MHH: A Novel Protocol for Mobility Management 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*
[3] *Department of Computer Science and Engineering, Florida Atlantic University*
Corresponding author: Prof. Jiannong Cao, csjcao@comp.polyu.edu.hk

Abstract

Mobility management is an important issue for publish/subscribe systems to support mobile clients. The objectives of mobility management for publish / subscribe are to achieve short handoff delay and low message overhead, while at the same time guaranteeing reliable message delivery. Although mobility management has been extensively studied, the indirect communication style of publish/subscribe systems brings new challenges in designing mobility management solutions. In this paper, we propose a reliable and high-performance mobility management protocol, called multi-hop handoff (MHH) protocol, which sufficiently meets the requirements of publish/subscribe systems. A prototype was implemented for the MHH protocol and experiments were performed to compare the performance of MHH with two representative existing protocols. The experimental results demonstrate the efficiency improvement made by the proposed MHH protocol over the existing protocols.

1. Introduction

A Publish/subscribe (pub/sub) system is a type of message-oriented middleware that supports loosely coupled communication among multiple participants. In pub/sub systems, *publishers* publish information to *event brokers* in the form of *events*, *subscribers* subscribe to a particular category of events within the system, and event brokers ensure the timely and reliable delivery of published events to all interested subscribers. The pub/sub paradigm makes information producers and consumers fully decoupled in time, space and flow [1], so it has been widely applied in many industries such as banking, manufacturing, and transportation.

This work is supported in part by Hong Kong Research Grant Council under the CERG grand PolyU 5103/06E and the Hong Kong Polytechnic University under the ICRG grant G-YE57.

With the increasing popularity of wireless communication networks and mobile handheld devices, mobile computing has been attracting more and more attention in recent years. It has been agreed that the advantages of pub/sub, including loose coupling and asynchronous operations, make it a desirable communication paradigm in the mobile environments [2, 3]. However, most existing pub/sub systems are mainly built on fixed wired networks and need to be extended in order to operate in mobile environments.

There are two main types of mobile network environments: infrastructured wireless networks and ad hoc networks [4]. Pub/sub systems can be implemented for both types of networks. In this paper, we only focus on pub/sub systems in an infrastructured network. In such systems, the event brokers of the pub/sub system are interconnected in a wired network, while the clients can be portable devices carried by mobile users. Event brokers act as the access points for mobile clients; mobile clients can disconnect from one event broker and, after a period of time, reconnect at another event broker.

Mobility management is one of the key issues for any mobile systems, which consists of two tasks: location management and handoff management [5]. In a pub/sub system, when a mobile client moves to a different location and reconnects to the network, mobility management is needed to transfer its subscriptions and the events waiting to be delivered to it to the new location. Although mobility management has been extensively studied in the mobile computing community [5, 6], the indirect communication style of pub/sub systems brings new challenges in designing mobility management solutions:

1) In a pub/sub system, a publisher does not know the addresses of event receivers when it publishes an event. The event is forwarded to the intermediate nodes according to the content of the event and the

subscriptions in the system, and finally reaches all subscribers that are interested in the event. Therefore, the location of a subscriber is implied in the subscription information of a pub/sub system rather than explicitly known to publishers, which makes the traditional location management technologies not applicable to pub/sub systems.

2) Traditional handoff management technologies usually incur the loss or out-of-order delivery of some messages during a handoff process. To achieve reliable message delivery, either the sender or the receiver should detect the loss of messages so that they can be retransmitted, and each message should be labeled with an incremental sequence number so that the receivers can buffer the messages to filter out duplication and re-order them. Unfortunately in a pub/sub system, as a subscriber just selectively receives a part of the published events, the sequence numbers of received messages are inherently incontinuous, so a subscriber cannot judge the loss of an event based on the sequence number. On the other hand, as publishers and subscribers do not know the addresses of each other, the acknowledge mechanisms cannot be used for publishers to detect message loss. As a result, in addition to the conventional objectives such as short handoff delay and low message overhead, the handoff protocol for pub/sub systems should also be reliable enough to avoid the loss of any events and, better, to ensure desirable message delivery semantics and ordering[1].

