NovaPS: A Robust Publish/Subscribe Overlay for Multimedia Communication

Yuanqiang Huang, Zhongzhi Luan, Depei Qian
Sino-German Joint Software Institute
Beihang University
Beijing, China
yuanqiang.huang@jsi.buaa.edu.cn

Jiali Du, Xiang Ni, Lan Gao
Sino-German Joint Software Institute
Beihang University
Beijing, China
lan.gao@jsi.buaa.edu.cn

Abstract—**The publish/subscribe communication paradigm is becoming more and more popular in a large number of applications. It provides flexibility in subscribing content of interest without prior bindings. We believe that this technique is valuable to multimedia communication which requires location independence and asynchronous communication. In this paper, we present NovaPS, a robust Publish/Subscribe overlay for reliable communication between the publisher and subscriber with high delivery ratio and fast notification/subscription matching, while keeping relatively low false positives rate, time/space overhead and communication cost. A detailed analysis of keys is given in this paper. Experiment results show that our NovaPS overlay can achieve the above features nicely in the dynamic network environment.**

Keywords-publish/sbuscribe;multimedia communication;robust

I. INTRODUCTION

Publish/Subscribe is a communication paradigm supporting multiple loosely coupled message exchange among multiple parties. It distinguishes two basic roles: publisher and subscriber. The former is the sender of messages while the latter is the receiver of messages. Subscribers usually don't have prior knowledge about publishers, but only declare their own interests, including topic of the notification, attributes and so on. When any notification published by a publisher can satisfy the subscribers' interests, it will be routed to them automatically. A key feature in distributed Publish/Subscribe is decoupling of publishers and subscribers both in both time and space. Both parties involved get in touch with each other in the interest-based information space without any previous bindings between them. It leads to the exchange of messages not subject to the existence of messages' sender or receiver. To complete the task, publisher is only required to send messages to a specific information space, and messages that satisfy interests will be sent to corresponding subscribers later. Due to the great prospect of asynchronous, anonymous and many-to- many (N-to-N) communication, Publish/Subscribe continues to be a hot research topic in distributed system.

Publish/Subscribe is very attractive to large-scale, highly dynamic environment and has been applied successfully to applications such as stock analysis[1], E-Commerce [2], RSS[3], distributed coordination[4], Electronic Auction[5], and online games[6], etc. We consider Publish/Subscribe can also fit multimedia communication which usually adopt traditional client/server communication model. The using of Publish/Subscribe paradigm, as depicted in Figure 1 is of the following merits: i) It helps separation of application logic from the underlying addressing mechanisms. Clients as subscribers don't need to know the location (IP address and port) of the multimedia server, and the multimedia server as the publisher doesn't need to know the status of requesting clients (including their addresses and the number of clients, etc) neither. The location independence between server and clients provides the flexibility to insert fault-tolerant mechanism such as a smoothly switching to backup server without disturbing clients when the current server fails. ii) Synchronization between server and the requesting clients is not necessary. The multimedia server can publish the data without knowing the existence of requesting clients. Supported by some of the buffering mechanism, the data can be buffered by the communication middleware. Specific optimization mechanism can be used to transfer the data to somewhere close to the potential clients. Once the clients connect to the system, they can get the required data immediately. The buffered data can be removed when it is kept too long or no enough buffer space. iii) It helps to mitigate the pressure to the server when having a lot of requests from clients. The server only needs to send data once, and the data will be delivered to all requesting clients through some specific multicast technology. iv) It is easier to evaluate and predict the performance of the multimedia service accurately due to decoupling of the multimedia server and the clients.

In this paper, we focus on the methods and mechanisms of robust overlay to guarantee Publish/Subscribe quality for multimedia communication. Our contributions to this topic include:

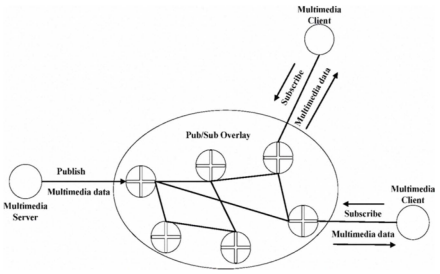


Figure 1. Multimedia communication using Pub/Sub paradigm

CHINACOM 2010, August 25-27, Beijing, China

DOI 10.4108/chinacom.2010.99

- We introduce a simple but efficient neighbor node maintenance method to construct and maintain an overlay with a topology of k-regular random graph with low cost and desirable properties.
- We propose an index-based content matching between notification and subscription, which reduces the cost of content matching in both time and space This mechanism is especially suited to multimedia communication.
- We introduce a rendezvous-based dynamic routing method in making tradeoff between communication overhead and delivery performance. It can significantly reduce the communication overhead with relatively small compromise of the delivery performance.

II. THE NOVAPS OVERLAY

A. Content space model

In the multimedia communication, different clients may subscribe to the same set of media frames, for example users may choose to watch the same online video, so every client as the subscriber will issue subscriptions for related video attributes like media_type, media_source, support_codec, and frame_resolution, etc. These attributes are usually assigned a unique value. Moreover, it is possible that subscription to attribute value in the form of a range instead of a single point. For example, in video image processing applications, different client may subscribe to different data blocks of a frame at the same time for compressing or repairing purpose. In addition, in applications with layered video codec feature[17][7], clients may choose to receive only some of the layers to achieve an acceptable video quality under their specific network condition, so they also may specify the IDs of the video streaming layer.

