Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 362076, 11 pages
http://dx.doi.org/10.1155/2014/362076

Research Article

Self-Adapting Routing Overlay Network for Frequently Changing Application Traffic in Content-Based Publish/Subscribe System

Meng Chi, Shufen Liu, and Changhong Hu

College of Computer Science and Technology, Jilin University, 2699 Qianjin Street, Changchun, Jilin 130012, China

Correspondence should be addressed to Shufen Liu; liusf@mail.jlu.edu.cn

Received 21 October 2013; Revised 2 March 2014; Accepted 4 March 2014; Published 2 April 2014

Academic Editor: Fuzhong Nian

In the large-scale distributed simulation area, the topology of the overlay network cannot always rapidly adapt to frequently changing application traffic to reduce the overall traffic cost. In this paper, we propose a self-adapting routing strategy for frequently changing application traffic in content-based publish/subscribe system. The strategy firstly trains the traffic information and then uses this training information to predict the application traffic in the future. Finally, the strategy reconfigures the topology of the overlay network based on this predicting information to reduce the overall traffic cost. A predicting path is also introduced in this paper to reduce the reconfiguration numbers in the process of the reconfigurations. Compared to other strategies, the experimental results show that the strategy proposed in this paper could reduce the overall traffic cost of the publish/subscribe system in less reconfigurations.

1. Introduction

The publish/subscribe paradigm has been broadly applied in many application scenarios, such as wireless sensor networks, large-scale distributed simulation, information dissemination, network monitoring, enterprise application integration, and ubiquitous systems. In fact, many improvements will be achieved when the underlying infrastructure of the system incorporates the publish/subscribe paradigm which decouples consumers and producers in terms of space, time, and synchronization, provides anonymous communication mechanism [1], and acts as a specialized mediator to provide scalability and adaptability. These improvements exactly meet the requirements of the modern systems design and implementation. Consequently, the publish/subscribe system, also called event-based system, has been recognized and researched both in academia and in industry recently.

A publish/subscribe system consists of a notification service, producers, and consumers. Producers issue advertisements to declare notifications they are willing to send and publish notifications. Consumers issue subscriptions to subscribe notifications they are interested in and react to the notifications. A notification is a reification of an event which is an arbitrary detectable state change in a publish/subscribe system. A subscription is a filter that is a Boolean-valued function that tests a notification and returns either true or false [2]. The notification service is responsible for conveying notifications and delivering every published notification to all consumers that have registered matching subscriptions. It forms an overlay network consisting of the brokers connected by links. From the perspective of expressiveness, the publish/subscribe system can be classified into five categories: channel-based, topic-based, type-based, content-based, and semantic-based. Among these categories, the content-based publish/subscribe paradigm is the most commonly applied scheme, so we focus on the content-based publish/subscribe system in this paper [3–7].

In the large-scale distributed simulation and sensor networks area, the system designer models thousands of entities to simplify abstractions of reality. To enable these independent entities (in our actual application scenario, it contains plane, ship, weather, data collect, data review, and data analyze entities) to work together with other entities toward the same goal, interoperability is the most important technical

driving requirement that a large-scale distributed simulation must meet. Among the current existing underlying communication mechanisms that have been applied in simulation, the publish/subscribe paradigm is the most appropriate one to address the requirement of the interoperability, so we use the publish/subscribe systems as our communication layer. In the process of the simulation, an entity may be a producer, a consumer, or both; a simulator may simulate one or many entities; a simulation node may contain one or multiple simulators; and the same simulation node may start different simulators in different simulation stages. Consequently, in the same simulation node, the type and the flow of the message, referred to as the application traffic in this paper, will be frequently changed during different stages of the simulation process, and this frequently changing application traffic will result in a problem that the topology of the publish/subscribe overlay network cannot rapidly adapt to it, which means that the overall traffic cost in the overlay network would not be reduced obviously through a series of reconfigurations by existing reconfiguration strategies. Especially, when a large number of simulators are deployed in a very limited simulation hardware environment or the simulation process must be speeded up, the problem will get even worse [8–15].

In this paper, we propose a self-adapting routing strategy of the publish/subscribe system. The strategy firstly trains the traffic information and then uses this training information to predict the application traffic in the future. Finally, the strategy reconfigures the topology based on this predicting information to reduce the overall traffic cost.

This paper is organized as follows. Section 2 discusses related work on this topic. Section 3 describes and formalizes the problem. Section 4 presents the strategy we propose. Section 5 shows and discusses the experimental results. Section 6 presents our conclusion.

2. Related Work

Several approaches of self-adapting routing strategy in the publish/subscribe system to reduce the overall traffic cost have been researched in different ways in the past. We present and discuss these approaches as follows.

TERA is proposed in the literature [16] which is based on the following idea. Brokers that want to subscribe to a topic are required to join the corresponding topic overlay. When a notification is published, it is first routed to an access point broker, and then the broker diffuses this notification in the topic overlay associated with matched topic, in order to forward it to all the other subscribers. This strategy will reduce the overall traffic cost. Another similar work CAN proposed in the literature [17], but there is only one single access point existing for each topic overlay in this strategy. Contrarily, to avoid traffic hot spots and single point of failures, TERA does not impose a single access on this architecture. However, these mentioned strategies are confined within the topic-based case, and they are not extendible to the more generalized content-based case.

There are several other overlay networks built on the P2P network, such as Pastry [18] and Chord [19]. The P2P network provides the publish/subscribe system with scalability, efficiency, and redundancy properties. However, the optimization problem to reduce the traffic cost is at the P2P network layers, and then the P2P network does not care for the notifications, the subscriptions, and the advertisements. Consequently, this problem will be resolved in the P2P network.

A publish/subscribe overlay decision problem (PSODP) is defined from Jaeger et al.¡?ehlt?¿ in the literature [20], and a self-organizing algorithm of broker overlay infrastructure to solve this problem has been presented. The main idea of their algorithm is to reduce the distance between brokers that consume a lot of identical notifications, while respecting communication and processing costs. The process of the algorithm consists of evaluation phase, consensus phase, and reconfiguration phase. It assigns to each node a cache that stores recorded information about the notification consumed by the broker, and then the evaluation and consensus of the reconfiguration are based on these recorded information. This algorithm only considers the forwarding message in the past. Once the application traffic changes frequently, the topology of the publish/subscribe overlay network cannot rapidly adapt to this application traffic in advance.

Another work from Migliavacca and Cugola in [21], an optimal content-based routing (OCBR), is defined, and a distributed algorithm to reduce the overall routing cost was proposed. The main idea of this algorithm is to create a direct link between the two neighbors that have the highest forwarded traffic and then disconnect itself from one of them to preserve the global number of links. The feature of this algorithm is that the improvement to the local routing cost it measures using the rules equals the improvement in the global routing cost, which will guarantee that the local improvement could be beneficial for the global overhead. However, a large number of reconfigurations will be executed by using this algorithm. It will result in using a large number of transaction operations such as locking operations, and finally it will have an influence on the performance of the system.