In this paper, we address the above issues and propose a novel, reliable and high-performance mobility management protocol for pub/sub systems, called *multi-hop handoff* (*MHH*) protocol. The MHH protocol achieves its goals by making use of the *acyclic* network topology and *reverse path forwarding* routing protocol, which are widely used in existing pub/sub systems. In designing MHH, we divide a reconnection process of a mobile client into two tasks: *subscription migration* and *event migration*, which are performed in parallel. Functionally, subscription migration is similar to the location update in traditional mobility management protocols, while event migration is to forward all undelivered events to the new broker during a handoff process. When a client reconnects to a new broker, subscription migration is performed hop-by-hop along the path from the original broker to the new broker. After each hop, the client's subscription is moved to the next broker closer to the destination. In performing event migration, the original broker transfers the stored events for the client to the new broker, and all the brokers on the path collect the in-transit events and send them to the new broker directly. In this way, the MHH protocol can be made to guarantee the exactly-once and ordered delivery of events to mobile clients, and at the same time achieve a very short handoff delay and low message overhead.

We have developed a prototype system to implement the proposed MHH protocol and carried out experiments to evaluate the protocol's performance, in comparison with some representative existing protocols. The experimental results demonstrate the efficiency improvement by the proposed MHH protocol over the existing protocols in terms of message overhead and handoff delay.

The remainder of the paper is organized as follows. In Section 2, we discuss the related work, comparing our protocol with some existing solutions. In Section 3, we describe the system model and assumptions. In Section 4, we introduce the MHH protocol in detail. In Section 5, we describe the experiments for performance evaluation and discuss the evaluation results. Finally, in Section 6, we conclude the paper with a summary.

2. Related Work

Mobility management has been extensively studied in the mobile computing community. In terms of network protocol stack, existing work can be classified into one of four layers: *link layer*, *IP layer*, *transport layer* and *application layer* [6]. Although the IP layer is conceptually the best place to provide the mobility support, the IP-layer mobility mechanisms have not been widely deployed for a number of reasons such as the requirement for infrastructure upgrade and the change of kernel on the mobile hosts. On the other hand, the application-layer mobility protocols can easily overcome the drawback of IP-layer mobility protocol and provides an efficient way for existing applications to support mobile scenarios. Therefore, our work is on the application layer so that the existing pub/sub systems can be easily extended to support mobile clients.

In terms of system design principles, existing mobility management solutions can be divided into two categories: *end-system-centric* and *network-centric*. The end-system-centric solution aims to minimize the change of the network backbone and provide the most function of mobility management on the end systems, the most notable one being Mobile IP [7]. On the other hand, the network-centric solution aims to minimize the changes of the end systems and provide the most function of mobility management on the network backbone; some representative work includes the

[1] We mean *publisher order* of events in this paper, i.e., for two events satisfying the subscription of a client, if they are published by the same publisher, the first published event will first arrive.

mobility management in cellular networks and IEEE 802.16e [8]. In a pub/sub system, as the network backbone is composed of event brokers usually owned by the same organization, the network-centric solution is a natural choice so that the burden and change on mobile hosts can be minimized.

In recent years, a number of mobility management protocols have been proposed for pub/sub systems. One widely-used protocol [9-11] works as follows. When a client is disconnected, the subsequently arrived events are stored in a queue on the client's last visited broker. When the client reconnects to a different broker, the client will re-issue its subscription at the new broker, and at the same time keep its subscription on the original broker for a while. After a pre-defined period, the system ensures that the client's subscription on the new broker is made known to all other brokers, so the system cancels the client's subscription on the original broker, and transfers the undelivered events in the original broker to the new broker. To guarantee the exactly-once and ordered delivery of events to the mobile client, the system needs to create another queue within the new broker to buffer the arrived events for the client during the handoff period, and then merge the events in the two queues, delete the duplicated events, sort them into correct order, and finally deliver them to the client. In the remaining part of this paper, we refer to the protocol as the *sub-unsub* protocol, as it involves a subscribing operation and an unsubscribing operation.

The sub-unsub protocol has two drawbacks: (1) it may take a long time for all brokers to receive the client's newly-issued subscription upon reconnection, especially when the network size is very large. Therefore, the client has to wait for a long time before it can receive any events, resulting in a very long handoff delay. (2) When a mobile client moves frequently, the undelivered events for the mobile client will be frequently moved between different brokers, which may greatly increase the overhead on the network traffic.