Due to these requirements in multimedia communication, we define the content space in Publish/Subscribe mode as $\{topic,(at_1,at_2,...at_n),payload\}$. Here *topic* indicates the topic-relate information that publisher and subscriber are interested in. It can also be considered as a unique identification of an interest group, at_i stands for attributes of the topic, which indicate detailed schema of the topic. The value ranges of these attributes constitute a comprehensive space of the interest in the corresponding topic. The *payload* is the actual load processed by receivers, such as frame data which is in binary bytes transparent to the communication middleware. In fact *topic* itself is a special unique attribute. Although different topics may have the same attributes definitions, subscribers with same attributes but different topics will not be considered as members of the same interest group. However subscribers in the same interest group could declare filtering constraints on these attributes to indicate their real interests. A filtering constraint is defined as: $FC=(fc_1,fc_2,...,fc_n)$. Following the definition of subscription in [9], filtering constraints FC is actually an expression of a set of predicates, that is $FC=p_1 \wedge p_2 \wedge \cdots \wedge p_n$. Each p_i is a predicate about the attribute at_i, consisting of attribute name (*an*) and constraint on this attribute (*ac*). The constraint ac consists of operator and the boundary value for attribute *av*, that is $p=\{an,\{operator,av\}\}$. For example, members in the interest group may issue filter constraints $\{10<X<20,Y=ABC\}$ to declare that they are interested in the notifications in which the value of attribute X is greater than 10 and less than 20, while the value of attribute Y is "ABC".

B. Constructing and maintaining the overlay

Our overlay relies on a membership management service to assign system local view to each node [8] so that every node can use its local view to setup neighbor relationship with other nodes.

In this paper we first cluster the nodes by interest topics. Nodes with the same topic form an overlay having the topology of K-regular random graph. Notifications with the same topic can be broadcast in this overlay. Since the possibility that a node changing its interest topic is small, the overlay topology is relatively stable and the K-regular random graph could guarantee that message can be reliably and accurately delivered to every node in the overlay. It should be noticed that even though broadcasting in the topic overlay network brings much bandwidth redundancy for transferring messages, it increases the success ratio of message delivery in case of faults. Through message bundle we can further mitigate the pressure on the network bandwidth caused by redundancy.

Overlay topology

Unlike other random graph, the degree of each vertex in K-regular random graph is fixed to K. When the degree of node k>= 3, K-regular random graph has some attractive properties: i) When randomly removing linear subset of its vertexes or edges, the remaining vertexes are still connected (with a very high probability to keep all connected) [10], this feature makes network with K-regular random graph topology fault tolerant, which can ensure message delivery to the remaining nodes even if many nodes or links are failed; ii) The diameter of the graph grows logarithmically with the total number of vertexes in the graph. This feature indicates the number of hops for routing a message to any other node will increase very slowly with the incense of total number of nodes; iii) Any two vertexes in the graph usually have K disjoint paths, which is called K-connected. [11] has proved that the probability that K-regular random graph is not K-connected is $O(N^{2-K})$, showing that the greater value of K is, the less possibility that the K-regular random graph is not K-connected. And when the system size is large enough (N is much larger than 1), the probability that K-regular random graph is not K-connected is very small (probability of non-k-regular is 1/N) even if K is set to 3. For overlay network based on K-regular random graph, each node needs only to interconnect with constant (k) number of neighbor nodes to guarantee that the properties of K-regular graph become effective. Here, we do not distinguish publisher or subscriber, because we assume every node can either be publisher or subscriber. The high symmetry of the K-regular random graph gives every node capability of being either a publisher or a subscriber. So the publish-subscribe overlay network based on K-regular random graph can easily implement a common n-to-n pub-sub paradigm.

The neighbor node maintenance

If a node is allowed to issue many subscriptions for different topic, it may be clustered into different interest groups, resulting in maintenance overhead since neighbor links increase linearly with the number of groups it joins. To reduce

the overhead on nodes, we utilize the fact that multiple nodes could have subscribed to the same topic to squeeze the node's degree. We consider that a node should choose the neighbor nodes that share more common topics with it. With this strategy various interest groups may share the most common node interconnections, and the maintenance overhead can be reduced naturally. But blindly choosing neighbors with the most common interest topics will also lead to a situation that a few nodes will form isolated clique, which destructs the connectivity of the whole group. Moreover randomly choosing a neighbor node can guarantee small probability of forming the cliques, thus the overlay has better connectivity. Based on the above consideration, we define our strategy for choosing and updating neighbor nodes as the follows: Each node periodically chooses a neighbor based on its local knowledge and checks whether its current degree (number of neighbor nodes) has exceeded the maximum degree. When the degree exceeds the limit, the node should reduce its neighbors until its degree is below the maximum limit, meanwhile the node will try to maintain the maximum of the average interest overlap with all its neighbors.

