Hindawi
Security and Communication Networks
Volume 2020, Article ID 8884924, 14 pages
https://doi.org/10.1155/2020/8884924

Research Article

Improving Topic-Based Data Exchanges among IoT Devices

Fu Chen,[1,2] **Peng Liu**,[2] **Jianming Zhu**,[1] **Sheng Gao**,[1] **Yanmei Zhang**,[1] **Meijiao Duan**,[1] **Youwei Wang**,[1] **and Kai Hwang**[3]

[1]*Department of Computer Science, Central University of Finance and Economics, Beijing102206, China*
[2]*College of Information Sciences and Technology, The Pennsylvania State University, State College 16801, PA, USA*
[3]*Chinese University of Hong Kong (CUHK), Hong Kong, China*

Correspondence should be addressed to Fu Chen; chenfu@cufe.edu.cn

Received 11 June 2020; Revised 12 September 2020; Accepted 16 October 2020; Published 31 December 2020

Academic Editor: Honghao Gao

Data exchange is one of the huge challenges in Internet of Things (IoT) with billions of heterogeneous devices already connected and many more to come in the future. Improving data transfer efficiency, scalability, and survivability in the fragile network environment and constrained resources in IoT systems is always a fundamental issues. In this paper, we present a novel message routing algorithm that optimizes IoT data transfers in a resource constrained and fragile network environment in publish-subscribe model. The proposed algorithm can adapt the dynamical network topology of continuously changing IoT devices with the rerouting method. We also present a rerouting algorithm in Message Queuing Telemetry Transport (MQTT) to take over the topic-based session flows with a controller when a broker crashed down. Data can still be communicated by another broker with rerouting mechanism. Higher availability in IoT can be achieved with our proposed model. Through demonstrated efficiency of our algorithms about message routing and dynamically adapting the continually changing device and network topology, IoT systems can gain scalability and survivability. We have evaluated our algorithms with open source Eclipse Mosquitto. With the extensive experiments and simulations performed in Mosquitto, the results show that our algorithms perform optimally. The proposed algorithms can be widely used in IoT systems with publish-subscribe model. Furthermore, the algorithms can also be adopted in other protocols such as Constrained Application Protocol (CoAP).

1. Introduction

The Internet of Things (IoT) has emerged as the next evolution of the technology which covers a wide variety of devices and system platforms, such as embedded systems, networked sensors, actuators, and smart home devices. Inevitably, the resource starved nature in most IoT devices makes it error-prone to efficiently manage and maintain a reliable IoT system, not to mention the security issues associated with some scarcely attended IoT devices [1, 2]. One of the biggest challenges in IoT is the management and monitoring of an IoT system, especially when heterogeneous devices are networked in geographically distributed environments. And IoT devices are often used to monitor industrial production, energy consumption, agriculture condition, and business operation, even in sensitive medical application. Therefore, the cross-platform and cross-application data flows among IoT devices are often extremely important. The protocols that allow machine-to-machine, device-to-device, and device-to-server communication to communicate with each other become very important in IoT system. In general, wireless network protocols have evolved in the form of WiFi, ZigBee, Bluetooth, and routing protocols from multihop to ad hoc and mobile ad hoc networks (MANET); the data protocols play an important role in IoT protocol stack, which includes Message Queuing Telemetry Transport (MQTT) [3], Constrained Application Protocol (CoAP) [4], AMQP [5], and WebSocket [6]. Comparatively MQTT seems to be more promising as seen in Figure 1 showing the trends comparison of MQTT, COMP, and AMQP, according to Google trends dataset [7]. This paper includes the communication efficiency of MQTT. All the information in the MQTT is organized with Tree structure and topics-based naming [8].

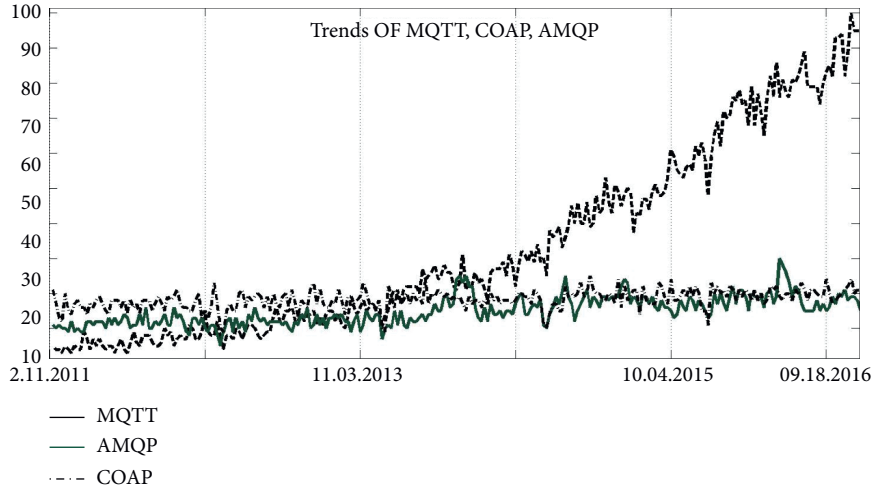

Figure 1: Trends comparison of MQTT, CoAP, and AMQP.

The history of MQTT dates back to 1999, which was invented by IBM. MQTT uses the so-called "publish/subscription" model to transmit data among the IoT things. MQTT is more mature and stable than CoAP and AMQP. Due to the low-cost and power-constrained nature of most IoT devices, the communication efficiency is extremely important. For example, if we can improve the power efficiency, the IoT device can be operated for even longer time period than low power efficiency without having to replace batteries. This is one of the fundamental motivations for our focus on IoT data communication protocol efficiency in this paper. An IoT system consists of a large number of internet-connected resource-constrained and dynamic nature devices [9]. Therefore achieving data transmission efficiency is very important in this type of distributed networks. Data transmission in many IoT systems use publish-subscribe model, as in MQTT. In AMQP protocol, clients send messages to the AMQP server with topics, and server listens for messages with those topics and routes the messages to the applications subscribed to the AMQP server [5, 10]. MQTT uses a similar approach where message brokers use topics to route messages from publishing clients to subscribing clients [11]. To exchange data between devices with publish-subscribe model, matching, filtering, and routing of messages based on topics, as the bridge of publisher and subscriber, have extreme importance in IoT system. Especially when there are plenty of messages produced by a large number of IoT devices, improving the efficiency and effect of topic matching, filtering, and routing is vital. The work in this paper aims to address this important need of data communication in IoT system. To address this important need, we propose two algorithms to improve data exchange efficiency in IoT system. The two important contributions of this paper are as follows:

(i) Development of a topic routing algorithm to communicate data between IoT devices

(ii) Development, implementation, and evaluation of a Broker Network to improve the survivability and robustness of connected-device data exchange in IoT

We evaluate our algorithms with MQTT protocol implementation Mosquitto [3, 12]. We have performed extensive experiments and results show that with our algorithms we can dramatically improve data exchange efficiency while significantly reducing transmission time. The organization of the paper is as follows. In Section 2, we present background and motivation of this work highlighting main concerns that arise in the data communication in IoT and state of the art in the field. Section 3 presents our algorithms in detail. The implementation and evaluation of our algorithms are presented in Section 4. Finally, conclusions are drawn in Section 5.