Another typical solution is the *home-broker* [9] protocol with a similar idea to Mobile IP. Each client is assigned a *home broker*, which maintains the subscription for the client. When a mobile client connects to a broker other than its home broker, i.e., a *foreign broker*, the registration process is initiated. The foreign broker will register the current location of the client to the home broker, which will forward all events stored for the client to the foreign broker. This protocol can overcome the drawbacks of the sub-unsub protocol, but same as the Mobile IP, it has the following two drawbacks: (1) it is not reliable and may lose some of events destined at the mobile clients. It is possible that the mobile client has moved away from the current foreign broker while an event sent from the home broker to the client is in transit. (2) It suffers from the triangle routing problem because all events are delivered to the mobile client via the home broker, which may significantly increase the network traffic. Although the triangle routing problem in Mobile IP can be overcome by an optimization mechanism, i.e., the corresponding node sends data directly to the new address of the mobile node, the mechanism cannot be directly applied in pub/sub systems as publishers and subscribers do not know the addresses of each other.

We have proposed a *two-phase handoff* protocol [12] which can guarantee the exactly-once and ordered delivery of events to mobile clients with a low cost. However, there may be conflicts among the concurrent handoff processes executing the protocol and, consequently, some events may be delayed in being transferred to their destinations. In contrast, the handoff process of a client in the MHH protocol does not affect the event delivery of other clients, so the MHH protocol can naturally support the concurrent moving of clients without any performance degradation.

In addition to the aforementioned protocols, there are also several other mobility management protocols for pub/sub systems [13-15], which are more or less similar to the sub-unsub or home-broker protocol.

3. System Model

Publish/subscribe systems can be generally divided into two categories: *subject-based* and *content-based*. In *subject-based* systems, each event belongs to one of a fixed set of *subjects*. Publishers are required to label each event with a subject name; subscribers subscribe to all events under a particular subject. On the other hand, in *content-based* systems, each subscriber defines a *filter* according to the internal structure of events; all events that satisfy the filter will be sent to the subscriber. Compared with subject-based pub/sub systems, content-based systems are more expressive and flexible because they enable subscribers to express their interests with a finer level of granularity. Therefore, recent research on pub/sub is mainly focused on content-based systems [16-18]. The work described in this paper is based on the content-based system model.

In our system model, there are multiple event brokers each serving a certain number of clients (publishers or subscribers). The clients can move around while the event brokers are interconnected in a wired network. The event brokers are organized into an overlay that forwards events from publisher to all interested subscribers.

The design of the proposed protocol is based on the *acyclic* event broker network and the *reverse path forwarding* based routing protocol. To simplify the routing strategy, the mainstream content-based pub/sub systems (such as SIENA [16], JEDI [13], Rebeca [17], etc.) organize the overlay into an acyclic structure, as shown in Figure 1. In the figure, each circle node (labeled with B_i) represents an event broker, and each rectangle node (labeled with C_i) represents a client.

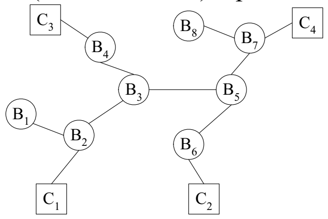


Figure 1. Acyclic structure of a pub/sub system

The widely-used routing protocol in a content-based pub/sub system is the reverse path forwarding protocol. Each broker knows in advance of a spanning tree rooted at it. When a broker receives a subscription from its clients, it forwards the subscription to other brokers through its spanning tree. When a broker receives a published event from its clients, it forwards the event to other brokers through the reverse path of the subscription messages.

Each event broker maintains a filter table to record the subscriptions of its neighbors. The neighbors of a broker include both the neighboring brokers and the clients that directly connect to the broker. The filter table of a broker can be represented as the set $\{(nb, f)\}$, where each pair means that neighbor *nb* is interested in the events that satisfy the filter *f*.

We require that each broker maintains a routing table for the broker overlay network, which records the next hops for any other broker in the network. The routing table of a broker can be represented as the set $\{(nb, destination)\}$, where each pair means that the current broker can reach the *destination* via neighbor *nb* in the overlay network.

Like most previous work on content-based routing protocols [13, 16, 17], our work is based on the assumptions that there are no failures on event brokers and on the links between them, and the message delivery on each link is FIFO ordered. There is also some existing work addressing the issues on how to achieve fault-tolerance in pub/sub systems when the above assumptions are relaxed. These issues are out of the scope of this paper, and will not be further discussed.

4. MHH: Multi-Hop Handoff Protocol

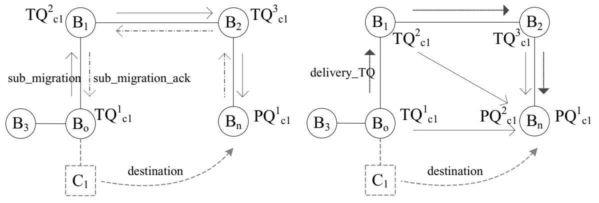
Similar to the sub-unsub protocol, in the MHH protocol, when a client disconnects from the system, the subsequently arrived events will be stored on the client's last visited broker. The following two types of queues are defined to store events: 1) *Persistent Queue* (*PQ*): to store potentially large number of events for a considerably long period; 2) *Temporary Queue* (*TQ*): to temporarily store events during the handoff period.

