PAPER

Design of Content-Based Publish/Subscribe Systems over Structured Overlay Networks

Shou-Chih Lo †, *Regular member* and Yi-Ting Chiu †

Summary
The management of subscriptions and events is an important task in the content-based publish/subscribe system. A good management mechanism can not only produce lower matching costs to speed up the delivery of matched events to the interested subscribers but can also induce good load balancing for subscription storage. In this paper, we consider the construction of this kind of system over a peer-to-peer overlay network and propose two message-to-node mapping schemes for system management. We both analyze and simulate the performance of the proposed schemes. The simulation results show the superiority of our schemes over existing ones.

Key words:
Publish/Subscribe, Peer-to-Peer, Overlay Networks, Load Balancing

1. Introduction

The publish/subscribe (p/s) paradigm is an efficient paradigm for interconnecting applications in a distributed environment [1]. Many distributed applications such as stock trading, online auction, and news distribution benefit from the p/s paradigm. P/S systems contain information providers (also called publishers) who publish events to the system and information consumers (also called subscribers) who subscribe to particular categories of events within the system by issuing subscriptions. The system ensures the timely delivery of published events to all interested subscribers. A p/s system is implemented on a network of brokers that are responsible mainly for routing events between publishers and subscribers.

There are two general categories of p/s systems [2]: subject-based or content-based. In subject-based systems, each event belongs to one of a fixed set of subjects (also known as groups, channels, or topics). Publishers label each event with a subject; subscribers subscribe to all the events within a particular subject. The subscribers will then eventually receive all the events that are associated with that subject. However, a subscriber would get much more information than needed.

Content-based systems, on the other hand, are not constrained to the notion that an event must belong to a particular subject. Each content-based p/s system supports a predefined schema with n attributes $\{A_1, A_2, ..., A_n\}$ over which publishers could set values to publish events and subscribers could set constraints to get the ideal events. For example, a subscriber could issue a subscription [Stock = IBM, Price < 95, Volume > 1000], and a publisher could issue an event [Stock = IBM, Price = 90, Volume = 1200]. An event is said to match a subscription if all the attribute values satisfy the constrain conditions. The matching of events to the set of all subscriptions stored in the brokers is done based on the contents (values of attributes) [3]. Nowadays many modern applications require the content-based p/s with high expressiveness in subscribers' aspects.

In traditional p/s systems [4-8], the brokers are organized into a tree-shaped overlay network. These systems are simple but suffer from the fault tolerance problem. In order to have a scalable, fault-tolerant, and self-organizing capability, some existing content-based p/s systems [9-12] are built on top of a peer-to-peer (P2P) architecture. The conceptual way of these systems managing the subscriptions and events is based on a message-to-node mapping provided by each P2P protocol. The message-to-node mapping can be considered as that the set of subscriptions and events (domain set) is mapped to the set of brokers (range set). This mapping has to satisfy the intersection rule [10] that an event and its matched subscription have to be mapped to the same broker. Those brokers that are mapped to by at least one subscription or event are called rendezvous brokers (RBs).

There are three main operations in these P2P-overlayed content-based p/s systems: (1) storing the subscriptions into RBs, (2) delivering the events to RBs, and (3) matching the arriving events against to all the subscriptions stored in RBs. Operations (1) and (2) yield message routing costs to the system, while operation (3) yields data retrieval and matching operation costs to the system.

An ideal system had better provide lower costs mentioned above. Having a system providing lower match costs is particularly important, since faster event delivering to the interested subscribers ensures the property of timely delivery of events. Obviously, the match cost on an RB increases with the quantities of subscriptions it stores. Given a definite number of subscriptions issued into the system, the more the RBs for subscriptions are, the lower the match cost on an RB is. A

Manuscript received
Manuscript revised
† The authors are with Dept. of CSIE, Dong Hwa University, Taiwan

good message-to-node mapping method should generate as many RBs as possible for load balancing and keep low message routing costs as well.

In this work, we consider the design of message-to-node mapping methods. We discuss two mapping methods and provide their performance evaluations in terms of the load balance for subscription storage and the costs of subscription storing, event delivery, and event-subscription matching.

The remainder of this paper is organized as follows. Section 2 introduces some background knowledge and presents the related work. Section 3 describes our proposed mechanisms. Section 4 shows the performance evaluation. In section 5, we draw our conclusions.

2. Background and Related Work

We first introduce some knowledge about P2P overlay networks. The P2P overlay networks have attracted much attention due to desired characteristics for a large-scale distributed environment. Some system protocols like Chord [13], CAN [14], and Pastry [15] have been proposed to build large file sharing applications. These protocols are based on the DHT (distributed hash table) mechanism which allows shared files to be uniformly distributed into peers in the P2P network.

