Article

Dependable Fire Detection System with Multifunctional Artificial Intelligence Framework

Jun Hong Park, Seunggi Lee, Seongjin Yun, Hanjin Kim and Won-Tae Kim *

Computer Science and Engineering, Koreatech University, Cheonan-si 31253, Korea;
astrada1@koreatech.ac.kr (J.H.P.); sss8412@koreatech.ac.kr (S.L.); hiysr0308@koreatech.ac.kr (S.Y.); gks359@koreatech.ac.kr (H.K.)
* Correspondence: wtkim@koreatech.ac.kr; Tel.: +82-41-560-1485

Received: 31 January 2019; Accepted: 27 April 2019; Published: 30 April 2019

Abstract: A fire detection system requires accurate and fast mechanisms to make the right decision in a fire situation. Since most commercial fire detection systems use a simple sensor, their fire recognition accuracy is deficient because of the limitations of the detection capability of the sensor. Existing proposals, which use rule-based algorithms or image-based machine learning can hardly adapt to the changes in the environment because of their static features. Since the legacy fire detection systems and network services do not guarantee data transfer latency, the required need for promptness is unmet. In this paper, we propose a new fire detection system with a multifunctional artificial intelligence framework and a data transfer delay minimization mechanism for the safety of smart cities. The framework includes a set of multiple machine learning algorithms and an adaptive fuzzy algorithm. In addition, Direct-MQTT based on SDN is introduced to solve the traffic concentration problems of the traditional MQTT. We verify the performance of the proposed system in terms of accuracy and delay time and found a fire detection accuracy of over 95%. The end-to-end delay, which comprises the transfer and decision delays, is reduced by an average of 72%.

Keywords: fire detection; dependability; IoT; artificial intelligence; distributed MQTT; SDN

1. Introduction

Dependable fire detection systems with high accuracy and promptness are essential for the safety of smart city services. Since there are so many false alarms in legacy fire detection systems, they occasionally make system operators take risky actions such as turning off low-precision fire detection systems [1]. Improving fire detection systems could prevent many accidents due to fires. During 2009–2012, excluding malicious calls 48% of all fire alarms were false alarms [2]. Of all 6,684,500 fire accidents in United States, 4,879,685 cases occurred where fire detection systems were installed [3]. Unfortunately, 20% of the fire detection systems in the USA do not correctly work [1].

Commercial fire detection systems generally have a simple sensor with low accuracy and are sensitive to sensor failure and malfunction which makes it hard to detect fires [4]. In addition, even if the sensor is operating normally, faults may occur in the system because of the limitations of the detection capability of the sensor. A smoke sensor may send false alarms because of the incorrect recognition of heavy dust as smoke [5]. Fire detection systems with image sensors have difficulty detecting fires in the blind spots of cameras. Therefore, it is necessary to combine heterogeneous sensors for improved fire detection performance [6].

The existing fire detection systems use rule-based algorithms with static parameters that make it difficult to accurately detect fire in dynamic situations [7]. Fire detection systems require suitable thresholds with specific parameters depending on the system's installation environment. In addition, the data sensed in the same place may change with time. The dynamic threshold is essential for applying rule-based algorithms, but it is impossible for normal users to calculate these thresholds and apply them to their systems. In particular, when a fire occurs, the temperature data show a rapid increase in a short time [8,9]. Rule-based algorithms with static thresholds do not identify a fire when the sensing data do not exceed the threshold value.

In general, artificial intelligence (AI) is suitable for identifying situations that are difficult to classify using the features extracted from the sensing data [10]. For example, AI can determine the special characteristics set or special features of various fire situations using the collected data and apply them to make decisions in actual fire events. Since machine learning (ML) is categorized as a subfield of AI, ML inherits the characteristics of AI, providing machines with the ability to make decisions by learning from data and optimizing the functional weights or parameters of algorithms instead of relying on the user to specify rules according to the situation [11]. Each ML algorithm may utilize specific types of training data depending on the purposes. For example, the Convolutional Neural Network (CNN) requires image data to recognize objects and the Recurrent Neural Network (RNN) is suitable for processing sequential data, including voice and text [12,13]. However, it is difficult for ML to adapt to dynamic changes because the values of the ML variables are fixed after training. Flexibility must be supported to compensate for the problems with ML. Since ML algorithms create different decision probabilities based on the same target, the results can incorporate the mutual-complementary factors of an ensemble method to obtain a better performance than a single algorithm [14]. Our ensemble method integrates various AI algorithms into a multifunctional AI framework that combines the advantages of each ML and has enhanced flexibility.

Combustible materials and a delayed reaction time are the main causes of fire spread, accounting for 40% and 28% of all fires, respectively [15]. Prompt fire detection effectively decreases the fire spread that follows primary causes. Fire detection systems are important for early fire detection because combustible materials create fires that spread explosively. There are two types of delayed reaction time: the time to detect a fire in a location and the time to move from a fire station to this location. If the arrival delay exceeds 5 min, property damage is doubled compared to cases with delays below 5 min [16]. Since the delay time to move to the fire site is affected by external factors, such as traffic congestion, we aim to reduce reaction delay by minimizing fire detection delays. Fire detection delays are due to two main causes: decision delays and transfer delays. Decision delays depend on the hardware performance and the complexity of the fire decision algorithms installed on the fire detection systems. Since transfer delays are subject to network congestion or specific bottleneck nodes on the forwarding paths, the latency time should be managed in the serious cases. While the transfer delays absolutely affect the promptness of the fire detection system, most fire detection systems do not consider the minimization of the transfer delays.

Recent studies on advanced fire detection systems have been made based on IoT middlewares [17–19], such as MQTT, which is the common IoT middleware used by the most IoT platforms including oneM2M [20,21]. IoT middlewares like MQTT that use message queue mechanisms do not support real-time quality of service (QoS) policies because they are designed as a centralized architecture which has difficulties in supporting real-time services [22]. In spite of the problems, MQTT has been widely used in smart city services based on IoT. Therefore, studies complementing the problems are required to support the time critical services [23,24].

In this paper, we propose a dependable fire detection system using a multifunctional AI framework which interworks a novel Direct-MQTT. The main features proposed in this paper are as follows:

- Improved speed of fire detection by minimizing the delay of data transmission by deleting the queuing delay that occurs in the central broker when using the MQTT protocol.
- Combined and analyzed complex fire sensor data using multi-functional AI framework to improve fire detection accuracy.

This paper is divided as follows: in Section 2, we summarize the legacy fire detection systems, AI, and IoT middlewares. Section 3 describes the proposed fire detection system architecture and explains the multifunctional AI framework and Direct-MQTT. In Section 4, we demonstrate the performance of the proposed system in terms of the fire detection accuracy and the minimization of data transfer delay. Finally, the conclusion and future works regarding the fire detection system are provided.

2. Related Works

2.1. The Legacy Fire Detection Systems

Many studies have been conducted to combine fire detection systems with ML algorithms to analyze the images and sensor data and obtain accurate results. Shen proposed a study of flame detection using the YOLO framework [25]. YOLO is a framework that is used in various fields, including flame detection, because R-CNN-based multi-object recognition is possible. In his study, he trained models with 76% fire detection capability using 196 fire images. However, the number of fire images used for training may not be sufficient to effectively train the model. Kaabi proposed a study to detect fire smoke using a Deep Belief Network (DBN) [26]. A total of 482 data sets were used for the training and a model with 95% accuracy was produced. Since his method recognizes smoke only, it is impossible to detect fire with a certain flame recognition. Hu proposed a Neural Network of Deep Convolutional Long-Recurrent Networks (DCLRN) and conducted real-time fire detection experiments [27]. It also used a method of detecting flame movement by combining the optical flow method. He used about 10,000 images obtained from 70 fire videos in total and made a model with 93.3% accuracy. However, there are cases where faults occur from mistaking flames and lights. If he used it with other sensors, he would be able to correct those errors. Saputra proposed a fire detection system that directly implemented a fuzzy algorithm and detected fire using multiple sensors [28]. The fuzzy algorithm integrates multiple sensor data and deduces the results. However, the fuzzy algorithm has a disadvantage of using a static threshold called a fuzzy factor. S-FDS, as proposed by Jang, is a smart fire detection system that combines CNN and fuzzy logic and collects sensor data and image data using heterogeneous sensors and CCTV [29]. S-FDS preprocesses the image data using a CNN algorithm to detect fires. However, the CNN algorithm that analyzed these images could not quickly and accurately detect fires in blind spots, such as underneath a table. In particular, the CNN algorithm is difficult to use in a place where a camera cannot be used because of privacy problems, such as restrooms. In order to solve the blind spot problem of the CCTV, the fuzzy logic derives the fire recognition probability from the analyzed image data and the heterogeneous sensor data. However, S-FDS is less flexible because it is based on static, rule-based algorithms.

2.2. Artificial Intelligence

Recently, various algorithms have used statistical techniques for data analysis. Algorithms can have different characteristics depending on the calculation formula, and the data analysis result may vary depending on which algorithm is used. There is an ML algorithm for analyzing data. ML algorithms can be divided into shallow learning, such as Support Vector Machine (SVM), decision trees, and K-Means, as well as deep learning, which uses neural network layers such as a Deep Neural Network (DNN) and CNN [30–34]. The use of a deep learning model for classification is more flexible than a shallow learning model in the real environment. In general, DNN is used to determine the current value through quantified data or analyze the changing situation through continuous data. However, immediate detection is difficult because data should be collected over a period of time so that DNN may measure the changing situation. In addition, CNN is used to analyze the situation by detecting features in image data. When measuring flames in an image using video data, the image feature capture may be delayed because the flame is small if the fire has only just begun [35]. Fuzzy algorithms express a closeness to each situation that is not clearly divided using the membership function. The membership function of the fuzzy algorithm can change its range when the environment changes, but the general fuzzy algorithm ignores these changes. To solve this problem, there is an adaptive fuzzy algorithm that can update the membership function [36]. However, the adaptive fuzzy algorithm does not filter out the exception data due to sensor errors, so it does not obtain accurate results.

