MicroPost: 분산형 소셜 애플리케이션을 위한 효율적인 이벤트 통지 아키텍처의 설계

MicroPost: The Design of an Efficient Event Notification Architecture for Distributed Social Applications

배준현, Joonhyun Bae*, 김상욱, Sangwook Kim**

Abstract Emerging social networking services provide a new paradigm for human-to-human communication. However, these services are centralized and managed by single service provider. In this paper, we propose *MicroPost*, a decentralized event notification service architecture for social applications based on publish/subscribe model. In our design space, event brokers are structured as an overlay network which provides the substrate of distributed peer-to-peer lookup service for storing and retrieving subscriptions with hashed keys. Event clients interact with event brokers to publish or subscribe social messages over the wide-area network. Using XML standards, we present an efficient algorithm to forward events for rendezvous-based matching in this paper. In our design space, the cost of routing is $O(\omega \log_k N)$, where N is the number of event brokers, ω is the number of meta-data obtained from event messages, and k is a constant, which is selected by our design, to divide the identifier space and to conquer the lookup of given key. Consequently, what we achieved is an asynchronous social messaging service architecture which is decentralized, efficient, scalable, and flexible.

Keywords: Social Networking Service, Content-Based Publish/Subscribe, Structured Broker Overlay, Distributed Hash Table, Rendezvous-based Matching and Routing Algorithm

1. Introduction

Emerging social networking services provide a new paradigm for human-to-human communication. Such a service, so-called *microblogging*, has become popular as means of disseminating personal information to an implicit group of people, e.g., Twitter, Jaiku, and Pownce[1]. However, these services are centralized and managed by single service provider. It means that the communication is limited and controlled. Hence, it is valuable to have an infrastructure for decentralized social communication.

In this paper, we propose *MicroPost*, a novel event notification service architecture for distributed social applications based on publish/subscribe model. The main idea of this paper is to exploit the expressiveness of Content-Based Publish/Subscribe[2-5] model to enable human-to-human communication. In addition, we aim to design a decentralized, efficient, scalable, and flexible architecture for social messaging service. Distributed Hash Table(DHT)[6,7] is our choice of design to achieve efficient routing of subscriptions and publications for rendezvous-based matching and notifications. Using XML-based event types, we also achieve more flexible architecture which is able to be extended to various application areas, i.e., photo-sharing, collaborative learning, or large-scale peer-to-peer auction, etc.

*주저자 : 경북대학교 전자전기컴퓨터학부 석사과정 e-mail: jhbae@cs.knu.ac.kr
**교신저자 : 경북대학교 전자전기컴퓨터학부 교수 e-mail: swkim@cs.knu.ac.kr

The main contribution of this paper is to introduce an efficient algorithm for forwarding events for rendezvous-based matching. With a global hash function, we utilize the meta-data of event messages for indexing and routing. In this paper, we show and prove that the cost of routing is $O(\omega \log_k N)$, where N is the number of event brokers, ω is the number of meta-data in event messages, and k is a constant which is selected by our design.

This paper is organized as follows. In Section 2, we review previous researches and existing solutions. In Section 3, we describe the system model in our design space. In Section 4, we present an efficient algorithm for routing events for rendezvous-based matching. In Section 5, we evaluate our approach with regard to the efficiency of routing algorithm. Finally, we conclude this paper in Section 6.

2. Related Work

In this section, we review previous researches in terms of content-based publish/subscribe model and distributed hash table. Then, we investigate existing solutions related to our approach.

Content-based Publish/Subscribe has been an active research area[2-5]. SIENA[2] provided a fundamental contribution to this area. It is based on a brokers' network architecture, and the subscription routing is based on the filtering algorithm exploiting containment relationships. HERMES[3] is a distributed event-based middleware. It utilize rendezvous-based routing over an overlay network. Especially, its event model is designed by XML Schema[13], and the subscription model is represented by XPath[14]. These XML-based modeling conforms to our approach in terms of flexibility. SCRIBE[5] is built on a structured overlay network infrastructure, i.e., Pastry[7]. It allows an efficient and scalable message matching and routing in application layer. However, its subscription model is not *content-based* but *topic-based*. Because the topic-based subscription model is less expressive, it is not appropriate for our requirements. REBECA[4] is an object-oriented notification service framework which provides a generic routing engine. It is used in many research projects to support configurable systems, mobility of users, and security aspects. Particularly, supporting mobile client is a highly valuable feature of our targeted architecture.

