IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 3, 2018, accepted October 16, 2018, date of publication October 23, 2018,
date of current version November 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2877609

Reaching Agreement in an Integrated Fog Cloud IoT

SHU-CHING WANG[1], SHIH-CHI TSENG[1], KUO-QIN YAN[2], AND YAO-TE TSAI[3]
[1]Department of Information Management, Chaoyang University of Technology, Taichung 413, Taiwan
[2]Department of Business Administration, Chaoyang University of Technology, Taichung 413, Taiwan
[3]Department of International Business, Feng Chia University, Taichung 407, Taiwan

Corresponding authors: Kuo-Qin Yan (kqyan@cyut.edu.tw) and Yao-Te Tsai (yaottsai@fcu.edu.tw)

This work was supported by the Ministry of Science and Technology under Grant MOST 107-2221-E-324-005-MY3.

ABSTRACT The Internet of Things (IoT) is a technology paradigm that provides a global network of services through a wide variety of smart devices. In order to provide a high flexible and reliable platform of IoT, an IoT platform that integrated fog and cloud computing (IFCIoT) is considered in this paper. In the IFCIoT platform, fault tolerance is an important issue. In order to deal with the impact of a failed component before performing certain special tasks, it is worth paying attention to reach a common agreement in the event of a failure. However, most previous protocols for the agreement problem of distributed computing are not suitable for the IFCIoT platform. The protocol proposed by this paper can achieve agreement among all fault-free nodes with the minimal rounds of message exchanges and tolerate the maximum number of dormant and malicious faulty components in the IFCIoT platform. The theoretical proof of the complexity and the correctness is illustrated.

INDEX TERMS Internet of things, fog computing, cloud computing, agreement problem.

I. INTRODUCTION

The IoT is often characterized by many small smart things with limited storage and processing capabilities, as well as issues related to reliability, performance, security, and privacy [3]. Because Cloud computing has almost unlimited capabilities in terms of storage and processing power, hence it can solve some of the problems encountered in the IoT. Therefore, the IT paradigm that combines the two technologies of Cloud and IoT can provide current and future Internet. When connected devices can communicate with each other and integrate with vendor-managed inventory systems, customer support systems, business intelligence applications and business analytics, the true value of the enterprise IoT can be fully realized [9].

Cloud computing is an Internet-based computing paradigm that provides shared resources and on-demand access. Although the Cloud computing paradigm can handle large amounts of data generated by IoT applications, the transmission of large amounts of data has become a challenge for Cloud computing due to limited bandwidth. Therefore, data must be considered to be processed near the data source, and Fog computing can provide a possible solution to this problem.

The Fog computing can be used to process data near the data source. Fog computing moves applications, services, data, computing power, and decision making from centralized nodes to the logical extremes of the network. Fog computing significantly decreases the data volume that must be moved between end devices and the Cloud, and it enables data analytics and knowledge generation to be occurred at the data source. In addition, compared with the Cloud, Fog has a dense geographical distribution, which helps to obtain better positioning accuracy [19].

In order to provide a high flexible and reliable platform of IoT, an integrated Fog Cloud IoT (IFCIoT) was proposed by Munir *et al.* [11]. IFCIoT can harness the benefits of the IoT, Fog, and Cloud computing in a unified archetype. In other words, the IFCIoT platform can provide higher performance, higher energy efficiency, faster response time, scalability, and better localization accuracy for the future applications of IoT.

In order to ensure the reliability of the IoT, a mechanism that allows a set of nodes to reach the agreed value must be constructed [20]. Such a unanimity problem was called *agreement problem* [8]. The agreement problem is defined as all fault-free nodes must reach agreement when certain components may fail in a distributed environment. Namely, the goal of agreement is to achieve a common value for the fault-free nodes. There are three kinds of agreement issues, the *Byzantine agreement* [1], [2], [4], [6], [14], [15], [18], *consensus* [10], [17] and *interaction consistency* (*IC*) [5], [13], [16].

In this study, the *consensus* and *IC* problems of IFCIoT will be explored.

The *consensus* problem was defined by Meyer and Pradhan [10]. Each node selects an initial value to start and communicates with each other by exchanging messages. The solution to the consensus problem is defined as a protocol is proposed to meet following conditions [10]:

Consensus: All fault-free nodes agree on a common value.

Validity: If the initial value of each fault-free node i is v_i then all fault-free nodes should agree on the value v_i.

A closely related sub-problem, the *interactive consistency problem* (IC problem) has been studied extensively [5]. The definition of IC problem is to make the fault-free nodes reach interactive consistency. As with the consensus problem, each node chooses an initial value and exchanges with the others. In the interactive consistency, each node i has its initial value v_i and agrees on a set of common values. Therefore, interactive consistency is achieved if the following conditions are met [5]:

Consistency: Each fault-free node agrees on a set of common values $V = [v_1, v_2, \ldots, v_n]$.

Validity: If the initial value of fault-free node i is v_i, then the i-th value in the common vector V should be v_i.

In addition, there are two types of symptoms of node failure, namely dormant fault and malicious fault (also called Byzantine faults) [8]. The dormant fault of a fallible node includes crashes and omissions: a crash failure occurs when a node is permanently disconnected, and an omission failure occurs when the node is temporarily fails to send or receive messages on time or at all. The behavior of malicious nodes is unpredictable and unfathomable. We propose a protocol to make all fault-free nodes reach agreement without the influence of any kind of fault scenarios. Any kind of violations and fault scenarios can be occurred at any time and any places. A node with a malicious fault can work with other faulty nodes to affect the fault-free nodes reaching the agreement value. In this study, the agreement problem for nodes in the dual failure mode that combining dormant and malicious fault is solved, in which malicious faults and dormant faults may simultaneously exist in the system. In these cases, the fault tolerance capability of the system is maximized.

In IFCIoT paradigm, there are many nodes connected to each other. Even if some components fail, the fault-free nodes need to obtain the same value and reach an agreement, so the protocol is necessary to make the system still work correctly. In this study, the agreement problem for dormant and maliciously faulty nodes in IFCIoT is reconsidered. The proposed protocol, IFCIoT Agreement Protocol (IFCAP) of IFCIoT, allows all fault-free nodes to agree on a minimum number of message exchanges and tolerate the maximum number of allowed faulty components.

The remainder of this paper is organized as follows. Section II presents the integrated Fog Cloud IoT (IFCIoT) proposed by Munir *et al.* Section III shows the proposed IFCIoT Agreement Protocol (IFCAP). An example of executing the proposed protocol IFCAP is given in Section IV. Section V is responsible for proving the correctness and complexity of our new protocol. Finally, Section VI gives conclusions of this research.

