Chapter 3

The Architectures of Event Notification Service System in Mobile Computing Environment

This chapter identifies requirements for event notification systems and concentrates on the aspect of how to conceptually model an event notification service system for mobile computing environment. Designs of event notification service system are different on wireless network with base stations and ad-hoc network (without base station) as the characteristics of these environments are different. A conceptual architecture of wireless event notification service system is presented on the wireless network with base stations which is widely used in practice (e.g., GSM cellular network).

3.1 Introduction

In this chapter, we present an overview of an architecture of event notification service (ENS) system for mobile computing environment. In the process, we discuss what functionalities event notification service system should have and how these can be realized by different architectural components of event notification service system. The architecture has a significant impact on functionality and scalability of a system. For example, with a centralized event notification service, it is relatively easy to implement complex filtering of notifications, but it is evidently difficult to obtain a scalable service. The main problems of the hierarchical topology are the overloading of higher-level servers. A failure in one server disconnects all the subnets reachable from its parent server and all the client subnets from each other.

The routing of notifications is controlled by subscriptions/unsubscriptions and the routing algorithms must take care of disseminating notifications too. Different algorithms adopt different strategies and require more or less complex data structures and
computations on event servers. Event notification service system has been recognized as a suitable application catering services to mobile users [18, 49]. But, movement of clients causes issues such as losses and duplications of events and re-routing of events to new locations of clients. We realize the importance of architecture and denote this chapter for its study.

In the following, the related work is described in Section 3.2. The requirements for event notification systems are identified in Section 3.3. Section 3.4 presents possible designs of event notification service systems for different mobile/wireless networks and a basic conceptual architecture for mobile network is presented in Section 3.5. The services for some of these requirements are designed in the next chapters by extending this basic architecture of event notification service system. Finally, we give a summary of the chapter.

### 3.2 Related Work

Several event notification service systems for the mobility of clients have been proposed in SIENA [10, 11], JEDI [18], Elvin [79] and REBECA [27, 87] as described in Chapter 2. Generally, in these ENS systems, during the time of disconnection notified events are stored at the event server at which the user registered last. On reconnection, the stored events are forwarded to the user's new location. Mobile Push [64] also proposes a similar approach for content dissemination service supporting mobile users. Huang and Garcia-Molina [34, 35] emphasized the problem of supporting mobility in event notification service systems by describing issues that make an event notification service system adaptable to dynamic changes in mobile and ad-hoc networks. Cugola [19] gave preliminary analysis of some issues in designing event notification service by identifying the requirements of underlying wireless communication infrastructure.

A technique for achieving timeliness and reliability for real-time event-based communication in ad-hoc wireless networks has been proposed by Hughes and Cahill [36]. Their conceptual model is the first to directly address the issue of achieving timeliness and reliability in dynamic networks.
3.3 Requirements of Event Notification Service Systems

This section describes the requirements of event notification service systems and it consists of two parts: Typical requirements for every event notification service and additional requirements which depend on different application domains and different environments.

3.3.1 Typical Requirements

Typical requirements of an event notification service system for any applications and environments are as follows:

1. Delivery of an event to related consumers only: Notifications should be delivered to consumers that match one of their subscriptions.

2. Delivery of an event exactly once to a consumer: Each notification should be delivered once to a consumer to avoid duplication of notifications.

3. Delivery of events without losses: All notifications matching one subscription of a consumer should be delivered to the consumer. A consumer should receive his/her interested event if the event was published.

4. Delivery of events in the same order: Notifications should be delivered in the same order with respect to their publications. Out-of-order event streams can be a problem if the order of events is significant, for example to establish a trend in the movements of a stock's price [34]. If event A was generated before event B, A must be delivered (to all consumers) before B. Some ENS systems may place strict ordering only for events published/generated by a producer/source. But some can tolerate disorder of events generated by different sources.

5. Fault-tolerant mechanism: Isolated network or component failures (e.g., failures of event servers and link between them) should not affect the entire system. It is necessary to minimize the impact of faults occurring in an event notification service system.