We assume that the maximum degree, that is, the total number of neighbors of a node is MAX_{neighb}, the number of neighbor nodes maintained in each interest group is K. Our neighbor node maintenance procedure is depicted in Figure 2:

Revoking the neighbor relationship with the neighbor who violates the MAX_{neighb} limit and shares the smallest number of common interest topics will help both sides to reduce degree. Redirecting a node means that node A revokes the neighbor relationship with some neighbor and then redirects that node to another neighbor with which it has the smallest number of common topics. This will has the least affect on the average overlap. If node A's degree is still below the maximum limit, it randomly chooses a node and determines the current common topics between them. If any common topic has not been K-coveraged, node A will setup a neighbor relationship with that node. The establishment and revocation of the neighbor relationship can refer to the methods and protocols in [13]. In addition, we noticed that each node needs to obtain information such as the current number of neighbor nodes and interest topics from its neighbors to complete the neighbor updating procedure, so neighbors need to exchange status with each other in real time. The practical method is to push the relevant information to its neighbor nodes when a node changes its status. Since the message may be lost in real environment, each node should send periodic heartbeat to its neighbor nodes to determine whether they have received the previous message and resend the message if it was lost.

```
On every node (A):
If A's degree is greater than MAX_neighb
    selecting a neighbor who also violates its own limit (MAX_neighb) and share the least common
    interest topics with A;
    If such neighbor exists
        A revokes the neighbor relationship with this node;
    If such neighbor doesn't exist
        select neighbor B who has the least neighbors;
        select another node C which share the least common interest topics with A;
        redirect C to B;
if A's degree is less than the limit MAX_neighb
    randomly choose node D who is not A's neighbor yet from the local view provided by
    membership service;
    calculate the common interest topics of A and D, denoted as t_comm;
    for each topic t in t_comm
        if A has less than K neighbors being interested in topic t
            A establish neighbor relationship with D;
            Exit the loop;
sleep for a fixed period;
```

Figure 2. Neighbor node maintenance procedure on every node

C. Content Matching

As we mentioned before, filtering constraints is the conjunctive expression of multiple predicates. Each predicate is a constraint logic for a specific attribute. The most commonly used predicate is to determine wether the value of an attribute is within the valid range or precisely equal to a specific number or string. In this paper we assume that subscription predicates are range-predicate or equal-predicate. To accelerate matching speed, we should first index every notification and subscription, and then confirm match through comparing indices of notification and subscription.

A method with combination of the hash-based dimension reduction and the Bloom Filter is proposed for indexing notification and subscription. Since each notification and subscription filter is a multi-dimensional description about attributes, we firstly turn this multi-dimensional description into a unique one-dimensional value, which we called dimension reduction. The dimension reduction for subscription is most complicated. Since filtering constraints consist of range-predicates and equal-predicates, we assume that the predicts in $FC=(fc_1, fc_2, \ldots fc_m, \ldots, fc_n)$ are arranged in a certain order, $fc_i\ (1<=i<=m)$ are range-predicate and $fc_j(m+1<=j<=n)$

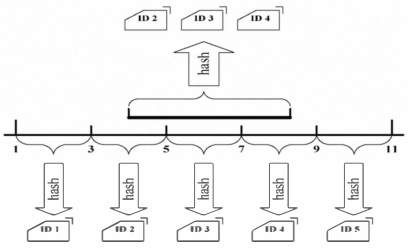


Figure 3. A example of hashing for numerical range

are equal-predicates. Equal-predicate actually declares a point in the space, and the value of the point is a constant which is on the right side of equal-sign, so we can directly hash this constant value and generate a unique hash value. However, for range-predicate, it declares a range in space, so we cannot map it to a unique point by hash function directly. We divide the value space of each attribute into R disjoint but continuous unit-ranges. Any value located in same unit-range will be hashed to same value. We name this hash value as the ID of corresponding unit-range and different range has its own different ID. As we can see from the Figure 3 below, numerical values ranged from 1 to 11 are evenly divided to five ranges, and values in each range will have the same hash value, the ID of the range. Here value ranging from 1 to 11 is eventually hashed into 5 discrete points. As we can see any range declared by range-predicate will have overlap with some continuous unit-ranges. We assume the number of unit-ranges which have overlap with the range of some scope-predicate is $r_i(i=1,2\ldots,m)$, so the FC will be eventually hashed to $\prod_{i=1}^{m} r_i$ n-dimensional Cartesian products (the number of overlap with unit-ranges will

always be 1 to any equal-predicate, so it will have no effects on the size of result set). N is the number of attributes associated with the indexed subscription. Each dimension of the Cartesian product corresponds to a unit-range ID for an attribute. Finally every Cartesian product will be hashed again and result in $\prod_{i=1}^{m} r_i$ hash values in the end. Obviously $\prod_{i=1}^{m} r_i$ is determined by the range declared by each range-predicate. When $\prod_{i=1}^{m} r_i$ is larger, we need to do more hash operations and use more space to save the hash values.

Notification will be treated as a subscription in which predicates are all equal ones. If the notification matches a subscription, it is within the range declared by this subscription, and its hash value will be equal to some IDs generated by this subscription. So we could determine some notification matches a subscription by checking set inclusion of hash results. We noticed that the range declared by a range-predicate would partially cover at most two unit-ranges. Assuming the range of a range-predicate is l_1^i, and the sum of the overlapped unit-range is $l_2^i (l_2^i \geq l_1^i)$.On the premise of values of attribute distributing uniformly, the false positives rate for one attribute is $1 - l_1^i / l_2^i$. Since we have m range-predicates, so the aggregated false positives rate is $1 - \prod_{i=1}^{m} (l_1^i / l_2^i)$. So the granularity of partition of value-space and the rate of false positives compete with each other. If we could learn in advance or predict the distribution of attribute value at publisher side, false positives rate can be reduced further by implementing fine-grained unit-partition at dense-distributing part and coarse-grained one at sparse-distributing part.

