Achieving Bounded Delay on Message Delivery in Publish/Subscribe Systems

Jinling Wang[1, 2], Jiannong Cao[1], Jing Li[2], Jie Wu[3]
[1] *Department of Computing, Hong Kong Polytechnic University*
[2] *Institute of Software, Chinese Academy of Sciences, Beijing, China*
[3] *Department of Computer Science and Engineering, Florida Atlantic University*
Corresponding author: Prof. Jiannong Cao, csjcao@comp.polyu.edu.hk

Abstract

Publish/subscribe (pub/sub) systems are very suitable for the dissemination of dynamic information over the Internet. As dynamic information is usually characterized by a short lifetime, both publishers and subscribers may specify the delay requirement on message delivery. Although existing pub/sub systems can easily be extended so that publishers and subscribers can specify their delay requirements, it remains a challenging problem to improve the efficiency of pub/sub systems so that as many messages can be successfully delivered as possible, while the network traffic does not increase significantly. In this paper, we propose an efficient approach for pub/sub systems to achieve bounded delay on message delivery. Three message scheduling strategies are proposed for the system to make use of available bandwidth efficiently. Simulation results show that our strategies enable subscribers to receive significantly more valid messages than traditional strategies, while the network traffic just increases slightly.

1. Introduction

A Publish/Subscribe (pub/sub) system is a type of message-oriented middleware that supports loosely coupled interaction in distributed environments. A pub/sub system is composed of three types of participants: *information publishers*, *information subscribers*, and *message brokers*. Information subscribers issue subscriptions to message brokers specifying their interest in certain information. Information publishers send information to message brokers. The message brokers forward the published information to all relevant subscribers. In a large-scale pub/sub system, there are usually multiple message brokers that are organized into an overlay network to forward messages from publishers to subscribers. As the pub/sub system enables subscribers to flexibly express their interests in certain information and get the needed information in a timely way, it is very suitable for the dissemination of dynamic information over the Internet, such as stock trading information, auction information, traffic information, etc.

Dynamic information is usually characterized by a short lifetime; an expired message is usually useless to end users. In some existing industrial standards of pub/sub systems such as JMS [1] and CORBA Notification Service [2], publishers can specify the allowed delay for a published message. Nevertheless, in some cases subscribers may also want to specify the allowed delivery delay; different subscribers may have different delay requirement for the same message. For example, for a message about the traffic information of a certain place, the subscribers near the place may require a shorter delay than those far from the place. Considering the different quality of service provided to different subscribers, the service providers may charge for different fees to the subscribers.

Although delay bound is a very important QoS requirement for pub/sub systems, little work has been done on it so far. While we can easily extend the existing pub/sub model to allow the publishers and subscribers to specify their delay requirement, it remains a challenging problem to improve the efficiency of pub/sub systems to ensure that as many messages can be successfully delivered as possible, while the network traffic does not increase significantly.

Message scheduling is one of the key issues for a pub/sub system to efficiently support bounded delay. For all messages that are waiting to be sent out on a message broker, the broker should give priority to those that have shorter remaining lifetime and/or satisfy more subscriptions. In the case that the subscribers pay a different price for different delay requirement, priority should also be given to the messages for which the

This work is partially supported by University Grant Council of Hong Kong under the CERG grant B-Q822 (PolyU 5183/04E), and the Hong Kong Polytechnic University under the ICRG grant G-YD63.

subscribers agree to pay more. Furthermore, to avoid wasting network resource, the brokers should delete as early as possible the messages in transit that have expired.

In a large-scale pub/sub system, there are a number of factors that make efficient message scheduling very difficult. First, there may be multiple subscribers for a given message, each with different delay requirement and price. Thus, it becomes an NP-complete problem to get an optimal schedule[1]. Second, a message usually goes through multiple brokers before it reaches a subscriber, and the end-to-end delay of the message is affected by the scheduling decisions of all these brokers. Finally, a broker needs to know the bandwidth available to each subscriber in order to decide how to schedule the messages, while the current Internet can just provide a best-effort transmitting service and the actual bandwidth varies from time to time.