2. Background

2.1. Need of the Data Exchange Efficiency Problem in IoT. Devices in IoT system often have limited processing capabilities, low bandwidth, and small memory capacities and are powered by batteries with limited working hours. Therefore, improving the usage efficiency of the network and reducing power usage in IoT scenarios are extremely important. As a promising data communication protocol in IoT, MQTT is a machine-to-machine (M2M), lightweight publish-subscribe messaging transport protocol. In publish-subscribe model, subscriber registers its interest topics to a broker and publisher sends data to a broker [13–15]. Publish clients and subscribe clients exchange data through a broker with topic matching. When there are huge message flows exchanged by broker, issue of latency may be inevitable [16]. The latency not only decreases the speed of data transmission but also wastes device battery to decrease the overall device life-span [17–20]. In this paper, we focus on data exchange efficiency in an environment where there is a limited Internet access or there are serious network jitter problems for several hundred miles around. The emphasis is transferring data based on subject-based route to a data center. Essentially, it is about a new gateway and topic-based routing. First, we need to classify necessity of transferring data between gateways. MQTT in IoT is used in data exchange which is different from corresponding sensor network about data integration.

We depict an agriculture processing scenario to clarify the necessity for data transmission with topic-based routing. Consumers today often put significant value on fresh, high quality food. Farmers need to get information from fields, take actions to crop depending on weather conditions, and show the information about the produce to the end consumers with public access interface. This information includes identifying the drought and weather conditions, periodic measurements of monitoring data with gateway, depending on intensity of drought decide about irrigation and watering, monitoring pest disasters, and then taking actions about the spraying pesticides with drone, without distinguishing nodes from the broker's point of view, which brings difficulty due to large number of devices. If there are plenty of things in one IoT system, it is necessary to identify nodes having the corresponding information and to identify these nodes dramatically and efficiently.

If this information is configured manually, while not the automatically, it is a huge challenge to manage thousands of nodes. When the network structure is changed, the configurations need to be done. Therefore, automatic discovering and configuring the IoT network are a huge challenge. And agriculture processing system scenario has the following features: data from farm in fields to table on Internet, field data spanning up to one year, wide geographical areas of crops, and fragile network conditions.

As noted earlier, MQTT used in IoT system emphasizes data exchange, not mainly for data integration. The question arises, why in the agriculture process system we need data exchange? There are several reasons to make it necessary for data exchange in agriculture processing system and data exchange about storm, hurricanes, cyclones bad weather. It is known to all that storm may influence several hundreds of miles. It is very important to spread related information about agriculture. In fact, weather conditions have a heavy influence on the crop management. Data exchanges among different farms are very important. For example, in Pennsylvania, there may have been dozens of farms located in a region where pests and diseases spread very quickly. Similarly, farms in remote and regional areas in Australia are spread over on thousands of acres. If the information around one regain can exchange on time, precautions can be taken to prevent or reduce loss from any natural disaster which may otherwise affect drastically the agriculture processing system.

2.2. Essence of the Problem Studied in This Paper. The IoT things are spread around hundreds of miles in field or plain. According to the mechanism of MQTT protocol, data exchanges among different nodes and brokers should be configured according to the IP addresses and topic names. Only after manual configuration or by configuration files, the data can be exchanged along specific path to another node, which gets damaged crushes down, we will not know until proactive testing is performed. Neighboring nodes might have information about the damaged nodes but the important question here is how to dynamically figure out which nodes have the corresponding data and how to arrive at that node with the shortest path automatically. In our proposed algorithms, we have focused on this problem. On the contrary, if the reconfiguration is done manually or through CLI scripts, data exchange with MQTT in IoT is not reliable and is not easy to recover. Maintenance and management of data exchange in this model is a hard task which can be costly. Therefore, a reliable, automatic data discovery and data exchange model with a shortest path in a fragile, dynamic network environment is very important. With our algorithms, we can find the specific nodes for specific data with one topic and get the shortest path to that nodes dynamically and automatically. We need a smarter recovery program, which can save money (automatically) to avoid remote login and can perform recovery efficiently. Generally speaking, through the broadcast method and the way of active subscription, we can achieve data exchange automation. The existing practices often involve manual configuration where messages are read or processed manually. In our proposed model, we aim for automatic data exchange.

3. Broker Network

Data collection is one of the key functions in IoT system. However, battery powered devices in IoT often work in low bandwidth and fragile network environments. One of the major challenges is to provide efficient, stable, and scalable services in IoT. For example, to extend a device's effective working life-span as much as possible, efficiency should be considered to save power. At the same time, to ensure the availability and scalability of data communication services, broker network is a natural choice in IoT system [21, 22]. The broker network can provide stable connections among the devices to handle the data traffics in IoT. In this section, we describe our proposed novel topic-based data transmission algorithm to efficiently exchange data in IoT system.

3.1. Topic-Based Data Routing in MQTT. When servers in the cloud system gather data from things as messages in IoT system, we will route the messages flow through broker network in IoT to cloud. For example, client 1 subscribes to broker 1 with topic "top" and client 2 publishes to broker 2 with topic "top." Broker 2 should route the message published by client 2 to broker 1, so that client 1 can get the corresponding data. This is the so-called topic-based message routing problem in IoT. The key point here is the efficiency when broker nodes synchronize data among brokers, which include the following:

(i) Forwarding all messages to the other bridges

(ii) Routing of messages to the brokers

For the CPU, bandwidth, and memory constrained devices, the routing decision algorithm should be simple enough to save resource. Normally, we can divide the nodes into two types: one is routing node and the other is terminal subscriber node or publisher node. Our data routing decision algorithm is depicted in Algorithm 1 called TBRouting.

In summary, there are three scenarios:

(i) One to one directly

(ii) One to many directly

(iii) One to remote nodes through routers

In one to remote nodes through routers scenario, the cloud servers gather data from devices as messages transmit through the local IoT network. We address the issue of collecting specific data and how to transfer it efficiently in our algorithm through finding the shortest path based on topics for message delivery to cloud or server. That is to say, a routing protocol is considered. Our message routing decision algorithm is depicted in Figure 2.

In Figure 2, we depict message routing scenario. There are 7 routers, router ①~router ⑦ connected with each other wirelessly and to the Internet or cloud shown in Figure 2. ① is connected to Internet or cloud, and ⑥ is connected to sensor network. When messages are sent to router ⑥ through sensor network, the message will be routed from router ⑥ to router ①. Because the messaging router here is not the typical router in a typical network, therefore, we define the message routing algorithm below and routing table in the following sections. For the fragile network and constrained hardware devices, the topology may be changed constantly [23]. Therefore, the path and router table may change with the topology. Just as the scenario shown in Figure 2, the routing algorithm is introduced in the following steps:

(1) Figuring out the shortest path between router ⑥ and router ①, it is a single-source shortest path topic-based problem in graph theory. We can address it, for example, with Dijkstra's algorithm. Here, the shortest path is router serial ⑥ $\Longrightarrow$ ④ $\Longrightarrow$ ② $\Longrightarrow$ ①.

(2) In reverse sequence, router ① subscribes topics from router ②, router ② subscribes topics from router ④, and router ④ subscribe topics from router ⑥.

(3) When sensors publish topics to router ⑥, the messages will be pushed back to hop by hop until router ①.

When the topology structure of router network changes, we can figure out the new shortest path from the source to the end as described in Step (1) dynamically. The "routing tables" are critical to the system because IoT network may not be using the traditional TCP/IP protocol stack which makes real-time data transmission with the shortest path without considering the network protocol even more challenging. That is to say, regardless of TCP/IP or Zigbee, the IoT system can still find the data source and the shortest path transparently based on topics. In the next section, we will describe this algorithm in detail.

3.2. Topic-Based Data Routing Protocol. In order to facilitate problem description, first we provide some relevant definitions. Then algorithm is depicted in detail with an example. We present efficiency evaluation of our algorithm in Section 4.

3.2.1. Definitions

Definition 1. Routing table structure: in order to make nodes find each other and in the shortest path, some information should be stored in every nodes. Table 1 provides the structure of the routing table.

Definition 2. Subscribing node: node 1 subscribes to node 2; node 1 is the so-called subscribing node.