II. THE IFCIOT PARADIGM

Due to the advancement and development of various information technologies, the computations for providing various applications have become more complicated [19]. The Cloud computing environment allows users to use related applications over the Internet. The majority of Cloud computing infrastructure consists of reliable services delivered through data centers and built on servers with different levels of virtualization technologies [19]. In order to provide high quality services, distributed systems must have high stability to provide an instance of many users using a given environment. In this section, the IFCIoT used in our study is discussed.

The IFCIoT paradigm proposed by Munir *et al.* [11] is shown in Figure 1. The architecture provides federated Cloud services for IoT devices through intermediate Fog. The federated Cloud services are provided by federated Cloud, which can include multiple internal and external Cloud servers to meet business and application needs. As shown in Figure 1, the Fog layer includes a number of Fog nodes. In the Fog layer, most of the processing is handled by the Fog node. In the IFCIoT architecture, the entire Fog deployment can be local or distributed at the local or regional level to provide information for centralized parent systems and services. Also, each running Fog node is autonomous to ensure that the services it provides can run uninterrupted.

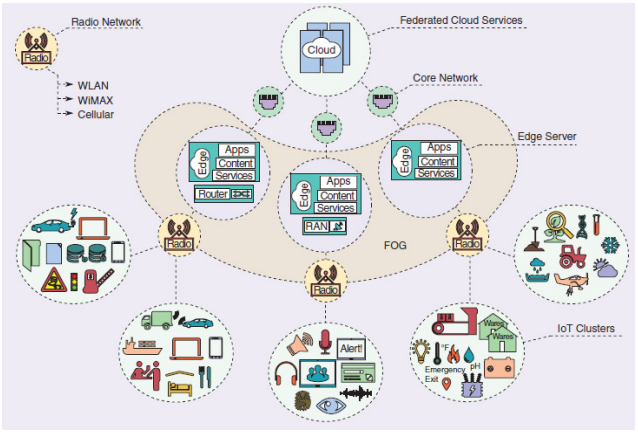


FIGURE 1. The IFCIoT architectural paradigm [11].

In this study, the IFCIoT is redefined and shown in Figure 2. There are three layers in the IFCIoT: *Sensor layer*, *Fog layer* and *Cloud layer*. The Sensor layer is constructed by IoT clusters; each IoT cluster is consisted by sensor nodes, which is responsible for sensing the data required by the IoT application. The Fog layer is constructed by a set of Fog groups; each Fog group consists of a large number of Fog nodes that process specific information and services. The Cloud layer is made up of many Cloud nodes, which provide Cloud users' services.

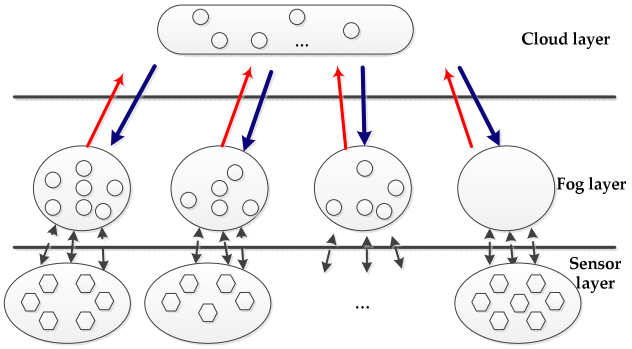


FIGURE 2. The redefined IFCIoT.

In the IoT environment, through the combination of a large number of sensors, various types of sensing data in real life can be collected. These huge sensing data from all over are used, and then a wide range of application services can be provided. For example, IFCIoT can be used in the disaster monitoring systems. At IFCIoT, the sensing data of the sensors in different IoT cluster are sent to the corresponding Fog groups in the Fog layer, and the data are processed by the Fog nodes in the specific Fog group. The related monitoring information of different IoT cluster is collected by each Fog group, and then the collected information is analyzed in each Fog group. Finally, the status of the area being monitored is then transmitted to the disaster prevention center at the Cloud layer, so that government decision-making can be provided.

In short, IFCIoT can analyze and process data through Fog nodes instead of focusing on Cloud computing. By coordinating and managing the computing and storage resources at the edge of the network, the growing demand for connected devices and the IoT can be met through Fog computing. According to the above features, the IFCIoT can be made as an appropriate platform for providing the critical services and applications of IoT, including smart life, smart industry, ... and so on [9].

In this study, the consensus problem will be addressed at the Fog layer, and IC problem will be addressed at the Cloud layer of IFCIoT. The following assumptions have been made in this study:

1. Each node in the IFCIoT can be uniquely identified.
2. The faulty node with IFCIoT is in dual failure mode.
3. When a node performs a message transfer, all messages will be encoded with the Manchester code [7]. Therefore, the dormant/omission failure node can be detected.

III. THE PROPOSED PROTOCOL

In this study, the agreement problem is discussed in the IFCIoT, and there is no delay in the inclusion of nodes or transmission media in our study. Therefore, the nodes in the IFCIoT that execute the proposed protocol IFCAP should receive messages from other nodes within a predictable period of time. If the node fails to receive the message in time, the message must be affected by the failed component.

In this study, IFCAP is used to solve the agreement problem of simultaneous presence of dormant and malicious faulty nodes in IFCIoT. Among them, the sensor node of the Sensor layer is used to sense the data required by a specific IoT application. Then, the majority of the sensing data can be represented as the value sensed by the IoT cluster. However, when the number of faulty nodes in an IoT cluster exceeds half, the representative value of the IoT cluster cannot be obtained. Therefore, $n_{Rj} > \lfloor (n_{Rj}\text{-}1)/2 \rfloor + f_{mRj} + f_{dRj}$ where n_{Rj} is the number of sensor nodes, f_{mRj} is the total number of allowable malicious faulty sensor nodes, f_{dRj} is the total number of allowable dormant faulty sensor nodes in IoT cluster R_j of sensor layer. The condition, $n_{Rj} > \lfloor (n_{Rj} - 1)/2 \rfloor + f_{mRj} + f_{dRj}$, is used to describe the number of sensor nodes required in IoT cluster R_j of Sensor layer.

In IFCAP, the *Consensus* is applied to the Fog nodes of Fog layer and the *Interactive Consistency* is applied to each Cloud node in Cloud layer. According to previous studies [8], the number of faulty components allowed in an agreement problem is determined by the total number of nodes. In Lamport *et al.*'s protocol [8], the constraints is $n > 3f_m$ where n is the number of nodes and f_m is the total number of allowable malicious faulty nodes in the distributed system. In the protocol proposed by Wang and Wang [17], the constraints is $n > \lfloor (n-1)/3 \rfloor + 2f_m + f_d$ where f_m is the total number of allowable malicious faulty nodes and f_d is the total number of allowable dormant faulty nodes in the distributed system. Therefore, the constraint of Fog layer is $n_{Fj} > \lfloor (n_{Fj} - 1)/3 \rfloor + 2f_{mFj} + f_{dFj}$ where n_{Fj} is the number of Fog nodes, f_{mFj} is the total number of allowable malicious faulty Fog nodes and f_{dFj} is the total number of allowable dormant faulty Fog nodes in Fog group F_j of Fog layer. The condition, $n_{Fj} > \lfloor (n_{Fj} - 1)/3 \rfloor + 2f_{mFj} + f_{dFj}$, is used to describe the number of Fog nodes required in Fog group F_j of Fog layer.

