International Journal of Advanced Smart Convergence Vol.12 No.2 173-181 (2023)
http://dx.doi.org/10.7236/IJASC.2023.12.2.173

IJASC 23-2-21

Improvement of IoT sensor data loss rate of wireless network-based smart factory management system

Tae-Hyung Kim*, Young-Gon Kim**

* Ph.D Student, Dept. of Computer Engineering, Tech University of Korea, Korea
kth10310@gmail.com

** Professor, Dept. of Computer Engineering, Tech University of Korea, Korea
ykkim@tukorea.ac.kr(corresponding author)

Abstract

Data collection is an essential element in the construction and operation of a smart factory. The quality of data collection is greatly influenced by network conditions, and existing wireless network systems for IoT inevitably lose data due to wireless signal strength. This data loss has contributed to increased system instability due to misinformation based on incorrect data. In this study, I designed a distributed MQTT IoT smart sensor and gateway structure that supports wireless multicasting for smooth sensor data collection. Through this, it was possible to derive significant results in the service latency and data loss rate of packets even in a wireless environment, unlike the MQTT QoS-based system. Therefore, through this study, it will be possible to implement a data collection management system optimized for the domestic smart factory manufacturing environment that can prevent data loss and delay due to abnormal data generation and minimize the input of management personnel.

Keywords*: IoT, MQTT, MQTT-SN, Smart-Factory, Multicast Routing.*

1. Introduction

As the IoT convergence service market such as smart factory expands with the development of technology using digital communication and MCU, the need for accurate data collection such as various environments/facilities/conditions is increasing [1]. Data collection is an essential factor in building/operating such a smart factory, and it is not easy to deploy in the field because data collection depends on the state of the network. In addition, sensors have different formats and transmission methods depending on wired and wireless methods, so accurate information collection is becoming increasingly complex and difficult. Existing studies have been conducted to overcome the shortcomings of wired and wireless sensor methods in collecting such information and to overcome errors caused by signal abnormalities or non-standard data and connection systems [2]. In this work, I aim to improve overall system performance by reducing errors that occur in the detection signal itself or in the acquisition process. This improvement in data quality can contribute to lower

Manuscript Received: May. 18, 2023 / Revised: May. 24, 2023 / Accepted: May. 29, 2023
Corresponding Author: *ykkim@tukorea.co.kr*
Tel: +82-31-8041-0530, Fax:+ 82-31-8041-0539,
Professor, Dept. of Computer Engineering, Tech University of Korea, Korea

operational/maintenance costs and improved product yield by reducing misreporting rates and minimizing manager intervention. Through this, it is possible to overcome errors due to signal abnormalities or non-standard data and connection systems. The domestic IoT and smart factory related markets are modularized by combining small sensor devices with small or research MCUs. Therefore, it is somewhat difficult to digitize and accurately interpret the sensor's signals and recognize and process them equally on multiple platforms. If you look at a single system, each has its own personality and is a great system, but even if it is a similar environment, similar products, and similar platforms, it has the disadvantage of having to newly define and build all parts. It is difficult to establish a smooth and accurate system in areas where strong standardization organizations for data, signals, and systems are still insufficient. In addition, the role of a manager for IoT-based equipment /facilities /information collection is essential, and minimizing this requires active intervention or management of systems such as data errors, misdetection, and false alarms [3].

In smart factory, MQTT-based message delivery methods are commonly used for data collection and transmission. Considering the challenging conditions of establishing network infrastructure, it is typical to deploy wireless networks. The wireless communication technologies used include WiFi, ZigBee, and Bluetooth, but they often suffer from limited communication range and transmission instability. Moreover, in the conventional setup where sensors are directly connected to a single gateway, difficulties arise in transmitting data properly when the wireless environment deteriorates. To address this issue, a large number of gateways are required. In this study, I propose a multi-gateway system utilizing MQTT-SN and MQTT to explore optimal transmission paths and ensure system stability in wireless environments by delivering data through multiple gateways.