In this paper, we propose an efficient approach for pub/sub systems to support bounded delay on message delivery. We first assume that the available bandwidth of each link in the broker network satisfies a certain probability distribution, and the parameters of the distribution can be estimated based on the measured data. Then we propose three metrics to guide the scheduling of message delivery: *Expected Benefit* (EB), *Postponing Cost* (PC), and *Expected Benefit plus Postponing Cost* (EBPC), and the corresponding scheduling strategies are *maximum EB first*, *maximum PC first*, and *maximum EBPC first*. The EB metric of a message means the expected benefit gained by sending the message in the first place. The PC metric of a message means the expected cost of postponing the sending of the message, which reflects the urgency of the sending task. The EBPC metric combines the EB and PC together to determine the order of message delivery.

We evaluated the performance of the proposed approach by simulations and compared our scheduling strategies with two widely used strategies in the network community, namely *FIFO* and *minimum remaining lifetime first*. Simulation results show that our strategies enable subscribers to receive significantly more valid messages than the traditional strategies, while the network traffic increases only slightly. To the best of our knowledge, this is the first work that addresses the message scheduling in pub/sub systems.

The remainder of the paper is organized as follows. In Section 2, we discuss the related work, comparing our approach with some existing solutions. Section 3 describes the system topology, delay model and routing protocol we assumed. Section 4 presents the system objectives and the data structure maintained by each broker. Section 5 describes the proposed message scheduling strategies. In Section 6, we describe the simulations for performance evaluation and discuss the simulation results. Finally, in Section 7, we conclude the paper with a summary.

[1] The problem can be reduced from the *job scheduling with penalties* problem [3], which is NP-complete.

2. Related Work

Delay bound for packet delivery has been extensively studied in network research communities. Existing works mainly use one of the two mechanisms: *resource reservation* and *priority control*. The resource reservation mechanism requires all participating nodes reserve enough resource apriori to guarantee the QoS of a certain transmission task. As the mechanism involves the negotiation, reservation, allocation and release of resources among all participating nodes, it is complex and does not scale well. On the other hand, in the priority control mechanism, each node determines the schedule of message delivery based on local information without negotiation with other nodes, so it is more scalable but may not strictly guarantee the QoS of a transmission task.

In terms of the network protocol stack, existing works can be classified into one of the three layers: MAC, IP, and overlay. Some representative works are shown in Table 1. According to this classification, our work is on the overlay layer and makes use of the priority control mechanism.

Table 1: Some representative works on delay bound

	MAC	IP	Overlay
Resource reservation		IntServ/RSVP[4]	QRON [5]
Priority Control	IEEE 802.11e[6]	DiffServ [7]	OverQoS[8]

As the current Internet can only provide the best-effort service, a number of works [5, 8, 9] have been done to provide QoS on the overlay layer so that the Internet can support real-time applications such as multimedia streaming. Similar to our work, these works also use measured data to get the characteristics of the underlying Internet connection, such as available bandwidth, delay, and packet loss rate, and then provide QoS-aware routing to the upper applications. However, the works described in [5, 9] assume that the available bandwidth of each link in the overlay is fixed without considering the dynamics of the underlying Internet. The work in [8] is similar to our work in that it also assumes that the available bandwidth satisfies certain probability distribution, but it performs QoS-aware routing based on the lower bound of the available bandwidth, i.e., the value that can be guaranteed with a high probability. Compared with these works, our work

performs message scheduling based on the parameters of the probability distribution of the available bandwidth, which can make use of available bandwidths more efficiently. Furthermore, the indirect communication of pub/sub systems is very different from the traditional unicast or multicast communication, which makes the existing QoS work on overlay networks hardly applicable to pub/sub systems.

Although a lot of work has been done on the pub/sub systems in recent years [10-16], little has been done on the QoS-related issues. As far as we know, the only work that has considered the bounded delay problem is in [16], where the authors proposed to use the resource reservation mechanism to ensure the bounded delay of message delivery. However, a pub/sub system differs from the traditional communication systems in that there is no end-to-end connection between message senders and receivers. As there may be different delay requirements for different (*publisher*, *subscriber*, *message*) tuples, the use of resource reservation mechanism would be extremely expensive if not impossible.

3. System Model

In this section, we introduce the system topology, delay model and routing protocol on which our work is based.