For the same reason, the constraint of Cloud layer is $n_C > \lfloor (n_C - 1)/3 \rfloor + 2f_{mC} + f_{dC}$ where n_C is the number of Cloud nodes, f_{mC} is the total number of allowable malicious faulty Cloud nodes and f_{dC} is the total number of allowable dormant faulty Cloud nodes in Cloud layer. This condition, $n_C > \lfloor (n_C - 1)/3 \rfloor + 2f_{mC} + f_{dC}$, is used to describe the number of Cloud nodes required in Cloud layer.

Through the implementation of IFCAP, the agreement problem of dual faulty nodes in IFCIoT can be solved. Based on the IFCIoT, IFCAP allows each node to transmit messages to other nodes without being affected by the faulty nodes. The execution of the IFCAP is initiated by the sensor nodes of the Sensor layer to obtain information about the specific application service. The nodes of Fog layer need to execute *Consensus Process* and the nodes of Cloud layer need to execute *Interactive Consistency Process*. In the *Consensus Process* and *Interactive Consistency Process*, the function *Agreement* will be called. There are two phases of function *Agreement*, one is the *Msget Phase*, and the other is *Demk Phase*. The parameters of *Agreement* include σ, v_s, and n_A, where σ is the required rounds, v_s is the initial

value and n_A is the number of nodes participating in the agreement.

Fischer and Lynch [5] and Wang and Wang [17] proved that $\lfloor (n-1)/3 \rfloor + 1$ is the necessary and sufficient rounds of message exchanges to solve an agreement problem, where n is the number of nodes in the underlying network. Based on the works of Fischer and Lynch [5] and Wang and Wang [17], $\lfloor (n-1)/3 \rfloor + 1$ rounds of message exchanges are the lower bound for solving the agreement problem. Therefore, the required rounds σ is $\lfloor (n_{Fj}-1)/3 \rfloor + 1$ when Fog nodes execute the function *Agreement* of *Consensus Process*, where n_{Fj} is the number of nodes in Fog group F_j of Fog layer and $n_{Fj} > 3$. And, the required rounds σ is $\lfloor (n_C-1)/3 \rfloor + 1$ when Cloud nodes execute the function *Agreement* of *Interactive Consistency Process*, where n_C is the number of nodes in Cloud layer and $n_C > 3$.

In the function *Agreement*, the received messages of *Msget Phase* are stored in a hierarchy structure called the hierarchy tree (h-tree). The h-tree is used by each node to store messages received by other nodes in the *Msget Phase*. (See Appendix A for a detailed description of h-tree.) The h-tree will be maintained during the execution of the IFCAP for each fault-free node. In the first round of *Msget Phase*, the initial value of node i is transmitted to other nodes. According to the assumption of this study, the sender node of the message can be identified by the receiver node. Then, after the first round of *Msget Phase* ($\sigma > 1$), each node sends the value at level $\sigma - 1$ in its own h-tree to all other nodes. After completing the σ rounds of message exchanges, each node will perform the *Demk phase*.

To mitigate the effects of maliciously faulty nodes, the node name does not repeat at any vertex of the h-tree. Therefore, each fault-free node must reorganize the h-tree using the following reorganization rules:

- The leaves in level σ of the h-tree will be deleted.
- The vertices with duplicate node names will be deleted.

Subsequently, the function *VOTE*(α) will be used by all fault-free nodes to eliminate the effects of faulty nodes and obtain common value. Among them, the function *VOTE* only calculates the non-value "α" of all the vertices of the α-th level of the h-tree (excluding the last level of the h-tree), where $1 \leq \alpha \leq \sigma$. Since *VOTE*($\alpha$) is a common value, the impact of faulty nodes will be removed and each fault-free node can reach an agreed value.

In order for all fault-free nodes to agree, each node must collect enough exchange messages from all other nodes. The value received by the exchange can help the fault-free node collect enough exchange messages. The execution steps of IFCAP are as follows:

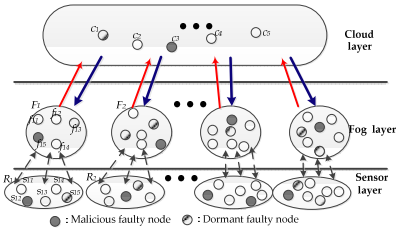
Step 1: The nodes of the Sensor layer execute the *Data Gathering Process*.
Step 1.1: The sensor node senses the related information for the specific application service in a particular region.
Step 1.2: The related information for the specific application service is transferred to the corresponding Fog group of Fog layer.
Step 2: The nodes of Fog layer execute *Consensus Process*.
Step 2.1: The node f_{ij} receives the sensing information transferred from sensor nodes in the IoT cluster R_j of Sensor layer.
Step 2.2: The Fog node takes the majority value of the sensing data received from Sensors layer, and the majority value is used as the initial value (v_i) of Fog node f_{ij} to execute function *Agreement*.
Step 2.3: The required rounds σ ($= \lfloor (n_{Fj}-1)/3 \rfloor + 1$) are calculated, where n_{Fj} is the number of Fog nodes in Fog group F_j of Fog layer. Execute *Agreement*(σ,v_i, n_{Fj}). Then the agreement vector of IoT cluster R_j is obtained.
Step 2.4: Take the majority value of the agreement vector, and then the Consensus value is gotten.
Step 2.5: The Consensus value is transferred to Cloud layer.
Step 3: The nodes of Cloud layer execute *Interactive Consistency Process.*
Step 3.1: The node c_j receives the Consensus values transferred from nodes in the Fog group F_j of Fog layer.
Step 3.2: The received Consensus values from nodes in the Fog group F_j of Fog layer are taken as the majority. Moreover, the majority value is used as the initial value (v_j) of c_j when function *Agreement* is executed.
Step 3.3: The required rounds σ ($= \lfloor (n_C-1)/3 \rfloor + 1$) are calculated, where n_C is the number of Cloud nodes in Cloud layer. Execute *Agreement*(σ,v_j, n_C), then the agreement vector is obtained.
Step3.4: The obtained vector is IC vector value.

The detail of IFCAP is shown in Appendix B.