Definition 3. Subscribing node queue: A subscribes to B, B subscribes to C, C subscribes to D, and "ABC" is the so-called subscribing node queue.

Definition 4. Weight (T): the number of node in a string.

Definition 5. Host_Topic (T): the ultimate data consumer and corresponding topic. For example, "Host_Topic (ABCD/TOP)" = "D/TOP".

Definition 6. Searched_Set: a node set, in which every node has finished building the routing table.

Definition 7. NotSearched_Set: a node set, in which every node has not finished building the routing table.

Definition 8. Adjacent (X): a set of adjacency of node X.

Definition 9. Match (top): determining whether the node can provide the corresponding topic data.

3.2.2. Routing Algorithm. The algorithm essentially completes the following tasks. First, the algorithm determines the node where the topic and corresponding data reside. Second, the algorithm dynamically finds the shortest path. Then the data can be transported to Internet. That is to say, the algorithm solves that which node has the topic and data and how the data can be transported to the ultimate user. We solve this problem with our new routing algorithm named Topic-Based Routing Algorithm (TBRG). The rerouting algorithm is given as follows. In the algorithm, "**R**" is the ultimate data consumer in Algorithm 2.

Next, we elaborate our algorithm through a concrete example illustrated in Figure 3, which shows an example of nodes structure of IoT system. We assume Ⓐ is a node in data center, data ultimate consumer, and ⓘ is the data source connecting the sensor.

① Initialization of node Ⓐ in Table 2.

② A subscribing to nodes Ⓑ, Ⓓ, and Ⓔ, that is, Ⓐ ⇒ Ⓑ, Ⓐ ⇒ Ⓓ, Ⓐ ⇒ Ⓔ, as shown in Tables 3–5.

③ Next, B and E subscribing to node Ⓒ and node Ⓕ, respectively, that is, node Ⓑ ⇒ node Ⓒ and node Ⓔ ⇒ node Ⓕ in Tables 6 and 7.

④ Next, node Ⓒ and node Ⓕ subscribing to nodes ⓘ and Ⓖ, respectively, that is, node Ⓒ ⇒ node ⓘ and node Ⓕ ⇒ node Ⓖ as shown in Table 8, VX.

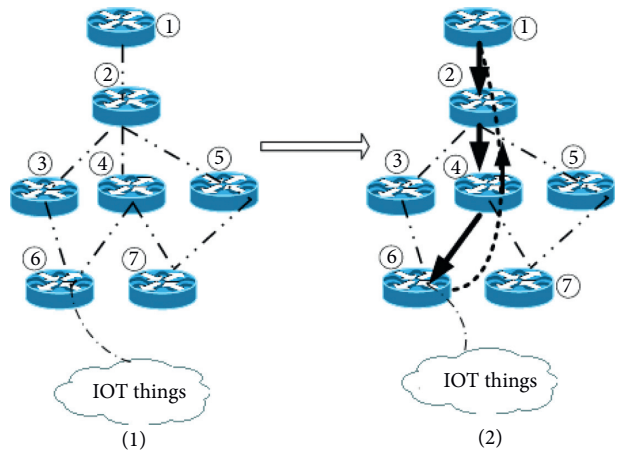

Figure 2: Message routing decision algorithm IoT network.

(1) **procedure TBR**outing.
(2) When a broker receives data with topics from one client, the broker matches topics in its topic tree.
(3) If the node is just the terminal subscriber, then broker forwards it to that node, and then go to 5. If the node is not the terminal consumer of the topic, go to 4.
(4) If the node is not the terminal consumer of the topic, the data will be forwarded to the other broker nodes, which is the next hop of the routing.
(5) Continuing with 3, with next hop, data eventually will be sent to cloud or terminal nodes. The messages are sent with the shortest path to the final destination.
(6) The topic may be sent to many different receivers who subscribe the topics. All the receivers will receive the same messages.
(7) The broker will decide whether to store the topic or not.

Algorithm 1: TBRouting.

Table 1: Structure of the routing table.

Node	Path and topics	Weight
Subscribing node	Subscribing node queue	Number of nodes in queue

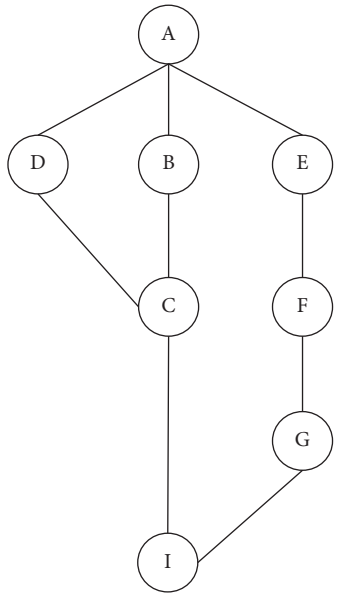

Figure 3: Example of topic routing protocol.

⑤ Node ⑥ subscribing to Node I, that is, node ⑥ ⇒ node I'. In the algorithm, this step is less likely to happen because of the value of weight (CBA/top) in Step ④; the routing information is "CBA/top", as shown in Tables 9 and 10.

Figure 4 shows a bipartite graph. Set (1) is a topic set $x1 \sim xn$ corresponding to published data. Set (2) is a topic set $y1 \sim yn$ corresponding to subscribed data. The next algorithm is to match topics from one node in set (1) to another node in set (2) step by step according to the shortest path in reverse order. In addition, when we upgrade system or update the status about things in field, that is to say, delivering the instruction set from cloud to IoT devices, the instruction dataset is routed reversely with a series of publish and then subscribing among routers.

3.3. Rerouting Algorithm in MQTT. Each broker may serve for thousands of clients' connections. In case one broker crashes down, clients should have the ability to connect to another broker and work continually and smoothly, which is called the rerouting problem. To address this problem, we propose a rerouting algorithm to take over the topic-based session flows with a controller when a broker crashes down. Although it is almost impossible to communicate when broker crashes down, data can still be communicated by another

```
(1) Procedure TBRG
(2)   RoutingTable ⟵ initially, ultimate consumer node R;
(3)   Searched_Set ⟵ R;
(4)   NotSearched_Set ⟵ R;
(5)   while NotSearched_Set ≠ ∅ do
(6)     X ∈ NotSearched_Set;
(7)     for all Adjacent(X) do
(8)       NodeX ∈ Adjacent(X);
(9)       if NodeX ∉ Searched_Set then
(10)        NotSearched_Set ⟵ NodeX;
(11)        X⇒subscribing to NodeX;
(12)        Node X ⟵ subscriping request P;
(13)        i ⟵ Weight(P);
(14)        top ⟵ Host_Topic(P);
(15)        if !Match(top) then
(16)          addItemToRoutingTable(NodeX);
(17)          Searched_Set ⟵ Nodex;
(18)        else if Weight(Route_NodeX(Top)) > i then
(19)          Rewrite(Route_NodeX(P));
(20)     NotSearched_Set − X;
(21)     Searched_Set ⟵ X;
```

Algorithm 2: TBRG.

Table 2: Routing table of node A.

Node	Path and topics	Weight
0	0/top	0

Table 3: Routing table of node B.

Node	Path and topics	Weight
A	A/top	1

Table 4: Routing table of node D.

Node	Path and topics	Weight
A	A/top	1

Table 5: Routing table of node E.

Node	Path and topics	Weight
A	A/top	1

Table 6: Routing table of node C.