3.1. System Topology

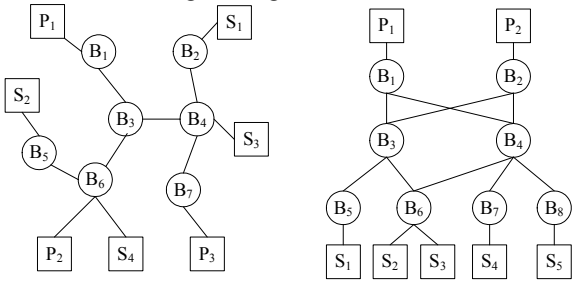
There are mainly two types of topology for the overlay of pub/sub systems:

1) Acyclic graph, as applied in Siena [10], JEDI [11], and Rebeca [12]. In this topology, there is only one path between any pair of message brokers. Each broker can serve both publishers and subscribers.

2) Mesh, as applied in DCP [13], Gryphon [14], and XRoute [15]. In this topology, there may be multiple paths between a pair of brokers. Some brokers connect to publishers, some to subscribers, while the other brokers do not connect to either publishers or subscribers but work as intermediate nodes to forward messages.

The two types of topology of pub/sub systems are shown in Figure 1. In the figure, message brokers are denoted by B_i, publishers denoted by P_i, while subscribers denoted by S_i.

In conventional scenarios for dynamic information dissemination, there may be just a small number of publishers while a large number of subscribers, so the mesh topology is more suitable for such applications. Therefore, our work is based on the mesh topology of pub/sub systems. In the following, we call the neighbors of a broker through which the broker reaches publishers the *upstream neighbors*, and the neighbors through which the broker reaches subscribers the *downstream neighbors*. TCP protocol is supposed to be used in forwarding messages between brokers.



a) Acyclic graph **b) Mesh**

Figure 1. Two types of topology for pub/sub systems

3.2. Delay Model

The end-to-end delay of a message is composed of the delay on each link as well as the delay on each broker in the path from the publisher to the subscriber. We will discuss these two types of delay respectively.

1. Delay on each broker

The delay on a broker is determined by the way the broker processes the message. Generally speaking, the function of a broker can be divided into three modules: *message receiving*, *message processing*, and *message forwarding*, as shown in Figure 2.

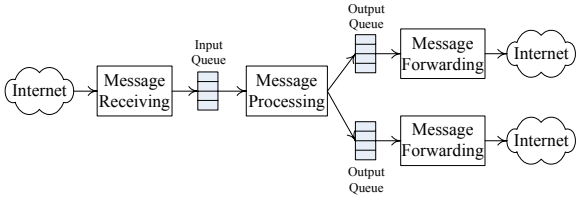


Figure 2. Functional modules of a message broker

Each broker maintains two types of message queues: input queue and output queue. There is only one input queue that is used to store the unprocessed incoming messages. On the other hand, one output queue is created for each downstream neighbor, which is used to store messages waiting to be sent to the neighbor.

As with most research on the delivery delay, we ignore the time a message spends on waiting in the input queue[2]. Therefore, the delay of a message on a broker is composed of the following two parts: 1) the time spent in the output queue, which is called *scheduling delay*; 2) the time required by the message processing module, which is called *processing delay*. We assume that the processing delay of each broker for each message is of the same value, denoted by *PD*.

[2] The size of the input queue is greater than 0 only when the message arrival rate is greater than the processing rate of messages, which rarely happens as the bottleneck of the system is usually on the network transmission.

2. Delay on each link

Each link in the overlay network of a pub/sub system is actually an end-to-end connection of the underlying Internet. In recent years, research on the Internet measurement shows that, although the Internet cannot provide a deterministic guarantee on the transmission delay, the end-to-end delay is generally stable. The end-to-end delay of an IP packet can be considered to satisfy the shifted gamma probability distribution [17, 18], while the variation is surprisingly small[3]. The measured data in [8] also shows that the end-to-end available bandwidth on the Internet is fairly stable. Therefore, we can reasonably assume that the available bandwidth of each link in the pub/sub system satisfies a certain probability distribution.

Although the one-way delay of an IP packet satisfies the shifted gamma distribution, the transmission rate of a TCP connection is jointly determined by the round trip time of IP packets and the size of the TCP window, while the latter is further affected by many factors. Therefore, we assume the available bandwidth of each link in the pub/sub system satisfies the normal distribution. For each link l_i in the overlay of a pub/sub system, we use the average time needed to transmit one kilo-byte of data to measure its transmission rate, denoted by TR_i. We denote the probability distribution of TR_i by $TR_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i and σ_i^2 are the mean value and variation respectively. Each broker estimates the parameters of the probability distribution of the transmission rate to each neighbor by some tools of network measurement.