To improve the efficiency and scalability of content-based pub/sub systems, many peer-to-peer approaches are proposed[8-9]. These approaches are mainly based on Distributed Hash Table[6,7], which is a substrate of structured peer-to-peer overlay network. The advantages of DHT-based structured P2P overlay, such as decentralization, scalability, fault-tolerance, and self-organization, are essential in our design space. Among various DHT systems, we enhanced Chord[6] because of its simplicity and popularity.

The matching problem is also an important part in our design space. Although there are numerous algorithms[10-12], we feel that none of the various algorithms solves our requirements in the literature of social messaging. In order to realize our approach, we need string-attribute-friendly matching and routing algorithms. Although two researches, DHTStrings[11] and PastryStrings[12], cope with similar problems, these researches focus on the problem of rich queries. Therefore, we devised a new algorithm for matching and routing content in the context of *folksonomies*, so-called *social-tagging*[1]. We argue that this approach is highly effective in our targeted service architecture, where information is controlled by human beings.

3. System Model

In this section, we describe the MicroPost, our proposed architectural model of social messaging service based on publish/subscribe paradigm.

3.1 System Architecture

In our approach, we assume that all the people act as both *publishers* and *subscribers*. That is to say, anyone can produce a new message in the system and consume messages created by other people.

Figure 3.1 shows the components of event notification service infrastructure for the MicroPost. It is composed of *event clients* and *event brokers*. An

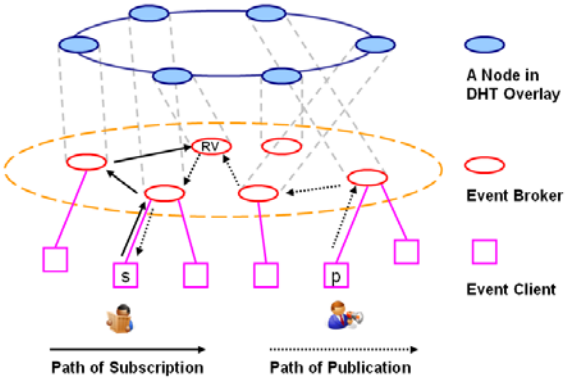


Figure 3.1 Event Notification Service Infrastructure

event client is a user interface for people to issue and fetch *events* to and from the system. An *event broker* is a system interface for brokering the events to and from either the event client or the other event broker. We assume that all the event brokers are distributed over the wide-area DHT overlay network. When a subscriber submits a subscription, it is forwarded to a rendezvous node by the event broker he/she is connected with. A publication, which is issued by a publisher, is also routed to a rendezvous node for matching with subscriptions, and is notified to the subscribers who are interested in.

Figure 3.2 shows the architectural design of our system. In physical network layer, we divide the network connections into two types. The connection among brokers is regarded as reliable, and the connection between client and broker is deemed as being connected intermittently, taking mobile clients into accounts. In overlay network layer, the topology of event brokers' network is structured on the basis of Chord, one of the most popular DHT system. The publish/subscribe layer provides interfaces and libraries to application layer allowing the functionality of publishing or subscribing social messages. It is desirable for various applications to appear in the application layer on top of the pub/sub service layer.

3.2 Event Model

An *event type* τ is a structure of an application-specific event defined by XML Schema[13]. XML Schema Language provides a means for defining the structure, content, and semantics of XML documents.