Node	Path and topics	Weight
B	BA/top	2

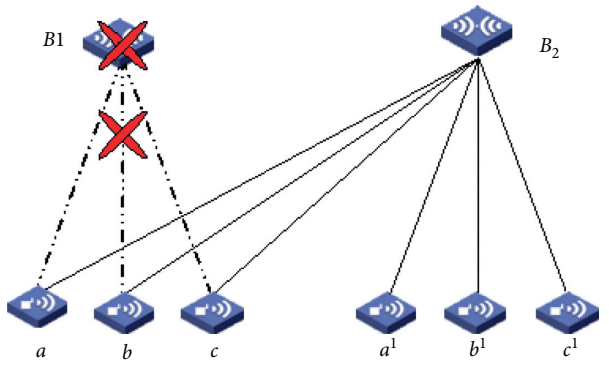

Figure 4: Routing tables in message router.

broker with rerouting mechanism. Therefore, high availability can be gained with this model in IoT, as shown in Figure 5.

3.3.1. Definitions. As in Section 3.2, we first provide some relevant definitions to facilitate the node replacing problem statement.

Definition 10. Topic_Session: when node X publishes a topic "t" to a broker named "Brk," and node Y subscribes the same topic "t" to the same broker "Brk" just illustrated in Figure 6(a) shown; we call this binary relation *Topic_Session* with notation $X\mathrm{TSR}(t, \mathrm{brk})Y$. If node Y only can get topic "t" from node X, we use notation $X!\mathrm{TSR}(t, \mathrm{brk})Y$ to express this relation. We depict the relation with Structural Operational Semantics as follows:

$$\text{Node } X \xrightarrow{\text{Pub}(t)} \text{Brk},$$
$$\text{Node } Y \xrightarrow{\text{Sub}(t)} \text{Brk},$$
$$\text{Node } X \xrightarrow{\text{Pub}(t)} \text{Brk},$$

Normally, we can use process expression to depict communication behavior. Actually, from the perspective of mathematics, this is a binary partial order relation. That is to

Table 7: Routing table of node F.

Node	Path and topics	Weight
E	EA/top	2

Table 8: Routing table of node of G.

Node	Path and topics	Weight
F	FEA/top	3

Table 9: Routing table of node I.

Node	Path and topics	Weight
C	CBA/top	3

Table 10: Possible routing table of node I'.

Node	Path and topics	Weight
G	GFEA/top	4

Table 11: Topic session flow.

No.	Things	Broker	Sub/Pub	Topic	On/off
1	a	B_1	0	t_1	1
2	a	B_1	1	t_6	0
3	a	B_2	1	t_2	1
4	b	B_1	1	t_3	1
5	a	B_1	0	t_4	1
6	c	B_2	1	t_5	0
7	c	B_1	0	t_6	11

say, $X \xrightarrow{\mathrm{TSR}(t)} Y$ and $Y \xrightarrow{\mathrm{TSR}(t)} X$ are different pair set. We use this concept to indicate the data flow.

Definition 11. $Pub(t) \cdot X$: processing node X has an action Pub with a parametric t. In this paper, behavior *Pub* is published.

Definition 12. $Sub(t) \cdot X$: processing node X has an action Sub with a parametric t. In this paper, behavior *Sub* is subscription.

Definition 13. Broke Strong Behavior Equivalence$\sim$: there are two brokers A and B. If the following relation exists, we call this relation $A \backsim B$:

(i) $X\,\mathrm{TSR}(t1, A)Y$ with topic "$t1$"

(ii) $X\,\mathrm{TSR}(t2, B)Y$ with topic "$t2$"

Definition 14. Broker Weak Behavior Equivalence$\approx$: there are two brokers A and B. If the following relations exist, we call this binary relation weak behavior equivalence $A \approx B$:

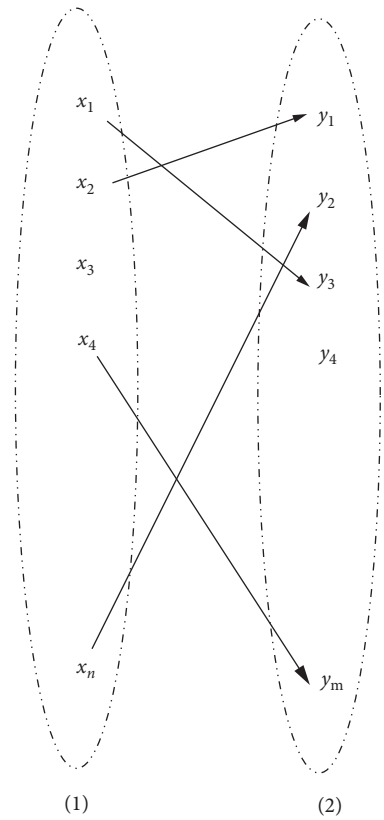

Figure 5: Fault migrating.

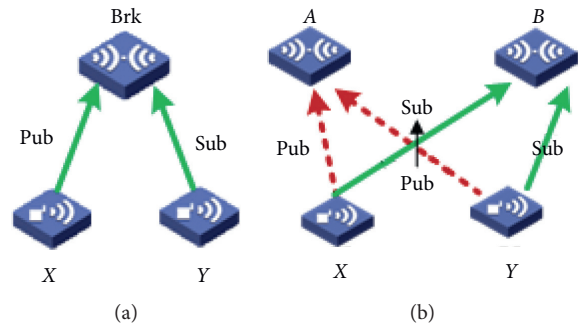

Figure 6: Topic_Session.

(i) $X\,\mathrm{TSR}(t1, A)Y$ or $Y\,\mathrm{TSR}(t1, A)X$ with topic "$t1$"

(ii) $X\,\mathrm{TSR}(t2, B)Y$ or $Y\,\mathrm{TSR}(t2, B)X$ with topic "$t2$"

If broker A crashes down, we want to find another broker B to take over the topic session flows in broker A, in which brokers A and B at least have the Strong Behavior Equivalence relation A$\backsim$B, as Figure 7 showed. In this section, we will focus on how to find an appropriate broker to take over the topic session flow in the crashed broker. With this behavior, the Topic_Session can continue.

3.3.2. Problem Analysis. In normal wireless network, if one node loses its connection, the node may update its status and broadcast its identification information to all neighbor nodes, just as shown in Figure 6(b). Then other sensors or actuators

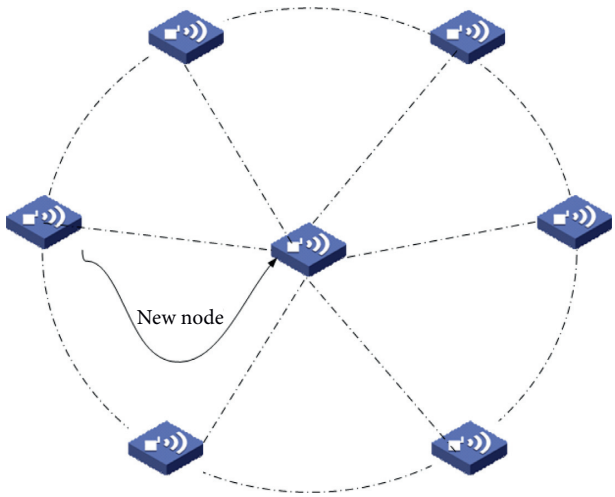

Figure 7: Wireless network broadcast.

may connect to this node. In this model, for example, in Figure 6(b), node X and node Y may find the other nodes.

The IoT network node is different from the normal wireless network node as follows.

(i) Node Y gets the topic "t" from node X through broker A with subscription behavior. However, when broker A crashes down, nodes X and Y must communicate with another broker with the same behavior. Or else node Y cannot get the topic "t" from node X. The situation may be worse when nodes X and Y have the relation $X!(t, \mathrm{brk})Y$.

(ii) In MQTT, one broker is specified when nodes publish or subscribe to a broker. That is to say, the subscriber or publisher cannot automatically reconfigure the broker. For example, MQTT protocol with QoS 0 and the node even are not aware when the broker crashes down.

The problem here is that nodes and brokers have an equal status. So nodes almost know nothing about each other. Because of mutual independence to each other, one node cannot change another node's behavior.