IV. AN EXAMPLE OF EXECUTING IFCAP

Taking the disaster monitoring systems constructed by IFCIoT as an example to execute IFCAP is presented in Figure 3. In this example, in Region R_1 of Sensor layer, the sensor node s_{12} is assumed in malicious fault and s_{15} is assumed in dormant fault; in Fog group F_1 of Fog layer, Fog node f_{15} is assumed in malicious fault and f_{13} is assumed in dormant fault; Cloud node c_3 is a malicious faulty node and c_1 is a dormant faulty node in Cloud layer. In IFCAP, the transmitted message is encoded using the fault detection code [7], so the receiver can always detect messages routed through the dormant failed node. Therefore, messages routed through the dormant faulty node can be detected and replaced with λ in the received message. In the meantime, the behavior of malicious faulty nodes is unpredictable, arbitrary and undetectable.

When IFCAP is executed, in *Data Gathering Process*, each sensor node in the Sensor layer senses the monitoring status.



Sensor layer

s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
1	0	1	1	λ

(a)

Fog layer

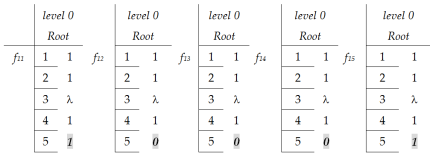
Majority(1,0,1,1,λ)=1

f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
1	1	λ	1	0

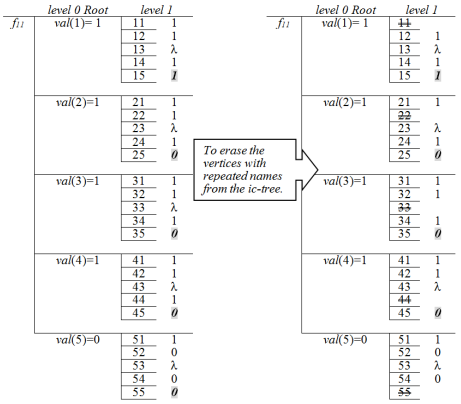
$\sigma = \lfloor (n_{F_1}-1)/3 \rfloor + 1 = \lfloor (5-1)/3 \rfloor + 1 = 2$

Execute *Agreement*(σ, v_i, n_{F_1})=(2,1,5)

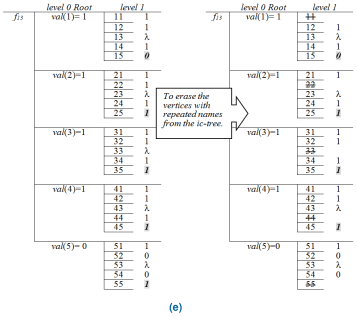
(b)



(c)



(d)



(e)

VOTE(1)= majority(*val*(12),*val*(13),*val*(14),*val*(15))= majority(1, 1, 1, 1)=
VOTE(2)= majority(*val*(21),*val*(23),*val*(24),*val*(25))= majority(1, 1, 1, 0)=
VOTE(3)= majority(*val*(31),*val*(32),*val*(34),*val*(35))= majority(1, 1, 1, 0)=
VOTE(4)= majority(*val*(41),*val*(42),*val*(43),*val*(45))= majority(1, 1, 1, 0)=
VOTE(5)= majority(*val*(51),*val*(52),*val*(53),*val*(54))= majority(1, 0, 0, 0)=
Consensus *val*ue of f_{11} =(1,1,1,1,0)=1

VOTE(1)= majority(*val*(12),*val*(13),*val*(14),*val*(15))= majority(1, 1, 1, 0)=
VOTE(2)= majority(*val*(21),*val*(23),*val*(24),*val*(25))= majority(1, 1, 1, 1)=
VOTE(3)= majority(*val*(31),*val*(32),*val*(34),*val*(35))= majority(1, 1, 1, 1)=
VOTE(4)= majority(*val*(41),*val*(42),*val*(43),*val*(45))= majority(1, 1, 1, 1)=
VOTE(5)= majority(*val*(51),*val*(52),*val*(53),*val*(54))= majority(1, 0, 0, 0)=
Consensus *val*ue of f_{12} =(1,1,1,1,0)=1

(f)

Cloud Layer

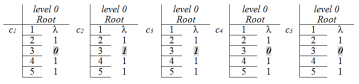
Majority(1,1, λ,1, 0)=1

c_1	c_2	c_3	c_4	c_5
λ	1	0	1	1

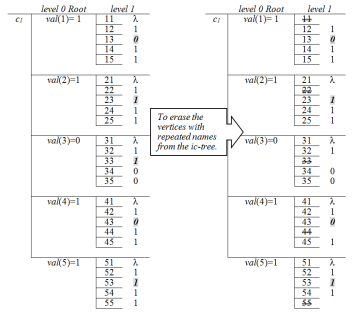
$\sigma = \lfloor (n_C-1)/3 \rfloor + 1 = \lfloor (5-1)/3 \rfloor + 1 = 2$

Execute *Agreement*(σ, v_i, n_C)=(2,1,5)

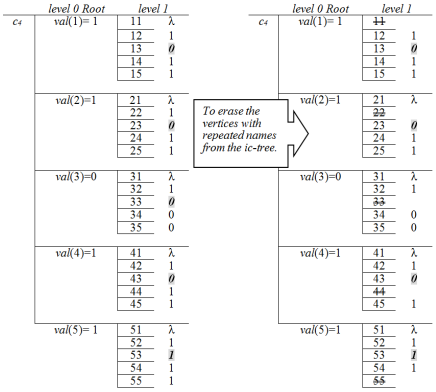
(g)



(h)



(i)



(j)

VOTE(1)= majority(*val*(12),*val*(13),*val*(14),*val*(15))= majority(1, 0, 1, 1)=
VOTE(2)= majority(*val*(21),*val*(23),*val*(24),*val*(25))= majority(1, 1, 1, 1)=
VOTE(3)= majority(*val*(31),*val*(32),*val*(34),*val*(35))= majority(0, 1, 0, 0)=
VOTE(4)= majority(*val*(41),*val*(42),*val*(43),*val*(45))= majority(1, 1, 0, 1)=
VOTE(5)= majority(*val*(51),*val*(52),*val*(53),*val*(54))= majority(1, 1, 1, 1)=
IC_{c2}=(1,1,0,1, 1)

VOTE(1)= majority(*val*(12),*val*(13),*val*(14),*val*(15))= majority(1, 0, 1, 1)=
VOTE(2)= majority(*val*(21),*val*(23),*val*(24),*val*(25))= majority(1, 0, 1, 1)=
VOTE(3)= majority(*val*(31),*val*(32),*val*(34),*val*(35))= majority(0, 1, 0, 0)=
VOTE(4)= majority(*val*(41),*val*(42),*val*(43),*val*(45))= majority(1, 1, 0, 1)=
VOTE(5)= majority(*val*(51),*val*(52),*val*(53),*val*(54))= majority(1, 1, 1, 1)=
IC_{c4}=(1,1,0,1,1)

(k)

FIGURE 3. The sensing data of each node in the IoT cluster R_1 of Sensor layer. (b). The initial value of each node in Fog group F_1 of Fog layer. (c). The h-tree of each node in Fog group F_1 of Fog layer at the first round of *Msget* Phase. (d). The final h-tree of f_{11} after the *Msget* Phase and the h-tree of f_{11} by *Demk* Phase. (e). The final h-tree of f_{12} after the *Msget* Phase and the h-tree of f_{12} by *Demk* Phase. (f). The common value *VOTE*(*i*) by in *Demk* Phase of Fog layer. (g). The initial value of each node in Cloud layer. (h). The h-tree of each node in Cloud layer at the first round of *Msget* Phase (i). The final h-tree of c_2 after the *Msget* Phase and the h-tree of c_2 by *Demk* Phase. (j). The final h-tree of c_4 after the *Msget* Phase and the h-tree of c_4 by *Demk* Phase. (k). The common value *VOTE*(*i*) in *Demk* Phase of Cloud layer.