2.3. IoT Middleware

Many types of IoT middleware have been studied for their ability to connect various devices and transmit data in large-scale IoT systems. MQTT, CoAP, HTTP, XMPP are representative IoT middleware protocols [37–40]. MQTT is a standardized protocol in OASIS and can maintain the connectivity of multiple devices using a broker. Broker-based protocols may have data concentration problems and single point of failure issues because many devices connect and transmit data to the broker. MQTT limits the number of devices that are connected to a broker to 1024 to prevent queue delays because of the concentration of multiple packets. MQTT provides low-level QoS and does not consider data transfer delays, which make it difficult to use in complex network situations. DM-MQTT, as proposed by Park, considers the multiple subscribers through multicasting [41]. However, this mechanism has not solved the traffic concentration due to the rendezvous point, and it is not effective in situations with few subscribers. Santamaria [42] proposed a way to reduce the serving time of the system by using the data filter applied to the server and the MQTT protocol to process a lot of data of the ioT devices used in the e-Health system. However, this proposal cannot reduce the data transfer delay from the publishing device to the server because there is no improvement of the MQTT protocol. Santamaria proposed an anti-terrorism surveillance system [43] that can respond quickly to suspicious objects when they are found in three layers. Distributed surveillance systems are able to respond more quickly to cloud computing using edge computing technology. However, a method of applying only edge computing without considering protocol improvement cannot solve the problem of traffic overload and data concentration because of a large amount of data transmitted from multiple devices. CoAP is a standardized protocol in the IETF CoRE working group and transmits data in a RESTful method. Since CoAP is essentially designed with a peer-to-peer protocol, it has difficulty in supporting large IoT systems that require multiple edge nodes. The well-known HTTP is an application-level protocol for hypermedia information systems. HTTP transmits data in a RESTful method. Since HTTP does not consider a device with low computing power such as a sensor node, there is a problem like a big header overhead in the process of transmitting even simple sensor data. XMPP is a protocol for instant messaging developed by the Jabber open source community in 1999. XMPP is designed with scalability in mind for real-time systems and various network environments. XMPP, which is an XML message-based communication, has a distributed system such as e-mail, but has the weakness of lack of QoS and end-to-end encryption. Functional comparison of communication middleware is shown in Table 1. In addition, there are many IoT middleware such as AMQP, a lightweight peer-to-peer protocol, and Hy-LP, which fuses various protocols. In this paper, we use MQTT protocol considering the advantages of standards based protocol.

Table 1. Functional Comparisons of Communication Middleware.

	MQTT	**CoAP**	**HTTP**	**XMPP**
Basis protocol	TCP	UDP	TCP	TCP
Data transmission	Pub/Sub	RESTful	RESTful	Pub/Sub
Quality of Service	Three policies	Not supported	-	Not supported
Support scale	Large scale	Middle scale	Large scale	Large scale
Communication architecture	Centralized	Centralized	Centralized	Distributed

3. Proposed Fire Detection System Architecture

The structure of the proposed system is shown in Figure 1. The architecture consists of IoT gateways, a fire detection server, and a software-defined networking (SDN) controller.

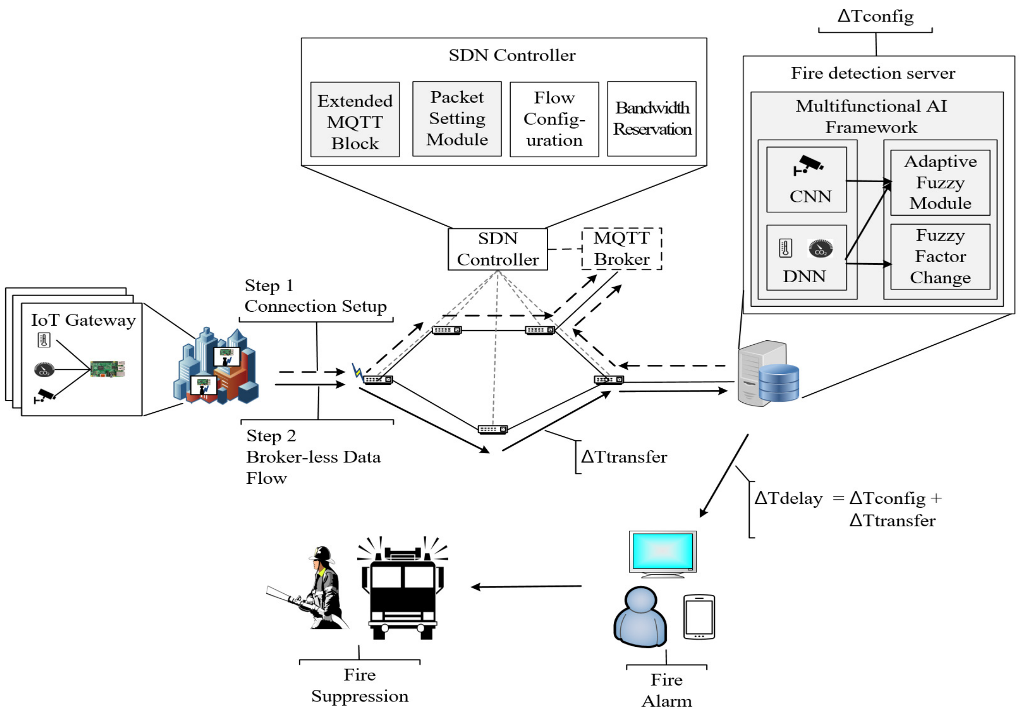


Figure 1. Dependable fire detection system with a multifunctional AI framework.

3.1. Architecture Overview

The fire detection system may be constructed as a large-scale system that collects data from multiple IoT gateways. The IoT gateways send the collected data from heterogeneous sensors, which include temperature, humidity, gas sensors, and cameras to a multifunctional AI framework that identifies the fire. The multifunctional AI framework uses multiple machine learning to analyze the heterogeneous data to calculate the fire probability for each sensor. The multiple ML algorithms include a CNN analyzing the fire image data from the camera, and a DNN processing the time series data from heterogeneous sensors. The fire probabilities for each sensor measured through MLs are combined into a single fire probability by using an adaptive fuzzy algorithm, which modifies the dynamic fuzzy factor according to the multiple ML results to infer the fire probability suitable for the environment where the system is installed. The fire detection system transmits data using the MQTT protocol, which is a broker-based middleware that can create a bottleneck problem and cause data transfer delays. The SDN controller minimizes this data transfer delay by solving the bottleneck problem with the data transmission path distribution mechanism. The SDN controller makes the direct paths between the fire detection server and the IoT gateways.

3.2. Multifunctional AI Framework

The multifunctional AI framework is designed to provide context adaptability, high accuracy of fire detection, and rapid decision-making by combining various AI technologies. It can accurately analyze a variety of data types, integrate the analyzed data as required by the system, flexibly adapt to changes in time and space. As mentioned above, the multifunctional AI framework consists of a CNN algorithm, a DNN algorithm, and an adaptive fuzzy algorithm. The framework has three blocks: an IoT data collection block, a context preprocessing block, and a context decision block. The overall structure of the multifunctional AI framework is shown in Figure 2.

The IoT data collection block subscribes to the MQTT session and continuously gathers heterogeneous data from the IoT gateway. In addition, it classifies and feeds the data to the context preprocessing block along with each data type. All MQTT data is stored in the database and used as input data for the sequential data analysis modules through the data preprocessing module.

The context preprocessing block has multiple ML algorithms that can be replaced with others depending on the input data types. It combines the image data analysis module with CNN and the sequential data analysis module with DNN. The multiple ML uses the ensemble method to more effectively analyze heterogeneous sensor data than a single ML would.

The context decision block uses an adaptive fuzzy module to fuse the outputs from the context preprocessing block in order to calculate the exact probability of the fire. In addition, it can simultaneously reconfigure the fuzzy factors depending on the context. The dynamic fuzzy factors automatically change to deduce appropriate results for the target environment. We will perform experiments to compare the proposed adaptive fuzzy algorithm and other ML algorithms that may be used in the context decision block instead of the adaptive fuzzy module.

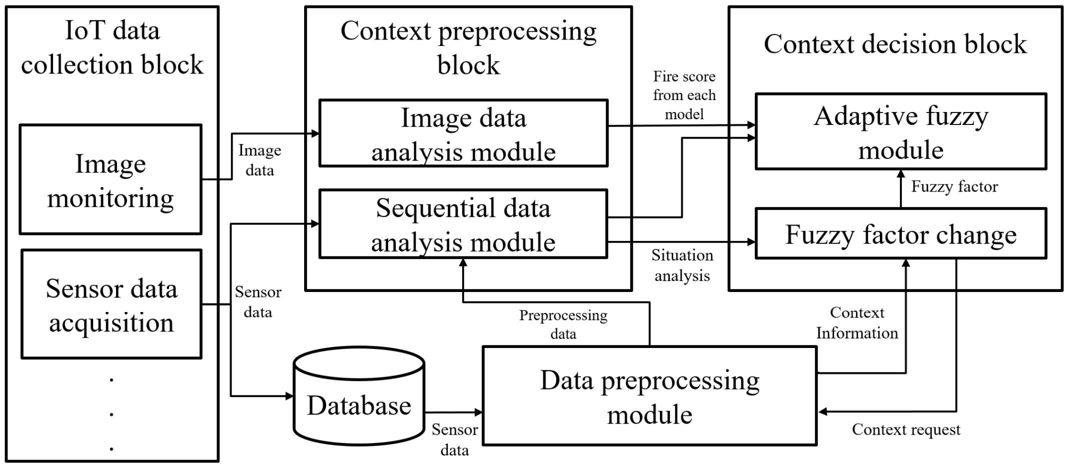


Figure 2. The proposed multifunctional AI framework.

3.2.1. Image Data Analysis Module

The image recognition algorithm using CNN has proved its effectiveness and is used in various industrial fields, including fire detection services [44,45]. In this study, we also apply CNN to a multifunctional AI fire detection system (MAI-FDS) in order to enhance the performance of fire detection in terms of visual intelligence. The CNN algorithm is constructed as shown in Figure 3. It consists of five convolution layers and one fully-connected layer. The image is resized to a size of 256 $\times$ 256. Each convolution layer has a 0-padding, 5 $\times$ 5 size feature map and 2 $\times$ 2 max pooling. The depth of the feature map in the first layer is 8, which is doubled every time it passes through the hierarchy. After passing through five layers, 256 images of 8 $\times$ 8 are generated. Finally, the image is classified through a fully-connected layer of 1024 neurons.

