease the task of janitors so they did not have to go to the all housing so that they could concentrate to transport trash from a trash can that gave notification to the Android only. To meet those needs, we used NodeMCU, HC-SR04 ultrasonic sensor, internet network, Message Queue Telemetry Transport (MQTT) as the Protocol in IoT, and Android to display the data of the trash can height. In addition, this research was tested based on a system performance test. The parameters of the test were delay, number of packets, throughput, availability, and reliability of the system with the scenario of changement in the number of nodes and changement of delivery break time. Further, an analysis of the obtained result was conducted.

Research [5] had created a trash can system monitoring with notification of the rising level of the trash through social media. However, in this research, the altitude level notification was sent through Android. The other differences from this research were the performance test that had been done on the system. The performance test parameters were delay, throughput, availability, and reliability.

Then, on research [6], a similar system was already made. However, the difference from this research was the microcontroller that was used. They used Arduino Uno while on this research NodeMCU was used as a microcontroller. Also, the MQTT broker that was used was also different. They used Blynk for MQTT broker while this research used Mosquitto for MQTT broker. The other difference on this research were performance test that had been done on the system. The performance test parameters were delay, throughput, availability, and reliability.

On research [7], delay test had been done for MQTT protocol with time delivery gap difference scenarios. However, it used more than one sensor and yet using local network. On this research, the test was done using only one sensor and using a public network.

In the research [8], it discussed the performance of MQTT and HTTP specifically for bandwidth usage. On this study, it focused on measuring the performance of MQTT with a scenario of changement in the number of nodes and changement in the delay time for sending messages with test parameters of delay, throughput, availability, and reliability.

Furthermore, research [9] compared the performance of MQTT with Constrained Application Protocol (COAP). It could be seen that several parameters were compared such as delay, bandwidth usage, and PDR. However, this study focused at the MQTT performance with the existing scenarios namely changing in the number of nodes and changement in message delivery time especially to find out the reliability of MQTT based on the interval of message delivery time to the MQTT broker which was seen from the value of availability and reliability.

The main purpose of this research was to apply IoT in an everyday task which was trash height monitoring as already explained previously. The other purpose was to analyze the performance of the usage of the MQTT protocol in the system. The tested parameters were delay, throughput, availability, and reliability.

IoT is possible to be a human dependence in daily life. Based on two earlier researches that had already been explained and also the example from this research. It was predicted in 2020 that the number of devices that connected to the Internet approximately about 50 billion devices. The devices will not only smartphone or computer device but also objects that were frequently used in daily life [10]. The same thing was also mentioned in research [11] that in the future computer will take over part of human jobs such as electronic devices controlling from far distance through the internet network. The definition of IoT, according to [12], was an invention which able to resolve the existing problems through the merger of technology with social problems in terms of standardization technique as global infrastructure. To meet the standard needs of the information society which allow advanced services with the ability of interconnection, another source described that a connection between daily things such as a smartphone, sensor, and actuator with human or thing itself through internet network was defined as IoT in [13]. In addition, ITU-T was made a formula of the system description for IoT which were already in common use as the picture below.

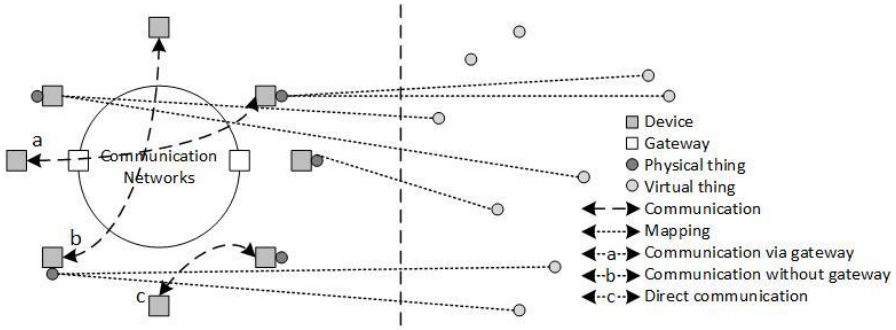

Fig.1. Technical Description of The System of IoT [12]

Figure 1 illustrates the technical description of the IoT system which more precisely formulated as machine-to-machine (M2M) communication and device-to-device (D2D) communication. Therefore, the deployment of IoT could be interpreted as M2M communication. On block (a) of Fig.1, the technical system connects the device with another device using a gateway and telecommunication network, in this case, it is an internet. Meanwhile, free WiFi acted as a gateway.

II. RESEARCH METHOD

A. Message Queue Telemetry Transport (MQTT)

Message Queue Telemetry Transport (MQTT) was a protocol that runs over TCP/IP protocol. MQTT was known as a protocol that had small packet size, low power supply needed, and used publish/subscribe system as work principle based on research [14]. MQTT protocol was used as IoT support according to research [15]. MQTT was suitable to connect M2M communication

On M2M communication, there were other protocols that commonly used in the worldwide such as REST and WebSocket. Both of them were HTTP based on research [16]. Also, this research discussed the comparison between MQTT and HTTP. As already explained previously, MQTT is a protocol that had small packet size and required only small resources. However due to MQTT had small packet size, it also required smaller bandwidth. This had been proved in the research [8] that the use of MQTT required less bandwidth than the use of HTTP if only more devices were connected. MQTT is suitable for a system that has limited resources as had already been explained previously. It happened because MQTT used a publish/subscribe method where subscribers or clients do not need to periodically update the data because each client or subscriber will automatically receive the latest data according to the subscribed topic so that it could save the use of resources due to the decrease in the computing process [16].