6. Scalability: A scalable distributed system is one that can easily cope with the addition of users and sites, and whose growth involves minimal expense,
performance degradation, and administrative complexity [51]. Scalability is a crucial requirement for Internet-scale distributed applications.

### 3.3.2 Additional Requirements

Different event notification service systems may have the following requirements depending on application domains and environments.

1. **Timely delivery of events**: Many applications such as stock trading system, auction system, reservation system, sports reporting and traffic condition monitoring systems need to manage deadline bound data items. A user may be willing to receive sports and news information with delay by a few minutes. But it may not tolerate delays in quotes of stock prices.

2. **Support for mobility and disconnection of clients**: The ENS system must have mobility support for applications of mobile users in both fixed and wireless networks.

3. **Expressiveness (specifying events and subscriptions)**: Some applications such as stock trade and auction system must provide powerful data model to specify events and subscriptions. For applications using location-dependent information, the service needs to specify location-related subscriptions and events.

4. **Security requirements**: The levels of security requirement for different applications differ to each other. For example, security requirements of e-business and e-commerce applications are higher than weather/traffic report.

### 3.4 Mobility in Event Notification Service System

Mobility poses new challenges that have not previously been addressed. Client's movement introduces some problems. For example, clients may lose some events while they are disconnected. Mobility entails the study of systems in which components change location, in a voluntary or involuntary manner, and move across a space that may be defined to be either physical or logical.
Host/physical mobility is similar to what in the area of mobile computing is called terminal mobility or roaming. It allows a terminal to be identified by a unique terminal identifier independent of its point of attachment to the network. Hosts move while the virtual names remain fixed. For example, mobile phones are moving while the phone numbers remain the same; a web server is accessible through the same domain name. This kind of mobility is managed at the network layer and, therefore the movement of the host is completely transparent at the application layer. This chapter focuses primarily on host mobility that mobile users move with their mobile devices.

Code/logical mobility involves the code and data movement among hosts. It can redefine bindings between the software components and the network hosts when they move from a node to another in network (for example, mobile agent). The ability to relocate code is a powerful concept that is a wide-spread interest.

### 3.4.1 Mobile Computing Environment

The design of distributed algorithms and protocols has traditionally been based on an underlying network architecture consisting of static hosts, i.e., the location of a host within the network does not change [5]. In traditional distributed computing environment, different forms of mobility exist. RPC (Remote Producer Call) and RMI (Remote Method
Invocation) mechanisms are forms of control mobility where a thread of control moves from one (static) machine to another.

Mobile devices consist of portable computing devices, such as notebook computers and handheld devices. A mobile device being capable of wireless networking allows its application components to interact with components hosted by other mobile devices through wireless communication while moving in a mobile computing environment. Because of fundamental limitations of power and form factor, handheld mobile devices have low computing power (CPU speed and memory), restricted power consumption, short battery life, small displays and different limited input devices (e.g., a phone keypad, voice input, etc.). Similarly, wireless data networks present a more constrained communication environment compared to wired networks. Because of fundamental limitations of power, available frequency spectrum, and mobility, wireless data networks tend to have low bandwidth, more latency and low connection stability.

Characteristics of mobile computing environment differ from distributed computing environment due to above various limitations of mobile devices and wireless data network. The mobile computing applications should have the capability to respond to the changes of its environment and resource requirements. Mobile computing environments can use either the infrastructure or the ad-hoc network model for wireless communication [16]. Access points may be connected to a fixed network, such as Intranet or Internet, and act as portals allowing the components under their control to connect to the fixed network. Mobile computing requires integration of handheld mobile devices with existing data network. A mobile computing system is described in Figure 3.1. A host that can move while retaining its network connections is a mobile host (MH). A mobile host (MH) can connect to the network from different locations at different times. The infrastructure machines that communicate directly with the mobile hosts are called mobile support stations (MSS). The geographical area within which a MSS support mobile hosts is called a cell. All MHs that have identified themselves with a particular MSS are considered to be local to the MSS. Each MSS maintains a list of identities of local MHs.