To sum up, when node X and node Y have the relation $X!(t, \mathrm{brk})Y$, nodes X and Y must synchronize migration to other nodes. Publisher may continue to publish data to broker A even when A has crashed down with QoS 0. Therefore, we must clarify the substitution conditions that how a broker can be replaced by another broker partly or totally. This raises the following questions:

(i) What conditions should be met when a broker substitutes one or a few brokers to take over that broker?

(ii) Who will we find a crashed down broker?

(iii) How can we take measures on the nodes or brokers to take over that broker?

(iv) Where should we store the relation $X\,\mathrm{TSR}(t, \mathrm{brk})Y$ set in that crash down broker? So we can keep topic related sessions to continue the data communication.

3.3.3. Broker Substitution Conditions. According to process behavior equivalence, if brokers A and B have a weak behavior equivalence relation $A \approx B$, then nodes X and Y have the approximate communication ability with A and B as shown in Figure 8.

The process expression of broker A is P_A, which is defined by the following syntax:

$$
\begin{aligned}
&P_A ::= \textstyle\sum_{i\in I} \prod \cdot A \,|\, ! \prod \cdot A,\\
&\textstyle\prod ::= Pub(t_i);\\
&Sub\langle t_i \rangle;
\end{aligned}
$$

The notation $!\prod \cdot A$ is loop execution behavior. And t_i is a topic $\in$ Topic_Set, which is a topic set to be published or subscribed. $\prod$ is an action prefix. Then the broker substitution conditions are

$$
\begin{aligned}
&\overline{Pub<t_i>} \cdot X \,|\, Pub(t_i) \cdot A \,|\, Sub(t_i) \cdot A \,|\, \overline{Sub<t_i>} \cdot Y;\\
&\overline{Pub<t_j>} \cdot X \,|\, Pub(t_j) \cdot B \,|\, Sub(t_j) \cdot B \,|\, \overline{Sub<t_j>} \cdot Y;\\
&\Rightarrow A \sim B, X\,\mathrm{TSR}(t, \mathrm{brk})Y;\\
&\begin{pmatrix} \overline{Pub<t_i>} \cdot X \,|\, Pub(t_i) \cdot A \vee Sub(t_i) \cdot A \,|\, \overline{Sub<t_i>} \cdot X \\ \vee \overline{Pub<t_i>} \cdot A \,|\, Pub(t_i) \cdot X \vee Sub(t_i) \cdot X \,|\, \overline{Sub<t_i>} \cdot A \wedge \end{pmatrix},\\
&\begin{pmatrix} \overline{Sub<tj>} \cdot Y \,|\, Sub(tj).B \vee Pub(tj) \cdot B \,|\, \overline{Pub<tj>} \cdot Y \\ \vee \overline{Sub<t_j>} \cdot B \,|\, Sub(t_j).Y \vee Pub(t_j) \cdot Y \,|\, \overline{Pub<t_j>} \cdot B \end{pmatrix},\\
&\Rightarrow A \approx B;\\
&\alpha \cdot X \,|\, \overline{\alpha} \cdot A \,|\, \beta \cdot B \,|\, \overline{\beta}.Y + \overline{\alpha} \cdot X \,|\, \alpha \cdot A \,|\, \beta \cdot B \,|\, \overline{\beta}.Y +\\
&\alpha \cdot X \,|\, \overline{\alpha} \cdot A \,|\, \overline{\beta} \cdot B \,|\, \beta \cdot Y + \overline{\alpha} \cdot X \,|\, \alpha \cdot A \,|\, \overline{\beta} \cdot B \,|\, \beta \cdot Y;\\
&\alpha \in \{Pub, Sub\},\\
&\Rightarrow A \approx B;\\
&\text{where,} \quad t_i, t_j \in Topic\_Set.
\end{aligned}
\tag{1}
$$

Essentially, Strong Behavior Equivalence $\backsim$ is a binary homomorphism relation. And Weak Behavior Equivalence $\approx$ is roughly approximate relation. If brokers A and B meet following conditions,

$A \backsim B$ or $A \approx B$, $XTSR(t, A)$, Y, $t \in Topic\_Set$, topic sessions on A can be transferred to broker B.

3.3.4. Pub/Sub Flow Controller and Measures. As noted in the previous section, publishers are not aware if a broker crashes down. Even the publisher knows the absence of broker, the topic-based session cannot go on without outside help, not to mention finding one or a few brokers to take over the shutdown broker. Therefore, the extra controller is necessary to ensure the survivability of IoT system, just as Software Defined Networks (SDN) do [24]. As shown in Figure 9, there is a controller over brokers to schedule the whole IoT network. The main purposes of the controller are as follows:

(i) Reviving of dead broker through heart beating

(ii) Figuring out the substitute node

(iii) Transferring the topic-based session flow from crash down broker to alternative broker

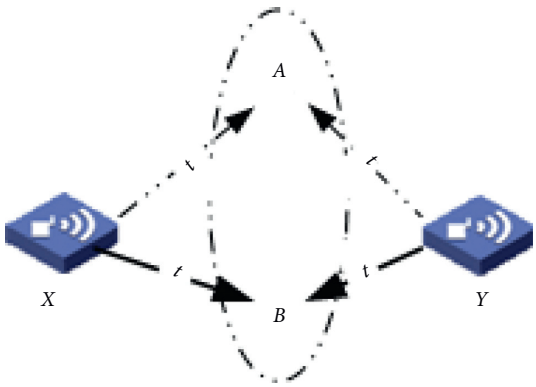

Figure 8: $A \approx B$.

But the controller will never be used as data communication. Its main usage is to manage the nodes in IoT network.

Through heart beating, for example, subscribing "hello world" to managed node on a regular timer with Quality of Service (QoS), it is easy to get the information in broker node in order to determine a substitute node. To figure out substitute node, the controller should collect the following information in regular time. For example, as shown in Table 11, in regular intervals.

In Table 11, if the action is subscribe then 1, else then 0. And if the broker is just performing normal then 1, else then 0.

Through the content in Table 11, we can deduce nodes that can be replaced with each other. For example, B1 and B2 can replace each other, as shown in the proof below:

$$\begin{aligned}
&\because \mathrm{Sub}(t)\cdot a|\,\mathrm{Sub}(t6)\cdot \overline{\mathrm{Pub}(t6)}B1|\,\mathrm{Pub}(t6)\cdot c,\\
&\therefore \Rightarrow c\,\mathrm{TSR}(t6,\ B1)a;\\
&\because \begin{cases} \mathrm{Sub}(t2)\cdot a|\,\mathrm{Sub}(t2)\cdot \overline{\mathrm{Sub}(t5)}B2|\,\mathrm{Sub}(t5)\cdot c, \\ \mathrm{Pub}(t1)\cdot a|\,\mathrm{Pub}(t1)\cdot \overline{\mathrm{Pub}(t6)}B1|\,\mathrm{Pub}(t6)\cdot c, \end{cases} \quad (2)\\
&\therefore \Rightarrow B1 \approx B2.
\end{aligned}$$

This comes to the conclusion $c!(t, B2)a$. This means when broker B1 crashes, the B2 can replace B1, with Structural Operational Semantics depicted as follows:

$$c \xrightarrow{\mathrm{Pub}(t6)} B1 \xleftarrow{\mathrm{Sub}(t6)} a \approx c \xleftarrow{\mathrm{Sub}(t6)} B2 \xrightarrow{\mathrm{Pub}(t6)} a. \quad (3)$$

The Algorithm 3 is showed below.