Similarly, from another work from Nitto Di et al. in [22] based on the approach in the literature [21], the algorithm in this paper additionally considers the relationship between traffic and subscriptions to achieve the same goal of the OCBR and introduces a locking mechanism to prevent the conflicts among concurrent reconfigurations. This approach could reduce some unnecessary reconfigurations, but it also would not address the requirement of rapidly adapting the frequently changing application traffic.

Consequently, the strategy in this paper considers the forwarding message both in the past and in the future, and a predicting path is introduced in this paper to reduce the reconfiguration numbers. This will address the requirement of rapidly adapting the frequently changing application traffic to reduce the overall cost.

3. Problem Statement

The strategies, mentioned in the previous section, only consider the notification cost. The strategy proposed in this paper additionally considers both advertisements and subscriptions cost during forwarding and processing in the system,

assuming that the system uses the subscription-based routing algorithm with advertisements.

Based on the above consideration, the problem we discuss in this paper can be defined as follows. In the condition of frequently changing application traffic and considering the traffic cost of notifications, subscriptions, and advertisements, to minimize the overall traffic cost by adapting the topology of the overlay network in publish/subscribe system. We call it publish/subscribe overlay optimization for frequently changing application traffic (*PSOO-FCAT*) problem.

In the following, we formalize the *PSOO-FCAT* problem. In this paper, we consider the content-based publish/subscribe system, consisting of several application components and a notification service. In the notification service, an overlay network consists of brokers connected by links, which is described by an acyclic graph $G = (V, E)$, where V denotes a set of brokers and E denotes a set of links. The notification forwarding algorithm actually creates a separate spanning tree ST_n in the overlay network G for every notification n according to the subscription routing table (SRT); the advertisement forwarding algorithm creates a spanning tree AST_a in G for every advertisement a according to the advertisement routing table (ART); the subscription forwarding algorithm creates a spanning tree SST_s in G for every subscription s according to the SRT and the ART. All the algorithm results are shown as follows:

$$\begin{aligned}
&\text{Notification Forwarding Algorithm}\\
&(n, \mathrm{SRT}) \longrightarrow \mathrm{ST}_n\\
&\text{Advertisement Forwarding Algorithm}\\
&(a, \mathrm{ART}, G) \longrightarrow \mathrm{AST}_a\\
&\text{Subscription Forwarding Algorithm}\\
&(s, \mathrm{ART}, \mathrm{SRT}, G) \longrightarrow \mathrm{SST}_s.
\end{aligned} \tag{1}$$

In the publish/subscribe system, the main cost is generated from the process that a message, such as the notification, is sent through a link arising a $\mathrm{cost}_{\mathrm{message}}(\mathrm{edge})$ and processed by the broker arising a $\mathrm{cost}_{\mathrm{message}}(\mathrm{vertex})$. So we consider both costs for every notification, advertisement, and subscription during the system functioning. Actually, we only need to regard the vertexes and the edges in the spanning tree as the cost function argument, for the reason that the message will be distributed along the whole spanning tree. N is a set of the notification that needs to be distributed by the producers; A is a set of the advertisements that needs to be distributed by the producers; S is a set of the subscriptions that needs to be distributed by the consumers. All the cost functions are shown as follows:

$$\begin{aligned}
\mathrm{cost}(N, \mathrm{SST}) &= \sum_{n \in N} \left(\sum_{v \in V(\mathrm{ST}_n)} \mathrm{cost}_n(v) + \sum_{e \in E(\mathrm{ST}_n)} \mathrm{cost}_n(e) \right),\\
\mathrm{cost}(A, \mathrm{AST}, G) &= \sum_{a \in A} \left(\sum_{v \in V(\mathrm{AST}_a)} \mathrm{cost}_n(v) + \sum_{e \in E(\mathrm{AST}_a)} \mathrm{cost}_n(e) \right),\\
\mathrm{cost}(S, \mathrm{AST}, \mathrm{SST}, G) &= \sum_{s \in S} \left(\sum_{v \in V(\mathrm{SST}_s)} \mathrm{cost}_s(v) + \sum_{e \in E(\mathrm{SST}_s)} \mathrm{cost}_s(e) \right).
\end{aligned} \tag{2}$$

The overall cost for distributing all notifications, advertisements, and subscriptions in publish/subscribe system is given by

$$\begin{aligned}
&\mathrm{cost}(N, A, S, \mathrm{AST}, \mathrm{SST}, G)\\
&\quad = \mathrm{cost}(N, \mathrm{SST}) + \mathrm{cost}(A, \mathrm{AST}, G)\\
&\qquad + \mathrm{cost}(S, \mathrm{AST}, \mathrm{SST}, G).
\end{aligned} \tag{3}$$

With the above concept, we can formally define the PSOO-FCAT problem.

Definition 1 (PSOO-FCAT problem). Define a stage: an overlay network $G = (V, E)$; a notification set N with the notification forwarding algorithm, an advertisement set A with the advertisement forwarding algorithm, and a subscription set S with the subscription forwarding algorithm; an advertisement routing table (AST) set and a subscription routing table (SRT) set; $\mathrm{cost}_{\mathrm{message}}(\mathrm{edge})$ function and $\mathrm{cost}_{\mathrm{message}}(\mathrm{vertex})$ function.

Given *a stage set.*

For each *stage lasting a time period Δt in stage set,*

> ***Find*** *the spanning tree for every notification, advertisement, and subscription on G,*
>
> ***minimize*** *the cost (N, A, S, AST, SST, G).*

End for

Note that the spanning tree finding process will be finished in a time period, which is another difference from the current work problem definition.

Obviously, the *PSOO-FCAT* problem cannot be solved in polynomial time. We can easily prove the fact that the *PSOO-FCAT* problem could be coincided with the PSODP [20] using the restriction that only considering the notification cost, in the formula (3), $\mathrm{cost}(A, \mathrm{AST}, G) = 0$ and $\mathrm{cost}(S, \mathrm{AST}, \mathrm{SST}, G) = 0$. The PSODP has been proven that the problem is not solvable in polynomial time [20]. Consequently, the *PSOO-FCAT* problem is also not solvable in polynomial time.

4. Self-Adapting Routing Strategy

In this section, we present our self-adapting routing strategy of the publish/subscribe system. For convenience, we call the notification, the subscription, and advertisement as message in the following parts.