Take the Chord protocol as an example. All peers will be uniformly hashed into an identifier circle space and then are connected into a ring structure. Each peer is associated with the identifier number it is hashed into and may maintain a routing table (or called *finger table*) to other neighboring peers. A shared file will be stored into one peer by referring to the hash value of the selected key (e.g., file name). That is, a shared file with hash value k will be stored into the *successor peer* of identifier k which is the first peer encountered with its identifier larger than k.

The operations of a p/s system over this Chord structure may look like the example in Fig. 1. The brokers are constructed into a ring structure. Each subscription is stored into one broker upon one selected key. For example, subscriptions s_1 and s_2 are delivered to their successor brokers according to the hash values of "Price" and "Stock", respectively. An event is delivered to several brokers upon more than one selected key. For example, event e is delivered to three brokers according to the hash values of "Stock", "Price", and "Volume". In this example, e matches with s_1. These message deliveries are routed based on the finger tables and their delivery costs are measured in logical hops (steps from one broker to the other brokers in the overlay network).

Below, we introduce four p/s systems that are built over the P2P network. Ferry [9] and the proposed systems by Triantafillou et al. [10] and Baldoni et al. [11] are built over the Chord framework, while Meghdoot [12] is built over the CAN framework.

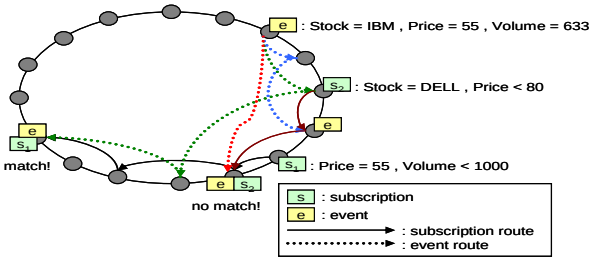


Fig. 1 Main operations of p/s systems over a P2P network.

Based on the mapping method used in the P2P network, we can classify these p/s systems into two broad categories: *attribute-name-based* and *attribute-value-based*. The attribute-name-based schemes store and deliver the subscriptions and events upon the selected key from the attribute names involved (For example, we select Price for s_1). While the attribute-value-based ones perform these jobs upon the selected key from the attribute values involved (For example, we select 55 for s_1).

Ferry [9] belongs to the attribute-name-based category. The RBs are those brokers that are mapped to by the hash values of all attribute names in the schema. A subscription which is issued by a subscriber with a user identifier SID is stored into the RB with its associated identifier being the largest one of smaller than SID (also called predecessor broker). Basically, the subscription storing in Ferry is irrelevant to what the subscription is but relevant to by which subscriber the subscription is sent. An event will be sent to all RBs to guarantee the intersection rule. The disadvantage of this scheme is that the number of RBs is limited (bounded by the number of attributes defined in the schema) and the event delivery cost is high.

The remaining three systems belong to the attribute-value–based category. Assume that each attribute in the schema has a value domain. For a numerical attribute, the value domain contains the set of values by enumerating all numbers between the lower bound and the upper bound in steps of a given precision. Then, the RBs are those brokers that are mapped to by the hash values of all value domains in the schema. Naturally, the number of RBs is greater than that in the attribute-name-based scheme.

We can consider that a subscription is the specifications of range values for some attributes. In [10], a subscription is stored upon the hash values of distinct values within all ranges involved in the subscription. In [11], a subscription is stored according to the distinct values in one value range associated to one selected attribute in the subscription. These two systems will send a subscription to more than one RB, which is called *subscription replication phenomenon* in this paper. The replication phenomenon becomes more serious when most

attributes specified in the subscription contain huge value ranges. An event will be delivered to those RBs that are mapped to by all attribute values involved in the event. The disadvantage of above two systems is that subscription storing costs are high.

In Meghdoot [12], a subscription is mapped to only one RB by translating the range specifications into one point in $2n$ dimensional space (n is the number of attributes in the schema). An event will be delivered to a half of RBs in average. Moreover, only a half of brokers function as RBs. The maintenance cost on this high dimensional space is considerably high.

3. Message-To-Node Mapping

In this section, we propose two message-to-node mapping methods: hybrid mapping (Hybrid), and interval index mapping (IIM). These two methods are built over the Chord framework where a hash function $h()$ is used. We would use the following example to illustrate the basic operations.

Subscription s: [Stock = IBM, 42 < Price < 49]

Event e: [Stock = IBM, Price = 47]

A subscription basically contains predicates which are conjunctively combined and each predicate has the form of A_i θ *value* (θ is any relational operator). An event also contains predicates conjunctively combined but with the form of A_i = *value*. An attribute in the schema is called an equal-type one if the associated predicate with the attribute in any subscription contains only the equal operator. For example, "Stock" is an equal-type attribute in s. An attribute is called a range-type one, if it is not an equal-type one. For example, "Price" is a range-type attribute in s. Assume that we have labeled each attribute with its type in the schema in advance.