The CNN algorithm learns two classes of fire situations and warning situations. A fire situation is determined when an actual fire is detected in the image, and a warning situation is obtained when a fire risk is recognized in the image. For example, a lighter flame does not mean an actual fire but just the probability of a fire risk. There is a score for each class, and the result is the class with the highest score. If the scores of both classes are all lower, the algorithm judges the situation as normal. Table 2 shows the training data, training frequency, and accuracy of the CNN model. The score of CNN's fire situation class is entered as a factor of the fuzzy algorithm.

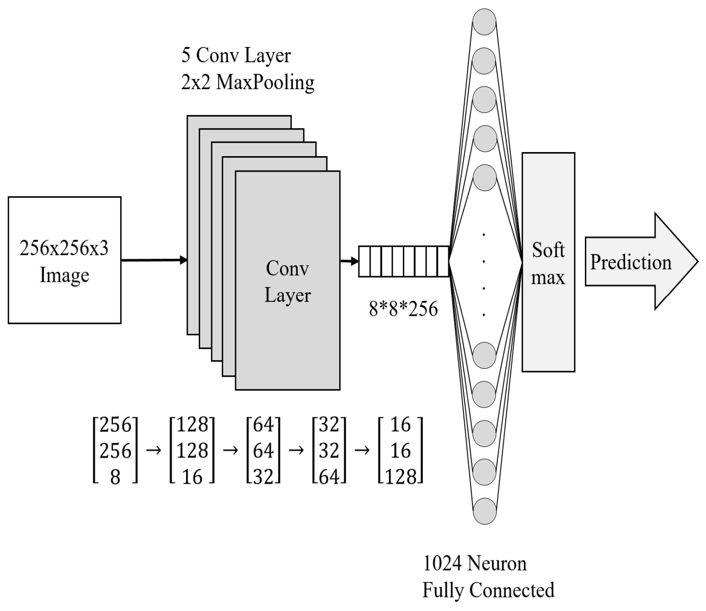


Figure 3. The proposed Convolutional Neural Network architecture to analyze the image data.

Table 2. CNN Model training set and accuracy.

	Training Data	**Testing Data**
Fire	8500	300
Matches	1119	300
Accuracy	90.3%	91%
Training Steps	14000	-

3.2.2. Sequential Data Analysis Module

To compensate for the blind-spot problem of image sensors, we use a heterogeneous sensor and the DNN algorithm to analyze the sensor data [46]. The input vector of the DNN is the continuous sensor data change value, which is configured to detect a fire situation. The configuration of the DNN model is shown in Figure 4. The DNN model consists of five layers: the input layer, the output layer, and three hidden layers. The DNN model uses three types of changed input data: Final-First-Difference (FFD), which is the difference between the final value and the first value; Final-Max-Difference (FMD), which is the difference between the final value and the maximum value; and Final-Average-Difference (FAD), which is the difference between the final value and the average value. The database sends the temperature, humidity, and gas sensor data to the data-preprocessing module, which refines the data sent from the database to the input values required by the DNN model. The converted input data passes through the hidden layer, which is $100 \times 100 \times 100$, and through the output layer to the fire, warning, and normal situations. The DNN algorithm is divided into fire, warning, and general situations. Figure 5 depicts these three situations and their different graph shapes.

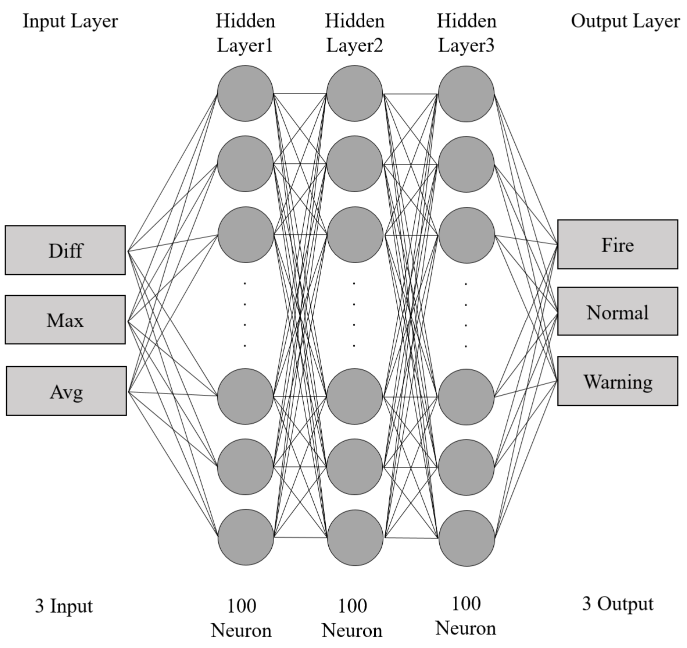


Figure 4. The proposed Deep Neural Network architecture for sequential architecture.

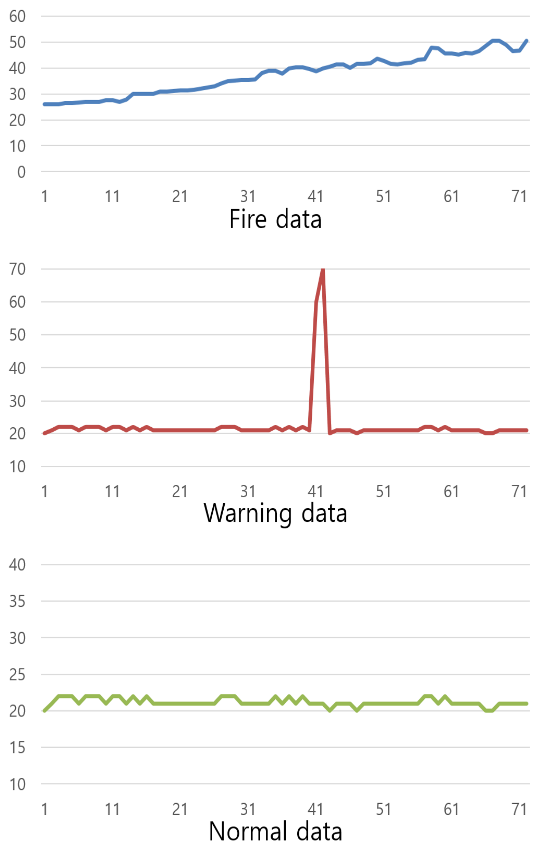


Figure 5. Learning data patterns classified into three classes.

These graphs show a temperature change over 1 min. The fire situation is a graphical representation of actual fire data, which shows a sudden change of more than 20 °C per minute. The normal data show almost constant graph movement and the abnormal situation shows abnormal graph movement over a short time. The training number and accuracy of the DNN algorithm are shown in Table 3.

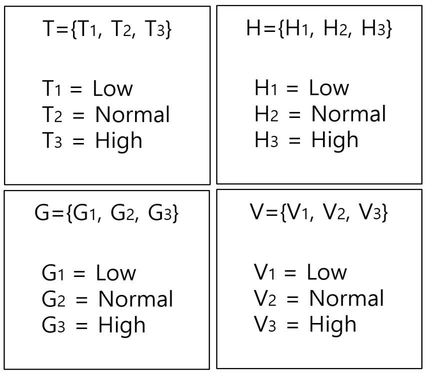
Table 3. DNN Model training set and accuracy.

	Training Data	Testing Data
Normal data	10,000	1000
Fire data	10,000	1000
Warning data	3000	300
Accuracy	99%	99%
Training Steps	69,000	-

Finally, the DNN algorithm derives the fire, warning, and normal situations. The warning situation shows only a short change in the data. The system decides if this situation is a sensor failure or a spark ignition around the sensor. Since it returns to normal data after a short period, it is not considered a fire situation, and only a warning message is sent for confirmation. The results of the fire situation and the normal situation are used in the fuzzy algorithm to calculate the final fire probability, which is identical to the CNN algorithm.

3.2.3. Adaptive Fuzzy Module

Though we can identify the fire, warning, and normal situations in each CNN and DNN algorithm, we also use a fuzzy algorithm by fusing the result of sensor and image data to obtain a more accurate fire probability [47]. In particular, the fuzzy algorithm is required to obtain the precise fire probability because the sensor data are not constant. The input set is defined as temperature (T), humidity (H), gas (G), image (V) and fire (F). Each input set is defined as shown in Figure 6.



$T=\{T_1, T_2, T_3\}$ T_1 = Low T_2 = Normal T_3 = High	$H=\{H_1, H_2, H_3\}$ H_1 = Low H_2 = Normal H_3 = High
$G=\{G_1, G_2, G_3\}$ G_1 = Low G_2 = Normal G_3 = High	$V=\{V_1, V_2, V_3\}$ V_1 = Low V_2 = Normal V_3 = High

Figure 6. Fuzzy set definition of the adaptive fuzzy algorithm.

The fuzzy algorithm uses a membership function that corresponds to the fuzzy set to obtain the membership grade of the fuzzy rule. Each member function can be expressed as a graph according to its argument value. Figure 7 shows an example of a fuzzy set membership function.

The membership function of the general fuzzy algorithm is divided into probability based on the data on the horizontal axis of the graph. The fuzzy algorithm that finds the grade of membership based on a fixed threshold is difficult to apply to a variety of environments [48]. If the humidity is low or a certain amount of gas is generated, there is a high possibility that a continuous false detection may occur. In particular, since the temperature continuously changes, it is possible to decide that the situation is a fire situation using the fixed threshold value even if a fire is not present. Therefore, the fuzzy algorithm must change the threshold value according to the situation. The fuzzy factor is the data that distinguishes between the low and high values of the grade of membership. The adaptive fuzzy algorithm is constructed by changing the fuzzy factor.