We want to show a fact before we introduce our idea. If we have the information below, we could predict traffic accurately. (1) We only consider the notification, (2) if an advertisement contains the type of message that a client will send, the number of the message that a client will send, and the client message sending rate.

If a publish/subscribe system has these information, we will know the overlay network behavior; it means we know how the notifications are changed in the future. So we can change the overlay network structure in advance.

The basic idea is based on the above fact; however in practice the advertisement only contains the type of message that a client will send, which can provide some prediction information. In addition, we believe that there must be an association between the messages in the past and the messages in the future, so we introduce similarity rate concept.

Next, we introduce our approach; the basic target of the self-adapting routing strategy is to reduce the distance between the brokers that will forward a lot of the identical messages in the future, in which the identical messages mean that these messages all match the same filter. To figure out the message that satisfies above conditions, the broker has to train the forwarding times for each message, and it will make a prediction based on the assumption that the messages will have the similar forwarding times if the structures of these messages are similar. In addition, we only predict notifications according to advertisements in this paper, for the reason that the system does not know what and where advertisements and subscriptions will be distributed by the producers or the consumers. Finally, we could evaluate the traffic cost to guide the adaptation of the network based on this information.

As a result of the prediction, a predicting path, signifying an evaluation to the traffic condition of the local environment, was introduced in this paper. The broker finds out the message with the highest traffic cost and notifies the neighbor of this broker to execute the same process, and this process will be extended to its local environment. If the messages that were found by each broker in this series of processes are identical, these brokers form a predicting path. The reconfiguration is based on the predicting path generated and it could reduce the reconfiguration numbers effectively, because the predicting path could be regarded as a whole unit during the reconfiguration phase and it could be reused in the reconfiguration phase.

The strategy we presented in this paper is composed of three phases: prediction phase, evaluation phase, and reconfiguration phase. The training process is executed by the broker during the three phases. In the prediction phase, the broker predicts the traffic cost periodically and then generates the predicting path. The definition and the process of the predicting path generation will be discussed in Section 4.1. If the predicting path is created in this phase, the predicting path starts the evaluation phase and it is important to note that the overlay network is not actually reconfigured in this phase. In the evaluation phase, the predicting path constructs a reconfiguration according to the reconfiguration model between the two neighbors whose messages that have highest traffic cost is identical to this predicting path, and the system will determine whether this reconfiguration will be beneficial for the overall cost of the overlay network. If the reconfiguration is sensible to be executed in the system, the reconfiguration phase will start. In the reconfiguration phase, the actual reconfiguration constructed by previous phase is executed by the system.

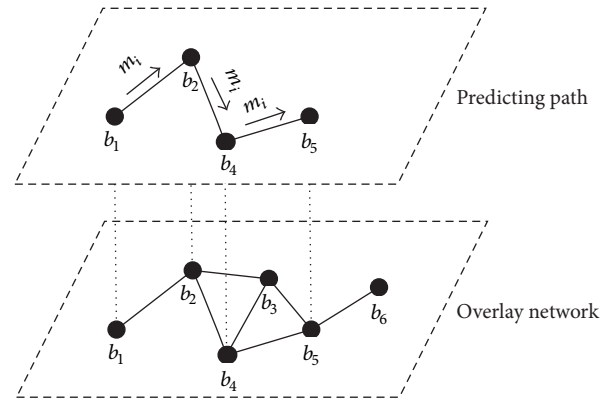

Figure 1: Predicting path.

In the following, we explain the predicting path, reconfiguration model, prediction phase, evaluation phase, and reconfiguration phase in detail.

4.1. Predicting Path

Definition 2 (predicting path). A predicting path is a path graph $P = (V', E')$ in the overlay network $G = (V, E)$, and it is a sequence of links E' that connect a sequence of brokers V'. The messages, called predicting message, which have the highest traffic cost in the predicting path and match a certain subscription (filter) are forwarded from the first broker to the last broker.

The predicting path is a finite path, which always has at least a first broker and a last broker, and it is also a simple path, which does not have the repeated brokers. Figure 1 shows a predicting path. b_1, b_2, b_3, b_4, b_5, and b_6 comprise the overlay network. b_1, b_2, b_4, and b_5 comprise the predicting path. The message m_i has the highest traffic cost in this predicting path and it forwards from b_1 to b_5.

To generate a predicting path, we must define the traffic cost at first. The traffic cost can be divided into two parts. The first part represents the training information, using the recording cost formula (4) to calculate. The other is the predicting information, using formula (6) to calculate. Consequently, the traffic cost (7) is the sum of the training cost and the predicting cost. In the following, we show these definitions.

Definition 3 (training cost). Training cost is a total processing cost of the specified filter that is recorded by each broker in counter table

$$\text{training cost}\left(\langle f, d\rangle, \text{ct}\right) = \sum_{\langle f,d\rangle \in \text{ct}} \text{cost}\left(f\right), \tag{4}$$

where f denotes filter, d denotes direct, and ct denotes count table. The broker maintains the *counter table* to count the forwarding times for each message. The structure of the *counter table* is ⟨filter, source broker⟩ *and* ⟨filter, destination broker⟩. Notifications, subscriptions, and advertisements are represented by *filter* structure in the *counter table*. The *direction* of the message is recorded as input (output) if the broker receives (distributes) the message. Obviously, formula (4) is an instance of formula (3) with the restriction that only considers the processing cost.

Before giving the predicting cost definition, we define similarity rate at first.

Definition 4 (similarity rate). Similarity rate is a degree of the similarity between the specified filters

$$\text{similarity rate}\left(\left\langle f_i, d\right\rangle, \left\langle f_j, d\right\rangle\right) = \frac{\text{count}\left(\text{equal}\left(\text{pair}\left(\langle f_i, d\rangle\right), \text{pair}\left(\left\langle f_j, d\right\rangle\right)\right)\right)}{\text{count}\left(\text{pair}\left(\left\langle f_j, d\right\rangle\right)\right)}, \tag{5}$$

where f denotes filter and d denotes direct. In the formula, the pair function returns the attribute pair set of the filters. The equal function returns true if the attribute pairs are equivalent. The count function returns the number of the attribute pairs. We believe that the structure of the filter has little influence on its forwarding times, so in this paper we only consider the identical attribute pair between the filters.