After we decompose a subscription to several hash values, we use Bloom filter (BF) to integrate these values into binary vector index. BF is a highly space-efficient random data structure. It can store *n* elements into a bit array of *m* bits. BF will use *d* different hash functions to operate on the same element to obtain *d* hash values *(d<m)* that correspond to *d* positions in the bit array and then set the bits at these *d* positions to 1. When we need to determine whether one element is saved in the bit array, we use the same hash functions on this element to get *d* hash values and search whether the corresponding *d* positions are all 1. If so, this element has been stored, otherwise not. The hash values of subscription are inserted into an empty Bloom filter one by one (the elements number *n* equals to the size of the hash results of subscription). When we want to know whether a notification matches a subscription or not, we just need to determine if the index (hash value) of the notification can be found in the subscription's bloom filter. If so, the event notification matches the current subscription.

As stated before, the rate of false positive for hash-based dimension reduction procedure is $P_1 = 1 - \prod_{i=1}^{m} (l_1^i / l_2^i)$. The BF procedure also introduces false positive which is depended on the size of bit array (m), the number of element stored (n) and the number of hash functions used (d). The rate of false positives for BF is: $P_2 = (1 - (1 - 1/m)^{dn})^d \approx (1 - e^{-dn/m})^d$. The final rate of false positives for the whole indexing procedure is $P = 1 - (1 - P_1)(1 - P_2)$.

D. Dynamic routing of notification and subscription

Although indexing notification and subscription can accelerate matching speed and eliminate false negative, topic-based publish-subscribe above still can't distinguish subscribers with different filter constraints in the same interest group. To implement content-based routing, we proposed a dynamic routing method for notification and subscription. Intrinsically it belongs to rendezvous-based methods. Different from routing-table based methods [9][14][15][16], rendezvous-based method guides notifications and subscriptions to "meet" at the intermediate node in charge of the *keys* that both are mapped to. As mentioned above, the Bloom Filter is a bit array of m bits, we use it to represent the *key*. For comparing, we assign an ID with m bits for every node. So we propose the steps below to identify rendezvous nodes for notification and subscription:

(1) Choose the nodes having the most common 1's with the notification/subscription key. The number of common 1's is the number of 1' in a bits array which is the result of node ID "AND" notification/subscription key.

(2) Choose the nodes which have the most 1's among the ones from step 1.

It should be noticed, even if E≾S, E and S may still have no common rendezvous. So it is required to minimize this possibility.

Lemma: Assuming all the nodes in the system are reachable and the number of 1's of their IDs is same, the less the number of 1's in the node ID, the higher possibility that the notification and matched subscription have the same rendezvous.

Proof: Assuming node N_1 is a rendezvous for notification E, the number of common 1's between N_1 and E, denoted by K_1, is maximum. As E ≾ S, the number of common 1's between E and S, denoted by Y_1, is greater than or equal to K_1, $Y_1 >= K_1$. If N_1 isn't a rendezvous for S, then there must be a rendezvous N_2 which has the greater number of common 1's with S, denoted by Y_2. Since the number of 1's in any node ID, denoted by X, is fixed, so we can have $Y_1 < Y_2 <= X$. Besides, the number of common 1's between N_2 and E, denoted by K_2, should not be great or equal than K_1, otherwise N_2 will fit better to be a rendezvous for E than N_1, which is contradict to the fact that N_1 is a rendezvous for E. Since $K_2 >= 0$, $K_1 > K_2$, $Y_1 >= K_1$, $Y_1 < Y_2 <= X$, so finally we have **$1 <= Y_1 < Y_2 <= X$ (Y_1, Y_2, X should be integers)**. Assuming X=n (n>=1) and $<Y_1, Y_2>$ is the combination of Y_1 and Y_2 that satisfy the constraint above. When Y_2=n, Y_1=1, 2,..., n-1. So there are n-1 valid combinations. Similarly, when Y_2=n-1, there are n-2 valid combinations, and so on. So we have 1+2+…+n-1= n*(n-1)/2 combinations that satisfy the constraint. Because we have n^2 combinations of $<Y_1, Y_2>$ altogether when X=n, it can be inferred that the probability that $<Y_1, Y_2>$ satisfies the constraint is: P= (n*(n-1)/2)/n^2=1/2-1/(2n). As shown in Figure 4, in case of X=4, the number of valid $<Y_1, Y_2>$ combinations is 6, the probability that satisfies the constraint is evaluated to 3/8. Now we can get the conclusion that when n get smaller, the probability that satisfy the constraint ($1 <= Y_1 < Y_2 <= X$) is smaller as well, which means a higher probability that notifications can match subscriptions. □

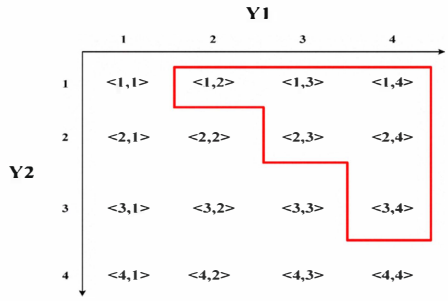


Figure 4. The valid combinations of $<Y_1,Y_2>$ in case of X=4

From the analysis above, we can see if X=1, the probability that notifications and the matched subscriptions have the same rendezvous is 100%, which means every pair of notification and subscription can have a common rendezvous. However, it implies that most of nodes in the system will be chosen as rendezvous for each notification and subscription, which turns the routing procedure into broadcast again. As a result, it is necessary to make a balance between accuracy of routing and the message overhead in real application.