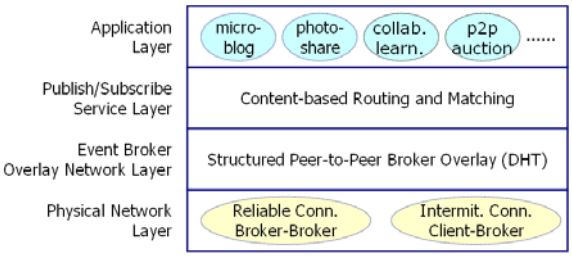


Figure 3.2 Hierarchical System Architecture

An event ε is an XML document that conforms to the event type τ. A publisher can publish an event by submitting an XML document to the system.

A subscriber can subscribe selected events by specifying his/her interests according to the structure of event type. A subscription σ is expressed by XPath[14]. XPath is an XML Path Language for addressing parts of an XML document, providing basic facilities for manipulation of strings, numbers and boolean values. A subscription can be represented by a path expression that returns a boolean value. For example, Figure 3.3(a) and 3.3(b) show an example of subscription and publication messages in XML format.

Event brokers store all the subscriptions collaboratively. Whenever an event broker receives a new event from an event client, it should find the subscribers interested in the event. An event ε *matches* a subscription σ, *if and only if* the boolean path expression of σ returns true when it compiled on ε. After the system finds the interested group of people, it notifies the event to the subscribers. Figure 3.4 depicts the process of matching an event with subscriptions in rendezvous-based routing.

3.3 Social Messaging Model

In our design space, social messaging is about human-to-human communication. Let $B=\{b_1, \cdots, b_n\}$ be a set of event brokers, and let $P=\{p_1, \cdots, p_m\}$ be a set of people who is connected to an event broker with an event client. Any person p_i, who is connected to b_j, which is denoted by $p_i \Rightarrow b_j$, should have the functionality of publishing and subscribing messages to and from

any other person in the network, i.e., $\forall (p_x \Rightarrow b_y), p_x \in P, b_y \in B,$.

```
<?xml version="1.0" encoding="UTF-8"?>
<micropost>
  <publish>
    <from>jhbae@knu.edu</to>
    <to>joonion@kmaru.com</to>
    <body>Hello, World!</body>
    <tag>greeting</tag>
  </publish>
</micropost>
```

(a) publication message

```
<?xml version="1.0" encoding="UTF-8"?>
<micropost>
  <subscribe>
    <by>joonion@kmaru.com</by>
    <where>//from[text()='jhbae@knu.edu']</where>
  </subscribe>
</micropost>
```

(b) subscription message

Figure 3.3 An example of events in XML format

We categorized the model of communication as the following:

i) **One-to-One Messaging:** When a $p_i \Rightarrow b_x$ wants to subscribe the messages from $p_j \Rightarrow b_y$, p_i submits a subscription σ containing the predicate of XPath expression like "//from[text()='$p_j.id$']." On the other side, $p_i \Rightarrow b_x$ can receive all the messages to herself by specifying a predicate like "//to[text()='$p_i.id$']."

ii) **Group Messaging:** A group of people also can communicate in the same manner as one-to-one messaging. With a virtual identifier, g_v, all the members of $G = \{p_1, \ldots, p_q\}$ can subscribe the events directed to g_v, with a predicate of "//to[text()='$g_v.id$']." Then, anyone can send a message to this group by specifying the identifier of g_v as a recipient in a message.

iii) **Interest-based Messaging:** In our messaging model, by adding 〈tag〉 element to an event type, we can share many kind of information, i.e., urgent news, personal opinions, or curious questions, with wide variety of people who have the same tastes as us. A person can subscribe selected messages from all the people by specifying the constraints on the *tags*, i.e., user-defined meta-data of the messages. As an example, when a $p_i \Rightarrow b_x$ is interested in all the messages related with "Social Messaging," she would subscribe with a predicate like the following: //tag[text()='Social' and text()='Messaging']. Then, when $p_j \Rightarrow b_y$ publishes a message tagged by 'Social,' 'Messaging,' and 'Service,' $p_i \Rightarrow b_x$ is notified with the message.