2. Previous Research and principal Background

2-1. MQTT (The Message Queuing Telemetry Transport) Overview

MQTT protocol is a broker-based IoT middleware widely used in IoT systems. MQTT is a lightweight message protocol in the publish/subscribe model for IoT environments that uses MQTT brokers to manage subscriber interest as topics, which can be hierarchically configured to efficiently connect a large number of sensors and devices [4]. The location of the broker is important because the MQTT protocol exchanges data over topics, and all devices with the same topic must be connected to the broker. Broker-based protocols can cause data concentration problems and single point of failure problems because many devices connect to the broker and transmit data. In addition, there is a problem that the broker for data transmission is fixed in the initial deployment stage because it does not support multicast by default [5].

2-2. MQTT-SN (MQTT-Sensor Network)

MQTT-SN (MQTT for Sensor Network) is a protocol for sensor nodes that operates in a process similar to MQTT [6][7], which supports multicast and can be applied to low-computing devices, making it suitable for wireless sensor network-based smart factory manufacturing management systems Wireless mesh networks (WMN) are also becoming more widely introduced because wireless mesh networks can cover a wide range using relatively low-power wireless when delivering messages from node to node and maintain high reliability by avoiding interference using alternative paths and channels. One of the most important development issues in wireless mesh networks is multicast routing. Multicast is a key technology that provides data communication

in an efficient manner. A type of communication between many nodes that efficiently propagate data from a single source node to a set of destinations, the mechanism is that a message is transmitted only once from one link in the network to another, then replicated at a branch point and propagated to another destination. Most of the existing studies on WMNs have focused on single-radio and single-channel measurements, but recently, multicast routing has been used to increase the efficiency of WMN [8].

2-3. Existing Research on Distributed MQTT Gateway

The MQTT protocol has a structure where "sensor nodes generating data" and "brokers" are connected in a peer-to-peer manner, and it does not support multicasting. Therefore, it is not suitable for smart factory manufacturing environments that require multiple sensors and gateways. To compensate for this problem, research on a distributed MQTT method that supports multicasting was conducted, but existing studies proposed a non-standardized MQTT protocol, which lacked future scalability[9], it was confirmed that it was possible to change the gateway to a master-slave broker structure that supports SDN-based multicast based on the standard MQTT protocol, but [10][11] There was a limit to analyzing the MQTT broker function and changing the gateway when connecting the sensor and the broker(gateway). The optimization method based on workload capacity [computing resources] and average traffic of nodes and gateways does not match the characteristics of smart factory manufacturing environment data that does not generate large amounts of traffic and does not reflect MQTT's messaging priorities for data delivery in the event of abnormal alarms [12]. An SDN multicasting support gateway based on messaging priorities and QoS was proposed [13], but this study focused on signal processing speed between brokers and does not reflect data loss variables due to increased or decreased signal strength by distance in a wireless environment. In this paper, I analyzed the problems of these existing studies and proposed and designed a method that considers both the number of wireless signals and access APs between sensors and gateways and MQTT messaging priorities as an alternative.

3. Proposed System Architecture

The overall architecture of the system constructed in this study consists of wireless modular sensor nodes that sense and process information, gateways that act as slave brokers to relay data between sensor nodes and the master broker, and a cloud server platform that manages topic subscriptions between sensor nodes and gateways and performs the functions of a master broker. Through this architecture, data from each sensor is transmitted to the server via the optimal gateway, allowing for monitoring, big data collection, and storage in a cloud No-SQL database for anomaly detection and future use in big data analysis. Additionally, by utilizing IoT standard protocols and platforms, seamless integration with other applications and heterogeneous devices is facilitated. The Proposed System Structure is depicted in Figure 1.

3.1 Data flow between sensor nodes and brokers

The sensor nodes that generate data are connected to gateways via WiFi wireless LAN. To collect data, each gateway is equipped with an MQTT slave broker. Regardless of which broker located in each gateway the sensor node is connected to, it should be able to publish/subscribe to all topics. To achieve this, coordination between the master broker and slave brokers is required. For this purpose, subscription messages from subscribers generated by each slave broker are flooded (Subscription Flooding: SF) to all other slave brokers,

synchronizing the subscription tables among the gateways. Each slave MQTT broker must maintain forwarding tables for all topics used by MQTT clients in the network.