Definition 5 (predicting cost). Predicting cost is a processing cost of the filter that has the similar forwarding times:

$$\begin{aligned}
&\text{predicting cost}\left(\langle f, d\rangle, \text{ct}\right) \\
&\quad = \text{predicting cost}\left(\begin{array}{c}\langle f_i, d\rangle, \\ \max \text{similarity rate}\left(\langle f_i, d\rangle, \text{ct}\right), \\ \text{ct}\end{array}\right) \\
&\quad = \text{predicting cost}\left(\left\langle f_i, d\right\rangle, \left\langle f_j, d\right\rangle, \text{ct}\right) \\
&\quad = \sum_{\langle f_j,d\rangle \in \text{ct}} \text{cost}\left(\left\langle f_j, d\right\rangle\right) \\
&\qquad \times \text{similarity rate}\left(\left\langle f_i, d\right\rangle, \left\langle f_j, d\right\rangle\right),
\end{aligned} \tag{6}$$

where f denotes filter, d denotes direct, and ct denotes count table. The first few steps are to figure out the ⟨filter, direct⟩ that has the maximum similarity rate. In the formula, ⟨filter$_j$, direct⟩ has the maximum similarity rate relative to ⟨filter$_i$, direct⟩.

Definition 6 (traffic cost). Traffic cost is the sum of the recording cost and the predicting cost:

$$\begin{aligned}
&\text{traffic cost}\left(\langle f, d\rangle, \text{ct}\right) \\
&\quad = \text{recording cost}\left(\langle f, d\rangle, \text{ct}\right) \\
&\qquad + \text{predicting cost}\left(\langle f, d\rangle, \text{ct}\right) \\
&\quad = \sum_{\langle f,d\rangle \in \text{ct}} \text{cost}\left(f\right) + \sum_{\langle f_j,d\rangle \in \text{ct}} \text{cost}\left(\left\langle f_j, d\right\rangle\right) \\
&\qquad \times \frac{\text{count}\left(\text{equal}\left(\text{pair}\left(\langle f_i, d\rangle\right), \text{pair}\left(\left\langle f_j, d\right\rangle\right)\right)\right)}{\text{count}\left(\text{pair}\left(\left\langle f_j, d\right\rangle\right)\right)},
\end{aligned} \tag{7}$$

where f denotes filter, d denotes direct, and ct denotes count table. The broker finds out the message that has the highest traffic cost using this formula and notifies the neighbor of this broker to execute the same process, and this process will be extended to its local environment. Actually, the process of the predicting path generation is the process of the prediction. In the following, we introduce the reconfiguration model. The predicting path constructs a reconfiguration according to this reconfiguration model in the evaluation phase.

4.2. Reconfiguration Model

Definition 7 (reconfiguration model). Reconfiguration model ⟨reconfiguration unit$_a$, predicting path, reconfiguration unit$_b$⟩ is that the system removes the link between the reconfiguration unit$_a$ and predicting path and inserts the link between the reconfiguration unit$_a$ and reconfiguration unit$_b$, while the reconfiguration unit$_a$ and reconfiguration unit$_b$ are not null. If one of them or both of them are null, the reconfiguration model regards the first broker (the last broker) of the predicting path as the reconfiguration unit$_a$ (reconfiguration unit$_b$).

In the definition, the reconfiguration unit is the neighbor of the predicting path, and the reconfiguration unit$_a$ and the reconfiguration unit$_b$ are not the same neighbor. The reconfiguration unit could be the broker or the predicting path. We give some examples of the reconfiguration model. Figure 2(a) shows the reconfiguration model ⟨predicting path$_a$, predicting path$_b$, predicting path$_c$⟩. Figure 2(b) shows the reconfiguration model ⟨broker$_a$, predicting path, broker$_b$⟩. Figure 2(c) shows the reconfiguration model ⟨null, predicting path, null⟩.

4.3. Prediction Phase. The main goal of this phase is to generate the predicting path. We show the algorithm of the predicting path generation in Algorithm 1. The broker executed this algorithm periodically. In the process of description of our algorithm, we assume that messages cannot be lost.

In predicting path generation sponsor process, the broker figures out the traffic cost for each notification, subscription, and advertisement (line1~line6) using formula (7); the cal traffic cost function returns a message structure containing

predicting path generation algorithm
predicting path generation sponsor:
(1) *For each* $notification_i$*, in counter table*
(2) $result_n$ *+ = cal traffic cost(*$notification_i$*)*
(3) *For each* $subscription_i$*, in counter table*
(4) $result_s$ *+ = cal traffic cost(*$subscription_i$*)*
(5) *For each* $advertisement_i$*, in counter table*
(6) $result_a$ *+ = cal traffic cost(*$subscription_i$*)*
(7) *result =* $result_n \cup result_s \cup result_a$
(8) *predicting message = cal max traffic cost(result)*
(9) *If equal(predicting message, notified message)*
(10) *recv path list += broker*
(11) *Else*
(12) *new path list = broker*
(13) *neighbor = neighbor(predicting message direction)*
(14) *notify(neighbor, predicting message, traffic cost, path list)*

predicting path generation listener:
(1) *listen to notified predicting message*
(2) store*(predicting message, traffic cost, path list)*

Algorithm 1: Predicting path generation algorithm.

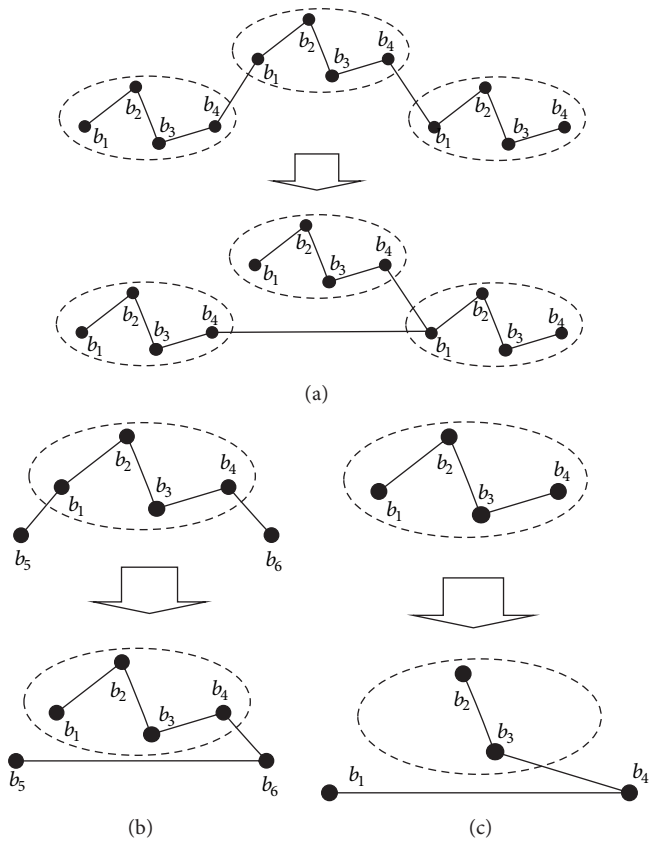

Figure 2: (a) Reconfiguration model. (b) Reconfiguration model. (c) Reconfiguration model.

the message content and the traffic cost value; result_n is structure list that records the traffic cost results.