3.1 Motivations

Attribute-name-based schemes do not cause the subscription replication phenomenon so as to have low subscription storing costs. However, the design philosophy of Ferry incurs high event delivery costs. We can refine this scheme as follows: A subscription is stored into one RB that is mapped to by the hash value of one attribute name randomly selected in the subscription, and an event is then delivered to all RBs that are mapped to by the hash values of all attribute names in the event. This refined scheme is referred to as the pure attribute-named (PAN) mapping. PAN is simple but suffers from the scalability problem as Ferry due to the set of RBs being bounded by the schema. A large volume of subscriptions would overwhelm the relatively small set of RBs such that the matching cost spent on an RB is increased. Moreover, PAN would suffer from the load unbalancing problem when attributes in the schema are unevenly involved in subscriptions. An RB that is mapped to by hot attribute names will store more subscriptions than other RBs.

Attribute-value-based schemes though have the subscription replication phenomenon but can provide more RBs. The domain set of this message-to-node mapping contains all the distinct values in value domains of all attributes in the schema. Under the same hash function $h()$, a large domain set can generate more RBs. However, a serous subscription replication phenomenon would amortize the advantage of having a large set of RBs.

Therefore, we propose the Hybrid scheme which combines the advantages of attribute-name-based and attribute-value-based schemes. Hybrid is based on the switching between attribute names and attribute values. A subscription is stored upon one randomly selected attribute as well. However, the attribute name is referred to if this attribute is of range type; while the attribute value is referred to if this attribute is of equal type. For example, s will be stored into the successor broker of h("IBM") if Stock is selected, and the successor broker of h("Price") if Price is selected. An event is then delivered to the RBs by considering all attributes in the event using the same switching rule. For example, e is delivered to the successor brokers of h("IBM") and h("Price"). The domain set of Hybrid is larger than that of PAN due to the including of some attribute values. In general, Hybrid behaves well if most attributes are equal-type ones. However, the load unbalancing problem can not be avoided if most hot attributes are of range type.

Next, we propose the IIM scheme by following the attribute-value-based scheme but reducing the subscription replication overhead. Instead of enumerating all values in steps of a given precision for a range-type attribute, we enumerate all values in steps of a given interval (which is larger than the precision). We uniformly divide the value domain of a range-type attribute into several intervals. For example, if the value domain of Price is [0, 500], we can generate 100 intervals like [0, 5), [5, 10), [10, 15), and so on. The left boundary numbers 0, 5, and 10 in this sequence of intervals are called *interval indices*. The range [42, 49] of Price in s falls into intervals with interval indices 40 and 45. Then, s is stored into the successor brokers of $h(40)$ and $h(45)$ if Price is selected. e is then delivered to the successor brokers of h("IBM") and $h(45)$ (47 falls into the interval with interval index 45). A subscription will be stored into more brokers when using a smaller interval size. IIM would perform poorly if a too large or small interval size is chosen. We found that IIM outperforms Hybrid if most attributes are of range type.

3.2 Subscription Storing and Event Delivery

In the following, we give the formal descriptions about the operations of subscription storing and event delivery

for the proposed two mapping schemes. For each range-type attribute A_i in the schema, we have the notations:

- $V_{max}(A_i)$: the maximal value in the value domain of A_i.
- $V_{min}(A_i)$: the minimal value in the value domain of A_i.
- $Interval(A_i)$: the set of interval indices induced from the intervals divided from the value domain of A_i. The number of interval indices is denoted by $|Interval(A_i)|$.
- $Index(v, A_i)$: the interval index of value v in the value domain of A_i.

For an attribute A_i in a subscription or event, we have the notations:

- $V_{high}(A_i)$: the right boundary value of the range of A_i specified in the subscription.
- $V_{low}(A_i)$: the left boundary value of the range of A_i specified in the subscription.
- $V_{equal}(A_i)$: the attribute value of A_i specified in the subscription or event by an equal operator.
- $Name(A_i)$: the attribute name of A_i.

Hybrid:

Subscription storing

(1) Randomly select an attribute A_i in the subscription s.
(2) If A_i is an equal-type attribute,
(3) Store s in the successor broker of $h(V_{equal}(A_i))$.
(4) If A_i is a rang-type attribute,
(5) Store s in the successor broker of $h(Name(A_i))$.

Event delivery

(1) For each attribute A_i in the event e,
(2) If A_i is an equal-type attribute,
(3) Deliver e to the successor broker of $h(V_{equal}(A_i))$.
(4) If A_i is a rang-type attribute,
(5) Deliver e to the successor broker of $h(Name(A_i))$.