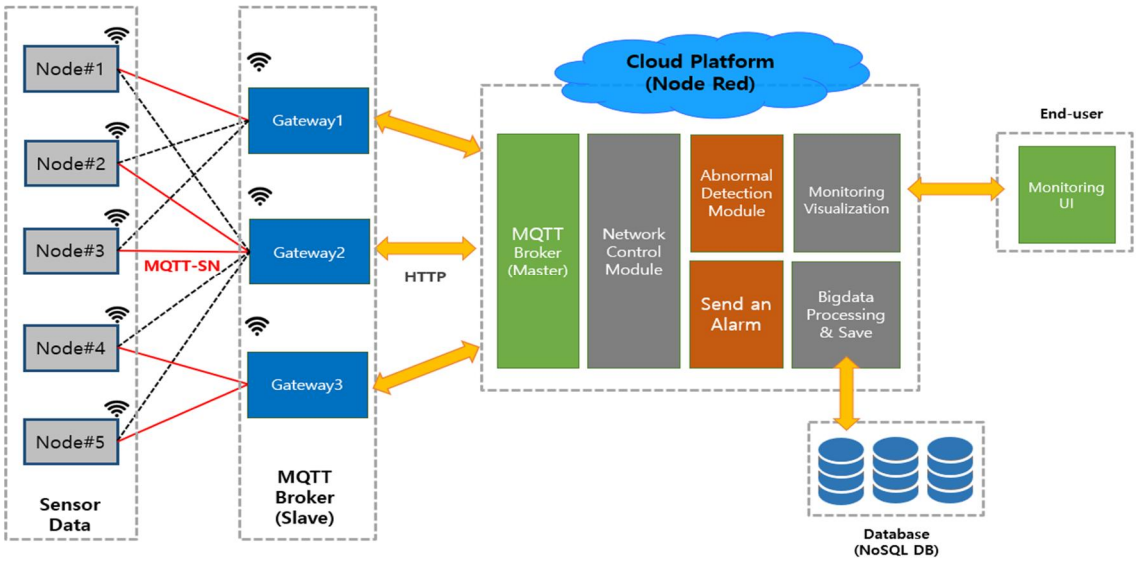

Figure 1. Proposed System Structure

Additionally, to facilitate integration with the master broker located in the cloud, the slave brokers forward all subscription requests for topics they don't possess to the central broker. The central broker receives and manages subscription messages from all gateways connected to it. The transition of a sensor node and a slave broker to another slave broker located in an adjacent gateway is necessary based on factors such as wireless signal strength, available terminal quantity, and priority message processing according to Quality of Service (QoS). The MQTT protocol establishes the TCP connection before exchanging MQTT protocol messages between the broker and the client. Subsequently, packet exchange for connection establishment, data publishing/subscribing occurs through the MQTT protocol. When a client wants to terminate the connection, it notifies the server using the MQTT protocol. Upon receiving the notification, the server sends a FIN packet to terminate the TCP connection instead of sending a separate MQTT protocol message. The MQTT protocol does not provide a separate protocol to induce reconnection of an existing client to a different broker, and even if the forwarding rules for the flow are changed, the TCP-based connection between the broker and the client is not reestablished. Therefore, to support dynamic creation, reconnection, and termination of slave brokers, a hierarchical structure is established based on the master broker. Each slave broker delegates the processing of a specific set of topics managed by the master broker [e.g., node#1/gateway1/temp] and receives the subscription table for the corresponding topics included in the message header. The network control module located in the cloud informs all slave brokers in the connected gateways about the delegation of specific topics. Based on a predefined priority comparison algorithm, the network control module determines the route that each topic should follow. Consequently, if a topic published by sensor node #1 needs to be routed through gateway 2 instead of gateway 1, subsequent publishing/subscribing for that topic is redirected to the distributed broker located in gateway 2 (e.g., node#1/gateway2/temp). This proposed distributed broker structure allows for the management of a specific set of topics by distributed brokers, enabling optimization of paths between sensor nodes and gateways. The data transmission process among the sensor nodes, slave brokers, and master broker. The data transfer sequence diagram is depicted in Figure 2.