The sensing data of each node in the IoT cluster R_1 of Sensor layer is shown in Figure 3(a). In *Consensus Process*, each Fog node in Fog group F_1 receives the sensing monitoring status transferred from sensor nodes in the Region R_1. The received monitoring status are taken as the majority and the majority value is used as the initial value (v_i) of Fog node in Fog group F_1 when function *Agreement* is executed. Since node s_{12} is a malicious faulty node, it is assumed that the transmitted monitoring status is unpredictable. And, the messages pass through the dormant faulty node s_{15}, then the received message transmitted by s_{15} is replaced by λ. However, the total number of failed nodes cannot exceed half of the total number of nodes in Region R_1, the majority value obtained is still the correct values. Then, the number of rounds required, $\sigma = \lfloor (n_{Fj} - 1)/3 \rfloor + 1$, is computed and *Agreement*(σ, v_i, n_{Fj}) is executed. The initial value of each node in Fog group F_1 of Fog layer is shown in Figure 3(b).

In this example, there are five nodes in Fog group F_1, Fog node f_{15} is a malicious faulty node and f_{13} is a dormant faulty node. Therefore, 2 rounds ($\sigma = \lfloor (5 - 1)/3 \rfloor + 1 = 2$) are required to exchange the messages when *Agreement* is executed. Figure 3(b) is the initial value of each node in Fog group F_1. During the first round of *Msget Phase*, the initial value of each node of Fog group F_1 is sent to all nodes of Fog group F_1 and stores the received n_{F1} (=5) values in the corresponding root of each h-tree, as shown in Figure 3(c). In the second round, each node transmits the values in the root of the corresponding h-tree to other nodes in Fog group F_1 and stores the received values in level 1 of the n_{F1} (=5) corresponding h-tree. Subsequently, in the *Demk Phase*, every fault-free node reorganizes the h-tree by removing those vertices with duplicate node names. Figures 3(d) and 3(e) are the corresponding h-trees of nodes f_{11} and f_{12}. The function *VOTE* is then applied to the h-tree root of each node to obtain common value. Get the Consensus value of the agreement by taking majority values for the common value obtained by each node. The Consensus value of nodes f_{11} and f_{12} is obtained and shown in Figure 3(f). Finally, the Consensus value of each Fog group in the Fog layer is transferred to Cloud layer.

In the *Interactive Consistency Process*, the Cloud node in the Cloud layer receives the Consensus value sent by Fog nodes in the Fog group of Fog layer. The received Consensus values are taken as the majority. In addition, the majority value is used as the initial value of Cloud node when function *Agreement* is executed. The initial value of each node in Cloud layer is shown in Figure 3(g).

In this example, there are five nodes in Cloud layer, since Cloud node c_3 is a malicious faulty node and c_1 is a dormant faulty node. There are 2 rounds ($\lfloor (5 - 1)/3 \rfloor + 1 = 2$) are required to execute *Agreement*. Figure 3(h) is the initial value of each Cloud node in Cloud layer. During the first round of *Msget Phase*, the initial value of each node of Cloud layer is transmitted to all nodes of Cloud layer and stores the received n_C (=5) values in the corresponding root of each h-tree, as shown in Figure 3(i). In the second round, each node

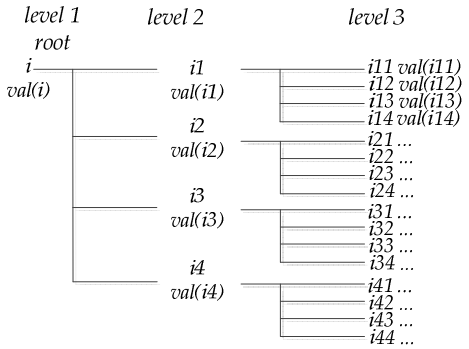


FIGURE 4. An example of h-tree.

transmits the values in the root of the corresponding h-tree to other nodes in Cloud computing layer and stores the received values in level 1 of the n_C (=5) corresponding h-trees. Subsequently, in the *Demk Phase*, the h-tree is reorganized and the reorganized h-tree of nodes c_2 and c_4 is shown in Figures 4(i) and 4(j). The function *VOTE* is then applied to the h-tree root of each Cloud node to obtain common value.

The common value obtained through function *Agreement* represents the state of a particular application. The IC value is a vector, and each element in the vector is the majority value obtained through *Agreement* function. Each element is used to present the status of a specific application. The IC vector value of nodes c_2 and c_4 is shown in Figure 3(k). Eventually, the agreement is reached in IFCIoT. Finally, the service of disaster monitoring can be supported by each Cloud node in Cloud layer.

V. THE CORRECTNESS AND COMPLEXITY OF IFCAP

The correctness and complexity of IFCAP will be proven in this section. In order to verify that IFCAP can resolve the agreement problem, the correctness of IFCAP will be reviewed in accordance with the following two terms:

Fault-free

1. vertex: If node i is fault-free, then vertex αi of h-tree is a fault-free vertex.
2. Common value: $val(\alpha i)$ is the common value of vertex αi for a fault-free vertex αi in the h-tree.

Since the fault-free node always sends the same value to all nodes, hence, the fault-free vertices of such h-trees are common.

Lemma 1: Fault-free nodes can detect the message(s) sent by dormant faulty nodes.

Proof: The IFCAP encodes the message before the message is transmitted in Manchester code [12], [17], so messages sent from the dormant faulty node can be detected by the fault-free node.

Theorem 1: The fault-free node can receive messages from the transmitting node without being affected by any faulty nodes between the sending node and the receiving node if $n > \lfloor (n - 1)/3 \rfloor + 2f_m + f_d$ *where n is the number of nodes in a distributed network,* f_m *is the total number of allowable malicious faulty nodes and* f_d *is the total number of allowable dormant faulty nodes in a network.*

Proof: Through Lemma 1, the effects of dormant faulty nodes can be eliminated in each round of message exchange. In addition, the impact of malicious faulty nodes in each round of message exchange can also be ruled out if $n > \lfloor (n-1)/3 \rfloor + 2f_m + f_d$. This is because the fault-free sending node sent $(n-1)$ copies of the message to all other node. Thus, in the worst case, the fault-free receiving node receives the $(n-1) - f_d$ message sent by the fault-free sending node, where f_d is the number of messages that can be detected from the dormant faulty node. At the same time, since $(n-1) - f_d > 2f_m$, the fault-free receiving node can determine the fault-free messages by taking majority values.