We distinguish two different types of client mobility: 1) *Proclaimed move*: the client informs the system of its destination before it disconnects; 2) *Silent move*: the client disconnects without giving any notice.

While the proclaimed move is uncommon in the real world, it provides an easy basis for us to start with. In this section, we first describe a basic MHH protocol which can support the proclaimed move of clients. Then we extend the basic protocol to support the silent move of clients. To efficiently handle frequent moving of mobile clients, the protocol is further extended by maintaining a distributed linked list for each disconnected client.

4.1. Proclaimed move

In the proclaimed move case, when a client specifies its destination and disconnects from the current broker, the subscription migration is performed hop-by-hop along the path from the current broker to the new broker. After each hop, the client is considered to become an offline client of the next broker closer to the new broker. At the same time, all the brokers on the path will collect the in-transit events and send them to the new broker directly.



a) subscription migration b) event migration

Figure 2. Proclaimed move of mobile clients

Suppose a client C_1 has defined a subscription with filter f_1, and the current broker of C_1 is B_o. Before C_1 disconnects, it tells B_o that its destination is B_n. Let B_o reach B_n via the neighbor B_1, as shown in Figure 2. B_o will act as follows upon the disconnection of C_1:

1) Add an entry (B_1, f_1) in its filter table, which means that B_1 is now interested in the events satisfying filter f_1.
2) Mark the entry (C_1, f_1) in its filter table with a label "B_1", which means that B_o now only accepts events coming from B_1 for C_1.
3) Send a *sub_migration* message to B_1 to notify the migration of C_1. The *sub_migration* message contains the following information: i) the identifier of the mobile client; ii) the filter defined by the mobile client; iii) the broker that the client will

move to.

After B_o sends out the *sub_migration* message, the incoming events from B_1 for client C_1 will be stored in a temporary queue TQ^1_{c1}. For events coming from other neighbors and satisfying filter f_1, B_o will send them to B_1 as there is already an entry (B_1, f_1) in its filter table.

For a broker B_i on the path from B_o to B_n, suppose it reach B_o via the neighbor B_{i-1} and reach B_n via the neighbor B_{i+1}. B_i will acts as follows upon receiving the *sub_migration* message:

1) Add an entry (B_{i+1}, f_1) in its filter table, which means that B_{i+1} is now interested in the events that satisfy filter f_1.
2) Delete the entry (B_{i-1}, f_1) in its filter table as the mobile client has moved away from B_{i-1}.
3) Regard C_1 as a local offline client and add an entry (C_1, f_1) in its filter table with a label "B_{i+1}", meaning that C_1 will only accept events coming from B_{i+1}. The subsequently arrived events for C_1 will be stored in a temporary queue.
4) Send back a *sub_migration_ack* message to B_{i-1}, which is used to push the in-transit events on the link from B_i to B_{i-1} into B_{i-1}.
5) Send a *sub_migration* message to B_{i+1} to continue the subscription migration process.

As the message delivery on each link is FIFO ordered, when B_{i-1} receives the *sub_migration_ack* message, all events sent out by B_i before the *sub_migration_ack* message have already arrived at B_{i-1}. Then B_{i-1} deletes the entry (C_1, f_1) from its filter table, i.e., it will not accept new events for client C_1.

After several steps, the destination broker B_n will receive the *sub_migration* message and the subscription migration process finishes. The subscription information in the system is successfully updated to reflect the change of the location of the mobile client. As C_1 may reconnect at B_n after a considerably long time, B_n will create a persistent queue PQ^1_{c1} to store the subsequently arrived events for C_1.

To simplify description, here we did not discuss the case when multiple clients define the same filter. To handle such a case, the *sub_migration* message should contain a value to indicate whether the sender will cancel the filter of the mobile client. Furthermore, if the *covering* operation [16] is used in the system, the *sub_migration* message should also contain the covered filters of the canceled filter.