4. Implementation and Evaluation with Mosquitto or IoTivity

We implement a working prototype of the above algorithm using C and MQTT on Ubuntu system with Mosquitto and its client libmosquitto.lib as shown in Figure 10. MQTT is a machine to machine open source protocol originally developed by IBM. MQTT is particularly suitable for resource constrained IoT devices on fragile networks environment. As discussed in Section 1, MQTT is the most popular protocol used in IoT system now. And Mosquitto is an open source implementation of the MQTT protocol.

In our evaluation system, Figure 10, we deployed 25 nodes and divided nodes into two groups, one as broker and another as client. On the clients, we deployed a shell program to continuously publish data. And the nodes in broker network collect the data with subscription.

4.1. Efficiency with Scale of Data Transmission. Intuitively, in resource constrained device, the efficiency will decrease dramatically with data scale increasing. The trend of data transmission with data scale is one of the experimental contents of in this section. Especially to the specific protocol, the data transmission laws are more worthy of attention. In this section, we will evaluate the average transmission time with increased data scale. In Figure 11, vertical axis is time cost, and horizontal axis is the amount of data transmitted. From Figure 11, we can observe that average transfer completion time will increase dramatically from a specific point.

In broker network, a huge number of topics will be transmitted through the nodes. However, according to the above experiments, we know that the efficiency will decrease dramatically with increasing topics transmitted. Therefore, reducing the number of transmission nodes can reduce propagation delay time, and improve the transmission efficiency. This is just our primary goal in this paper. As the evaluation system shows with our algorithms proposed in this paper we can reduce the number of transmission nodes, which can improve the data transmission efficiency. At the same time, according to the experiment results, whole system efficiency will be improved as evident from the experiment results in the next section.

4.2. Efficiency Improvements. As Figure 12 showed, we used 25 virtual machines as a testbed to test algorithms proposed in this paper. Resources allocated to a virtual machine are limited. For example, memory in each node is less than 300 M. And the nodes are divided into two groups: one group is mainly for broker network and the other group is used for generating data. We deployed a Mosquitto broker in each node. In order to test any scale of the data, we used an endless loop shell script to publish a topic per second continuously; the script is as follows:

```
#!/bin/sh
while true; do
    echo hello.
    sleep 1
done | mosquitto_pub –l –t "Topic"
```

The other nodes can get the data through Mosquitto_sub with a counter. The counter is used to control the data scale. In the program, except subscription and counter, we calculate the time of data transmission. Through a series of subscription behaviors, the contents with corresponding topics can be passed to the final data consumer indirectly node by node and step by step. Each node automatically becomes a data generator pump. Just as described above, we have built an IoT simulation network with libmosquitto library and virtual machines as shown Figure 12, where

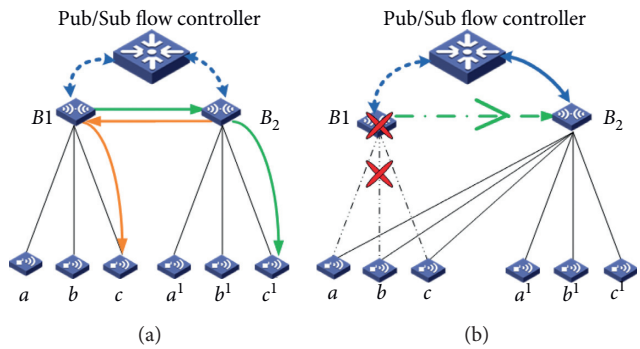

Figure 9: IoT controller.

(1) **Procedure** ReRouting
(2) *Session_Flow_Table* ⟵ *initialization*;
(3) **While** true **do**
(4) *Heartbeating broadcast*;
(5) *On/off* ⟵ *Heartbeatingbroadcast*;
(6) **for all** *No de X* ⟵ *Broker Set* **do**
(7) *PullingthePub/Subinformation*;
(8) *derive Broke Strong Behavio Equivalence*~;
(9) *deriveWeak Behavior Equivalence*≈
(10) *derive Topic_Sessionrelationship*;
(11) *Update Topic_SessionSet*;

Algorithm 3: ReRouting.

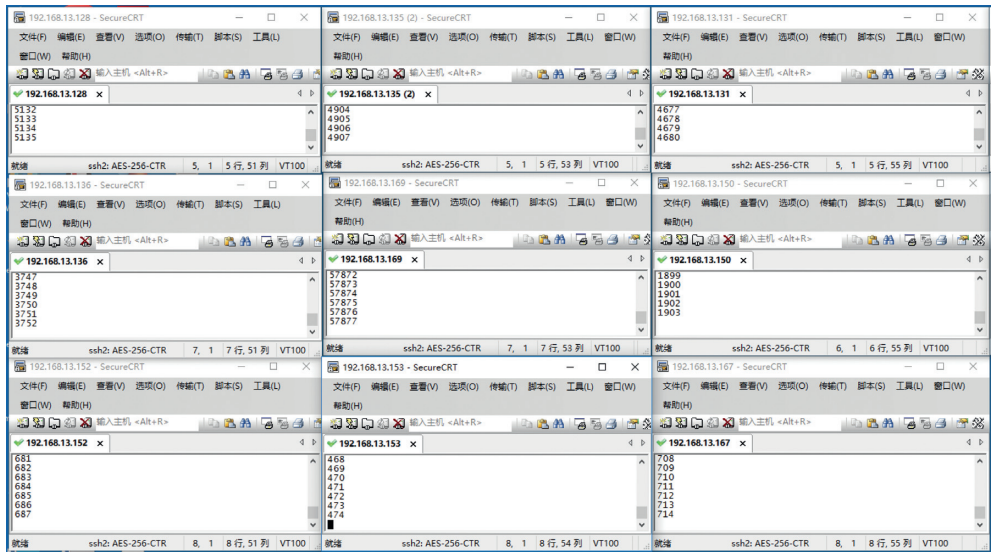

Figure 10: Evaluation system.

nodes 1 ~ 7 are broker network nodes; the rest of the nodes are data generator. Node 1 is the final data consuming node. Node 6 is the edge routing node. Data collected from the IoT will be transmitted to node 6 first and then transmitted to node 1 through broker networks.

Just as described above, our algorithm can determine nodes where the topic and corresponding data reside, and it dynamically adopts the shortest paths between nodes in data consumer and edge routers. The shortest paths mean fewer nodes needed for data transmission. Fewer transmission nodes can save data transmission time. We use the network topology shown in Figure 12 to evaluate the algorithms presented in this paper. In Figure 12, the data may be transmitted through ⑥, ⑦, ④ ③, ②, and ①. And the shortest path is ⑥, ④, ②, and

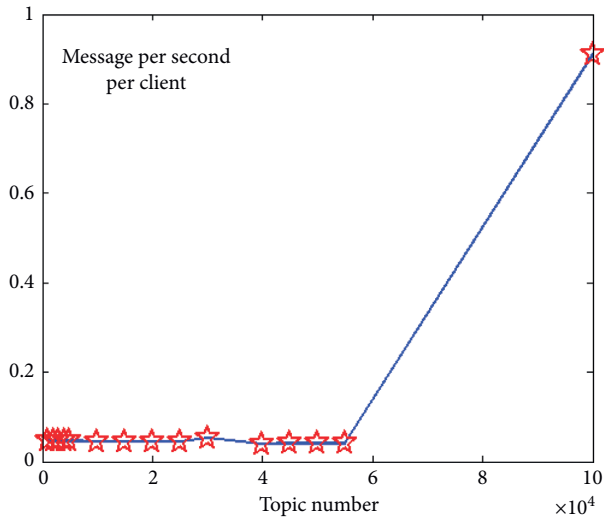

Figure 11: Transmission time with increased data scale.