However, even though MQTT has those advantages, it also has weaknesses. There are 3 levels of Quality of Service (QoS) on MQTT protocol based on research [17] as follows:

a) QoS level 0: "At most once delivery"
At this level, the message was sent only once without confirmation from the sender so that the message will arrive once or not at all. In other words, it allowed the message to be sent are missing [17].

b) QoS level 1: "At least once delivery"
At this level, the message would at least be received once by the client. If the client did not receive the message, the sender will resend the message with the DUP bit. However, in this case, it allowed the duplicate message to be sent [17].

c) QoS level 2: "Exactly one delivery"
At this level, the message will be received exactly once without any failure or duplicate message [17].

Due to MQTT had 3 levels of QoS as explained previously, a higher QoS level, the message delivery delay time would be greater [16] and the need for bandwidth consumption would also be greater [10] although it would not significant. It is one of the MQTT weakness.

Figure 2 shows the working scheme of the MQTT protocol. It can be seen that there are two main components which were needed to implement the MQTT protocol. There are MQTT broker and MQTT client. On this research, the MQTT broker that was used for research purpose was Mosquitto and MQTT client which existed on the monitoring application as output.

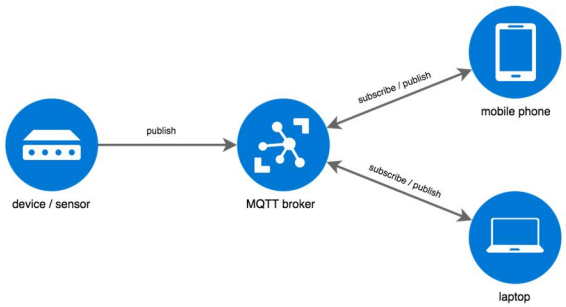

Fig.2. MQTT Protocol Working Scheme [18]

Mosquitto is an open source message broker (EPL/licensed EDL) which implements MQTT protocol version 3.1 and 3.1.1 [19]. MQTT broker had the same assignment as well as a server which was to receive the published data from NodeMCU. Then, those data were forwarded to the application that had been made. The content of the data that had been sent was about the height of the trash inside the trash can. Those data had a special identity inside MQTT known as a topic. That topic would later be known by the broker to continue to the application so that there were no swapped data from each NodeMCU with application due to the topic that was used must be different.

B. Model Design System

Figure 3 displays that in the first block it contains the hardware that was used in this system prototype using ultrasonic HC-SR04 sensors to measure distance or height of the trash on the trash can that was integrated with a microcontroller. In this case, it was used NodeMCU v3. Then, NodeMCU would be connected to a Wi-Fi modem as described in the second block. Once connected with Wi-Fi modem, then NodeMCU could send the generated data by the sensor to a server which in this case it was used Mosquito as MQTT broker planted in a Virtual Private Server (VPS) through the internet network. After the data were processed by the server, then the data could be accessed by local janitors through an application that had been created.

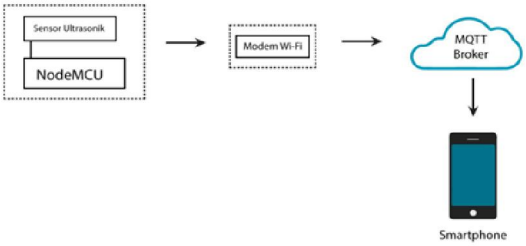

Fig.3. Model Design System

C. Execution Flowchart

Figure 4 describes the system execution flowchart. First of all, the initialization process was done by NodeMCU to activate the sensor as well as the process of connecting internet network through WiFi modem on NodeMCU. Then, the sensors would pick up the data periodically and would be processed in the NodeMCU. It did notify the height of the trash in the trash can whether it was over the specified height limit or not yet. When the data that obtained from the sensor had passed the specified height limit, then the system would give a notification directly to a smartphone application. It was made to perform trash can monitoring if it had passed the height limit. However, if it had not crossed the line that was specified at the certain height of the trash, it would remain on view state through the display application for monitor the height of the trash. Furthermore, it had conducted a system performance test.

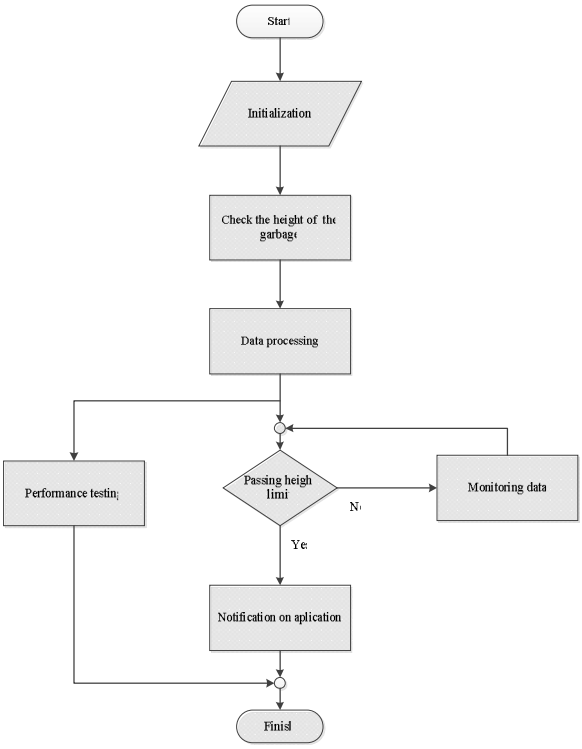

Fig.4. Execution Flowchart

D. Test Scenario

In this test scenario, the objectives were to test the functionality and the system performance whether the overall functionality and system performance test went well or not. The functionality of system includes testing on hardware and software. While testing the system performance includes delay, throughput, availability, and reliability. System performance testing scenarios were performed at the Table 1:

Table 1. Test Scenario

No	State
1	When the active node 1 sending data with break 100 ms
2	When the active node 2 sending data with break 100 ms
3	When the active node 3 sending data with break 100 ms
4	When the active node 1 sending data with break 500 ms
5	When the active node 2 sending data with break 500 ms
6	When the active node 3 sending data with break 500 ms
7	When the active node 1 sending data with break 1000 ms
8	When the active node 2 sending data with break 1000 ms
9	When the active node 3 sending data with break 1000 ms

Table 1 explains that there were 9 scenarios with differentiation on active node number and differentiation on break time delivery. Break time delivery data is the delivery gap data from NodeMCU to MQTT broker that used break time delivery of 100 ms, 500 ms, and 1000 ms. To calculate the performance of the system, the author used Wireshark as used in research [7]. The experiment testing conducted for 30 times from each script and performed for 60 seconds for each experiment.