IIM:

Subscription storing

(1) Randomly select an attribute A_i in the subscription s.
(2) If A_i is an equal-type attribute,
(3) Store s in the successor broker of $h(V_{equal}(A_i))$.
(4) If A_i is a rang-type attribute,
(5) $B = Index(V_{low}(A_i), A_i)$,
(6) While $B \le Index(V_{high}(A_i), A_i)$,
(7) Store s in the successor broker of $h(B)$;
(8) $B = B + \lceil (V_{\max}(A_i) - V_{\min}(A_i)) / |Interval(A_i)| \rceil$

Event delivery

(1) For each attribute A_i in the event e,
(2) If A_i is an equal-type attribute,
(3) Deliver e to the successor broker of $h(V_{equal}(A_i))$.
(4) If A_i is a rang-type attribute,
(5) Deliver e to the successor broker of $h(Index(V_{equal}(A_i), A_i))$.

3.3 Cost Analysis

We estimate the performance of the message-to-node mapping schemes by the metrics of subscription storing, event delivery, and event-subscription matching costs.

- Total subscription storing cost (C_{sub}): the amount of logical hop counts needed to deliver all subscriptions to the corresponding RBs in the P2P network.
- Total event delivery cost (C_{event}): the amount of logical hop counts needed to deliver all events to the corresponding RBs in the P2P network.
- Total event-subscription matching cost (C_{match}): the amount of attribute numbers of subscriptions that are examined by all events (We assume that a subscription with k attributes will be examined k times for any event).

The following terms are used in the cost analysis:

- N_s: the total number of subscriptions issued in the system.
- N_e: the total number of events issued in the system.
- N_b: the total number of brokers in the system.
- ATT_s: the average number of attributes specified in a subscription.
- ATT_e: the average number of attributes specified in an event with the hash values of their attribute names/values being distinct.
- D: the domain set of the message-to-node mapping (The cardinality of D is denoted by $|D|$).

Let $f(|D|)$ be a function that returns a value between 0 and 1 and the value is directly proportional to $|D|$. We use $N_b \times f(|D|)$ to represent the number of RBs. Suppose there is a schema $S = \{A_1, A_2, ..., A_x, A_{x+1}, ..., A_n\}$ with the first x attributes are all equal-type ones and the other attributes are all range-type ones. Let $Domain(A_i)$ be the value domain of A_i. We assume that the n attributes in the schema have the equal probability to be specified in a subscription or event.

In the following, we give the cost model of each proposed scheme. Moreover, Ferry, PAN, and the proposed scheme by Baldoni et al. (called PAV, Pure Attribute-Value) are discussed as well.

Hybrid:

D is composed of the value domains of all equal-type attributes and the attribute names of all range-type attributes. In Chord with N_b brokers, the average logical hop counts to locate a successor broker is $\frac{1}{2}\log(N_b)$. The average number of subscriptions stored in an RB is $\frac{N_s}{N_b \cdot f(|D|)}$. Hence we have,

$$D = \bigcup_{i=1\sim x} Domain(A_i) \cup \bigcup_{j=x+1\sim n} Name(A_j)$$

$$C_{sub} = N_s \cdot \tfrac{1}{2}\log(N_b)$$

$$C_{event} = N_e \cdot ATT_e \cdot \tfrac{1}{2}\log(N_b)$$

$$C_{match} = N_e \cdot ATT_e \cdot \frac{N_s}{N_b \cdot f(|D|)} \cdot ATT_s$$

IIM:

D is composed of the value domains of all equal-type attributes and the interval indices of all range-type attributes. If an equal-type attribute is selected (with probability *x*/*n*), the subscription is stored into one RB. However, if a range-type attribute is selected (with probability (*n*-*x*)/*n*), the subscription is stored into *I* RBs, where *I* is the average number of interval indices of an attribute with their hash values being distinct. Hence, we have,

$$D = \bigcup_{i=1\sim x} Domain(A_i) \cup \bigcup_{j=x+1\sim n} Interval(A_j)$$

$$C_{sub} = N_s \cdot (\frac{x}{n} + \frac{n-x}{n} \cdot I) \cdot \frac{1}{2}\log(N_b)$$

$$C_{event} = N_e \cdot ATT_e \cdot \frac{1}{2}\log(N_b)$$

$$C_{match} = N_e \cdot ATT_e \cdot \frac{N_s \cdot (\frac{x}{n} + \frac{n-x}{n} \cdot I)}{N_b \cdot f(|D|)} \cdot ATT_s$$

Ferry:

D is composed of the attribute names in *S*. Any event will be delivered to all RBs. Hence we have,

$$D = \bigcup_{i=1\sim n} Name(A_i)$$

$$C_{sub} = N_s \cdot \frac{1}{2}\log(N_b)$$

$$C_{event} = N_e \cdot N_b \cdot f(|D|) \cdot \frac{1}{2}\log(N_b)$$

$$C_{match} = N_e \cdot N_s \cdot ATT_s$$

PAN:

D is composed of the attribute names in *S*. Hence we have,

$$D = \bigcup_{i=1\sim n} Name(A_i)$$

$$C_{sub} = N_s \cdot \frac{1}{2}\log(N_b)$$

$$C_{event} = N_e \cdot ATT_e \cdot \frac{1}{2}\log(N_b)$$

$$C_{match} = N_e \cdot ATT_e \cdot \frac{N_s}{N_b \cdot f(|D|)} \cdot ATT_s$$

PAV:

D is composed of the value domains of all attributes in *S*. If an equal-type attribute is selected, the subscription is stored into one RB. However, if a range-type attribute is selected, the subscription is stored into *R* RBs, where *R* is the average number of distinct values within the value range of the attribute. Hence, we have,

$$D = \bigcup_{i=1\sim n} Domain(A_i)$$

$$C_{sub} = N_s \cdot (\frac{x}{n} + \frac{n-x}{n} \cdot R) \cdot \frac{1}{2}\log(N_b)$$

$$C_{event} = N_e \cdot ATT_e \cdot \frac{1}{2}\log(N_b)$$

$$C_{match} = N_e \cdot ATT_e \cdot \frac{N_s \cdot (\frac{x}{n} + \frac{n-x}{n} \cdot R)}{N_b \cdot f(|D|)} \cdot ATT_s$$

Performance comparison:

The C_{sub} values of PAN, Hybrid, and Ferry are all the same and equal to the delivery cost to one RB. The C_{sub} values of IIM and PAV are higher than those of the other three schemes due to the subscription replication phenomenon. The C_{sub} value in IIM is smaller than that in PAV due to $I < R$. The C_{event} values of PAN, Hybrid, IIM, and PAV are all the same, since an event has to be delivered to ATT_e nodes. However, the C_{event} value in Ferry is very high. The value of C_{match} is dominated by the average number of subscriptions that are stored in an RB. The number of RBs is relevant to the size of *D* and the relationship of these $|D|$'s values is: PAN = Ferry < Hybrid < IIM < PAV. Therefore, the C_{match} values of PAN and Ferry are larger than those of the other schemes.

4. Simulations

We evaluate the performance of the proposed schemes using simulation programs. Below, we introduce our environmental settings and experimental results.

4.1 Environmental Settings

We consider a system with 4096 static Chord nodes and generate totally 10000 subscriptions first and 30000 events next into the system. The number of attributes defined in the schema is 20 by default and can be varied between 10 and 30. The *x* percent of attributes in the schema belong to equal-type ones and *x* is varied from 0 to 100. Each equal-type attribute has a value domain of size 100 and these 100 distinct integer values are randomly selected from the range 0~5000. The i^{th} ($i \geq 1$) range-type attribute in the schema has the value domain of [0, MIN_RN+GP(*i*-1)], where MIN_RN and GP have default values of 500 and 150, respectively. The precision of a range-type attribute is set to 1. The value domain of a range-type attribute is divided into *I* intervals (denoted by IIM: *I*) and *I* (interval number) is set to 12, 25, or 50.

The probability of an attribute in the schema to be specified in a subscription follows the Zipfian distribution which is commonly used to model the data access skewness. The frequency of occurrence of the n^{th} ranked item is defined to be $1/n^{\theta}$, where a larger value of θ denotes a more skewed distribution. θ is set to 1.5 by default. The selection of a value from the value domain of an equal-type attribute in a subscription also follows the Zipfian distribution. The range of a range-type attribute is specified in a subscription by randomly selecting two boundary values from the value domain. The selection of attributes and values in an event however follows the uniform distribution. That is, a half of attributes in the schema will be selected in an event.

We additionally use the following two metrics besides C_{sub}, C_{event}, and C_{match} to evaluate performance:

- Load balance: We calculate the standard deviation of the numbers of subscriptions stored in RBs. This metric indicates how well a mechanism deals with data access skewness.

- Total match latency: The sum of the match latencies of all events. The match latency of an event is defined as the maximal amount of attributes examined by an event among all the RBs the event is delivered to. The match latency is a factor to show the system response time in a parallel processing environment.

4.2 Experimental Results

Experiment A: Load Balance

We measure the standard deviation of the numbers of subscriptions stored in all RBs. A system with a small standard deviation is said to be load balancing. Two simulation results with 75% and 25% equal-type attributes under different values of θ are shown in Fig. 2. More subscriptions involve the same attribute names (hot attributes) and attribute values when θ increases. We found that Ferry performs independently with the value θ due to its unique subscription storing behavior of using the identifiers of subscribers. PAN suffers from the load unbalancing problem, since the deviation value increases as θ increases. Most subscriptions with hot attributes are stored in the same RB due to using the same attribute name to hash. PAV has a low deviation value when most attributes are of equal type (see Fig. 2a) but a high deviation value when most attributes are of rang type (see Fig. 2b). The diversity of attribute values of a hot attribute can solve the load unbalancing problem. In our simulation settings, the value domains are quite different for equal-type attributes but are mostly overlapped for range-type attributes. Therefore, involving more range-type attributes in the subscription has a higher probability to be hashed into the same RB with other similar subscriptions. Hybrid has a good load balancing feature due to mixing attribute names and attribute values. However, Hybrid behaves like PAN when all attributes are of range type and the good feature is broken. The deviation value of IIM decreases when fewer intervals are generated. The reason is that more interval indices of different range-type attributes with overlapped value domains become identical as more intervals are generated. Generally, we can conclude that attribute-value-based schemes outperform the attribute-name-based ones in load balancing.