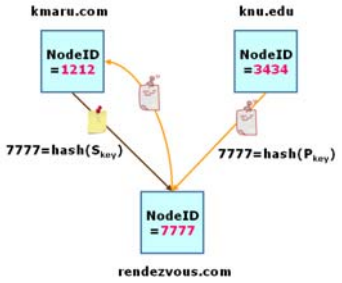


Figure 3.4 An example of notification

4. Matching and Routing

In this section, we present algorithms for routing subscriptions and publications for rendezvous-based matching over a structured broker overlay network.

4.1 Matching in an Event Broker

The problem of matching an event with a group of subscriptions is formally stated as follows:

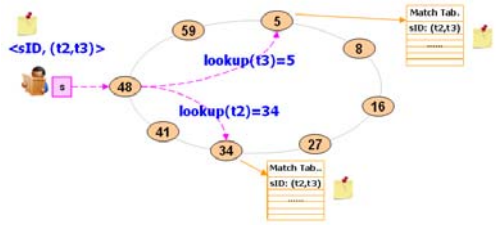
Given an event ε and a set of subscriptions S, find a subset S' of S which fulfills the condition that all the elements of S' cover the given event ε, namely:

$$S' = \{\sigma \mid \sigma \in S, \sigma \text{ covers } \varepsilon\} \quad (1)$$

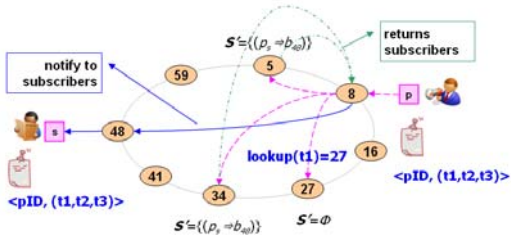
In naive approach, we can test if ε matches σ for each subscription in S. Then, the cost of matching is $O(|S|)$ intuitively. It is too expensive for large-scale social applications, where the number of subscriptions is the same as that of Internet users. Although there are numerous efficient matching algorithms[10], we take a different approach on the basis of our assumptions: The content itself contains useful information to match with certain subscriptions.

As explained above, we already defined that the matching in our design space is to test if the predicate of boolean XPath expression, specified by the subscribers, returns true. Hence, to minimize the candidate subscriptions, we can classify subscriptions

by hashing them with the values of text nodes appeared in the predicates.



(a) Subscription Forwarding



(b) Publication Forwarding and Matching

Figure 4.1 Routing and Matching Process

Since the three types of element nodes(〈from〉, 〈to〉, and 〈tag〉) are tested for filtering messages in our previous example, the system should push subscriptions into the hash table according to the hashvalues in the predicates of path expression. When a new message arrives at a broker, the broker checks only the subscriptions in the buckets of the hash table according to the hash values of the text nodes in that message.

With a consistent hash function, like *SHA-1*, we are guaranteed that there are few, if any, collisions. Hence, for each text node in an *atomic subscription* containing equality predicates, the cost of matching is $O(1)$. Even if there are several subscriptions on the same bucket in the case that several subscriptions have the same value in their predicates, the cardinality of S which is to be tested is small enough, that is to say, $|S| \approx |S'|$.

4.2 Rendezvous-based Matching

New events rendezvous with matching subscriptions in the nodes which are responsible for the hash keys of values in its own text nodes of XML document. Event brokers collaborate to store subscriptions and to process new events. When a new subscription arrives at a broker, it forwards the subscription to other brokers which are responsible for this subscription.

Our subscription forwarding strategy depends on the fact that a subscription may have several hash keys from the predicates of its XPath expression. Hence, a *composite subscription* which has several values in its predicates should be forwarded through many ways. Figure 4.1(a) shows that a composite subscription identified by sID with two identifiers of t2 and t3, denoted by 〈sID, (t2, t3)〉, is forwarded to two nodes respectively.

After a subscription is stored in rendezvous nodes, a broker receives an event from a client and parses it to extract the values of its text nodes from XML document. Then, it also forwards the message to the rendezvous nodes. The rendezvous broker tests all the local subscriptions to find the S' i.e., a set of subscriptions covering received event. Figure 4.1(b) shows that an event identified by eID, with three idenfifiers of t1, t2, and t3, is forwarded to three nodes respectively. In this case, the two nodes in the node of 5 and 34 find the subscription successfully, and the matching in the node of 27 fails.