E. Performance Test Parameters

The performance test was an attempt to find out the system performance that resulted in this research. The parameters tested were as follows:

a) Delay

The definition of delay according to research [20], was the time required in the package or data delivery from the sender to the receiver. The calculation formula of the delay is as follow:

$$Delay = \frac{\text{length of packets (bit)}}{\text{bandwith of networks } (\frac{\text{bit}}{\text{s}})} \quad (1)$$

b) Throughput

The definition of throughput is the data transfer speed which actually ongoing in a process of data transmission. Usually, served in units of B/S (bytes per second) or bps (bits per second) [20]. The throughput calculation formula is as follow:

$$\text{Throughput} = \frac{\text{the number of data sent(byte)}}{\text{total data transmission time (s)}} \quad (2)$$

c) Reliability and Availability

Reliability is the ability of the sent data by the device of the user which would be up on the side

of the sender device within a particular observation. While availability is average time over the long term of the ratio between the time length of data or system that can perform their function over time in total. The calculation of reliability and availability are as follows:

$$\text{Realibility} = \frac{(\text{sent}-\text{failed})}{\text{sent}} \text{ x } 100\ \% \quad (3)$$

$$\text{Availability} = \frac{\text{sent}}{(\text{sent}+\text{failed})} \text{ x } 100\ \% \quad (4)$$

Reliability and availability were calculated on the overall system which means between the sensor to the application. The purpose of the formula as explained above where to send the successful server packages receipt or MQTT broker to the application and failure server packages that failed to send to the application.

III. RESULT

Having previously been described in chapter II point D which is about the testing scenario, then at this point will be presented test results that have been conducted namely include: hardware and software functionality testing and performance testing system includes: delay, throughput, and reliability and availability. Performance testing scenarios are conducted based on Table 1.

A. Hardware Testing

Hardware testing is to find out that the functionality of the hardware used on a system that is functioning already made appropriate functions are designed. For testing functionality that is done can be seen in Table 2.

Table 2. Testing The Functionality of Hardware

No	Hardware Function	Description
1	NodeMCU integration with Sensors HC-SR04	successfully
2	NodeMCU can be connected to the internet network	successfully
3	The sensor can detect the height of the garbage	successfully
4	NodeMCU can send data from the sensors to the broker MQTT	successfully

B. Software Testing

Software testing is to find out that the functionality of the software that is used on a system that is made already functioning appropriately. For testing functionality that is done can be seen in Table 3.

Table 3. Testing The Functionality of Software

No	Hardware Function	Description
1	Broker MQTT can receive data published by NodeMCU	successfully
2	MQTT broker can forward the data obtained to the application	successfully
3	The application can receive the data forwarded by the broker MQTT	successfully
4	The application may provide notification when the height limit has already reached the limit	successfully

Fig.5. The Display of Monitoring Application

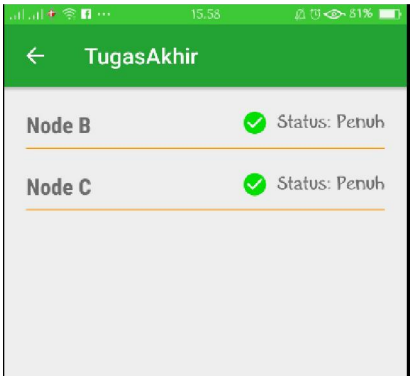

Fig.6. The View to Know The Trash Can is Already Full

Figure 5 is a monitoring display of the height data from the trash can and Fig.6 is a view to know whether the trash can is full or not yet.

C. Results of Delay Toward Changes in The Number of Nodes

The measured delay is a delay between NodeMCU with MQTT broker and the delay between MQTT broker with applications, then from 2 delay can be added to get end to end delay. Measuring delay using software Wireshark mounted on side of MQTT broker so Wireshark can observe each package delivered by NodeMCU as well as the package in the forward to the application. At this point will be discussed regarding the comparison of delay to changes in the number of nodes in each delivering breaks time i.e. the time of 1000 ms, 500 ms, and 100 ms and to change the number of its nodes namely

when the node 1 active, node 2 active, and node 3 active.

Table 4. The Results of The Delay Toward Changes in The Number of Nodes at Break Delivering 1000 ms

The Number Of Active Nodes	1	2	3
The average delay between the NodeMCU with the broker (s)	1,0109	1,0311	1,0327
The average delay between the broker with the application (s)	1,0121	1,0315	1,0360
End to End Delay Average (s)	2,0231	2,0626	2,0687

Table 5. The Results of The Delay Toward Changes in The Number of Nodes at Break Delivering 500 ms

The Number Of Active Nodes	1	2	3
The average delay between the NodeMCU with the broker (s)	0,5212	0,5275	0,5302
The average delay between the broker with the application (s)	0,5251	0,5278	0,5303
End to End Delay Average (s)	1,0463	1,0553	1,0604

Table 6. The Results of The Delay to Changes in The Number of Nodes at Break Delivering 100 ms

The Number Of Active Nodes	1	2	3
The average delay between the NodeMCU with the broker (s)	0,1302	0,1369	0,1376
The average delay between the broker with the application (s)	0,1303	0,1376	0,1385
End to End Delay Average (s)	0,2606	0,2745	0,2761

Table 4-6 are the result of the effect of the number of nodes on the delay at each delivery break time, the biggest delay is at the time of 3 active nodes, it occurs either when using delivery break at 1000 ms, 500 ms, or 100 ms and the smallest delay is at the time of 1 active node, it happens whether using 1000 ms, 500 ms, or 100 ms delivery break.