Experiment B: Subscription Storing

Fig. 3 shows the total logical hops needed for subscription storing of different schemes under different percentages of equal-type attributes. PAV is not plotted here due to a large-scale cost over other schemes. For example, the cost of PAV is 29 times the cost of IIM:50. IIM suffers from a high storing cost due to subscription replication but the cost is largely reduced when using a less number of intervals, if most attributes are of range type (lower percentage). Hybrid and PAN perform equally and independently with the percentage. Ferry thought has the same storing cost as PAN in our cost analysis but experiences a little more cost than PAN in our simulation due to the storing on a predecessor broker. IIM, Hybrid, and PAV behave equally when all attributes are of equal type (100% percentage).

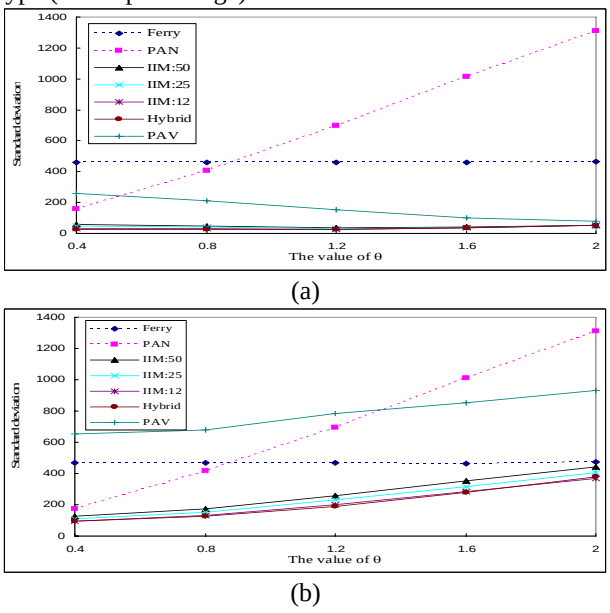


Fig. 2 Load balance: (a) 75% equal-type attributes and (b) 25% equal-type attributes.

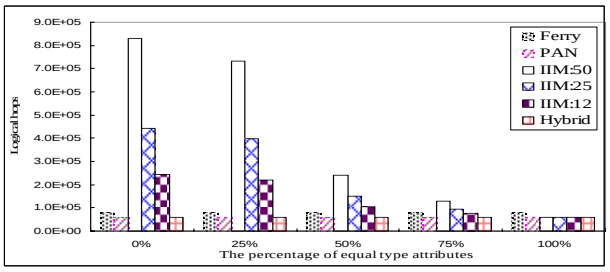


Fig. 3 Hops needed for subscription storing.

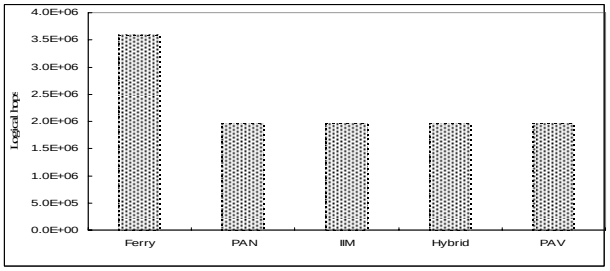


Fig. 4 Hops needed for event delivery.

Experiment C: Event Delivery

Fig. 4 shows the total logical hops needed for event delivery of different schemes. We found that Ferry takes much more cost than the other schemes due to sending events to all RBs. The other four schemes yield the same cost due to an event being delivered to the same number of RBs.

Experiment D: Event-Subscription Matching

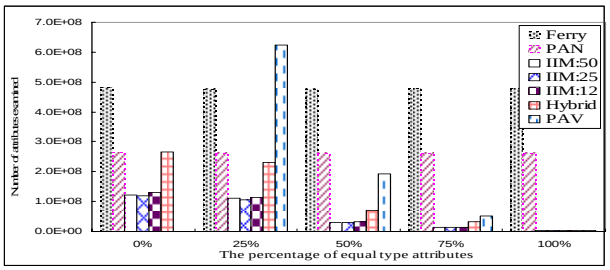
Fig. 5 shows the event-subscription matching cost in terms of match overhead and match latency under different percentages of equal-type attributes. The results of PAV when all attributes are of range type are not plotted here due to large-scale values.