In general, the fuzzy factor of the adaptive fuzzy algorithm is updated to adapt to situation change on the basis of the suitable formula [49]. Although the adaptive fuzzy algorithm needs to update the fuzzy factor depending on the situation, it is necessary to ignore the fuzzy factor change when the fire data or warning data as shown in Figure 5 is input. We propose an improved adaptive fuzzy algorithm that can more flexibly determine fuzzy factor changes than existing adaptive fuzzy algorithms. In the proposed MAI-FDS, the result of the context preprocessing block is used to decide whether to update the fuzzy factor. The update process of the fuzzy factor is as follows. All data transmitted from the IoT gateway is stored in the database. When data analysis is completed through the DNN algorithm in the context preprocessing block, the result is sent to the fuzzy factor change function. If the DNN algorithm decides that the situation is normal, the fuzzy factor change function requests the average data from the database, which calculates the average value of the data for the last five minutes and replies. The fuzzy factor change function sets the average value as the fuzzy factor and derives the final fire probability through the fuzzy algorithm. The DNN algorithm does not change the fuzzy factor if it decides that the result is a fire or a warning situation. In the case of a fire situation, the data show a sudden change over five minutes and increases the average value. Therefore, if the fuzzy factor is changed by this average value, the final fire probability is inaccurate. In addition, when the sensor obtains data such as temperature or humidity, unpredicted exception data that may affect the accuracy of the algorithm may be received. Such anomalous data may bad affect the update in the fuzzy factor. If the DNN algorithm detects the exception data by analyzing the obtained data, it is not applied to the update of the fuzzy factor. The fuzzy membership function that is redefined by the fuzzy factor change function is shown in Figure 8.

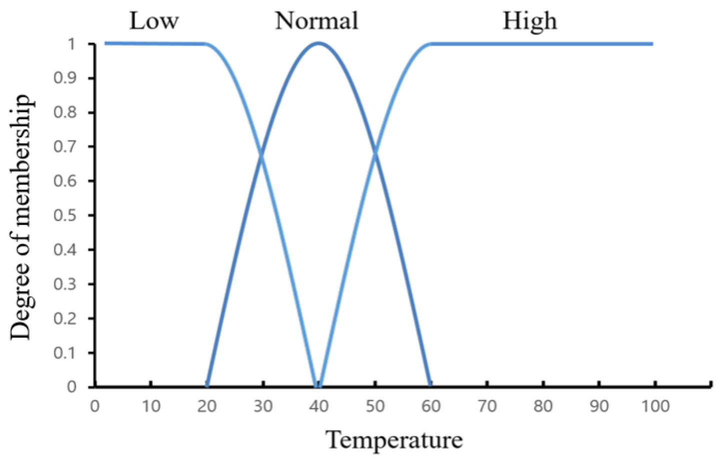


Figure 7. Example of a temperature-dependent function in general fuzzy algorithms.

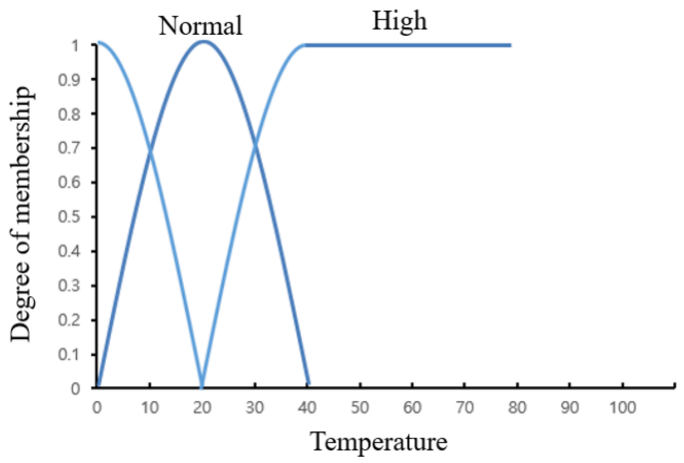


Figure 8. Example of a modified fuzzy membership function.

When the fuzzy factor changes, the membership function of each fuzzy set changes according to the change in the target environment. Therefore, accurate fire probabilities can be calculated through dynamic criteria. For example, let us suppose that the fuzzy factor of the temperature membership function is 40. When the average temperature of the environment is 10 °C, the membership function of the existing fuzzy algorithm does not change. However, the adaptive fuzzy algorithm membership function is changed to 10 °C by changing the fuzzy factor. In this case, when a fire occurs and the temperature change suddenly increases to 30 °C, the existing fuzzy algorithm has a low fire probability because the fuzzy factor is 40 °C, but the probability of the adaptive fuzzy algorithm is high. Thus, the adaptive fuzzy algorithm enables precise and speedy fire measurement in a variety of environments.

3.3. Direct-MQTT

The fire detection delay time can be divided into the decision delay and the transfer delay. The decision delay is the computation time for the calculation of the fire probability depending on the complexity of the fire detection algorithm and the computing power of the fire detection server. The transfer delay includes processing delay, transmission delay, propagation delay and queuing delay on SDN switches. In addition, the queuing delay on a MQTT broker is an important consideration for time critical IoT services. The standard MQTT protocol is based on a broker [50]. All publishers and subscribers are connected to the broker to transmit data. This mechanism does not require the Discovery process because the publisher and the subscriber do not have to check each other's location, which solves the problem of traffic overload caused by discovery. However, the data transmitted to the broker can be concentrated, which may cause traffic congestion and consequently, data transfer delay. An example of the existing MQTT protocol traffic flow using an SDN is shown in Figure 9.

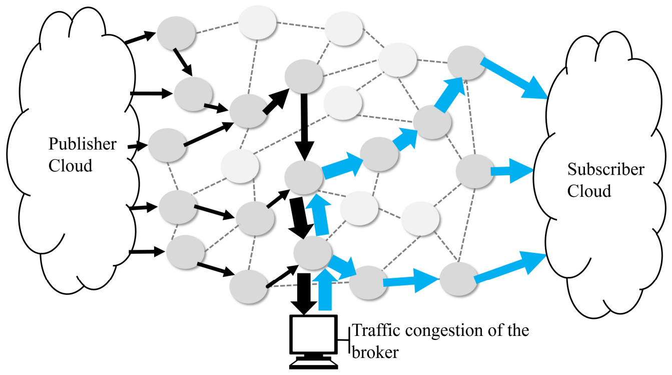


Figure 9. Example of the existing MQTT data transfer path in a mesh topology.

Figure 9 shows the path of the data centered on the central broker. The broker queuing delay is added to the data transfer delays because of the data concentration and the delay time (T_{total_delay}) is calculated as in (1):

$$T_{total\_delay} = T_{decision} + T_{transfer} + T_{broker\_queuing} \tag{1}$$

Since the value of $T_{broker\_queuing}$ may be measured by tens of seconds, reducing the broker queuing delays should be essential in order to speed up the fire detection system. We propose Direct-MQTT as a novel MQTT mechanism to minimize the broker queuing delays. The mechanism solves the traffic concentrated on the broker by functionally integrating the SDN controller and the extended MQTT broker. The structure of the integrated SDN controller is shown in Figure 10.

The SDN controller adds an extended MQTT block to distribute the flow of data during the routing process. The standard MQTT protocols transmit data using topics. Publishers and subscribers are connected through the broker and transfer messages. The publisher sends the topic that contains data to the broker, and the subscriber receives the topic from the broker. The broker stores the subscriber's IP address and information on their topic subscription. The SDN controller collects the information on the subscribers stored in the broker. The SDN controller determines the flow of all MQTT packets based on their information to prevent transfer delay. The transmission process of each packet is shown in Figure 11.

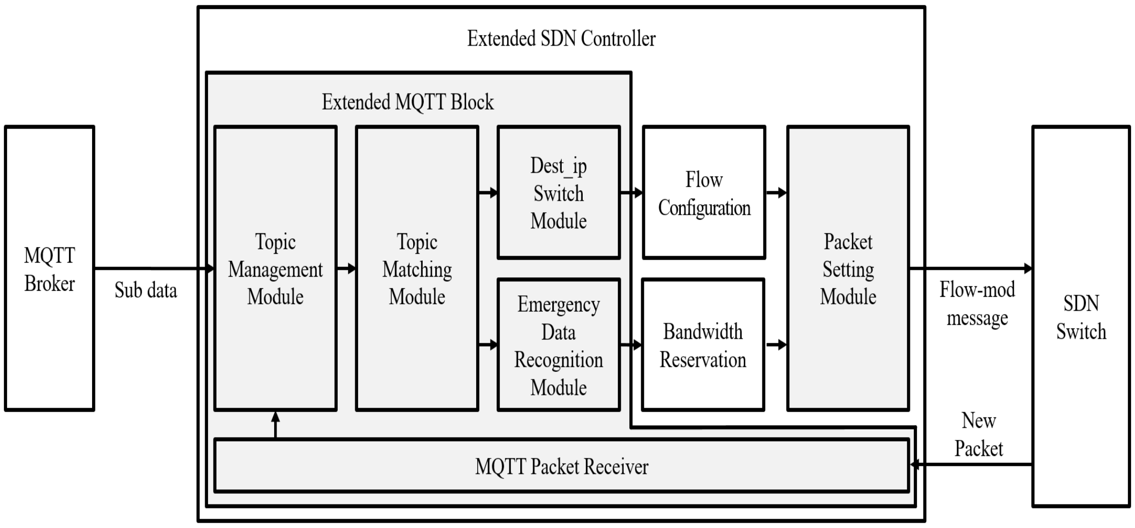


Figure 10. The proposed SDN controller using the extended MQTT Block.

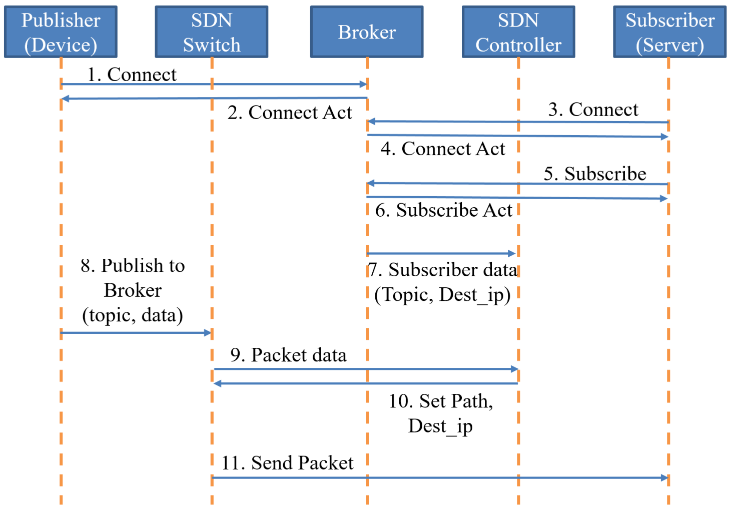


Figure 11. Data transfer sequence diagram of Direct-MQTT.