On receipt of leave ( ) from a local MH, it is deleted from the list. MSS adds the MH in the list of local MHs when the MH sends register (id_MH). When a MH connects to a new MSS, the handoff procedure is executed between the previous MSS and new MSS.
When a MH leaves a MSS, it will eventually show up in some MSS. The MH may not always be able to supply the identity of its previous MSS with the `reconnect` message; in that case, the new MSS may have to query each MSS to determine the previous location of the MH.

Ad-hoc networks are self-organizing wireless networks (Figure 3.2) composed of mobile nodes and requiring no fixed infrastructure [57]. It allows application components to communicate with each other without the aid of access points or a fixed network. Rather, a peer collection of stations within range of listening to each other may dynamically configure themselves into a temporary wireless network [75]. For example, mobile devices can communicate to one another when one comes into range of another as it happens in Bluetooth technology. Ad-hoc networks are extremely useful in scenarios where a natural disaster has wiped out the infrastructure or where rapid deployment is required and an infrastructure is not possible (for example, in the battlefield). Ad-hoc networks can also play a role in civilian forums such as electronic classroom, convention centers, and construction sites [57].

### 3.4.2 Event Notification Service Systems and Mobile Computing Environment

A good event notification service system has to deal gracefully with both the producer’s and consumer’s offline [34]. After a user is disconnected or out of reach, event notification
service system needs to queue the user's events so that they can be delivered later when the user comes back online. In an event notification service system for mobile computing environment, producers and consumers can reside on mobile devices. Event servers may reside on fixed host as they require a fair amount of computing resources. For example, filtering of events may need to check the events against a large amount of subscriptions stored in routing table. But in event notification service for ad-hoc network, event servers must be installed on mobile devices. Hence applications which require fast and large computing power cannot be used in ad-hoc network. The architectures of event notification service systems for mobile environment can be designed as follows.

1. Wireless network with base stations (e.g., cellular network): When a mobile consumer moves to a new location served by a new event server, subscriptions of the mobile consumer need to be re-subscribed for new paths of notifications. In this case, a consumer can carry its subscriptions and re-subscribes them to new event server. It causes an advantage that the consumer can receive new events even if the old event server is temporarily down. But it needs storage overhead of subscriptions at mobile users and usage of precious bandwidth required for re-subscribing these subscriptions. Hence the event server of previous location
transfers subscriptions of mobile consumer to the event server of new location and the new event server re-subscribe these subscriptions on behalf of mobile consumer. The event notification service systems for wireless network with base station can be implemented as follows.

(a) Extending on an existing ENS system as described in Figure 3.3. This architecture is also used in [10]. As base stations of wireless network do not have functionalities of ENS, some base stations are necessary to connect to some event servers of existing ENS system. Hence, event servers need to be extended their functions for mobile users. These extended functionalities consist of storing subscriptions and notifications of mobile users and transferring these subscriptions and notifications to new event servers of mobile users.

(b) Adapting functionality of ENS system to the mobile environment (in Figure 3.4). This architecture can operate itself without the support of fixed ENS system as mobile support stations (MSS) have functionalities of event servers. In many mobile applications such as information on traffic jams, traffic maps and free parking spaces, consumers who are interested events are mobile users. If producers publish these events to wireless network, subscriptions of mobile users are not necessary to forward to event servers of fixed network of ENS system. Hence, this architecture of ENS system should
be designed for applications which provide services to mobile users. The details of components and their functions are described in Section 3.5.

(2) Ad-hoc network (e.g., Bluetooth): The time-varying capacity of wireless links, limited resources and node mobility make maintaining accurate routing information very difficult in ad-hoc wireless networks [36]. Routing protocols traditionally rely on flooding of queries to discover a destination and they are based on a source initiated query/reply process. In event notification service, the routing of an event is only based on subscriptions and the event is delivered to all consumers who issued matched subscriptions. In [19], the designs of event notification service systems for ad-hoc network are described as follows.