Obviously, both the publishers and subscribers need to make their own notification and subscription disseminated to the possible rendezvous by routing. In order to guide the routing on intermediate nodes, forwarding messages is required to bring with some information. The common message data structure for notification and subscription is shown in Figure 5 (a). **Key_{BF}** represents the Bloom Filter key for notification or subscription. To avoid messages passing through some node repeatedly, **Route** stores the nodes that messages have already reached. **MReplica** and **MHop** are used to prevent message explosion during routing. **MReplica** indicates the allowed maximum number of replicas for a subscription in system, and **MHop** denotes the allowed hop counts for a message routing. Both **MReplica** and **MHop** are statistical optimum values. Source stores the source address of message. Thus, when there is a match at the intermediate nodes, message can be sent directly to the destination.

Data Structure, denoted as RM:
- Key_{BF}: the Bloom Filter key for notification/subscription
- Route: the nodes that have been through
- MReplica: the maximum number of copies allowed (only for subscription)
- MHop: the maximum hop count allowed
- Source: the address of subscribing nodes

(a) The general data structure for routing message

On every node (A):
1. **RM.MHop**--;
2. **RM.Route**.add (A);
3. Find the most possible rendezvous Set_{rn} from neighbor nodes and itself.
4. if Set_{rn} include self
5. **RM.MReplica**--;
6. save **RM** locally;
7. If **RM.MReplica** != 0 && **RM.Hop** != 0
8. **Nodes** = Set_{rn} - **RM.Route**;
9. If **Nodes** ≠ ∅
10. **Nodes** = {neighbors}- **RM.Route**;
11. for node n in **Nodes**
12. copy route message **RM** and replace **MReplica** with (**MReplica**/ **Nodes.size**)
13. forward this new **RM** message to n.

(b) The routing algorithm for subscribing message

Figure 5. Dynamic routing of subscription

Figure 5 (b) shows the routing algorithm for subscription. On every node, subscriptions may be forwarded from its neighbor nodes or generated by itself. **MHop** will be reduced by 1 and the current node ID added to Route when a subscription message coming in, which means the current node has been gone through. Then current node will check itself and its neighbor nodes to find some nodes which are most possible to be rendezvous. If current node is the only one most possible rendezvous compared to its neighbor nodes, **MReplica** will be reduced by 1, and a copy of subscription message will be stored in it (From line 1 to line 6).The message is allowed to be sent out if neither **MHop** nor **MReplica** has been reduced to 0, which means there is a need to find more rendezvous. Firstly, the algorithm will remove the nodes recorded in **Route**. It should be noticed that there exists a probability that the message may have traversed all the candidate nodes. In this case, we choose those nodes that haven't be traversed before from neighbor nodes. At last, in order to ensure that sum of **MReplica** in message copies would not be greater than the original one, **MReplica** in each forwarded copy is averaged by the number of forwarding branches (From line 7 to line 13). As average value of **MReplica** may not be an integer in practical situation, we use round-robin manner in which node distribute the residue one by one in round-robin fashion.

The routing process of notification is basically the same with subscription. When a notification arrives at a node, in order to find matched subscription, it will match all the stored subscription information on that node. Besides, **MReplica** will not be considered when routing a notification, so the propagation distance of the notification message is subject only to value of **MHop**.

III. EVALUATION

A. Experiment Environment

We have implemented the prototype of NovaPS in Java and simulated it on an experimental platform with ten physical nodes connected by 1G bps LAN in our lab. Eight of the nodes are equipped with 2GHz Intel Xeon 8 processor, 4GB RAM running Red Hat with Linux kernel 2.4.21, the others are equipped with 2GHz Intel Xeon 4 processor, 7GB RAM running Red Hat with Linux kernel 2.6.18. For each simulation, 1GB memory is allocated for JVM of SUN jdk1.6.0_12.

B. Experiment Configuration and Results Analysis

Regarding multimedia communication, range-predicates may be used to restrict the area (horizontal and vertical range) of the frame block in audio or video processing applications. Generally the resolution of unit block is 16*16, and the resolution of video is 320*240, like video on Youtube[20] and Youku[21], so the maximum horizontal range is 20 and the maximum vertical range is 15. The number of hash values from a subscription is 300. For the Scaled Video Coding applications, the range-predicates are used to restrict the layers needed for video coding, and the number of layers is no more than 10 in general. So we will get 10 hash results at most. Besides, we can see the values of the attributes (horizontal, vertical and layer, etc) are aligned with respect to the boundary of the unit range. The false positives introduced by hash-based dimension reduction method would be zero, which means that the rate of

false positives of notification-subscription matching depends only on the configuration of Bloom filter. Now we can see, for multimedia communication, our matching introduces neither heavy time/space overhead, nor higher rate of false positives during dimension-reduction.