Once the matching subscriptions are found, the broker should add the event to the queues of the subscriptions at the brokers of subscribers. The events in the subscription queue are consumed by the event client when the owner of the subscription is connected to the system.

4.3 Lookup Protocol

Although the original Chord paper[2] shows that the average path length is about $(1/2)\log_2 N$, it scales lenearly along with the number of nodes. Hence, we enhanced the performance of lookup operation for the scalability of the MicroPost.

Like Chord, Micropost also assigns keys to nodes with *consistent hashing*. The hash function assigns an m-bit identifier to each event broker using *SHA-1* as a base hash function. Hence, the size of identifier

space M is $2^m=2^{160}$, where m=160.

To accelerate lookups, we redesign the routing table of traditional Chord system. Instead of *finger*

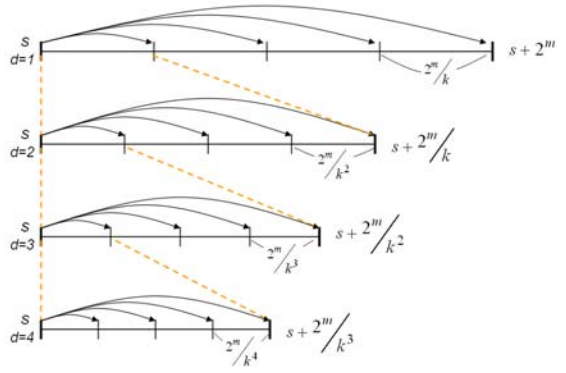


Figure 4.2 An example raking from s with k=4, d=4

Table 4.1 Definitions of Rake Table

Notation	Definition
rake teeth, k	the number of rake teeth to divide the key space into k subspaces and to conquer the lookup of given key
rake depth, d	the depth of raking, $1 \le d \le \lfloor \log_k N \rfloor$
rake[d].tooth[i]	the first node on the ring of key space that succeeds $s+i \cdot \left(\frac{N}{k^d}\right), 1 \le i \le k-1$, where s is the hash key of raking node and d is the depth of raking

table, we defined *rake tables* as shown in Table 4.1. In our design space, a node has better notion of closer nodes on the ring of identifier space. Figure 4.2 shows an example of raking from node s, with given number of k=4 and d=4.

The selection of *k,* the number of *rake teeth,* and *d,* the number of *rake tables* is the most important choice for enhancing the performance of lookup. In the case of *k=2,* and *d=m,* the identifier space is divided into a half size of subspace to the depth of m raking tables. Although this is exactly equivalent with the traditional Chord's finger table, it does not looks like being so efficient. Hence, we will investigate the golden section of k and d in the following section.

First of all, let us consider the impact of rake teeth k. If we choose an arbitrary value of k, a raking table of one-depth divides the entire key space into k subspaces sized by $2^m/k$. Recursively, we can proceed to the maximum depth of raking, until the size of subspace becomes lesser than k. From this recognition, we induce the following theorem:

Theorem 1. With a given number of rake teeth k, the maximum depth of raking D is the floor of $\log_k N$ in N-node network i.e., $D = \lfloor \log_k N \rfloor$.

Proof. Let k be the number of rake teeth, and N be the number of nodes. The first rake table of one-depth divides the entire identifier space into M/k subspaces, where M is the number of total identifier space. With high probability, it also divides the number of nodes to be covered as the same proportion, i.e., N/k. Recursively, the raking table of d-depth divides the uncovered nodes into N/k^d. As the maximum depth of raking D does not goes bigger than an integer which fulfills the following condition: $N/k^D < k$, i.e., $N < k^{D+1} \Rightarrow D < \log_k D - 1$. Hence, D is a ceiling of $\log_k N - 1$, and finally, we get the equation: $D = \lfloor \log_k N \rfloor$. The probability that the raking with k divides the nodes into N/k uniformly, is discussed and proved in [6]. ■

On this finding, we have the deterministic size of routing table as the following lemma:

Lemma 1. With a given number of rake teeth k in N-node network, the number of states in rake tables T is $(k-1)\lfloor \log_k N \rfloor$.