We assume the transmission rates of different links are independent, so the transmission rate of a path composed of multiple links also satisfies the normal distribution. For a path p composed of n links $l_1, l_2, \ldots, l_n$, let TR_p denote the transmission rate of the path. We have $TR_p \sim \mathcal{N}(\sum\mu_i, \sum\sigma_i^2)$.

We call the delay needed for a message to be transmitted along a path (exclude the delay on the brokers in the path) the *propagation delay* of the message. For a message with size of m kilo-bytes, its propagation delay on path p is $m \times TR_p$.

3.3. Routing Protocol

The routing protocol of pub/sub systems determines the path via which the messages in the system travel from the senders to the receivers. Existing routing protocols for mesh-structured pub/sub systems fall into two categories: *single-path* routing [14, 15] and *multi-path* routing [13]. In single-path routing, the system just selects one path for a message to travel from a publisher to a subscriber, while in multi-path routing, a message are transmitted via all possible paths from a publisher to a subscriber to improve reliability. To decrease the network traffic, in our solution we assume the single-path routing protocol is used. The criterion for path selection is to minimize the mean value of the transmission rate of the path.

4. Preliminaries

4.1. System Objectives

To simplify discussion, we mainly focus on the following two scenarios in this paper:

- Publisher-specified delay (PSD): the allowed delay of message delivery is specified by publishers while subscribers do not specify any delay requirement.
- Subscriber-specified delay (SSD): Each subscriber specifies its requirement on the allowed delay of message delivery and at the same time gives a price for each successfully arriving message, while publishers do not specify any delay requirement.

Our work can easily be extended to the case where both publishers and subscribers specify their delay requirements on message delivery.

In the PSD scenario, the objective of the system is to make subscribers receive as many valid messages as possible. Suppose the publishers have published k messages in a given period, denoted by $m_1, m_2, \ldots, m_k$. For a message m_i, let the number of subscribers interested in it be ts_i, and the number of subscribers that receive it before the deadline be ds_i. We can define a metric called *delivery rate* of the system as follows:

$$\sum_{i=1}^{k} ds_i \Big/ \sum_{i=1}^{k} ts_i \qquad (1)$$

The system objective is to maximize the above delivery rate.

In the SSD scenario, as different delay requirements correspond to different prices, the system objective is to maximize the *total earning* rather than the delivery rate. Suppose there are n subscribers in the system, denoted by $s_1, s_2, \ldots, s_n$. Let $price(s_i)$ denote the price of each valid message provided by subscriber s_i, and $msg(s_i)$ denote the number of all valid messages received by subscriber s_i in a given period. The total earning of the system in the period can be calculated as follows:

$$\sum_{i=1}^{n} price(s_i) \times msg(s_i) \qquad (2)$$

[3] According to the measured data in [18], the mean value of the one way end-to-end packet delay on a cross-Atlantic path with 22 hops is 108.2 ms, while the standard error is just 3.083ms.

4.2. Data Structure

To achieve the system objectives, the brokers should have the necessary information to perform efficient message scheduling. In our solution, each broker maintains a *subscription table* with the following format:

$$\{(subscriber, filter, dl, pr, nb, NN_p, \mu_p, \sigma_p^2)\}$$

where each item describes the following information of a subscription:

- A *subscriber* is interested in all messages that satisfy the given *filter*.
- The subscriber has specified the worst-case delay *dl* allowed for the messages, and is willing to pay the price *pr* for any valid message.
- The current broker can reach the subscriber via the neighbor *nb*.
- Let p denote the path from the current broker to the subscriber. The variable NN_p is the number of intermediate nodes on the path p, while μ_p and σ_p^2 are the mean value and the variation of the transmitting rate of the path respectively.

To simplify description, we assume each subscriber just issues one subscription. Hereafter, we do not strictly differentiate a subscriber from its subscription.

5. Message Scheduling Strategies

In this section, we first introduce the three message scheduling strategies proposed for pub/sub systems, and then introduce the mechanism used by the system to detect invalid messages in transit.

We mainly focus on the SSD scenario in this section as it is more complex than the PSD scenario. To apply the proposed scheduling strategies in the PSD scenario, we just need to set the price in the corresponding expressions to be 1, and change the delay requirement to be specified by publishers rather than subscribers.