We run simulations with the number of subscription topics ranging from 1 to 10, and the number of nodes ranging from 1000 to 10000. Every subscription topic is attached with 3 attributes, each of which has an integer value ranging from -5 to 5. Besides, in our experiments, we use three subscription distributions: i) uniform distribution; ii) exponential distribution with mean of 14, which means the 10% of the most popular items account for 51% of the total [18]; iii) Zipf distribution with α exponent setting to 1[19], for choosing topic and range of attributes respectively.

Reliability of the overlay

In this section, we examine the reliability problem in our pub-sub overlay with K-regular random graph topology. As Figure 6 (a) shows, with average number of subscriptions per node from 1 to 5 and 3 regular neighbors per topic, the average node degree is less than 11 with three different subscription distributions, which means each node could build approximate 3-coverage for each subscription topic with 11 neighbor links on average. We also can learn two inferences: i) the average node degree would decrease when the system size is getting large; ii) the average node degree would decrease with the subscription distribution skewed, as illustrated in Table I. For the first inferring, we think that caused by the fact that nodes have more chances to find the counterparts of common interests when the scale of the system becoming larger and more nodes being assigned interests within a limited content space. Similarly for the second inferring, nodes tend to have more common interests while the number of different subscription topics becomes smaller (optional interests shrinking). Moreover, Figure 6(b) and 6(c) present to what degree our NovaPS overlay topology approximates the K-regular random graph. We can see from Figure 6(b) that experimental diameter of the overlay is at most one hop more than the theoretical diameter of the K-regular random graph, which means the worst latency (in hops) between any two nodes is very close to that of the K-regular random graph. The number of disjointed paths in our overlay is also very close to that of the K-regular random graph with k=3, which means on average every node can receive complete messages even if losing connections with its k-1 neighbors. Besides, the results of average distance tell us the average latency between two nodes is almost half of the worst one. (c) gives us a snapshot about the ability of overlay fault-tolerance. The percentage of lived nodes which are not isolated can be greater than 80% even if half of the nodes are removed randomly from the overlay. The isolated nodes increase sharply when more than 50% nodes are removed. It is because the node removal has caused many nodes to lose all their neighbors and reconstruction takes too long time to complete.

TABLE I. NUMBER OF DIFFERENT SUBSCRIPTION TOPICS WITH THREE DISTRIBUTIONS

#nodes	1000	3000	5000	7000	10000
Unifrom's	96	100	100	100	100
Zipf's	50	69	75	74	81
Exponential's	27	30	37	44	45

False positives of Matching

Since we have restricted the maximum size of the result set by subscription dimension reduction (n) to 1000, it's reasonable to set the size of the bits array in Bloom Filter (m) to be double of n. Here we set m to 2048. Firstly, we let n be equal to a randomly generated number, say 343, and adjust the number of hash functions used in Bloom Filter (d) to obtain the curve of the rate of false positives (RFP) when matching 1,000,000 notifications with some subscription using our method. We can see from Figure 7 (a) that the actual RFP is even lower than the theoretical false positive rate value of the Bloom Filter. We consider this is because the sample space is limited in our experiments, even if we have issued 1,000,000 notifications whose attribute values are distributed uniformly in their value space. But an the theoretical RFP and the RFP in our experiments shows the same pattern: when d is small at beginning, it is unlikely that two different notifications will mapped to the same index; but when d increases continually, it will cause more mapping conflicts and raise RFP after some threshold point. In our case, the threshold point is d=4. Figure 7 (b) shows another phenomenon.When d is fixed to 5, we find that RFP doesn't always increase while n increasing. Actually RFP begins to decrease at the point of n=600, which may contradict with analytical result of the Bloom Filter. We

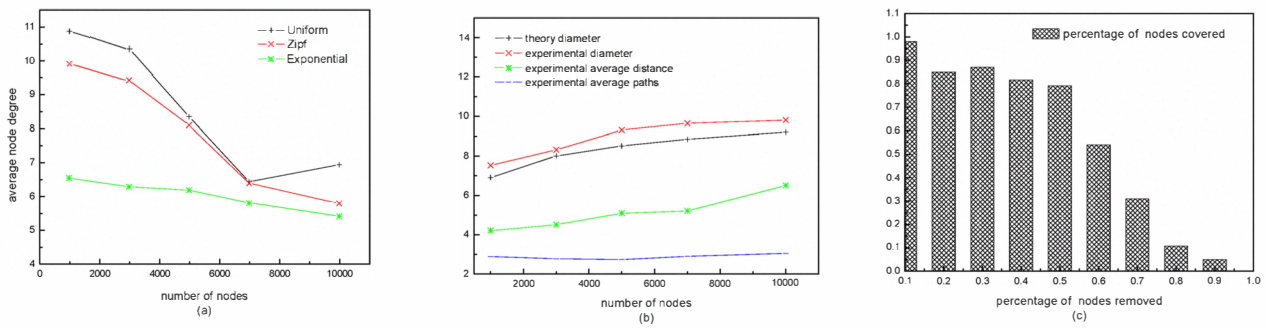


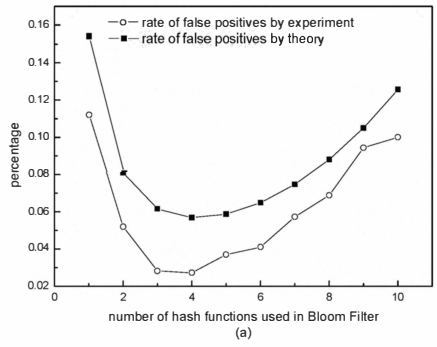
Figure 6. Reliability of NovaPS overlay with 5000 nodes

consider the reason for this phenomenon is the value space of the attributes in our experiments is limited. When n becomes larger, the ranges of the attributes in subscription are close to the maximum, the possibility of a notification falling outside a subscribed space is smaller. So the RFP decreases correspondingly.