Proof. Again, let k be the number of rake teeth. Then, the number of raking tables is resolved by the equation in Theorem 1, and is the same as the number of maximum depth of raking D. Since the range of subspace divided by raking node itself is covered by the nested rake tables, we need to maintain the notion of $k-1$ neighbors. Hence, the total number of state in all the rake tables is $T = (k-1)D = (k-1)\lfloor \log_k N \rfloor$. ■

Now that the size of routing table is deterministic, we find that the performance of lookups is also deterministic in our design space, with high probability. We can state the asymptotic cost of lookups as the following:

Theorem 2. With a given number of rake teeth k, the average path lengths to find a node responsible

for a key is $O(\log_k N)$ with the notion of $O(k\log_k N)$ neighbors in $N-$node network.

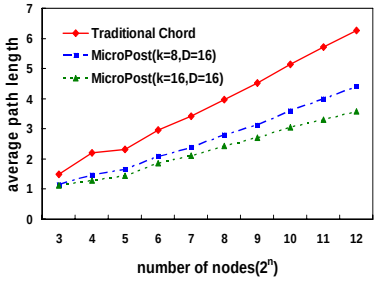


Figure 5.1

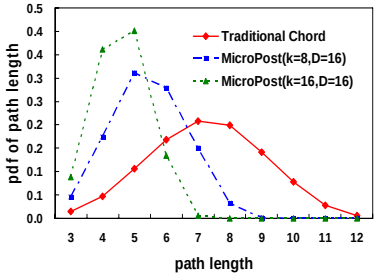


Figure 5.2

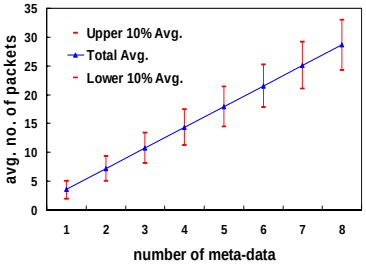


Figure 5.3

Table 5.1 Rake tables and average path lengths(N=2^{32})

k	D	T	APL(N=2^{12})	k	D	T	APL(N=2^{12})
2	32	32	6.26702	32	6	186	3.24751
4	16	48	5.02908	64	5	315	2.79897
8	10	70	4.39697	128	4	508	2.50793
16	**8**	**120**	**3.58154**	256	4	1020	2.45554

Proof. Suppose that a node maintains the states of $O(k\log_k N)$ neighbors in its rake tables. Let k is the number of rake teeth. At each hop of routing, the number of nodes is divided into N/k according to the same reasoning in Theorem 1. As the depth of raking d increases, the number of nodes to be covered diminishes to N/k^d. When the number of nodes to be covered is lesser than k, it is guaranteed to find the successor of given key by the definition shown in Table 1. Hence, the average path lengths converges into the number of raking tables which is deterministic by Theorem 1. i.e, $O(D)=O(\log_k N)$. ■

Since the cost of lookup operation is verified to be $O(\log_k N)$ in $N-$node network, it is trivial to prove that the number of packets required to forward events for rendezvous-based matching is $O(\omega\log_k N)$, where ω is the number of meta-data obtained from event messages. i.e., $\omega\times O(\log_k N)=O(\omega\log_k N)$.

5. Evaluation

In this section, we evaluate the efficiency of lookup operation by simulation.

We first consider the relationship between the average path lengths and the size of routing table. Table 5.1 shows the variation of average path lengths according to the size of raking teeth and raking depth. We can see that the average path length(APL) is shorter than the $(1/2)\log_2 N$, even though the size of routing table is smaller than m=160. Hence, we can argue that the cost of routing in our design has better performance with lesser states about neighbor nodes. It halves the average path lengths of Chord only with 3/4 neighbors, where k=16 and D=8 in the Table 5.1.

To understand the impact on the performance, we conducted an experiment which shows the average path lengths while the number of nodes increases. Figure 5.1 shows the result of our experiment. We changed the number of nodes from 2^3 to 2^{12}, and estimated the average path lengths. As expected, our design shows significantly better performance over traditional Chord with lessor neighbors.