5.1. Maximum EB First

The EB value of a message is the expected earning it can bring to the system when all remaining nodes always send the message in the first place. As the objective of the system is to maximize the total earning, it is reasonable to send the messages with maximum EB values first.

For a message m waiting to be sent out on a broker N, suppose it can satisfy n subscriptions denoted by s_1, s_2, …, s_n. Let the function $success(s_i, m)$ denotes the probability that message m can arrive at s_i in the specified delay. The EB value of message m (denoted by EB_m) is calculated as follows:

$$EB_m = \sum_{i=1}^{n} succcess(s_i, m) \times price(s_i) \qquad (3)$$

According to the above expression, the EB value of a message is determined by the number of subscriptions that are interested in the message, the prices of the subscriptions, and the probability that the message arrives at the destinations before deadline. As a result, in the *maximum EB first* scheduling strategy, a message will have a higher priority in delivery if it can satisfy more subscriptions, or it has higher probability to arrive at the destinations in time, or the subscriptions provide higher prices for the message.

Now we discuss how to calculate the function $success(s_i, m)$. Let $hdl(m)$ denotes the delay that has already occurred when the message m arrives at the current broker. The current broker can get the value of $hdl(m)$ by subtracting the publishing time of the message from the current time. For a subscription s_i that is interested in the message, let the function $fdl(s_i, m)$ denote the delay that will occur before message m reaches s_i. It is composed of the following three parts (let p denote the path from the current node to s_i):

- The processing delay on all nodes in path p;
- The scheduling delay on all nodes in path p;
- The propagation delay in path p.

As the scheduling delay on other nodes is unknown to the current node, we just assume the scheduling delay of the message on all nodes in path p is 0. So the function $fdl(s_i, m)$ can be calculated as follows:

$$fdl(s_i, m) = NN_p \times PD + size(m) \times TR_p \qquad (4)$$

where $size(m)$ is the size of message m, NN_p is the number of nodes on path p and TR_p is the transmitting rate of p.

Let the function $adl(s_i)$ denote the maximum allowed delay required by the subscriber s_i. The value of $success(s_i, m)$ can be calculated as follows:

$$success(s_i, m) = P\{hdl(m) + fdl(s_i, m) \le adl(s_i)\} \qquad (5)$$

As the probability distribution of TR_p is known, we can easily get the value of the function for any messages and subscriptions.

5.2. Maximum PC First

We notice that in the *maximum EB first* strategy, the message with higher probability to successfully arrive at destinations is sent with a higher priority. However, the higher probability of successful delivery means that the delivery of the message is not very urgent. In other words, it may be harmless to postpone the sending of the message for a period, so that other messages with lower success probability can be sent out in advance. To overcome this disadvantage of the *maximum EB first* strategy, we propose another scheduling strategy in which the order of message delivery is determined by the urgency degree of messages.

We use the *postponing cost* (PC) to indicate the urgency degree of a message. For a message m that is

waiting to be sent out on broker N, the *aforementioned EB* metric is the expected benefit it can bring to the system if all remaining nodes send it in the first place. We can calculate another expected benefit, denote by *EB′*, when the current node sends the message in the second place, while all other nodes still send it in the first place. We call the value of ($EB - EB'$) the *postponing cost* of the message, which means the expected cost when the current broker postpones the sending of the message once. The sending task of the message is considered to be urgent if the cost is high.

To compute the value of *EB′*, we must know the time needed to send out the first message. As the first message has not been chosen yet at this time, we estimate the time as the average size of all messages multiplied by the mean value of the transmitting rate on the link from the current node to the corresponding neighbor, denoted by *FT*.