Lemma 2: The dormant faulty sending node can be detected by a fault-free node.

Proof: Since there are at most $\lfloor (n-1)/3 \rfloor$ malicious faulty nodes in the network, if the value of λ is greater than or equal to $(n-1) - \lfloor (n-1)/3 \rfloor$, the sending node is the node that dormant faulty; therefore, there are at most $\lfloor (n-1)/3 \rfloor$ non-λ values.

Theorem 2: The dormant faulty nodes in the network can be detected by fault-free node.

Proof: According to the research results of Fischer and Lynch [5] and Wang and Wang [17], the number of required rounds of message exchanges in IFCAP is $\sigma = \lfloor (n-1)/3 \rfloor + 1$ where n>3. Thus, there are at least two rounds of message exchanges during the *Msget Phase*. Each of the fault-free nodes receives the initial value of other nodes during the first round of message exchange and receives the owned messages of the other nodes during the second round of message exchange. Thus, according to Lemma 2, the dormant faulty nodes in the network can be detected by fault-free node.

Lemma 3: All the fault-free vertices in h-tree are common.

Proof: In the h-tree, the fault-free vertex α has at least $2\lfloor (n-1)/3 \rfloor + 1$ children at the level $\lfloor (n-1)/3 \rfloor + 1$ or higher, and at least $\lfloor (n-1)/3 \rfloor + 1$ of which have fault-free. The common value of these $\lfloor (n-1)/3 \rfloor + 1$ fault-free vertices is common, and the majority value of the vertex α is common. If the level of α is less than $\lfloor (n-1)/3 \rfloor + 1$, then the fault-free vertex α is common in the h-tree. Therefore, all fault-free vertices of the h-tree are common.

Theorem 3: The root of the h-tree of a fault-free node is common.

Proof: According to Lemma 3, the theorem can be proved.

Theorem 4: IFCAP solves the agreement problem in the IFCIoT.

Proof: To prove the theorem, (A_1) and (A_2) should be proved.

(A_1): Root of the h-tree is common. By Theorem 3, $(\boldsymbol{A}_1)$ is satisfied.

(A_2): VOTE$(i) = v$ for all fault-free nodes, if the initial value of the node is v_i, say $v = v_i$. Since most nodes are fault-free, they transmit messages to all other nodes. The value of fault-free vertices for the h-tree of all fault-free nodes is v. As a result, each fault-free vertex of the h-tree is common, and its common value is v. Using Theorem 3, this root is common. The computed value VOTE$(i) = v$ is stored in the root of the h-tree for all fault-free nodes. $(\boldsymbol{A}_2)$ is satisfied.

Theorem 5: The number of required rounds of message exchanges by IFCAP is the minimum.

Proof: The total number of required rounds of message exchanges by IFCAP can be discussed by three layer of IFCIoT.

(1) **Sensor layer**: In Sensor layer, each sensor passes the received sensing data to Fog layer. Therefore, only one round of message exchange is required.
(2) **Fog layer**: All nodes need to exchange messages during the *Msget Phase*, so it is very time consuming in this phase. Fischer and Lynch [5] and Wang and Wang [17] pointed out that in a distributed system consisting of n nodes, the $\lfloor (n-1)/3 \rfloor + 1$ rounds are the minimum number of rounds to send enough messages to reach an agreement. However, in the fallible Fog layer, the node may be in a dormant or malicious faulty state. In addition, each node in the fallible Fog layer must still exchange messages with other nodes. Therefore, the minimum number of rounds indicated by Fischer and Lynch [5] and Wang and Wang [17] can be applied to Fog layer. In other words, in Fog layer, there are n_{Fj} nodes in Fog group F_j of Fog layer, IFCAP needs $\lfloor (n_{Fj} - 1)/3 \rfloor + 1$ rounds to exchange messages. In an F-groups Fog layer, the nodes in each Fog group execute IFCAP parallel, where F is the total number of groups in the Fog layer of IFCIoT. Therefore, the required rounds of executing IFCAP by each node in all Fog groups are depended on the number of nodes in Fog group.
(3) **Cloud layer**: As in the discussion of the number of message exchanges required in the Fog layer. In the Cloud layer, the research results of Fischer and Lynch [5] and Wang and Wang [17] can still be applied. In Cloud layer, there are n_C nodes in Cloud layer, IFCAP needs $\lfloor (n_C - 1)/3 \rfloor + 1$ rounds to exchange messages.

In short, according to the discussion of the number of rounds required by IFCAP in the three layers of IFCIoT separately, the total number of required rounds of IFCAP is minimal.

Theorem 6: The number of allowable faulty nodes by IFCAP is the maximum.

Proof: The total number of allowable faulty nodes by IFCAP can be discussed by three layer of IFCIoT.

(1) **Sensor layer**: Since the number of faulty nodes in each IoT cluster of Sensor layer does not exceed half, and the majority value of the IoT cluster can be determined. Therefore, $n_{Rj} > \lfloor (n_{Rj} - 1)/2 \rfloor + f_{mRj} + f_{dRj}$ where n_{Rj} is the number of sensor nodes, f_{mRj} is the total number of allowable malicious faulty sensor nodes, and f_{dRj} is the total number of allowable dormant faulty sensor nodes in IoT cluster R_j of Sensor layer. T_{FS} is the total number of allowable faulty nodes in Sensor layer, $T_{FS} = \sum_{j=1}^{R} f_{Rj}$ where R is the total number of IoT clusters in Sensor

layer, f_{Rj} is the total number of allowable faulty sensor nodes in IoT cluster R_j and $f_{Rj=}f_{mRj}+f_{dRj}$.

(2) **Fog layer**: In the protocol proposed by Wang and Wang [17], the constraints is $n > \lfloor(n-1)/3\rfloor+2f_m+f_d$ where n is the number of nodes in a distributed network, f_m is the total number of allowable malicious faulty nodes and f_d is the total number of allowable dormant faulty nodes in the distributed system. In this study, it is assumed that the fault state of the node is also a dual failure mode. However, the fault status of our assumption is also that nodes are dual failure mode. Therefore, the research result of Wang and Wang [17] can be applied to $n_{Fj} > \lfloor(n_{Fj}-1)/3\rfloor+2f_{mFj}+f_{dFj}$ where n_{Fj} is the number of Fog nodes, f_{mFj} is the total number of allowable malicious faulty Fog nodes and f_{dFj} is the total number of allowable dormant faulty Fog nodes in Fog group F_j in the Fog layer. Then, $T_{FF} = \sum_{j=1}^{F} f_{Fj}$ where F is the total number of Fog groups in the Fog layer of IFCIoT, and $f_{Fj=}f_{mFj}+f_{dFj}$. T_{FF} is the total number of allowable faulty nodes in Fog layer.