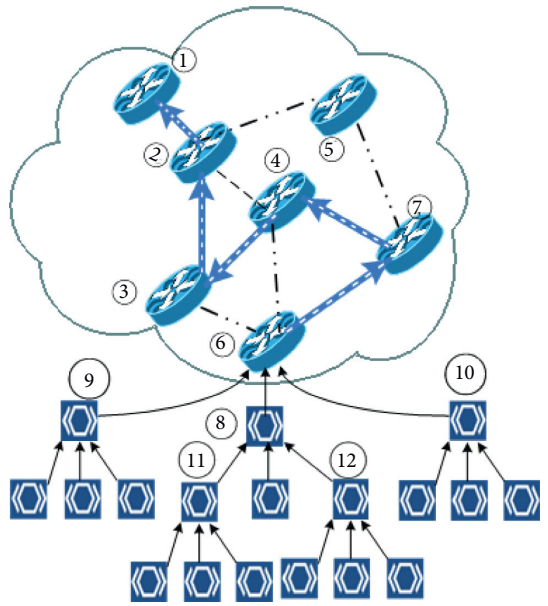

Figure 12: Data transmission without the shortest path.

①, as shown in Figure 13. That is to say, the path in Figure 13 is just the path we want to use to transmit the data.

To get the influence of the data transmission path with the data scale, we obtain the evaluation results with different workload to test the algorithm. The primary performance metrics are data transmission completion time. We get the data from 500 to 10000 items with 500 as a step. We run a Mosquitto on each node in transmission network as a broker. At the same time, we develop a program based on libmosquitto as a client to get the data and calculate the transmission completion time on each node. Figure 14 shows the performance comparison of our algorithm with different amounts of time. At the same time, to compare the entire system performance influence, the average transmission completion times of all nodes are measured, which is shown in Figure 15. In fact fewer transmission nodes can reduce energy consumption for whole system, which is not the scope of this paper.

We compare the data transmission with and without the shortest path in Figure 14. First, we observe that almost in each data scale the transmission completion time with the shortest path is far less than without it. Second, the amount of data transmitted and the transmission completion time is not a simple linear relationship. The comparison of transmission completion time about edge router is depicted in Figure 16. We illustrate the influence to the transmission node under these two different situations using Figure 16. In Figure 16, we observe that the data transmission path has an important influence on the whole system performance. The solution in edge router and in final data consumer has a similar trend. It is sufficient to show that data transmission

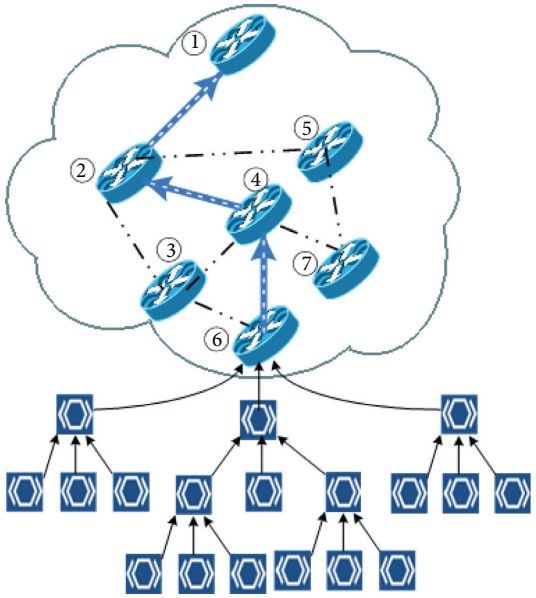

Figure 13: Data transmission with the shortest path.

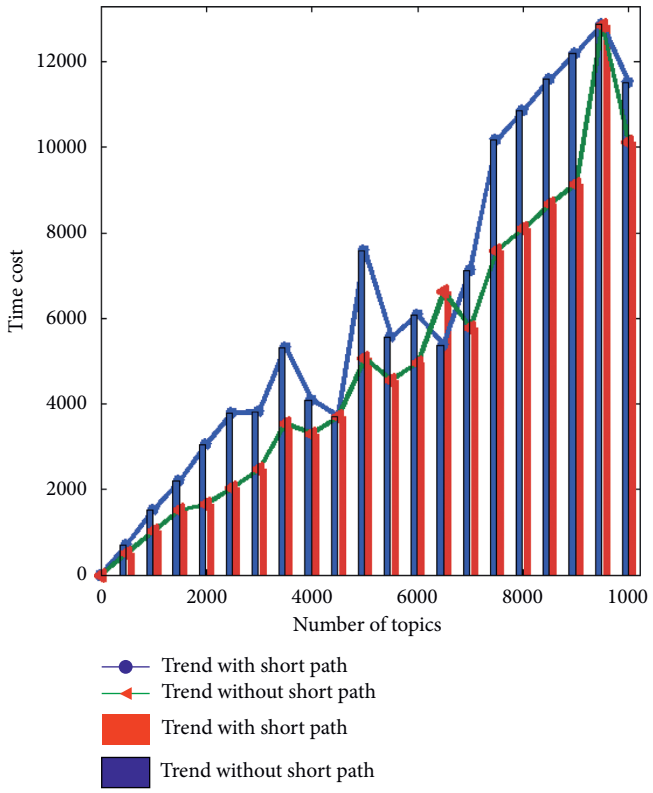

Figure 14: Comparison of data transmission completion time to the data consumer.

path has a significant impact on the network system. We can see from the results that the shortest path has high efficiency.

In order to further verify this view, we compute the average transmission completion time of all the nodes in the path shown in Figure 15 and observe similar trend as in Figure 16.

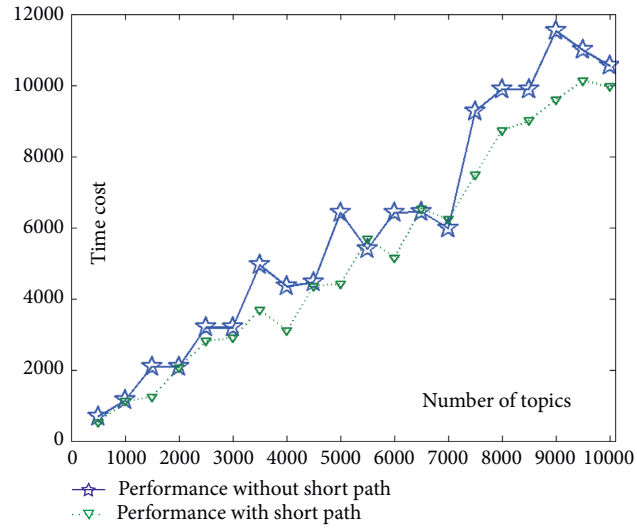

Figure 15: Comparison of the overall performance on two path.

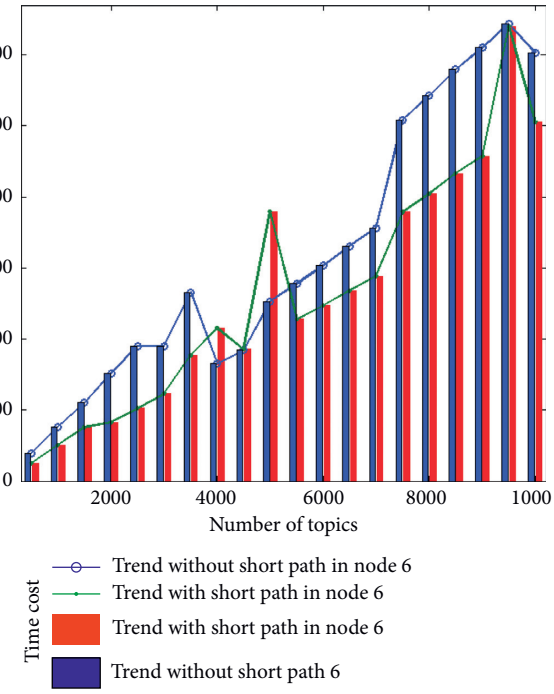

Figure 16: Comparison of data transmission completion time to the edge router nodes.