The broker finds the predicting message (line7~line8) based on above results. It firstly gets these result lists and then selects the message that has the highest traffic cost using the max traffic cost function that returns the message structure.

If the broker has received the notified message and the predicting message is identical with the notified message (the notified message needs to satisfy the time stamp; this ensures the availability of the notified message), the broker should add its own to the received predicting path list, or else the broker should create a new list containing its own (line9~line12).

The broker uses the neighbor function to get the next broker it should notify. This process depends on the direction of the predicting message. Then the broker will send it, the predicting message and its traffic cost to the next broker (line13~line14).

In predicting path generation listener process, if the broker received a notified predicting message, it should store it to wait for generating the predicting path.

In real scenarios, when the topology of the overlay network grows, the predicting path generation process requires an exhaustive approach to notify the neighbor broker.

Once the predicting path is generated, the path list containing the predicting path will be sent to master broker. The master broker, which has the lowest traffic cost among the predicting path, is elected by the predicting path. Thus the master broker stores the path list. Finally, the predicting path will enter the evaluation phase. The master broker creation process and negotiation message are not discussed here.

We give a simple example to understand this algorithm. In Figure 1, an overlay network contains broker b_1, b_2, b_3, b_4, b_5 and b_6. Assuming that the broker b_1, b_2, b_4, b_5 have the same predicting message, the order of finding predicting message is b_2, b_5, b_1, b_4, the order of traffic cost of predicting message from low to high is b_2, b_1, b_5, b_4. The predicting message direction is b_1 to b_2, b_2 to b_4, b_4 to b_5. According to our algorithm, b_2 will first notify b_4, the master broker is b_2; b_5 do nothing, the master broker is b_2; b_1 will notify b_2, the master broker is b_2, b_4 will notify b_5, the master broker is b_2. In the end, the predicting path is b_1, b_2, b_4, b_5 that is mastered by b_2.

4.4. Evaluation Phase. In this phase, there are two steps: (1) Several reconfigurations will be constructed based on the predicting path according to the reconfiguration model. (2) The reconfigurations will be evaluated to determine whether they will be beneficial for the overall cost.

The single broker and predicting path are both regard as reconfiguration unit in this phase, so the interaction is between the reconfiguration units. The reconfiguration construction process is similar with the predicting path generation process, the difference is the former need not to calculate out the predicting path.

As shown in Algorithm 2, in reconfiguration construction sponsor process, the master broker figure out the directions of the predicting message_{rc}. The predicting message_{rc} is reserved from the predicting phase. (line1~line2) And then

reconfiguration construction algorithm
reconfiguration construction sponsor:
(1) *Direct1 = Direction(predicting* message_{rc}*)*
(2) *Direct2 = NegDirection(predicting* message_{rc}*)*
(3) $Notify_{Direct1}$ *(predicting* message_{rc}*)*
(4) $Notify_{Direct2}$ *(predicting* message_{rc}*)*

reconfiguration construction listener:
(1) *listen to notified predicting* message_{rc}
(2) *If equal(predicting* message_{rc}*, notified* message_{rc}*)*
(3) *Rely to sponsor*

Algorithm 2: Reconfiguration construction algorithm.

the master broker notifies the corresponding brokers. (line3~line4)

In reconfiguration construction sponsor process, the broker listens to whether the other broker has to request the reconfiguration construction. If the notified predicting message_{rc} equaled with its own predicting message_{rc}, the broker should rely to the notified master broker that it satisfies the condition of the reconfiguration.

However, this reconfiguration is generated from local environment, so this reconfiguration may lead to an overall traffic cost increase. To prevent it, the reconfiguration generated must be evaluated to determine whether the reconfiguration will be beneficial for the overall cost of the overlay network. The master broker will ask every broker in the predicting path to evaluate their own traffic cost using formula (7) assuming that the reconfiguration has executed, and these brokers will send the evaluation result to the master broker after evaluation. In the end, the master broker will make a decision whether the reconfiguration is executed. After these processes, if the reconfiguration is sensible to execute, the reconfiguration starts the reconfiguration phase. Figures 2(a), 2(b), and 2(c) are several cases of the reconfiguration.

4.5. Reconfiguration Phase. In the reconfiguration phase, the actual reconfiguration constructed by previous phase will be executed by the system. There might be some reconfigurations stored in the same one master broker, so the master broker uses a reconfiguration list to store the reconfigurations and the reconfiguration that has the greatest benefits will be in the front of the list. The master broker will pick the reconfiguration from the front side of the list to execute.

Since the same broker may be involved in different reconfigurations, the system has to apply a locking strategy to avoid reconfiguration conflicts before the reconfiguration is executed.

(i) At the beginning of the reconfiguration, the master broker sends locking instructions to the brokers involved in this reconfiguration.

(ii) If the broker has received the locking instruction, the broker will switch to the locking state and will send a confirmation instruction back to the master broker.

(iii) If the master has received the confirmation instructions from all the brokers involved in the reconfiguration, the master will send reconfiguration instructions to these brokers.

(iv) If these brokers have received the reconfiguration instructions, they will reconfigure the topology of the overlay network according to the reconfiguration instruction. When the broker has finished reconfiguration, it will send the complete instruction to the master broker.

(v) If the master broker receives the complete instructions from all the brokers involved in the reconfiguration and it will send the unlocking instructions to these brokers and these brokers switch to the normal state.

We should note that the brokers involved in the reconfiguration do not mean all brokers in the reconfiguration, and it means that the brokers should insert or remove the link from another broker.

5. Experiments

In this section, we simulate the experiments on ProtoPeer [3]. The ProtoPeer is a prototyping toolkit that allows for switching between the event driven simulation and live network deployment without changing any of the application code [3]. The most outstanding feature of the ProtoPeer is that the experiment results recording process will be not influenced by the other running process, so this ensures that the results are relatively accurate. The experiments were performed in live deployment mode. To verify our idea, we only use simple routing here to implement our simulations.

We designed four simulation experiments to evaluate the performance of our strategy.

(1) To validate whether the strategy proposed in this paper has the ability to reduce the traffic cost of the overlay network.

(2) To validate whether the strategy proposed in this paper has the ability to reduce the traffic cost of the overlay network, in the condition that the application traffic changes frequently.

(3) To validate whether the strategy proposed in this paper has the ability to reduce the traffic cost of the overlay network, in the condition that the message relevancy is low.

(4) To validate whether the strategy proposed in this paper has less reconfiguration numbers, in the condition that the application traffic changes frequently.

We also have performed simulation: OCBR strategy in the literature [21] and OCBR-E strategy in the literature [22] where "E" is an acronym of extension in "OCBR-E", as we will compare the result performed by our strategy simulation to both OCBR and OCBR-E results. The cost function to calculate the experiments result is the average value among our cost function and OCBR cost function.