For total match overhead shown in Fig. 5a, we found that attribute-name-based schemes (Ferry and PAN) perform independently with the percentage, while attribute-value-based schemes (IIM, Hybrid, and PAV) perform well with fewer range-type attributes (larger percentage). The overhead comparison is: Ferry > PAN > Hybrid > IIM. PAV suffers from more overhead with more range-type attributes than the other schemes but performs well with fewer range-type attributes.

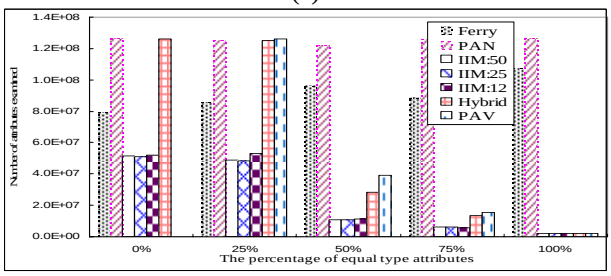
As we have explained in the cost analysis, the match overhead is relevant to the average number of subscriptions stored in an RB. This number decreases as more RBs are used but increases as more replicated subscriptions are generated. Hybrid benefits from more RBs generated when most attributes are of equal type. PAV generates many RBs but also generates many replicated subscriptions. The latter effect overwhelms the former one when most attributes are of range type. The two effects get balanced in IIM using interval domains. IIM performs best among all with an appropriate interval number (25 in the figure). A too large or small interval number may increase the overhead instead.

Fig. 5b shows the total match latencies of different schemes with various settings. The three attribute-value-based schemes outperform attribute-name-based ones when the percentage is greater than 50%. Hybrid and PAV become worse when the percentage is smaller than 50%. IIM has the best performance among all.

The match latency is an indicator of the system response time and is affected by the degree of load balancing. An RB with the most of subscriptions stored becomes a bottleneck. PAN using attribute names has the unbalancing problem, so the match latency is high. PAV when most attributes are of range type also suffers from the unbalancing problem as discussed in Experiment A. Hybrid behaves more similar to PAN when most attributes are of range type and hence has a high latency in this situation.



(a)



(b)

Fig. 5 Event-subscription matching: (a) match overhead and (b) match latency.

Experiment E: Schema Size

Fig. 6 shows the total match latencies of different schemes under different schema sizes. The percentage of equal-type attributes is set to 50%. As the number of attributes in the schema increases, the average number of attributes specified in a subscription increases but more RBs are generated as well. These factors have a conflicting effect on the match latency. Therefore, the match latencies of most schemes, particularly for attribute-value-based ones, are decreased and then increased as the schema size is getting large. The optimal value appears as the size is 20.

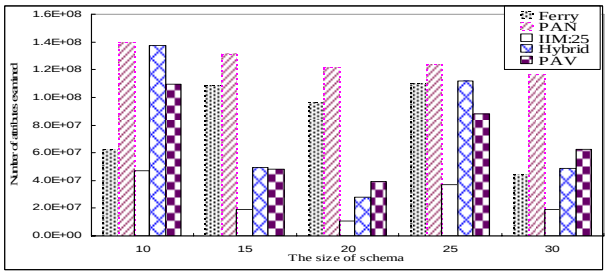


Fig. 6 Match latencies under various schema sizes.

Experiment F: Optimal Interval Number

With a smaller interval number, the subscription storing cost in IIM decreases as shown in Fig. 3 but the match overhead increases as shown in Fig. 5. We try to find the optimal interval number that minimizes the total cost (C_{sub}+C_{event}+C_{match}). This experiment is conducted by varying the value domain [0, MIN_RN+GP(i-1)] of the ith

range-type attribute. We vary the MIN_RN value from 100 to 500. First, we set GP = 0 (i.e., each range-type attribute has the same value domain) and let each range predicate in a subscription have an average size of D/4 (D is the size of its value domain). Then, we change the size of the range predicate to be 20 on average. Next, we let most range predicates have similar value ranges (particularly labeled as skew in the figure). We found that the optimal interval numbers for these settings are between 30 and 49 (see Fig. 7). At the last, we change GP = 50 (i.e., the value domain of a range-type attribute is getting large). The optimal interval numbers are between 13 and 24. That is, we prefer to use a large interval number if the value domains of range-type attributes are highly overlapped and a small interval number otherwise.