5. Conclusions

We have presented novel topics-based data routing protocols and algorithms in MQTT. The routing algorithms optimize IoT data transfers with constrained resource and fragile network environments in publish-subscribe model. Our proposed algorithms can adapt the dynamical network topology and are capable of adopting continuously changing devices with rerouting method in IoT system. Through experimental results, we have demonstrated efficiency of our algorithms for message routing, and dynamically adapting the continually changing device and network topology. We

have evaluated our algorithms with a prototype with a Mosquitto based on MQTT protocol. With the experiments performed and simulations on Mosquitto, the results show that our algorithms work well. The proposed algorithms can be widely used in IoT system with publish-subscribe model. Moreover, the algorithms have the capability to be tested in other protocols such as CoAP [25]. From security point of view, identity and authorization of nodes are critical problems [26–30]. Threats such as avoiding data to be published or subscribed by attackers will have adverse effects on the broker in IoT system. In this work, we have focused on the data exchange efficiency in IoT, mainly under the MQTT protocol; in our future work, we aim to study security issues in IoT [31–34].

From security view, identity and authorization of nodes are critical problems avoiding data to be published or subscribed by hacker or crack down the broker in IoT. In this paper, we focus on the data exchange efficiency in IoT, mainly under the MQTT protocol and not involving the security problem, which we will deeply research in future work.

Data Availability

Any data can be obtained from the dropbox url (https://www.dropbox.com/s/vjm06c4z5kbg79i/%E9%87%87%E9%9B%86%E6%95%B0%E6%8D%AE3.txt?dl=0).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors thank Professor Zia Tanveer for his kind review and help. This work was supported in part by National Science Foundation of China under grant nos. 61672104, 61170209, 61702570, and 61602537 and U1509214, Program for New Century Excellent Talents in University under grant no. NCET-13-0676, and Shenzhen Institute of Artificial Intelligence and Robotics for Society.

References

[1] S. Deng, Z. Xiang, P. Zhao et al., "Dynamical resource allocation in edge for trustable internet-of-things systems: a reinforcement learning method," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, p. 6103, 2020.

[2] M. Fischer, D. Kümper, and R. Tönjes, "Towards improving the privacy in the MQTT protocol," *GIoTS*, vol. 19, pp. 1–6, 2019.

[3] IoTivity, "IoTivity," 2017, https://www.iotivity.org/.

[4] J. Sathish Kumar and M. A. Zaveri, "Clustering for collaborative processing in IoT network," in *Proceedings of the Second International Conference on IoT in Urban Space (Urb-IoT 16)*, pp. 95–97, ACM, New York, NY, USA, 2016.

[5] CoAP, "Constrained application protocol (CoAP)," 2017, http://coap.technology/.

[6] AMQP, "Advanced message queuing protocol (AMQP)," 2017, https://www.amqp.org/.

[7] WebSocket, "WebSocket," 2017, https://www.websocket.org/index.html.

[8] Y. Zhao, K. Kim, and N. Venkatasubramanian, "DYNATOPS: a dynamic topic-based publish/subscribe architecture," in *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (DEBS '13)*, pp. 75–86, ACM, New York, NY, USA, 2013.

[9] M. Stolpe, "The internet of things: opportunities and challenges for distributed data analysis," *ACM SIGKDD Explorations Newsletter*, vol. 18, p. 1, 2016.

[10] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[11] AllSeen Alliance, "Linux foundation," 2017, https://allseenalliance.org/framework/documentation/learn/core/standard-core.

[12] MQTT, "Message queuing telemetry transport (MQTT)," 2017, https://mqtt.org/.

[13] B. Karakostas and N. Bessis, *Intelligent Brokers in an Internet of Things for Logistics*, ACM, New York, NY, USA, 2016.

[14] H. Gao, W. Huang, and Y. Duan, "The cloud-edge based dynamic reconfiguration to service workflow for mobile ecommerce environments: a QoS prediction perspective," *ACM Transactions on Internet Technology*, vol. 36, 2020.

[15] B. Aziz, "A formal model and analysis of an IoT protocol," *Ad Hoc Networks*, vol. 36, p. 49, 2016.

[16] W.-J. ChenRahul, "Responsive mobile user experience using MQTT and IBM MessageSight," *IBM Redbooks*, vol. 3, 2014.

[17] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for internet-of-things services," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4532–4542, 2020.

[18] S. Sakulin, A. Alfimtsev, E. Tipsin, V. Devyatkov, and D. Sokolov, "User interface distribution method based on pi-calculus," *International Journal of Distributed Systems and Technologies*, vol. 10, no. 3, pp. 1–20, 2019.

[19] F. Xiong, A. Li, H. Wang, and L. Tang, "An SDN-MQTT based communication system for battlefield UAV swarms," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 41–47, 2019.

[20] P. Kumar and B. Dezfouli, "Implementation and analysis of QUIC for MQTT," *Computer Networks*, vol. 150, pp. 28–45, 2019.

[21] H. Ziekow, *In-Network Event Processing in a Peer To Peer Broker Network for the Internet of Things*, Springer-Verlag, Berlin, Germany, 2007.

[22] S. D. Madhu Kumar and U. Bellur, "An underlay aware, adaptive overlay for event broker networks," in *Proceedings of the 5th Workshop on Adaptive and Reflective Middleware (ARM '06)*, ACM, New York, NY, USA, 2006.

[23] M. A. Jaeger, H. Parzyjegla, G. Mühl, and K. Herrmann, "Self-organizing broker topologies for publish/subscribe systems," in *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC '07)*, pp. 543–550, ACM, New York, NY, USA, 2007.

[24] G. De Luca and Y. Chen, "Visual IoT/robotics programming language in pi-calculus," *ISADS*, vol. 16, pp. 23–30, 2017.

[25] A. Larmo, A. Ratilainen, and J. Saarinen, "Impact of CoAP and MQTT on NB-IoT system performance," *Sensors*, vol. 19, no. 1, p. 7, 2019.

[26] D. J. Wu, A. Taly, A. Shankar, and D. Boneh, "Privacy, discovery, and authentication for the internet of things," *Computer Security-ESORICS 2016*, vol. 23, pp. 301–319, 2016.

[27] S. P. Mahambre and U. Bellur, "An adaptive approach for ensuring reliability in event based middleware," in

Proceedings of the Second International Conference on Distributed Event-Based Systems (DEBS '08), pp. 157–168, ACM, New York, NY, USA, 2008.

[28] A. J. Hintaw, S. Manickam, S. Manickam, S. Karuppayah, and M. F. Aboalmaaly, "A brief review on MQTT's security issues within the internet of things (IoT)," *Journal of Communications*, vol. 14, no. 6, pp. 463–469, 2019.

[29] H.-Y. Chien, X.-A. Kou, M.-L. Chiang, and C. Su, "Secure and efficient MQTT group communication design," *CSII (Selected Papers)*, vol. 19, pp. 177–186, 2019.

[30] R. Singh Bali, F. Jaafar, and P. Zavarsky, "Lightweight authentication for MQTT to improve the security of IoT communication," *ICCSP*, vol. 19, pp. 6–12, 2019.

[31] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, 2020.

[32] K. Mladenov, *Formal Verification of the Implementation of the MQTT Protocol in IoT devices*, Master Thesis, University of Amsterdam, Amsterdam, Netherlands, 2017.

[33] A. P. Haripriya and K. Kulothungan, "Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things," *EURASIP J. Wireless Comm. and Networking*, vol. 2019, p. 90, 2019.

[34] G. Kim, S. Kang, J. Park, and K. Chung, "An MQTT-based context-aware autonomous system in oneM2M architecture," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8519–8528, 2019.