(3) **Cloud layer**: The research result of Wang and Wang [17] also can be applied to Cloud layer. Therefore, $n_C > \lfloor(n_C-1)/3\rfloor+2f_{mC}+f_{dC}$ where n_C is the number of Cloud nodes, f_{mC} is the total number of allowable malicious faulty Cloud nodes and f_{dC} is the total number of allowable dormant faulty Cloud nodes in Cloud layer.

It can be obtained through the proofs of Theorems 5 and 6, that IFCAP requires the minimum number of message exchanges and tolerates the maximum number of dormant and malicious faulty nodes so that the fault-free nodes can reach a common agreement. Therefore, the optimality of the protocol is proven.

VI. CONCLUSIONS

In this study, a high flexible and reliable IoT platform is used that integrated Fog computing and Cloud computing (IFCIoT). By using IFCIoT, the disaster monitoring systems and other application systems can be constructed. The agreement problem is one of the most important topics in distributed systems and has been extensively studied. Among them, the network topology is an important factor in discussing agreement problem. However, IFCIoT is the new architecture for IoT applications. It provides distributed system design and practice to support user-oriented services. In this study, the IFCAP protocol is proposed to make all fault-free nodes to achieve agreement. Moreover, all fault-free nodes can perform subsequent related applications and services with the agreement values. The proposed protocol IFCAP can use a minimum number of message exchanges and tolerate the maximum number of malicious and dormant faulty nodes allowed in a fallible IFCIoT.

In previous studies, the agreement protocols were designed in various network topologies [5], [6], [8], [12], [14], [15], [17]. A comparison of the studies among the most previous relative researches is given in Table 1. Those works reach agreement underlying different topologies respectively, including Broadcasting Network (BCN), Fully Connected Network (FCN), Multicasting Network (MCN), Wireless Sensor Network (WSN), Cloud Computing environment (CC) and WSN based IoT (WIoT). All those previous protocols are not suitable for IFCIoT due to the difference of network topology. To enhance fault-tolerance of IFCIoT, it is the first time a protocol IFCAP is proposed in this study to solve the agreement problem. The proposed protocol ensures that all fault-free nodes in IFCIoT can reach an agreement to cope with the influences of the faulty components by using the minimum number of message exchanges, while tolerating the maximum number of faulty components at any time.

TABLE 1. The comparisons among previous approaches and the proposed protocol IFCAP.

Topology / Results	BCN	FCN	MCN	WSN	CC	WIoT	IFCIoT
Babaoglu et al. [1]	V						
Fischer [5] Lamport et al. [8] Siu et al. [12]		V					
Wang et al. [14]			V				
Yan et al. [18]				V			
Chiang et al. [4] Wang et al. [16]					V		
Wang et al. [15]						V	
IFCAP							V

In a network topology, the fallible components are not only nodes, but also transmission media. Therefore, the IFCAP protocol will be extended in the future to achieve agreement in the IFCIoT topology regardless of nodes or transmission media have been damaged. In addition, the further considerations of pre-defined protection strategy for unreliable network, the impact of failures in communication traffic, and fog nodes mobility are premeditated in the future.

APPENDIX A
THE HIERARCHY TREE (h-TREE)

During the IFCAP is executed, the h-tree is maintained by each fault-free node. In the first round of *Msget Phase*, the initial value of node i is transmitted to other nodes. When the message sent from the node i is received, the received value is stored, denoted as $val(i)$, at the root of h-tree. In the second round, each node transmits root value of its h-tree to all other nodes. If node *1* sends message $val(i)$ to node *2*, then the received message is stored by node *2* and denoted as $val(i1)$ in vertex $i1$ of node *2's* h-tree. Similarly, if node *2* sends message $val(i1)$ to node *1*, the received message is named $val(i12)$ and stored in vertex $i12$ of node *1*'s h-tree in the third round. Generally, message $val(i12\ldots n)$, stored in the vertex $i12\ldots n$ of an h-tree, implies that the message just received was sent through the node i, the node $1, \ldots,$ the node n; and the node n is the latest nodes to pass the message. In summary, the root of h-tree is always named i to denote that the stored message is sent from the node i in the first round; and the vertex of an h-tree is labeled by a list of node names. The node name list contains the names of the nodes through which the stored message was transferred. Figure 4 shows an example of h-tree.

APPENDIX B
THE PROTOCOL IFCAP

The detail of the proposed protocol IFCAP is shown in Figure 5 as below.

IFCAP

The nodes of Sensor layer execute *Data Gathering Process.*
The nodes of Fog layer execute *Consensus Process.*
The nodes of Cloud layer execute *Interactive Consistency Process.*

Data Gathering Process (for the sensor node s_{ij} in the IoT cluster R_i of Sensor layer, $1 \le i \le n_{Ri}$ where n_{Ri} is the number of sensor nodes in IoT cluster R_i of Sensor layer)

1. The sensor node s_{ij} senses the related information for the specific application service.
2. The related information for the specific application service is transferred to the corresponding Fog group of Fog layer by sensor node s_{ij}.

Consensus Process (for the node f_{ij} in the Fog group F_i of Fog layer, $1 \le i \le n_{Fi}$ where n_{Fj} is the number of nodes in Fog group F_j of Fog layer and $n_{Fj} > 3$)

1. The node f_{ij} receives the sensing information transferred from sensor nodes in the IoT cluster R_i of Sensor layer. If the received information is transferred by a dormant sensor node, then the received information is replaced by λ.
2. The received information from sensor nodes in the IoT cluster R_i is taken as the majority. In addition, the majority value is used as the initial value (v_i) of f_{ii} when function *Agreement* is executed.
3. The required rounds σ (= $\lfloor (n_{Fi}-1)/3 \rfloor + 1$) are calculated. Execute *Agreement*(σ, v_i, n_{Fi}), then the agreement vector of IoT cluster R_i is obtained.
4. Take the majority value of the agreement vector, and then the Consensus value can be gotten.
5. The Consensus value is transferred to Cloud layer.

Interactive Consistency Process (for the node c_i in the Cloud computing layer, $1 \le j \le n_C$, where n_C is the number of nodes in Cloud computing layer and $n_C > 3$)

1. The node c_i receives the Consensus values transferred from nodes in the Fog group F_i of Fog layer. If the received Consensus value is transferred by a dormant Fog node, then the received Consensus value is replaced by λ.
2. The received Consensus values from nodes in the Fog group F_j of Fog layer are taken as the majority. Moreover, the majority value is used as the initial value (v_i) of c_i when function *Agreement* is executed.
3. The required rounds σ (= $\lfloor (n_C-1)/3 \rfloor + 1$) are calculated. Execute *Agreement*(σ, v_i, n_C), then the agreement vector is obtained.
4. The obtained vector is IC value.