D. Results of Delay Toward Changes The Break Time Delivery

At this point, the delay is measured the same as a scenario a measurement delay toward changes in the number of nodes that is a delay between NodeMCU with MQTT broker and the delay between the MQTT broker with the application and then from 2 delay can be added so it will get end to end delay. At this point would be seen the impact of break time delivery against the resulting delay, testing is done using break delivery 1000 ms, 500 ms, and 100 ms where testing is conducted on the conditions of each active node number namely when 1 active node, 2 active nodes and when 3 active nodes. The break time is the delivery time on the program at NodeMCU to determine the time interval NodeMCU do publish data obtained from sensors to MQTT broker, for example when setting 1000 ms then the NodeMCU will be posting data to broker every 1 second of all.

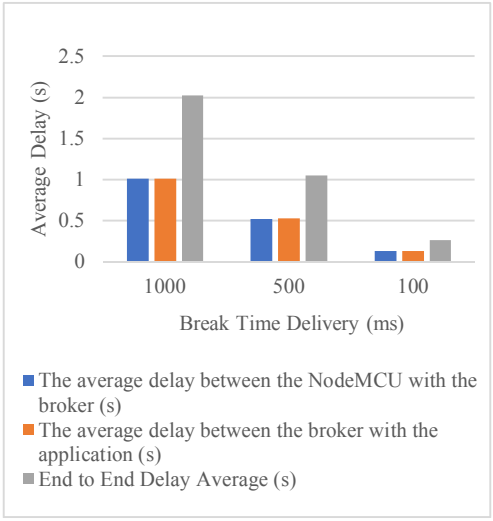

Fig.7. Delay Comparison Chart Toward Change Delivery Time With 1 Active Nodes

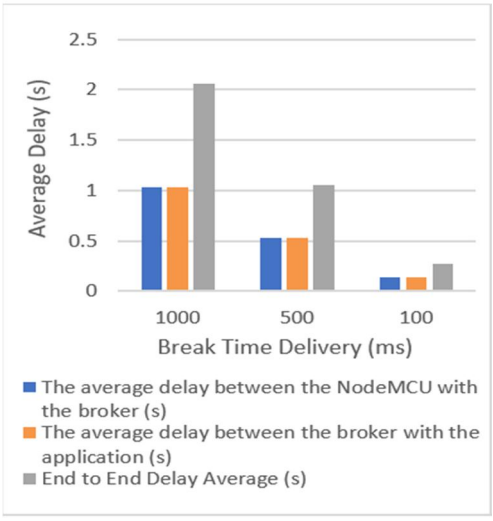

Fig.8. Delay Comparison Chart Toward Change Delivery Time With 2 Active Nodes

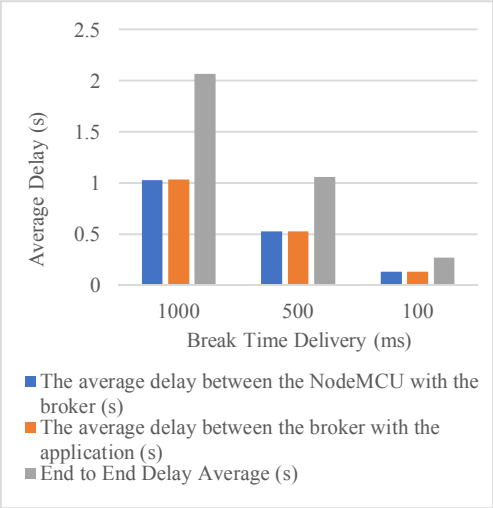

Fig.9. Delay Comparison Chart Toward Change Delivery Time With 3 Active Nodes

Figure 7-9 are the result of the effect of the time of delivery break towards the delay at each active node condition, the biggest delay occurs when using 1000 ms delivery break, it occurs whether the condition is at 1 active node, 2 active nodes and 3 active nodes and the smallest delay exists when using 100 ms delivery break, it happens whether the condition of 1 active node, 2 active nodes or 3 active nodes.

E. The Results of The Number of Packets and The Throughput TowardChanges The Number of Nodes

For the calculation of Throughput very effected towards the number of packets, because in equation (2) elaborated that the amount of the package is directly proportional to the large throughput obtained, therefore at this point will be elaborated as well test results against the number of packets, before going to come by throughput. For the calculation of throughput and the number of the package can be seen through Wireshark after performing data retrieval. At this point would be seen the impact of a number of nodes against the amount of the package as well as the resulting throughput.

Table 7. The Results of The Number of Packets Toward Changes in The Number of Nodes at Break Delivering 1000 ms

Number Of Nodes	The Average Number Of Packets Received Broker MQTT	Average Number Of Packets Sent From The Application To The Broker
1	59,2	59,17
2	58	57,9
3	57,69	57,62

Table 8. The Results of The Number of Packets Toward Changes In The Number of Nodes At Break Delivering 500 ms

Number Of Nodes	The Average Number Of Packets Received Broker MQTT	Average Number Of Packets Sent From The Application To The Broker
1	114,10	113,97
2	112,17	112,10
3	111,49	111,36

Table 9. The Results of The Number of Packets Toward Changes In The Number of Nodes At Break Delivering 100 ms

Number Of Nodes	The Average Number Of Packets Received Broker MQTT	Average Number Of Packets Sent From The Application To The Broker
1	458,70	458,20
2	436,38	435,68
3	433,62	432,60

Table 7-9 showed the impact of the change of the number of nodes against the number of packages, it can be seen that the greater number of the node then the number of packages has decreased while using break time delivery at 1000 ms 500 ms or 100 ms, it of course also affects the throughput values decrease as there is in the Table 10-12.