Publishers and subscribers are connected through the broker and transfer messages. The subscriber sends a subscribe message to the broker to notify it of the subscription topic. The broker sends the subscription topic and IP address of the subscriber to the SDN controller, which creates the subscriber information table in the topic management module of the extended MQTT block. The publisher sends the collected data to the broker and the SDN switch sends the first incoming data to the SDN controller for routing. The MQTT Packet Receiver Module in the extended MQTT block checks the MQTT packet and sends it to the topic-matching module, which matches the topic of the packet information with the topic on the subscriber information table. After the topic is matched, the emergency data recognition module identifies whether it is emergency data. The dest_ip switch module changes destination address of the packet and the SDN controller sets its path. Finally, the packet setting module sends the data of the set route and the destination address to the SDN switch, which determines the flow of the MQTT packet through the configuration data transmitted from the SDN controller. Since the changed destination address is the subscriber's IP address, the data sent from the publisher is delivered to the subscriber instead of the broker. As a result, traffic distribution minimizes data transfer delays by eliminating queuing delays in the broker.

An example of the traffic flow of the proposed MQTT protocol is shown in Figure 12. Compared with Figure 9, the flow of MQTT data in Figure 12 is more distributed. In Figure 9, only the optimal path to the central broker is set, although the SDN is used. In addition, the broker and subscribers are set to a single path with a one-to-one connection. However, in Figure 12, an SDN controller with an extended MQTT block distributes the broker's centralized data path, creating a path directly to the subscriber. Minimizes data transfer delays by eliminating queuing delays in the broker by distributing traffic to all data from the publisher, not to the broker but directly to the subscriber.

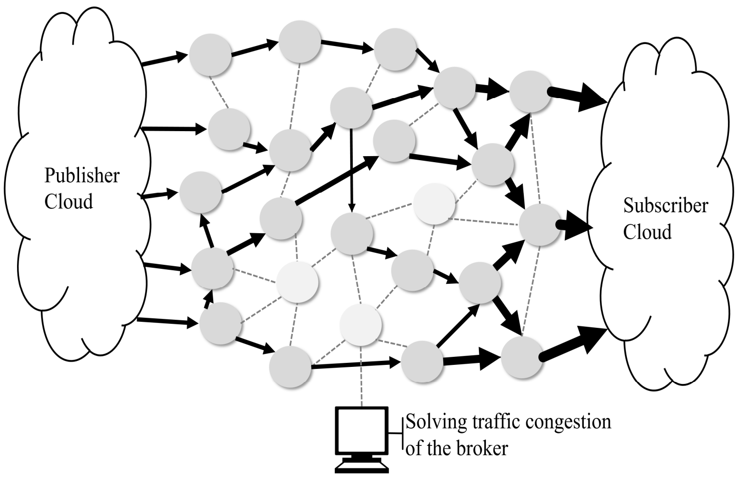


Figure 12. Example of the Direct-MQTT data transfer path in a mesh topology.

4. Analysis

In this section, we perform fire detection and data transfer delay tests to verify the MAI-FDS. In order to verify the accuracy and speed of the algorithm, we compare MAI-FDS and EFDS fire probability graphs. The data transfer delay tests measure the latency of standard MQTT, CoAP, and Direct-MQTT in a complex network environment.

4.1. Experimental Environment

Actual fire data are required to verify the accuracy and promptness of MAI-FDS. Experiments with real fires are limited because of costs and accident risks. In addition, several environmental variables make it impossible to generate identical fire effects. To solve this problem, we used a virtual environment that repeatedly reproduced the characteristics of an actual fire. The Fire Dynamics Simulator (FDS) [51], which is an open-source fire simulation developed by the U.S. National Institute of Standards and Technology (NIST), has proven its reliability in previous studies. The simulated data of the FDS, in which the model is finely structured, showed errors within 10% of the actual fire data [52–54]. We used the FDS to generate fires in a virtual environment to obtain temperature, humidity, and gas sensor data. Table 4 shows the types of fuel and fuel sources in the virtual environment generated by the FDS. The required data for the image analysis algorithms used a fire video generated in a similar space as the environment configured in the FDS. The network testbed is based on the Mininet SDN emulator [55]. As the SDN controller, we use the Ryu controller [56] to manage the entire network flow. Figure 13 shows the network topology, which simulated the scenarios in Figures 9 and 12 in this experiment.

Figure 13 shows the network topology with 24 SDN switches. The total number of hosts was 18. On the left, there are the 10 hosts that acted as the publisher, which serves as a gateway for collecting sensor data. As the MQTT broker, we used a Mosquitto broker [50] in the center of the network topology. On the right, there are the 7 connected hosts that acted as the subscriber. All switches were connected to the SDN controller to control the network flow. Based on the topology in Figure 13, the number of published samples and subscriber hosts changed and the test was performed. In the experimental environment, all paths were specified as the minimum path using Dijkstra's algorithm [57].

Table 4. Fire source and fuel source in the simulation.

Fuel Source	**Fuel Name**	**Chemical Formula**
	N-HRPTANE	Heptane, C_7H_{16}
Fire Source	**Object Name**	**Elements of Object**
	Gas range frame	PLASTIC (75%) STEEL (25%)
	Futon/Pillow	FABRIC
	Sofa	FOAM (80%) FABRIC (20%)

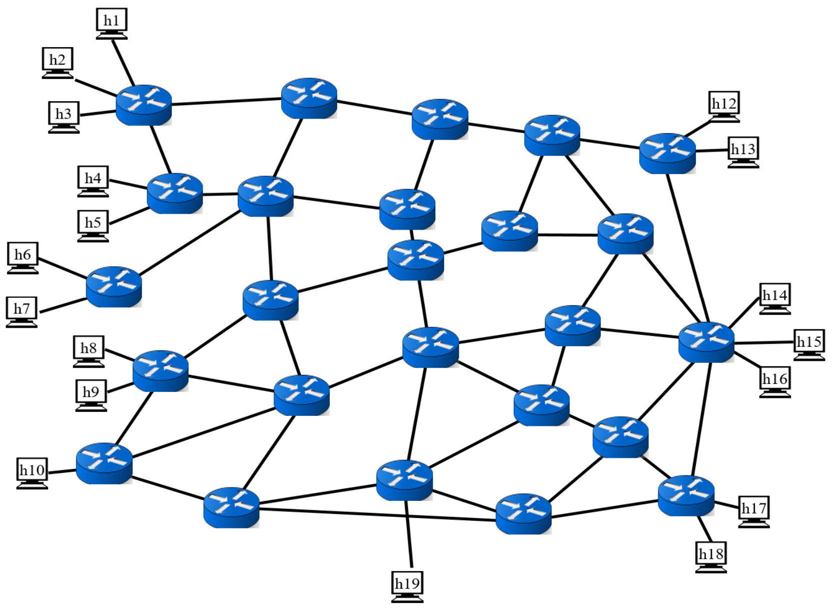


Figure 13. Network topology used in the experiments.

4.2. Fire Detection Algotrithm

In order to verify the speed of the fire decision algorithms used in MAI-FDS, three algorithms were compared: the temperature threshold algorithm, the CNN-based algorithm, and the early fire detection system (EFDS) [28]. The fire decision algorithm uses temperature, humidity, gas, and image data as decision data. The temperature threshold algorithm is based on a simple decision algorithm by temperature data and determines a fire if the temperature exceeds 50 °C. The CNN-based algorithm uses image data to determine fire. The EFDS configured with a fuzzy algorithm uses both temperature, humidity, and gas sensor data, but it does not use image data. The experiment data are shown in Figure 14.

The experiment was carried out over six minutes. The DNN and fuzzy algorithms set an adaptive threshold based on five minutes of data. MAI-FDS calculates the current fire probability by combining newly collected data and accumulated 5-min data. Figure 14 shows the change in the experimental data. In the video data, the fire was reduced to 131 s, the temperature to 132 s, the gas concentration to 138 s, and the humidity to 136 s. Using this data, the fire probability graph per second is shown in Figure 15.

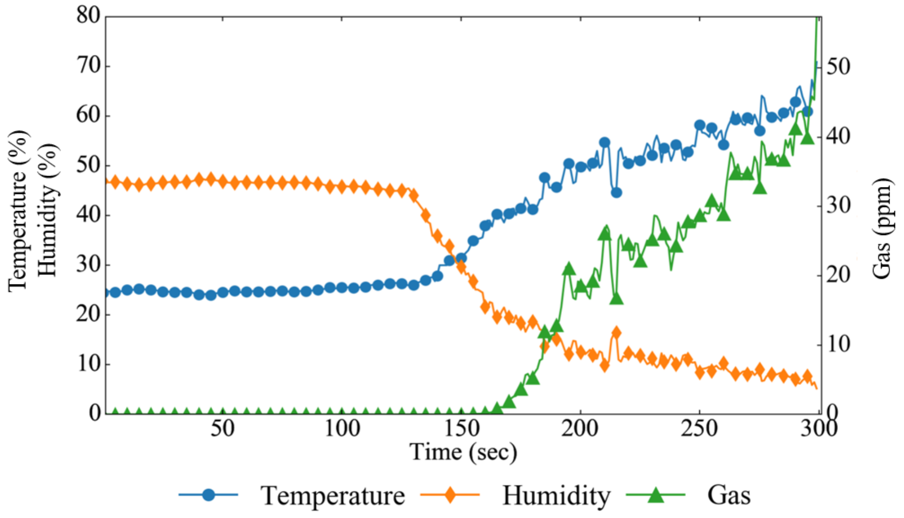


Figure 14. Changes in temperature, humidity, and gas data in the experiment.