(a) Event servers are installed on a set of MHs. Consumers must store and carry their subscriptions and periodically refresh their subscriptions to connected event servers. If the event server is new, the consumers re-subscribe automatically their subscriptions. Otherwise, the old event server could go out of reach and a new event server takes over without knowing about the consumer's subscriptions. It needs to enable dynamic reconfiguration of the system whenever an event server is not anymore reachable through wireless communication.

(b) Each MH acts as a producer/consumer and the service does not rely on event server topology. Event notifications may be filtered at both the producer and the consumer side. All hosts need to cooperate to deliver events from producers to interested consumers. Hence alternative dissemination techniques of events are required in case of lack of event servers.

3.5 Architecture of Wireless Event Notification Service System

This section focuses on event notification service system in wireless network with base station described in Section 3.4.2. Figure 3.5 shows a basic conceptual architecture of an event server of event notification service system for mobile computing environment. Each event server consists of several components that are linked together to provide high
performance event notification service. In the following sections, the details of these components are described.

Event servers are interconnected with each other to establish distributed event notification service system as shown in Figure 3.4. For simplicity, it is assumed that every MSS is installed with functionalities of event servers although in practice, only some MSSs can be used as event servers. It follows *subscription forwarding* routing strategy as described in Chapter 2.

### 3.5.1 Subscription, Notification and Routing Table

Each event server maintains a routing table (RT) for propagation of subscriptions and notifications. Routing table (RT) is a collection of tuples \(<\text{subscription}, \ <\text{consumers}, \ <\text{incoming servers}, \ <\text{outgoing servers}>; \text{i.e.} \ RT := \{ <S, CID, IS, OS> \}

In RT, the consumers associated to a subscription \(S\) mean the local consumers who issue subscription \(S\) (i.e., RT.\(S.C_{ID}\)). Incoming servers of subscription \(S\) (i.e., RT.\(S.JS\)) are the event servers from which subscription \(S\) is sent, and outgoing servers of subscription \(S\) (i.e., RT.\(S.OS\)) are the event servers to which \(S\) is forwarded to.

Registration of subscriptions by each consumer at every event server leads to explosion of information as well as traffic. This problem is managed by using subscriptions *covering method* as described in Section 2.4.1. In covering method, when an event server receives a new subscription \(S_{\text{new}}\), it checks whether \(S_{\text{new}}\) is covered by previous forwarded subscriptions \(S_{\text{old}}\) (i.e., \(S_{\text{new}} \subseteq S_{\text{old}}\)) from its routing tale. The detail of covering method can be found in [13, 14].

As and when a mobile user ceases interest to an event, it can unsubscribe to stop receiving the event. For unsubscribing, the relevant subscription is not only removed from the routing table of local event server, but also removed from the outgoing servers. For example on receiving unsubscription \(S_u\) from a mobile consumer, an event server deletes \(S_u\) and the mobile consumer's subscriptions covered by \(S_u\) from routing table. Then \(S_u\) is forwarded to some neighboring event servers (i.e., RT.\(S_u.OS\)) for unsubscribing.

If a notification \(N\) (an event) is received from a producer, an event server executes *filtering* (matching) process. In filtering process, the event server matches (filters) \(N\) with
all subscriptions stored in RT. A notification $N$ matches a subscription $S$ if $S$ covers $N$ ($N \subseteq S$). If RT has a subscription $S$ matched with $N$ (i.e., $\exists S \in RT \mid N \subseteq S$), the results of filtering process are:

1. a set of local consumers ($LC$) residing in the area covered by local event server (i.e., $LC \in RT.S.C.I.D$) and
2. a set of neighboring event servers ($R_{ES}$) to which $N$ needs to be forwarded (i.e., $R_{ES} \in RT.S.IS$).

Then, the event server delivers $N$ to $LC$ and forwards $N$ to $R_{ES}$.