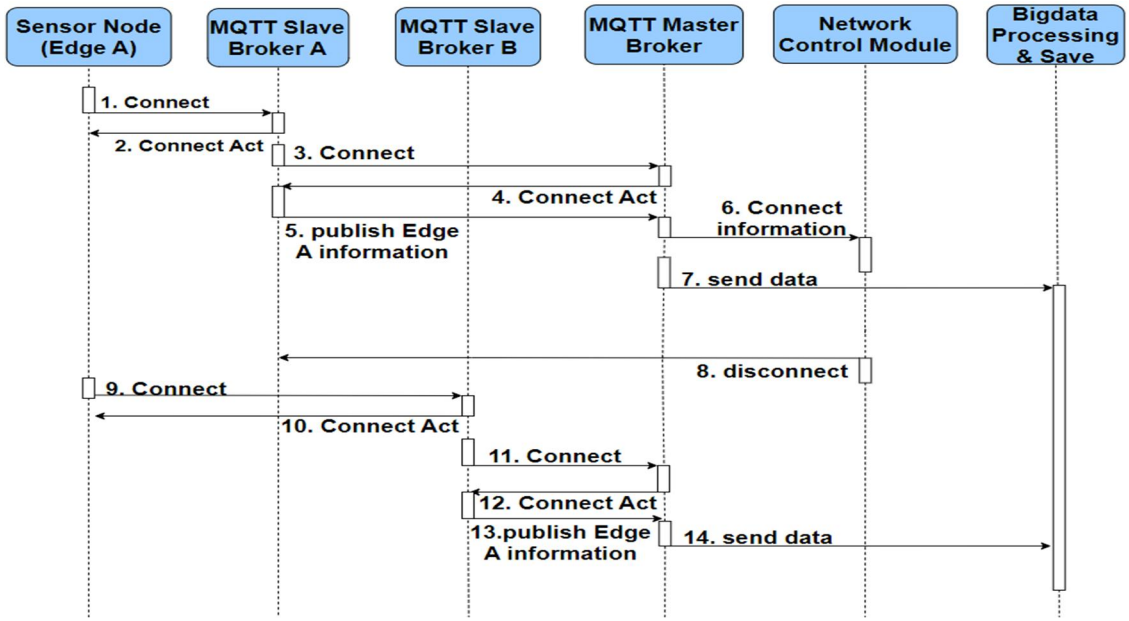

Figure 2. Data transfer sequence diagram

4. Implementing the Proposed System

The wireless sensor nodes consist of sensors for temperature, fire, and gas signals, as well as NodeMCU for signal processing. NodeMCU is programmed in an Arduino-based development environment using ESP8266EX. The implementation of the sensor nodes was done in C++ using Visual Studio Code version 1.28.2 with the PlatformIO IDE plugin, configuring data collection and the MQTT protocol. The configuration of the gateway and cloud server connections, as well as the authentication process, was done using JSON format. The MQTT broker used for message transmission is mosquitto version 1.5.3.

The gateways where the slave brokers are installed were set up using an Intel i5 PC with Windows 10 and the virtual machine software VMware. "Raspbian stretch with Desktop" was installed and configured on the virtual machine. Publishers were configured using the Python MQTT library from Paho, while subscribers were configured using the mosquitto_sub program.The network control module responsible for managing the subscriptions of each slave broker and the master broker was implemented using the Python MQTT library from Paho. IBM Cloud Bluemix was used as the cloud platform, and the middleware program used was IBM Node-Red. IBM Watson IoT Platform was used for integration between the sensor nodes and the middleware platform. MongoDB, a cloud-based database, was used, and functionalities such as checking and querying database tables/data, and exporting data were performed using MongoShell and Robo3t. The system functionalities of the middleware were programmed using Node.js and JavaScript, while HTML was used for the user event message page in the monitoring UI. The implementation of the proposed system image is shown in Figure 3.