In the case that the current node sends the message m in the second place while all other nodes send it in the first place, let $fdl'(s_i, m)$ denote the delay that will occur before m reaches subscriber s_i, and $success'(s_i, m)$ denote the probability that message m arrives at s_i in the required delay. We can calculate them with the following expressions:

$$fdl'(s_i, m) = fdl(s_i, m) + FT \quad (6)$$

$$success'(s_i, m) = P\{hdl(m) + fdl'(s_i, m) \le adl(s_i)\} \quad (7)$$

So we get the *EB′* value and postponing cost (denoted by PC_m) of message m as follows:

$$EB'_m = \sum_{i=1}^{n} succcess'(s_i, m) \times price(s_i) \quad (8)$$

$$PC_m = EB_m - EB'_m \quad (9)$$

5.3. Maximum EBPC First

As the expected benefit and the postponing cost are both important factors in determining the order of message delivery, we design a new metric that combines the two factors together, called *Expected Benefit plus Postponing cost* (EBPC). In the *maximum EBPC first* strategy, the order of message delivery is determined by both the expected benefit and the urgency degree of the messages. The calculation of EBPC value for message m is as follows:

$$EBPC_m = r \times EB_m + (1 - r) \times PC_m \quad (10)$$

where r is the weight of EB in determine the order of message delivery. The range of r is [0, 1], and the actual value can be set empirically to achieve the best results.

5.4. Detection of Invalid Messages

The system should delete any message that cannot arrive at any subscriber within its specified delay period as early as possible to avoid the unnecessary network traffic. To improve the utility of available bandwidth, in our solution, the brokers not only delete the messages that have already expired, but also the messages that are unlikely to arrive at any subscriber within the specified delay, although they may not expire yet.

For a message m waiting to be sent out on a broker N, suppose it can satisfy n subscriptions denoted by s_1, s_2, …, s_n. The condition to delete the message on broker N is as follows:

$$\forall i \in \{1,...,n\} : success(s_i, m) \le \varepsilon \quad (11)$$

where ε is a relatively small value. The value of ε is set to 0.05% in our simulation described in the next section.

6. Performance Evaluation

We have implemented the proposed scheduling strategies in Java and evaluated their performance with a simulated network and a variety of simulated work load[4]. In this section, we describe our experimental study and discuss the performance evaluation results.

6.1. Simulation Setup

We simulate a layered structure of broker network as shown in Figure 3， which is similar to the topology used in [13, 14]. In the simulated network, there are 32 brokers that are divided into 4 layers. In the first layer of the broker network there are 4 brokers each connected to a message publisher. In the second layer there are 4 brokers each connected to all brokers of the first layer. In the third layer there are 8 brokers, each randomly connected to 2 brokers of the second layer. In the fourth layer there are 16 brokers, each randomly connected to 2 brokers of the third layer. Each of the fourth-layer brokers serves 10 message subscribers, so there are a total of 160 subscribers in the system.

The mean value of the transmission rate on each link is randomly generated in the range of 50 milliseconds (ms) to 100 ms, and the standard error of the transmission rate on each link is 20ms. The processing delay of each message on a broker is 2ms.

Each publisher continuously publishes messages at a certain rate; we call the average number of messages published by each publisher per minute as the *publishing rate* of the system. The length of the whole test period is 2 hours. The size of each message is 50K bytes. The message head of each message is composed of the following contents: $\{A_1=x_1, A_2=x_2\}$, where A_1, A_2 are attribute names and x_1, x_2 are *double*-type values randomly chosen from the range of (0, 10).

Each subscriber has defined a subscription with the form of "$A_1<x_1 \wedge A_2<x_2$", where x_1, x_2 are also values randomly chosen from the range of (0, 10). Therefore, for each published message, there are on average

[4] The implementation codes can be downloaded from http://jlwang.rocklv.net/pubsub_bd/pubsub_bd.html

$(1/2)^2$=25% subscribers in the system that are interested in it.

We simulate the delay requirement in the PSD and SSD scenarios respectively. In the PSD scenario, the delay requirement of each message is randomly generated in the range from 10 seconds to 30 seconds. In the SSD scenario, the delay requirement of each subscription is randomly chosen from the set {10s, 30s, 60s}, and the corresponding price is {3, 2, 1} respectively.

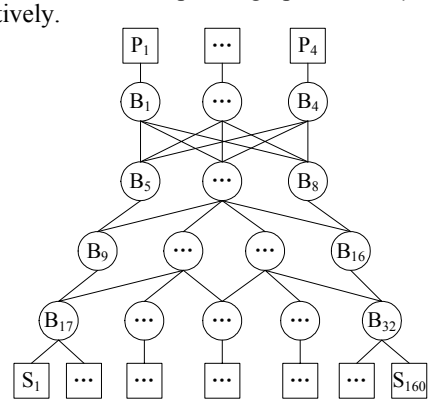


Figure 3. Topology of the simulated network

In addition to our proposed scheduling strategies, we also implemented two other widely used scheduling strategies: *FIFO* and *minimum remaining lifetime (RL) first*, to evaluate the performance of them under the same environment and workload conditions. Hereafter, we call the five strategies as the EB, PC, EBPC, FIFO and RL strategies respectively.