Table 10. The Results of The Throughput Toward Changes In The Number of Nodes At Break Delivering 1000 ms

Number Of Nodes	Average Throughput Between NodeMCU with Broker MQTT (B/s)	Average Throughput Between MQTT Broker with the application (B/s)
1	65,27	77,2
2	64,18	75,97
3	64,03	75,86

Table 11. The Results of The Throughput Toward Changes In The Number of Nodes At Break Delivering 500 ms

Number Of Nodes	Average Throughput Between NodeMCU with Broker MQTT (B/s)	Average Throughput Between MQTT Broker with the application (B/s)
1	125,97	148,80
2	123,75	146,28
3	123,27	145,58

Table 12. The Results of The Throughput Toward Changes In The Number of Nodes At Break Delivering 100 ms

Number Of Nodes	Average Throughput Between NodeMCU with Broker MQTT (B/s)	Average Throughput Between MQTT Broker with the application (B/s)
1	506,10	597,17
2	481,20	567,82
3	478,16	564,52

Table 10-12 showed the impact of the change of the number of nodes against the value of throughput, it can be seen that the greater number of the node then the value of throughput has decreased while using delivery break time at 1000 ms, 500 ms, or 100 ms.

F. The Results of The Number of Packets and The Throughput to Changes The Break Time Delivery

Having previously been analyzed comparison of the number of packets and the throughput toward changes the number of nodes, the points will discuss the comparison of the number of packets and the throughput to changes the break time delivery on each condition of the active node that is when the active node 1, 2 and 3.

Figure 10-12 are compared of the number of packages for each break time delivering in conditions of active node 1, 2 and 3, the faster the shows from that figure the break time of delivery, the greater number of packages will be sent.

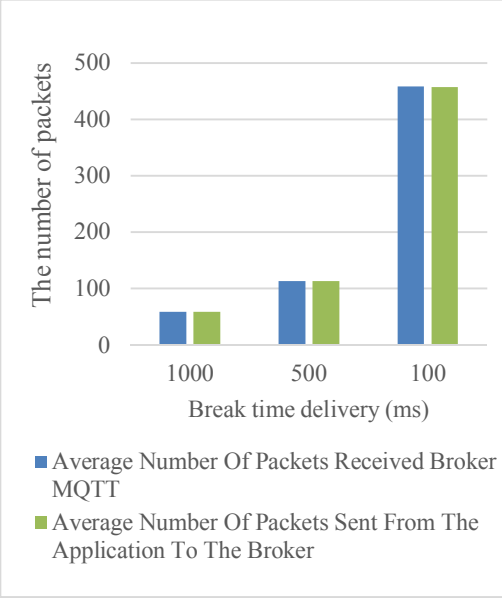

Fig.10. Comparison Chart of The Number of Packets Toward The Change of Delivery Break With 1 Active Node

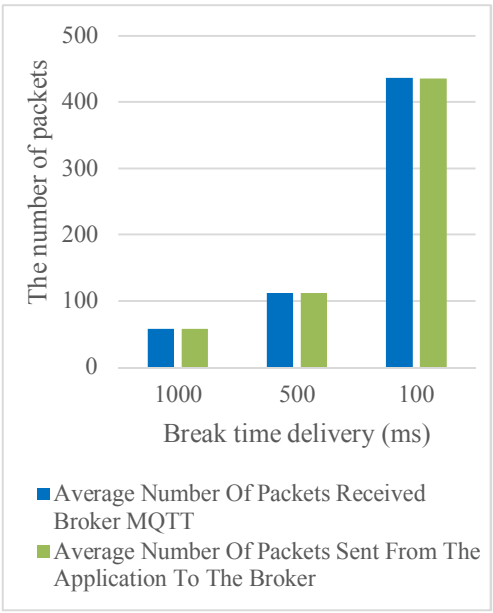

Fig.11. Comparison Chart of The Number of Packets Toward The Change of Delivery Break With 2 Active Nodes

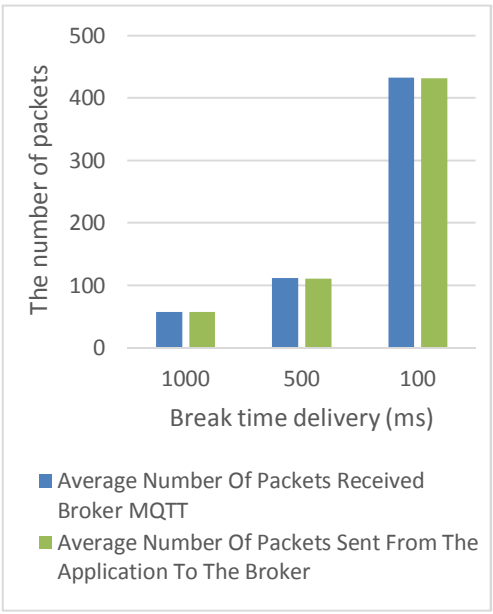

Fig.12. Comparison Chart of The Number of Packets Toward The Change of Delivery Break With 3 Active Nodes

Figure 13-15 are the result of the effect of the delivery break time towards the value of throughput generated at each active node condition, the greatest throughput exists when using 100 ms delivery break, it occurs whether the condition of 1 active node, 2 active nodes or 3 active nodes and the smallest throughput exists when using 1000 ms delivery break, it occurs whether the condition of 1 active node, 2 active nodes or 3 active nodes.