We designed the simulation experiments based on our actual application scenarios. We used Brite [23] to create the topology and generated an overlay network topology with 5000 nodes randomly and chose 1000 nodes randomly to deploy the broker on the topology. The other simulation parameters are as follows.

Subscriptions. The name/value pairs are the structure of our simulation data model; we randomly generated 1000 different types of subscription as S_1 and 200 different types of subscription as S_2. Then we generated 400 different types of subscription as S_3 based on S_2 and 400 different types of subscription as S_4 based on S_3, since we want to enhance the similarity rate among the elements of sample collection. The total number of the subscription types is 2000.

Notification and Advertisements. We generated 10000 different types of notifications based on the 1500 generated subscriptions and generated 8000 advertisement according to 8000 different types of notification.

Producers and Consumers. We deployed 200 producers, 200 consumers, and 50 pair of the $\langle \text{producer}, \text{consumer} \rangle$ that connected to brokers randomly. Each producer published 40 types of notification, and each consumer subscribed 8 types of notification.

Sending Frequency. A notification is sent every 2 simulation ticks by a producer.

Simulation Stage. We designed 10 different stages in which every producer publishes different notifications and consumer subscribes different notification in different stages. We divided the generated notifications into 10 different collections and divided the generated subscriptions into 10 different collections. As a result, 10 different stages for producer map to 10 different notification collections and 10 different stages for consumer map to 10 different subscription collections.

$Cost_{message}(edge)$ and $Cost_{message}(edge)$. The values of both cost functions were randomly chosen from 1 to 10.

5.1. Adaptivity. In this simulation experiment, our goal is to validate whether the strategy proposed in this paper has the ability to reduce the traffic cost of the overlay network, so we only use one stage in this experiment. The simulation experiment was performed in 30000 ticks. The results were recorded by us at every 500 ticks. The average value was calculated as shown in Figure 3, and the standard deviations are given in Table 1.

In Figure 3, the PSOO-FCAT cost is 50253 at tick 0. The PSOO-FCAT cost is not changed before tick 1500 and the change point is in period from tick 1500 to 2000, because the PSOO-FCAT was generating the predicting path. The PSOO-FCAT cost decreases sharply from tick 1500 to 2000, which indicates that the PSOO-FCAT started the actual reconfigurations. The PSOO-FCAT cost decreases steadily from tick 2000 to 8000, which indicates that the application traffic was gradually adapting the topology of the overlay

Table 1: Standard deviations of adaptivity.

PSOO-FAT: 1.182, 1.117, 1.197, 0.985, 1.119, 1.151, 1.077, 0.917, 0.974, 0.950, 1.167, 0.971, 0.838, 1.133, 1.130, 0.883, 0.908, 0.900, 1.112, 1.035, 1.069, 0.809, 0.843
OCBR: 1.059, 1.165, 0.843, 1.163, 1.122, 1.055, 0.986, 0.971, 0.951, 1.050, 0.817, 0.981, 0.929, 0.909, 0.889, 1.037, 0.871, 1.022, 0.892, 1.014, 1.154, 1.051, 1.061
OCBR-E: 1.175, 0.865, 0.830, 1.080, 1.093, 1.169, 0.906, 1.193, 0.848, 0.925, 1.141, 1.127, 1.012, 1.041, 0.919, 1.168, 0.855, 0.960, 0.955, 0.896, 1.057, 1.097, 1.066

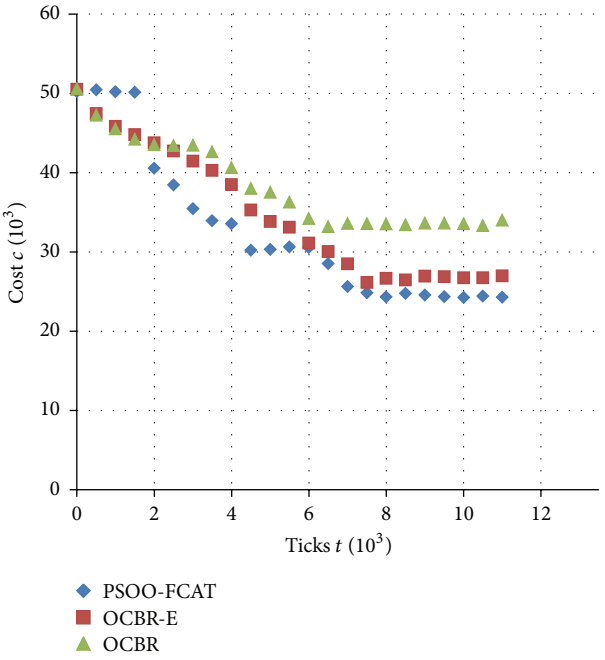

Figure 3: The result of adaptivity.

network in this period. The PSOO-FCAT cost tends to be steady after tick 8000; it means that the application traffic has been basically adapted the topology of the overlay network. As a result, the strategy proposed in this paper has the ability to reduce the traffic cost of the overlay. Compared to other strategies, the OCBR and the OCBR-E have the same ability to reduce the traffic cost of the overlay network. After tick 6000, the PSOO-FCAT cost and the OCBR-E cost basically have the similar cost value, which indicates that during a long period, both PSOO-FCAT and the OCBR-E could reduce the traffic cost at the same level.

5.2. Adaptivity with Frequently Changing Application Traffic. In this simulation experiment, our goal is to validate whether the strategy proposed in this paper has the ability to reduce the traffic cost of the overlay network, in the condition that the application traffic changes frequently. Different from the previous experiment, we limited a stage which is switched to the next stage in every 4000 ticks. The simulation experiment

Table 2: Standard deviations of adaptivity with frequently changing application traffic.