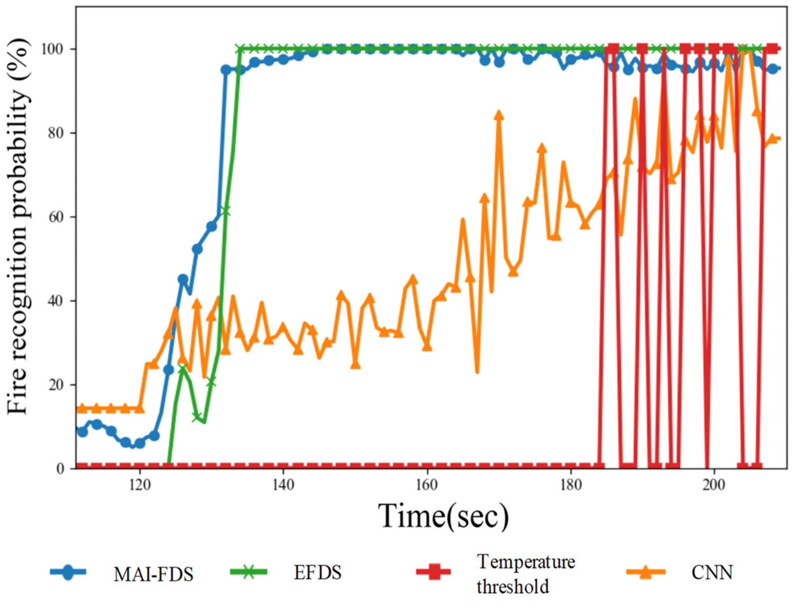


Figure 15. Measured fire probability variation of MAI-FDS, The CNN algorithm, the EFDS and the temperature threshold algorithm.

In this paper, we did not consider a situation where a sensor fails or the system is turned off and cannot detect a fire. We first verify the speed of fire detection with the assumption that the sensor and system will operate normally. The fire detection time of the temperature threshold algorithm is the slowest due to limitations of the detection capability of the sensor because it only analyzes temperature data. The CNN algorithm is fast at initial detection, but because of image quality limit, it takes a long time for the probability of fire to rise to a certain level. The EFDS using both temperature, humidity, and gas sensor data is similar to MAI-FDS. However, MAI-FDS recognizes fire by 6 s faster than the EFDS algorithm when the fire detection threshold is set to 40%. By increasing the fire threshold to 90%, MAI-FDS is 16 s faster than the CNN.

The difference in fire detection time between the other algorithms and MAI-FDS occurs in the processing of sensor data and adaptive fuzzy algorithms. Unlike algorithms that use static rules, MAI-FDS uses z deep-learning algorithm and can train changes in temperature and change the threshold value of the fire decision. When calculating fire probability using temperature data, the temperature threshold algorithm does not show a high probability until it exceeds 50 °C. MAI-FDS shows a high probability of fire if the temperature sharply increases even before it reaches 50 °C. In Figure 14, the temperature shows a rapid change of 10 °C for 30 s between 130 and 160 s. Since the temperature was less than 50 °C at 160 s, the temperature threshold algorithm showed a low fire probability for the sensor data. MAI-FDS detects sudden changes in data and reconfigures the temperature threshold value of the decision algorithm. MAI-FDS outputs high fire probabilities due to updated thresholds even at low temperatures. The EFDS that detect low humidity data recognize fire faster than the temperature threshold algorithm that uses only temperature data. The EFDS is also slower than MAI-FDS because of the limitations of rule-based algorithms. When the fire probability threshold for fire alarm is set to 90%, the fire alarm occurrence time of each algorithm is as follows: MAI-FDS is 12 s; EFDS is 14 s; the CNN algorithm is 72 s; and the temperature threshold algorithm is 62 s.

In order to verify the fire recognition flexibility of the adaptive fuzzy algorithm, we compare fire probabilities of various algorithms using the restaurant data during cooking activities. The temperature, humidity and gas data of a restaurant steeply fluctuate as the cooking process. When the cooking begins, all sensor data will rise. Sometimes the humidity and gas data show opposite results depending on the item being cooked. The identification of fire in a kitchen can be made by the changing data in a short time. Gas data can increase drastically under cooking, but temperature and humidity data are plotted with a gentle slope. If a fire actually occurs, all the sensing data including temperature, humidity and gas will show corrupt changes in a specific period. The adaptive fuzzy algorithm recognizes this difference, changes the threshold value, and can decide the fire situation according to the changed threshold even when the similar patterns of temperature, humidity, and gas data are received. The input data for the experiments are temperature, humidity and gas data generated in the restaurant for 24 h. The image data are excluded from the input data because it is not affected by the adaptive fuzzy algorithm. The data assume a situation in which cooking takes place from 10:00 to 15:00, 16:00 to 22:00, and a fire occurs at 11:00. Changes in data during the cooking process show a gradual change due to kitchen ventilation. Data in a fire situation is a time when the ventilation system does not work and shows a rapid change due to an explosion of fire. The probability of fire recognition over time is shown in Figure 16. The temperature threshold algorithm often recognize fire because more than 50 °C of temperature data are detected during cooking. Most of the cooking process is highly humid, but if a lot of gas is generated under specific circumstances, the EFDS maybe consider situation as a fire due to high temperature and high gas. Since the adaptive fuzzy algorithm of MAI-FDS analyzes the data for five minutes, the data in the cooking process with gentle slope is not identified as a fire. The fuzzy factor is adjusted with the variation of the data, and the probability of a fire does not change even if the temperature exceeds the threshold value, in this case 50 °C.

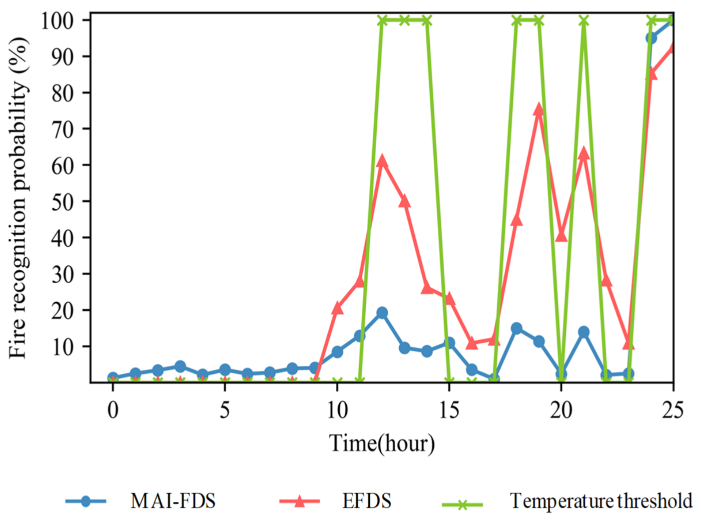


Figure 16. Fire recognition probabilities of MAI-FDS with adaptive fuzzy, the EFDS and the temperature threshold algorithm.

4.3. Data Tranmission Experiments

The experiments of the transfer delay time for HTTP, CoAP, the standard MQTT and Direct MQTT are made in terms of scalability, that is to say, the number of publishers and subscribers. The experimental environment is shown in Figure 13. The number of switches was 24, and the location of the broker was set to the center of the topology. The network congestion for the propagation delay, queuing delay, and transfer delay was set to 5 ms per path. To test a situation in which the quantity of sensor data increased, we increased the number of publishers from the host, which acts as a gateway. In this experiment, each host increased the number of publishers by 25, starting from one. Finally, the subscriber collected up to 1000 samples and calculated the average delays.

Figure 17 shows the transfer delay test results for the fire data. The standard MQTT showed increasing latency when it sent more than 500 samples. The standard MQTT uses a broker to transmit data, resulting in a broker queue delay. The CoAP protocol also shows a similar transfer delay because it transmits data through a central server. HTTP has higher latency than standard MQTT and CoAP due to header overhead. Since Direct-MQTT eliminates queue delays in the broker by distributing the data transmission path, it can transmit data 30% faster than the standard MQTT. However, the Direct-MQTT also showed that when many publishers send data, the delay increases with the network congestion.

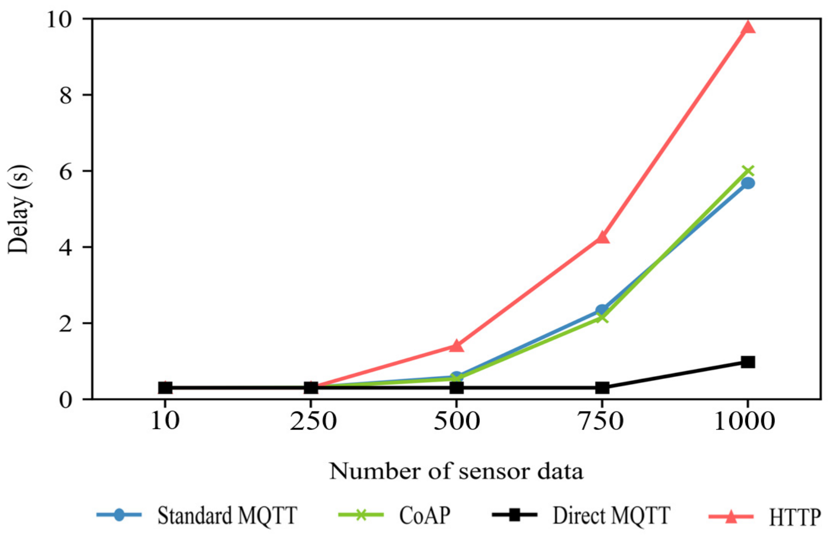


Figure 17. Changes in data transfer delay depending on the number of fire samples.

The experiments showed that an increase in the number of publishers is based on a single subscriber. Therefore, if the broker is near the subscriber, the standard MQTT obtains a similar result to the Direct-MQTT. However, in a network configuration where there are multiple subscribers, the location of the broker cannot be set near a specific subscriber. Since the IoT system transmits data between multiple publishers and subscribers, it is necessary to check the transfer delay in multi-subscriber situations. In this experiment, we configured the network topology as the default setting. It consisted of 24 switches, a centrally located broker, and 25 publishers from 10 hosts transmitting data. The congestion level of the network was set to 5 ms. We measured the transfer delay by increasing the number of subscribers from one to seven. We assumed that all publishers transmitted the data to all subscribers.

Figure 18 shows the data transmission delay with an increasing number of subscribers. For multiple subscribers, the standard MQTT publisher sent the data to the broker without further action. Additional operations on the standard MQTTs were performed on the broker. For a single subscriber, the broker deletes the data from the queue immediately after it sends the data. However, with multiple subscribers, the broker increases the processing delays because the process of sending data to all subscribers is completed and the data are removed from the queue. When there were seven subscribers, an average delay of 8 s was measured with a maximum delay of 52 s. The HTTP protocol and the CoAP protocol have additional latency in the process of sending data from a single server to multiple clients. Distributed servers solve latency issues, but many servers have management difficulties and cost issues. Direct-MQTT had a shorter total delay time than the standard MQTT because there was no broker queuing delay.