Now we describe the event migration process in the MHH protocol. When the subscription migration finishes, all in-transit events during the handoff period have been stored in the temporary queues on the brokers along the path from B_o to B_n, so we just need to send the events in these queues to B_n. When B_o receives the *sub_migration_ack* message, it begins to send events in TQ^1_{c1} to B_n as it will not add any new events in the queue. To guarantee the ordering of events, B_n should create another persistent queue PQ^2_{c1} to store the events coming from the temporary queues.

When B_o finishes transferring the events in TQ^1_{c1}, it will delete TQ^1_{c1} and send out a *deliver_TQ* message to the next hop B_1, asking B_1 to send events in TQ^2_{c1} to B_n. Following the above process, B_n will receive a *deliver_TQ* message after a while. At that time, the events in all *TQ*s have arrived at B_n, so the event migration process finishes.

When C_1 reconnects to the system at B_n, B_n will first deliver the events in PQ^2_{c1} to C_1, then deliver the events in PQ^1_{c1}, and finally delete the two queues.

Now we briefly outline the correctness proof of the protocol. First we explain the correctness of the subscription migration. Let the path from B_o to B_n be p. As the broker overlay is in an acyclic structure, any node outside the path p will reach B_o and B_n via the same neighbor, so the filter tables of these nodes do not need to be changed when a client moves from B_o to B_n. For node B_i in path p, suppose it reaches B_o via neighbor B_{i-1} and reaches B_n via neighbor B_{i+1}. When the client moves from B_o to B_n, B_i will just need to update the filter items for B_{i-1} and B_{i+1} in its filter table accordingly.

Now we explain the correctness of the event migration. In the case where there is just one hop from B_o to B_n, the exactly-once and ordered event delivery can be achieved as the message delivery on each link is FIFO ordered. When there are multiple hops from B_o to B_n, the protocol is also correct as the whole handoff process can be considered as a sequence of one-hop handoff processes.

4.2. Silent move

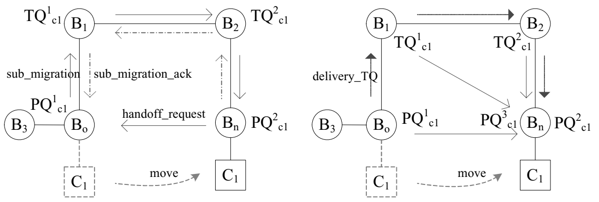
In the silent move case, we require that each client maintains the identifier of its last-visited broker during the disconnection period. When a client reconnects at a broker other than its original broker, the hand process will be triggered.

Suppose a client C_1 has defined a subscription with filter f_1 and its current connected broker is B_o. Upon detecting the disconnection of C_1, B_o will create a persistent queue PQ^1_{c1} to store the undelivered events for C_1.

Suppose C_1 reconnects at B_n after a while, as shown in Figure 3. C_1 will provide the identifier of B_o and then B_n will send a *handoff_request* message to B_o to begin the handoff process. The *handoff_request* message contains the identifier of the mobile client and the identifier of the broker that the client now connects.

When B_o receives the *handoff_request* message, it will perform the handoff in a way similar to the proclaimed move case. The difference is that, B_o does not need to create a temporary queue TQ^1_{c1} in this case;

it just appends the in-transit events for C_1 from B_1 to the end of PQ^1_{c1}.



a) subscription migration b) event migration

Figure 3. Silent move of mobile clients

When B_n receives the *sub_migration* message, it will create a persistent queue PQ^2_{c1} to store the newly arrived events for C_1 rather than deliver them to the client directly, as the event migration may not have finished yet.

When B_o receives the *sub_migration_ack* message sent by B_1, it will begin to deliver the events in PQ^1_{c1} to B_n. When B_n receives these immigrant events from other queues, it will begin to deliver them to C_1. As the transmission of events from B_n to C_1 is in a wireless network which may be slower than that in the wired broker network, B_n will create another persistent queue PQ^3_{c1} to buffer the immigrant events.

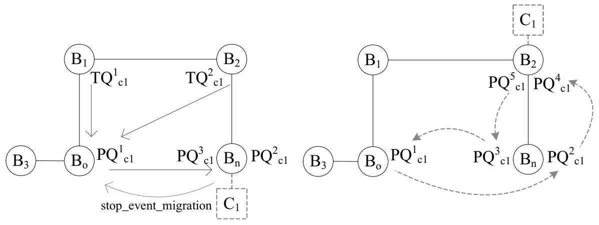
From the above discussion we can see that when a mobile client reconnects to a new broker, it can get the undelivered events just after a round of message exchange between the original broker and the new broker, which takes a very short time. On the other hand, in the sub-unsub protocol, the client has to wait for the finish of the whole handoff process before it can receive any events.