PSOO-FAT: 0.977, 0.944, 0.999, 0.926, 1.120, 1.069, 1.056, 0.949, 0.931, 0.980, 1.196, 0.980, 1.044, 0.931, 1.050, 0.826, 1.018, 0.802, 0.976, 0.911, 0.938, 1.150, 0.903
OCBR: 0.909, 1.000, 0.877, 1.088, 1.128, 1.151, 0.971, 0.870, 1.194, 1.073, 0.811, 1.004, 0.845, 0.849, 1.015, 0.971, 0.839, 1.056, 0.938, 0.929, 0.907, 0.856, 1.062
OCBR-E: 1.082, 0.800, 1.003, 1.168, 0.900, 0.896, 0.972, 1.002, 1.071, 1.051, 1.155, 0.834, 0.887, 0.891, 0.855, 0.987, 1.029, 0.936, 1.196, 1.030, 1.051, 1.045, 0.850

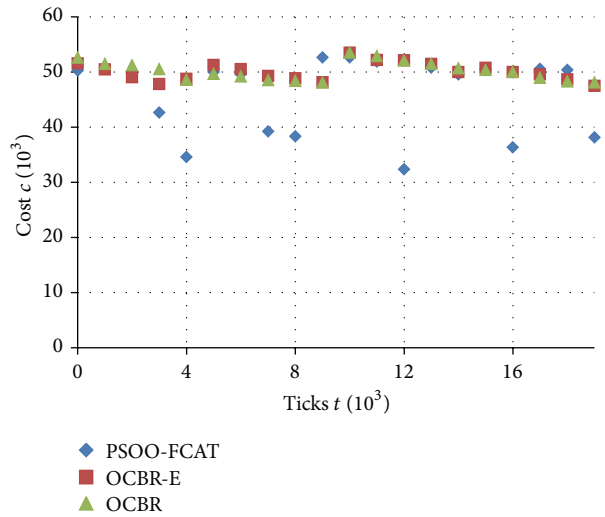

Figure 4: The result of adaptivity with frequently changing application traffic.

was performed in 20000 ticks. The results were recorded by us in every 1000 ticks. The average value was calculated as shown in Figure 4, and the standard deviations are given in Table 2.

In Figure 4, the PSOO-FCAT cost is 51193 at tick 0. The PSOO-FCAT cost is not changed before tick 2000, because the PSOO-FCAT was generating the predicting path. The PSOO-FCAT cost decreases from tick 2000 to 4000, which indicates that the PSOO-FCAT started the actual reconfigurations and the application traffic was gradually adapting the topology of the overlay network in this period. The PSOO-FCAT cost increases sharply from tick 4000 to 5000, because the stage was switched to the next stage in tick 4000. In the next stage period, the change of the PSOO-FCAT cost is similar with the previous stage period. Compared to other strategies, the OCBR cost and the OCBR-E cost that have been reduced are not more obvious than the PSOO-FCAT cost, because the PSOO-FCAT could predict the application traffic to reduce the traffic cost. In the next stage period, the change of the OCBR cost and the OCBR-E cost is similar with the previous stage period. The experiment results show that our strategy has the ability to reduce the traffic cost of the overlay network, in the condition that the application traffic changes frequently, and the efficiency was obvious.

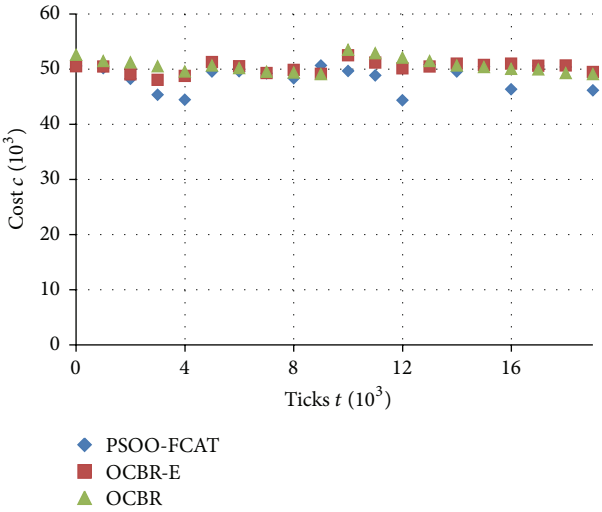

Figure 5: The result of adaptivity with low message relevancy.

5.3. Adaptivity with Low Message Relevancy. In this simulation experiment, our goal is to validate whether the strategy proposed in this paper has the ability to reduce the traffic cost of the overlay network, in the condition that the message relevancy is low. Different from the previous experiment, subscriptions simulation parameter is changed. The new subscriptions are generated as follows. We randomly generated 1600 different types of subscription as S_1 and 200 different types of subscription as S_2. Then we generated 200 different types of subscription as S_3 based on S_2. The total number of the subscription types is 2000. The notification generation rule was not changed. Consequently, the message relevancy is becoming low. The simulation experiment was performed in 20000 ticks. The results were recorded by us in every 1000 ticks. The average value was calculated as shown in Figure 5, and the standard deviations are given in Table 3.

In Figure 5, the PSOO-FCAT cost is 50534 at tick 0. The PSOO-FCAT cost is not changed before tick 2000, because the PSOO-FCAT was generating the predicting path. The PSOO-FCAT cost decreases from tick 2000 to 4000, which indicates that the PSOO-FCAT started the actual reconfigurations and the application traffic was gradually adapting the topology of the overlay network in this period, but the change is not obvious compared with the previous experiment, because the message relevancy is lower than the previous experiment; thus the similarity rate is becoming low. The PSOO-FCAT cost increases sharply from tick 4000 to 5000, because the stage was switched to the next stage in tick 4000. In the next stage period, the change of the PSOO-FCAT cost is similar with the previous stage period. Compared to other strategies, the OCBR cost and the OCBR-E cost that have been reduced are not more obvious than the PSOO-FCAT cost, because the PSOO-FCAT could predict the application traffic not only depending on message relevancy but the advertisement. In the next stage period, the change of the OCBR cost and the OCBR-E cost is similar with the previous stage period. The experiment results show that our strategy has the ability to reduce the traffic cost of the overlay network, in the condition that the message relevancy is low.

Table 3: Standard deviations of adaptivity with low message relevancy.

PSOO-FAT: 1.123, 1.155, 0.862, 0.800, 0.889, 0.912, 1.093, 0.965, 1.142, 0.877, 0.818, 0.864, 0.870, 1.170, 0.951, 0.970, 0.994, 1.032, 1.058, 0.800, 0.818, 0.946, 0.897
OCBR: 1.133, 0.943, 1.082, 1.179, 1.002, 1.178, 1.119, 0.834, 0.837, 1.093, 1.041, 0.865, 1.187, 0.804, 0.906, 0.920, 0.877, 0.846, 0.852, 1.120, 1.176, 1.145, 0.876
OCBR-E: 0.806, 0.842, 1.188, 0.900, 0.956, 1.026, 1.151, 1.025, 0.803, 0.974, 1.124, 1.134, 0.941, 0.866, 0.913, 0.950, 0.904, 0.939, 1.189, 0.943, 1.037, 0.914, 0.859

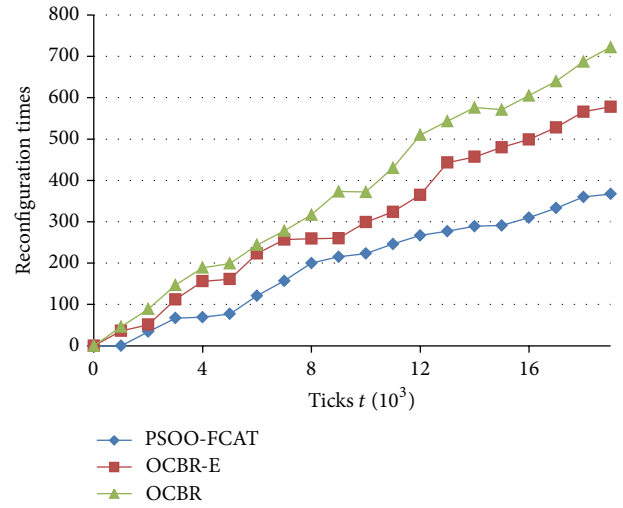

Figure 6: The result of reconfiguration numbers.