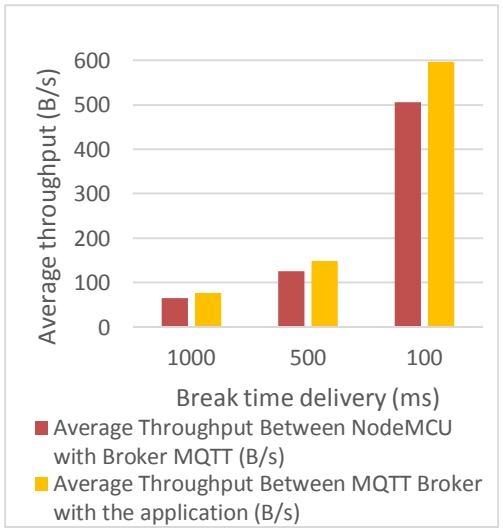

Fig.13. Comparison Chart of The Throughput Toward The Change of Delivery Break With 1 Active Node

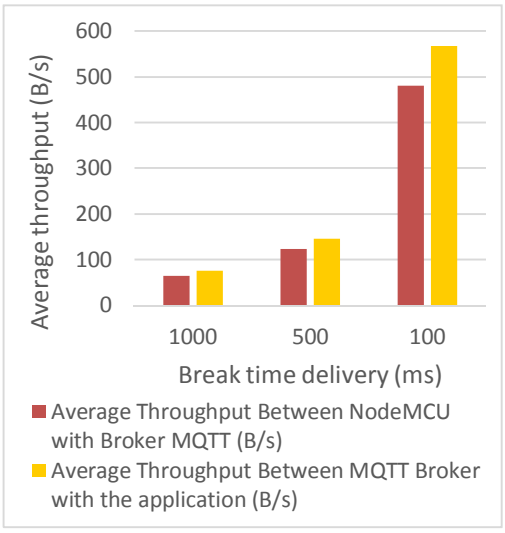

Fig.14. Comparison Chart of The Throughput Toward The Change of Delivery Break With 2 Active Nodes

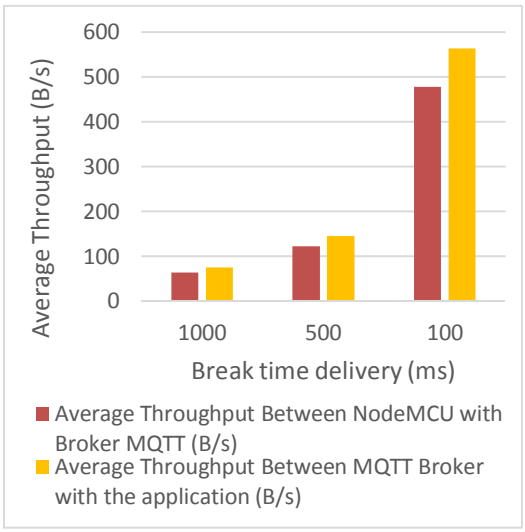

Fig.15. Comparison Chart of The Throughput Toward The Change of Delivery Break With 3 Active Node

G. The Results of Availability and Reliability System

Reliability and availability for the measurement of the overall system that is done when the data is sent from NodeMCU to get to the application. Availability and reliability depend on the number of packets that failed to receive by the application, the number of packets failed and the availability and reliability values below are the results of the effect of the time interval for sending data from NodeMCU to the broker.

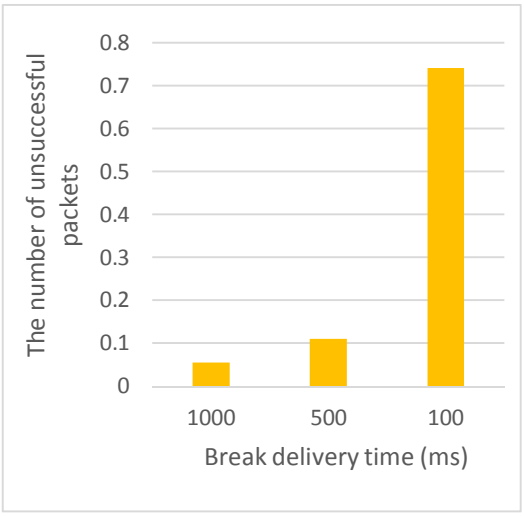

Fig.16. Comparison Chart of The Number of Packets That Failed Are Forwarded To The Application

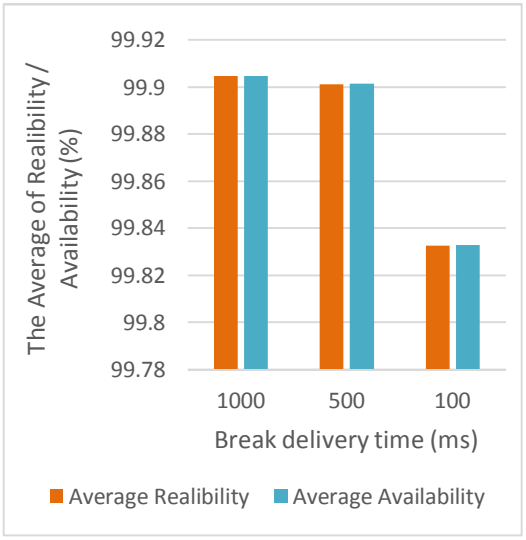

Fig.17. The Comparison Chart of Reliability and Availability System

Figure 16 is the comparison of the number of failed packets forwarded to the application at each delivery break time, it can be seen that the number of failed packets mostly when using 100 ms delivery break time, there is a package that does not reach the recipient because in this study using level 0 Quality of Service. Then Fig.17 is the comparison of availability and reliability of the system at each delivery break time, the smallest availability and reliability of the

system exist when using 100 ms delivery break and the biggest when the delivery break is at 1000 ms.