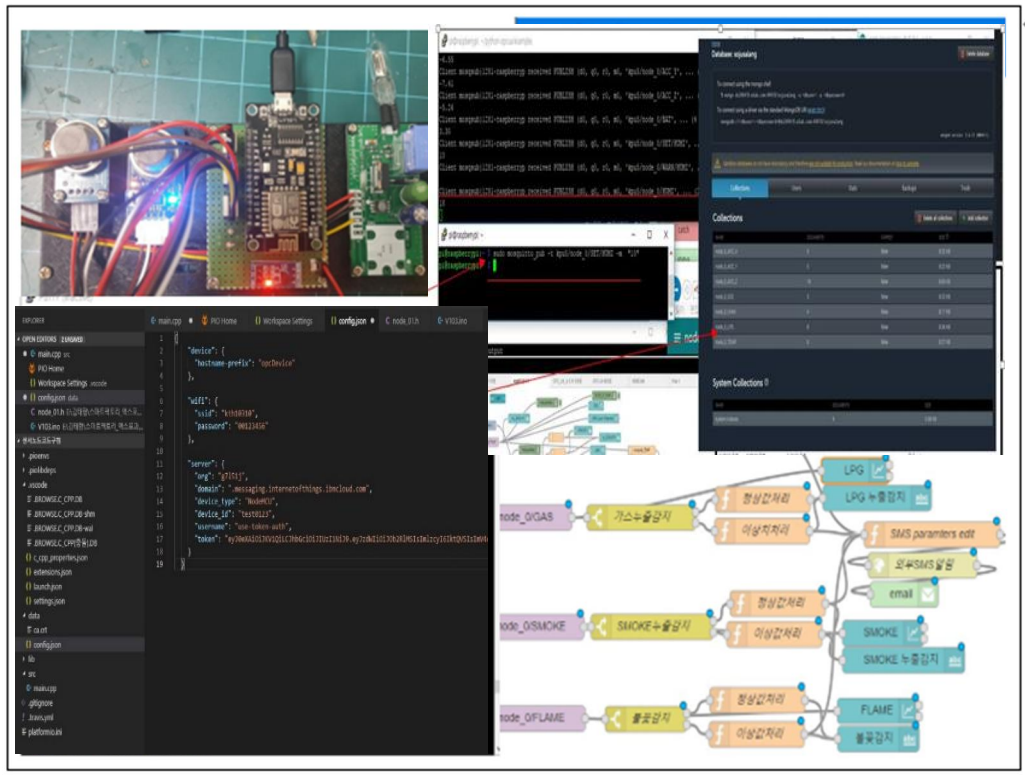

Figure 3. Implementing the Proposed System Image

5. Simulation and Validation

The system implementation involved creating a simulation environment using one sensor node and two gateways. One of the gateways, reflecting the fluctuation of wireless signal strength in a real smart factory environment, periodically changed its position to vary the Wi-Fi RSSI value. Through this, the process of validating the data collected by the sensors finding the optimal gateway route and being transmitted to the server via the master broker was verified. The data collected from the sensor node was designed to reflect the characteristics of low-power IoT implementation. Each subscriber had 5 topics, with a topic publishing rate of once every 5 minutes, and the sub-topic length was set to 40 bytes. To validate the normal operation of the implemented system, candles were lit, and temperature and fire signal data were collected from the sensor node. Data from 12 hours before and after the time when fire was detected were extracted from the cloud server. As shown in Figure 4, through event logs from the gateways and server, it was confirmed that the data was successfully transmitted to the server by changing the gateway when there was a delay in wireless signal. The fire and temperature data were extracted as CSV files from the server, and time series analysis was performed using the statistical program R for monitoring. Additionally, Wireshark was used to collect packets exchanged between the sensor nodes, gateways, and server at 5-minute intervals for the analysis of end-to-end delay and message loss. The measurement of end-to-end delay and message loss involved counting the retransmission request packets of the 5-minute data packets. As observed in Figure 5 and Figure 6, communication between the sensor and gateway experienced delays and losses due to signal attenuation. However, it can be confirmed that sending data to the server through route optimization resulted in no delay or loss.