Agreement(σ, v_s, n_A) (σ is the required rounds, v_s is the initial value and n_A is the number of nodes participating in the agreement)

Msget Phase:

If $r = 1$ then:
1) The initial value (v_s) of each node is broadcasted to other nodes in the same group parallel.
2) Each node receives and stores the n_A values sent by n_A nodes of same group in the respective root of its h-tree.
3) If the received value is transmitted by a dormant faulty node, then the received value is replaced with λ and stored.

For $1 < r \le$ σ, do:
1) Each node transmits the values at level $r-1$ in its h-tree to other nodes in same group parallel.
2) Each receiver node stores the received values in the corresponding vertices at level r of its h-tree.
3) If the received value is transmitted by a dormant faulty node, then the received value is replaced with λ and stored.

Demk Phase:

Step 1: The h-tree is reorganized to remove vertices with duplicate node names.
Step 2: The function *VOTE* is used at the root i of each h-tree, and the common value *VOTE*(i) is obtained.

VOTE(α)= If the α is a leaf, then outputs the value α.
If the majority value in the set of {*VOTE*(αj)|$1 \le j \le n_A$ and vertex αj is a child of vertex α} exists
then outputs the majority value
else a default value ϕ is outputted.

FIGURE 5. Protocol IFCAP.

REFERENCES

[1] O. Babaoglu and R. Drummond, "Streets of byzantium: Network architectures for fast reliable broadcasts," *IEEE Trans. Softw. Eng.*, vols. SE–11, no. 6, pp. 546–554, Jun. 1985.
[2] S. Bonomi, A. Del Pozzo, M. Potop-Butucaru, and S. Tixeuil, "Approximate agreement under mobile Byzantine faults," *Theor. Comput. Sci.*, to be published, doi: 10.1016/j.tcs.2018.08.001.
[3] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "On the integration of cloud computing and Internet of Things," in *Proc. Int. Conf. Future Internet Things Cloud*, Barcelona, Spain, Aug. 2014, pp. 23–30.
[4] M.-L. Chiang, C.-L. Chen, and H.-C. Hsieh, "An agreement under early stopping and fault diagnosis protocol in a cloud computing environment," *IEEE Access*, vol. 6, pp. 44868–44875, Sep. 2018.
[5] M. J. Fischer and N. A. Lynch, "A lower bound for the time to assure interactive consistency," *Inf. Process. Lett.*, vol. 14, pp. 183–186, Jun. 1982.
[6] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Oper. Syst. Princ.*, Shanghai, China, 2017, pp. 51–68.
[7] F. Halsall, *Data Communications, Computer Networks and Open Systems*, 4th ed. Reading, MA, USA: Addison-Wesley, 1995, pp. 112–125.
[8] L. Lamport, R. Shostak, and M. Pease, "'The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[9] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Bus. Horizons*, vol. 58, no. 4, pp. 431–440, 2015.
[10] F. J. Meyer and D. K. Pradhan, "Consensus with dual failure modes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 2, no. 2, pp. 214–222, Apr. 1991.
[11] A. Munir, P. Kansakar, and S. U. Khan, "IFCIoT: Integrated fog cloud IoT: A novel architectural paradigm for the future Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 6, no. 3, pp. 74–82, Jul. 2017.
[12] H.-S. Siu, Y.-H. Chin, and W.-P. Yang, "A note on consensus on dual failure modes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 3, pp. 225–230, Mar. 1996.
[13] P. Thambidurai and Y.-K. Park, "Interactive consistency with multiple failure modes," in *Proc. 7th Symp. Reliable Distrib. Syst.*, Oct. 1988, pp. 93–100.
[14] S. C. Wang, K. Q. Yan, and C. F. Cheng, "Efficient multicasting agreement protocol," *Comput. Standards Interfaces*, vol. 26, no. 2, pp. 93–111, Mar. 2004.
[15] S.-C. Wang, K.-Q. Yan, C.-L. Ho, and S.-S. Wang, "The optimal generalized Byzantine agreement in cluster-based wireless sensor networks," *Comput. Standards Interfaces*, vol. 36, no. 5, pp. 821–830, Sep. 2014.
[16] S.-C. Wang, S.-S. Wang, and K.-Q. Yan, "Reaching optimal interactive consistency in a fallible cloud computing environment," *J. Inf. Sci. Eng.*, vol. 34, no. 1, pp. 205–223, Jan. 2018.
[17] S.-S. Wang and S.-C. Wang, "The consensus problem with dual failure nodes in a cloud computing environment," *Inf. Sci.*, vol. 279, pp. 213–228, Sep. 2014.
[18] K. Q. Yan, S. C. Wang, C. S. Peng, and S. S. Wang, "Optimal malicious agreement protocol for cluster-based wireless sensor networks," *Scienceasia*, vol. 40, pp. 8–15, Feb. 2014.
[19] M. Yannuzzi, R. Milito, R. Serral-Gracià, and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog computing, cloud computing, and more fog computing," in *Proc. IEEE 19th Int. Workshop Comput. Aided Modeling Design Commun. Links Netw.*, Athens, Greece, Dec. 2014, pp. 325–329.
[20] Y. Zhang and M. R. Lyu, "QoS-aware Byzantine fault tolerance," *QoS Predict. Cloud Service Comput.*, pp. 105–120, Aug. 2017.

SHU-CHING WANG received the B.S. degree in computer science from Feng Chia University, the M.S. degree in electrical engineering from National Chen-Kung University, and the Ph.D. degree in information engineering from National Chiao-Tung University, Taiwan. She is currently a Professor with the Department of Information Management, Chaoyang University of Technology, Taiwan. Her current research interests include distributed computing, cloud computing, and Internet of Things.

SHIH-CHI TSENG is currently a Student in the Doctoral Program of the Department of Information Management, Chaoyang University of Technology, Taiwan. His current research interests include distributed system, fault tolerant, and cloud computing.

KUO-QIN YAN received the B.S. and M.S. degrees in electrical engineering from the Chung Cheng Institute of Technology and the Ph.D. degree in computer science from National Tsing-Hua University, Taiwan. He is currently a Professor with the Department of Business Administration, Chaoyang University of Technology, Taiwan. His current research interests include distributed data processing, parallel processing, fault-tolerant computing, mobile computing, and ubiquitous computing.

YAO-TE TSAI received the Ph.D. degree in industrial and systems engineering from Auburn University in 2015. He is currently an Assistant Professor with the Department of International Business, Feng Chia University. His research interests include Internet of Things, transportation safety, data analytics, supply chain management, and operations management.