According to theoretical analysis of the Bloom Filter, we can get, for a given m and n, the number of hash functions that minimizes RFP is $d = \frac{m}{n} ln2$, as proved in Figure 7 (a). So taking the optimal d, the expected RFP p and fixed n, we have optimal $m = -\frac{nlnP_2}{(ln2)^2}$ [12]. Assuming the RFP of BF expected is 1%, optimal $m=9.5n$. So while obtaining the most popular value of n in some hotspot subscriptions, we can control RFP by adjusting m.

Efficiency of dynamic routing

In this section, we focused on verify if our dynamic routing method can help to achieve the balance between message cost and coverage of subscription in an interest group. In all experiments, we set maximal hops for routing message (**MHop**) and maximal number of copies of subscriptions (**MReplica**) to log(N), which N is the total number of nodes in this group. As illustrated in Figure 8 (a), the cumulative percentage of notifications is the percentage of notifications that meet less than b percent of matched subscriptions during routing. Here b is called notification coverage, which implies the ability of our dynamic routing method to deliver a notification to its matched subscriptions. From the figure we can see the cumulative



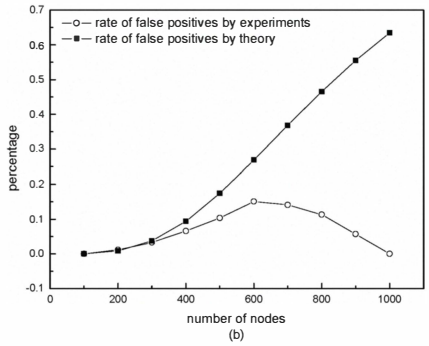


Figure 7. Comparison for RFP with experiments and theory analysis

percentage of notifications will decrease with the decreasement of number of 1's in node ID (nb_1) when considering same value of notification coverage. This verified the theory analysis above which has declared that the possibility of notification meet a matched subscription during routing will increases with the decreasement of nb_1. We also can infer most of notifications (>=80% when nb_1 <=3) can achieve the coverage of more than 50%. Figure 8 (b), (c) confirm our inferring. In Figure 8 (b), the coverage averaged by 200,000 notifications gets very close to 50% even nb_1 is set to 10. And Figure 8 (c) gives us the reason of choosing relatively big value of nb_1 even it's known for sure that bigger nb1 would deteriorate the notification coverage. It verifies the analysis above again that bigger nb_1 can make the possibility of notification meeting matched subscription lower with the reward that routing message cost is reduced as well. So it gives us a way to balance bandwidth overhead and delivery performance in real applications.

IV. CONCLUSION AND FUTURE WORKS

We have presented NovaPS, a robust overlay for multimedia communication in dynamic environment. As shown in our evaluation, NovaPS can scale well when subscription distribution is skewed and system size is big. It can deliver notifications to all matched subscribers with high reliability even half of the nodes happened an error. Due to the content matching is crucial for improving multimedia performance in Publish/Subscribe, we adopt a index-based method to speed matching between notification and subscription, and obtain a controllable false positives rate through compromising a little space overhead. Dynamic routing in NovaPS gives us another chance to find the tradeoff between communication overhead and delivery performance when considering bandwidth cost. We can also see from the evaluation above, a relatively small compromising of delivery performance can lead to big reduction of communication overhead. In multimedia communication applications which have requirement of 100% delivery ratio, we could use the strategy that firstly each node can deliver messages in the way of dynamic routing whenever a notification or subscription comes, then it can bundle a number of received notifications in a broadcasting message, and broadcasting this message in a relative big period, which can mitigate the bandwidth pressure and guarantee 100% delivery ratio meanwhile.

As we said before, building NovaPS relies on the membership management to find neighbors with similar interests. Membership management should consider the clustering of interests (subscription) when allocating local membership views for each node. How to measure the similarity of interests is a key problem. In the future, we consider using network delay between nodes, the overlap of interests between nodes and the history behavior of the nodes jointly to measure proximity relationship between two nodes. We think node i should choose j as neighbor when they have smaller communication latency. Similarly, when the j's interests are more similar to i and j is tested to be harmless, i should choose j as neighbor.