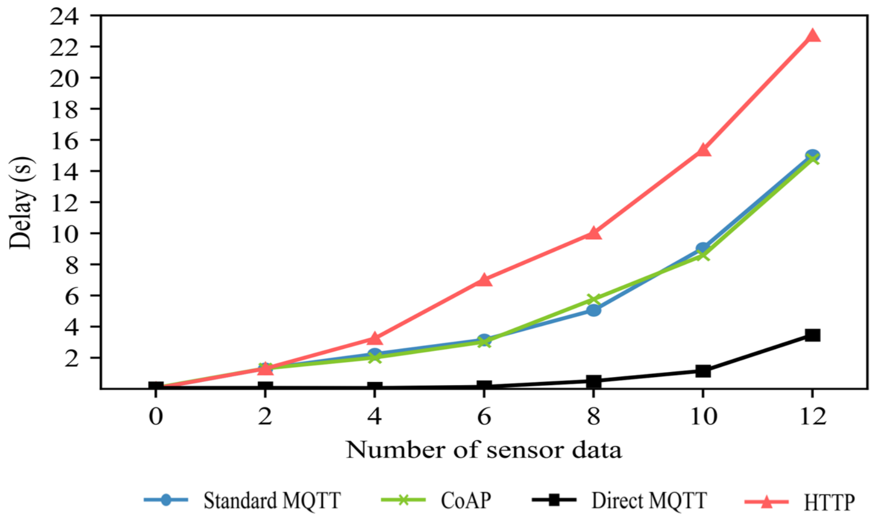


Figure 18. Changes in data transfer delay depending on the number of subscribers.

By comparing the Direct-MQTT delay times in Figures 17 and 18, we may note that the total delay increases when the number of subscribers increases in the same network topology. In Figure 17, the measured total delay was 0.3 s, but in Figure 18, it was 3 s. In the Direct-MQTT, each publisher used unicast to directly send data to multiple subscribers. The Direct-MQTT has the disadvantage of increasing the number of unicast routes due to the increase in number of publishers, subscribers, and final delay because of the network congestion in the path.

As described in Equation (1), the end-to-end delay is the sum of the decision delay and the data transfer delay, or the time between fire occurrence and detection. In order to compare the end-to-end delay, we simulated a fire probability measurement experiment where data was collected from 750 heterogeneous sensors. The network topology of the experiment is shown in Figure 13 and the data in the experiment is shown in Figure 14. MAI-FDS transmits the data using Direct-MQTT and the others transmit the data using the standard MQTT. The experimental results of all systems are compared in Figure 19. The calculated decision delay of MAI-FDS was 5 and 13 s end-to-end, and the decision delay of CNN algorithm was 12 s and 72 s end-to-end, as shown in Figure 15. The data transfer delay of Direct-MQTT is nearly 0 s, and the data transfer delay of the standard MQTT only exceeds 2 s when 750 sensor samples are sent, as shown in Figure 17. Thus, the end-to-end delays of MAI-FDS are approximately 5 s and 13 s, and the end-to-end delays of fuzzy algorithm are approximately 14 s and 74 s.

The calculated end-to-end delays of each system are similar to the measurement delays, as shown in Figure 19. As a result, MAI-FDS is approximately 72 % faster than fuzzy algorithm.

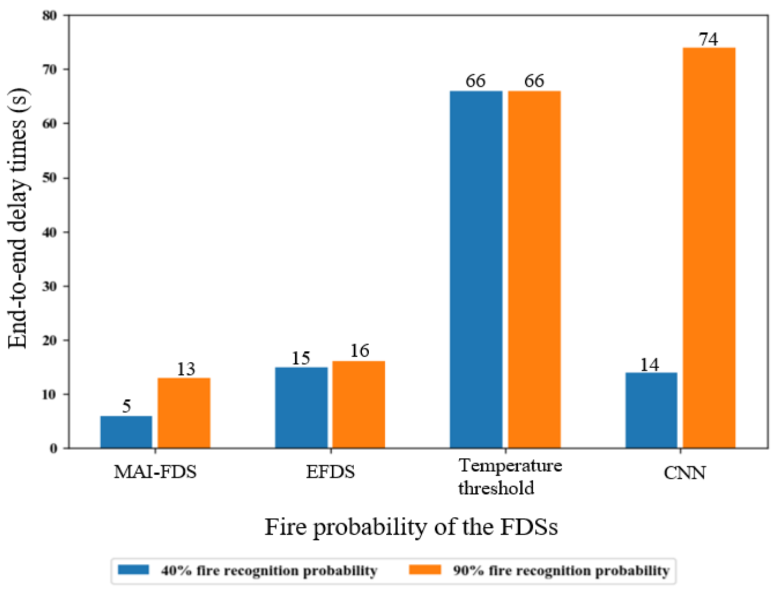


Figure 19. Total end-to-end delay time according to the fire probability of each.

5. Conclusions

Fire detection systems should consider two important engineering factors: fire detection probability and transfer delay. Since the existing fire detection systems use rule-based algorithms or simple AI algorithms, there are low detection accuracy problems caused by different environmental conditions, in addition to little usage of heterogeneous sensors. In addition, the legacy systems have difficulty satisfying promptness requirements because they do not consider the transfer delays of fire events caused by traffic congestion on a specific node or a server. In this paper, we proposed the usage of MAI-FDS to solve these problems. MAI-FDS adopts a multifunctional AI framework with an IoT data collection block, a context preprocessing block, and a context decision block. Each functional block in the framework has high flexibility depending on its applications. MAI-FDS showed much higher fire recognition capability than the legacy FDSs based on rules, CNN, and fuzzy methods. In addition, the adaptive fuzzy algorithm reduces the false alarms by calculating the amount of data change. In experiments using restaurant data, there were cases where static algorithms were decided as a fire at the time of cooking. MAI-FDS, which uses an adaptive fuzzy algorithm, did not deduced these as a fire at the time of cooking because the data showed no rapid change. Incorporating Direct-MQTT in MAI-FDS improved the speed of the fire detection system through its path distribution mechanism. The SDN controller supported by Direct-MQTT created direct paths between each publisher and the corresponding subscribers. This direct path mechanism effectively reduced the transfer delays by eliminating the queuing delays in the broker. As a result, MAI-FDS achieved an average fire detection accuracy of above 95% and reduced the end-to-end transfer delay by 67% compared with that of the existing fire detection systems. We believe that the research results of MAI-FDS will be an important system of future smart cities and will help prevent the spread of fires through accurate fire detection. Accurate fire detection and reduced latency of critical alarm events improve the reliability of the disaster mitigation system. In a future smart city where all infrastructures can be intelligent, a reliable disaster mitigation system will effectively support rapid firefighting measures such as optimizing the route to the fire scene. However, MAI-FDS adapts in all environments, making it difficult to accurately determine the fire. Adaptive fuzzy algorithms and multifunctional AI frameworks have been designed for environmentally adaptive fire detection systems, but the ability to detect fires in extreme environments is still weak. If a melting furnace is used in a factory, the temperature is very high and there is a continuing presence of flames, which can increase the false positive rate in MAI-FDS. Therefore, in future research, it is necessary to develop a system that can modify the fire detection algorithm and develop a multifunctional AI framework to adaptively detect the fire in the real environment.

Author Contributions: Conceptualization, J.H.P., S.L.; methodology, J.H.P., S.Y. and S.L.; software, validation, S.Y., S.L. and H.K.; formal analysis, H.K., S.L.; investigation, J.H.P., S.L.; writing—original draft preparation, J.H.P., H.K.; writing—review and editing, J.H.P., S.L.; visualization, J.H.P., S.Y.; supervision, project administration, funding acquisition, W.-T.K.

Funding: This research was supported by the Institute for Information & communications Technology Promotion (IITP) (No. 2018-0-01456-002), and by the National Research Foundation of Korea (No. 2017R1A2B4010875) from the Korean government (MSIP).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Karter, M.J. *False Alarm Activity in the US 2012*; National Fire Protection Association: Quincy, MA, USA, 2013.
2. Ahrens, M. *Trends and Patterns of US Fire Loss*; National Fire Protection Association: Quincy, MA, USA, 2017.
3. Ahrens, M. *Smoke Alarms in US Home Fires*; Fire Analysis and Research Division: Quincy, MA, USA, 2009.
4. Kwon, O.H.; Cho, S.M.; Hwang, S.M. Design and implementation of fire detection system. In Proceedings of the Advanced Software Engineering and Its Applications, Hainan Island, China, 13–15 December 2008; pp. 233–236.
5. Chen, S.J.; Hovde, D.C.; Peterson, K.A.; Marshall, A.W. Fire detection using smoke and gas sensors. *Fire Saf. J.* **2007**, *42*, 507–515. [CrossRef]
6. Zervas, E.; Mpimpoudisb, A.; Anagnostopoulos, C.; Sekkasb, O.; Hadjiefthymiades, S. Multisensor data fusion for fire detection. *Inf. Fusion* **2011**, *12*, 150–159. [CrossRef]
7. Bahrepour, M.; van der Zwaag, B.J.; Meratnia, N.; Havinga, P. Fire data analysis and feature reduction using computational intelligence methods. *Adv. Intell. Decis. Technol.* **2010**, *4*, 289–298.
8. Hietaniemi, J.; Hostikka, S.; Vaari, J. FDS Simulation of Fire Spread—Comparison of Model Results with Experimental Data. Available online: https://www.vtt.fi/inf/pdf/workingpapers/2004/W4.pdf (accessed on 28 April 2019).
9. Sandström, J. Temperature Calculations in Fire Exposed Structures with the Use of Adiabatic Surface Temperatures. Master's Thesis, Luleå University of Technology, Luleå, Sweden, 2008.
10. Harman, M. The role of artificial intelligence in software engineering. In Proceedings of the 2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), Zurich, Switzerland, 5 June 2012; pp. 1–6.
11. Nguyen, T.T.; Armitage, G. A survey of techniques for internet traffic classification using machine learning. *IEEE Commun. Surv. Tutor.* **2008**, *10*, 56–76. [CrossRef]
12. Krizhevsky, A.; Sutskever, L.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *1*, 1097–1105. [CrossRef]
13. Mikolov, T.; Karafiát, M.; Burget, L.; Černocký, J.; Khudanpur, S. Recurrent neural network based language model. In Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Chiba, Japan, 26–30 September 2010; pp. 1097–1105.
14. Pouyanfar, S.; Chen, S.-C. Semantic event detection using ensemble deep learning. In Proceedings of the 2016 IEEE International Symposium on Multimedia (ISM), San Jose, CA, USA, 11–13 December 2016; pp. 203–208.
15. National Fire Data System. Available online: http://www.nfds.go.kr/fr_fact_0101.jsf (accessed on 12 March 2019).
16. Won, J.S.; Kim, S.G. *A Study Focused on Responding to Fire-Related Accidents*; The Seoul Institute: Seoul, Korea, 2016.
17. Imteaj, A.; Rahman, T.; Hossain, M.K.; Alam, M.S.; Rahat, S.A. An IoT based fire alarming and authentication system for workhouse using Raspberry Pi 3. In Proceedings of the 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox's Bazar, Bangladesh, 16–18 February 2017; pp. 899–904.