5.4. Reconfiguration Numbers. In this simulation experiment, our goal is to validate whether the strategy proposed in this paper has less reconfiguration numbers, in the condition that the application traffic changes frequently. We selected 50 ticks to use as the time period Δt training the traffic information in OCBR strategy, OCBR-E strategy, and our strategy. We recorded that the reconfiguration numbers that the traffic costs decreased at the same point. The other simulation parameters are same to the simulation experiment in Section 5.2.

In Figure 6, the PSOO-FCAT reconfiguration numbers are 0. The PSOO-FCAT reconfiguration numbers still remain 0 at tick 1000, because the PSOO-FCAT was generating the predicting path and it did not actually reconfigure the topology of the overlay network. The PSOO-FCAT reconfiguration numbers increase after tick 1000, which indicates that the PSOO-FCAT started the actual reconfigurations. Compared to other strategies, we can see that our strategy could reduce the traffic cost in less reconfiguration numbers. Meanwhile, in the same experiment condition, the previous simulation experiment shows that our strategy has the ability to reduce the traffic cost of the overlay network. So the strategy

proposed in this paper has less reconfiguration numbers, in the condition that the application traffic changes frequently.

6. Conclusion

In this paper, we propose a self-adapting routing strategy for frequently changing application traffic in content-based publish/subscribe system. The strategy firstly trains the traffic information and then uses this training information to predict the application traffic in the future. Finally, the strategy reconfigures the topology of the overlay network based on this predicting information to reduce the overall traffic cost. A predicting path is also introduced in this paper to reduce the reconfiguration numbers in the process of the reconfigurations. The experimental results show that the strategy could reduce the overall traffic cost of the publish/subscribe system in less reconfigurations. In the future work, we will implement our strategy in our prototype system.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003.

[2] G. Mühl, L. Fiege, and P. Pietzuch, *Distributed Event-Based Systems*, vol. 1, Springer, 2006.

[3] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, "ProtoPeer: a P2P toolkit bridging the gap between simulation and live deployement," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (ICST '09)*, 2009.

[4] S. Zeng, L. Li, and D. Liao, "A distributed flow rate control algorithm for networked agent system with multiple coding rates to optimize multimedia data transmission," *Mathematical Problems in Engineering*, vol. 2013, Article ID 983637, 11 pages, 2013.

[5] D. J. Dubois, Y. Bando, K. Watanabe, and H. Holtzman, "Lightweight self-organizing reconfiguration of opportunistic infrastructure-mode WiFi networks," in *Proceedings of the 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO '13)*, 2013.

[6] P. Eugster, "Type-based publish/subscribe: concepts and experiences," *ACM Transactions on Programming Languages and Systems*, vol. 29, no. 1, article 6, 2007.

[7] M. Diallo, V. Sourlas, P. Flegkas, S. Fdida, and L. Tassiulas, "A content-based publish/subscribe framework for large-scale content delivery," *Computer Networks*, vol. 57, no. 4, pp. 924–943, 2013.

[8] E. Christian, D. Cotroneo, and S. Russo, "On reliability in publish/subscribe services," *Computer Networks*, vol. 57, no. 5, pp. 1318–1343, 2013.

[9] R. Feng, G. Wang, and K. Huang, "Research on test and training enabling architecture (TENA)," *Acta Simulata Systematica Sinica*, 2004.

[10] L. Shou-Chih, "Efficient content-based publish/subscribe systems over peer-to-peer networks," *Journal of Internet Technology*, vol. 13, no. 5, pp. 713–724, 2012.

[11] K. L. Morse, *Interest Management in Large-Scale Distributed Simulations*, Information and Computer Science, University of California, Irvine, Calif, USA, 1996.

[12] Z. Yaxiong and J. Wu, "Building a reliable and high-performance content-based publish/subscribe system," *Journal of Parallel and Distributed Computing*, vol. 73, no. 4, pp. 371–382, 2013.

[13] J. Chen, M. Díaz, B. Rubio, and J. M. Troya, "PS-QUASAR: a publish/subscribe QoS aware middleware for Wireless Sensor and Actor Networks," *Journal of Systems and SOftware*, vol. 86, no. 6, pp. 1650–1662, 2013.

[14] H. Aino, J. Törnroos, and M. Elo, "Network process analysis: an event-based approach to study business network dynamics," *Industrial Marketing Management*, vol. 42, no. 8, pp. 1213–1222, 2013.

[15] L. Xia, Q. S. Jia, and X. R. Cao, "A tutorial on event-based optimization: a new optimization framework," *Discrete Event Dynamic Systems*, 2013.

[16] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni, "TERA: topic-based event routing for peer-to-peer architectures," in *Proceedings of the International Conference on Distributed Event-Based Systems (DEBS '07)*, pp. 2–13, June 2007.

[17] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *Networked Group Communication*, Springer, 2001.

[18] A. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*, Springer, 2001.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," in *Proceedings of the Applications, Technologies, Architectures, and Protocols for Computers Communications (ACM SIGCOMM '01)*, pp. 149–160, August 2001.

[20] M. A. Jaeger, H. Parzyjegla, G. Mühl, and K. Herrmann, "Self-organizing broker topologies for publish/subscribe systems," in *Proceedings of the Symposium on Applied Computing*, pp. 543–550, March 2007.

[21] M. Migliavacca and G. Cugola, "Adapting publish-subscribe routing to traffic demands," in *Proceedings of the Inaugural International Conference on Distributed Event-Based Systems (DEBS '07)*, pp. 91–96, New York, NY, USA, June 2007.

[22] E. Nitto Di, D. J. Dubois, and R. Mirandola, "Overlay self-organization for traffic reduction in multi-broker publish-subscribe systems," in *Proceedings of the 6th International Conference on Autonomic Computing (ICAC '09)*, pp. 61–62, Barcelona, Spain, June 2009.

[23] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: an approach to universal topology generation," in *Proceedings of the 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '01)*, pp. 346–353, Cincinnati, Ohio, USA, August 2001.