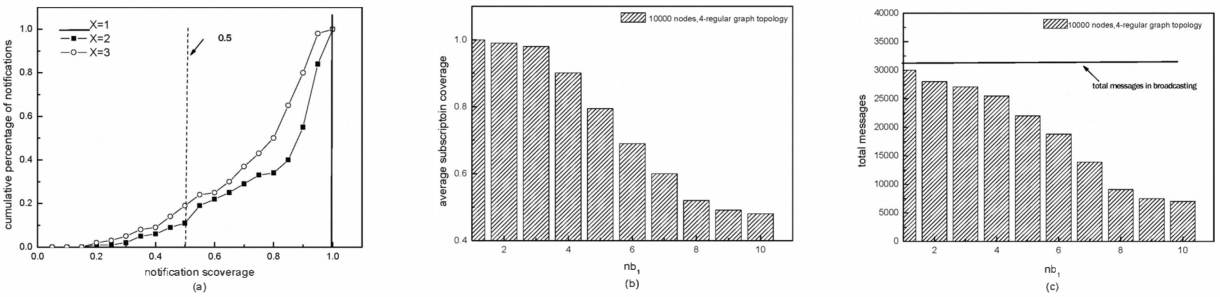


Figure 8. Efficiency of dynamic routing with 200,000 notifications issued totally

ACKNOWLEDGMENT

We would like to thank Dr. Chengjian Wen for his hearty helps on theoretical working in this paper. The NovaPS project is funded by the National High Technology Development Program (863 Program) under Grand No. 2009AA01Z144, No. 2006AA01A106, No. 2009AA01A131.

REFERENCES

[1] K.Betz, "A scalable stock web service." In:Proc.of the 2000 International Conference on Parallel Processing,Workshop on Scalable Web Services.

[2] I.M.Arntzen and D.Johansen, "A stateful and open publish-subscribe structure for online marketplalces." In:J.Dingel and R.Strom editors,Proc.of the 4th International Workshop on Distributed Event-Based Systems(DEBS'05).

[3] Hongzhou Liu and Emin Gun Sirer, "A measurement study of RSS,A publish-subscribe system for web micronews." In:Proc.of Internet Measurement Conference(IMC).2005.New Orleans,Louisiana.

[4] Viktor S. Wold Eide , Frank Eliassen , Ole-Christoffer Granmo , Olav Lysne, "Scalable Independent Multi-level Distribution in Multimedia Content Analysis," in Proceedings of the Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems: Protocols and Systems for Interactive Distributed Multimedia, p.37-48, November 26-29, 2002 .

[5] C.Bornhovd,M.Cilia,C.Liebig,et al, "An infrastructure for meta-auctions." In:Proc.of Second International Workshop on Advance Issues of E-Commerce and Web-based Information Systems(WECWIS'00).2000.San Jose,

[6] A.R.Bharambe, S.Rao and S.Seshan, "Mercury:A scalable publish-subscribe system for Internet games." In:Proc.of the 1st Workshop on Network and System Support for Games[C].2002.Braunschweig,Germany:ACM Press.p.3-9.

[7] D.-P. Wu, Y.T. Hou and Y.-Q. Zhang, "Scalable video coding and transport over broadband wireless networks." In Proceedings of IEEE 89(1) (2001).

[8] A. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Scamp: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication," in Proc. Third Int'l Workshop Networked Group Comm., Nov. 2001.

[9] A. Carzaniga, D. S. Rosenblum, and A. L.Wolf, "Design and evaluation of a wide-area event notification service." *ACM* Transactions on Computer Systems, 2001.

[10] R. Melamed, I. Keidar, "Araneola: A scalable reliable multicast system for dynamic environments." in: Third IEEE International Symposium on Network Computing and Applications (IEEE NCA), 2004.

[11] N. c. Wormald, "Models of random regular graphs." surveys in Combinatorics, 1999 (LMS Lecture Note Series 267, Eds J.D.Lamb and D.A.Preece), 239-298.

[12] Chazelle. Bernard, Kilian. Joe, Rubinfeld.Ronitt, Tal. Ayellet, "The Bloomier filter: an efficient data structure for static support lookup tables." In Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 30–39 ,2004.

[13] Gregory Chockler , Roie Melamed , Yoav Tock , Roman Vitenberg, "SpiderCast: a scalable interest-aware overlay for topic-based pub/sub communication." In Proceedings of the 2007 inaugural international conference on Distributed event-based systems, June 20-22, 2007, Toronto, Ontario, Canada

[14] G. Cugola, E. Di Nitto, and A. Fuggetta, "The JEDI event-based infrastructure and its application to the development of the OPSS WFMS." *IEEE Trans. Softw. Eng.*, 27(9):827–850, 2001.

[15] G. M¨uhl, L. Fiege, F. C. G¨artner, and A. Buchmann. "Evaluating advanced routing algorithms for content-based publish/subscribe systems." In MASCOTS '02: Proc. 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications *Systems (MASCOTS'02)*, page 167, Washington, DC, USA, 2002. IEEE Computer Society.

[16] P. Pietzuch and J. Bacon, "Hermes: A Distributed Event-Based Middleware Architecture." In 1st International Workshop on Distributed Event-Based Systems (DEBS'02), July 2002.

[17] S. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," IEEE J. Select. Areas Commun., vol. 16, pp. 983–1001, Aug. 1997.

[18] Y. Tock, N. Naaman, A. Harpaz, and G. Gershinsky, "Hierarchical clustering of message flows in a multicast data dissemination system." In 17th IASTED Int'l Conf. Parallel and Distributed Computing and Systems, 2005.

[19] H. Liu, V. Ramasubramanian, and E. G. Sirer. "Client behavior and feed characteristics of rss, a publish-subscribe system for web micronews." In *IMC*, 2005.

[20] YouTube. http://www.youtube.com/.

[21] YouKu. http://www.youku.com/.