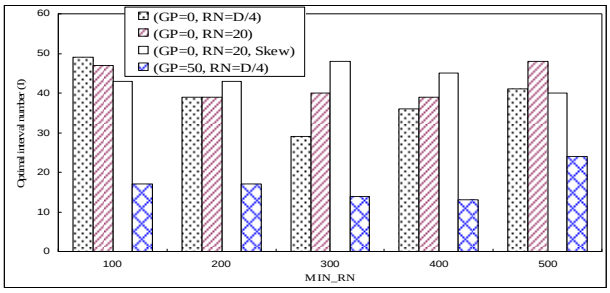


Fig. 7 The setting of optimal interval numbers.

5. Conclusions

The content-based p/s system is a very appealing interaction model and its capability can be fully enhanced over a P2P overlay network. The performance of this kind of system is dominated by the message-to-node mapping schemes. We found that the existing schemes either have high event delivery costs or high subscription storing costs. In this paper, we proposed two new mapping schemes: Hybrid and IIM. Hybrid performs well when the most of attributes are of equal type. IIM outperforms Hybrid if most attributes are of range type.

In the future, we will extend the work to both the location-dependent service environment and the mobile ad hoc network environment. The new challenges include the location tracking, filtering of location-dependent criteria, and handling of node mobility.

References

[1] C. Raiciu, D. S. Rosenblum, and M. Handley, "Revisiting Content-Based Publish/Subscribe," *IEEE International Conference on Distributed Computing Systems Workshops* (ICDCSW), pp. 19-24, 2006.

[2] Y. Liu and B. Plale, "Survey of Publish Subscribe Event Systems," *Technical Report TR-574*, Indiana University, Computer Science Dept., May 2003.

[3] Z. Shen and S. Tirthapura, "Faster Event Forwarding in a Content-Based Publish-Subscribe System through Lookup Reuse," *IEEE international Symposium on Network Computing and Applications*, 2006.

[4] G. Banavar, T. Chandra, B. Mukherjee, and J. Nagarajarao, "An Efficient Multicast Protocol for Content-Based Publish-Scribe Systems", *Proc. 19th IEEE International Conference on Distributed Computing Systems*, pp. 262–272, 1991.

[5] A. Carzagina and A. L. Wolf, "Forwarding in a Content-Based Network", *Proc. ACM SIGCOMM'03*, pp. 163-174, August 2003.

[6] A. Carzagina, M. J. Rutherford, and A. L. Wolf, "A Routing Scheme for Content-Based Networking", *Proc. IEEE INFOCOM'04*, vol. 2, pp. 918-928, March 2004.

[7] G. Cugalo, E. D. Nitto, and A. Fuggetta, "The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS", *IEEE Trans. on Software Engineering*, vol. 27, no. 9, pp. 827-850, September 2001.

[8] F. Y. Cao and J. P. Singh, "MEDYN: Match-Early and Dynamic Multicast for Content-based Publish/Subscribe Service Networks", *Proc. Fourth International Workshop on Distributed Event-Based Systems*, pp. 370–376, June 2005.

[9] Y. Zhu and Y. Hu, "Ferry: An Architecture for Content-Based Publish/Subscribe Services on P2P Networks," In Proceedings of the ICPP'05, 2005.

[10] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg, "Content-Based Publish-Subscribe over Structured Overlay Networks," Proc. 25th IEEE ICSCS, 2005.

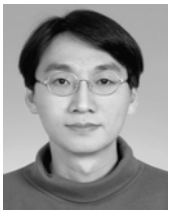
[11] P. Triantafillou and I. Aekaterinidis, "Content-based Publish-Subscribe over Structured P2P Networks," *International Conference on Distributed Event-Based Systems*, 2004.

[12] A. Gupta, O.D. Sahin, D. Agrawal, and A. E. Abbadi, "Meghdoot: Content-Based Publish/Subscribe over P2P Networks," *Proc. 5th ACM/IFIP/USENIX International Conference on Middleware*, pp. 254-273, October 2004.

[13] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications," *Proc. SIGCOM*M'01, 2001.

[14] S. Ratnasaour, P. Francis, M. Handley, and R. Karp, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM'01*, vol. 31, no. 4, pp.161-172, August 2001.

[15] A. Rowstron and P. Druschel, "Pastry: Scalable Distributed Object Location and Routing for Large-Scale peer-to-Peer Systems," *Proc. IFIP/ACM International Conf. on Distributed Systems Platforms* (Middleware), November 2001.



Shou-Chih Lo received the Ph.D. degree in computer science from National Tsing Hua University, Taiwan, in 2000. He joined the Computer & Communication Research Center at National Tsing Hua University in 2000 as a Postdoctoral Fellow. He has been with National Dong Hwa University, Taiwan, since 2004 and is now an assistant professor in the Department of Computer Science and Information Engineering. His current research interests are in the area of mobile and wireless networks.

Yi-Ting Chiu received the B.S. degree in applied mathematics from Chung Yuan Christian University, Taiwan, in 2005, and the M.S. degree in computer science and information engineering from National Dong Hwa University, Taiwan, in 2007.