In the next, we estimated the probability of average path lengths. Figure 5.2 shows the probability density function(PDF) of lookups in a network of N=2^{12}. It also shows that the enhancement of lookup performance is not skewed but uniform. Hence, we can argue that the total routing cost in whole network diminishes exponentially.

Finally, to see the variation of routing cost with the number of meta-data, we conducted an experiment with composite subscriptions. In a network of N=2^{12}, we varied the number of indexes in events from 1 to 8. Figure 5.3 shows that the average number of

packets in upper 10% path lengths, in lower 10% path lengths, and in total path lengths. The number of packets in total APL converges into $O(\omega \log_k N)$, as expected in our asymptotic analysis in previous section.

6. Conclusion

In this paper, we proposed a novel event notification service architecture for social applications based on asynchronous content-based publish/subscribe model. With the standards of XML Schema and XPath for defining event and subscription model, we exploit the expressiveness of CBPS as means of human-to-human communication.

In our design space, we adopt DHT as a substrate of decentralized and scalable broker overlay network for forwarding the social messages over the wide-area network with rendezvous-based matching.

In particular, we enhanced the performance of routing algorithm on top of Chord. The algorithm proposed in this paper halves the average path lengths of traditional Chord only with 3/4 entries of routing table. In the result, the cost of routing for forwarding and matching is $O(\omega \log_k N)$ in N-node network.

In the future work, we have a plan for solving the skewness of tags which has the property of Zipfian distribution in real world application of MicroPost.

References

[1] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: Understanding microblogging usage and communities," Proc. of the 9th WebKDD and 1st SNA-KDD 2007, San Jose, CA, USA, pp. 56-65, 2007.

[2] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," Foundations of Intrusion Tolerant Systems (OASIS'03), 2003.

[3] P. Pietzuch and J. Bacon, "Hermes: A Distributed Event-Based Middleware Architecture," Proc. of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW '02), 2002.

[4] G. Mühl, A. Ulbrich, K. Herrmann, and T. Weis, "Disseminating Information to Mobile Clients Using Publish-Subscribe," IEEE Internet Computing, vol. 8, no. 3, pp. 46-53, May 2004.

[5] A. Rowston, A. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale notification infrastructure," Proc. of the 3rd International Workshop on Networked Group Communication(NGC2001), 2001.

[6] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Transactions on Networking, vol. 11, pp. 17-32, Feb. 2003.

[7] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pp. 329-350, Nov. 2001.

[8] W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. Buchmann, "A peer-to-peer approach to content-based publish/subscribe," Proc. of the 2nd international workshop on distributed event-based systems, San Diego, CA, USA, pp. 1-8, 2003.

[9] D. Shi, J. Yin, Z. Wu, and J. Dong, "A Peer-to-Peer Approach to Large-Scale Content-Based Publish- Subscribe," Proc. of the 2006 IEEE/ACM international conference on Web Intelligence and Intelligent Agent Technology, Washington, DC, USA, pp. 172-175, 2006.

[10] F. Fabret, F. Llirbat, J. Pereira, and D. Shasha, "Efficient matching for content-based publish/subscribe systems," Technical Report, INRIA, 2000.

[11] I. Aekaterinidis and P. Triantafillou, "Internet scale string attribute publish/subscibe data networks," Proc. of the 14th ACM international conference on Information and knowledge management, Bremen, Germany, pp. 44-51, 2005.

[12] I. Aekaterinidis and P. Triantafillou, "PastryStrings: A Comprehensive Content-Based Publish/Subscribe DHT Network," Proc. of the 26th IEEE International Conference on Distributed Computing Systems(ICDCS'06), pp. 23-23, 2006.

[13] W3C. "XML Schema Part 0: Primer," W3C Recommen- dation, World Wide Web Consortium, May 2001.

[14] W3C. "XML Path Language(XPath) Version 1.0," W3C Recommendation, World Wide Web Consortium, Nov. 1999.