In the SSD scenario, as there may be multiple subscribers interested in a same message, the message may have multiple remaining lifetimes each corresponding to a subscription, which makes it difficult to apply the RL strategy. To solve this problem, we use the average value of the remaining lifetimes of each message for scheduling.

The following metrics are defined to evaluate the performance of the different strategies:

- *Delivery rate*: used in the PSD scenario, as defined by expression (1);
- *Total earning*: used in the SSD scenario, as defined by expression (2);
- *Message number*: the total number of messages received by all brokers, which reflects the overall traffic on the network.

6.2. Simulation Results

Figure 4 shows the performance comparison of the EB, PC and EBPC strategies. In this group of experiments, the publishing rate is 10, and the value of *r* in expression (10) varies from 0 to 100%. Figure 4(a) shows the total earning of the three strategies in the SSD scenario. It shows that the performance of the PC strategy is worse than the EB strategy, while the combination of EB and PC gets some advantage when *r* is in the range of (23%, 100%). Figure 4(b) shows the delivery rate of the three strategies in the PSD scenario. The performance of EB strategy is close to that of PC strategies, while the combination of EB and PC is always better.

Since the performance of the EBPC strategy is close to that of the EB strategy with slight improvement, we just compare the EB, PC, RL and FIFO strategies in the following. Figure 5 shows the performance comparison of the four strategies in different publishing rates in the SSD scenario. Figure 5(a) shows that the total earning always increases in the EB and PC strategies, while the EB strategy can achieve more earnings than the PC strategy. On the other hand, in the FIFO and RL strategies, the total earning decreases after it reaches the peak. The reason is that with the increase of publishing rate, the congestion of the network becomes more and more serious, resulting in fewer messages arriving at the destinations in time. The RL strategy has the worse performance because in a pub/sub system composed of multiple brokers, the messages with very small lifetime can hardly reach subscribers in the required delay. If these messages are sent out first, the bandwidth is vainly expended resulting in the poor utility of network resource.

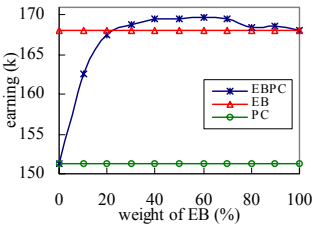
Figure 5(b) shows the number of messages generated by the four strategies in the SSD scenario. The EB and PC strategies incur almost the same network traffics, which is a little more than that of FIFO and RL strategies. When the publishing rate is 15, the EB strategy incurs 23% and 64% more messages than the FIFO and RL strategies respectively, while the total earning of the EB strategy is 5 and 10 times that of the FIFO and RL strategies. Therefore, the EB strategy can make use of the network resource more efficiently.

Figure 6 shows the performance of the four strategies in different publishing rates in the PSD scenario. As the system capacity is limited, the delivery rate certainly decreases with the increase of publishing rate. Figure 6(a) shows that the EB and PC strategies achieve almost the same delivery rate, which is higher than that of the other two strategies. When the publishing rate is 15, the delivery rates in the EB, FIFO and RL strategies are 40.1%, 22.5% and 11.6% respectively. Figure 6(b) shows the number of messages generated by the four strategies. When the publishing rate is 15, the EB strategy incurs just 17% and 60% more network traffics than the FIFO and RL strategies respectively.

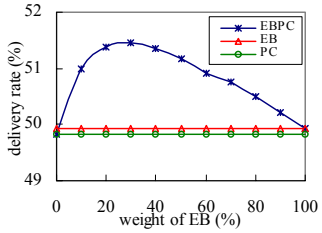
7. Conclusion

Pub/sub systems are very suitable for dynamic information dissemination. However, as dynamic information is usually characterized by the short lifetime, information publishers or subscribers may specify the delay requirement on message delivery. In this paper, we propose an efficient approach for pub/sub systems to achieve bounded delay on message delivery. Three message scheduling strategies are proposed for pub/sub systems to make use of available

bandwidths efficiently. Simulation results shows that our strategies enable subscribers to receive significantly more valid messages than traditional strategies, while incurring slightly more network traffic.