IV. DISCUSSION

A. Analysis of Delay Toward Changes in The Number of Nodes

Table 4-6 show that the number of nodes affects the resulted delay. The number of nodes, the bigger the delay was, despite the increment level of the delay was not that great. It occurred either using delivery break time of 1000 ms, 100 ms, or 500 ms. Meanwhile, when using delivery break time of 1000 ms, the least value of end-to-end delay happened in the condition of 1 active node was 2.0231 seconds and the greatest value in the condition of 3 active nodes was 2.0687 seconds. Meanwhile, when using delivery break time of 500 ms, the least value of end-to-end with the condition of 1 active node was 1.0463 seconds and the greatest value with the condition of 3 active nodes was 1.0604 seconds. It was the last tests using 100 ms delivery break time with the smallest value of end-to-end delay with the condition of 1 active node was 0.2605 seconds and the greatest value with the condition of 3 active nodes was 0.2760 seconds. At most of the time, the delivery gap was indicated as an increase in delay with the number of nodes getting bigger. This was similar to research [19] when the delay resulted in getting increased then the greater number of nodes would be. MQTT had published section and subscribe section. When there were data in the publish to the broker, the broker would forward it to the client who subscribes to the data. The first data in the publish section would automatically be received by the broker and also would be forwarded to the client.

If there was only 1 node that parses publish data to the broker, then the broker would only handle those data from the node. However, when there was more than 1 node that parses publish data to the broker, then the broker would handle the data one by one. It was the one that might cause the delay getting increased when the number of nodes increased too due to the broker would handle incoming single data so that the data entry should be waited for the later process by the broker. Therefore, they reach destination longer.

B. Analysis of Delay Toward Changes The Break Time Delivery

Figure 7-9 show that changes break time delivery can affect the resulting delay. On research [7] can also be seen that the results of the average delay become smaller if the break time delivery faster, it occurs when both the conditions of the active node 1,2 or 3. Most great delay when using break delivery 1000 ms and the least when using delivering 100 ms break. The interlude time delivery program at NodeMCU to determine the time interval NodeMCU do publish data come by to MQTT Broker. When set 1000 ms (1 second) then the NodeMCU will be posting publish data to broker every 1 second of all, when the broker gets the data from the NodeMCU then the broker will immediately forward those data to the application with the appropriate topic, because the application will subscribe to continuous data that is in the broker. Therefore, if publish data conducted by NodeMCU getting faster, then the broker will also receive data faster and forward the data to the application, so that way the application will also receive data more quickly. It was the one that caused the break effect on delivering time delay generated.

C. The Analysis of The Number of Packets and The Throughput Toward Changes The Number of Nodes

Table 10-12 indicate that the value throughput tends to decrease with the number of nodes, it is because the number of packages produced also tended to decline due to the number of packages is directly proportional to the throughput value generated.

As already described in the analysis of the impact of the number of nodes against delay i.e. within the broker MQTT handle singly data by NodeMCU, then for a time done that is the same as when testing against the impact of break time delivery that is 60 seconds. When there is only 1 active node, then that node can freely send data to brokers over a span of 60 seconds, and the broker will also easily pass on such data to the receiver because there is only 1 sender.

Furthermore, unlike in nodes multiply, then data from the nodes have to queue up to be handled by the broker because brokers handle singly incoming data. It also makes the broker should attempt to handle the received packets in a span of 60 seconds, because the broker handles the incoming data by one basis it makes the broker takes a longer time to handle the incoming data, it is that can make the number of packets that can be handled by the broker will be less if the number of nodes that posting publish more data.

D. The Analysis of The Number of Packets and The Throughput to Changes The Break Time Delivery

Figure 13-15 are a comparison of throughputs for each break time delivery in conditions of active node 1, 2 and 3. The picture showed the faster time of delivery, the throughput values gap is getting bigger, it is caused by the number of packets that are sent more and more as can be seen in Fig.10-12.

On this research the testing done for 1 minute on each of the screenplay to change in break time delivery or change the amount of node, meaning that when the moment of break delivery 1000 ms NodeMCU sending the data to the broker every 1 second once, and then at break delivering 100 ms i.e. means NodeMCU will send the data to the broker every 0.1 second. Thus, it can be seen that in the same span of time is 60 seconds, use the NodeMCU break is faster delivering time will send much more data to

brokers, hence the break effect on delivery incredibly the number of packages that are delivered.

The value throughput effect on the number of packages, as in equation (2) explained that the value of throughput is directly proportional to the amount of data sent, in this case, the total delivery time data has no effect since all scenarios for total the time is the same i.e. approximately 60 seconds, thus the break time delivery because of a change to the number of packages, then it will also directly affect the value of the resulting throughput.

E. The Analysis of Availability and Reliability System

From Fig.16 can be seen that the number of packets forwarded from the broker failed to applications tend to increase over time as increased Queuing break delivery data from the NodeMCU to the broker, from the results that also affects the value of the reliability and system availability because of the equations (3) and (4) the number of packets that failed is inversely proportional to the value of the reliability and the resulting system availability. The value of the reliability and availability of i.e.at the time using the break delivery 1000 ms with a value of 99.905% and the smallest i.e. while using the delivering 100 ms break value 99.833%.

The break time delivery data from the NodeMCU to the broker also affect the time the broker receives the data, with the principle of publish/subscribe broker will forward the data to the client subscribe. If publish data from NodeMCU to broker the delivering time can be arranged, but another case with decisions and applications while downloading data from broker-subscribe, it is done continuously without stopping. In time delivery of data from the NodeMCU to the broker very quickly, then the broker will also receive data very quickly and will have the task of forwarding such data to the application very quickly also or in other words the broker will be very busy so that data that are forwarded is not perfect, that allows some loss of packets or data that should be passed to the application.

V. CONCLUSION

From the result and test analysis, the conclusion that can be taken is the number of nodes, the value of the delay will be higher, and the number of packets and throughput will be lower. The biggest result of end-to-end delay which is 2.06875 seconds when the packet delivery is set to 1000 ms with 3 active nodes and the smallest result is 75.86 Bytes/s when the packet delivery is set to 1000 ms with 3 active nodes. Then based on delivery break time is that the faster delivery break time, the result of the delay will be faster and the number of packets and throughput will be higher,but the value of system reliability and availability tend to decrease due to on this research it uses level 0 QoS so that it is possible that the sent data is lost. The smallest result of end-to-end delay which is 0.26055 seconds when the packet delivery is set to 100 ms with 1 active node, the biggest result of throughput is 597.17 Bytes/s when the packet delivery is set to 100 ms with 1 active node,and the biggest result of availability and reliability is 99.905% when the packet delivery is set to 1000 ms and the smallest result is 99.833% when the packet delivery is set to 100 ms. For further research,it is better if the number of nodes can be added so that the impact of delay, total packets, and throughput can be known much more accurate and also it can be added bad smell detection sensor on system, so that the monitoring is not only based on height but also through the bad smell.