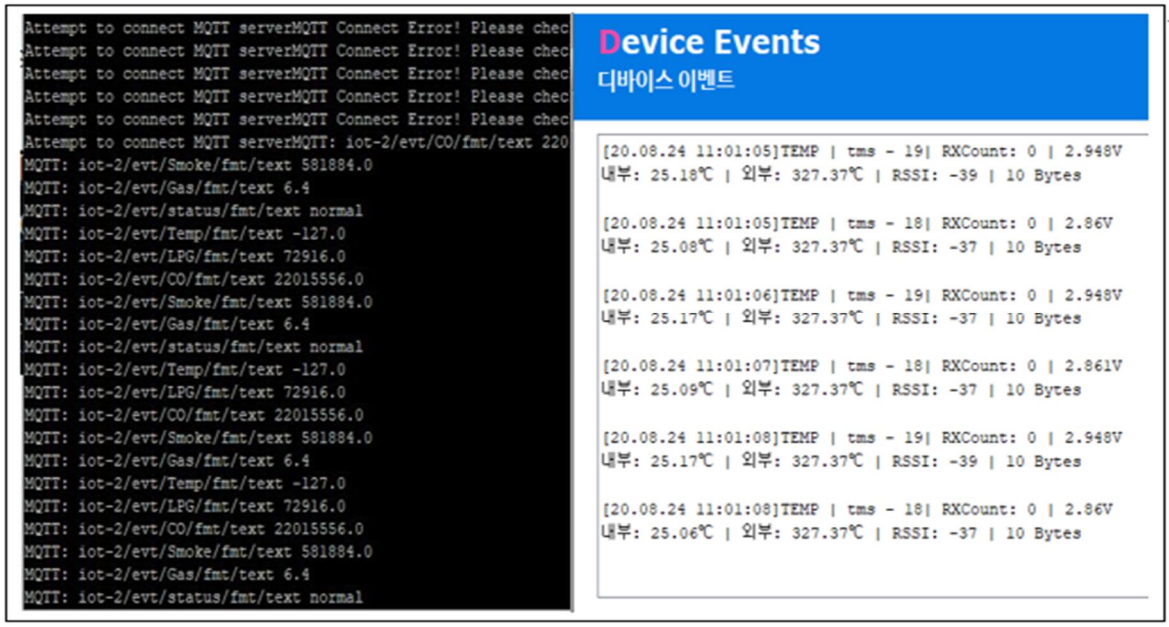

Figure 4. The event log screens for gateway and server data reception

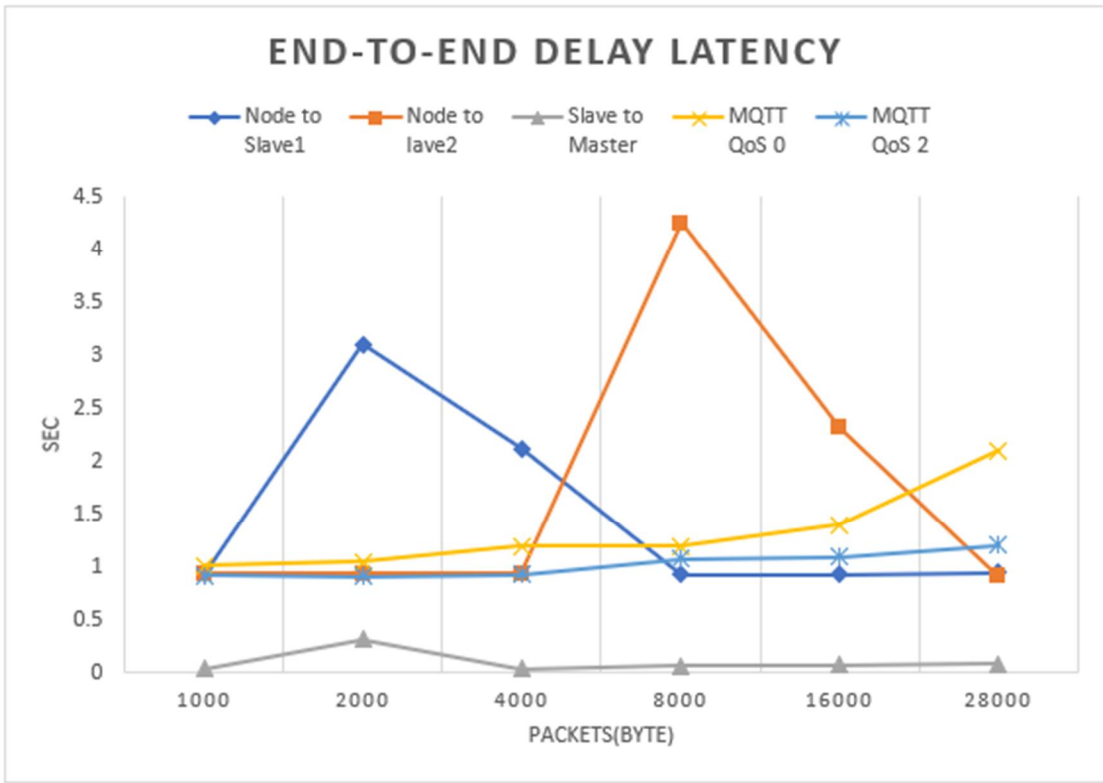

Figure 5. Analysis Results of End-to-End Delay Latency

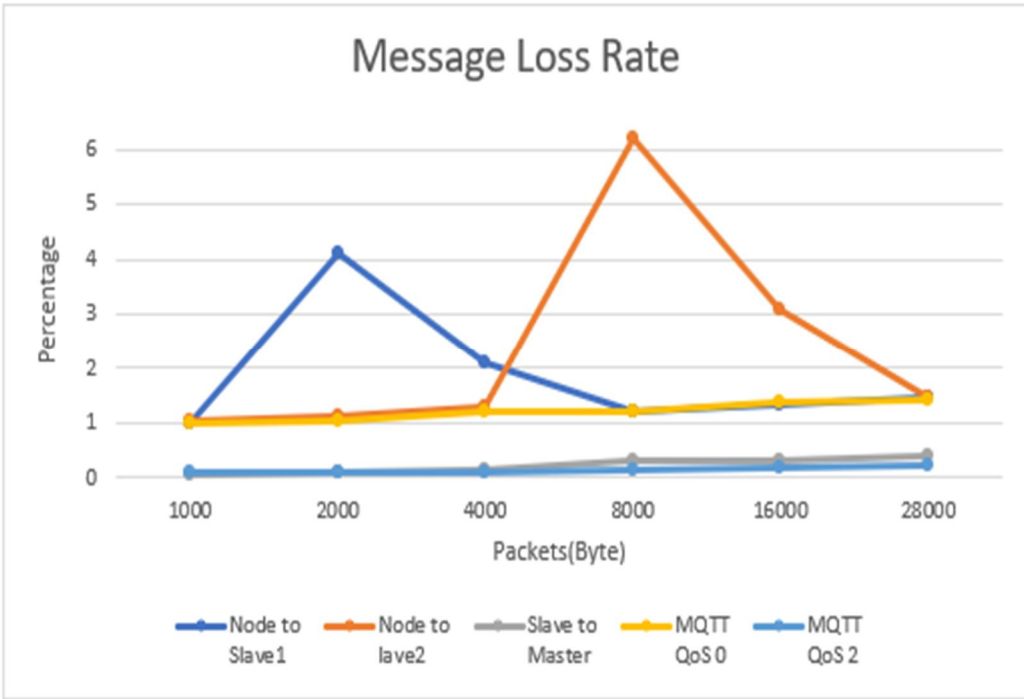

Figure 6. Analysis Results of Network Packet Loss

6. Conclusion and expected effects

In this study, I designed a distributed MQTT IoT smart sensor and gateway structure that supports wireless multicasting for smooth sensor data collection. Through this, it was possible to derive significant results in the service latency and data loss rate of packets even in a wireless environment, unlike the MQTT QoS-based system. Through this study, if communication quality problems occur due to security, signal interference, and echo, which are disadvantages of wireless networks, communication failures can be prevented by immediately changing to alternative routes. It also improves heterogeneous compatibility through compliance with IoT standards and easy connectivity with existing systems. Learn to categorize the causes of failure by data type to prevent the loss of manpower due to misinformation. Reduction, activation of misinformation through multidimensional monitoring; advanced prevention through fault diagnosis. Cloud platforms have the advantage of reducing system development costs and time and increasing flexibility and scalability.

Stable sensing data collection is possible in smart factory manufacturing environments because it prevents data loss and delay caused by abnormal data generation. It enables low-cost deployment based on wireless networks and flexibly responds to changes in the entire production process. If communication quality problems occur due to security, signal interference, and echo, which are disadvantages of radio, you can immediately change to an alternative route to prevent communication failures. Compliance with IoT standards, improved heterogeneous compatibility through easy connectivity with existing systems. Learn to categorize the causes of failure by data type to prevent the loss of manpower due to misinformation. Reduction, activation of misinformation through multidimensional monitoring; advanced prevention through fault diagnosis. Cloud platforms are expected to reduce system development costs and time, and increase flexibility and scalability.