18. Lim, Y.; Lim, S.; Choi, J.; Cho, S.; Kim, C.K.; Lee, Y.W.; Zhang, H.; Hu, H.; Xu, B.; Li, J.; et al. A fire detection and rescue support framework with wireless sensor networks. In Proceedings of the 2007 International Conference on Convergence Information Technology (ICCIT 2007), Gyeongju, Korea, 21–23 November 2007; pp. 135–138.
19. Lee, K.C.; Lee, H.H. Network-based fire-detection system via controller area network for smart home automation. *IEEE Trans. Consum. Electron.* **2004**, *50*, 1093–1100.
20. MQTT Protocol Binding, oneM2M-TS-0010. Available online: http://www.onem2m.org/images/files/deliverables/TS-0010-MQTT_Protocol_Binding-V1_5_1.pdf (accessed on 28 April 2019).
21. CoAP Protocol Binding, oneM2M-TS-0008. Available online: http://www.onem2m.org/images/files/deliverables/TS-0008-CoAP_Protocol_Binding-V1_3_2.pdf (accessed on 28 April 2019).
22. Vijayalakshmi, S.R.; Muruganand, S. A survey of Internet of Things in fire detection and fire industries. In Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), Palladam, India, 10–11 February 2017; pp. 703–707.
23. McKeown, N. Software-Defined Networking. Available online: https://www.cs.odu.edu/~{}cs752/papers/sdr-infocom_brazil_2009_v1-1.pdf (accessed on 28 April 2019).
24. Tortonesi, M.; Michaelis, J.; Morelli, A.; Suri, N.; Baker, M.A. SPF: An SDN-based middleware solution to mitigate the IoT information explosion. In Proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 27–30 June 2016; pp. 435–442.
25. Shen, D.; Chen, X.; Nguyen, M.; Yan, W.Q. Flame detection using deep learning. In Proceedings of the 2018 4th International Conference on Control, Automation and Robotics (ICCAR), Auckland, New Zealand, 20–23 April 2018; pp. 416–420.
26. Kaabi, R.; Sayadi, M.; Bouchouicha, M.; Fnaiech, F.; Moreau, E.; Ginoux, J.M. Early smoke detection of forest wildfire video using deep belief network. In Proceedings of the 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, Tunisia, 21–24 March 2018; pp. 1–6.
27. Hu, C.; Tang, P.; Jin, W.; He, Z.; Li, W. Real-Time Fire Detection Based on Deep Convolutional Long-Recurrent Networks and Optical Flow Method. In Proceedings of the 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 9061–9066.
28. Saputra, F.A.; Al Rasyid, M.U.H.; Abiantoro, B.A. Prototype of early fire detection system for home monitoring based on Wireless Sensor Network. In Proceedings of the 2017 International Electronics Symposium on Engineering Technology and Applications (IES-ETA), Surabaya, Indonesia, 26–27 September 2017; pp. 39–44.
29. Jang, J.Y.; Lee, K.W.; Kim, Y.J.; Kim, W.T. S-FDS: A Smart Fire Detection System based on the Integration of Fuzzy Logic and Deep Learning. *J. Inst. Electron. Inf. Eng.* **2017**, *54*, 50–58.
30. Yao, J.; Raffuse, S.M.; Brauer, M.; Williamson, G.J.; Bowman, D.M.; Johnston, F.H.; Henderson, S.B. Predicting the minimum height of forest fire smoke within the atmosphere using machine learning and data from the CALIPSO satellite. *Remote Sens. Environ.* **2018**, *206*, 98–106. [CrossRef]
31. Ngoc-Thach, N.; Ngo, D.B.T.; Xuan-Canh, P.; Hong-Thi, N.; Thi, B.H.; NhatDuc, H.; Dieu, T.B. Spatial pattern assessment of tropical forest fire danger at Thuan Chau area (Vietnam) using GIS-based advanced machine learning algorithms: A comparative study. *Ecol. Inf.* **2018**, *46*, 75–85. [CrossRef]
32. Muhammad, K.; Ahmad, J.; Lv, Z.; Bellavista, P.; Yang, P.; Baik, S.W. Efficient Deep CNN-Based Fire Detection and Localization in Video Surveillance Applications. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *99*, 1–16. [CrossRef]
33. Liu, W.; Wang, Z.; Liu, X.; Zeng, N.; Liu, Y.; Alsaadi, F.E. A survey of deep neural network architectures and their applications. *Neurocomputing* **2017**, *234*, 11–26. [CrossRef]
34. Kim, Y.J.; Kim, E.G. Image based fire detection using convolutional neural network. *J. Korea Inst. Inf. Commun. Eng.* **2016**, *20*, 1649–1656. [CrossRef]
35. Xu, S.; Man, M.W.; Chi, C.C. Deep neural networks versus support vector machines for ECG arrhythmia classification. In Proceedings of the 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Hong Kong, China, 10–14 July 2017; pp. 127–132.
36. Bui, D.T.; Bui, Q.T.; Nguyen, Q.P.; Pradhan, B.; Nampak, H.; Trinh, P.T. A hybrid artificial intelligence approach using GIS-based neural-fuzzy inference system and particle swarm optimization for forest fire susceptibility modeling at a tropical area. *Agric. For. Meteorol.* **2017**, *233*, 32–44.
37. MQ Telemetry Transport. Available online: http://mqtt.org (accessed on 12 March 2019).

38. Bormann, C.; Castellani, A.P.; Shelby, Z. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Comput.* **2012**, *2*, 62–67. [CrossRef]
39. Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berners-Lee, T. Hypertext Transfer Protocol—HTTP/1.1. IETF RFC 2616. 1999. Available online: https://www.rfc-editor.org/rfc/rfc2616.txt (accessed on 30 April 2019).
40. Klauck, R.; Kirsche, M. XMPP to the rescue: Enhancing post disaster management and joint task force work. In Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications Workshops, Lugano, Switzerland, 19–23 March 2012; pp. 752–757.
41. Park, J.H.; Kim, H.S.; Kim, W.T. DM-MQTT: An Efficient MQTT Based on SDN Multicast for Massive IoT Communications. *Sensors* **2018**, *18*, 3071. [CrossRef]
42. Santamaria, A.F.; De Rango, F.; Serianni, A.; Raimondo, P. A real IoT device deployment for e-Health applications under lightweight communication protocols, activity classifier and edge data filtering. *Comput. Commun.* **2018**, *128*, 60–73. [CrossRef]
43. Santamaria, A.F.; Raimondo, P.; Tropea, M.; De Rango, F.; Aiello, C. An IoT Surveillance System Based on a Decentralised Architecture. *Sensors* **2019**, *19*, 1469. [CrossRef]
44. Sharif Razavian, A.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN features off-the-shelf: An astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Stockholm, Sweden, 23–28 June 2014; pp. 806–813.
45. Oquab, M.; Bottou, L.; Laptev, I.; Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1717–1724.
46. Li, J.; Zhao, R.; Huang, J.T.; Gong, Y. Learning small-size DNN with output-distribution-based criteria. In Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association, Redmond, WA, USA, 14–18 September 2014.
47. Xu, Z. Intuitionistic fuzzy aggregation operators. *IEEE Trans. Fuzzy Syst.* **2007**, *15*, 1179–1187.
48. Dotcenko, S.; Vladyko, A.; Letenko, I. A fuzzy logic-based information security management for software-defined networks. In Proceedings of the 16th International Conference on Advanced Communication Technology, Pyeongchang, Korea, 16–19 February 2014; pp. 167–171.
49. Tong, S.; Yongming, L. Observer-based fuzzy adaptive control for strict-feedback nonlinear systems. *Fuzzy Sets Syst.* **2009**, *12*, 1749–1764. [CrossRef]
50. Light, R.A. Mosquitto: Server and client implementation of the MQTT protocol. *J. Open Sour. Softw.* **2017**, *2*, 265. [CrossRef]
51. McGrattan, K.; Hostikka, S.; Floyd, J.; Baum, H.; Rehm, R.G.; Mell, W.; McDermott, R. *Fire Dynamics Simulator (Version 5), Technical Reference Guide*; NIST Special Publication: Gaithersburg, MD, USA, 2004.
52. Hu, L.H.; Fong, N.K.; Yang, L.Z.; Chow, W.K.; Li, Y.Z.; Huo, R. Modeling fire-induced smoke spread and carbon monoxide transportation in a long channel: Fire dynamics simulator comparisons with measured data. *J. Hazard. Mater.* **2007**, *140*, 293–298. [CrossRef]
53. Kerber, S.I.N. Evaluation of the Ability of Fire Dynamic Simulator to Simulate Positive Pressure Ventilation in the Laboratory and Practical Scenarios. Ph.D. Thesis, University of Maryland, College Park, MD, USA, 2005.
54. McGrattan, K.B.; Baum, H.R.; Rehm, R.G. Large eddy simulations of smoke movement. *ASHRAE Trans.* **1999**, *105*, 426. [CrossRef]
55. An Instant Virtual Network on your Laptop (or other PC). Available online: http://mininet.org (accessed on 12 March 2019).
56. Ryu SDN Framework. Available online: https://osrg.github.io/ryu (accessed on 12 March 2019).
57. Thomas, S.; Gayathri, I.K.; Raj, A. Joint design of Dijkstra's shortest path routing and sleep-wake scheduling in wireless sensor networks. In Proceedings of the 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 1–2 August 2017; pp. 981–986.