4.3. Processing of Frequent Moving

In some cases a mobile client may move very frequently; before a handoff process finishes, the client may disconnect and after a while reconnect at another broker. To avoid the frequent moving of undelivered events between different brokers, the basic MHH protocol is further extended. As the silent move case is more prevalent, we just consider the case of frequent silent moving in this section. The frequent proclaimed moving can be processed in a similar way.

In our solution, once a client disconnects before the handoff finishes, the event migration from the original broker to the new broker will stop immediately, as we don't know the destination at which the client will reconnect. The system maintains a distributed linked list (called *PQlist*) for each disconnected client on the brokers that have been visited by the client. The elements of the distributed linked list are *PQs* that store the undelivered events for the clients. When the client reconnects, the system will deliver the events in the *PQlist* to the client according to their order in the *PQlist*.

Suppose C_1 disconnects before the handoff finishes in the example shown in Figure 3. At the time of C_1's disconnection, if all events in PQ^1_{c1} have already been moved to B_n, then the events in the *TQ*s will continue to be moved to B_n as there are usually very few events in the *TQ*s. However, if the migration of events in PQ^1_{c1} has not finished yet, B_n will send out a *stop_event_migration* message to B_o, asking B_o to stop the event migration, as shown in Figure 4(a). When B_o receives the message, it will send a *deliver_TQ* message to B_1, asking the brokers on the path from B_o to B_n to deliver the events in *TQ*s to B_o rather than B_n. In such a way, when the migration of events in the *TQ*s finishes, the *PQ*s for C_1 form a distributed linked list, i.e., $PQ^3_{c1} \rightarrow PQ^1_{c1} \rightarrow PQ^2_{c1}$. When C_1 reconnects, the systems will deliver events in *PQlist* to C_1 according to their order in the list.



a) C_1 disconnects from B_n b) C_1 reconnects and disconnects

Figure 4. Frequent moving of mobile clients

Now suppose C_1 reconnects at B_2 after disconnected from B_n. B_2 will create two persistent queues PQ^4_{c1} and PQ^5_{c1} for C_1 as described in Section 4.2. The queue PQ^4_{c1} is used to store the newly arrived events after the subscription migration finishes, and the queue PQ^5_{c1} is used to store the immigrant events from the other queues. If the client disconnects again during the event migration from PQ^3_{c1} to PQ^5_{c1}, the *PQlist* of C_1 will become $PQ^5_{c1} \rightarrow PQ^3_{c1} \rightarrow PQ^1_{c1} \rightarrow PQ^2_{c1} \rightarrow PQ^4_{c1}$, as shown in Figure 4(b).

It should be noted that a number of cases need to be considered to successfully maintain the *PQlist*, as the mobile client may disconnect at various stages of a handoff process. Due to space limitation, we do not investigate this issue further.

5. Performance Evaluation

We have implemented the MHH protocol on a prototype pub/sub system and evaluated the performance of the protocol with a simulated network and various workload conditions. In this section, we describe our experimental study and discuss the performance evaluation results.

5.1. Experiment Setup

In the experiments, we simulated a wireless network with k^2 base stations organized into cells and a number of mobile clients connected to the base stations with wireless links. The base stations are organized into k rows with each row containing k stations. Each base station directly connects to its neighboring stations with wired links. Any pair of stations can connect with each other via the shortest path in the network. We assume that the message delivery time over each wired link is 10ms and the message delivery time over each wireless link is 20ms.

In the pub/sub system built over the above network, each base station acts as an event broker and a minimum cost spanning tree of the network is built to serve as the acyclic overlay.

In the initial state, each broker serves 10 clients, so there are a total of $10 \times k^2$ clients in the system. We randomly choose 20% of the clients in the network to let them move in the network. As the silent move model is more common in the real world than the proclaimed move model, we just simulate the silent move of mobile clients. The mobility pattern of each mobile client is as follows. Each mobile client disconnects and reconnects from time to time, and the location of each time of connection is randomly chosen from all base stations. The lengths of connection periods and disconnection periods for mobile clients are random variables that satisfy the exponential distribution.

Each client in the system has defined a subscription and each client publishes events continuously with the rate of one event every 5 minutes. We generate events and subscriptions carefully so that for each published event, there are on average 6.25% clients in the system interested in it.