### 3.5.2 Subscription Manager

Subscription manager consists of two parts: Subscription Queue of mobile consumers and Covering process. In first case, when subscription manager receives a subscription $S$ from a local mobile consumer, it stores $S$ in Subscription Queue with identity of MH (i.e., $<\text{id}_{\text{MH}}><S>$) and then $S$ is sent to Covering process. In Covering, $S$ is stored in routing
table (RT) and covering method is executed to forward $S$ to neighboring event servers. The
detail of covering method is described in Section 3.5.1. During handoff of the mobile
consumer, its subscriptions stored in Subscription Queue are transferred to its new
location. After handoff process, MH and its subscriptions are deleted from the list of
subscription queue. In case of receiving a subscription $S$ from neighboring event servers,
subscriptions received from neighboring event servers are not required for handoff. Hence,
it is not necessary to store $S$ in subscription queue and the covering process is executed
directly. In both cases, subscription manager sends the subscription $S$ to communication
manager. The functionality of communication manager is described in Section 3.5.4.

Subscription manager can receive an unsubscription $S_u$ from a local mobile consumer
or a neighboring event server. On receiving unsubscription $S_u$ from a mobile consumer, $S_u$
and the mobile consumer's subscriptions covered by $S_u$ are deleted from Subscription
Queue and is sent to covering process. Covering process also deletes $S_u$ and the mobile
consumer's subscriptions covered by $S_u$ from routing table. Then it sends $S_u$ to
communication manager to forward it to neighboring event servers. In case of receiving $S_u$
from neighboring event servers, if $S_u$ is not related with Subscription Queue, then it is
deleted from routing table.

3.5.3 Notification Manager

The notification manager can receive events/notifications from producers and neighboring
event servers. On receiving a notification $N$, notification manager executes Filtering
process to match $N$ with all subscriptions in routing table (RT). The detail of filtering is
described in Section 3.5.1. Filtering process stores matched notifications that are to be
forwarded to neighboring event servers, in Notification Buffer. Notification Buffer stores
these notifications temporarily until it receives acknowledgements of receipt of notices
from neighbors. Notification manager uses Notification Queue to store matched
notifications of local mobile consumers with the identities of the mobile consumers (i.e.,
$id_MH$ $N$). Notification manager sends matched notifications to communication
manager to deliver/forward them to interested consumers/event servers. If a mobile
consumer is disconnected or unreachable, Notification Queue stores notifications for a
certain period to deliver to the consumer later. When the mobile consumer reconnects to
the current event server, notifications of the mobile consumer are delivered. If a mobile
consumer connects to a new event server, the related notifications are transferred to the
new event server. Notifications which are stored for a mobile consumer MHs in
Notification Queue are deleted in the following three cases:

1. The notifications have been delivered to the MHs. (i.e., communication manager
have executed “Deliver_N” process.)
2. The notifications have been transferred to a new event server of MH during
handoff.
3. The deadline (expiry time) of the notifications elapsed. Even if storage of
notifications at an event server is not a concern, the sheer amount of time and
precious bandwidth required to transmit all of the queued notifications to the
consumer when it reconnects might be unreasonable. Moreover, many time-
sensitive events may become useless when the consumer reconnects to the service
(For example, events of whether/traffic information and stock price).

3.5.4 Communication Manager

Communication manager receives subscriptions/unsubscriptions from subscription
manager and notifications from notification manager. Communication manager executes
“Forward_N” to forward a notification N to neighboring event servers. It executes
“Deliver_N” to deliver a notification N to a local mobile consumer. On receiving
subscription S “Forward_S” is executed to forward S to neighboring event servers. If
communication manager receives unsubscription Su, it executes “Cancel_S” to inform
neighboring event servers to cancel subscription S covered by Su (i.e., to unsubscribe S).

3.6 Summary

This chapter concentrates on the aspect of how to conceptually model an event notification
service system for mobile computing environment. First the typical requirements in
designing of event notification service systems are identified. Additional requirements which are dependent on different application domains and environments are also described. Then the mobile computing environment and its limitation, and possible designs of event notification service system are presented. Designs of event notification service system are different on wireless network (with base station) and ad-hoc network (without base station) as the characteristics of these environments are different. After that, a conceptual architecture of wireless event notification service system is presented and it focuses on the wireless network with base stations. As wireless network with base stations is widely used in practice (e.g., GSM cellular network), it is selected to design event notification service on it. This basic architecture will be used in the following chapters to study the research problems dealt in this thesis.