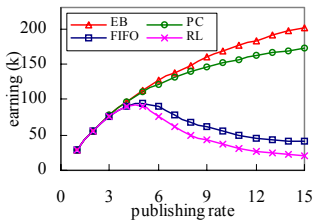


a) Subscriber-specified delay

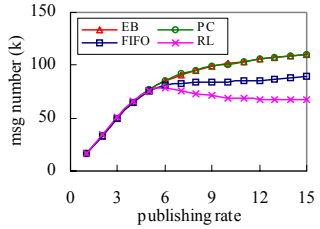


b) Publisher-specified delay

Figure 4. Performance comparison of EB, PC and EBPC strategies

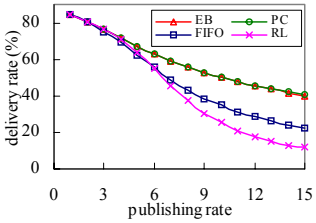


a) Total earning

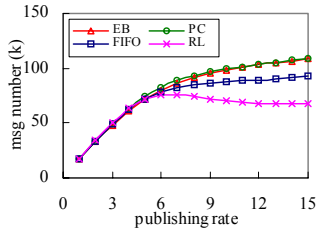


b) Message number

Figure 5. Performance comparison in the SSD scenario



a) Delivery rate



b) Message number

Figure 6. Performance comparison in the PSD scenario

References

[1] Sun Microsystems Inc. JMS Specification version 1.0.2b. Aug. 2001. http://java.sun.com/products/jms

[2] OMG. CORBA Notification Service specification version 1.0.1. Aug. 2002. http://www.omg.org/corba

[3] C. Y. Lee and S. J. Kim. Parallel genetic algorithms for the earliness-tardiness job scheduling problem with general penalty weights. Computers & Industrial Engineering: 28(2): 231-243, 1995.

[4] J. Wroclawski. The use of RSVP with IETF integrated services. IETF RFC 2210, Sep. 1997.

[5] Z. Li and P. Mohapatra. QRON: QoS-Aware Routing in Overlay Networks. IEEE Journal on Selected Areas in Communications, 22(1): 29-40, Jan. 2004.

[6] IEEE 802.11 WG. IEEE Std 802.11e/D4.0, Nov. 2002.

[7] D. Grossman. New terminology and clarifications for DiffServ. IETF RFC 3260, Apr. 2002.

[8] L Subramanian, I Stoica, H Balakrishnan, and R Katz. OverQoS: An Overlay Based Architecture for Enhancing Internet QoS. USENIX NSDI'04, 2004

[9] S. Y. Shi and J. S. Turner. Routing in overlay multicast networks. IEEE Infocom 2002, pp. 1200–1208.

[10] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf: Design and evaluation of a wide-area event notification service. ACM Trans. on Computer Systems. 19(3): 332-383, 2001.

[11] G. Cugola, 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. 27(9): 827-850. 2001.

[12] G. Muhl: Large-Scale Content-Based Publish/Subscribe Systems. PhD thesis. Darmstadt University of Technology, Germany. 2002

[13] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh-Based Content Routing using XML. Proc. of the 18th ACM SOSP, pp. 160-173. Oct. 2001.

[14] S. Bhola, R. Strom, S. Bagchi, Y. Zhao, and J. Auerbach. Exactly-once delivery in a content-based publish-subscribe system. In Proc. of IEEE DSN 2002, pp. 7-16.

[15] R. Chand and P. A. Felber. A Scalable Protocol for Content-Based Routing in Overlay Networks. In Proceedings of 2nd IEEE International Symposium on Network Computing and Applications. pp. 123-130. Apr. 2003.

[16] N. Carvalho, F. Araujo, and L. Rodrigues. Scalable QoS-Based Event Routing in Publish-Subscribe Systems. In Proceedings of 4th IEEE International Symposium on Network Computing and Applications. Jul. 2005.

[17] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, P. Mieghem, and H. Uijterwaal. Analysis of End-to-End Delay Measurements in the Internet. Presentation at RIPE41. Jan. 2002.

[18] A Corlett, D.I. Pullin, and S. Sargood. Statistics of One-Way Internet Packet Delays. Presentation at 53rd IETF. Mar. 2002.