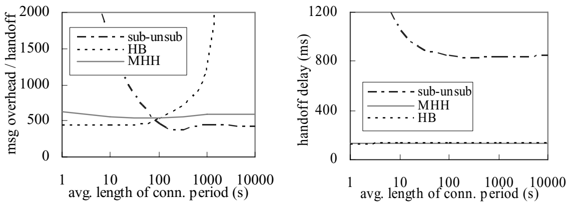
We also implemented the sub-unsub protocol and the home-broker protocol to compare the performance of the three protocols under the same environment and workloads. To avoid the loss of events in the sub-unsub protocol, the interval between the subscribing operation and the unsubscribing operation in each handoff is set to be the maximum time for message delivery between any two stations in the network. Note that while the MHH protocol and sub-unsub protocol can both provide reliable event service, the home-broker protocol is not reliable and may incur the loss of some events during a handoff process.

The following metrics are used to evaluate the performance of the three protocols:

- *Message overhead per handoff*: the total overhead on the network traffic caused by mobile clients divided by the number of handoff processes. Network traffic is measured as the total hops that all messages traveled in the network.
- *Average handoff delay* of all handoff processes for mobile clients. We call the period from a client's reconnection time to the time it receives the first event as the *handoff delay*.

5.2. Experiment Results

Figure 5 compares the performance of the three protocols with different lengths of client connection periods. The curve *HB* stands for the home-broker protocol. In this group of experiments, there are 100 base stations in the system and the average length of disconnection periods for mobile clients is 5 minutes. The average length of connection periods for mobile clients increases from 1 second to 10,000 seconds, which reflects the different moving frequency of mobile clients.



a) msg overhead vs. conn. Period b) handoff delay vs. conn. period

Figure 5. Varying the length of conn. periods

Figure 5(a) shows the message overhead per handoff of the three protocols. The message overhead of the sub-unsub protocol increases sharply when the average length of connection period is lower than 100 seconds, because the system has to frequently move the bulk of undelivered events between different brokers when the mobile clients disconnect during the handoff process. The message overhead per handoff of the home-broker protocol increases sharply when the average length of connection periods is greater than 100 seconds, because all events are delivered to the mobile client via the home broker and the total message overhead is mainly determined by the number of events subscribed by mobile clients and has little relation with the number of handoff process. When the length of a connection period increases, the number of handoff processes decreases correspondingly, resulting in the linear increase of the message overhead per handoff process[2].

In comparison with the other two protocols, the MHH protocol always incurs a low overhead on network traffic, because it combines the subscription migration with the distributed linked lists to overcome the problems of the other two protocols.

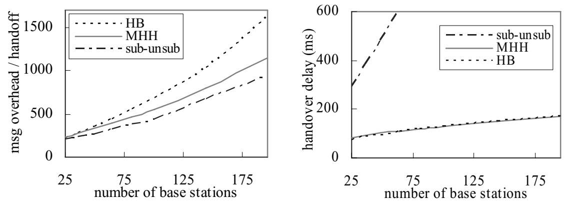
Figure 5(b) shows the average handoff delay of the three protocols. The average handoff delay of the sub-unsub protocol is much higher than the other two protocols, as a mobile client has to wait for the finish of the handoff process before receiving any events. The average handoff delay of the MHH protocol and the home-broker protocol are almost the same and, as they are mainly determined by the length of path from the original broker to the new broker, they do not change with the average length of connection period.

Figure 6 compares the performance of the three protocols with different network sizes. In this group of experiments, the number of base stations in the system increases from 25 to 196, and the average lengths of both connection periods and disconnection periods for mobile clients are 5 minutes.

Figure 6(a) shows the message overhead per handoff of the three protocols. Although the message overhead under the three protocols all increase with the network size, the margins between them increases

[2] The *HB* curve in Figure 5(a) is exponential because the scale of X-axis is exponential.

when the network size scales up. The message overhead under the home-broker protocol increases faster than that of other two protocols because the triangle routing problem becomes more serious with the increase of the network size. On the other hand, the message overhead under the sub-unsub protocol increases slower because the number of subscriptions also increases with the increase of network size, so a subscription is more likely to be covered by other subscriptions, resulting in less message overhead for subscribing and unsubscribing operations.



a) msg overhead vs. network size b) handoff delay vs. network size

Figure 6. Varying the network size

Figure 6(b) shows the average handoff delay of the three protocols. As the handoff delay of the MHH and home-broker protocol are mainly determined by the length of path from the original broker to the new broker, the average handoff delay is mainly determined by the average distance between any pair of nodes. On the other hand, in the sub-unsub protocol the system has to ensure that all brokers receive the mobile client's newly-issued subscription before a handoff finishes, so the handoff delay is mainly determined by the maximum distance between any pair of nodes, which is much larger than the average distance. Therefore, the sub-unsub protocol incurs much longer handoff delay than the other two protocols.