References

[1] H.M. Kwon, V. Kumaran, and S. Gupta, "A New Broadcast Scheduling Scheme," The Journal of the Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 11, No. 2, pp. 63-72, June 2011.

[2] D. Bradley, D. Russell, I. Ferguson, J. Isaacs, A.MacLeod, and R. White, "The internet of things–the future or the end of mechatronics,"Mechatronics", Vol.27, pp.57–74, 2015.
DOI: https://doi.org/10.1016/j.mechatronics.2015.02.005

[3] Y.-J. Yoon, T.-H. Kim, J.-H. Lee, and Y.-G. Kim, "Big Data Refining System for Environmental Sensor of Continuous Manufacturing Process using IIoT Middleware Platform," The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 18, No. 4, pp. 219–226, Aug. 2018.
DOI: https://doi.org/10.7236/JIIBC.2018.18.4.219

[4] Y.-S. Jeong, "Linking Algorithm between IoT devices for smart factory environment of SMEs," Journal of Convergence for Information Technology, Vol. 8, No. 2, pp. 233–238, Apr. 2018.
DOI: https://doi.org/10.22156/CS4SMB.2018.8.2.233

[5] S.-C. Oh and Y.-G. Kim, "A Study on MQTT based on Priority Topic for IIoT," The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 19, No. 5, pp. 63–71, Oct. 2019.
DOI: https://doi.org/10.7236/JIIBC.2019.19.5.63

[6] Hokyun Park. "Development of IoT Gateway System for Indoor Air Quality Monitoring Using MQTT and NodeMCU" The Society of Convergence Knowledge Transactions Vol.8, No.4, pp.89-97, 2020
DOI: https://doi.org/10.22716/sckt.2020.8.4.039

[7] Schütz B.; Bauer, J.; Aschenbruck, N. "Improving Energy Efficiency of MQTT-SN in Lossy Environments Using Seed-Based Network Coding" In Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks [LCN], pp. 286–293, Oct.2017.
DOI: https://doi.org/10.1109/LCN.2017.87

[8] Amaran, M.H.; Noh, N.A.M.; Rohmad, M.S.; Hashim, H. "A comparison of lightweight communication protocols in robotic applications". Procedia Computer Science, 76, pp.400–405. Dec. 2015.
DOI: http://dx.doi.org/10.1016/j.procs.2015.12.318

[9] Moonsik Kang,"High Performance QoS Multicast Routing Scheme for Real-Time Mobile Multimedia Applications in Wireless Mesh Networks." The Society of Convergence Knowledge Transactions Vol.8, No.4, pp.89-97, 2020
DOI: https://doi.org/10.22716/sckt.2020.8.4.039

[10] YeRin Im, Mingyu Lim. "QoS Level 3: A Synchronous Communication Mechanism in MQTT Protocol for IoT" The Transactions of the Korean Institute of Electrical Engineers, Vol. 70, No. 6, pp. 893-904, 2021
DOI: https://doi.org/10.5370/KIEE.2021.70.6.893

[11] Y. Xu, V. Mahendran and S. Radhakrishnan, "Towards SDN-based Fog Computing: MQTT Broker Virtualization for Effective and Reliable Delivery," in Proc. IEEE COMSNETS, pp. 1-6, Jan. 2016.
DOI: https://doi.org/10.1109/COMSNETS.2016.7439974

[12] T. Rausch, S. Nastic and S. Dustdar, "EMMA: Distributed QoS-Aware MQTT Middleware for Edge Computing Applications," in Proc. IEEE IC2E, pp. 191-197, May 2018.
DOI: https://doi.org/10.1109/IC2E.2018.00043

[13] Geonwoo Kim, Jiwoo Park, Kwangsue Chung. "MQTT-based Gateway System for Auto-configuration of IoT Devices and Services" The Journal of KIISE, Vol. 46, No. 4 pp. 385-390. 2019.
DOI: https://doi.org/10.5626/JOK.2019.46.4.385

[14] Hyunseong An, Woojin Sa, Seungku Kim. "Design and Implementation of RPL-based Distributed MQTT Broker Architecture" The Journal of Korea Multimedia Society, Vol. 21, No. 9, pp. 1090-1098, Sep. 2018
DOI: https://doi.org/10.9717/kmms.2018.21.9.