REFERENCES

[1] Muhammad Irfan Denatama,Doan Perdana,and Ridha Muldina Negara,"Analisis Perbandingan Kinerja Protokol Routing DSDV dan OLSR Untuk Perubahan Kecepatan Mobilitas pada Standar IEEE 802.11ah," *Jurnal Infotel*,vol. 8,no. 2,pp. 100-106,November 2016.

[2] RATNASIH,RISKI MUKTIARTO NUGROHO AJINEGORO,and DOAN PERDANA,"Analisis Kinerja Protokol Routing AOMDV pada VANET dengan Serangan Rushing," *ELKOMIKA*,vol. 6,no. 2,pp. 232-243,Mei 2018.

[3] MUHAMAD IRFAN KURNIAWAN,UNANG SUNARYA,and ROHMAT TULLOH,"Internet of Things : Sistem Keamanan Rumah berbasis Raspberry Pi dan Telegram Messenger," *ELKOMIKA*,vol. 6,no. 1,pp. 1-15,Januari 2018.

[4] RAHMI AGUS MELITA,SUSETYO BAGAS BHASKORO,and RUMINTO SUBEKTI,"Pengendalian Kamera berdasarkan Deteksi Posisi Manusia Bergerak Jatuh berbasis Multi Sensor Accelerometer dan Gyroscope," *ELKOMIKA*,vol. 6,no. 2,pp. 259-273,2018.

[5] Rizka Amalia Rufaidah,Denny Darlis,and Hafidudin ,"PERANCANGAN DAN IMPLEMENTASI SISTEM MONITORING TUMPUKAN SAMPAH BERBASIS MIKROKONTROLER DENGAN NOTIFIKASI MEDIA SOSIAL," 2014.

[6] Harshitha N,Nehashree K Ruthika, Rhea Benny, Varsha S P, and Keerthi Kumar M, "IoT based Smart Garbage and Waste Monitoring System using MQTT Protocol," *International Journal of Engineering Research & Technology (IJERT)*, vol. 6, no. 13,pp. 1-5,2018.

[7] Hudan Abdur Rochman,Rakhmadhany Primananda,and Heru Nurwasito,"Sistem Kendali Berbasis Mikrokontroler Menggunakan Protokol MQTT pada Smarthome," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*,vol. 1,no. 6,pp. 445-455,Juni 2017.

[8] Tetsuya Yokotani and Yuya Sasaki, "Comparison with HTTP and MQTT on Required Network Resources for IoT," *International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*,2016.

[9] Niccolò De Caro, Walter Colitti, Kris Steenhaut, Giuseppe Mangino, and Gianluca Reali, "Comparison of two lightweight protocols for smartphone-based sensing," *IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*,2013.

[10] Salsalina Oktaria F. Tarigan,Imanta Herry Sitepu,and Maclaurin Hutagalung,"Pengukuan Kinerja Sistem Publish / Subscribe Menggunakan Protokol MQTT," *Jurnal Telematika*,pp. 25-30.

[11] Apri Junaidi,"INTERNET OF THINGS,SEJARAH,TEKNOLOGI DAN PENERAPANNYA : REVIEW," *Jurnal Ilmiah Teknologi Informasi Terapan*,vol. 1,no. 3,pp. 62-66,Agustus 2015.

[12] ITU-T, "Overview of the Internet of things," 2012.

[13] Mochamad Fajar Wicaksono,"IMPLEMENTASI MODUL WIFI NODEMCU ESP8266 UNTUK SMART HOME," *Jurnal Teknik Komputer Unikom*,vol. 6,no. 1,pp. 1-6,2017.

[14] Periyaldi ,Arief Bramanto W.P.,and Agusma Wajiansyah,"Implementasi Sistem Monitoring Suhu Ruang Server Satnetcom Berbasis Internet Of Things (IOT) Menggunakan Protokol Komunikasi Message Queue Telemetry Transport (MQTT)," *JURNAL TEKNOLOGI TERPADU*,vol. 6,no. 1,pp. 23-29,april 2018.

[15] Totok Budioko,"SISTEM MONITORING SUHU JARAK JAUH BERBASIS INTERNET OF THINGS MENGGUNAKAN PROTOKOL MQTT," *Seminar Riset Teknologi Informasi (SRITI)*,pp. 353-358,2016.

[16] Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, and Jesus Alonso-Zarate, "A Survey on Application Layer Protocols for the Internet of Things," *Transaction on IoT and Cloud Computing*,2015.

[17] OASIS. (2014) MQTT Version 3.1.1. [Online]. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[18] Equan Pr. (2015,October) JavaScript & IoT. [Online]. https://jsiot.pw/mengenal-mqtt-998b6271f585

[19] Gede Okky Satria, Gandeva Bayu Satrya, and Anton Herutomo, "IMPLEMENTASI PROTOKOL MQTT PADA SMART BUILDING BERBASIS OPENMTC," *e-Proceeding of Engineering*, vol. 2, no. 2,pp. 6530-6537,Agustus 2015.

[20] Rika Wulandari,"ANALISIS QoS (QUALITY OF SERVICE) PADA JARINGAN INTERNET (STUDI KASUS : UPT LOKA UJI TEKNIK PENAMBANGAN JAMPANG KULON – LIPI)," *Jurnal Teknik Informatika dan Sistem Informasi*,vol. 2,no. 2,pp. 162-172,Agustus 2016.