From the above experimental results we can see that, comparing with the sub-unsub protocol, the MHH protocol can always achieve much shorter handoff delay, and it also incurs much less message overhead when clients are moving frequently. Compared with the home-broker protocol, the MHH protocol achieves the same handoff delay, but it incurs much less message overhead when clients are moving less frequently, and it is also more scalable when the network size increases. What's more, the home-broker protocol is not reliable and may incur the loss of some events during a handoff process, while the MHH protocol can guarantee the exactly-once and ordered event delivery.

6. Conclusion

In this paper, we proposed MHH, a novel mobility management protocol for pub/sub systems. MHH is based on the acyclic event broker overlay and the reverse path forwarding routing protocol widely used in existing pub/sub systems. It can guarantee the exactly-once and ordered delivery of events to mobile clients while achieving a low handoff delay. It incurs very low message overhead regardless of the moving frequency of clients. A prototype system is built to implement the MHH protocol, and the experimental results demonstrate its performance improvement over the representative existing protocols.

References

[1] P. Th. Eugster, P. A. Felber, R. Guerraoui, A.-M. Kermarrec. The many faces of publish/subscribe. ACM Computing Surveys, 35(2): 114 – 131, 2003.

[2] G. Cugola and H.-A. Jacobsen: Using publish / subscribe middleware for mobile systems. ACM SIGMOBILE Mobile Computing and Communications Review, 6(4):25-33, 2002.

[3] Y. Huang and H. Garcia-Molina: Publish/Subscribe in a Mobile Environment. In Proc. of 2nd ACM Intl. Workshop Data Engineering for Wireless and Mobile Access, pp. 27-34, 2001.

[4] E. M. Royer and C.-K. Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless networks. IEEE Personal Communications. Apr. 1999.

[5] I. F. Akyildiz, J. Xie and S. Mohanty. A Survey of Mobility Management in Next-Generation All-IP-Based Wireless Systems. IEEE Wireless Communications, pp. 16-28. Aug. 2004

[6] N. Banerjee, W. Wu, S. K. Das, S. Dawkins and J. Pathak. Mobility support in wireless Internet. IEEE Wireless Communications, 10(5): 54-61, 2003.

[7] C. Perkins. IP Mobility Support for IPv4, Revised. Technical Report RFC 3220, IETF Network Working Group. Jan. 2002.

[8] IEEE 802.16e Task Group. IEEE Std 802.16e-2005. Feb. 2006.

[9] I. Burcea, H.-A. Jacobsen, E. DeLara, V. Muthusam, and M. Petrovic. Disconnected Operation in Publish / Subscribe Middleware. In Proc. of IEEE Intl. Conf. on Mobile Data Management, pp. 39-51. 2004.

[10] U. Farooq, E. W. Parsons, and S. Majumdar. Performance of Publish/Subscribe Middleware in Mobile Wireless Networks. In Proc. of 4th Intl. Workshop on Software and performance, Jan. 2004

[11] M. Caporuscio, A. Carzaniga, and A. L. Wolf. Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications. IEEE Trans. on Software Engineering 29(12): 1059-1071, 2003.

[12] J. Wang, J. Cao, J. Li. Supporting Mobile Clients in Publish/Subscribe Systems. In Proc. of 25th Intl. Conf. on Distributed Computing Systems Workshops, pp. 792-798. Jun. 2005.

[13] 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.

[14] L. Fiege, F. C. Gartner, O. Kasten, and A. Zeidler. Supporting mobility in content-based publish/subscribe middleware. In: Proc. of 4th ACM/IFIP/USENIX Intl. Middleware Conference, pp. 103–122. 2003.

[15] I. Podnar and I. Lovrek. Supporting Mobility with Persistent Notifications in Publish/Subscribe Systems. In Proc. of 3rd Intl. Workshop on Distributed Event-Based Systems. May 2004.

[16] 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.

[17] G. Muhl: Large-Scale Content-Based Publish/Subscribe Systems. PhD thesis. Darmstadt Univ. of Technology. 2002

[18] J. Wang, B. Jin, and J. Li. An Ontology-based Publish/Subscribe System. In Proc. of 5th ACM/IFIP/USENIX Intl. Middleware Conference, pp. 232-253. Oct. 2004.